Hachage, Signatures et Certification

Bruno MARTIN,
Université Côte d'Azur

Plan

Classification

- MDC ou MIC (message integrity code) à IV=0
  - fonctions de hachage à sens unique (OWHF)
  - fonctions de hachage résistantes aux collisions (CRHF)
- MAC
  - assure authentification et intégrité
  - IV≠0

Conditions à satisfaire

Une fonction de hachage h calcule

$$z = h(x)$$

où x message de taille arbitraire et z empreinte de taille fixe.
h **à sens unique** (OW), i.e.

- $h(x)$ rapide à calculer à partir de x
- z difficile à inverser.

Résistance aux collisions

collision : 2 mots $x \neq x'$ tels que $h(x) = h(x') = z$.
h vérifie

- **préimage-résistance** si, donné h, z, il est difficile de trouver un x tq $h(x) = z$
- **2e préimage-résistance** si, donnés h, x, il est difficile de trouver x' tq $x \neq x'$ et $h(x) = h(x')$
- **résistance aux collisions** si donné h il est difficile de calculer la moindre collision (x, x').

Difficile = sécurité calculatoire (sait-on faire mieux ?)
Relations entre les propriétes de résistance :

1. résistance aux collisions $\Rightarrow$ 2e préimage-résistance
2. résistance aux collisions ne garantit pas la préimage résistance

Formalisation d'une fonction de hachage

Définition

Une fonction de hachage Π est la donnée de 2 algos PPT (Gen, h) qui satisfont :

- *Gen est un algo PPT reçevant 1^n et produisant s utile pour choisir une fonction de hachage au sein d'une famille*
- *$\exists \ell$ polynôme tq h_s sur l'entrée $x \in \{0,1\}^\star$ fournit $h_s(x) \in \{0,1\}^{\ell(n)}$ où n est la valeur du paramètre de sécurité implicite dans s*

Si h_s n'est définie que pour des entrée $x \in \{0,1\}^{\ell'(n)}$ pour $\ell'(n) > \ell(n)$, (Gen, h) est une fonction de hachage à longueur fixée sur des entrées de taille $\ell'(n)$ (fonction de compression).

Formalisation de la sécurité

Définition (Expérience de recherche de collisions Hash-coll$_{A,\Pi}(n)$)

1. *Engendrer s par Gen(1^n)*
2. *l'adversaire A reçoit s et construit x, x'. Si Π est une fonction de compression d'entrées de taille $\ell'(n)$, $x, x' \in \{0,1\}^{\ell'(n)}$*
3. *la sortie de l'expérience est 1 ssi $x \neq x'$ et $h_s(x) = h_s(x')$. Dans ce cas, A a trouvé une collision.*

Formalisation de la résistance aux collisions

Définition

Une fonction de hachage $\Pi = (Gen, h)$ est résistante aux collisions si pour tout adversaire A PPT, il existe une fonction negl tq

$$Pr(Hash\text{-}coll_{A,\Pi}(n) = 1) \leq negl(n)$$

C'est la plus forte condition de sécurité qu'on puisse demander à une fonction de hachage.

Modèle de l'oracle aléatoire

Boîte noire déterministe H qui, sur l'entrée x, engendre une sortie aléatoires y et qui fonctionne comme une fonction à sens unique.

Définition

Un oracle aléatoire RO prend en entrée un mot binaire de longueur arbitraire et retourne une suite aléatoire infinie (ou tronquée).

Permet de construire une fonction de hachage CR. La probabilité de succès de tout adversaire A PPT dans l'expérience suivante est négligeable :

1. on choisit une fonction aléatoire H
2. A réussit s'il trouve $x \neq x'$ tel que $H(x) = H(x')$

Modèle abstrait qui permet de construire des preuves de sécurité et qui a donné pratiquement de bons résultats.

Attaques

- contre MDC :
  - OWHF : donné z, trouver x tq $h(x) = z$.
  - CRHF : trouver deux entrées $x \neq x'$ tq $h(x) = h(x')$ (attaque des anniversaires).
- contre MAC :
  - sans connaître k, donnés $(x_i, h(x_i))$, calculer $x, h_k(x)$, $x \neq x_i$
  - KPA, CPA, ...
  - falsification sélective et existentielle

(beaucoup plus de résultats de sécurité prouvée)

Objectifs de sécurité

Type de H	Conception	sécurité	Adversaire
OWHF	préimage-res	2^n	trouve préimage
	2-préimage-res	2^n	trouve 2^e préimage
CRHF	rés. collisions	$2^{n/2}$	trouve collision
MAC	key recovery	$2^{\sharp k}$	trouve clé MAC
	sécurité calc.	$\min\{2^n, 2^{\sharp(k)}\}$	trouve nouv. MAC.

n : taille empreinte
$\sharp k$: taille clé (IV)

Quelques attaques de base

- tentatives répétées : une fonction de hachage de long. n a une robustesse idéale si elle satisfait les bornes sup. de OWHF et CRHF
- recherche exhaustive de clé MAC (KPA) demande $2^{\sharp k}$ opérations
- deviner MAC demande 2^n opérations
- précalcul d'empreintes `http://gdataonline.com/` (fini)
- parallélisation pour 2^e préimage

Attaque par force brute

En $O(|D|)$ sur $h : \{0,1\}^k \times D \to \{0,1\}^n$ tq $(D, <) = (D_1, \dots D_d)$.

