

Fonction de hachage et signatures électroniques

Christophe Chabot

Université de Limoges, XLIM-DMI,
123, Av. Albert Thomas
87060 Limoges Cedex France
05.55.45.72.77
christophe.chabot@unilim.fr

Licence professionnelle Administrateur de Réseaux
et de Bases de Données
IUT Limoges



Sommaire

Fonction de hachage

Signatures numériques

Notion de fonction de hachage

- ▶ **Définition (Fonction de Hachage) :**

Une fonction de hachage H est une application facilement calculable qui transforme une chaîne binaire de taille quelconque t en une chaîne binaire de taille fixe n , appelée empreinte de hachage.

- ▶ On parle de *collision* entre x et x' lorsque

$$x \neq x' \text{ et } H(x) = H(x')$$

- ▶ Si y est tel que $y = H(x)$, alors x est appelé préimage de y

Qu'est-ce qu'une fonction de hachage ?

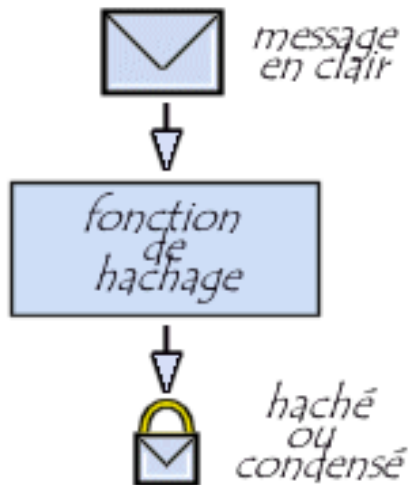


Fig.: www.infoclick.fr/ccm/crypto/signature.htm

Un exemple de fonction de hachage en action

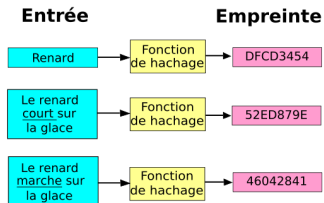


Fig.: fr.wikipedia.org/wiki/Fonction_de_hachage

Intérêt cryptographique

- ▶ Primitive cryptographique très importante !
 - ▶ Schémas de signature (PSS)
 - ▶ Schémas de chiffrement (OAEP)
 - ▶ Intégrité de documents
 - ▶ Message Authentication Code (MACs)
- ▶ Facilement calculable
- ▶ Compression

Propriétés des fonctions de hachage

- ▶ **Propriétés de base** : compression et facilité de calcul.
- ▶ **Propriétés additionnelles** :
 - ▶ Résistance à la préimage
 - ▶ étant donné y , il est difficile de trouver x tel que $y = H(x)$
 - ▶ Résistance à la seconde préimage
 - ▶ étant donné x , il est difficile de trouver $x' \neq x$ tel que $H(x) = H(x')$

Propriétés des fonctions de hachage

- ▶ Résistance à la collision
 - ▶ il est difficile de trouver x et x' tels que $H(x) = H(x')$.
- ▶ Fonction de Hachage à Sens Unique
 - ▶ résistance à la préimage et à la seconde préimage
- ▶ Fonction de Hachage résistante aux collisions
 - ▶ résistance à la seconde préimage et à la collision

Paradoxe des anniversaires

Dans une assemblée de 23 personnes, la probabilité qu'au-moins 2 d'entre-elles aient leur anniversaire le même jour est égale à $\frac{1}{2}$. (en ne tenant pas compte de l'année de naissance.)

Paradoxe des anniversaires

Problème sous-jacent :

- ▶ Fonction de hachage $H : \{0, 1\}^t \rightarrow \{0, 1\}^m$ avec $t > m$.
- ▶ On pose : $n = 2^m = |\{0, 1\}^m|$
- ▶ k messages $x_{i_{1 \leq i \leq k}}$ aléatoires ($k \ll n$)

Question : quelles est la probabilité pour obtenir une collision à partir des messages x_i ?

