Hachage et certification

Contenu

Hachage

Certification

Hachage

Signature utilisable seulement pour des petits messages.

Solution naïve : découper le message à signer en blocs (de 160 bits, voire plus) et signer chaque bloc.
Plusieurs problèmes :

- taille de la signature
- lenteur des algos de signatures

Solution par le hachage

Utiliser une **fonction de hachage cryptographique**, rapide à calculer, qui transforme un message de longueur arbitraire en une empreinte numérique de taille fixe. On signe l'empreinte :

message	m	longueur arbitraire
	$\downarrow$	
empreinte	$z = h(m)$	160 bits
	$\downarrow$	
signature	$s = \mathrm{sig}_{sk}(z)$	320 bits

Principe : Bob signe m :

- il calcule l'empreinte $z = h(m)$
- signe en $s = \mathrm{sig}_{sk}(z)$ et transmet (m, s).

N'importe qui vérifie (m, s) en recalculant l'empreinte $\hat{z} = h(m)$ puis en utilisant le procédé de vérification de $\mathrm{ver}_{pk}(\hat{z}, s)$.

Conditions à satisfaire

Une fonction de hachage h calcule

$$z = h(m)$$

pour m message de taille arbitraire et z, empreinte de taille fixe.
On veut que h soit être **à sens unique**, i.e.

- $h(m)$ doit être facile à calculer à partir de m
- z doit être difficile à inverser.

Collision : 2 mots distincts (x, x') tq $h(x) = h(x')$.
h est

- **faiblement résistante** aux collisions si, pour x fixé, il est difficile de calculer une collision.
- **fortement résistante** aux collisions s'il est difficile de calculer la moindre collision (x, x').

Paradoxe des anniversaires

Donnée : $B = (b_1, \ldots, b_k) \in \{1, 2, \ldots, n\}^k$.
Problème : proba p d'avoir ≥ 2 éléments de B identiques ?
On prend k messages m_i tirés aléatoirement avec $i \in [1, k]$ et on étudie la proba pour que deux m_i aient la même image.
$z_i = h(m_i)$. On cherche la valeur de k.
Complémentaire= Proba que les z_i soient tous différents :

$$1-p = q = \frac{1}{n^k} \prod_{i=0}^{k-1} (n-i) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right) = 1 \left(1 - \frac{1}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right)$$

Où 1 est la proba de tirer z_1, $(1 - \frac{1}{n})$ celle de tirer $z_2 \neq z_1$ (car il y a une chance sur n que $z_1 = z_2$),..., $(1 - \frac{i}{n})$ celle de tirer $z_{i+1} \neq z_1, \ldots, z_i$.

On approche $\prod_{i=1}^{k-1} (1 - \frac{i}{n})$ par $\prod_{i=1}^{k-1} e^{-\frac{i}{n}}$ car
$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \cdots$, donc $e^{-x} \approx 1 - x$ si x est petit (ce qui est notre cas). Donc $1 - \frac{i}{n} \approx e^{-\frac{i}{n}}$.

$$q = \prod_{i=1}^{k-1} (1 - \tfrac{i}{n}) \approx \prod_{i=1}^{k-1} e^{-\frac{i}{n}} = e^{-\frac{1}{n}\sum_{i=1}^{k-1} i} = e^{-\frac{(k-1)k}{2n}}$$
$$\ln q \approx -\frac{(k-1)k}{2n}$$
$$2n \ln(\tfrac{1}{q}) \approx k^2$$
$$\sqrt{2n \ln(\tfrac{1}{1-p})} \approx k$$

$p = 1/2$, proba d'avoir au moins une collision pour $k \approx \sqrt{2n \ln 2}$

Exemple : $n = 365$, $k = 23$ personnes ; on a plus d'une chance sur deux que deux personnes aient le même jour anniversaire.
Utilité : trouver la taille n de l'image par la fonction de hachage pour éviter les collisions. On a k en $O(\sqrt{n})$.