Algorithme

$x_1 \xleftarrow{\mathcal{U}} D$; $y := h_k(x_1)$
pour $i := 1 \dots d$ **faire**
  si $(h_k(D_i) = y \wedge x_1 \neq D_i)$ **alors**
    retourne x_1, D_i
  fsi
retourne *echec*

Paradoxe des anniversaires

Donnée : $B = (b_1, \dots, b_k) \in \{1, 2, \dots, n\}^k$.
Problème : Proba p qu'il existe au moins 2 élts identiques de B ?

$$1 - p = q = 1\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\dots\left(1 - \frac{k-1}{n}\right) = \prod_{i=1}^{k-1}\left(1 - \frac{i}{n}\right)$$

1 proba de tirer b_1 ; $\left(1 - \frac{1}{n}\right)$ proba de tirer $b_2 \neq b_1$ car $P(b_2 = b_1) = \frac{1}{n}$;
$\left(1 - \frac{i}{n}\right)$ proba de tirer $b_i \neq b_1, \dots, b_{i-1}$
Comme $1 - x \leq e^{-x}$: $q \leq \prod_{i=1}^{k-1} e^{-\frac{i}{n}} = e^{-\frac{1}{n}\sum_{i=1}^{k-1} i} = e^{-\frac{(k-1)k}{2n}}$.
Passage au logarithme : $2n \ln q = -(k-1)k \Leftrightarrow 2n \ln \frac{1}{q} \propto k^2$
Pour $p > \frac{1}{2}, k \geq \sqrt{2n \ln 2}$ et $k \propto O(\sqrt{n})$ car $\sqrt{2 \ln 2} = 1,17$
Exemple : $n = 365$, $k = 23$ personnes. $(n = 12,\ k = 5)$.
Utilité : dimensionner la longueur n de l'empreinte d'une fonction de hachage pour éviter les collisions.

Attaque basée sur le paradoxe de $h : \{0,1\}^\star \to \{0,1\}^n$

Calculer et trier des couples $(x, h(x))$ pour détecter des collisions.
Combien de couples pour en trouver 2 identiques parmi 2^n à $p > 1/2$?
Hyp : les images par h suivent une distribution uniforme.
Pour k entrées on a $p > 1/2$ d'avoir une collision avec $k \propto O(\sqrt{2^n}) \approx 2^{\frac{n}{2}}$

n	50	100	150	200
k	25	50	75	100

En calculant $> 2^{n/2}$ empreintes, on a une collision avec proba $> 1/2$.
Pour que h soit CR, on choisit n pour que le calcul de $2^{n/2}$ images par h soit irréaliste. A ce jour, $n \geq 128$ voire même $n \geq 160$.

Fonctions de hachage efficaces

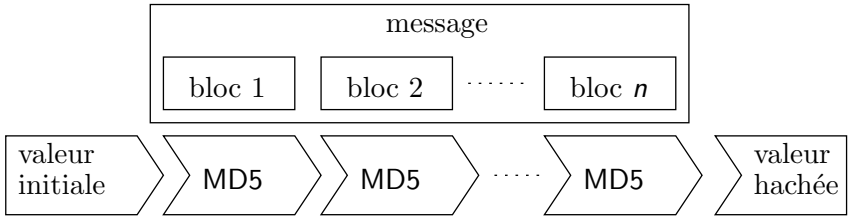
Les plus courantes :

nom	bits	tours×étapes	vitesse relative
MD5	128	4×16	1
SHA	160	4×20	0,28
SHA 3	256	24	0,25

A quoi servent-elles et comment sont-elles construites ?

Hachage par compression (Merkle-Damgård)

Dans la plupart des algos, on découpe x en n blocs et on effectue :



Construire une fonction de hachage cryptographique

Partir d'un chiffre symétrique e_k, construire une fonction de compression :

$$g : \{0,1\}^m \to \{0,1\}^n \qquad \text{pour } m, n \in \mathbb{N}, \quad m > n$$

Etendre la fonction de compression en fonction de hachage :

$$h : \{0,1\}^\star \to \{0,1\}^n \qquad \text{pour } n \in \mathbb{N}$$

Construction d'une fonction de compression

A partir de $e_k : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$
on construit une fonction de compression

$$g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \qquad \text{pour } n \in \mathbb{N}$$

Le chiffre est utilisée soit directement s'il est résistant aux collisions soit en le $\ll$ perturbant $\gg$ un peu plus comme p.e.

$$g(k,x) = e_k(x) \oplus x$$
$$g(k,x) = e_k(x) \oplus x \oplus k$$
$$g(k,x) = e_k(x \oplus k) \oplus x$$
$$g(k,x) = e_k(x \oplus k) \oplus x \oplus k$$

Construction d'une fonction de hachage (Merkle-Damgård)

Soit $r = m - n > 1$. On construit $h : \{0,1\}^\star \to \{0,1\}^n$ à partir de $g : \{0,1\}^m \to \{0,1\}^n$ (compression).
Soit $x \in \{0,1\}^\star$ et ℓ sa longueur en binaire.

