C. Carrez

Désignation dans les systèmes répartis

1. Principes généraux

Les noms

nom = information permettant d'identifier un objet
adresse = information permettant d'identifier un emplacement
    adresse mémoire d'exécution
    adresse mémoire de stockage

liaison = mécanisme qui associe un nom et un objet
interprétation d'un nom = mécanisme permettant de passer du nom à l'objet.

La protection

mécanismes pour empêcher les accès non autorisés aux objets.

- Qui est le demandeur?
- Qu'a-t-il le droit de faire?
- Puis-je faire confiance à l'exécutant?

Système distribué

changement d'échelle pour le nombre d'objets et d'acteurs
communication au moyen d'un intermédiaire "douteux"

2. La désignation

2.1. Le nommage en général

Contexte d’interprêtation des noms

L’interprêtation d’un nom dépend d'un contexte plus ou moins étendu.

- Nom local: n’a de sens que dans le contexte d’un processus
- Nom global: indépendant de tout contexte

En conséquence, lorsque l’on transmet le nom d’un objet, il faut :

- soit être certain que le contexte est connu du récepteur
- soit transmettre la partie du contexte qui permet de l’interprêter
- soit transmettre un nom global (après transformation)

Les 2 niveaux de noms

On peut distinguer 2 niveaux dans le système de nommage:

- Nom externe (ou symbolique) = chaîne de caractères
- Nom interne = nombre binaire
- Gestion des noms assure la mise en correspondance entre les deux niveaux

Notion de nom unique

problèmes:

1. deux objets différents doivent avoir des noms globaux différents
2. quand un objet est détruit, ne pas avoir à rechercher toutes les références à cet objet
3. pouvoir identifier un objet dans tout le système, indépendamment de sa localisation

Idée: nom est un numéro attaché à l'objet pendant toute son existence, et qui n'est pas réutilisé pour un autre objet

Information portée par un nom

Nom “pur” = configuration de bit sans signification
autres noms = configuration de bit qui porte une information

information permettant de déduire en partie la localisation

-> déplacements limités de l’objet sans changer son nom

(mach= 12, vol=43, inœud=52) localise complètement le fichier, mais interdit son déplacement

information sur la structuration des objets

-> modifications limitées de la structure sans changer son nom

/A/B/C laisse supposer l’existence d’un répertoire A qui contient un sous répertoire B

2.2. Le nommage dans les systèmes distribués

La désignation est plus complexe qu'en centralisé:

- liaisons plus complexes
  - liaison machines vers adresses réseaux
  - liaison serveurs vers machines
- il faut conserver l'aspect dynamique de la liaison
  - migration d'objets et de serveurs
  - duplication d'objets et de serveurs

Interprêtation des noms

- Répertoires répartis et dupliqués (robustesse et facilité d'accès)
- Quel répertoire choisir? => accompagner les noms de suggestions de localisation

Problème de cohérence

Cohérence immédiate des copies multiples

toujours la valeur la plus récente => accéssibilité limitée

accéssibilité maximale => pas forcément la plus récente

- les noms changent rarement => incohérences rares
- un nom obsolète a un effet visible
- si on ne s'apercoit pas qu'un nom est obsolète, c'est sans importance.

Attention: pas toujours vérifié!

Cohérence à long terme des copies multiples

Si on arrête de modifier les données, toutes les copies deviendront identiques.

Facteur d'échelle:

pas de limite sur le nombre de machines ou le nombre d'objets,
grand nombre d'administrateurs plus ou moins indépendants

2.3. Transformation nom externe vers nom interne

Principe général: structure hiérarchique

- structure classique bien connue
- permet la prise en compte d'un grand nombre d'objets
- une structure graphique est possible, mais nécessite un ramasse-miettes réparti

Vision unique, ou vision dépendante de la machine

- le nom /A/B/C désignant un objet O sur une machine, est-il encore valable sur une autre machine pour désigner O
- les systèmes à montage à distance (NFS) sont dépendants

Transparence des noms

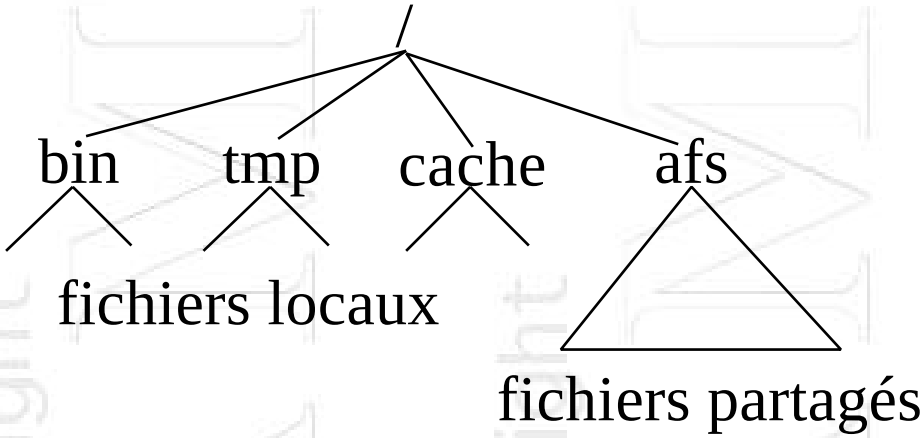
- transparence de localisation: le nom ne dit rien sur la localisation de l'objet
  /serveur/A/B est transparent à la localisation, car le serveur peut être déplacé
  /machine/A/B ne l'est pas
- indépendance de localisation: l'objet peut être déplacé n'importe où sans changer de nom

serveurs de nom spécialisés:

