Hachage cryptographique

1. **Principe**
2. **Qualités requises**
3. **Utilisations**
4. **MD5**
5. **SHA-1**
6. **Exemple**
7. **MAC**
8. **DSA**

1. Principe

Une *fonction de hachage* H calcule

$$h = H(m)$$

où m est un message de taille arbitraire, h est de longueur fixe 128 bits p. ex. On demande

- h est facile à calculer à partir de m;
- m est difficile à calculer à partir de h;
- difficile de trouver m' tel que $H(m) = H(m')$.

La dernière propriété est appelée *résistance à la collision.*

Un *hachage par compression* (comme MD5) décompose m en blocs

$$m = M_1 \cdots M_k$$

chaque M_i de longueur fixe (p. ex. 512 bits). On calcule

$$H_i = F(M_i, H_{i-1}), \quad 1 \leq i \leq k$$

Le résultat est $h = g(H_k)$ et dépend d'une valeur initiale H_0. La fonction F est la *fonction de compression.*

2. Qualités requises

- **résistance aux préimages (*one way* ou *à sens unique*): donnée y, difficile de trouver x tel que $H(x) = y$**
- **résistance à deux préimages (*weak collision resistance*): donnée x, difficile de trouver x' tel que $H(x) = H(x')$**
- **résistance aux collisions (*strong collision resistance*): difficile de trouver x et x' tels que $H(x) = H(x')$.**

Classification fonctionnelle

- **MDC (modification detection code), assure l'intégrité du message, car deux messages $m \neq m'$ donnent $H(m) \neq H(m')$.**
- **MAC (message authentification code) assure l'intégrité du message et de la source: données source *et* clé. La fonction de hachage dépend d'une clé.**

3. Utilisations

- **Authentification de messages: $H(m)$ est un *résumé* une *empreinte* (digest, fingerprint).**
- **Signature d'un message: La *signature* de A pour m est $E_k(H(m))$, où E_k est une fonction de cryptage.**

4. MD5
Message Digest

MD5 est une fonction de hachage à sens unique développée par Rivest. Ancêtre : MD4. Caractéristiques communes

1. **Complétion du texte source à une longueur multiple de 512 bits (par exemple).**
2. **Plusieurs *tours* (round), chaque tour à plusieurs *étapes* (steps)**

nom	bits	tours×étapes
MD5	128	4×16
SHA-1	160	4×20

Complétion

A. ***Complétion*** **du texte à une longueur en bits multiple de 512 par ajout de $10^n b_1 b_0$, avec b_0 et b_1 sur 32 bits, et $bin(b_0 b_1)$ est le nombre de bits dans le texte.**

Le texte complété est

$$m = M_1 \cdots M_k$$

M_i de 512 bits.

Initialisation de la valeur de hachage

B. La valeur de hachage courante dans 4 mots de 32 bits

$$(H_1, H_2, H_3, H_4)$$

initialisés à

$$(H_1, H_2, H_3, H_4) \leftarrow (h_1, h_2, h_3, h_4)$$

avec

$$\begin{array}{ll} h_1 = 0x67452301 & h_2 = 0xefcdab89 \\ h_4 = 0x10325476 & h_3 = 0x98badcfe \end{array}$$

Les tours

C. Pour chaque bloc de taille 512 faire

$$M = M[0] \cdots M[15] \qquad M[i] \text{ 32 bits}$$

- $(A, B, C, D) \leftarrow (H_1, H_2, H_3, H_4)$
- faire les 4 tours de 16 étapes avec les $M[j]$
- Mettre à jour

$$(H_1, H_2, H_3, H_4) \leftarrow (H_1 + A, H_2 + B, H_3 + C, H_4 + D)$$

D. Résultat (128 bits):

$$H_1 \| H_2 \| H_3 \| H_4$$

Les tours (détail)

Round 1

Pour $j = 0, \ldots, 15$ faire

$$\begin{aligned} T &\leftarrow A + f(B, C, D) + M[z_j] + y_j \\ (A, B, C, D) &\leftarrow (D, B + (T \lll s_j), B, C) \end{aligned}$$

Round 2

Pour $j = 16, \ldots, 31$, idem avec f remplacée par g.

Round 3

Pour $j = 32, \ldots, 47$, idem avec f remplacée par h.

Round 4

Pour $j = 48, \ldots, 63$, idem avec f remplacée par k.

Les fonctions (détail)

$$s_j = \begin{cases} (7\ 12\ 17\ 22) & (0 \leq j \leq 15) \\ (5\ 9\ 14\ 20) & (16 \leq j \leq 31) \\ (4\ 11\ 16\ 23) & (32 \leq j \leq 47) \\ (6\ 10\ 15\ 21) & (48 \leq j \leq 63) \end{cases} \qquad y_j = \lfloor 2^{32} sin(j+1) \rfloor$$

$$z_j = \begin{cases} j & (0 \leq j \leq 15) \\ 5j+1 & (16 \leq j \leq 31) \\ 3j+5 & (32 \leq j \leq 47) \\ 7j & (48 \leq j \leq 63) \end{cases} \qquad \begin{aligned} f(x,y,z) &= (x \wedge y) \vee (\bar{x} \wedge z) \\ g(x,y,z) &= (x \wedge z) \vee (y \wedge \bar{z}) \\ h(x,y,z) &= x \oplus y \oplus z \\ k(x,y,z) &= y \oplus (x \vee \bar{z}) \end{aligned}$$

5. SHA-1
Secure Hash Algorithm

A. ***Complétion*** **du texte comme pour MD5**

B. La valeur de hachage courante dans 5 mots de 32 bits

$$(H_1, H_2, H_3, H_4, H5)$$

initialisés à

$$(H_1, H_2, H_3, H_4, H_5) \leftarrow (h_1, h_2, h_3, h_4, h_5)$$

avec h_1, h_2, h_4, h_3 **comme avant et** $h_5 = 0xc3d2e1f0$.

