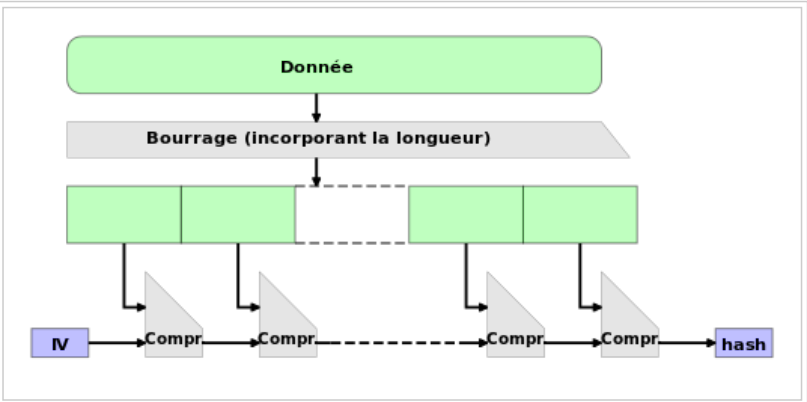
SHA-2

SHA-2 (*Secure Hash Algorithm*) est une famille de fonctions de hachage qui ont été conçues par la *National Security Agency* des États-Unis (NSA), sur le modèle des fonctions SHA-1 et SHA-0, elles-mêmes fortement inspirées de la fonction MD4 de Ron Rivest (qui a donné parallèlement MD5). Telle que décrite par le *National Institute of Standards and Technology* (NIST), elle comporte les fonctions, **SHA-256** et **SHA-512** dont les algorithmes sont similaires mais opèrent sur des tailles de mot différentes (32 bits pour SHA-256 et 64 bits pour SHA-512), **SHA-224** et **SHA-384** qui sont essentiellement des versions des précédentes dont la sortie est tronquée, et plus récemment **SHA-512/256** et **SHA-512/224** qui sont des versions tronquées de SHA-512. Le dernier suffixe indique le nombre de bits du haché.

Les algorithmes de la famille SHA-2, SHA-256, SHA-384 et SHA-512, sont décrits et publiés en compagnie de SHA-1 comme standard du gouvernement fédéral des États-Unis (Federal Information Processing Standard) dans le *FIPS 180-2 (Secure Hash Standard)* datant de 2002 (une prépublication pour appels à commentaires a été faite en 2001). La fonction SHA-224 est ajoutée un peu plus tard. La dernière version à ce jour, le *FIPS 180-4 (Secure Hash Standard)* date de mars 2012 et ajoute les fonctions SHA-512/256 et SHA-512/224[1].

En 2005, des problèmes de sécurité de SHA-1 ont été mis en évidence : il existe pour la recherche de collisions une attaque théorique nettement plus rapide que l'attaque générique des anniversaires sur les fonctions de hachage. Bien que l'algorithme de SHA-2 partage des similarités avec celui de SHA-1, ces attaques n'ont actuellement pas pu être étendues à SHA-2. Le NIST a cependant organisé un concours pour sélectionner une nouvelle fonction de hachage, SHA-3. Le concours a débouché fin 2012 sur le choix d'une nouvelle famille de fonctions dont la conception est très différente de SHA-1 et de SHA-2. La nouvelle famille de fonctions est présentée comme un autre choix possible, elle ne remet pas en cause l'utilisation de SHA-2 du moins dans l'immédiat.

