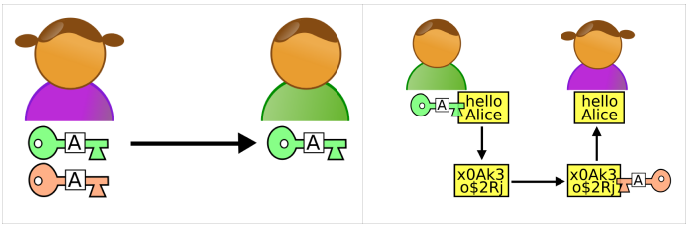
Chiffrement symétrique: nombre et diffusion des clefs

- Pour permettre à n personnes de communiquer,
  - il faut que chaque personne ait les clefs des (n-1) autres.
  - Au total: n(n-1) clefs
- Centrale de clefs:
  - Elle a la clef secrète de chaque utilisateur
  - Les utilisateurs lui envoient les messages
  - Elle les déchiffre et le rechiffre avec la clef privée du destinataire
  - Il faut n clefs
  - La centrale peut lire les messages de tout le monde et doit stocker les clefs de façon sûre.

Principe du chiffrement asymétrique

- La clef de chiffrement est publique
- La clef de déchiffrement est secrète et ne sert qu'au déchiffrement
- La clef publique
  - Ne permet pas de déchiffrer
  - Ne permet pas à un attaquant de trouver la clef privée



Chiffrement à clef publique

- Il est constitué de 3 algorithmes
  - L'algorithme de génération des clefs
    - (pk, sk)=KG(l) produit un couple clef privée, clef publique à partir d'un paramètre de sécurité l (souvent une saisie aléatoire)
  - L'algorithme de chiffrement
    - C=E(M, pk) utilise la clef publique pk
  - L'algorithme de déchiffrement
    - M=D(C, sk) utilise la clef privée sk
  - On a M=D(E(M, pk), sk)
  - On a **parfois** M=E(D(M, sk), pk). Les procédés qui vérifient cette seconde propriété peuvent servir de base à un procédé de signature

Le chiffrement RSA

- Soit deux grands nombres premiers impairs p et q
- n=pq
- Soit e un entier tel que
  - $1<e<\varphi(n)=(p-1)(q-1)$. $\varphi(n)$ est l'ordre du groupe multiplicatif (Z/nZ)*
  - PGCD(e, (p-1)(q-1))=1 (i.-e. e est premier avec $\varphi(n)$)
- Soit d tel que $d=e^{-1} \bmod \varphi(n)$ (d est l'inverse de e dans (Z/nZ)*) donc il existe u tel que $ed+u\varphi(n)=1$
- D'après le théorème d'Euler ($x^{\varphi(n)}=1 \bmod n$)
  - $(m^e)^d=m^{ed}=m^{1-\varphi(n)}=m \bmod n$

RSA:

- Informations publiques :
  - n=pq (p et q premiers)
  - e
- Informations privées:
  - $d=e^{-1} \bmod \varphi(n)$: clef secrète

- Génération de clefs: ((n,e), d)=KG(l)
- Chiffrement: $C=E(m)=m^e \bmod n$
- Déchiffrement: $m=D(C)=c^d=m^{ed}=m \pmod n$

Propriétés de RSA

- Propriété multiplicative
  - Le chiffré d'un produit est égal au produit des chiffrés
    - $(m_1m_2)^e=m_1^e\, m_2^e=c_1c_2 \bmod n$
  - Conséquence: attaque à chiffré choisi
    - Soit C un message chiffré (C=E(m), m est inconnu)
    - Supposons qu'on connaisse r et C' tels que
      - On sait déchiffrer C' en m'
      - $C'=r^eC$
    - Alors on sait déchiffrer C car m'=r.m

Propriétés de RSA

- Chiffrement et déchiffrement sont commutatifs
- $m=D(C(m))=m^{ed}=C(D(m))=m^{de}$
- Conséquence: RSA pourra être utilisé comme base d'un algorithme de signature:
  - On signe un document en chiffrant avec la clef privée
  - On le vérifie en déchiffrant avec la clef publique
- Pouvoir être déchiffré avec la clef publique est-il une preuve que ça a été chiffré avec une clef privée donnée ?

