PUBLIK FIPS 180-4

STANDAR PEMROSESAN INFORMASI FEDERAL PUBLIKASI

Standar Hash Aman (SHS)

KATEGORI: KEAMANAN KOMPUTER SUBKATEGORI: KRIPTOGRAFI

Laboratorium Teknologi Informasi
Institut Standar dan Teknologi Nasional
Gaithersburg, MD 20899-8900

Publikasi ini tersedia gratis dari: http://dx.doi.org/10.6028/NIST.FIPS.180-4

Agustus 2015

Departemen Perdagangan AS
Penny Pritzker, Sekretaris

Institut Nasional Standar dan Teknologi *Willie E. May, Wakil Sekretaris Standar dan Teknologi dan Direktur*

KATA PENGANTAR

Seri Publikasi Standar Pemrosesan Informasi Federal dari Institut Nasional Standar dan Teknologi (NIST) adalah seri publikasi resmi yang berkaitan dengan standar dan pedoman yang diadopsi dan diumumkan berdasarkan ketentuan Undang-Undang Manajemen Keamanan Informasi Federal (FISMA) tahun 2002.

Komentar mengenai publikasi FIPS diterima dengan senang hati dan harus ditujukan kepada Direktur, Laboratorium Teknologi Informasi, Institut Nasional Standar dan Teknologi, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

Charles H. Romine, Direktur
Laboratorium Teknologi Informasi

Abstrak

Standar ini menentukan algoritma hash yang dapat digunakan untuk menghasilkan intisari pesan. Intisari digunakan untuk mendeteksi apakah pesan telah diubah sejak intisari dibuat.

Kata kunci: keamanan komputer, kriptografi, intisari pesan, fungsi hash, algoritma hash, Standar Pemrosesan Informasi Federal, Standar Hash Aman.

Informasi Federal

Publikasi Standar Pemrosesan 180-4

Agustus 2015

Mengumumkan

STANDAR HASH AMAN

Publikasi Standar Pemrosesan Informasi Federal (FIPS PUBS) dikeluarkan oleh Institut Standar dan Teknologi Nasional (NIST) setelah disetujui oleh Sekretaris Perdagangan sesuai dengan Bagian 5131 Undang-Undang Reformasi Manajemen Teknologi Informasi tahun 1996 (Undang-Undang Publik 104-106), dan Undang-Undang Keamanan Komputer tahun 1987 (Undang-Undang Publik 100-235).

1. Nama Standar: Secure Hash Standard (SHS) (FIPS PUB 180-4).

2. Kategori Standar: Standar Keamanan Komputer, Kriptografi.

3. Penjelasan: Standar ini menetapkan algoritma hash aman - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 - untuk menghitung representasi ringkas data elektronik (pesan). Ketika pesan dengan panjang kurang dari 264 bit (misalnya SHA-1, SHA-224 dan SHA-256) atau kurang dari 2128 bit (untuk SHA-384, SHA-512, SHA-512/224 dan SHA-512/256) dimasukkan ke dalam algoritma hash, hasilnya adalah keluaran yang disebut intisari pesan. Panjang intisari pesan berkisar antara 160 hingga 512 bit, tergantung pada algoritmanya. Algoritma hash aman biasanya digunakan dengan algoritma kriptografi lainnya, seperti algoritma tanda tangan digital dan kode autentikasi pesan hash yang dikunci, atau dalam pembuatan angka acak (bit).

Algoritma hash yang ditentukan dalam Standar ini disebut aman karena, untuk algoritma tertentu, secara komputasi tidak mungkin 1) menemukan pesan yang sesuai dengan intisari pesan tertentu, atau 2) menemukan dua pesan berbeda yang menghasilkan intisari pesan yang sama. Setiap perubahan pada Pesan tersebut, dengan probabilitas yang sangat tinggi, akan menghasilkan intisari pesan yang berbeda. Hal ini akan mengakibatkan kegagalan verifikasi saat algoritma hash aman digunakan dengan algoritma tanda tangan digital atau algoritma autentikasi pesan hash yang dikunci.

Standar ini menggantikan FIPS 180-3 [FIPS 180-3].

4. Otoritas yang menyetujui: Sekretaris Perdagangan.

5. Badan Pemeliharaan: Departemen Perdagangan AS, Institut Standar dan Standar Nasional Teknologi (NIST), Laboratorium Teknologi Informasi (ITL).

6. Penerapan: Standar ini berlaku untuk semua departemen dan lembaga Federal untuk perlindungan informasi rahasia yang tidak dirahasiakan yang tidak tunduk pada Judul 10 Kode Amerika Serikat Bagian 2315 (10 USC 2315) dan yang tidak berada dalam sistem keamanan nasional sebagaimana didefinisikan dalam Judul 40 Kode Amerika Serikat Bagian 11103(a)(1) (40 USC 11103(a)(1)). Baik Standar ini maupun Standar Pemrosesan Informasi Federal (FIPS) 202 harus diterapkan di mana pun algoritma hash aman diperlukan untuk aplikasi Federal, termasuk sebagai komponen dalam algoritma dan protokol kriptografi lainnya. Standar ini dapat diadopsi dan digunakan oleh organisasi non-Pemerintah Federal.

7. Spesifikasi: Standar Pemrosesan Informasi Federal (FIPS) 180-4, Standar Hash Aman (SHS) (tertempel).

8. Implementasi: Algoritma hash aman yang ditentukan di sini dapat diimplementasikan dalam perangkat lunak, firmware, perangkat keras, atau kombinasi apa pun dari semuanya. Hanya implementasi algoritma yang divalidasi oleh NIST yang akan dianggap mematuhi standar ini. Informasi tentang program validasi dapat diperoleh di http://csrc.nist.gov/groups/STM/index.html.

9. Jadwal Pelaksanaan: Panduan mengenai pengujian dan validasi terhadap FIPS 180-4 dan hubungannya dengan FIPS 140-2 dapat ditemukan di IG 1.10 dari Panduan Implementasi untuk FIPS PUB 140-2 dan Program Validasi Modul Kriptografi di http://csrc.nist.gov/groups/STM/cmvp/index.html.

10. Paten: Implementasi algoritma hash aman dalam standar ini dapat dilindungi oleh Paten AS atau asing.

11. Kontrol Ekspor: Perangkat kriptografi tertentu dan data teknis yang terkait dengannya tunduk pada kontrol ekspor Federal. Ekspor modul kriptografi yang menerapkan standar ini dan data teknis terkaitnya harus mematuhi peraturan Federal ini dan dilisensikan oleh Biro Administrasi Ekspor Departemen Perdagangan AS. Informasi tentang peraturan ekspor tersedia di: http://www.bis.doc.gov/index.htm.

12. Kualifikasi: Meskipun tujuan Standar ini adalah untuk menentukan persyaratan keamanan umum, persyaratan untuk membuat intisari pesan, kesesuaian dengan Standar ini tidak menjamin bahwa implementasi tertentu aman. Otoritas yang bertanggung jawab di setiap lembaga atau departemen harus memastikan bahwa implementasi secara keseluruhan memberikan tingkat keamanan yang dapat diterima. Standar ini akan ditinjau setiap lima tahun untuk menilai kecukupannya.

13. Prosedur Pengabaian: Undang-Undang Manajemen Keamanan Informasi Federal (FISMA) tidak memungkinkan keringanan terhadap FIPS yang diwajibkan oleh Menteri Perdagangan.

14. Tempat Mendapatkan Salinan Standar: Publikasi ini tersedia secara elektronik melalui mengakses http://csrc.nist.gov/publications/. Publikasi keamanan komputer lainnya tersedia di situs web yang sama.