- assurent uniquement la traduction
  nom externe -> nom interne
- peuvent faire correspondre plusieurs noms internes à un même nom externe
  copies multiples du même objet

Andrew File System

Distinction fichiers locaux et fichiers partagés



Les fichiers partagés

- structurés en une arborescence unique
- partitionnés sur des volumes, un volume contenant une partie de l'arborescence.
- chaque fichier ou répertoire est identifié par un FID

32 bits	32 bits	32 bits
num volume	num Vnode	num unique

numéro unique permet la réutilisation du Vnode

- gèrent des volumes complets
- ne s'occupent pas des chemins d'accès, mais répondent aux FID
- sont responsables de la sécurité et de la protection
- partagent une base de données dupliquée

  num volume -> serveur gérant

  note: petite taille, peu de changements
- migration de volume par recopie, et mise à jour de la base
- existence de copies à lecture seule de volumes, indiquées dans la base

- font l'analyse du chemin d'accès
- cachent localement fichiers et répertoires
- cachent les associations

  <préfixe de nom> -> FID
- les serveurs sont responsables de la validité des caches

  le serveur notifie le client avant d'autoriser une modification par un autre (*callback*)
- vision unique car les montages de volume sont gérés par les serveurs et non par les clients

Système SOS

Arborescence unique partitionné verticalement entre serveurs
problèmes:

- maintient de la cohérence entre les partitions de l'arbre
- le client doit trouver le serveur qui gère l'objet

utilisation de caches de préfixes

- pour chaque préfixe -> liste des serveurs concernés

  exemple:

/A	1, 2
/A/D	1
/B	2, 3

- en recherche, s'adrésser à tous les serveurs du plus long préfixe

2.4. Localisation physique à partir du nom global

Par diffusion

- le nom global est envoyé à tous les sites, celui qui possède l'objet répond

Suivant une heuristique

- utilisation d'une suite de sites particuliers avant de faire une diffusion globale

Evaluation itérative

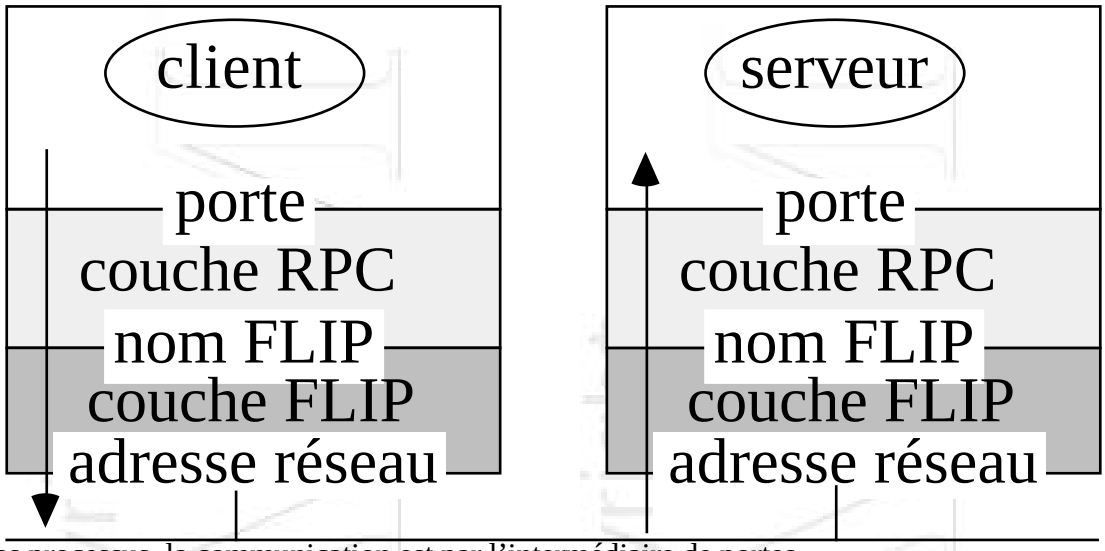
- les noms sont couplés à des hypothèses de localisation

Serveurs de localisation

- chaque site connaît au moins un de ces serveurs
- ils sont en général dupliqués pour permettre la tolérance aux pannes

Note: en général, des caches de localisations évitent de redemander

Principe d'Amoeba



Vue des processus, la communication est par l'intermédiaire de portes
Une porte est un nom global pur, identifiant un service
Tout processus reçoit, à la création, un "nom FLIP" (Fast Local Internet Protocol)

un nombre aléatoire de 64 bits.

ne change pas durant toute sa vie, même s'il migre

Coté client, le noyau dispose de tables qui associent les serveurs aux services:

porte -> nom FLIP

Tout message sur le réseau comporte

- le nom FLIP du processus émetteur
- l'adresse réseau du site qui l'héberge.

L'arrivée d'un message sur un site entraîne la mise à jour de la table.

nom FLIP -> adresse réseau

A son lancement, un serveur se déclare à son noyau en donnant la porte du service.

Lorsqu'un client fait une demande sur une porte,

1. La couche RPC recherche dans ses tables le nom FLIP
2. Si elle ne trouve pas, elle diffuse une demande de localisation.
3. Les sites sur lesquels un serveur s'est déclaré répondent.
4. Le site met à jour ses tables et choisit éventuellement le serveur.

Si un processus migre du site A vers le site B,

- toute demande de service arrivant sur A sera retournée avec l'indication de migration,
- le demandeur peut chercher la nouvelle localisation du processus par son nom FLIP
- il peut chercher une nouveau serveur.