- compléter x avec des "0" : $u = 0^i x$ t.q. $|u| \equiv 0 \mod r$
- compléter ℓ avec des "0" : $y = 0^j \ell$ t.q. $|y| \equiv 0 \mod r-1$
- découper y en blocs de $r-1$ bits et ajouter un "1" au début de chacun des blocs pour former le mot v
- construire $w = u0^r v$ composé de t blocs de longueur r.

Exemple

$r = 4$, $x = 11101$, $\ell = 101$. *On forme* $u = \mathbf{000}1\,1101$, $v = \mathbf{1}101$.

$$w = 0001\,1101\,\mathbf{0000}\,1101 = w_1 w_2 w_3 w_4 \ (t = 4)$$

H déf. inductive : $H_0 = 0^n$; $H_i = g(H_{i-1} w_i)$, $1 \le i \le t$, $h(x) = H_t$

Construction en éponge

Nouvelle construction (2007) d'une fonction de hachage [1] qui permet des preuves de sécurité.

F calcule une empreinte de taille fixée sur une entrée P de longueur qcq.

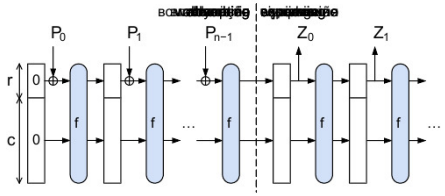


Figure – construction en éponge image Wikipedia

Etat interne de largeur $b = r + c$ bits ; de débit r et de capacité c.
f permute b bits et conçue algébriquement comme la boîte S de AES.

Construction en éponge

L'entrée est complétée avec une règle de bourrage réversible et découpée en blocs de r bits $(P_0, P_1, \ldots)$. Les b bits de l'état sont initialisés à 0 et on applique les 2 étapes de la construction :

absorption : blocs d'entrée XOR avec les r bits de l'état puis mélangés par f. Une fois les blocs absorbés :

essorage : les r premiers bits de l'état sont extraits et servent de sortie avant un mélange par f. Le nombre de blocs de sortie est fixé par l'utilisateur.

Les c derniers bits de l'état ne sont jamais affectés directement par l'entrée et ne produisent jamais de sortie.

Règle de bourrage

$P = M \,||\mathrm{pad}[x](\sharp M)$ dénote le bourrage de M à la taille x

Définition

Une règle de bourrage est compatible si elle ne retourne jamais le mot vide et si elle satisfait :

$$\forall n \geq 0, M, M' \in \mathbb{Z}_2^\star : M \neq M' \Rightarrow$$
$$M \,||\, pad[r](\sharp M) \neq M' \,||\, pad[r](\sharp M') \,||\, 0^{nr}$$

Exemples de règles de bourrage compatibles :

- bourrage simple qui ajoute à droite $10^\star$
- bourage multi-débit qui ajoute à droite $10^\star 1$

pour obtenir un multiple de la taille des blocs

Plan

Utilisation

- mécanisme de base dans les protocoles cryptographiques
- outil essentiel pour des applications
  - cryptographiques
  - et autres...
  - dont les preuves de travail pour les cryptomonnaies,...

Vérification de mots de passe

- mdp de A est p et le serveur mémorise $\overline{p} = h(p)$
- $A \to B$: p sur canal sûr ; B vérifie $h(p) = \overline{p}$

sûr tq Eve qui obtiendrait $\overline{p}$ ne peut retrouver p (OWHF)
attaques possibles par dictionnaire.

Comparaison par hachage

- A (resp. B) possède un grand fichier F_A (resp. F_B)
- ils veulent savoir si $F_A = F_B$
- $A \to B$: $h(F_A)$ et B teste $h(F_A) = h(F_B)$

Identification & authentification

identification affirmation de l'identité d'une entité au moyen de son identifiant. (Je suis Bruno)
authentification procédé de vérification de l'identité d'une entité. (Je suis Bruno et en voici la preuve)

L'identification permet de connaitre l'identité d'une entité et l'authentification de vérifier cette identité.

Mots de passe jetables de Lamport [3]

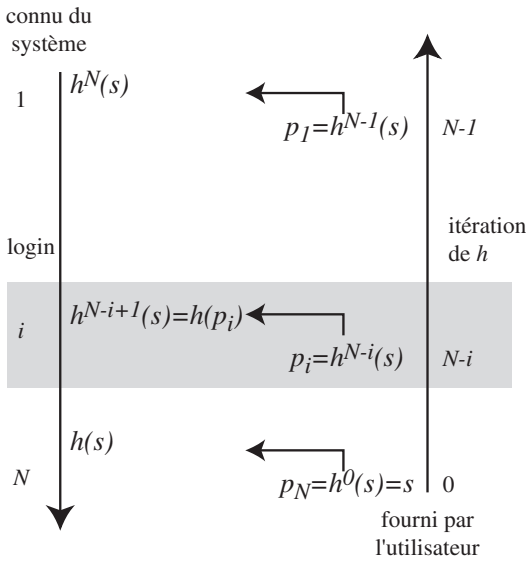
Application du hachage pour le service de contrôle d'accès.
Accès distant : identification par login/ mot de passe. Un pirate peut découvrir le mdp de l'utilisateur de 3 manières :

- en accédant aux informations contenues au sein du système, pe en accédant au fichier des mots de passe ;
- en interceptant l'envoi du mot de passe par l'utilisateur par l'espionnage de la ligne de connexion ou par un programme d'espionnage sur l'ordinateur de l'utilisateur ;
- par la divulgation du mot de passe de l'utilisateur si ce dernier a choisi un mot de passe facile à deviner.