Explosion avant les tours

$$M = M[0] \cdots M[15]$$

est développé en

$$M[0] \cdots M[79]$$

avec, pour $j = 16, \ldots, 79$,

$$M[j] \leftarrow (M[j-3] \oplus M[j-8] \oplus M[j-12] \oplus M[j-16]) \lll 1$$

puis initialisation par

$$(A, B, C, D, E) \leftarrow (H_1, H_2, H_3, H_4, H_5)$$

Les tours

Round 1

Pour $j = 0, \ldots, 19$ faire

$$T \leftarrow A \lll 5 \; + \; f(B, C, D) + E + M[j] + y_1$$
$$(A, B, C, D, E) \leftarrow (T, A, B \lll 30, C, D)$$

Round 2

Pour $j = 20, \ldots, 39$, idem avec f remplacée par h (!) et y_1 remplacé par y_2.

Round 3 Pour $j = 40, \ldots, 59$, idem avec f remplacée par g et y_1 remplacé par y_3.

Round 4 Pour $j = 60, \ldots, 79$, idem avec f remplacée par h (!!) et y_1 remplacé par y_4.

Les constantes sont

$$y_1 = \sqrt{(}2)/4,\ y_2 = \sqrt{(}3)/4, y_3 = \sqrt{(}5)/4, y_4 = \sqrt{(}10)/4$$

écrit comme long.

6. Exemple

```
import java.io.*;
import java.security.*;
import sun.misc.*;

public class Masher {

  public static void main(String[] args) throws Exception{
    // Check arguments.
    if (args.length != 1) {
      System.out.println("Usage: Masher filename");
      return;
    }
```

```
// Obtain a message digest object.
MessageDigest md = MessageDigest.getInstance("MD5");
// ou "SHA"

// Calculate the digest for the given file.
FileInputStream in = new FileInputStream(args[0]);
byte[] buffer = new byte[8192];
int length;
while ((length = in.read(buffer)) != -1)
  md.update(buffer, 0, length);  // ajout
byte[] raw = md.digest();        // fin
```

```
    // Print out the digest in base64.
    BASE64Encoder encoder = new BASE64Encoder();
    String base64 = encoder.encode(raw);
    System.out.println(base64);
  }
}
```

Résultats :MD5


```
Crj4dBrR6jNbWXbKyvdX/A==
DHVihNXrv6FSzd4JnHNJNA==
```


SHA


```
fTBRqkHDGKp5ANY34reTO14OgIM=
h0ae4ukFRdlWP4uOvABML6SnkWA=
```

7. MAC

MAC est un *message authentification code* qui utilise une fonction de cryptage E_k et le chaînage CBC. Le message m est décomposé:

$$m = m_1 \cdots m_t$$

On calcule

$$\begin{aligned} H_1 &= E_k(m_1) \\ H_i &= E_k(H_{i-1} \oplus m_i) \qquad i = 2, \ldots, t \end{aligned}$$

8. DSA

DSA (Digital Signature Algorithm) est un standard de signature, avec une clé publique.

Alice *crypte* la valeur de hachage $H(m)$ avec une clé secrète et un nombre choisi au hasard.

Bob *décrypte* en calculant $H(m)$ et en vérifiant une égalité grâce à la clé publique d'Alice.

DSA : données publiques

- deux nombres premiers grands p et q tels que $p-1$ est un multiple de q (p.ex. $q = 5$ et $p = 11$)
- un "générateur" $g \in Z_p^*$ tel que les nombres $g, g^2, \ldots g^q$ sont tous différents mod p, et $g^q \equiv 1$ mod p (par exemple $g = 3$, donnant $3, 9, 5, 4, 1$)

q a 160 bits, et p entre 512 et 1024 bits.

DSA : Alice

Alice choisit un entier x avec $1 < x < q - 1$, calcule $y = g^x$. La clé publique d'Alice est y, sa clé privée est x.

Pour *signer* un message m, Alice

- **choisit au hasard un entier k avec $0 < k < q$;**
- **calcule $r \equiv (g^k \bmod p) \bmod q$,**
- **calcule $\ell = k^{-1} \bmod q$**
- **calcule $s \equiv \ell \cdot (H(m) + xr) \bmod q$.**
- **la signature est (r, s).**

DSA : Bob

Bob a reçu le message m et la signature (r, s).
Bob connaît (p, q, g, y), et connaît la fonction de hachage H utilisée.

Pour *vérifier* la signature, il

- vérifie que $0 < r < q$ et $0 < s < q$;
- calcule $w \equiv s^{-1} \bmod q$;
- calcule $u_1 \equiv w \cdot H(m) \bmod q$
- calcule $u_2 \equiv r \cdot w \bmod q$;
- calcule $v \equiv (g^{u_1} y^{u_2} \bmod p) \bmod q$
- accept la signature si et seulement si $v = r$.