Attaque basée sur le paradoxe

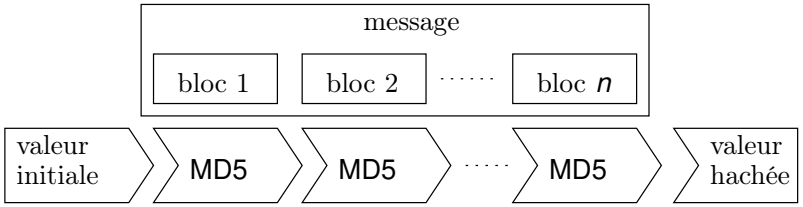
Calculer et trier autant de couples $(x, h(x))$ que possible. Détecter une (ou plusieurs) collisions.
Il y a 2^n valeurs qui correspondent aux «jours anniversaires» ; .
On suppose que les images par h suivent une distribution uniforme.
En considérant k entrées on a plus d'une chance sur deux d'avoir une collision avec $k \approx 2^{\frac{n}{2}}$. En passant au logarithme,

n	50	100	150	200
$\lfloor \log_2 k \rfloor$	25	50	75	100

Ainsi, en calculant un peu plus que $2^{n/2}$ images par h, on trouve une collision avec une probabilité $> 1/2$.
Pour h fortement résistante, on choisit n pour que le calcul de $2^{n/2}$ images par h soit irréaliste. A ce jour, $n \geq 160$.

Hachage par compression

Dans MD5, SHA, découper m en n blocs de longueur fixe puis :



Construire une fonction de hachage

Partir de e_k, chiffre sym., construire une fonction de compression

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

On utilise g pour construire une fonction de hachage :

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

Proposition

h est résistante aux collisions si g l'est aussi.

Fonction de compression

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

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

dont la taille de l'image par la fonction de hachage est n.
La fonction de chiffrement est utilisée soit directement si elle est résistante aux collisions soit en la «perturbant» :

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

Fonction de hachage

Merkle : construction de fonction de hachage à partir d'une fonction de compression $g : \{0,1\}^m \to \{0,1\}^n$.
Soit $r = m - n > 1$. On veut construire $h : \{0,1\}^\star \to \{0,1\}^n$.
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 : $r = 4$, $x = 11101$, $\ell = 101$. $u = \mathbf{000}1\,1101$, $v = \mathbf{1}101$.

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

H définie ind. : $H_0 = 0^n$ et $H_i = g(H_{i-1} w_i)$, $1 \le i \le t$

$$h(x) = H_t$$

Fonctions de hachage efficaces

Les fonctions de hachages utilisées en pratique sont construites comme précédemment. Les plus courantes :

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

Application à DSA

Digital Signature Algorithm : standard de signature qui combine une fonction de hachage (SHA) et DSS, qui améliore le schéma de signature d'El Gamal (en travaillant dans un sous-groupe).

Plan de l'exposé

Hachage

Certification

Certificat de clé publique

Un certificat de clé publique de B garantit la relation entre Id_B et $\underline{\mathrm{pk}}_B$, signée par un tiers de confiance, Ivan.
Utilité : déjouer l'attaque de l'homme du milieu
Un certificat de la clé publique de B contient

- sa clé publique
- des infos concernant B (nom, e-mail...)
- la signature d'un tiers de confiance Ivan

Ivan signe à la fois

- la clé
- les infos concernant B

Ivan certifie la relation entre la clé publique et l'identité de B.

Réalisation «asymétrique»

Certification réalisée au moyen d'un **schéma de signature**. Celui-ci consiste en [1] :

- une signature après hachage cryptographique
- une opération de vérification correspondant à la signature.

Exemple : si le contenu du message est celui décrit par la norme X509, on fournit de cette manière un **identificateur numérique** ou Digital ID, sorte de carte d'identité numérique.

Certificat (X.509)

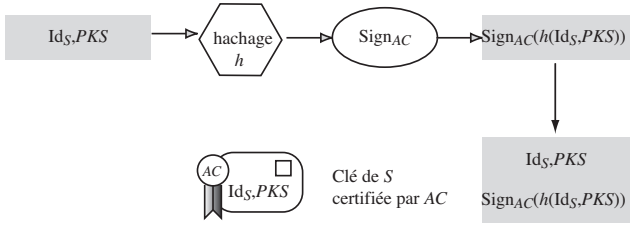
Associe une clé publique à l'identité d'un **sujet** ; comprend :