Remèdes

- conserver les mots de passe chiffrés. C'est ce qui est réalisé à l'heure actuelle dans la plupart des systèmes.
- ne pas utiliser un unique mot de passe mais une suite de mots de passe. Méthode utilisée dans les systèmes à mot de passe jetables (OTP : One time passwords) comme `S/Key`, `Opie`,... au moyen d'une fonction de hachage cryptographique.
- De plus en plus appel à des systèmes biométriques (empreintes vocales, digitales ou rétiniennes).

Fonctionnement des OTP



Fonctionnement des OTP

On fixe N nombre maximal de connexions et s secret initial. Le système et l'utilisateur partagent soit une fonction à sens unique soit une fonction de hachage cryptographique h.
Le i^{e} mot de passe p_i est $h^{N-i}(s)$. La suite des N mots de passe de connexion fournis par l'utilisateur est :

$$h^{N-1}(s), h^{N-2}(s), \ldots, h(h(h(s))), h(h(s)), h(s), s$$

et celle du système pour identifier l'utilisateur par :

$$h^{N}(s), h^{N-1}(s), \ldots, h(h(h(s))), h(h(s)), h(s)$$

Fonctionnement des OTP

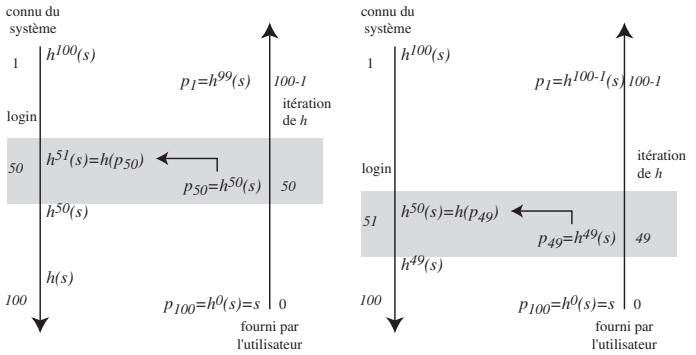
Connexion i/N ; connus du système : h^{N-i+1} et i n° de connexion.

Demande de connexion
- Système envoie un défi : numéro de connexion, i
- utilisateur utilise h en fonction de s, N et i : calcule $p_i = h^{N-i}(s)$ et le transmet au système

Vérification système reçoit $p_i = h^{N-i}(s)$, calcule $h(p_i) = h(h^{N-i}(s)) = h^{N-i+1}$ qu'il connaît. S'il y a égalité, il accepte la connexion et mémorise la valeur p_i reçue de l'utilisateur pour la prochaine connexion.

Chaque mdp est la valeur requise par le système pour la connexion suivante.

Deux connexions consecutives



Et maintenant ?

Avenir du hachage :

- MD5 : no future
- SHA-0 : collisions touvées
- SHA-1 ; recherche collisions en 2^{69} ; boincstats (over). Premières collisions construites en 2017 requièrent 1 siècle GPU mais plus rapide qu'une recherche exhaustive d'un facteur 100000

Et maintenant – suite – ?

- NIST : nouveau concours pour fonctions de hachage `http://csrc.nist.gov/groups/ST/hash/index.html` ; paramètres demandés (deadline était 20081031 !) :
  - conception : famille de fns, sorties de 224, 256, 384, 512 bits
  - compatible avec les standards cryptographiques
  - sécurité : CR, OW, autres critères de résistance
  - efficace : au moins aussi rapide que SHA-256
- SHA-3 : gagnant : Keccak ; connu depuis le 2 octobre 2012. Utilise la construction en éponge
  - empreinte de taille variable SHA3-384, SHA3-512,...
  - effet avalanche : 1 bit d'entrée changé modifie en moyenne 50% de la sortie
  - atteint les bornes supérieures de sécurité

Objectifs des PKI

- **confidentialité :** message chiffré doit rester secret.
- **authentification :** assurance de l'authenticité (expéditeur/origine)
- **identification :** déclaration de son identité
- **intégrité :** message n'a pas été altéré durant la transmission
- **non répudiation :** l'expéditeur ne peut nier l'envoi du message

Techniques utilisées

- **signature :** moyen d'associer l'expéditeur à un message
- **certificat :** attestation (d'un tiers) confirmant une affirmation (d'identité)
- **tiers de confiance :** autorité qui délivre les certificats
- **estampillage :** ajout dates ou jetons → unicité du message

Où se tenir au courant ?

Tous les standards sont enregistrés dans les PKCS [1] qui se trouvent à `https://arxiv.org/abs/1207.5446`

> *These standards cover RSA encryption, Diffie-Hellman key agreement, password-based encryption, extended-certificate syntax, cryptographic message syntax, private-key information syntax, and certification request syntax, as well as selected attributes.*

1. Public Key Cryptographic Standards

Signatures

But traditionnel de la crypto : assurer la confidentialité.
Autre application : les signatures introduites par Diffie et Hellman.
But des signatures : garantir **intégrité** et **authentification**.
Signature dépend de l'id. du signataire et du contenu du message.