Paradoxe des anniversaires (2)

- ▶ **Hypothèse** : les $z_i = H(x_i)$ sont aléatoires
- ▶ **But** : calculer $p_k = \{ \text{prob que les } \{z_i\}_{1 \leq i \leq k} \text{ soient tous } \neq \}$
 - ▶ $k = 2$: $p_2 = \frac{n-1}{n}$ ($n - 1$ possibilités dans le choix de z_2)
 - ▶ $k = 3$: $p_3 = \frac{n-1}{n} \frac{n-2}{n}$
 - ▶ ...
 - ▶ $p_k = \frac{n-1}{n} \frac{n-2}{n} \dots \frac{n-k+1}{n}$

$$p_k = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right)$$

⇒ prob qu'il y ait au moins une collision :

$$1 - p_k = 1 - \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right)$$

- ▶ Comment simplifier pour déterminer k tel que $1 - p_k > \frac{1}{2}$?

L'astuce qui tue

$$e^{-x} = 1 - x + \sum_{j=2}^{\infty} (-1)^j \frac{x^j}{j!} \quad \forall x \in \mathbb{R}, e^{-x} \geq 1 - x$$

$$\Rightarrow \forall i \in [1, k-1], e^{-\frac{i}{n}} \geq 1 - \frac{i}{n}$$

$$\begin{aligned} \prod_{i=1}^{k-1} (e^{-\frac{i}{n}}) \geq \prod_{i=1}^{k-1} (1 - \frac{i}{n}) &\equiv e^{-\frac{(k-1)k}{2n}} \geq p_k \\ &\equiv 1 - p_k \geq 1 - e^{-\frac{(k-1)k}{2n}} \end{aligned}$$

Bilan : pour avoir $1 - p_k > \alpha$, il suffit d'avoir $1 - e^{-\frac{(k-1)k}{2n}} > \alpha$

Notions vues en terminale S

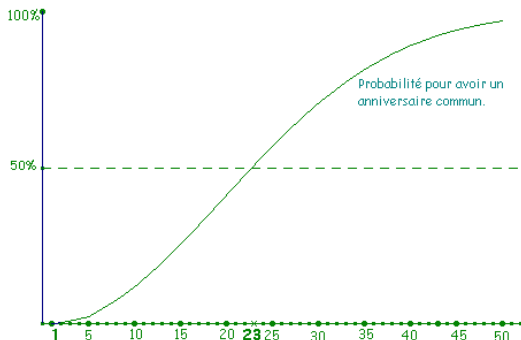
Pour $0 < \alpha < 1$ et $k > 1$:

$$\begin{aligned}
 1 - e^{-\frac{(k-1)k}{2n}} > \alpha &\equiv e^{-\frac{(k-1)k}{2n}} < 1 - \alpha \\
 &\equiv -\frac{(k-1)k}{2n} < \ln(1 - \alpha) \\
 &\equiv \frac{(k-1)k}{2n} > \ln\left(\frac{1}{1-\alpha}\right) \\
 \Rightarrow k^2 > (k-1)k > 2n \ln\left(\frac{1}{1-\alpha}\right)
 \end{aligned}$$

Bilan : $1 - pk > \alpha \Rightarrow k > \sqrt{2n \ln\left(\frac{1}{1-\alpha}\right)} = \mathcal{O}(\sqrt{n})$

- ▶ $n = 365, \alpha = \frac{1}{2} \Rightarrow k > 22,49 : k = 23$
- ▶ $n = 365, \alpha = 99\% \Rightarrow k > 57,98 : k = 58$

Probabilité pour avoir un anniversaire commun



Ce qu'il faut retenir du paradoxe des anniversaires

Concernant les fonctions de hachage

- ▶ Soit H la fonction de hachage $H : \{0, 1\}^* \rightarrow \{0, 1\}^m$
- ▶ La recherche brutale de collisions a plus d'une chance sur 2 d'aboutir après seulement $\mathcal{O}(2^{\frac{m}{2}})$ essais !
- ▶ **Remarque** : on est **sûr** d'aboutir après $\mathcal{O}(2^m)$ essais

Concernant les algorithmes de chiffrement

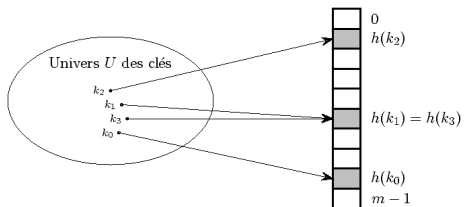
- ▶ Soit E un algorithme de chiffrement et $K \in \{0, 1\}^m$ une clé
- ▶ La recherche brutale de la clé dans le chiffrement $E_K(M)$ par une attaque à texte clair connu a plus d'une chance sur 2 d'aboutir après seulement $\mathcal{O}(2^{\frac{m}{2}})$ essais !
- ▶ **Remarque** : on est **sûr** d'aboutir après $\mathcal{O}(2^m)$ essais

Attaque de Yuval (attaque anniversaire)

Soit $H : \{0, 1\}^* \rightarrow \{0, 1\}^m$ une fonction de hachage.