Sommaire

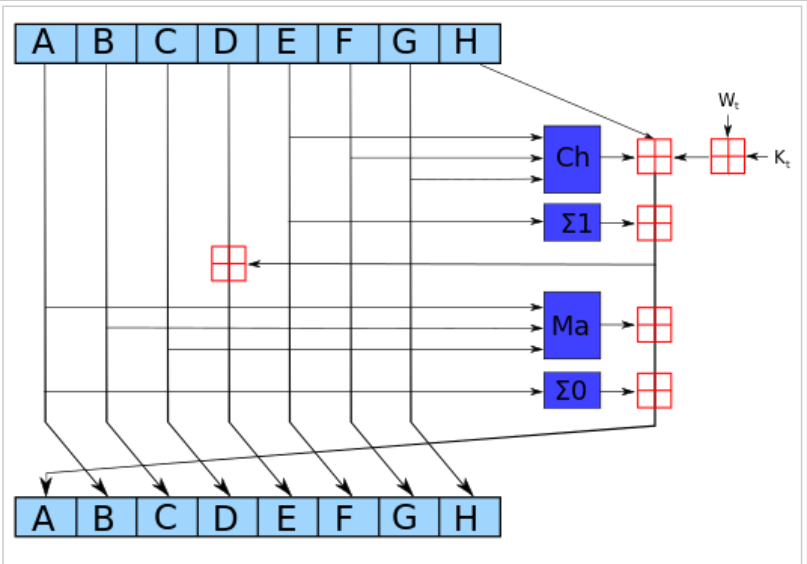


Les fonctions de la famille SHA-2 adoptent le schéma itératif de la construction de Merkle-Damgård

Description

Comme toutes les fonctions de hachage les fonctions SHA-2 prennent en entrée un message de taille arbitraire, avec une borne (toute théorique) pour celle-ci, et produisent un résultat (appelé « *hash* », *haché*, *condensat* ou encore *empreinte* ...) de taille fixe. La taille du haché est indiquée par le suffixe : 224 bits pour SHA-224, 256 bits pour SHA-256, 384 bits pour SHA-384 et 512 bits pour SHA-512.

Les algorithmes de la famille SHA-2 sont très semblables, il y a essentiellement deux fonctions différentes, SHA-256 et de SHA-512, les autres étant des variantes de l'une ou l'autre. Les fonctions SHA-256 et SHA-512 ont la même structure mais diffèrent par la taille des mots et des blocs utilisés. Cette structure est assez proche de celle de SHA-1, mais un peu plus complexe et en évite certaines faiblesses connues. Elle se rattache plus généralement à une famille de fonctions de hachage inspirées de MD4 et MD5 de Ron Rivest. On retrouve comme primitives l'addition pour des entiers de taille fixe n soit une addition modulo 2^n, opération non linéaire (au sens de l'algèbre linéaire) sur le corps des booléens $\mathbf{F}_2$, ainsi que des opérations bit à bit (xor et autres).



Un tour de la fonction de compression de SHA-2, celle-ci en comporte 64 (SHA-256) ou 80 (SHA-512) :
A, B, C, D, E, F, G et H sont des mots de 32 bits (SHA-256) ou 64 bits (SHA-512) ;
$\boxplus$ est l'addition modulo 2^{32} (SHA-256) ou 2^{64} (SHA-512), soit l'addition des entiers machines non signés de 32 ou 64 bits, qui est non linéaire sur $\mathbf{F}_2$;
les fonctions Ch et Ma se calculent par opérations bit à bit logiques, elles sont non linéaires ;
les fonctions $\Sigma 0$ et $\Sigma 1$ se calculent par décalage circulaire et xor (somme sur $\mathbf{F}_2$) ;
K_t est une constante (32 ou 64 bits) qui dépend du numéro de tour t ;
W_t est un mot de 32 (SHA-256) ou 64 bits (SHA-512) qui dépend du numéro de tour t ; il est obtenu par une procédure d'expansion à partir du bloc de donnée (512 ou 1024 bits) dont le traitement est en cours.

Comme toutes les fonctions de cette famille, elles suivent un schéma itératif qui suit la construction de Merkle-Damgård (sans opération de finalisation). La fonction de compression itérée possède deux entrées de taille fixe, la seconde entrée étant de même taille que la sortie de la fonction :

- une donnée obtenue par découpage du message à traiter, la taille est de 512 bits pour SHA-256 et SHA-224 et de 1024 bits pour SHA-512 et SHA-384,
- le résultat de la fonction de compression à l'itération précédente (256 bits pour SHA-256 et SHA-224, 512 pour SHA-512 et SHA-384).

Les entrées de la fonction de compression sont découpées

- en mots de 32 bits pour SHA-256 et SHA-224,
- en mots de 64 bits pour SHA-512 et SHA-384.

