Etablissement, gestion et certification des clés

A. Bonnecaze

Gestion des clés

Ensemble de processus comprenant

- La génération de clés
- La distribution de clés (secrète ou privée)
- Le stockage et l'archivage
- Remplacement/destruction de clés

Quelle clé pour quel algorithme?

Objectif crypto	Clef publique	Clef symétrique
Confidentialité	Chiffrement	Chiffrement
Authentification Origine des données	Signature	MAC
Distribution de clef	Diffie-Hellman	Différentes méthodes
Authentification Des entités challenge-response	- Signature - Déchiffrement - Custom	- MAC - Chiffrement

Génération de clefs

- Sources physiques : pas pratique
- Générateur de bit pseudo-aléatoires (PRBG)
  - LFSR
  - À base de RSA
  - À base de fonction de hachage
  - À base de DES

PRBG

- Basé sur une fonction de hachage
  - germe s, $z_n = h(s+n \bmod p)$ $n = 0,1,2\ldots$
- Basé sur DES
  - $K' = DES_K$(temps universel)
  - $R_n = DES_{K'}(n)$ $n = 0,1,2,\ldots$
  - Sécurité basée sur la clef secrète K
- Ce sont des générateurs utilisés dans la pratique

Architecture des clefs

- Hiérarchie des clefs
  - **Master keys** : elles ne sont pas protégées cryptographiquement. Distribuées manuellement et protégées physiquement (isolation électronique, porte blindée,…)
  - **Key-encrypting keys** : symétriques ou publiques, elles sont utilisées pour le transport ou le stockage d'autres clefs
  - **Data keys** : utilisées par l'utilisateur pour chiffrer et authentifier des données.

Objectifs cryptographiques

- Protocoles d'authentification
  - Fournir à une partie une assurance sur l'identité d'une autre partie (avec laquelle elle communique)
- Protocole d'établissement de clef
  - Établir un secret partagé entre les parties
- Protocole d'établissement de clef authentifiée
  - Établir un secret partagé entre parties dont les identités ont été ou peuvent être vérifiées
  - On dit aussi *distribution de clef authentifiée*

Distribution de clefs

- Distribution de clefs secrètes
  - La distribution doit assurer la confidentialité des clefs
  - Le service de distribution (s'il existe) doit être on-line
- Distribution de clefs publiques
  - La distribution doit assurer l'intégrité des clefs
  - Le service de distribution peut être off-line

Utilisation de clefs de session

- Une clef de session est éphémère. Par exemple le temps d'une connexion. Pourquoi?
  - Pour limiter les chiffrés qui peuvent faire l'objet d'une attaque
  - Pour limiter l'exposition en matière de période de temps et de quantité de données dans le cas de compromission de la clef
  - Eviter le stockage à long terme de clef : on utilise une clef nouvelle pour chaque besoin
  - Créer une indépendance entre sessions/applications

Distribution de clefs

- Si Alice et Bob veulent communiquer, comment vont-ils se mettre d'accord sur une clef secrète de session ?
- hypothèse : canal non sûr
- Bien sûr

  Alice et Bob pourraient se rencontrer en personne

  Alice pourrait envoyer la clef à Bob par courier sûr

Ces solutions ne sont pas acceptables dans le monde Internet

Distribution de clef sans serveur

- A et B partage une long-term clef k qui est utilisée pour établir de nouvelles clefs de session
- A choisit une clef et l'envoie à B en une passe :

$$A \rightarrow B : E_k(rA)$$

- Option : time-stamp, identité de B

Avec challenge-response

- On veut s'assurer que la clef est nouvelle (pas de rejeu) mais sans utiliser de time-stamp

  $A \leftarrow B : nB$

  $A \rightarrow B : E_k(rA, nB, B^*)$

- Le clef de session est rA

Avec challenge-response

- On veut que la clef soit fonction de A et B
- W = f(rA, rB)

  $A \leftarrow B : nB$

  $A \rightarrow B : E_k(rA, nA, nB, B^*)$

  $A \leftarrow B : E_k(rB, nB, nA, A^*)$

nA et nB sont des nonces utilisées pour garantir la « fraîcheur » de la clef

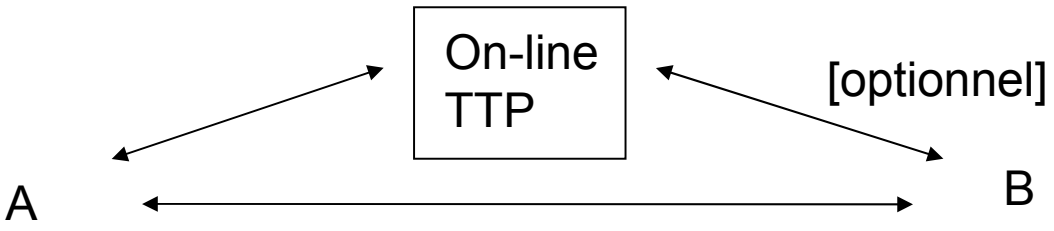
Sans clef partagée par A et B

- A choisit k comme clef de session
- p est premier (publique) p>k
- A (resp B) choisit aléatoirement a (resp b)
- A → B : $k^a \bmod p$ (1)
- A ← B : $(k^a)^b \bmod p$ (2)
- A → B : $(k^{ab})^{a^{-1}} \bmod p$ (3)
- B calcule $b^{-1} \bmod p-1$ et obtient k