Signature empêche deux types de fraudes :

- la falsification de la signature par le destinataire ;
- la non-reconnaissance du message par l'expéditeur.

Utilisation électronique légale depuis la loi 2000-230 du 13/3/2000 Art.3 :L'écrit sur support électronique a la même force probante que l'écrit sur support papier

Cahier des charges

Le cahier des charges d'une signature $\Sigma(m)$ est :

- elle doit être calculable par le signataire pour tout message m ;
- tout individu (surtout le destinataire) peut la vérifier ;
- elle doit être impossible à falsifier ;
- l'expéditeur ne doit pouvoir affirmer que sa signature a été imitée.

Mécanisme général de signature Σ

Une signature est composée de 3 algos PPT :

- génération de clés noté **gen** (pk, sk) fonction de 1^n
- signature (privée) noté **sig** qui, pour une clé fixée sk, retourne une signature s pour un clair m ;

$$\text{sig}_{sk}(m) = s$$

- vérification (déterministe et publique) noté **ver** qui, à une clé fixée pk et pour tout couple clair/signature (m, s) va vérifier si la signature correspond bien au clair.

$$\text{ver}_{pk}(m, s) = \begin{cases} \text{vrai si } s = \text{sig}_{sk}(m) \\ \text{faux si } s \neq \text{sig}_{sk}(m) \end{cases}$$

Expérience Sig-forge$_{A,\Sigma}(n)$

Formalise l'attaque du faussaire (Adversaire change m et calcule une signature valide en usurpant l'identité de Bob).

L'expérience Sig-forge$_{A,\Sigma}(n)$:

1. Gen(1^n) produit (pk, sk)
2. A reçoit pk et l'accès à un oracle $\text{Sig}_{sk}(.)$[2]. A renvoie (m, s). Soit Q l'ensemble des messages pour lesquels A a eu recours à l'oracle
3. A réussit l'expérience (i.e. renvoie 1) ssi
   1. $\text{ver}_{pk}(m, s) = 1$ et
   2. $m \notin Q$

2. l'oracle fournit $\text{sig}_{sk}(m)$ à tout m choisi par A

Sécurité des signatures Sig-forge$_{A,\Sigma}(n)$

Définition

Un mécanisme de signature Σ est existentiellement infalsifiable pour une attaque adaptative à messages choisis si pour tout adversaire A PPT, il existe negl tq :

$$Pr(\textit{Sig-forge}_{A,\Sigma}(n) = 1) \leq \textit{negl}(n)$$

Signer avec RSA

Bob désire envoyer un message M signé à Alice. Paramètres RSA :

	Privés	Publics
Alice	d_A	n_A, e_A
Bob	d_B	n_B, e_B

Procédé de signature :

$$\text{sig}_{sk}(M) = M^{d_B} \mod n_B = S$$

Vérification :

$$\text{ver}_{pk}(M, S) = \text{vrai} \Leftrightarrow S^{e_B} \mod n_B \equiv M$$

Exemple d'envoi d'un message secret signé (RSAKE)

Comment Bob peut-il envoyer à Alice un message secret signé ?
Fonctions de chiffrement et de déchiffrement d'Alice et Bob :

	Privés	Publics
Alice	$D_A(C) = C^{d_A} \mod n_A$	$E_A(M) = M^{e_A} \mod n_A$
Bob	$D_B(C) = C^{d_B} \mod n_B$	$E_B(M) = M^{e_B} \mod n_B$

Bob envoie le message $C = E_A(D_B(M))$
Et Alice le déchiffre en $E_B(D_A(C))$
Pour cela, il faut que $M < n_B < n_A$.

Falsification sans message

Faussaire reçoit $pk = (n, e)$, choisit $s \in \mathbb{Z}_n^\star$ calcule $m := s^e \mod n$

(m, s) valide qui n'a pas été signé par le propriétaire de sk !

Faussaire n'a pas le choix de m mais... que se passe-t-il dans le cas d'une authentification par défi ?

Falsification d'un message choisi

Si le faussaire obtient 2 signatures du propriétaire de sk il peut signer un message m de son choix :

- $m_1 \xleftarrow{u} \mathbb{Z}_n^\star$
- $m_2 := m.m_1^{-1} \mod n$
- obtient s_1 et s_2, signatures de m_1 et m_2
- $s := s_1 s_2 \mod n$ signe m !

Vous acceptez de signer n'importe quoi ?
Et vous savez si votre protocole favori signe ce qu'on lui présente ?

C'est plus sûr en hachant !

- En remplaçant m par $h(m)$, la signature par RSA devient plus sûre (hashed RSA signature scheme).
- Il faut h résistante aux collisions, sinon, on trouve $m_1 \neq m_2$ avec $h(m_1) = h(m_2)$ qui donnent les mêmes signatures.
- Les 2 attaques précédentes échouent (du fait de CR de h)
- Utilisé en pratique ; sous certaines hypothèses, prouvé sûr avec SHA-1.