La désignation dans les systèmes d'exploitation répartis, José Legatheaux Martins et Yolande Berbers, TSI vol 7 N° 4, 1988, pp 359-372.
Distributed Systems, 2nd edition, Editeur Sape Mullender, ACM Press, 1993
Systèmes d'exploitation systèmes centralisés et systèmes distribués, Andrew Tanenbaum, InterEditions, 1994
Le nommage de ressources réparties sur de grandes étendues géographiques, Jean-Pierre Le Narzul, Thèse Paris Sud, 1993.
Authentication in distributed systems: Theory and Practice, Buther Lampson, Martin Abadi, Michael Burrows, Edward Wobber, ACM Transactions on Computer Systems, vol 10, N°4, 1992, pp 265-310
Prudent Engineering practice for cryptographic protocols, Martin Abadi, Roger Needham, SRC-research report 125, DEC, 1994

La protection dans les Systèmes Répartis

C. Carrez, E. Gressier

1. La protection

1.1. problème général

Ce qu'on attend du système:

secret: pouvoir conserver des données secrètes

confidentialité: données fournies dans un but ne sont utilisées que dans ce but

authenticité: données indiquées comme venant de X viennent bien de lui

intégrité: données mémorisées par le systèmes ne peuvent pas être altérées

Ce que peut tenter un intrus:

inspection: lecture des fichiers, espionnage des paquets

fuite: obtenir d'un complice des données confidentielles

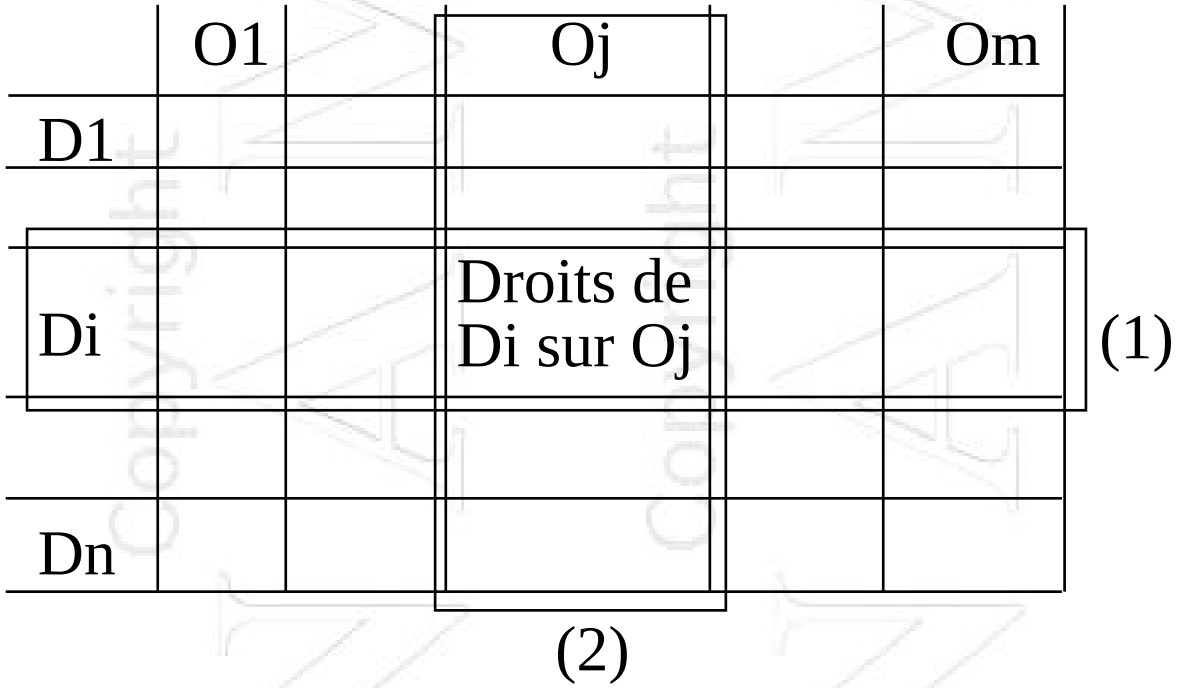
déduction: reconstruire un information sensible depuis un ensemble d'informations

déguisement: se faire passer pour un utilisateur autorisé.

1.2. Domaine et matrice d'accès

domaine = ensemble d'objets accessibles, ainsi que les opérations (droits) autorisées sur ces objets
un processus s'exécute dans un domaine, mais il peut en changer.
Les changements de domaine sont contrôlés

matrice de droit



(1) liste de droits, ou encore liste de capacité

(2) liste de contrôle d'accès

1.3. Où ranger les droits?

Une liste de contrôle d'accès, pour chaque objet

- près de l'objet, donc dans le serveur qui le gère (solution retenue pour AFS)
- problème: authentification du demandeur, et du domaine dans lequel il s'exécute

Une liste de capacité, pour chaque domaine

- près du processus, donc chez le client
- problème: falsification de la capacité par le client, ou par le réseau

Exemple d'Amoeba

Les portes

une porte est un couple de deux nombres assez grands, 64 bits, (P, G), reliés par une fonction non inversible F, connue de tous, de telle sorte que P = F(G)

G est privé, et n'est connu que du serveur qui reçoit les messages sur cette porte "get (G)"

P est connu de ceux qui peuvent envoyer des messages sur cette porte "put(P)"

la connaissance d'une porte est synonyme du droit d'envoyer des messages sur la porte