Informasi Federal
Publikasi Standar Pemrosesan 180-4

Spesifikasi untuk

STANDAR HASH AMAN

Daftar isi

1. PENDAHULUAN

Standar ini menetapkan algoritma hash aman, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, dan SHA-512/256. Semua algoritma tersebut adalah fungsi hash satu arah yang berulang yang dapat memproses pesan untuk menghasilkan representasi ringkas yang disebut *intisari pesan.* Algoritma ini memungkinkan penentuan integritas pesan: setiap perubahan pada pesan akan, dengan probabilitas yang sangat tinggi, menghasilkan intisari pesan yang berbeda. Properti ini berguna dalam pembuatan dan verifikasi tanda tangan digital dan kode autentikasi pesan, dan dalam pembuatan angka atau bit acak.

Setiap algoritma dapat dijelaskan dalam dua tahap: praproses dan komputasi hash.
Prapemrosesan melibatkan pengisian pesan, penguraian pesan yang diisi menjadi blok m-bit, dan pengaturan nilai inisialisasi yang akan digunakan dalam komputasi hash. Komputasi hash menghasilkan *jadwal pesan* dari pesan yang diisi dan menggunakan jadwal tersebut, bersama dengan fungsi, konstanta, dan operasi kata untuk secara berulang menghasilkan serangkaian nilai hash. Nilai hash akhir yang dihasilkan oleh komputasi hash digunakan untuk menentukan intisari pesan.

Algoritme-algoritme tersebut berbeda paling signifikan dalam kekuatan keamanan yang disediakan untuk data yang di-hash. Kekuatan keamanan fungsi-fungsi hash ini dan sistem secara keseluruhan ketika masing-masing fungsi digunakan dengan algoritme kriptografi lainnya, seperti algoritme tanda tangan digital dan kode autentikasi pesan hash yang dikunci, dapat ditemukan di [SP 800-57] dan [SP 800-107].

Selain itu, algoritme berbeda dalam hal ukuran blok dan kata data yang digunakan selama hashing atau ukuran intisari pesan. Gambar 1 menyajikan properti dasar algoritme hash ini.

Algoritma	Ukuran Pesan (sedikit)	Ukuran Blok (sedikit)	Ukuran Kata (sedikit)	Ukuran Intisari Pesan (sedikit)
SHA-1	< 264	512	32	160
SHA-224	< 264	512	32	224
SHA-256	< 264	512	32	256
SHA-384	< 2128	tahun 1024	64	384
SHA-512	< 2128	tahun 1024	64	512
SHA-512/224	< 2128	tahun 1024	64	224
SHA-512/256	< 2128	tahun 1024	64	256

Gambar 1: Properti Algoritma Hash Aman

2. DEFINISI

2.1 Glosarium Istilah dan Akronim

Sedikit	Digit biner yang memiliki nilai 0 atau 1.
Byte	Sekelompok delapan bit.
Bahasa Indonesia: FIPS	Standar Pemrosesan Informasi Federal.
NIST	Institut Nasional Standar dan Teknologi.
SHA	Algoritma Hash Aman.
Bahasa Inggris	Publikasi Khusus
Kata	Sekelompok 32 bit (4 byte) atau 64 bit (8 byte), tergantung pada algoritma hash aman.

2.2 Parameter, Simbol, dan Istilah Algoritma

2.2.1 Parameter

Parameter berikut digunakan dalam spesifikasi algoritma hash aman dalam Standar ini.

a, b, c, …, h	Variabel kerja yang merupakan kata-kata w-bit yang digunakan dalam perhitungan nilai hash, *H (i)* .
(aku h	Aku th nilai hash. *H* (0) adalah nilai hash *awal ; H (N)* adalah nilai hash *akhir* dan digunakan untuk menentukan intisari pesan.
(aku J h	*J* itu $^{\text{kata ke 1 dari } i}$ nilai hash, dimana ( ) *H* adalah kata paling kiri dari hash nilai i.
Kt	Nilai konstan yang akan digunakan untuk iterasi *t* dari perhitungan hash.
aku	Jumlah angka nol yang ditambahkan ke pesan selama langkah pengisian.
—	Panjang pesan, M, dalam bit.
M	Jumlah bit dalam blok pesan, *M (i)* .
M	Pesan yang akan di-hash.

$M^{(i)}$ Blok pesan i, dengan ukuran *m* bit.

$M_j^{(i)}$ *J* itu kata ke 1 dari *i* blok pesan, dimana $M_0^{(i)}$ adalah kata paling kiri dari blok pesan i.

N Jumlah bit yang akan diputar atau digeser saat suatu kata dioperasikan.

N Jumlah blok dalam pesan yang diberi bantalan.

T Kata w-bit sementara yang digunakan dalam perhitungan hash.

aku Jumlah bit dalam sebuah kata.

$H_t^{(i)}$ Itu *t* th w-bit kata jadwal pesan.

2.2.2 Simbol dan Operasi

Simbol-simbol berikut digunakan dalam spesifikasi algoritma hash aman; masing-masing beroperasi pada kata-kata *w*- bit.

ÿ Operasi Bitwise AND.

ÿ Operasi Bitwise OR (“inklusif-ATAU”).

ÿ Operasi Bitwise XOR (“eksklusif-ATAU”).

ÿ Operasi komplemen bitwise.

+ Penambahan modulo *2w*.

<< Operasi pergeseran ke kiri, di mana *x* << *n* diperoleh dengan membuang *n* paling kiri bit kata *x* dan kemudian menambahkan *n* angka nol di sebelah kanan pada hasilnya .

>> Operasi pergeseran ke kanan, di mana *x* >> *n* diperoleh dengan membuang *n* bit paling kanan dari kata *x* dan kemudian menambahkan *n* angka nol di sebelah kiri pada hasilnya .

Operasi berikut digunakan dalam spesifikasi algoritma hash aman:

ROTL ^{N}***(x)*** Operasi *putar ke kiri* (pergeseran kiri melingkar), di mana *x* adalah kata w-bit dan *n* adalah bilangan bulat dengan 0 ÿ *n* < w, didefinisikan oleh *ROTL* $^{N}(x)=(x << n) \text{ ÿ } x >> w - n)$. Nilai *x adalah :*

ROTR ^{N}***(x)*** Operasi *putar ke kanan* (pergeseran melingkar ke kanan), di mana *x* adalah kata w-bit dan *n* adalah bilangan bulat dengan 0 ÿ *n* < w, didefinisikan oleh $ROTR^{N}(x)=(x >> n) \text{ ÿ } x << w - n)$.

SHR ^{N}***(X)*** Operasi *pergeseran ke kanan ,* di mana *x* adalah kata w-bit dan *n* adalah bilangan bulat dengan ÿ *n* < w, didefinisikan oleh *SHR* ^{N}0 (x)=x >> n.

3. NOTASI DAN KONVENSI

String Bit dan Integer 3.1

Terminologi berikut terkait dengan string bit dan integer akan digunakan.

1. *Digit heksadesimal* adalah elemen dari himpunan {0, 1,…, 9, a,…, f}. Digit heksadesimal adalah representasi dari string 4-bit. Misalnya, digit heksadesimal “7” mewakili string 4-bit “0111”, dan digit heksadesimal “a” mewakili string 4-bit “1010”.

2. Sebuah *kata* adalah string w-bit yang dapat direpresentasikan sebagai urutan digit heksadesimal. Untuk mengubah sebuah kata menjadi digit heksadesimal, setiap string 4-bit diubah menjadi padanan digit heksadesimalnya, seperti yang dijelaskan dalam (1) di atas. Misalnya, string 32-bit

   1010 0001 0000 0011 1111 1110 0010 0011

   dapat dinyatakan sebagai “a103fe23”, dan string 64-bit

   1010 0001 0000 0011 1111 1110 0010 0011
   0011 0010 1110 1111 0011 0000 0001 1010

   dapat dinyatakan sebagai “a103fe2332ef301a”.

   Dalam seluruh spesifikasi ini, konvensi “big-endian” digunakan saat mengekspresikan kata 32- dan 64-bit, sehingga dalam setiap kata, bit paling signifikan disimpan di posisi bit paling kiri.