La fonction de compression répète les mêmes opérations un nombre de fois déterminé, on parle de *tour* ou de *ronde*, 64 tours pour SHA-256, 80 tours pour SHA-512. Chaque tour fait intervenir comme primitives l'addition entière pour des entiers de taille fixe, soit une addition modulo 2^{32} ou modulo 2^{64}, des opérations bit à bit : opérations logiques, décalages avec perte d'une partie des bits et décalages circulaires, et des constantes prédéfinies, utilisées également pour l'initialisation.

Avant traitement, le message est complété par bourrage de façon que sa longueur soit un multiple de la taille du bloc traité par la fonction de compression. Le bourrage incorpore la longueur (en binaire) du mot à traiter : c'est le renforcement de Merkle-Damgård (`(en)`*Merkle-Damgård strengthening*), ce qui permet de réduire la résistance aux collisions de la fonction de hachage à celle de la fonction de compression. Cette longueur est stockée en fin de bourrage sur 64 bits dans le cas de SHA-256 (comme pour SHA-1), sur 128 bits dans le cas de SHA-512, ce qui « limite » la taille des messages à traiter à 2^{64} bits pour SHA-256 (et SHA-224) et à 2^{128} bits pour SHA-512 (et SHA-384).

SHA-256

La fonction SHA-256 devient en 2002 un standard fédéral de traitement de l'information (FIPS du NIST). Elle produit un hachage de 256 bits. Les caractéristiques de SHA-256 sont les suivantes :

- taille du message : 2^{64} bits maximum
- taille des blocs : 512 bits
- taille des mots : 32 bits
- taille du condensé : 256 bits
- niveau de sécurité attendu (collisions) : 2^{128} bits (attaque des anniversaires)
- nombre de tours (fonction de compression) : 64

Algorithme

L'algorithme peut être découpé en deux phases

- Le prétraitement : le message est complété par remplissage comprenant la taille du message (renforcement de Merkle-Damgård) de façon à pouvoir le découper en blocs de 512 bits ;
- Le calcul du condensé par itération de la fonction de compression sur la suite des blocs obtenus en découpant le message (schéma itératif de Merkle-Damgård).

Symboles et termes utilisés

Paramètres

a, b, c, ..., h = variables de travail (en l'occurrence des mots de w bits), utilisées dans le calcul des hachés

$H^{(i)}$ = la valeur de hachage n° i. $H^{(0)}$ est la valeur initiale du hachage. $H^{(N)}$ est la dernière valeur de hachage.

$H_j^{(i)}$ = le mot (w bits) n° j de la valeur de hachage n° i, où $H_0^{(i)}$ est le mot de poids le plus fort (à gauche) de la valeur de hachage i.

K_t = constantes itératives selon la valeur de t, utilisées dans le calcul de hachage

k = nombre de 0 ajoutés au message lors du prétraitement (complément)
l = longueur du message M, en bits
m = nombre de bits contenus dans un bloc, soit 512 bits
M = message à traiter
$M^{(i)}$ = bloc n° i (m bits), du message M
$M_j^{(i)}$ = mot (w bits) n° j, du bloc (m bits) n° i, du message M
n = nombre de bits de décalage ou de rotation à appliquer au mot quand associé à une fonction binaire
N = nombre de blocs de m bits contenus dans le message M après complément
T = variable temporaire, mot de w bits, utilisée dans le calcul de condensé
w = nombre de bits contenus dans un mot, soit 32 bits.
W_t = le mot n° t du tableau déduit du message

Symboles

La notation hexadécimale utilisée ici sera: $0\mathrm{x}$

exemple: $H_0^{(0)} = 0\mathrm{x}12\mathrm{ab}34\mathrm{ef}$

$\wedge$ = opération binaire AND
$\vee$ = opération binaire OR
$\oplus$ = opération binaire XOR
$\neg$ = complément binaire
$+$ = addition modulo 2^w
$<<$ = décalage binaire à gauche, où $x << n$ s'obtient en supprimant les n bits de gauche de x et ajoutant n zéros à droite.
$>>$ = décalage binaire à droite, où $x >> n$ s'obtient en supprimant les n bits de droite de x et ajoutant n zéros à gauche.