- ▶ **Input** : M message initial ; \tilde{M} message corrompu
- ▶ **Output** : $M' \approx M$ et $\tilde{M}' \approx \tilde{M}$ tels que $H(M') = H(\tilde{M}')$
 1. générer $2^{\frac{m}{2}}$ modifications mineures de M notées M'
 - ▶ pour chacune, calculer $H(M')$ et les stocker dans une table T
 2. générer des \tilde{M}' , modifications mineures de \tilde{M} jusqu'à ce qu'un $H(\tilde{M}')$ soit dans T (collision)
- ▶ **Application** : répudiation
 - ▶ envoyer M et soutenir avoir envoyé \tilde{M}

Table de hachage



Historique des principales fonctions de hachage

- ▶ 1990 : MD4, Extended-MD4
- ▶ 1991 : **MD5**
- ▶ 1992 : HAVAL, RIPEMD
- ▶ 1993 : SHA
- ▶ 1994 : **SHA-1**
- ▶ 1996 : RIPEMD-128,160
- ▶ 1999 : Whirlpool-0
- ▶ 2002 : **SHA-256**,384,512, Whirlpool-T
- ▶ 2003 : SHA-224
- ▶ 2004 : **Whirlpool**

La fonction de compression de MD5

- ▶ Proposé par Rivest en 1991
- ▶ Blocs de $b = 512$ bits, sortie de $n = 128$ bits
 - ▶ 16 sous-blocs M_i de 32 bits
 - ▶ 64 constantes fixées K_i
- ▶ 64 rondes sur 4 sous-blocs de 32 bits A, B, C, D
 - ▶ 4×16 sous-rondes où F prend les valeurs :
 1. $F = (B \text{ AND } C) \text{ OR } (\overline{B} \text{ AND } D)$
 2. $F = (D \text{ AND } B) \text{ OR } (\overline{D} \text{ AND } C)$
 3. $F = B \oplus C \oplus D$
 4. $F = C \oplus (B \text{ OR } \overline{D})$
- ▶ Sécurité de MD5 face aux collisions
 - ▶ Force brute : $\mathcal{O}(2^{64})$ (attaque anniversaire)
 - ▶ MAIS collisions en $\mathcal{O}(2^{42})$ [Wang04] puis $\mathcal{O}(2^{30})$ [Sasaki05]
 - ▶ MD5 n'est plus considérée comme sûr aujourd'hui