3. Suatu *bilangan bulat* dapat direpresentasikan sebagai sebuah kata atau sepasang kata. Representasi kata panjang pesan, ÿ dalam bit diperlukan untuk teknik padding pada Bagian 5.1.

   Bilangan bulat antara 0 dan 232 -1 *inklusif* dapat direpresentasikan sebagai kata 32-bit. Empat bit paling tidak signifikan dari bilangan bulat direpresentasikan oleh digit heksadesimal paling kanan dari representasi kata tersebut. Misalnya, bilangan bulat 291=28 + 25 + 21 + 20 =256+32+2+1 direpresentasikan oleh kata heksadesimal “00000123”.

   Hal yang sama berlaku untuk bilangan bulat antara 0 dan 264 -1 *inklusif,* yang dapat direpresentasikan sebagai kata 64-bit.

   Jika *Z* adalah bilangan bulat, 0 ÿ *Z* < 264, maka Z=232X + Y, di mana 0 ÿ *X* < 232 dan 0 ÿ *Y* < 232 Karena · *X* dan *Y* dapat direpresentasikan sebagai kata 32-bit *x* dan y, bilangan bulat *Z* dapat direpresentasikan sebagai pasangan kata *(x,* y). Properti ini digunakan untuk SHA-1, SHA-224, dan SHA-256.

Jika *Z* adalah bilangan bulat, 0 ÿ *Z* < 2128, maka Z=264X + Y, di mana 0 ÿ *X* < 264 dan 0 ÿ *Y* < 264 Karena · *X* dan *Y* dapat direpresentasikan sebagai kata 64-bit *x* dan y, bilangan bulat *Z* dapat direpresentasikan sebagai pasangan kata *(x,* y). Properti ini digunakan untuk SHA-384, SHA-512, SHA-512/224 dan SHA-512/256.

4. Untuk algoritma hash aman, ukuran *blok pesan* - *m* bit - bergantung pada algoritma.

   a) Untuk **SHA-1, SHA-224** dan **SHA-256,** setiap blok pesan memiliki **512 bit,** yang direpresentasikan sebagai urutan enam belas **kata 32-bit.**

   b) Untuk **SHA-384, SHA-512, SHA-512/224** dan **SHA-512/256** setiap blok pesan memiliki **1024 bit,** yang direpresentasikan sebagai urutan enam belas **kata 64-bit.**

3.2 Operasi pada Kata Operasi berikut

diterapkan pada kata w-bit dalam kelima algoritma hash aman. SHA-1, SHA-224, dan SHA-256 beroperasi pada kata 32-bit (w=32), dan SHA-384, SHA-512, SHA-512/224, dan SHA-512/256 beroperasi pada kata 64-bit (w=64).

1. Operasi kata *logika* bitwise : ÿ ÿ dan ÿ (lihat Pasal 2.2.2).

2. Penambahan modulo *2w* .

   Operasi *x* + *y* didefinisikan sebagai berikut. Kata *x* dan *y* mewakili bilangan bulat *X* dan Y, di mana 0 ÿ *X* < *2w* dan 0 ÿ *Y* < *2w* . Untuk bilangan bulat positif *U* dan V, misalkan *U* modV adalah sisa pembagian *U* dengan V. Hitunglah

   Z = (X + *Y)* mod *2w* ·

   Maka 0 ÿ *Z* < *2w* . Ubahlah bilangan bulat *Z* menjadi sebuah kata, z, dan definisikan *z=x* + y.

3. Operasi *pergeseran ke kanan* ***SHR*** ÿ *n* **(x),** dimana *x* adalah kata w-bit dan *n* adalah bilangan bulat dengan 0 < w, didefinisikan oleh

   SHR Tentukan nilai x dan x + n

   Operasi ini digunakan dalam algoritma SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256.

4. Operasi *putar ke kanan* (pergeseran melingkar ke kanan) ***ROTR*** dan *n* **(x),** dimana *x* adalah kata w-bit adalah bilangan bulat dengan 0 ÿ *n* < w, didefinisikan oleh

   ROTR (x)=(x >> n) ÿ *(x* << *w* - n).

Jadi, *ROTR* $^{N}(x)$ setara dengan pergeseran melingkar (rotasi) *x* sebanyak *n* posisi ke benar.

Operasi ini digunakan oleh algoritma SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256.

5. Operasi *putar ke kiri* (pergeseran kiri melingkar), ***ROTL*** adalah ^{N}**(x),** dimana *x* adalah kata w-bit dan *n* bilangan bulat dengan 0 ÿ *n* < w, didefinisikan oleh

ROTL ^{N} Persamaan (x) dan (y – w) adalah persamaan yang menyatakan hubungan terbalik aktifasi antara x dan n. Persamaan (y + w) adalah persamaan yang menyatakan hubungan terbalik aktifasi antara x dan n.

Karena itu, *ROTL* $^{N}(x)$ setara dengan pergeseran melingkar (rotasi) *x* sebanyak *n* posisi ke keluar.

Operasi ini hanya digunakan dalam algoritma SHA-1.

6. Perhatikan hubungan ekuivalensi berikut, di mana *w* ditetapkan dalam setiap hubungan:

ROTL $^{N}(x)$ ÿ *ROTR* $^{kami}(X)$

ROTR $^{N}(x)$ ÿ *PUTARAN* $^{kami}(X)$

4. FUNGSI DAN KONSTANTA

4.1 Fungsi Bagian ini

mendefinisikan fungsi-fungsi yang digunakan oleh masing-masing algoritma. Meskipun algoritma SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 semuanya menggunakan fungsi yang serupa, deskripsi mereka dipisahkan menjadi beberapa bagian untuk SHA-224 dan SHA-256 (Bagian 4.1.2) dan untuk SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 (Bagian 4.1.3), karena input dan output untuk fungsi-fungsi ini adalah kata-kata dengan ukuran yang berbeda. Masing-masing algoritma menyertakan fungsi Ch(x, y, z) dan Maj(x, y, z); operasi eksklusif-ATAU (ÿ) dalam fungsi-fungsi ini dapat digantikan oleh operasi bitwise ATAU (ÿ) dan menghasilkan hasil yang identik.

4.1.1 Fungsi SHA-1

SHA-1 menggunakan serangkaian fungsi logika, f0, f1,…, f79. Setiap fungsi ft , di mana 0 ÿ *t* ÿ 79, beroperasi pada tiga kata 32-bit, x, y, dan z, dan menghasilkan kata 32-bit sebagai output. Fungsi *ft (x,* y, z) didefinisikan sebagai berikut:

$$\textit{kaki}\ (x, y, z) = \begin{cases} \text{Persamaan } (x, y, z) = (x \ddot{y} y) \ddot{y} (\ddot{y} x \ddot{y} z) & 0 \ddot{y} t \ddot{y} 19 \\ \textit{Paritas}(x, y, z) = x \ddot{y} y \ddot{y} z & 20 \ddot{y} \textit{dan} \ddot{y} 39 \\ \text{Maj}(x, y, z) = (x \ddot{y} y) \ddot{y} (x \ddot{y} z) \ddot{y} (y \ddot{y} z) & 40 \ddot{y} \textit{dan} \ddot{y} 59 \\ \textit{Paritas}(x, y, z) = x \ddot{y} y \ddot{y} z & 60 \ddot{y} \textit{dan} \ddot{y} 79. \end{cases} \quad (4.1)$$

4.1.2 Fungsi SHA-224 dan SHA-256

SHA-224 dan SHA-256 sama-sama menggunakan enam fungsi logika, di mana *setiap fungsi beroperasi pada kata 32-bit,* yang direpresentasikan sebagai x, y, dan z. Hasil dari setiap fungsi adalah kata 32-bit baru.