Signature par El Gamal

Soit p un nombre premier pour lequel DLP est difficile dans $\mathbb{Z}_p^\times$ et α un générateur de $\mathbb{Z}_p^\times$. Le message $M \in \mathbb{Z}_p^\times$ et sa signature est $(M, S) \in \mathbb{Z}_p^\times \times \mathbb{Z}_p^\times \times \mathbb{Z}_{p-1}$. L'ensemble des clés est $K = \{(p, \alpha, a, \beta) : \beta = \alpha^a \mod p\}$

Secrets	Publics
a	p, α, β

On choisit $k \in \mathbb{Z}_{p-1}^\times$ aléatoire et secret qui vérifie $\gcd(k, p-1) = 1$
On définit une signature comme :

$$\text{sig}_K(M, k) = (\gamma, \delta)$$

pour

$$\gamma = \alpha^k \mod p \qquad \delta = (M - a\gamma)k^{-1} \mod (p-1)$$

Exemple

Soit $p = 467$ et $a = 127$. On a bien que $\gcd(a, p-1) = 1$. Soit $\alpha = 2$ un générateur de $\mathbb{Z}_p^\times$. On calcule

$$\beta = \alpha^a \mod p = 2^{127} \mod 467 = 132$$

Si Bob veut signer le message $M = 100$ pour la valeur aléatoire $k = 213$ qui est tq $\gcd(k, p-1) = 1$, il calcule $k^{-1} \mod p-1$ par Euclide étendu qui donne $k^{-1} = 431$ alors,

$$\gamma = \alpha^k \mod p = 2^{213} \mod 467 = 29$$

et

$$\delta = (M - a\gamma)k^{-1} \mod (p-1) = (100 - 127.29).431 \mod 466 = 51$$

Fonctionnement – Vérification

Pour $M, \gamma \in \mathbb{Z}_p^\times$ et $\delta \in \mathbb{Z}_{p-1}$, on définit

$$\text{ver}_K(M, \gamma, \delta) = \text{vrai} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^M \mod p$$

Si la signature est construite correctement, la vérification authentifie la signature car :

$$\begin{aligned} \beta^\gamma \gamma^\delta &\equiv \alpha^{a\gamma} \alpha^{k\delta} \mod p \\ &\equiv \alpha^M \mod p \end{aligned}$$

en utilisant le fait que $a\gamma + k\delta \equiv M \mod (p-1)$

Exemple

On vérifie la signature de $(100, 29, 51)$ *par* $\text{ver}_K(M, \gamma, \delta) = \text{vrai}$ *:*

$$\beta^\gamma \gamma^\delta \equiv \alpha^M \mod p \Leftrightarrow 132^{29} 29^{51} \equiv 2^{100} \mod p \equiv 189$$

Digital Signature Standard DSA – 1991

- Variante d'El Gamal qui diminue la taille de la signature.
- DSS-DSA proposé en 1991 par D.W. Kravitz (NSA) ; adopté en 1993.
- Un module de 512 bits d'El Gamal donne une signature de 1024 bits. DSA : avoir une signature plus courte.
- Par une astuce, DSA raccourcit les tailles en offrant une signature de 320 bits sur un message de 160 bits en impliquant un module de 512 bits. Astuce : travailler dans un sous-groupe de $\mathbb{Z}_p^\times$ de taille 2^{160}.

Fonctionnement de DSS

Soit p un premier de 512 bits, q facteur premier de 160 bits de $p-1$ et α racine q^e primitive de 1 modulo p tq DLP dans le sous-groupe engendré par α est difficile. Message $h(M) \in \mathbb{Z}_p^\times$; signature : $(h(M), S) \in \mathbb{Z}_p^\times \times \mathbb{Z}_q \times \mathbb{Z}_q$. L'ensemble des clés est $K = \{(p, q, \alpha, a, \beta) : \beta = \alpha^a \mod p\}$

Privé	Publics
a	p, q, α, β

Choisir $1 < k \leq q-1$ aléatoire et secret ; la signature est :

$$\text{sig}_K(h(M), k) = (\gamma, \delta)$$

pour

$$\gamma = \alpha^k \mod q \quad \delta = (h(M) + a\gamma)k^{-1} \mod q$$

Exemple

On choisit $q = 101$ et $p = 78q + 1 = 7879$. Une racine primitive de $\mathbb{Z}_{7879}$ est 3 et on peut prendre

$$\alpha = 3^{78} \mod 7879 = 170$$

On suppose que $a = 75$, on a :

$$\beta = \alpha^a \mod 7879 = 4567$$

Bob souhaite signer M d'empreinte $h(M) = 1234$ et choisit comme valeur de $k = 50$. $k^{-1} \mod 101 = 99$. On a

$$\gamma = (170^{50} \mod 7879) \mod 101 = 2518 \mod 101 = 94$$

et

$$\delta = (1234 + 75.94)99 \mod 101 = 97$$

La signature du message d'empreinte 1234 est alors $(94, 97)$

Vérification par DSS

Pour $h(M) \in \mathbb{Z}_p^\times$ et $\gamma, \delta \in \mathbb{Z}_q$, $\text{ver}_K(h(M), \gamma, \delta) = \text{vrai} \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \mod p) \mod q = \gamma$

$$\text{pour} \begin{cases} e_1 = h(M)\delta^{-1} \mod q \\ e_2 = \gamma\delta^{-1} \mod q \end{cases}$$

NB : $\delta \not\equiv 0 \mod q$ car $\delta^{-1} \mod q$ est nécessaire à la vérification. Si on a $\delta \equiv 0 \mod q$, il faut choisir une nouvelle valeur de k.