MD5

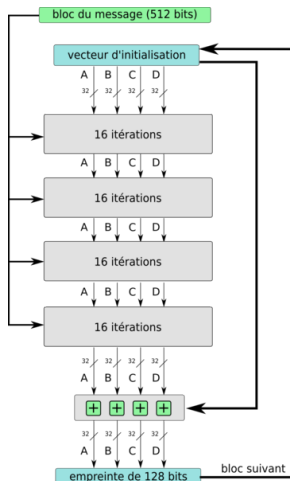


Fig.: <http://fr.wikipedia.org/wiki/MD5>

La fonction de compression de MD5

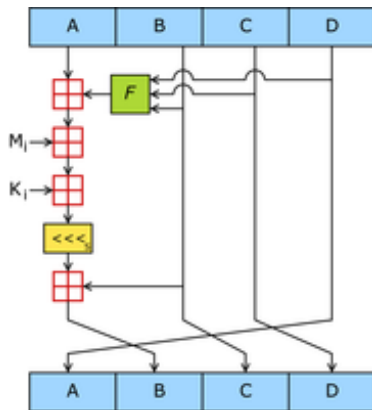


Fig.: Une opération de MD5

La fonction de compression de SHA-1

- ▶ Publié dans FIPS PUB 180-1 par NIST en 1995
- ▶ Blocs de $b = 512$ bits, sortie de $n = 160$ bits
 - ▶ 16 sous-blocs M_i de 32 bits
 - ▶ Etendus en 80 nouveaux blocs W_t
 - ▶ 4 constantes fixées K_t
 - ▶ 80 rondes sur 5 sous-blocs de 32 bits A,B,C,D,E
 - ▶ 4×20 sous-rondes où F prend les valeurs :
 1. $F = (B \text{ AND } C) \text{ OR } (\overline{B} \text{ AND } D)$
 2. $F = B \oplus C \oplus D$
 3. $F = (B \text{ AND } C) \oplus (B \text{ AND } D) \oplus (C \text{ AND } D)$
 4. $F = B \oplus C \oplus D$
- ▶ Sécurité de SHA-1 face aux collisions
 - ▶ Force brute : $\mathcal{O}(2^{80})$ (attaque anniversaire)
 - ▶ MAIS collisions possibles en $\mathcal{O}(2^{63})$ [Wang05]
 - ▶ Encore largement utilisé (signature, prot. anti-copie XBOX)

La fonction de compression de SHA-1

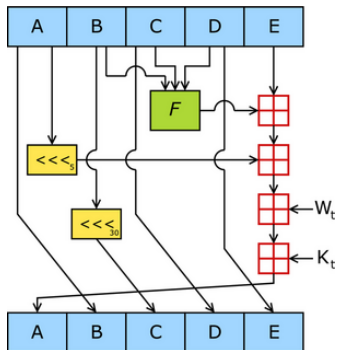


Fig.: La fonction de compression de SHA-1

La fonction de compression de SHA-256

- ▶ Publié dans FIPS PUB 180-2 en 2000
- ▶ Basé sur le chiffrement à clé secrète par bloc SHACAL-2
- ▶ Blocs de $b = 512$ bits, sortie de $n = 256$ bits
 - ▶ 16 sous-blocs M_i de 32 bits
 - ▶ Etendus en 64 nouveaux blocs W_t
 - ▶ 64 constantes fixées K_t
- ▶ 64 rondes sur 8 sous-blocs de 32 bits
- ▶ Sécurité de SHA-256
 - ▶ Force brute : $\mathcal{O}(2^{128})$ (attaque anniversaire)
 - ▶ Pas d'attaque connues pour le moment
 - ▶ (to be continued)

La fonction de compression de SHA-256

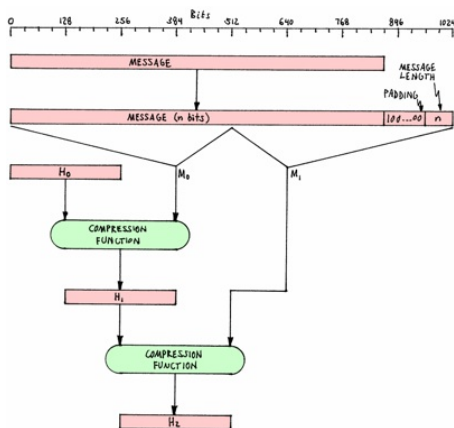


Fig.: La fonction de compression de SHA-256

Whirlpool

- ▶ Fonction de hachage recommandée par NNESSIE (2004)
- ▶ Basé sur la construction de Miyaguchi-Preneel

	Rijndael	Whirlpool
Taille Bloc	128, 160, 192, 224 ou 256	512
Nb rondes	10, 11, 12, 13 ou 14	10
Polynôme de réduction sur \mathbb{F}_{256}	$x^8 + x^4 + x^3 + x + 1$ (0x11B)	$x^8 + x^4 + x^3 + x^2 + 1$ (0x11D)
Orig. de la S-Box	$t : a \rightarrow a^{-1}$ sur \mathbb{F}_{256}	structure récursive
Orig. des const. de ronde	polynômes x^i sur \mathbb{F}_{256}	entrées successives de la S-Box

Whirlpool

- ▶ Blocs de $b = 512$ bits, sortie de $n = 512$ bits
- ▶ 10 rondes
- ▶ Sécurité de Whirlpool
 - ▶ Force brute : $\mathcal{O}(2^{256})$ (attaque anniversaire)
 - ▶ Pas d'attaques connues pour le moment
 - ▶ (to be continued)