$$\text{Persamaan } (x, y, z) = (x \ddot{y} y) \ddot{y} (\ddot{y} x \ddot{y} z) \quad (4.2)$$

$$\text{Maj}(x, y, z) = (x \ddot{y} y) \ddot{y} (x \ddot{y} z) \ddot{y} (y \ddot{y} z) \quad (4.3)$$

$$\ddot{y}_{-}^{\{256\}}(x) = ROTR^{2}(x) \ddot{y} ROTR\ 13(x) \ddot{y} ROTR\ 22(x) \quad (4.4)$$

$$\ddot{y}_{1}^{\{256\}}(x) = ROTR^{6}(x) \ddot{y} ROTR\ 11(x) \ddot{y} ROTR\ 25(x) \quad (4.5)$$

$$\ddot{y}_{0}^{\{256\}} \text{ Rumus } untuk \text{ ROTR } 17\ (x)\ (x) \ddot{y} ROTR\ 18(x) \ddot{y} SHR^{3}(X) \quad (4.6)$$

$$\ddot{y}^{\{256\}\ 1}\ \ddot{y} \text{ ROTR } 19\ (x) \ddot{y} SHR\ 10(x)\ ( ) \quad (4.7)$$

4.1.3 Fungsi SHA-384, SHA-512, SHA-512/224, dan SHA-512/256 Fungsi SHA-384, SHA-512, SHA-512/224, dan SHA-512/256 menggunakan enam fungsi logika, di mana *setiap fungsi beroperasi pada kata 64-bit,* yang direpresentasikan sebagai x, y, dan z. Hasil dari setiap fungsi adalah kata 64-bit baru.

$$\text{Persamaan}(x, y, z) = (x \text{ ÿ } y) \text{ÿ} (\text{ÿ}x \text{ ÿ } z) \tag{4.8}$$

$$\text{Maj}(x, y, z) = (x \text{ ÿ } y) \text{ÿ} (x \text{ ÿ } z) \text{ÿ} (y \text{ ÿ } z) \tag{4.9}$$

$$\text{ÿ}_{-}^{\{512\}}(x) = ROTR\ 28(x) \text{ ÿ } ROTR\ 34(x) \text{ ÿ } ROTR\ 39(x) \tag{4.10}$$

$$\text{ÿ}_{1}^{\{512\}}(x) = ROTR\ 14(x) \text{ ÿ } ROTR\ 18(x) \text{ ÿ } ROTR\ 41(x) \tag{4.11}$$

$$\text{ÿ}_{0}^{\{512\}}(x) = ROT^{1}(x) \text{ ÿ } ROTR^{8}(x) \text{ ÿ } SHR^{7}(X) \tag{4.12}$$

$$\text{ÿ}^{\{512\}}\ 1\ x = ROTR\ 19(x) \text{ ÿ } ROTR\ 61(x) \text{ ÿ } SHR\ (\ )^{6}(X) \tag{4.13}$$

4.2 Konstanta

4.2.1 Konstanta SHA-1

SHA-1 menggunakan urutan delapan puluh kata 32-bit konstan, K0, K1,…, K79, yang diberikan oleh

$$Kt = \begin{cases} 5a827999 & 0 \text{ ÿ } t \text{ ÿ } 19 \\ 6ed9eba1 & 20 \text{ ÿ } dan \text{ ÿ } 39 \\ 8f1bbcdc & 40 \text{ ÿ } dan \text{ ÿ } 59 \\ ca62c1d6 & 60 \text{ ÿ } dan \text{ ÿ } 79 \end{cases} \tag{4.14}$$

4.2.2 Konstanta SHA-224 dan SHA-256

SHA-224 dan SHA-256 menggunakan urutan yang sama dari enam puluh empat kata 32-bit konstan, $K_0^{\{256\}}, K_1^{\{256\}}, \text{ÿ}, K_{63}^{\{256\}}$. Kata-kata ini mewakili tiga puluh dua bit pertama dari bagian pecahan akar pangkat tiga dari enam puluh empat bilangan prima pertama. Dalam heksadesimal, kata-kata konstan ini adalah (dari kiri ke kanan)

428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2

4.2.3 Konstanta SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 Konstanta

SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 menggunakan urutan yang sama dari delapan puluh kata konstan 64-bit, $K_0^{\{512\}}, K_1^{\{512\}}, \ldots, K_{79}^{\{512\}}$. Kata-kata ini mewakili enam puluh empat bit pertama dari bagian pecahan akar pangkat tiga dari delapan puluh bilangan prima pertama. Dalam heksadesimal, kata-kata konstan ini adalah (dari kiri ke kanan)

428a2f98d728ae22 7137449123ef65cd b5c0fbcfec4d3b2f e9b5dba58189dbbc
3956c25bf348b538 59f111f1b605d019 923f82a4af194f9b ab1c5ed5da6d8118 d807aa98a3030242
12835b0145706fbe 243185be4ee4b28c 550c7dc3d5ff b4e2
72be5d74f27b896f 80deb1fe3b1696b1 9bdc06a725c71235 c19bf174cf692694
e49b69c19ef14ad2 efbe4786384f25e3 0fc19dc68b8cd5b5 240ca1cc77ac9c65 2de92c6f592b0275
4a7484aa6ea6e483 5cb0a9dcbd41fbd4 76f988da831153 b5
983e5152ee66dfab a831c66d2db43210 b00327c898fb213f bf597fc7beef0ee4
c6e00bf33da88fc2 d5a79147930aa725 06ca6351e003826f 142929670a0e6e70
27b70a8546d22ffc 2e1b21385c26c926 4d2c6dfc5ac42aed 53380d139d95b3df
650a73548baf63de 766a0abb3c77b2a8 81c2c92e47edaee6 92722c851482353b
a2bfe8a14cf10364 a81a664bbc423001 c24b8b70d0f89791 c76c51a30654be30
d192e819d6ef5218 d69906245565a910 f40e35855771202a 106aa07032bbd1b8
19a4c116b8d2d0c8 1e376c085141ab53 2748774cdf8eeb99 34b0bcb5e19b48a8
391c0cb3c5c95a63 4ed8aa4ae3418acb 5b9cca4f7763e373 682e6ff3d6b2b8a3
748f82ee5defb2fc 78a5636f43172f60 84c87814a1f0ab72 8cc702081a6439ec
90befffa23631e28 a4506cebde82bde9 bef9a3f7b2c67915 c67178f2e372532b
ca273eceea26619c d186b8c721c0c207 eada7dd6cde0eb1e f57d4f7fee6ed178
06f067aa72176fba 0a637dc5a2c898a6 113f9804bef90dae 1b710b35131c471b
28db77f523047d84 32caab7b40c72493 3c9ebe0a15c9bebc 431d67c49c100d4c
4cc5d4becb3e42b6 597f299cfc657e2a 5fcb6fab3ad6faec 6c44198c4a475817

5. PEMROSESAN AWAL

Praproses terdiri dari tiga langkah: padding pesan, *M* (Bagian 5.1), parsing pesan ke dalam blok pesan (Bagian 5.2), dan pengaturan nilai hash awal, *H* (0) (Bagian 5.3).

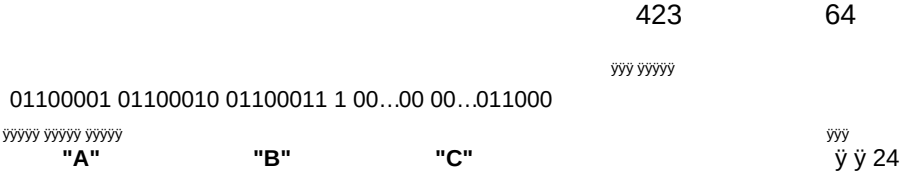
5.1 Menambahkan Pesan

Tujuan dari padding ini adalah untuk memastikan bahwa pesan yang dipadding merupakan kelipatan 512 atau 1024 bit, tergantung pada algoritmanya. Padding dapat disisipkan sebelum komputasi hash dimulai pada sebuah pesan, atau kapan saja selama komputasi hash sebelum memproses blok yang akan berisi padding.