Opérations sur les mots

Elles utilisent les conventions suivantes :

- les opérations binaires bit à bit (cf. symboles) ;
- l'addition modulo 2^w, soit 2^{32}, où l'opération $x + y$ est définie comme suit :
  - soient deux mots x et y représentant les nombres entiers X et Y, tels que $0 \leq X < 2^{32}$ et $0 \leq Y < 2^{32}$;
  - on a Z le résultat de l'addition modulo 2^{32} de X et Y, tel que $Z = (X + Y) \mathrm{modulo} 2^{32}$ et $0 \leq Z < 2^{32}$;
  - on convertit alors Z en un mot z, et on définit $z = x + y$;
- l'opération de décalage binaire à droite $SHR^n(x)$, où x est un mot de 32 bits et $0 \leq n \leq 32$, est définie par :
  - $SHR^n(x) = x >> n$;
- l'opération de rotation binaire par la droite $ROTR^n(x)$, où x est un mot de 32 bits et $0 \leq n \leq 32$, est définie par :
  - $ROTR^n(x) = (x >> n) \vee (x << (32 - n))$

Fonctions et constantes

Fonctions

Cette section décrit les fonctions utilisées lors du calcul des valeurs de hachage. SHA-256 utilise 6 fonctions logiques travaillant sur des mots de 32 bits notés x, y, z. Le résultat de chacune de ces fonctions est un nouveau mot de 32 bits en sortie.

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\Sigma_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$

$$\Sigma_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

Constantes

SHA-256 utilise 64 valeurs constantes de mots de 32 bits, notés $K_0^{\{256\}}, K_1^{\{256\}}, ..., K_{63}^{\{256\}}$. ces nombres représentent les 32 premiers bits de la partie décimale des racines cubiques des 64 premiers nombres premiers. Les valeurs suivantes sont exprimées en notation hexadécimale (base 16).

$K_0^{\{256\}}, ..., K_7^{\{256\}}$	0x428a2f98	0x71374491	0xb5c0fbcf	0xe9b5dba5	0x3956c25b	0x59f111f1	0x923f82a4	0xab1c5ed5
$K_8^{\{256\}}, ..., K_{15}^{\{256\}}$	0xd807aa98	0x12835b01	0x243185be	0x550c7dc3	0x72be5d74	0x80deb1fe	0x9bdc06a7	0xc19bf174
$K_{16}^{\{256\}}, ..., K_{23}^{\{256\}}$	0xe49b69c1	0xefbe4786	0x0fc19dc6	0x240ca1cc	0x2de92c6f	0x4a7484aa	0x5cb0a9dc	0x76f988da
$K_{24}^{\{256\}}, ..., K_{31}^{\{256\}}$	0x983e5152	0xa831c66d	0xb00327c8	0xbf597fc7	0xc6e00bf3	0xd5a79147	0x06ca6351	0x14292967
$K_{32}^{\{256\}}, ..., K_{39}^{\{256\}}$	0x27b70a85	0x2e1b2138	0x4d2c6dfc	0x53380d13	0x650a7354	0x766a0abb	0x81c2c92e	0x92722c85
$K_{40}^{\{256\}}, ..., K_{47}^{\{256\}}$	0xa2bfe8a1	0xa81a664b	0xc24b8b70	0xc76c51a3	0xd192e819	0xd6990624	0xf40e3585	0x106aa070
$K_{48}^{\{256\}}, ..., K_{55}^{\{256\}}$	0x19a4c116	0x1e376c08	0x2748774c	0x34b0bcb5	0x391c0cb3	0x4ed8aa4a	0x5b9cca4f	0x682e6ff3
$K_{56}^{\{256\}}, ..., K_{63}^{\{256\}}$	0x748f82ee	0x78a5636f	0x84c87814	0x8cc70208	0x90befffa	0xa4506ceb	0xbef9a3f7	0xc67178f2

Prétraitement

Cette opération se déroule en trois étapes : compléter le message M, découper le résultat en blocs, et initialiser les valeurs de hachage $H^{(0)}$

Complément de M

Il s'agit ici d'ajouter des informations à M pour qu'il soit d'une taille multiple de 512 bits :

- pour ce faire, on ajoute un bit "1" à la fin du message M,
- puis k bits "0" de bourrage, où k est la plus petite solution non négative de l'équation : $l + 1 + k = 448 \mod 512$ et

- un bloc de 64 bits correspondant à la représentation binaire de l.