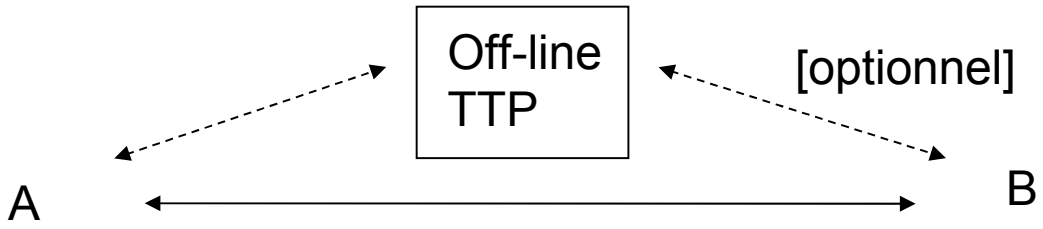
Distribution de clef avec serveur

- Un centre de clefs de confiance (TTP) On-line permet d'établir une clef de session
  - Chaque utilisateur à une "long-term private/secret key" qu'il partage avec le centre
- Remarque : pour la distribution de clef publique
  - Alice et Bob n'ont pas besoin de TTP
  - Un centre de clefs peut être utilisé pour stocker et distribuer les clefs publiques mais les utilisateurs ne partagent pas de clef privée avec le centre

Interaction avec le TTP



TTP peut communiquer avec A et B pendant le déroulement protocole



TTP ne participe pas
En temps réel

Rôles des TTP

- Autorité de certification (CA)
  - Authenticité des clefs
- Serveur de nom
  - Gestion (unicité) des noms
- Autorité de registre
  - Responsable des autorisations des utilisateurs
- Générateur, établissement, gestion de clefs
  - Publiques/privées, secrètes, passwords,…
- BD de certificats
  - BD accessible en read access

Distribution de clefs avec TTP

- Alice et Bob partagent chacun une clef secrète avec le serveur de clefs (TTP)
  - KB = clef de Bob ; KA = clef d'Alice
  - Le seul but de ces clefs est de communiquer avec le TTP
  - Ces clefs sont gardées d'une manière sûre par le TTP
  - Si KA n'est pas compromise, elle permet d'authentifier Alice au TTP et vice-versa

Protocole simple (crypto sym)

- Alice → TTP: A, B
  - TTP génère aléatoirement la clef de session K
- TTP → Alice: $E_{KA}(K)$, $E_{KB}(K)$
  - Alice déchiffre sa partie pour obtenir K
- Alice → Bob: $E_{KB}(K)$
  - Bob déchiffre sa partie pour obtenir K
- Alice → Bob: $E_{K}(M)$
  - Alice peut envoyer un message chiffré...

Replay Attacks

- Iwan intercepte les communications chiffrées
- Iwan renvoie les messages à Bob

  Iwan → Bob: $E_{KB}(K)$
- Bob déchiffre sa partie pour obtenir K

  Iwan → Bob: $E_{K}(M)$
- Iwan peut aussi prétendre être Alice
- Quelles peuvent être les conséquences ?

Protocole de Needham-Schroeder

(crypto sym)

- Alice → TTP: A, B, nA
- TTP → Alice: $E_{KA}(K, nA, B, E_{KB}(K, A))$
- Alice → Bob: $E_{KB}(K, A)$
- Bob → Alice: $E_K(nB)$
- Alice → Bob: $E_K(nB+1)$
- Alice → Bob: $E_K(M)$ etc.

- nA. nB sont des nombres aléatoires (nonces)

Autre Attaque

- Supposons que Iwan a cassé K et qu'il a gardé en mémoire

  Alice → Bob: $E_{KB}(K, A)$

- Iwan peut alors se faire passer pour Alice et envoyer des messages à Bob

Denning/Sacco

- Utiliser un timestamp T pour éviter le rejeu
- Alice → TTP: A, B
- TTP → Alice: $E_{KA}(K, B, T, E_{KB}(K, A, T))$
- Alice → Bob: $E_{KB}(K, A, T)$
- Alice → Bob: $E_K(M)$ etc.