5.1.1 SHA-1, SHA-224 dan SHA-256

Misalkan panjang pesan, M, adalah ÿ bit. Tambahkan bit “1” ke akhir pesan, diikuti oleh *k* bit nol, di mana *k* adalah solusi non-negatif terkecil untuk persamaan ÿ ÿ1ÿ *k* ÿ 448mod512 . Kemudian tambahkan blok 64-bit yang sama dengan angka ÿ yang diekspresikan menggunakan representasi biner. Misalnya, pesan (8-bit ASCII) **“abc”** memiliki panjang 8ÿ3 ÿ 24

jadi pesannya diisi dengan satu bit, kemudian 448 ÿ (24 ÿ1) ÿ 423 bit nol, dan kemudian panjang pesan, menjadi pesan berlapis 512-bit

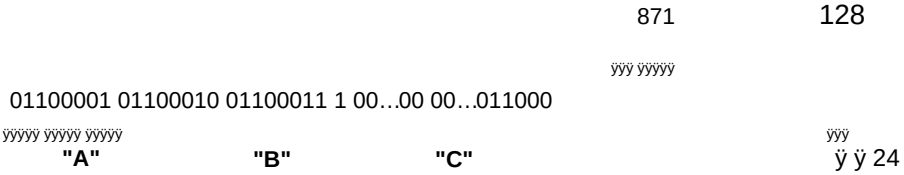


Panjang pesan yang diberi bantalan sekarang harus kelipatan 512 bit.

5.1.2 SHA-384, SHA-512, SHA-512/224 dan SHA-512/256

Misalkan panjang pesan M, dalam bit, adalah ÿ bit. Tambahkan bit “1” ke akhir pesan, diikuti oleh *k* bit nol, di mana *k* adalah solusi non-negatif terkecil untuk persamaan ÿ ÿ1ÿ *k* ÿ 896mod1024 . Kemudian tambahkan blok 128-bit yang sama dengan angka ÿ yang diekspresikan menggunakan representasi biner. Misalnya, pesan (8-bit ASCII) **“abc”** memiliki panjang 8ÿ3 ÿ 24

jadi pesannya diisi dengan satu bit, kemudian 896 ÿ (24 ÿ1) ÿ 871 bit nol, dan kemudian panjang pesan, menjadi pesan berlapis 1024-bit



Panjang pesan yang diberi bantalan sekarang harus kelipatan 1024 bit.

5.2 Parsing Pesan

Pesan dan bantalannya harus diurai menjadi blok *N* m-bit.

5.2.1 SHA-1, SHA-224 dan SHA-256

Untuk SHA-1, SHA-224 dan SHA-256, pesan dan padding-nya diurai menjadi *N* blok 512-bit, *M* (1) $M^{(2)}$…, $M^{(N)}$ Karena 512 bit dari blok input dapat dinyatakan sebagai enam belas bit 32-kata bit, 32 bit pertama dari blok pesan *i* dilambangkan $M^{(i)}_0$ 32 bit berikutnya adalah $M^{(i)}_1$ dan sebagainya sampai ke atas $M^{(i)}_{15}$.

5.2.2 SHA-384, SHA-512, SHA-512/224 dan SHA-512/256

Untuk SHA-384, SHA-512, SHA-512/224 dan SHA-512/256, pesan dan padding-nya diurai menjadi *N* blok 1024-bit, *M* (1) $M^{(2)}$…, $M^{(N)}$ Karena 1024 bit dari blok input dapat dinyatakan sebagai enam belas kata 64-bit, 64 bit pertama dari blok pesan *i* dilambangkan $M^{(i)}_0$ selanjutnya 64 bit adalah $M^{(i)}_1$ dan seterusnya sampai $M^{(i)}_{15}$.

5.3 Menetapkan Nilai Hash Awal *(H* (0))

Sebelum perhitungan hash dimulai untuk setiap algoritma hash yang aman, nilai hash awal, *H* (0), harus ditetapkan. Ukuran dan jumlah kata dalam *H* (0) bergantung pada ukuran intisari pesan.

5.3.1 SHA-1

Untuk SHA-1, nilai hash awal, *H* (0), harus terdiri dari lima kata 32-bit berikut, dalam hex:

$$H^{(0)}_0 = 67452301$$
$$H^{(0)}_1 = efcdab89$$
$$H^{(0)}_2 = 98badcfe$$
$$H^{(0)}_3 = 10325476$$
$$H^{(0)}_4 = c3d2e1f0$$

5.3.2 SHA-224

Untuk SHA-224, nilai hash awal, *H* (0), harus terdiri dari delapan kata 32-bit berikut, dalam heksagonal:

$$H^{(0)}_0 = c1059ed8$$
$$H^{(0)}_1 = 367cd507$$
$$H^{(0)}_2 = 3070dd17$$
$$H^{(0)}_3 = f70e5939$$
$$H^{(0)}_4 = ffc00b31$$
$$H^{(0)}_5 = 68581511$$
$$H^{(0)}_6 = 64f98fa7$$

$H7^{)0(} = befa4fa4$

5.3.3 SHA-256

Untuk SHA-256, nilai hash awal, *H* (0), harus terdiri dari delapan kata 32-bit berikut, dalam heksagonal:

Nilai $H0^{(0)} = 6a09e667$

$H1^{)0(} = bb67ae85$

$H2^{(0)} = 3c6ef372$

$H3^{(0)} = a54ff53a$

$H4^{(0)} = 510e527f$

$H5^{(0)} = 9b05688c$

$H6^{(0)} = 1f83d9ab$

$H7^{(0)} = 5be0cd19$

Kata-kata ini diperoleh dengan mengambil tiga puluh dua bit pertama dari bagian pecahan akar kuadrat delapan bilangan prima pertama.

5.3.4 SHA-384

Untuk SHA-384, nilai hash awal, *H* (0), harus terdiri dari delapan kata 64-bit berikut, dalam heksagonal:

$H0^{)0(} = cbbb9d5dc1059ed8$

$H1^{)0(} = 629a292a367cd507$

$H2^{)0(} = 9159015a3070dd17$

$H3^{)0(} = 152fecd8f70e5939$

$H4^{)0(} = 67332667ffc00b31$

$H5^{)0(} = 8eb44a8768581511$

$H6^{)0(} = db0c2e0d64f98fa7$

$H7^{)0(} = 47b5481dbefa4fa4$

Kata-kata ini diperoleh dengan mengambil enam puluh empat bit pertama dari bagian pecahan akar kuadrat bilangan prima kesembilan hingga keenam belas.

5.3.5 SHA-512

Untuk SHA-512, nilai hash awal, *H* (0), harus terdiri dari delapan kata 64-bit berikut, dalam heksagonal:

$H0^{)0(} = 6a09e667f3bcc908$

$H1^{)0(} = bb67ae8584caa73b$

H2(0) = 3c6ef372fe94f82b
H3(0) = a54ff53a5f1d36f1
H4(0) = 510e527pudar682d1
H5(0) = 9b05688c2b3e6c1f
H6(0) = 1f83d9abfb41bd6b
H7(0) = 5be0cd19137e2179

Kata-kata ini diperoleh dengan mengambil enam puluh empat bit pertama dari bagian pecahan akar kuadrat delapan bilangan prima pertama.

5.3.6 SHA-512/t

“SHA-512/t” adalah nama umum untuk fungsi hash t-bit berdasarkan SHA-512 yang outputnya dipotong menjadi *t* bit. Setiap fungsi hash memerlukan nilai hash awal yang berbeda. Bagian ini menyediakan prosedur untuk menentukan nilai awal untuk SHA-512/ *t* untuk nilai t tertentu.

Untuk SHA-512/t, *t* adalah bilangan bulat positif tanpa nol di depan sehingga *t* < 512, dan *t* bukan 384.
Misalnya: *t* adalah 256, tetapi bukan 0256, dan “SHA-512/t” adalah “SHA-512/256” (string ASCII sepanjang 11 karakter), yang setara dengan 53 48 41 2D 35 31 32 2F 32 35 36 dalam heksadesimal.