Exemples :

- Pour M = « abc », $l = 3 \times 8 = 24$ bits, $k = 448 - (l + 1) = 448 - (24 + 1) = 423$ bits de bourrage ;
  - on ajoute le bit "1",
  - puis quatre cent vingt-trois bits "0" de bourrage,
  - puis 64 bits finissant par "11000" (pour 24) à M ;

    on obtient alors un message complété à 512 bits.
- Pour M quelconque tel que $l = 500$ bits, $k = 448 - (l + 1) = 448 - (500 + 1) = -53$; comme k ne peut pas être négatif, on lui ajoute 512 en prenant en compte le modulo de l'équation, pour obtenir $k = 459$ bits de bourrage ;
  - on ajoute le bit "1",
  - puis quatre cent cinquante-neuf bits "0" de bourrage,
  - puis 64 bits finissant par "111110100" (pour 500) à M ;

    on obtient alors un message complété à 1024 bits.

Découpage en blocs

Le message complété est découpé en N blocs de 512 bits, notés $M^{(1)}, M^{(2)}, ..., M^{(N)}$. Chaque bloc de 512 bits est ensuite découpé en 16 mots de 32 bits, notés $M_0^{(i)}, M_1^{(i)}, ..., M_{15}^{(i)}$.

Initialisations

Les huit variables suivantes sont affectées de valeurs initiales comme suit :

$$
\begin{aligned}
H_0^{(0)} &= \mathrm{0x6a09e667} \\
H_1^{(0)} &= \mathrm{0xbb67ae85} \\
H_2^{(0)} &= \mathrm{0x3c6ef372} \\
H_3^{(0)} &= \mathrm{0xa54ff53a} \\
H_4^{(0)} &= \mathrm{0x510e527f} \\
H_5^{(0)} &= \mathrm{0x9b05688c} \\
H_6^{(0)} &= \mathrm{0x1f83d9ab} \\
H_7^{(0)} &= \mathrm{0x5be0cd19}
\end{aligned}
$$

Calcul du condensé (haché)

Pour ce traitement on utilisera

- un tableau de 64 mots, notés $W_0, W_1, ..., W_{63}$
- huit variables notées a, b, c, d, e, f, g, h
- huit variables contenant les valeurs de hachage, notées $H_0^{(i)}$ et initialisées précédemment en $H_0^{(0)}$

  Ces variables contiendront itérativement de nouvelles valeurs de hachage, $H^{(i)}$, pour finalement contenir le condensé de M, dans $H^{(N)}$.
- deux variables, notées T_1 et T_2, mots de 32 bits.

On traite successivement les N blocs de M selon les étapes suivantes

Pour i = 1 à N
{

1. On remplit le tableau W_t si $0 \leq t \leq 63$, selon

$$
W_t = \begin{cases} M_t^{(i)}, & 0 \leq t \leq 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16}, & 16 \leq t \leq 63 \end{cases}
$$

2. On initialise a,b, c, d, e, f, g et h avec les valeurs de hachage du tour précédent

$$
\begin{aligned}
a &= H_0^{(i-1)} \\
b &= H_1^{(i-1)} \\
c &= H_2^{(i-1)} \\
d &= H_3^{(i-1)} \\
e &= H_4^{(i-1)}
\end{aligned}
$$

$$f = H_5^{(i-1)}$$
$$g = H_6^{(i-1)}$$
$$h = H_7^{(i-1)}$$

3. Pour t = 0 à 63
{

$$T_1 = h + \Sigma_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$
$$T_2 = \Sigma_0^{\{256\}}(a) + Maj(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

}

4. Calcul des valeurs de hachage intermédiaires

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

}

Après répétition des quatre étapes ci-dessus pour les N blocs du message M, (i.e., après traitement de $M^{(N)}$), le condensé de 256 bits de M est obtenu par concaténation des valeurs

$$H_0^{(N)} || H_1^{(N)} || H_2^{(N)} || H_3^{(N)} || H_4^{(N)} || H_5^{(N)} || H_6^{(N)} || H_7^{(N)}$$

SHA-224

La fonction SHA-224 est publiée pour la première fois en 2004. Le résultat produit (haché, « hash » ou condensat) est de 224 bits. Elle a été spécialement conçue pour fournir une empreinte dont la taille correspond à quatre clés DES de 56 bits chacune. L'algorithme est celui de SHA-256 avec pour seules différences

- des valeurs différentes pour l'initialisation (variables h0, … , h7) ;
- une sortie tronquée à 224 bits (concaténation des contenus des 7 premières variables h0, … , h6).