RSA: sélection des paramètres

- Choisir p et q de taille similaire pour éviter la factorisation par courbes elliptiques
- Choisir p et q au hasard pour éviter que p et q soit trop proches (sinon p et q de l'ordre de racine de n)
- Choisir p et q des nombres premiers forts
  - p-1 a un grand facteur premier (pour éviter l'algo p-1 de Pollard). grand=au moins 200bits
  - p+1 a un grand facteur premier (pour éviter l'algo de Williams
  - r-1 a un grand facteur premier (attaques par cycles)
- Tiré de [Bresson]

Sécurité de RSA

- 2 méthodes d'attaque
  - Trouver d ou p et q : problème de factorisation
  - Trouver m à partir de m^e: extraction de racines e-ièmes
- En pratique, la factorisation est la seule méthode connue pour casser RSA
  - C'est un problème difficile
- Il n'a pas été prouvé que RSA était aussi solide que la factorisation
  - Apparition d'une attaque plus simple que la factorisation possible

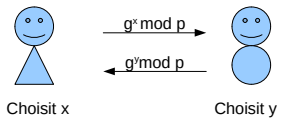
Diffie Hellman

- Années 1970: comment se passer de la transmission d'un secret partagé ?
- Réponses :
  - Chiffrement à clef publique: RSA 1977
  - Diffie Hellman 1976 : se mettre d'accord sur un secret partagé en n'échangeant que des informations publiques

Diffie Hellman: de l'importance la commutativité

- Exemple d'algorithme d'échange de clef :
  - Alice met un secret dans un coffre
  - Alice ferme le coffre avec un cadenas
  - Alice envoie le coffre à Bob
  - Bob ajoute au coffre un cadenas
  - Bob envoie le coffre à Alice
  - Alice retire son cadenas
  - Alice envoie le coffre à Bob
  - Bob retire son cadenas et lit le secret
- Le fait fondamental est que les cadenas peuvent être mis et retirés dans n'importe quel ordre
- Exemple incorrect: Bob met le coffre d'Alice dans un autre coffre fermé à clef par Bob

Diffie Hellman



- Valeurs publiques:
  - p un grand nombre premier
  - G un groupe multiplicatif de cardinal p-1
  - g un générateur de G
- Secret d'Alice: x, secret de Bob: y
- Secret commun: $K=g^{xy}=(g^x)^y=(g^y)^x$
- Attaque: trouver x connaissant g^x
  - Problème du logarithme discret
  - Pas d'attaque raisonnable si p est grand

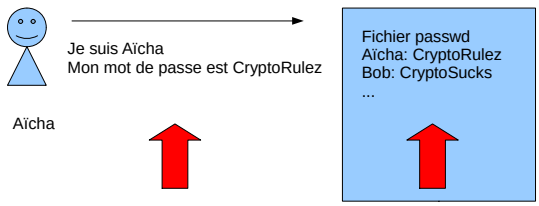
Hachage

- Fonction h à sens unique
  - Entrée de taille quelconque
  - Sortie de taille fixe
  - h(m) est appelé l'empreinte (hash) du message m
- Propriétés:
  - Facile à calculer
  - Inversion difficile: « matériellement impossible » de retrouver m à partir de h(m) (préimage)
  - Seconde préimage: m est donné. Trouver m' tel que h(m)=h(m') (enjeu: liberté sur certaines parties de m')
  - Collission difficile (ou faiblement résistante aux collisions): « matériellement impossible » de retrouver m' tel que h(m)=h(m')
  - Dispersion: changer un bit du message initial change de nombreux bits de la sortie