Nilai hash awal untuk SHA-512/t, untuk nilai t tertentu, akan dihasilkan oleh *SHA-512/t*
Fungsi Generasi IV di bawah.

Fungsi Generasi SHA-512/t IV

(mulai:)

Menunjukkan $H^{(0)}$ menjadi nilai hash awal SHA-512 sebagaimana ditentukan dalam Bagian 5.3.5 di atas.

Menunjukkan $H^{(0)}$ menjadi nilai hash awal yang dihitung di bawah ini.

H (0) adalah IV untuk SHA-512/t.

Untuk *i* = 0 sampai 7

{
$Hai^{(0)}$ = *Hai*(0) a5a5a5a5a5a5a5a5(dalam heksa).
}

H (0) = SHA-512 (“SHA-512/t”) menggunakan $H^{(0)}$ sebagai IV, di mana *t* adalah nilai pemotongan spesifik.

(akhir.)

SHA-512/224 *(t* = 224) dan SHA-512/256 *(t* = 256) merupakan algoritma hash **yang disetujui .** Algoritma hash SHA-512/t lainnya dengan nilai *t* yang berbeda dapat ditetapkan dalam [SP 800-107] di masa mendatang jika diperlukan. Berikut ini adalah IV untuk SHA-512/224 dan SHA-512/256.

5.3.6.1 SHA-512/224

Untuk SHA-512/224, nilai hash awal, *H* (0), harus terdiri dari delapan kata 64-bit berikut,

dalam heksadesimal:

$$H_0^{(0)} = 8C3D37C819544DA2$$
$$H_1^{(0)} = 73E1996689DCD4D6$$
$$H_2^{(0)} = 1DFAB7AE32FF9C82$$
$$H_3^{(0)} = 679DD514582F9FCF$$
$$H_4^{(0)} = 0F6D2B697BD44DA8$$
$$H_5^{(0)} = 77E36F7304C48942$$
$$H_6^{(0)} = 3F9D85A86A1D36C8$$
$$H_7^{(0)} = 1112E6AD91D692A1$$

Kata-kata ini diperoleh dengan menjalankan *Fungsi Generasi IV SHA-512/t* dengan *t* = 224.

5.3.6.2 SHA-512/256

Untuk SHA-512/256, nilai hash awal, *H* (0), harus terdiri dari delapan kata 64-bit berikut,

dalam heksadesimal:

$$H_0^{(0)} = 22312194FC2BF72C$$
$$H_1^{(0)} = 9F555FA3C84C64C2$$
$$H_2^{(0)} = 2393B86B6F53B151$$
$$H_3^{(0)} = 963877195940EABD$$
$$H_4^{(0)} = 96283EE2A88EFFE3$$
$$H_5^{(0)} = BE5E1E2553863992$$
$$H_6^{(0)} = 2B0199FC2C85B8AA$$
$$H_7^{(0)} = 0EB72DDC81C52CA2$$

Kata-kata ini diperoleh dengan menjalankan *Fungsi Generasi IV SHA-512/t* dengan *t* = 256.

6. ALGORITMA HASH YANG AMAN

Pada bagian berikut, algoritme hash tidak dijelaskan dalam urutan ukuran menaik. SHA-256 dijelaskan sebelum SHA-224 karena spesifikasi untuk SHA-224 identik dengan SHA-256, kecuali bahwa nilai hash awal yang digunakan berbeda, dan nilai hash akhir dipotong menjadi 224 bit untuk SHA-224. Hal yang sama berlaku untuk SHA-512, SHA-384, SHA-512/224, dan SHA-512/256, kecuali bahwa nilai hash akhir dipotong menjadi 224 bit untuk SHA-512/224, 256 bit untuk SHA-512/256, atau 384 bit untuk SHA-384.

Untuk setiap algoritma hash aman, mungkin ada metode komputasi alternatif yang menghasilkan hasil identik; salah satu contohnya adalah komputasi SHA-1 alternatif yang dijelaskan dalam Bagian 6.1.3.
Metode alternatif tersebut dapat diterapkan sesuai dengan standar ini.

6.1 SHA-1

SHA-1 dapat digunakan untuk melakukan hash pada pesan, M, yang memiliki panjang ÿ bit, di mana ÿ ÿÿ 20 . Algoritma tersebut menggunakan 1) jadwal pesan yang terdiri dari delapan puluh kata 32-bit, 2) lima variabel kerja yang masing-masing terdiri dari 32 bit, dan 3) nilai hash yang terdiri dari lima kata 32-bit. Hasil akhir dari SHA-1 adalah intisari pesan 160-bit.

64

Kata-kata dari jadwal pesan diberi label W0, W1,…, W79. Lima variabel kerja adalah *i* ( )

diberi label a, b, c, d, dan e. Kata-kata dari nilai hash diberi label *HH* 0 , berisi nilai hash awal, *H* (0), digantikan oleh setiap nilai hash perantara berikutnya Saya ( ) 1 , H Saya ( ) 4 yang akan

(setelah setiap blok pesan diproses), *H (i)*

dan diakhiri dengan nilai hash akhir, *H (N)* SHA-1 juga menggunakan satu kata sementara, T.

6.1.1 Praproses SHA-1

1. Tetapkan nilai hash awal, *H* (0), seperti yang ditentukan dalam Pasal 5.3.1.

2. Pesan tersebut dipadatkan dan diurai sebagaimana ditentukan dalam Bagian 5.

6.1.2 Perhitungan Hash SHA-1

Perhitungan hash SHA-1 menggunakan fungsi dan konstanta yang sebelumnya didefinisikan dalam Pasal 4.1.1 dan Pasal 4.2.1.
Penambahan (+) dilakukan modulo 232 .

Setiap blok pesan, *M* (1) Pria (2)…, ... M N) diproses secara berurutan, menggunakan langkah-langkah berikut:

Untuk i=1 sampai N:

{

1. Siapkan jadwal pesan, *{Wt}:*

$$Berat = \begin{cases} \text{(Saya) Gunung} & 0 \text{ ÿ } sampai \text{ ÿ}15 \\ ROTL1\ (\text{Berat-3 ÿ } Berat\text{-}8 \text{ ÿ Berat-14 ÿ Berat-16 }) & 16 \text{ ÿ } t \text{ ÿ } 79 \end{cases}$$

2. Inisialisasikan lima variabel kerja, a, b, c, d, dan e, dengan nilai hash (i-1) :

sebuah H (1) *saya*

bH *saya* (1) 1

c H *saya* (1) 2

d rumah (1) 3

dan H (1) 4

3. Untuk t=0 sampai 79:

{

T ROTL dari 5 ( ) ÿ *T* (*K t*) *Wt cde* ÿ ÿ ÿ

diedit

arus searah

c ROTL b ( ) 3 0

kembali

pada

}

4. Hitunglah *i* th nilai hash antara *H (i)* :

HaH Saya ÿÿ *saya)1(*

HbH Saya ÿÿ (1 *saya)1(1*

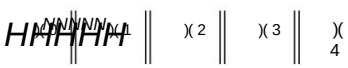
HcH (ÿÿ)1(2 2

HdH Saya ÿÿ 3)1(aku 3

Hehe Saya ÿÿ 4 *saya)1(* 4

}

Setelah mengulang langkah satu sampai empat sebanyak *N* kali (yaitu, setelah memproses *M (N)),* intisari pesan 160-bit yang dihasilkan dari pesan, M, adalah



6.1.3 Metode Alternatif untuk Menghitung Intisari Pesan SHA-1 Metode perhitungan hash

SHA-1 yang dijelaskan dalam Pasal 6.1.2 mengasumsikan bahwa jadwal pesan W0, W1,…, W79 diimplementasikan sebagai array dari delapan puluh kata 32-bit. Hal ini efisien dari sudut pandang meminimalkan waktu eksekusi, karena alamat Wt-3,…, Wt-16 pada langkah (2) Pasal 6.1.2 mudah dihitung.