Exemple de hachage avec Whirlpool

 **Whirlpool**

Fox

86A502DD96CEA5CC B967FDEC8E91C3F5
 FD4F4A677B6419AF 25002E1BE8194C0E
 260B7B9BEC8F0C91 5CDE385F84011F54
 95C6DBC5A56ACC26 BAD697888BD88205

The quick brown fox jumps over the lazy dog

B97DE512E91E3828 B40D2B0FDCE9CEB3
 C4A71F9BEA8D88E7 5C4FA854DF36725F
 D2B52EB6544EDCAC D6F8BEDDFEA403CB
 55AE31F03AD62A5E F54E42EE82C3FB35

The quick brown fox jumps over the lazy cog

DCE81FC695CFEA3D 7E1446509238DAF8
 9F24CC61896F2D26 5927DAA70F2108F8
 902F0DFD68BE085D 5ABB9FCD2E482C1D
 C24F2FABF81F40B7 3495CAD44D7360D3

Fig.: en.wikipedia.org/wiki/Image:Whirlpool_hash.svg

Bilan sur les fonctions de hachage connues

Fonction	Empreinte	Compl. requise	Rés. aux coll.	Comp. attaque
MD5	128 bits	$\mathcal{O}(2^{64})$	Cassé ¹	$\mathcal{O}(2^{30})$
SHA-1	160 bits	$\mathcal{O}(2^{80})$	Cassé ²	$\mathcal{O}(2^{63})$
HAVAL	256 bits	$\mathcal{O}(2^{128})$	Cassé ³	$\mathcal{O}(2^{10})$
SHA-256	256 bits	$\mathcal{O}(2^{128})$	Sûr	
Whirlpool	512 bits	$\mathcal{O}(2^{256})$	Sûr	

¹Sasaki&al.05

²Wang&al. Crypto 05

³Asiacrypt 04

Bilan sur les fonctions de hachage connues

- ▶ La résistance aux collision n'est pas toujours suffisante
- ▶ D'autres propriétés de sécurité peuvent être requises
 - ▶ Résistance aux collisions proches
 - Résistance aux collisions (x, x') ou $d(H(x), H(x'))$ faible
 - ▶ Résistance aux pseudo-collisions
 - Résistance aux collisions lorsque des IV peuvent différer
 - ▶ Pseudo-randomness
 - rendre difficile distinction entre H et une fonction aléatoire

Construction d'une fonction de hachage

- ▶ Définir une fonction de compression h
- ▶ Pour calculer l'empreinte d'un message M :
 - ▶ Application d'un padding à M pour que $|M| = k.b$
 - ▶ Découpage du message M en blocs de tailles b
 $M = M_1M_2...M_{k-1}M_k$ avec $|M_i| = b \forall i \in [1, k]$
 - ▶ Itération de la fonction h (IV : Initial Value) :
- ▶ Exemples connus : MD5, SHA-1, SHA-2, Whirlpool...

Signatures numériques

Idée générale des signatures électroniques

But des signatures manuscrites :

- ▶ prouver l'identité de leur auteur et/ou
- ▶ l'accord du signataire avec le contenu du document

La signature électronique dépend du signataire et du document/

Objectifs d'une signature électronique

- ▶ Une signature est authentique.
- ▶ Une signature ne peut être falsifiée (imitée).
- ▶ Une signature n'est pas réutilisable sur un autre document.
- ▶ Un document signé est inaltérable.
- ▶ Une signature ne peut pas être reniée.

Idée générale des signatures électroniques (2)

- ▶ Réalisation pratique :
 - ▶ Cryptosystèmes à clef secrète (et arbitre)
 - ▶ Cryptosystèmes à clef publique + fonction de hachage
- ▶ On préfère signer le hachage d'un document
 - ▶ Taille fixe suffisamment petite pour être utilisée efficacement par un cryptosystème à clé publique

Idée générale des signatures électroniques (3)