Hachage: un mauvais exemple

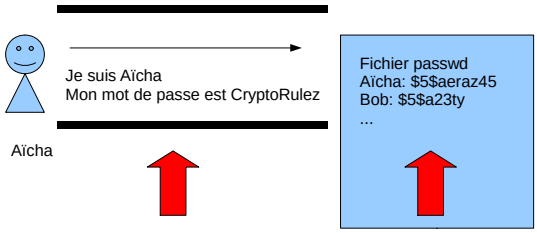
- Posons h(m)=b1 XOR b2 XOR b3 … XOR bn avec m=b1b2b3...bn
- h(10110)=1
- Trouver m tel que h(m)=0 (facile)
- Collisions: h(111)=h(001)=1

Hachage: applications



- Si le mot de passe :
  - Passe en clair sur le réseau : peut être espionné
  - Est stocké en clair: peut-être volé
  - Le vol de données personnelles et de mot de passe est un problème courant sur les sites WeB
  - => utiliser des mots de passe différents

Hachage: applications



- Solution :
  - Stocker l'emprunte du mot de passe

Hachage: applications

- Stockage des mots de passe:
  - On stocke h(mdp)=hash
  - Quand un utilisateur à authentifier saisit son mot de passe m,
    - h(m) = hash ? Oui => OK
  - Pb: éviter que deux utilisateurs ayant le même mot de passe aient la même valeur stockée
    - Ajout d'une valeur (grain de sel ou « salt ») : s
    - On stocke s et h(s+mdp)=hash
    - Stocker s : pour calculer h(s+m)
  - Attaque:
    - le pirate peut avoir h(mdp)
    - Il cherche m' tel que h(m')=h(mdp)

Hachage: applications

- Validation de l'intégrité des données
  - On obtient m (par un canal performant) et h (par un canal sur)
  - On vérifie que h=h(m) pour vérifier l'intégrité de m
  - Nécessite que h soit faiblement résistance aux collisions.
- Attaque:
  - Proposer m' (contenant un cheval de troie par ex.) tel que h(m')=h

Hachage: applications

- signature de données volumineuses
  - Signature de donnée m
    - On calcule h(m)=h
    - On signe h(m) avec un procédé de signature s : sig
    - On transmet m, h et sig
  - Vérification de la signature
    - On vérifie l'intégrité de h grâce à sig
    - On vérifie que h=h(m) pour vérifier l'intégrité de m
  - Nécessite que h soit faiblement résistance aux collisions.
- Attaque:
  - Proposer m' (contenant un cheval de troie par ex.) tel que h(m')=h

Hachage: attaque des anniversaires

- On calcule et on stocke autant de valeur de h qu'on le peut
- Question: combien en faut-il pour avoir un probabilité ½ de trouver une collision (2 valeurs différentes ayant la même empreinte) ?
- Réponse : (1+ sqrt(1+8*ln(2)*2^n))/2
  - Soit un peu plus que $2^{n/2}$ (n est la taille de l'empreinte en nombre de bits)
  - De nos jours (12/2008), 128 ou 160 sont des valeurs minimales

Fonction de compression: construction à partir d'algo de chiffrement

- g(k,x) = e(k,x)
- g(k,x) = e(k,x) XOR x
- g(k,x) = e(k,x) XOR x XOR k
- g(k,x) = e(k,x XOR k) XOR x
- g(k,x) = e(k,x XOR k) XOR x XOR k

- Si e est solide, on a l'impression que g est solide
- Rien de prouvé

Hachage: fonction de compression

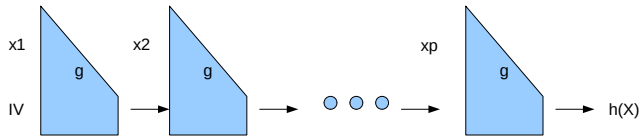
- Soit g :

$\{0,1\}^m \rightarrow \{0,1\}^n$

- r=m-n
- r > 0
- g résistante aux collisions

Idée: enchaîner g suffisamment pour passer d'un mot M de taille quelconque à un mot de taille donnée.
On pose M=X=x1x2x3x4...xp
- Avec taille de Xi=r
- Quitte à ajouter des bits de bourrage si la taille de X n'est pas divisible par r

On choisit un IV de taille n
IV+x1 est de taille m (+ : concaténation)
g(IV+x1) est de taille n