Namun, jika memori terbatas, alternatifnya adalah menganggap *{Wt}* sebagai antrean melingkar yang dapat diimplementasikan menggunakan larik enam belas kata 32-bit, W0, W1,…, W15. Metode alternatif yang dijelaskan dalam bagian ini menghasilkan intisari pesan yang sama seperti metode komputasi SHA-1 yang dijelaskan dalam Bagian 6.1.2. Meskipun metode alternatif ini menghemat enam puluh empat kata 32-bit, metode ini cenderung memperpanjang waktu eksekusi karena meningkatnya kompleksitas komputasi alamat untuk *{Wt}* pada langkah (3).

Untuk metode SHA-1 alternatif ini, misalkan MASK=0000000f (dalam heksadesimal). Seperti pada Bagian 6.1.1, penjumlahan dilakukan modulo 232. Dengan asumsi bahwa praproses seperti yang dijelaskan pada Bagian 6.1.1 telah dilakukan, pemrosesan *M (i)* adalah sebagai berikut:

Untuk i=1 sampai N:

{

1. Untuk t=0 sampai 15:

   {

   saya)(ÿ *MW*
   tidak ada

   }

2. Inisialisasikan lima variabel kerja, a, b, c, d, dan e, dengan nilai hash (i-1) :

   Ha—)1(

   Hb—)1(1

   Hc— *saya)1(*
   2

   HD (tinggi) *saya)1(3*

   Dia— *saya)1(4*

3. Untuk t=0 sampai 79:

   {

   ÿÿ *MASKER*

Jika *t* ÿ16 maka

{

ÿ *WROTLW* ( ) (*ÿÿ MASKER* *ÿKE* *ÿÿ MASKER* *ÿKE* *ÿÿ MASKER* *ÿWs*)

}

WKedcbfaROTLT ÿ ÿÿÿÿ Bahasa Inggris

CD

BROTLc adalah singkatan dari *bpotlc.*

di atas

Ta

}

4. Hitunglah *i* th nilai hash antara *H (i)* :

HaHa ÿÿ

HbH ÿÿ

HcH ÿÿ

HdH ÿÿ

Hehe ÿÿ

}

Setelah mengulang langkah satu sampai empat sebanyak *N* kali (yaitu, setelah memproses *M (N)),* hasil Intisari pesan 160-bit dari pesan, M, adalah

HHHHH

6.2 SHA-256

ÿ ÿÿ 20 64

SHA-256 dapat digunakan untuk melakukan hash pada pesan, M, yang memiliki panjang ÿ bit, di mana . Algoritme tersebut menggunakan 1) jadwal pesan yang terdiri dari enam puluh empat kata 32-bit, 2) delapan variabel kerja yang masing-masing terdiri dari 32 bit, dan 3) nilai hash yang terdiri dari delapan kata 32-bit. Hasil akhir dari SHA-256 adalah intisari pesan 256-bit.

Kata-kata dari jadwal pesan diberi label W0, W1,…, W63. Delapan variabel kerja diberi label a, b, c, d, e, f, g, dan h. Kata-kata dari nilai hash diberi label yang akan menampung nilai hash awal, *H* (0), digantikan oleh setiap nilai hash perantara yang berurutan .

HHH *aku aku aku*

(setelah setiap blok pesan diproses), *H (i)* dan diakhiri dengan nilai hash akhir, *H (N)* 256 juga menggunakan dua kata SHA- . sementara, T1 dan T2.

6.2.1 Praproses SHA-256

1. Tetapkan nilai hash awal, *H* (0), seperti yang ditentukan dalam Pasal 5.3.3.

2. Pesan tersebut dipadatkan dan diurai sebagaimana ditentukan dalam Bagian 5.

6.2.2 Perhitungan Hash SHA-256 Perhitungan hash

SHA-256 menggunakan fungsi dan konstanta yang sebelumnya didefinisikan dalam Pasal 4.1.2 dan Pasal 4.2.2. Penambahan (+) dilakukan modulo 232 .

Setiap blok pesan, *M* (1) *Pria* (2)…, ... *M N)* diproses secara berurutan, menggunakan langkah-langkah berikut:

Untuk i=1 sampai N:

{

1. Siapkan jadwal pesan, *{Wt}:*

$$Berat = \begin{cases} \overset{(Saya)}{Gunung} & 0 \ddot{y} \text{ sampai } \ddot{y}15 \\ \ddot{y}\ {}^{\{256\}}{}_{1}(Berat_{2}) \ddot{y} Berat \ddot{y}\ {}_{7} \ddot{y}\ {}^{\{256\}}{}_{0}(Berat_{15}) \ddot{y} Berat \ddot{y}\ {}_{16} & 16 \ddot{y} t \ddot{y} 63 \end{cases}$$

2. Inisialisasi delapan variabel kerja, a, b, c, d, e, f, g, dan h, dengan hash (i-1)st nilai:

sebuah H$^{(1)\,0}$

bH$^{(1)\,1}$

c H $^{saya\,(1)}_{2}$

d rumah $^{saya}_{(1)\,3}$

dan H $^{saya}_{(1)\,4}$

f H $^{(1)}_{5}$

g-H $^{saya\,(1)}_{6}$

jam $^{saya}_{(1)\,7}$

3. Untuk t=0 sampai 63:

{

T h ÿ ÿ ÿ {256} ( ) (, ÿ) *dan Ch efg Kt* ÿ {256} ÿ *Berat*

T ÿ {256} ( ) (, *abc besar ,)*

hggff

diedit T ÿ ÿ

arus searah

bahasa inggris

kembali

sebuah TT ÿ ÿ

}

4. Hitunglah *i* th nilai hash antara *H (i)* :

H a H ÿ ÿ *saya* (1)

Hb Hb Hb ÿ ÿ *saya* (1)

H c H ÿ ÿ (1)

H dan H ÿ ÿ *saya* (1)

H e H ÿ ÿ *saya* (1)

HfH *saya* ÿ ÿ *saya* (1)

Hg H ÿ ÿ (1)

Hh H ÿ ÿ (1) *saya*

}

Setelah mengulang langkah satu sampai empat sebanyak *N* kali (yaitu, setelah memproses *M (N)),* intisari pesan 256-bit yang dihasilkan dari pesan tersebut, M, adalah

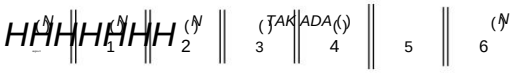
Tidak *N* *NNNNN*
HHHHHHHH ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )

6.3 SHA-224

SHA-224 dapat digunakan untuk melakukan hash pada pesan, *M,* yang memiliki panjang ÿ bit, di mana ÿ ÿÿ 20 64. Fungsi ini didefinisikan dengan cara yang sama persis seperti SHA-256 (Bagian 6.2), dengan dua pengecualian berikut:

1. Nilai hash awal, *H* (0), harus ditetapkan sebagaimana ditentukan dalam Pasal 5.3.2; dan

2. Intisari pesan 224-bit diperoleh dengan memotong nilai hash akhir, *H(N),* ke 224 bit paling kiri:



6.4 SHA-512

SHA-512 dapat digunakan untuk melakukan hash pada pesan, M, yang memiliki panjang ÿ bit, di mana . Algoritme tersebut menggunakan 1) jadwal pesan yang terdiri dari delapan puluh kata 64-bit, 2) delapan variabel kerja yang masing-masing terdiri dari 64 bit, dan 3) nilai hash yang terdiri dari delapan kata 64-bit. Hasil akhir dari SHA-512 adalah intisari pesan 512-bit.