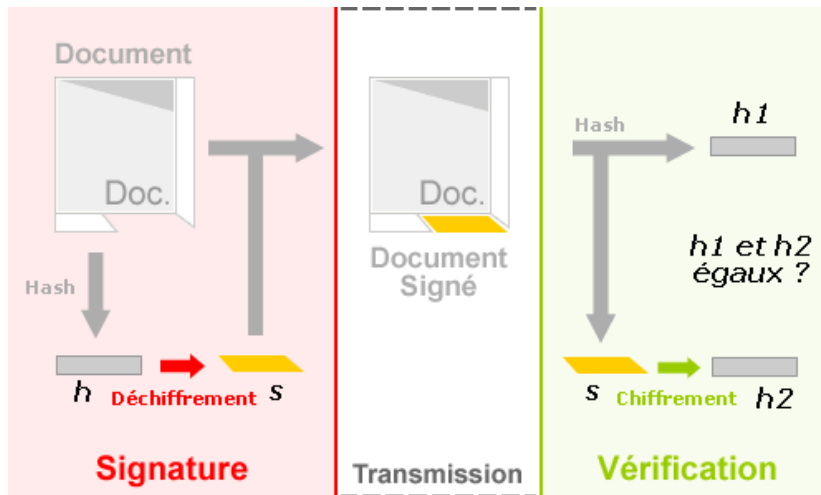
- ▶ Protocole de signature électronique sûr :
 - ▶ Impossible de falsifier la signature $s(M)$ d'un document M
 - ▶ sans connaître la clef secrète K (resp. K_d)
 - ▶ même en disposant de signatures d'autres documents.
 - ▶ Attention : impossibilité pratique
- ▶ Il existe d'autres conditions nécessaires de sécurité !
 - ▶ relève davantage des architectures de sécurité des cryptosystèmes à clef publiques (PKI) ou du secret entourant la clé secrète.

Idee générale des signatures électroniques (4)

Signature utilisant un cryptosystème à clef publique :

- ▶ Alice signe M en utilisant :
 - ▶ $h_M = H(M)$ le hachage de M
 - ▶ sa clé secrète K_d .
 - ▶ la fonction de déchiffrement D .
 - ▶ Résultat : $s(M) = D_{K_d}(h_M)$
- ▶ Document signé : $[M, s(M)]$
- ▶ Vérification de $[M, s(M)]$:
 - ▶ utilise la clé publique K_e d'Alice et la fonction de chiffrement E
 - ▶ $E_{K_e}(s(M)) = h_M \stackrel{?}{=} H(M)$
 - ▶ Seule Alice a pu générer $s(M)$

Idée générale des signatures électroniques



Signature RSA

Génération des paramètres : Identique à la génération des clefs de RSA

- ▶ Alice choisit au hasard deux nombres premiers p et q .
 - ▶ Alice calcule $n = p \cdot q$
 - ▶ Indicatrice d'Euler : $\phi(n) = (p - 1)(q - 1)$
- ▶ Alice choisit au hasard un entier e (impair) tel que $1 < e < \phi(n)$ et $\text{pgcd}(e, \phi(n)) = 1$
- ▶ Alice calcule alors l'entier d tel que $e \cdot d = 1 \pmod{\phi(n)}$.

Clef publique : (n, e)

Clef secrète : d

On suppose disposer d'une fonction de hachage à sens unique H connue publiquement.

Signature RSA (2)

Génération d'une signature RSA

Alice souhaite signer un document M

- ▶ Alice calcule $h_M = H(M)$ (on suppose $0 \leq h_M < n$)
- ▶ Signature de M : $s(M) = (h_M)^d \pmod n$
- ▶ Le document signé est alors $[M, s(M)]$.

Signature RSA (3)

Vérification d'une signature RSA

- ▶ Bob reçoit un document signé $[M', s(M)]$ d'Alice.
Ce document est potentiellement altéré/illégitime
- ▶ Il récupère la clé publique d'Alice (n, e)
- ▶ Il calcule $h_{M'} = H(M')$
- ▶ Il vérifie l'identité : $s(M)^e = h_{M'} \pmod n$
En effet : $s(M)^e = (h_M)^{e \cdot d} \pmod n = h_M \pmod n = h_M$ et si le document est authentique : $h_M = h_{M'}$.
- ▶ La sécurité est donc celle du cryptosystème RSA.
- ▶ Présentation simpliste et en l'état sujette à des attaques

Signature El Gamal

Génération des paramètres

- ▶ Alice choisit :
 - ▶ un nombre premier p
 - ▶ g une racine primitive modulo p .
 - ▶ un entier $a \in \{1, \dots, p - 2\}$ au hasard
- ▶ Elle calcule alors $A = g^a \pmod{p}$.