La protection suppose qu'il est impossible de contourner la transformation P = F(G):

- un intrus ne peut s'approprier un message envoyé sur la porte, car il ne connaît pas G
- le serveur répond par le biais d'une porte du client; un intrus ne peut s'approprier la réponse

les objets sont protégés par des capacités

Une capacité a 4 champs:

64 bits	32 bits	32 bits	64 bits
porte	objet	droits	contrôle

la porte est celle du serveur qui gère l'objet

le champs objet n'a de sens que pour le serveur; il identifie l'objet concerné

le champs droit contient 1 bit par opération

le champs contrôle permet au serveur de vérifier la validité de la capacité:

- à la création de l'objet, le serveur tire un nombre aléatoire N, qu'il conserve dans ses tables
- le champs contrôle = F(N xor droits), où F est une fonction non inversible.

Note: la révocation (pour tous) est simple: il suffit de tirer un nouveau nombre aléatoire N!

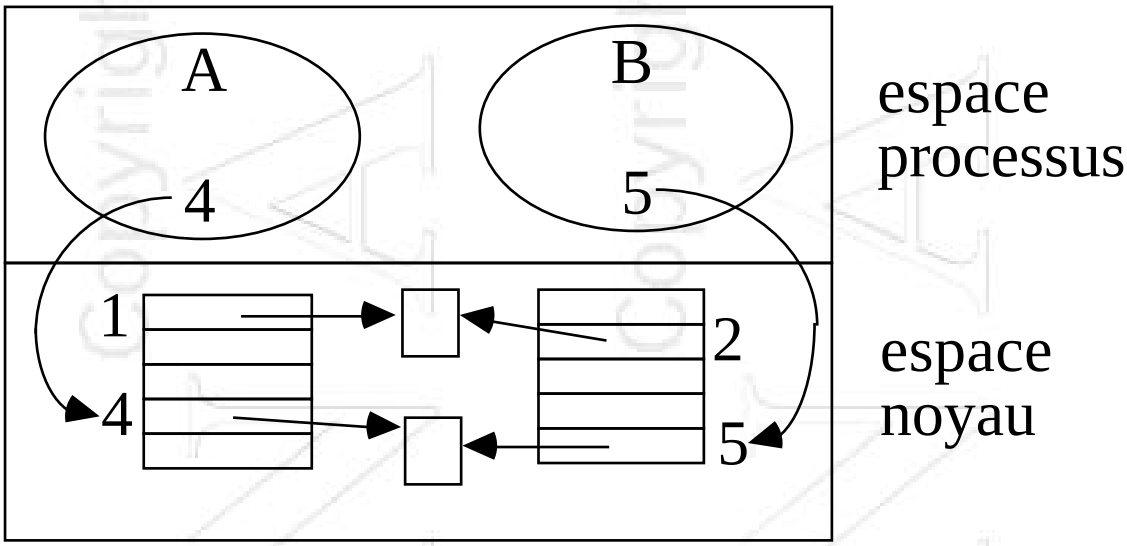
Exemple de Mach

les portes sont protégées par des capacités

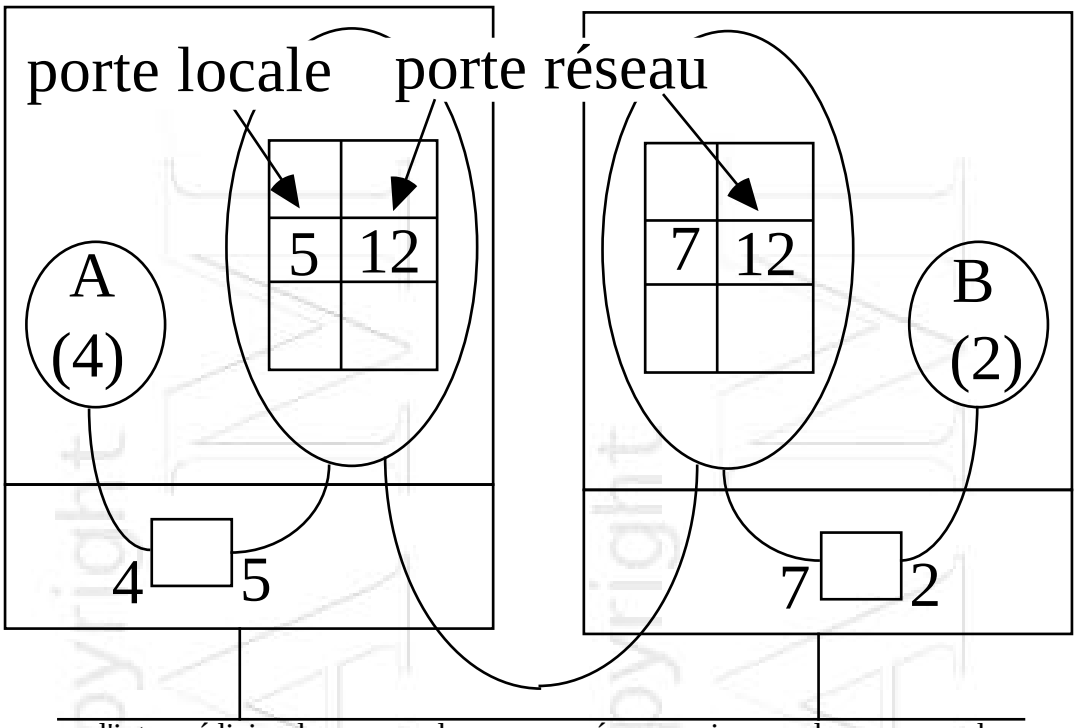
les capacités ne sont pas accessibles aux processus, qui les désignent par un nom local

droits: recevoir, envoyer, envoyer-une-fois

sur un site:



Entre sites:



On passe par l'intermédiaire du serveur de message réseau, qui assure la correspondance

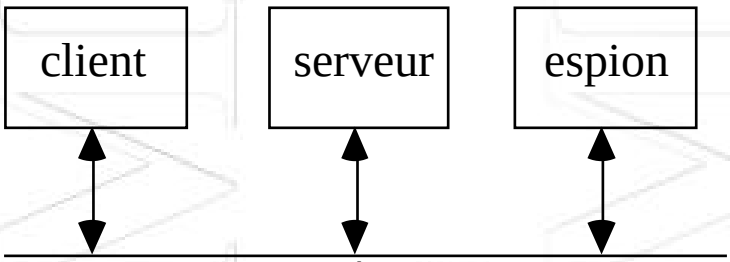
porte mandataire<->porte réseau

le transfert d’une capacité entre processus fonctionne ainsi:

- formatage du message pour indiquer qu’un emplacement du message est un nom local de capacité
- émission du message; le système remplace le nom local par la capacité...
- réception du message; le système range la capacité dans l’espace protégé, lui attribue un nom local et la remplace dans le message par ce nom

les messages sur le réseau peuvent être cryptés si nécessaire

2. Notion de sécurité



L’espion peut se faire passer pour le client auprès du serveur,
L’espion peut se faire passer pour le serveur auprès du client,
L’espion peut observer l’échange client-serveur
Nécessité d’authentification mutuelle et de conservation du secret

2.1. Cryptage par clé privée

la même clé est utilisée pour le cryptage et le décryptage

Message chiffré = cryptage(Message, Clé)

Message = décryptage(Message chiffré, Clé)

On note parfois $\{M\}_K$ le message M crypté par la clé K

Exemple DES (Data Encryption Standard)

utilisation de clés de 56 bits pour coder des blocs de 64 bits

fonction de cryptage/décryptage simple, efficace, peut être cablée,

pratiquement impossible de trouver la clé

2.2. Cryptage par clé publique clé privée

cryptage et décryptage utilisent des clés différentes

Message chiffré = cryptage (Message, Clé1)

Message=décryptage(Message chiffré,Clé2)

Clé-1 privée => qqun qui connaît Clé-2 trouve M

Clé-2 publique => seul celui qui a crypté connaît Clé-1

On notera $\{M\}_K$ le message M crypté par la clé K.

On note souvent K, la clé privée et K^{-1} la clé publique.

Exemple RSA (Rivest,Shamir,Adleman)

- basé sur la difficulté de décomposer les grands nombres (150 à 300 chiffres) en facteurs premiers
- fonction de cryptage/décryptage longues et coûteuses
- connaissant l'une des clés, il est pratiquement impossible de trouver l'autre
- les deux clés fonctionnent dans n'importe quel ordre

2.3. Hypothèses générales

1 le cryptage est parfait: décryptage impossible sans connaissance de la clé

2 il est possible de savoir si un message a été décrypté avec la bonne clé
Ce n'est pas le cas si le message est une suite de bits non prévisible.

3 $\{M\}_K = \{M'\}_{K'}$ => M = M' et K = K'

4 si un message est décrypté avec la bonne clé, alors quelqu'un l'a crypté avec la clé ad-hoc

Remarques :

si est K_A la clé privée connue de A, et K_A^{-1} la clé publique

1 $\{M\}_{K_A}$ vient de A, qui connaît M, et est compréhensible par tous (signature),

2 $\{M\}_{K_A^{-1}}$ est crypté par n'importe qui, mais n'est compréhensible que par A,

3 $\{\{M\}_{K_A}\}_{K_B^{-1}}$ vient de A, qui connaît M, et n'est compris que par B.

4 $\{\{M\}_{K_B^{-1}}\}_{K_A}$ vient de A et n'est compris que par B. Rien ne dit que A connaît M: le cryptage de M a pu être fait par n'importe qui!

Principe général: ne jamais crypter quelque chose que l'on ne connaît pas.

2.4. Serveur d’authentification

Il est impossible que tout couple de communiquants potentiels se partagent statiquement une clé secrète, car il y en a trop!
L’utilisation systématique de clés publiques est trop coûteuse (cryptage est 1000 à 5000 fois plus long).
Solution: si deux processus veulent partager une clé secrète, ils font appel à un serveur:

1 Ils ont tous les deux confiance en lui.

2 Le serveur garantit à chacun l’identité de l’autre, ce qui implique qu’il est sûr de l’identité de chacun!

3 Le serveur choisit la clé secrète, et garantit qu’ils sont les seuls à la connaître.

Le protocole entre processus et serveur se fait initialement par le biais de clés publiques:

1 je suis certain que le serveur a fourni la clé, car elle est cryptée par une clé secrète qu’il est seul à connaître, et dont je suis seul à connaître la clé de décryptage
2 je suis certain que cette clé a été fournie récemment car on utilise un mécanisme d’estampilles pour garantir la fraicheur

protocole d'authentification à clé privé avec serveur de Needham et Schroeder :

1. A -> S : N_A,A,B

2. S -> A : $\{N_A,B,K_{AB},\{K_{AB},A\}_{K_B}\}_{K_A}$

3. A -> B : $\{K_{AB},A\}_{K_B}$

4. B -> A : $\{N_B\}_{K_{AB}}$

5. A -> B : $\{N_B - 1\}_{K_{AB}}$

protocole d'authentification à clé publique avec serveur de Needham et Schroeder :