- Nouveaux problèmes ?

TTPs comme serveurs d'Authentification

- Les clefs KA et KB authentifient Alice & Bob au TTP et vice versa, si elles ne sont pas compromises
- Si le TTP reçoit un message chiffré par KA, il ne peut provenir que d'Alice
- Si Alice reçoit un message chiffré par KA, il ne peut provenir que de TTP
- TTPs peuvent être utilisés comme serveurs d'authentification

Needham-Schroeder (crypto asym)

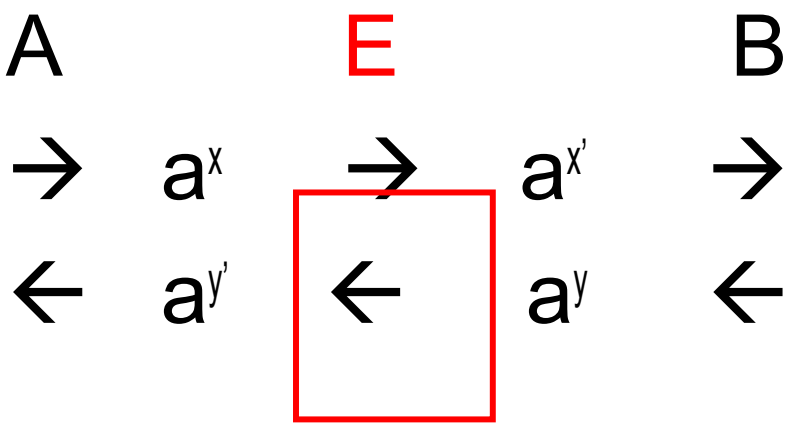
- Buts :
  - Authentification mutuelle
  - Echanger chacun un secret (k_1 et k_2)
  - Obtenir une clef de session $f(k_1,k_2)$
- $A \rightarrow B : P_B(k_1, A)$ (1)
- $A \leftarrow B : P_A(k_1, k_2)$ (2)
- $A \rightarrow B : P_B(k_2)$ (3)
- Comment éviter le chiffrement en (3)?

MTI (variante de DH)

- A et B veulent s'échanger une clef
- Clefs privées : A => a, B => b
- Clefs publiques : za = α^a, zb = α^b
- A→B : $\alpha^x \bmod p$
- A←B : $\alpha^y \bmod p$
- A calcule k = $(\alpha^y)^a zb^x \bmod p$
- B calcule k = $(\alpha^x)^b za^y \bmod p$

Sécurité des établissements de clefs (attaques)

- *Man in the middle*
- Sur le protocole de Diffie-Hellman



Utilisation des clefs

- Attention lors d'un chiffrement par logiciel si on utilise un SE multitache
- Lorsque le SE a une tache urgente à faire, il sauvegarde toutes les infos sur le disque
- La clef est donc sur le disque jusqu'à ce que l'ordinateur réécrive à cet endroit
- Un adversaire peut examiner le disque et retrouver la clef

Mise à jour des clefs

- A et B ont une clef k et une fonction f à sens unique communes
- A et B peuvent calculer f(k) et obtenir une nouvelle clef
- One time key

Stockage des clefs

- La clef de A peut être stockée
  - dans le cerveau de A
  - Dans une carte magnétique
  - Dans une clef USB
  - À moitié dans une carte et à moitié dans l'ordinateur
  - Chiffrée (par exemple avec DES) sur un disque
  - Dans une base de donnée centrale ou privée

Clef compromise

- Clef secrète
  - Changer la clef
- Clef privée
  - La clef publique ne doit plus être utilisée
  - Diffuser l'info à tous les serveurs du réseau
  - Essayer de savoir quand la clef a été compromise
  - L'utilisation de datation permet de différencier les messages suspects des légitimes
  - Utiliser une clef pour chaque application