Clef publique : (p, g, A) .

Clef secrète : a .

On suppose disposer d'une fonction de hachage à sens unique H connue publiquement.

Signature El Gamal (2)

Génération d'une signature El Gamal

- ▶ Alice souhaite signer un document M
- ▶ Alice calcule $h_M = H(M)$ (on suppose $0 \leq h_M < p$)
- ▶ Elle choisit au hasard un entier $k \in [1, p-2]$ tel que $\text{pgcd}(k, p-1) = 1$ ($\Rightarrow k-1 \in \mathbb{Z}_{p-1}$ existe).
- ▶ Signature de M : $s(M) = (r, s)$ avec $r = g^k \pmod p$ et $s = k^{-1}(h_M - a.r) \pmod{(p-1)}$
- ▶ Le document signé est alors $[M, s(M)]$.

Signature El Gamal (3)

Vérification d'une signature El Gamal

- ▶ Bob reçoit un document signé $[M', s(M)]$ d'Alice.
 - ▶ Rappel : $s(M) = (r, s)$
 - ▶ Ce document est potentiellement altéré/illégitime
- ▶ Il récupère la clé publique d'Alice (p, g, A)
- ▶ Il calcule $h_{M'} = H(M')$
- ▶ Il vérifie l'identité : $A^r r^s = g^{h_{M'}} \pmod p$

$$\begin{aligned} \text{En effet, } A^r r^s &= g^{a.r} \cdot g^{kk^{-1}(h_M - a.r)} \pmod p \\ &= g^{h_M} \pmod p \end{aligned}$$

Si le document est authentique : $h_M = h_{M'} \Rightarrow g^{h_M} = g^{h_{M'}} \pmod p$

Sécurité des signatures El Gamal

- ▶ Sécurité intimement liée à DLP dans \mathbb{F}_p^*
 - ▶ Résolution de DLP dans \mathbb{F}_p^*
 - ⇒ possibilité de calculer a à partir de A
 - ⇒ possibilité d'impersonaliser Alice
- ▶ Attention au choix des paramètres.

Le standard DSA

Génération des paramètres

- ▶ Alice génère un nb premier q de 160 bits ($2^{159} \leq q < 2^{160}$)
- ▶ Elle génère un nb premier p de 512 à 1024 bits vérifiant :

$$\exists t \in [0, 8] / 2^{511+64t} < p < 2^{512+64t} \text{ et } q | (p - 1)$$

cela assure que \mathbb{F}_p^* possède un sous-groupe d'ordre q

- ▶ Soit g' une racine primitive modulo p
- ▶ Un générateur du sous-groupe de \mathbb{F}_p^* d'ordre q est alors

$$g = g'^{\frac{p-1}{q}} \pmod{p}$$

Le standard DSA (2)

Génération des paramètres (suite)

Une fois choisis (p, q, g) :

- ▶ Alice choisit $a \in \{1, \dots, q - 1\}$
- ▶ Elle calcule $A = g^a \pmod p$

Clef publique : (p, q, g, A) .

Clef secrète : a

Le problème du logarithme discret sous-jacent se passe dans le groupe d'ordre q .

Le standard DSA (3)

Génération d'une signature DSA

Alice souhaite signer un document M :

- ▶ Alice calcule $h_M = H(M)$ (on suppose $1 \leq h_M < q - 1$)
- ▶ Elle choisit un entier $k \in \{1, \dots, q - 1\}$
- ▶ Signature de M : $s(M) = (r, s)$ avec

$$r = (g^k \bmod p) \bmod q \text{ et } s = k^{-1}(h_M + a.r) \bmod q$$

- ▶ Le document signé est alors $[M, s(M)]$.

Le standard DSA (4)

Vérification d'une signature DSA

- ▶ Bob reçoit un document signé $[M', s(M) = (r, s)]$ d'Alice.
- ▶ Il récupère la clé publique d'Alice (p, q, g, A)
- ▶ Il vérifie que les formats sont respectés : $1 \leq r, s \leq q - 1$
- ▶ Il calcule $h_{M'} = H(M')$
- ▶ Il vérifie l'identité :

$$r = [(g^{s^{-1}h_{M'}} \cdot A^{rs^{-1}} \cdot g^{rs^{-1}}) \bmod p] \bmod q.$$