Seul le serveur a connaissance des clefs publiques de A et B (K_A^{-1} et K_B^{-1}), les sites n'ont que la clef publique du serveur :

1. A -> S : A,B
2. S -> A : $\{B,K_B^{-1}\}_{K_S}$
3. A -> B : $\{N_A,A\}_{K_B^{-1}}$
4. B -> S : B,A
5. S -> B : $\{A,K_A^{-1}\}_{K_S}$
6. B -> A : $\{ N_A , N_B \}_{K_A^{-1}}$
7. A -> B : $\{N_B\}_{K_B^{-1}}$

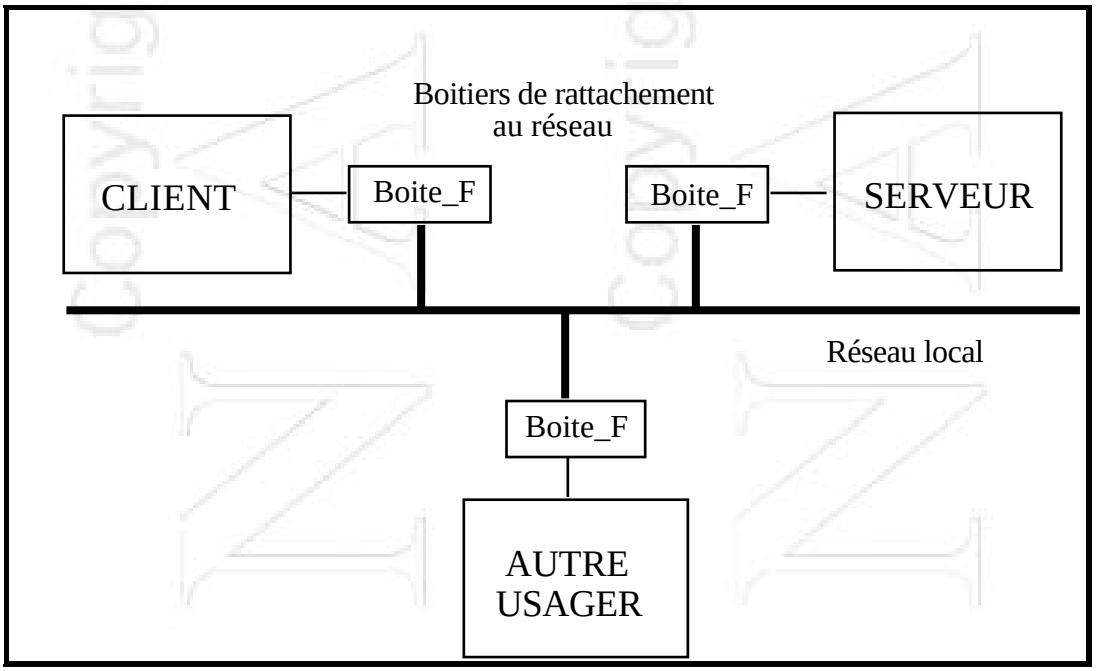
Le protocole n'est pas suffisant :

- l'étape 3 peut être un replay, il faut ajouter une estampille

- on peut se faire passer pour A dès qu'on s'empare de sa clef publique, le protocole suppose qu'aucun attaquant s'est procuré une clef publique, en particulier celle du serveur

Exercice: Etude du système d'authentification AMOEBA (G. Florin)

Ce problème s'intéresse à l'authentification des entités communicantes dans une relation client-serveur du système AMOEBA. On ne considère pas les autres problèmes de confidentialité, d'intégrité, ... Ce système réparti suppose que tous les utilisateurs du réseau de communication d'entreprise ne peuvent communiquer qu'au moyen d'un boitier matériel dédié aux fonctions de sécurité appelé dans la suite boite-F ("F-box").



Dans ce système les adresses de ports ou point d'accès de service de communication (adresses de communication) sont définies sur un grand nombre de bits (ici 48 bits) et tirés aléatoirement dans cet espace.

1 Ce simple choix est déjà une technique de protection contre certaines erreurs ou attaques. Lesquelles? Pourquoi? (répondez en quelques phrases)

Correction : Le fait d'utiliser des adresses tirées aléatoirement dans un grand espace de numérotation fait perdre la

correspondance entre l'adresse et le rôle d'un service. Il est donc plus difficile de déterminer une adresse pour un service donné dont on voudrait usurperait l'accès.

D'une part lorsqu'on est un utilisateur bien intentioné mais qui fait des erreurs de programmation et génère des adresses fausses ("par hasard") on peut difficilement (avec une probabilité très faible) tomber sur une adresse correcte et mettre en danger le bon fonctionnement du destinataire.

D'autre part si l'on est un utilisateur mal intentionné le grand nombre d'essais et d'erreurs qu'il faut effectuer dans des attaques "force brute" interdit pratiquement cette forme de piratage. I

Il est bien certain que ce choix et l'ensemble de remarques precedente ne conduit pas à un niveau de sécurité très élevé mais la technique est d'un coût très faible.

On utilise pour les fonctions de sécurité une technique de cryptage au moyen d'une fonction univoque F (fonction à sens unique, "one way function"). Par exemple dans la suite une adresse de port de communication aura en fait deux versions, une version en clair G (connue seulement de l'usager et de la boite_F) et une version cryptée P = F(G) (version qui circule sur le réseau).