Kata-kata dari jadwal pesan diberi label W0, W1,…, W79. Delapan variabel kerja diberi label a, b, c, d, e, f, g, dan h. Kata-kata dari nilai hash diberi label yang akan menampung nilai hash awal, *H* (0), digantikan oleh setiap nilai hash antara yang berurutan (setelah setiap blok pesan diproses), *H (i)* dan diakhiri dengan nilai hash akhir, *H (N)* SHA-512 juga menggunakan dua kata sementara, T1 dan T2.

6.4.1 Praproses SHA-512

1. Tetapkan nilai hash awal, *H* (0), seperti yang ditentukan dalam Pasal 5.3.5.

2. Pesan tersebut dipadatkan dan diurai sebagaimana ditentukan dalam Bagian 5.

6.4.2 Perhitungan Hash SHA-512

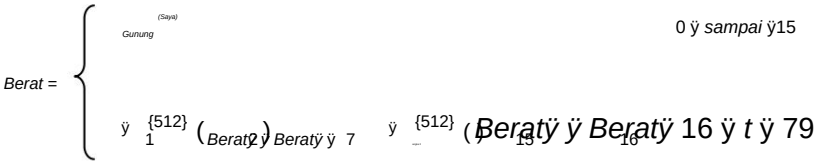
Perhitungan hash SHA-512 menggunakan fungsi dan konstanta yang sebelumnya didefinisikan dalam Pasal 4.1.3 dan Pasal 4.2.3. Penambahan (+) dilakukan modulo 264 .

Setiap blok pesan, *M* (1) diproses secara berurutan, menggunakan langkah-langkah berikut:

Untuk i=1 sampai N:

{

1. Siapkan jadwal pesan, *{Wt}:*



2. Inisialisasi delapan variabel kerja, a, b, c, d, e, f, g, dan h, dengan hash (i-1)st nilai:

sebuah H *saya* (1) 0

bH *saya* (1) 1

huruf H (1) 2

d rumah (1) 3

(1) *e H* *saya* 4

f H *saya* (1) 5

g-H *saya* (1) 6

jam (1) 7

3. Untuk t=0 sampai 79:

{

T h ÿ ÿ ÿ {512} ( ) (,) *dan Ch efg Kt* ÿ {512} ÿ *Berat*

T ÿ {512} ( ) *Mayor abc* (,)

sangat

senang T

ÿ ÿ

1

arus searah

bahasa Inggris

kembali

sebuah TT ÿ ÿ

}

4. Hitunglah *i* th nilai hash antara *H (i)* :

H a ÿ ÿ (1)

H b ÿ ÿ (1) 1

H c ÿ ÿ *saya* (1) 2

H dan H ÿ ÿ *saya* (1) 3

H e ÿ ÿ *saya* (1) 4

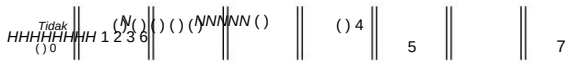
HfH ÿ ÿ (1) 5

Hg H ÿ ÿ *saya* (1) 6

Hh H ÿ ÿ *saya* (1) 7

}

Setelah mengulang langkah satu sampai empat sebanyak *N* kali (yaitu, setelah memproses *M (N)),* intisari pesan 512-bit yang dihasilkan dari pesan, M, adalah

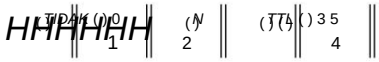


6,5 SHA-384

SHA-384 dapat digunakan untuk melakukan hash pada pesan M yang memiliki panjang ÿ bit, di mana algoritma^{128}. Itu 0 ÿ ÿ ÿ 2 didefinisikan dengan cara yang sama persis dengan SHA-512 (Bagian 6.4), dengan dua pengecualian berikut:

1. Nilai hash awal, *H* (0), harus ditetapkan sebagaimana ditentukan dalam Pasal 5.3.4; dan

2. Intisari pesan 384-bit diperoleh dengan memotong nilai hash akhir, *H (N)* untuk itu 384 bit paling kiri:



6.6 SHA-512/224

SHA-512/224 dapat digunakan untuk melakukan hash pada pesan M yang memiliki panjang ÿ bit, di mana0 ÿ ÿ ÿ 2^{128}. Algoritma ini didefinisikan dengan cara yang sama persis dengan SHA-512 (Bagian 6.4), dengan dua pengecualian berikut:

1. Nilai hash awal, *H* (0), harus ditetapkan sebagaimana ditentukan dalam Pasal 5.3.6.1; dan

2. Intisari pesan 224-bit diperoleh dengan memotong nilai hash akhir, *H (N)* untuk itu paling kiri 224 bit.

6.7 SHA-512/256

SHA-512/256 dapat digunakan untuk melakukan hash pada pesan M yang memiliki panjang ÿ bit, di mana0 ÿ ÿ ÿ 2^{128}. Algoritma ini didefinisikan dengan cara yang sama persis dengan SHA-512 (Bagian 6.4), dengan dua pengecualian berikut:

1. Nilai hash awal, *H* (0), harus ditetapkan sebagaimana ditentukan dalam Pasal 5.3.6.2; dan

2. Intisari pesan 256-bit diperoleh dengan memotong nilai hash akhir, *H (N)* untuk itu 256 bit paling kiri.

Nomor bilingual 7.

PEMOTONGAN RINGKASAN PESAN

Beberapa aplikasi mungkin memerlukan fungsi hash dengan panjang intisari pesan yang berbeda dari yang disediakan oleh fungsi hash dalam Standar ini. Dalam kasus seperti itu, intisari pesan yang terpotong dapat digunakan, di mana fungsi hash dengan panjang intisari pesan yang lebih besar diterapkan pada data yang akan di-hash, dan intisari pesan yang dihasilkan dipotong dengan memilih sejumlah bit paling kiri yang sesuai. Untuk panduan tentang memilih panjang intisari pesan yang terpotong dan informasi tentang implikasi keamanannya untuk aplikasi kriptografi yang menggunakannya, lihat SP 800-107 [SP 800-107].

LAMPIRAN A: Informasi Tambahan

A.1 Keamanan Algoritma Hash Aman Keamanan dari lima algoritma hash, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 dibahas dalam [SP 800-107].

A.2 Catatan Implementasi Contoh SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 tersedia di http://csrc.nist.gov/groups/ST/toolkit/examples.html.

A.3 Pengidentifikasi Objek

Pengidentifikasi objek (OID) untuk algoritma SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 dan SHA-512/256 diposting di http://csrc.nist.gov/groups/ST/crypto_apps_infra/csor/algorithms.html.

LAMPIRAN B: REFERENSI

[FIPS 180-3] NIST, Publikasi Standar Pemrosesan Informasi Federal 180-3, *Standar Hash Aman (SHS),* Oktober 2008.

[SP 800-57] Publikasi Khusus NIST (SP) 800-57, Bagian 1, *Rekomendasi untuk Manajemen Kunci: Umum,* (Draf) Mei 2011.

[SP 800-107] Publikasi Khusus NIST (SP) 800-107, *Rekomendasi untuk Aplikasi Menggunakan Algoritma Hash yang Disetujui,* (Direvisi), (Draf) September 2011.

LAMPIRAN C: Perubahan Teknis dari FIPS 180-3

1. Dalam FIPS 180-3, padding dimasukkan sebelum perhitungan hash dimulai. FIPS 140-4 menghilangkan batasan ini. Padding dapat dimasukkan sebelum perhitungan hash dimulai atau kapan saja selama perhitungan hash sebelum memproses blok pesan yang berisi padding.

2. FIPS 180-4 menambahkan dua algoritma tambahan: SHA-512/224 dan SHA-512/256 ke Standar dan metode untuk menentukan nilai awal untuk SHA-512/t untuk nilai t tertentu.

RALAT

Perubahan berikut telah dimasukkan ke dalam FIPS 180-4, pada tanggal yang ditunjukkan dalam tabel.

JENIS TANGGAL		MENGUBAH	NOMOR HALAMAN
5/9/2014	Perubahan Editorial	*“t* < 79” menjadi *“t* ÿ 79”	Halaman 10, Bagian 4.1.1, Baris 1