SHA-512

La fonction SHA-512 est apparue en même temps que SHA-256 et devient comme celle-ci en 2002 un standard fédéral de traitement de l'information (FIPS du NIST). Elle produit un haché de 512 bits.

L'algorithme est très similaire à celui de SHA-256 mais avec une différence importante dans la taille des données traitées : la taille de bloc est de 1024 bits (et non 512 bits), et l'algorithme opère sur des mots de 64 bits (la taille de mot-mémoire de beaucoup de processeurs modernes). En particulier les opérations arithmétiques, particulièrement optimisées sur ces processeurs se font sur 64 bits.

La structure de l'algorithme est la même que celle de SHA-256 mais

- le message est découpé en blocs de 1024 bits ;
- les nombres qui apparaissent (variables h0, … , h7, constantes …) sont sur 64 bits et l'addition se fait modulo 2^{64} ;
- les valeurs des initialisations et des constantes sont différentes (forcément puisque sur 64 et non 32 bits) ;
- la fonction de compression utilise 80 tours (au lieu de 64) ;
- la longueur du message (en binaire) ajoutée à celui-ci dans la phase de bourrage (renforcement de Merkle-Damgard) est un entier de 128 bits (big-endian) et non 64 bits, ce qui limite la taille théorique maximale du message à 2^{128} bits et non plus 2^{64} bits ;
- les valeurs des décalages et décalages circulaires sont modifiées de façon à tenir compte de la nouvelle longueur des mots.

SHA-384

La fonction SHA-384 est apparue en compagnie de SHA-256 et SHA-512. Elle produit un haché de 384 bits. L'algorithme est celui de SHA-512 avec pour seules différences

- des valeurs différentes pour l'initialisation (variables h0, … , h7) ;
- une sortie tronquée à 384 bits (concaténation des contenus des 6 premières variables h0, … , h5).

Cryptanalyse

SHA-256 est devenu le nouveau standard recommandé en matière de hachage cryptographique après les attaques sur MD5 et SHA-1. Les autres membres de la famille SHA ont été relativement peu cryptanalysés par rapport à SHA-0 et SHA-1. En 2003, Helena Handschuh et Henri Gilbert ont publié une analyse de SHA-256, 384 et 512. Leur étude montre que les autres membres de SHA ne sont pas atteints par les attaques qui avaient fait leurs preuves sur les autres fonctions de hachage (MD4, MD5 et SHA-1 entre autres). La cryptanalyse linéaire et différentielle ne s'appliquent pas.

En revanche, les deux cryptologues ont mis en évidence des faiblesses significatives sur des versions modifiées. En changeant les constantes ou les paramètres d'initialisation de manière à les rendre symétriques tout en remplaçant l'addition modulaire par un XOR, on obtient un hachage qui produit une empreinte symétrique si le message en entrée l'est également.

Références

1. voir la page Secure Hashing du NIST (http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html)

Voir aussi

Bibliographie

- **(en)** Henri Gilbert, Helena Handschuh : *Security Analysis of SHA-256 and Sisters*. Selected Areas in Cryptography 2003: 175-193
- Versions successives du *Secure Hash Standard* du NIST à partir de l'apparition de SHA-2 (voir aussi la page Secure Hashing (http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html))
  - FIPS 180-2 (http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf) version d'août 2001
  - FIPS 180-3 (http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf) version finale d'octobre 2008
  - FIPS 180-4 (http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf) mars 2012

Articles connexes

- Secure Hash Algorithm
- SHA-1
- SHA-3

Lien externe

- Calculateur en ligne (http://www.xorbin.com/tools/sha256-hash-calculator) Encodage de chaînes de caractères par les fonctions de hachage les plus utilisées.
- **(en)** Calculatrice qui génère SHA-512 (http://slavasoft.com/hashcalc/index.htm) (logiciel gratuit pour Windows)
- **(en)** Logiciel pour calculer et verifier SHA-512 et autres algorithmes (http://ocr.altervista.org/wordpress/footprint/) (logiciel gratuit)