2 Rappelez la définition d'une fonction univoque

Correction : Une fonction à sens unique F est une fonction telle que son calcul est facile mais qu'il est très difficile voire impossible de déterminer la fonction inverse F^{-1} lorsque l'on connait F. C'est l'un des éléments essentiels d'un système de mot de passe. On peut utiliser par exemple l'algorithme du DES fonctionnant avec pour clé la valeur X sur laquelle s'applique la fonction. On peut déterminer un couple clé publique et clé privée dans l'akgorithme RSA et détruire l'une des clés.

Les boites F ont pour rôle principal de calculer la fonction F selon le mode de fonctionnement suivant. Soit Gs une adresse choisie aléatoirement par un serveur S. Le serveur garde Gs secret et communique Ps=F(Gs) aux clients à qui il souhaite offrir son service. Un client posséde également une adresse Gc (avec Pc=F(Gc)). Il présente à sa boite_F locale pour l'émission une requète de la forme:

put (Ps, Gc, Kc, Dappel)

- Ps est l'adresse de port cryptée du destinataire

- Gc est l'adresse de port de l'émetteur (pour la réponse)

- Kc est un mot de passe (une clé secrète) authentifiant du client.

- Dappel est l'ensemble des données constituant la requête.

La boite_F du client effectue en fait une primitive notée put_F (pour la distinguer de celle du client)

put _F(Ps, Pc, F(Kc), Dappel)

Elle émet sur le réseau un message comportant une adresse client cryptée, une adresse serveur cryptée et la clé du client cryptée.

Le serveur demande la réception des messages à sa boite_F locale en lui présentant une requête:

get (Gs, X, Y, Dappel)

La boite_F du serveur calcule la clé cryptée du serveur Ps=F(Gs) et n'accepte que les messages qui comportent l'adresse cryptée Ps. Le serveur peut alors vérifier la requête du client et récupérer l'adresse X=Pc du client.
3 Comment le serveur peut-il vérifier l'identité du client?

Correction : En fait le serveur dispose de deux moyens de vérification: les adresses de ports et la clé (mot de passe) qui sont relativement redondants pour la fonction d'authentification souhaitée. Ces deux informations circulent cryptées. Le serveur peut donc disposer d'un fichier de mots de passe (ou de nom de ports) cryptés comme dans tout système d'authentification à l'ouverture d'une session en temps partagé. Comme les mots de passe cryptés ont été fabriqués par une boite_F en qui on peut avoir confiance il suffit de faire une comparaison entre la valeur présentée et la valeur stockée.

Le serveur répond ensuite au client par:
put(X=Pc,Gs,Ks,Dreponse)

La boite_F du serveur retransmet en fait sur le réseau:
put(Pc,Ps,F(Ks),Dreponse)

4 Expliquez pourquoi ce mécanisme interdit à un autre usager que le serveur d'usurper l'identité du serveur S. De la même façon le client est le seul à pouvoir recevoir la réponse. Pourquoi?

Correction : Pour arriver à se faire passer pour le serveur du point de vue de sa boite_F un utilisateur du système n'a pour seule solution que de connaitre Gs qui normalement est une information connue uniquement du vrai serveur S. La seule information qui circule sur le réseau est F(Gs). Si un intrus présente cette valeur à sa boite_F d'interface celle ci utilisera F(F(Gs)) qui n'est d'aucune utilité. La fonction F étant univoque il n'est pas possible de déduire Gs de F(Gs).

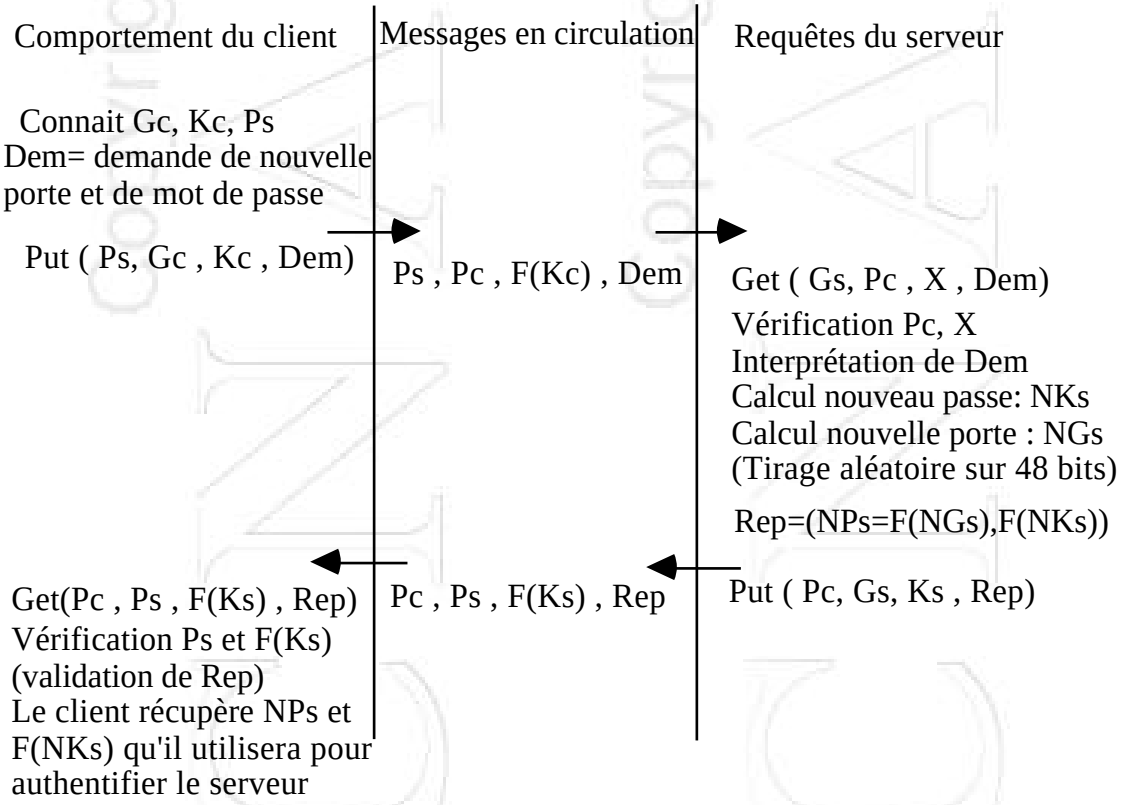
On souhaite maintenant augmenter la sécurité du système en introduisant une obligation de changement périodique des informations de sécurité employées. On s'intéresse donc à la procédure de changement de l'adresse de port de communication et du mot de passe du serveur.