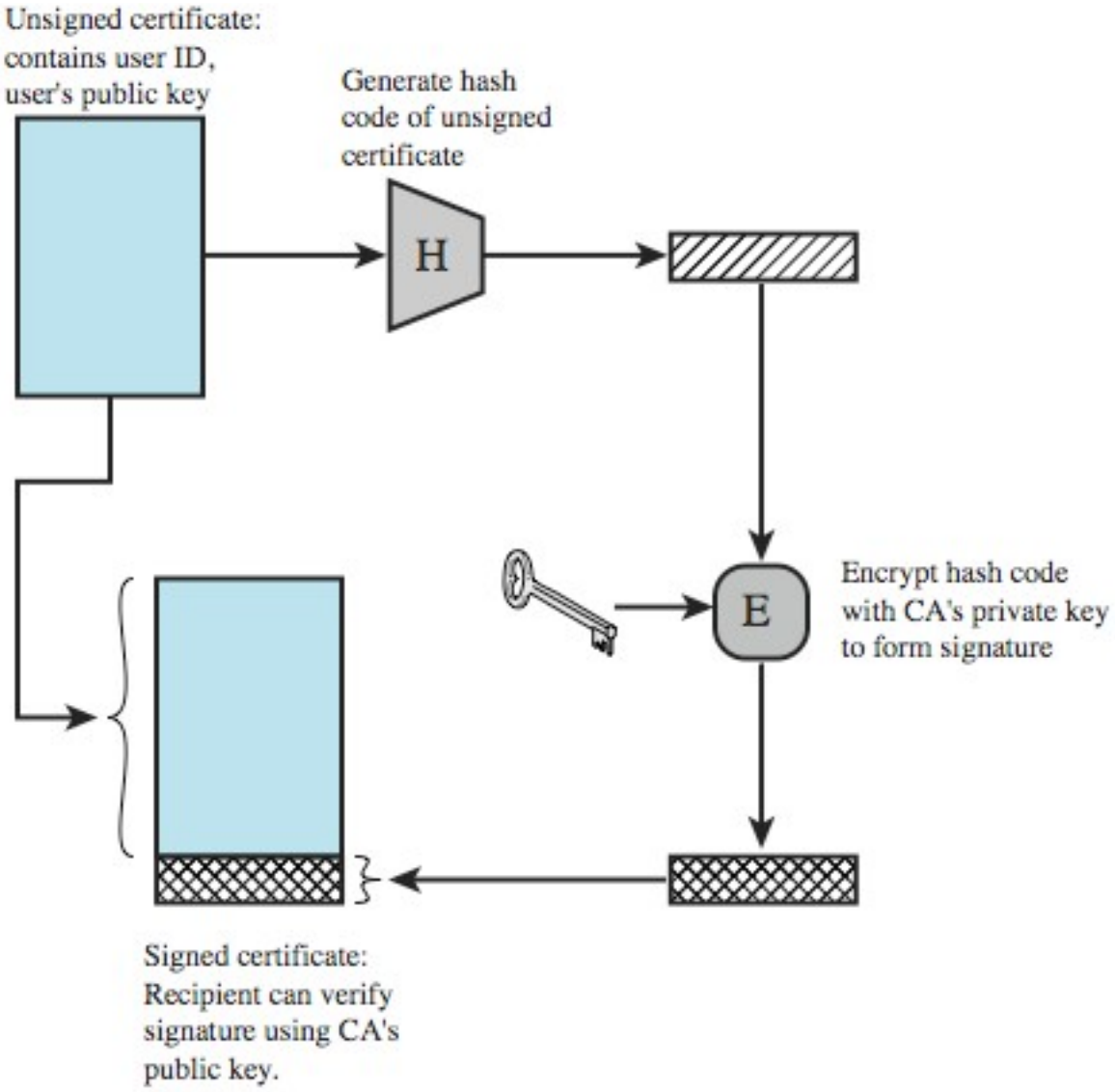
Destruction de clefs

- Les vieilles clefs permettent de lire d'anciens messages chiffrés
- Elles doivent être détruites de manière sûre
- Sur papier : brûler ou utiliser une machine à déchiqueter de bonne qualité
- Sur mémoire
  - La puce doit être broyée
  - Réécrire plusieurs fois sur le disque (ou broyer!)
  - Programme d'effacement de disque
  - Effacer le contenu des fichiers temporaires

Gestion de clefs publiques

- Pour obtenir la clef publique de Bob, Alice peut
  - L'obtenir auprès de Bob
  - L'obtenir auprès d'une BD centrale
  - L'obtenir à partir de sa propre BD privée
  - L'obtenir avec un certificat à une BD TTP

Certificats



Certificate:
Data:
Version: 3 (0x2)
Serial Number: 1 (0x1)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/Email=server-certs@thawte.com
Validity Not Before: Aug 1 00:00:00 1996 GMT
Not After : Dec 31 23:59:59 2020 GMT
Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/Email=server-certs@thawte.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption RSA Public Key: (1024 bit)
Modulus (1024 bit): 00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c: 68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06: 6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2: 6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:3a:c2:b5:66:22:
12:d6:87:0d
Exponent: 65537 (0x10001)
X509v3 extensions: X509v3 Basic Constraints: critical
CA:TRUE
Signature Algorithm: md5WithRSAEncryption
07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9: a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48: 3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9: 8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5: e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e: 70:47

Gestion distribuée des clefs (PGP)

- Suppose qu'il n'existe pas d'AC en laquelle Alice et Bob ont confiance
  - Utiliser des parrains qui signent les clefs publiques de leurs amis : Bob présente sa clef publique à Alice avec les signatures de Bernard et Christine
  - Si Alice connaît Bernard ou Christine, elle acceptera la clef comme valide