Exemple

La signature de 1234 est $(94, 97)$ *; vérification :*

$\delta^{-1} = 97^{-1} \mod 101 = 25$
$e_1 = 1234.25 \mod 101 = 45; \quad e_2 = 94.25 \mod 101 = 27$

et $(170^{45}4567^{27} \mod 7879) \mod 101 = 2518 \mod 101 = 94$

Inconvénient

Procédé plus lent que RSA d'un facteur compris entre 10 et 40.
Générer des clés est plus rapide que pour RSA.
Autre inconvénient : clé de 512 bits trop petite. La taille des clés DSA passe à 1024 ou 2048 (et le sous-groupe à 224 ou 256 bits).
La fonction de hachage est aussi en train de changer dans FIPS 186-3 qui devra utiliser SHA-224/256/384/512.

Plan

Contenu d'un certificat (X.509)

Associe une clé publique à l'identité d'un **sujet** ; comprend :

- **Subject** : Nom Distingué, Clé publique
- **Issuer** : Nom Distingué, Signature
- **Period of Validity** : date de début, date de fin
- **Administrative Information** : version, numéro de série
- **Extended Information** :

où l'information « Nom Distingué » comprend les champs :

Common Name : nom à certifier	Bruno Martin
Organization— Company : contexte	UCA
Organizational Unit : contexte spécifique	I3S
City/Locality : ville	Sophia Antipolis
State/Province : pour US	PACA
Country : code pays	fr

Politique d'utilisation par AC

L'**autorité de certification** spécifie les champs obligatoires et optionnels et impose évent. cond. de validité sur des champs

Exemple

Vérif. requiert que le champ **Common Name** *d'un serveur corresponde au nom de domaine du serveur (univ-cotedazur.fr).*

Le certificat délivré par AC garantit la relation (Identité, pk) en vérifiant que l'identité du détenteur de la biclé (pk,sk) correspond bien à celui inscrit dans le certificat.
L'AC s'en assure lors de la requête de certificat.
Si Alice requiert un certificat personnel, l'AC doit tout d'abord vérifier qu'Alice est bien la personne requérante.

Paradoxe

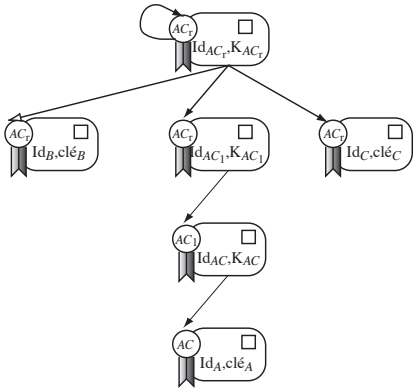
Comment connaît-on l'algorithme de vérification de l'autorité de certification ?

Quel est le modèle de confiance ?

- confiance directe
- confiance hiérarchique X509 (le plus utilisé)
- toile de confiance PGP (web of trust)

Chaîne de certification (confiance hiérarchique)

Une AC peut aussi fournir un certificat à une autre AC.
Bob remonte la chaîne de certification jusqu'à trouver une AC en qui il a confiance.

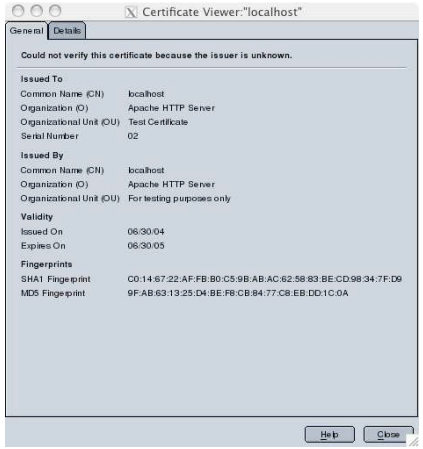


Mauvaise AC : attaques possibles (2011 : DigiNotar)

Création d'une AC « racine »

Problème : il faut une AC « racine » qui ne peut se faire certifier. Son certificat est auto-délivré. L'autorité de certification est identique au sujet certifié.
La confiance repose dans une large distribution de la *pk* de l'AC.
Les clients et les serveurs sont configurés pour faire confiance à certaines AC par défaut, comme `CertiSign` ou `VeriSign`.
Ces sociétés proposent des techniques de gestion des certificats, des procédures de vérification de l'information, délivrent des certificats.
Il est possible de se déclarer comme AC « racine », utile au sein d'un intranet, où la vérification de l'information est plus aisé.
Observons que la librairie `OpenSSL` délègue la confiance à l'OS.