5 Définissez l'échange permettant le changement sécurisé de l'adresse du serveur et de mot de passe du serveur (le client demande et récupère une nouvelle adresse de serveur cryptée NPs et un nouveau mot de passe du serveur).

Correction : Un aspect important du changement d'adresse et de mot de passe est que celui-ci doit se faire sous les anciennes protection. Le client doit pouvoir authentifier le serveur pour savoir s'il ne s'agit pas d'un serveur déguisé qui répond. Le changement doit prendre effet des la fin d'un échange de changement ce qui signifie qu'il ne peut y avoir de

panne (perte de message) pendant cette phase sous peine de ne plus pouvoir communiquer.

Le serveur doit calculer une nouvelle clé et une nouvelle adresse de port. Il doit les communiquer au client par message mais uniquement sous leur forme cryptée car le client ne doit pas pouvoir y accéder en clair. La boite_F ne peut faire ce travail. Elle ne sait le faire que sur les zones adresses et clés du message. On ne peut avoir mis dans ces zones directement les nouvelles adresses et clés car on ne pourrait pas vérifier l'origine de cette modification.



6 On suppose qu'un intrus C' a pu s'approprier illégalement un accès au serveur S (par un moyen non décrit, apprentissage des adresses et clé d'accès au serveur, usurpation d'identité d'un client C accrédité). Montrez comment le changement d'adresse et de mot de passe du serveur permet de détecter cette situation.

Correction : La situation d'usurpation décrite implique qu'un client accrédité était en fonctionnement correct (connexion correctement ouverte, procédure de changement de secret déjà opérante) et que suite à une usurpation deux clients ont simultanément accès à un serveur S (le vrai et le faux). La procédure avec changement implique que périodiquement les adresses de ports et les mots de passe sont changés. Les échanges sont point à point et lors d'un changement un seul site est demandeur du changement et est avisé de ce changement (le vrai ou le faux peu importe). Ensuite l'un des deux continue sur les anciennes valeurs d'accès et le serveur a la possibilité de détecter une anomalie pour faire cesser tous les transferts immédiatement.

7 La sécurité de cette solution repose essentiellement sur l'usage impératif et unique des boite_F par tous les sites. Si un intrus arrive à se brancher directement sur le réseau que doit-il faire pour utiliser illégalement un serveur? (2,5 points)

Correction : Un intrus peut en écoutant le réseau obtenir la totalité des informations qui circulent dans des messages corrects. En particulier les adresses cryptées des clients et des serveurs les clés cryptées des serveurs.

Il peut donc fabriquer sans difficultés des messages qui sont corrects (la technique est en fait pratiquement celle de reémission"replay") sur la partie contrôle des messages).

Le problème posé par les adresses (sur un grand espace d'adressage déjà souligné) est qu'il faut que l'intrus découvre parmi tout le traffic des messages qui ont une structure correspondante à celle du serveur qu'il veut atteindre. Il lui faut donc une connaissance du protocole du serveur pour lui permettre de reconnaitre en filtrant le traffic des messages correspondant au serveur qu'il veut atteindre.

Par la suite lorsqu'il a trouvé le bon serveur il doit également s'adapter en permanence aux changements d'adresse.

Notons enfin qu'il ne peut pas faire émettre ses messages par une boite_F dans la mesure ou il en aurait une aussi car il ne connait pas les clés secrètes des usagers.

Bibliographie

La désignation dans les systèmes d'exploitation répartis, José Legatheaux Martins et Yolande Berbers, TSI vol 7 N° 4, 1988, pp 359-372.
Distributed Systems, 2nd edition, Editeur Sape Mullender, ACM Press, 1993
Systèmes d'exploitation systèmes centralisés et systèmes distribués, Andrew Tanenbaum, InterEditions, 1994
Le nommage de ressources réparties sur de grandes étendues géographiques, Jean-Pierre Le Narzul, Thèse Paris Sud, 1993.
Authentication in distributed systems: Theory and Practice, Buther Lampson, Martin Abadi, Michael Burrows, Edward Wobber, ACM Transactions on Computer Systems, vol 10, N°4, 1992, pp 265-310
Prudent Engineering practice for cryptographic protocols, Martin Abadi, Roger Needham, SRC-research report 125, DEC, 1994
Désignation et protection dans les systèmes répartis. C. Carrez. Cours du Tronc Commun du DEA Système P6-Cnam-ENST.