Ce schéma simple n'est pas assez solide. Il le devient si on remplace X par une valeur convenablement choisie déduite de X : Construction de Merkle-Damgård

Hachage: Construction de Merkle-Damgård

- Transformation appliquée à M
  - Ajout de zéros en tête de façon à avoir un nombre de chiffres divisible par r
  - Ajout de r zéros en queue de M
  - Soit l la longeur de M
    - On découpe l en bloc de r-1 bits : l1l2l3...lq
    - On ajoute le chiffre 1 devant chaque li
    - On colle le résultat à la fin
  - Donc X=0..0M00..01l11l21l3...1lq
- On pose X=x1x2...xp et IV=O (n chiffres zéro)

Construction de Merkle-Damgård: exemple

- R=4, M=01001010101010101
  - On ajoute3 zéros en tête pour obtenir une taille multiple de 4 bits
  - On ajoute 4 zéros à la fin
  - La taille de M est 17 bits donc10001 en base 2. Cela fait 5 chiffres. O n ajotue un zéro devant pour avoir un nombre de chiffres divisible par 3.L_1=010 L_2=001
- X= « les 3 zéros » + M + « 4 zéros » + $1L_1$ + $1L_2$
- X=000<u>100101010101010101</u>0000<u>1010</u>1001
- Faire intervenir la taille dans le processus permet d'éviter des attaques où h(M)=h(M') mais taille M ≠ taille M'

Hachage: fonctions classiques

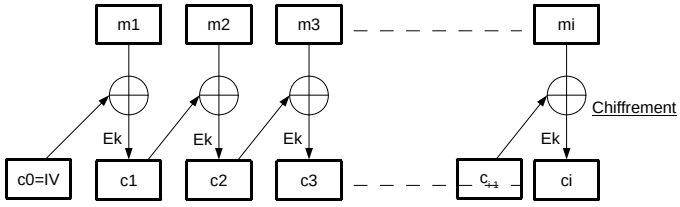
- SHA-1, SHA-256, SHA-512
- MD4, MD5
- MD4, MD5 et SHA-1 sont à éviter

MAC: Message Authentification Code

- But: authentifier un message à l'aide d'un authentifiant de petite taille (de 64 à 256 bits)
- Crypto à clef secrète:
  - un MAC est une fonction de hachage qui prend une clef secrète en argument
  - L'expéditeur et le destinataire du message doivent avoir la clef secrète
- Exemple d'utilisation
  - Alice envoie m à Bob avec son MAC h=H(k, m)
  - Bob reçoit m' et h' (a priori m'=m et h'=h)
  - Bob calcule H(k,m') et le compare à h'. égalité => OK
  - Un attaquant ne pourrait pas créer m' et h' valides sans connaître k.
  - Remarque: la fonction de déchiffrement ne sert pas.

MAC: CBC-MAC

- CBC-MAC: on applique la méthode CBC et on garde seulement le dernier bloc chiffré



Mac versus signatures

- Signature
  - Mécanisme à clef publique
  - Non répudiation
  - Authentification du signataire
  - Intégrité des données
- MAC:
  - Mécanisme à clef secrète
  - Authentification de l'émetteur
  - Intégrité des données

PKI: Public Key Infrastructure (IGC: Infrastructure de Gestion de Clefs)

- Problème:
  - comment être sur qu'une clef publique est valide et est bien la clef publique d'une personne donnée ?
- Solutions:
  - PGP: confiance transitive : WeB Of Trust
  - Certification par un tiers de confiance : IGC