Rupture dans la chaîne



Services de certification

Commercialisent, délivrent des certificats via procédure spécifique.
Gèrent des liste de révocation des certificats :

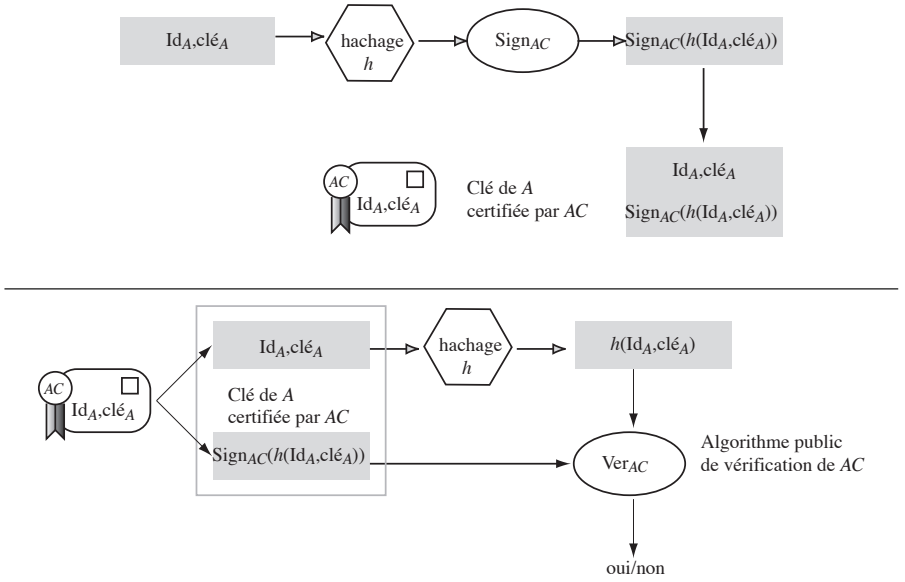
- liste des certificats corrompus ou invalides
- certificats cassés
- certificat d'un sujet obsolète (licenciement, panne serveur)

Réalisation « asymétrique » : au moyen d'un **schéma de signature avec appendice** qui consiste en [2] :

- une signature
- une vérification

sur le contenu d'un certificat.
Avec un contenu répondant à la norme X509, on fournit un **identificateur numérique** ou Digital ID, une "carte d'identité" numérique.

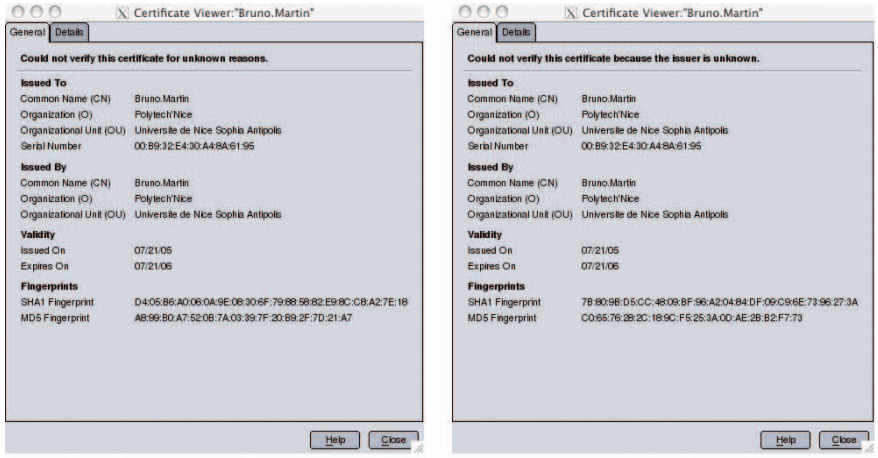
Certification & Vérification



Une attaque MIM de plus haut niveau

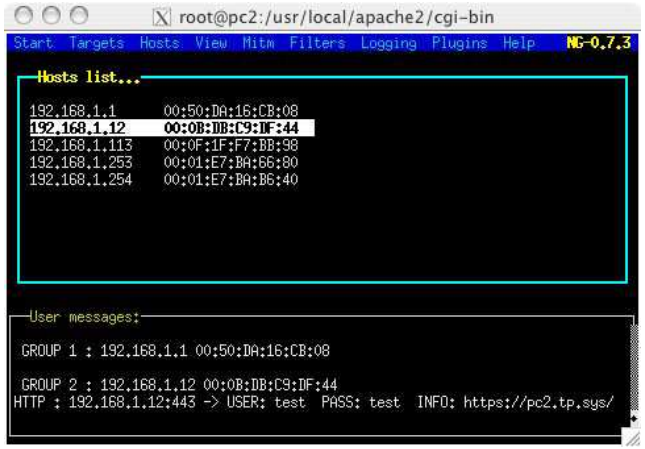
Porte sur la transmission d'un certificat dont l'AC n'est pas connue. C'est le cas, par exemple, d'un certificat "auto-délivré".

Certificat original et celui falsifié



Avec quel outil ?

Des logiciels libres comme `dsniff` ou `ettercap` permettent de mener à bien de telles attaques sur un LAN.



Attaque de l'homme du milieu en pratique sur un LAN

Hachage
Signatures
Certification

G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer.
Cryptographic sponge functions.
https://keccak.team/files/CSF-0.1.pdf, 2011.

RSA Laboratories.
PKCS ♯1 v2.0, RSA cryptography standard.
Technical report, RSA Data Security, 1998.

L. Lamport.
Password authentication with insecure communication.
Communications of the ACM, 24(11) :770–772, 1981.

Bruno MARTIN, Université Côte d'Azur Hachage, Signatures et Certification 69