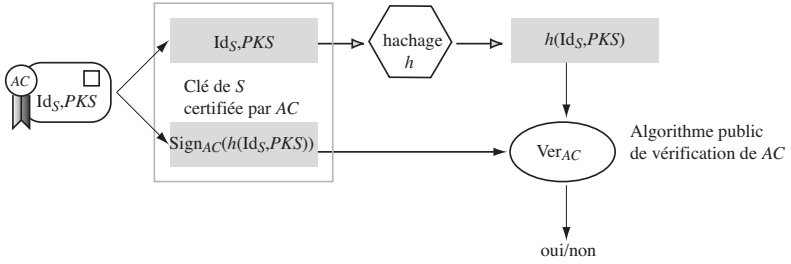
- **Subject** : Nom Distingué, Clé publique
- **Issuer** : Nom Distingué, Signature
- **Period of Validity** : date de début, date de fin
- **Administrative Information** : version, numéro de série
- **Extended Information** :

L'information « Nom Distingué » comprend :

- **Common Name** : nom à certifier — Bruno Martin
- **Organization Company** : contexte — U. Côte d'Azur
- **Organizational Unit** : contexte spécifique — UPinfo
- **City/Locality** : ville — Sophia Antipolis
- **State/Province** : pour US — PACA
- **Country** : code pays — fr

Certification & Vérification





Paradoxe

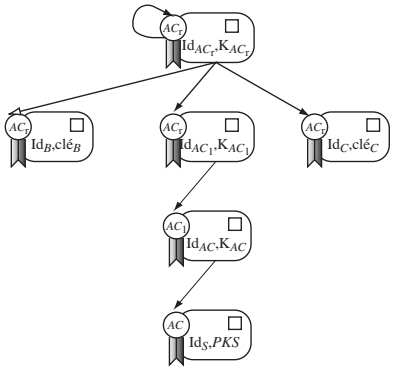
Comment connaît-on l'algorithme de vérification de l'autorité de certification ?

Quel est le modèle de confiance ?

- confiance directe (entre 2 entités, par transmission directe)
- confiance hiérarchique (le plus utilisé)
- toile de confiance (web of trust)

Chaîne de certification (hiérarchique)

AC peut aussi fournir un certificat à une autre AC. Alice peut ainsi remonter une chaîne de certification jusqu'à trouver une AC en qui elle a confiance.



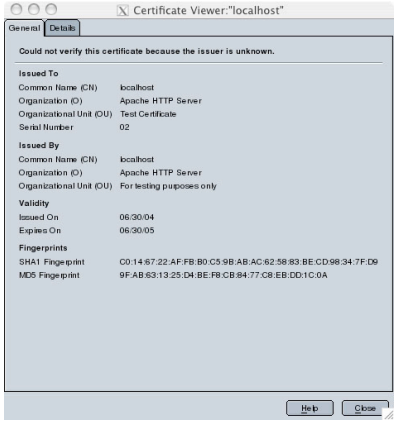
Mauvaise AC : attaques possibles (2011 : DigiNotar)

Création d'une AC «racine»

Problème des chaînes de certification : il faut une AC «racine». Celle-ci ne peut se faire certifier. Dans ce cas, le certificat est auto-signé par l'AC racine. L'entité qui délivre le certificat est identique au sujet certifié. La confiance réside dans une large distribution de la clé publique de l'AC.

Les systèmes d'exploitation sont configurés pour faire confiance à certaines AC par défaut, comme `CertiSign` ou `VeriSign` utilisés par des autorités intermédiaires comme `Let's Encrypt`.
Ces sociétés proposent des techniques pour demander des certificats, ont des procédures de vérification de l'information, délivrent et gèrent des certificats.

Rupture dans la chaîne



Toile de confiance

Tentative de concilier les modes directs et hiérarchiques.
La confiance est accordée par l'utilisateur selon le principe que plus on dispose d'information, meilleure est la confiance. On accorde sa confiance directement ou au moyen d'une chaîne de certification qui remonte jusqu'à un tiers connu selon le principe :

- Jean signe l'identifiant Paul dans le certificat de Paul ;
- lorsque Marie veut vérifier que Jean a signé Paul, elle utilise la clé publique de Jean pour vérifier sa signature dans le certificat de Paul ;
- Marie fait confiance à Jean. Elle reçoit un message signé de Paul. Elle vérifie la validité de la signature de Paul (clé récupérée depuis un serveur de clés). Comme elle fait pleinement confiance à Jean, elle valide de plus le fait que c'est bien Paul qui a signé ce message.

C'est le système utilisé par `OpenPGP`, `GnuPG` ou `PGP`.

Bibliographie


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