IGC: définition

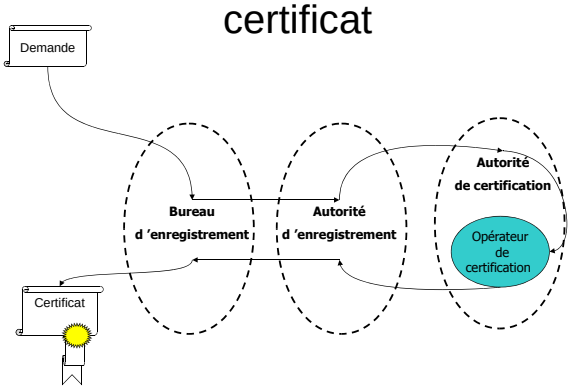
- IGC: ensemble de moyens matériels, de logiciels, de composants cryptographiques mis en oeuvre par des personnes, combinés par des politiques, des pratiques, des procédures requises qui permettent de :
  - créer
  - gérer
  - conserver
  - distribuer
  - révoquer

  des certificats basés sur la cryptographie asymétrique

éléments obligatoires d'une IGC

- 3 éléments obligatoires :
  - autorité de certification
  - autorité d'enregistrement
  - service de publication

PKI: processus de création d'un certificat



•autorité de certification (CA ou Certification Authority en anglais)

- autorité de confiance reconnue par une communauté d'utilisateurs
- délivre et gère des certificats de clefs publiques
- maintient une liste des certificats révoqués (LCR en français, CRL en anglais)
- les certificats sont conformes à la norme X.509
- génère les certificats à clef publique et garantit l'intégrité et la véracité des informations qu'ils contiennent en les signant avec sa clef privée

•Autorité d'enregistrement (RA: Registry Authority)

- intermédiaire entre l'utilisateur et l'autorité de certification.
- l'utilisateur s'adresse à elle
- en application de la politique de certification, elle vérifie les données de l'utilisateur :
  - identité
  - correspondance clef privée/publique
  - ...
- transmet les informations validées à l'AC

•Service de publication

- met à la disposition de la communauté les certificats générés par l'AC
- publie aussi la liste des certificats révoqués

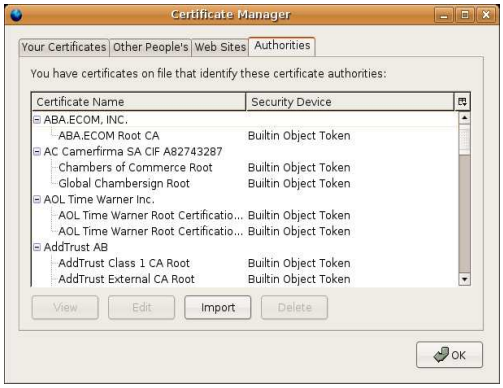
•Composants optionnels de l'IGC

- autorité d'horodatage (AH ou Timestamping Authority)
  - date des données qui lui sont transmises
  - Le Protocole D'horodatage (ou Time-Stamp Protocol) : rfc 3161
- service de séquestre:
  - stocke de façon sûre des clefs privées
  - pour permettre le déchiffrage des données en cas de perte
  - ne doit pas concerner les clefs de signature

•Certificat

- contient entre autre
  - l'identité de son propriétaire (personne, machine, ...)
  - sa clef publique signé par une AC
  - période de validité
  - type d'utilisation de la clef (champ optionnel)
  - ...

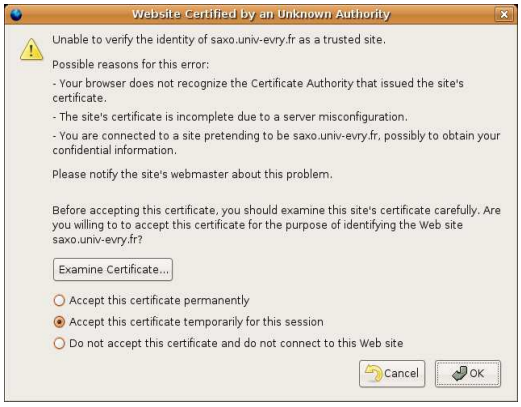
•Exemple: navigateur WeB



•Exemple: navigateur WeB:

- un client se connecte sur le site WeB de l'entreprise
- il obtient les références de l'AC et le certificat
- il vérifie le certificat
- il génère une clef de session qu'il transmet chiffrée au serveur de l'entreprise
- la session est maintenant chiffrée

•Exemple: Pb certification



•Exemple: certificat