CRYPTOGRAPHIC HASH FUNCTION DAN PENGGUNAANNYA DALAM *DIGITAL SIGNATURE*

Aditia Dwiperdana – NIM : 13505014

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15014@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang *cryptographic hash function* dan penggunaannya dalam *digital signature*. *Cryptographic hash function* adalah sebuah algoritma yang digunakan untuk menghasilkan *fingerprint* atau sidik jari dari suatu string biner. Sidik jari inilah yang nantinya akan digunakan untuk memeriksa integritas data (dengan menggunakan *digital signature*). Jika data yang diterima memiliki sidik jari yang berbeda maka kemungkinan data tersebut salah atau terjadi kesalahan dalam transmisi data.

Berbagai jenis dan versi dari *cryptographic hash function* telah dibuat dan digunakan secara umum, seperti MD4, MD5, SHA-0, SHA-1, RIPEMD, dan lain-lain. Sebagian dari algoritma yang telah disebutkan sudah tidak digunakan lagi. Sebagian besar sudah dianggap tidak aman karena dapat terjadi bentrokan atau *collision*, yaitu dua masukan yang berbeda menghasilkan nilai hash yang sama. Karena itu pada makalah ini akan dibahas mengenai *Secured Hash Algorithm* (SHA)-256, *Message Authentification Code* (MAC) *Hash Message Authentification Code* (HMAC), tiga buah fungsi hash yang 'aman' dan masih umum digunakan.

Untuk menjaga integritas suatu data, diciptakan suatu mekanisme yang disebut *digital signature*. Digital signature adalah kode khusus yang dihasilkan dari fungsi penghasil *digital signature*, dimana setiap file akan memiliki kode yang berbeda. Walau hanya berbeda satu bit, kode yang dihasilkan akan jauh berbeda (disebut sebagai *avalanche effect*), jadi tidak ada kata 'mirip' atau 'hampir sama' dalam pemeriksaan integritas data. Untuk menghasilkan *digital signature* tersebut digunakan fungsi hash tanpa kunci yang digabung dengan fungsi enkripsi. Penjelasan mengenai *digital signature* akan dijelaskan pada bagian akhir dari makalah ini.

Kata Kunci : Fungsi Hash, *Cryptographic Hash Function*, *digest*, *preimage resistant*, *second preimage resistant*, *collision*, *avalanche effect, SHA-256, MAC, HMAC*, *digital signature*.

1. Pendahuluan

Dalam pengiriman data dan penyimpanan data dibutuhkan suatu proses untuk menjaga integritas atau keutuhan data. Selain dengan enkripsi dan dekripsi, dapat juga dengan menggunakan *digital signature*. Metoda digital signature berbeda dengan enkripsi dekripsi, karena proses untuk menghasilkan *digital signature* tidak mengubah data asli (*plaintext*) menjadi kode rahasia (*chipertext*), tapi menghasilkan suatu kode dengan panjang tertentu (biasanya pendek, sekitar 160-512 bit), sehingga tidak memerlukan proses seperti dekripsi.

Pemeriksaan *digital signature* dilakukan dengan memeriksa *signature* yang diterima dengan *signature* baru yang didapat dari data yang diterima. Jika ada perbedaan satu bit saja, maka data tesebut dalah sata yang salah atau terjadi kesalahan dalam proses pengiriman. Hal ini dimungkinkan karena fungsi penghasil *digital signature* umumnya adalah fungsi satu arah atau *one way function* dan tidak dapat dibalik. Salah satu algoritma yang digunakan untuk menghasilkan *digital signature* adalah hash.

2. Fungsi Hash

2.1 Definisi hash

> *"hash* /hæʃ/ n *1 [U] cooked chopped meat. 2 (idm) make a hash of sth (infmrl) do sth badly." ~Oxford Learner's Pocket Dictionary*

Secara harfiah, hash berarti daging cincang yang dipanggang. Istilah hash muncul karena proses dalam algoritma tersebut mirip dengan proses 'cincang dan olah'. Orang yang dianggap pertama kali menggunakan istilah hash adalah Hans Peter Luhn dari IBM dalam sebuah memo pada tahun 1953. Penggunaan istilah hash asendiri dimaulai sekitar sepuluh tahun setelahnya.

Fungsi hash atau algoritma hash adalah suatu metoda yang menghasilkan suatu kode atau 'sidik jari' dari sebuah data. Fungsi ini memecah dan mengolah data untuk menghasilkan kode atau nilai hashnya. Nilai hash dari suatu fungsi hash memiliki panjang yang tetap (misalkan 128-bit untuk MD5, lihat tabel 1) untuk masukan dengan panjang sembarang.

Masukan	**Nilai hash (MD5)**
hello, world	22c3683b094136c3 3938391ae71b20f04
this is cleartext that anybody can easily read without the key used by encryption. It's also bigger than the box of text above.	bd18d50263b01456 f22e3ff0d003bf66

Tabel 1 Perbandingan panjang nilai hash menggunakan algoritma MD5

Fungsi hash yang digunakan dalam bidang keamanan dan enkripsi biasa disebut *cyrptographic hash function*, untuk membedakan dengan fungsi hash yang digunakan dalam pengaturan memori komputer. Tapi karena tidak membahas fungsi hash untuk pengeturan memori, untuk selanjutnya *cyrptographic hash function* akan disebut sebagai fungsi hash saja.

Fungsi hash dengan domain panjang masukan tak hingga bukanlah fungsi satu-ke-satu, melainkan banyak-ke-satu, sehingga tidak dapat dibalik. Tapi dengan membatasi domain panjang masukan, dapat dibuat fungsi hash yang merupakan fungsi satu-ke-satu.

Karena bukan fungsi satu-ke-satu, maka dapat terjadi bentrokan atau *collision* dalam hash, yaitu dua masukan yang berbeda memiliki nilai hash yang sama. Tapi peluang terjadinya bentrokan selalu mengecil seiring dengan berkembangnya fungsi hash. Contohnya, MD5 yang keluarannya sepanjang 128-bit memiliki 2^{128} keluaran yang berbeda, yaitu sekitar 3,4 x 10^{38} kemungkinan. Sedangkan SHA-512 memiliki 2512 atau sekitar 1,34 x 10^{154}.

Pada pelaksanaanya, bentrokan dapat dimanfaatkan untuk hal yang merugikan. Misalnya membuat *password* lain dari *password* yang sudah ada untuk mengelabui sistem komputer yang menggunakan hash untuk verifikasi *password*, atau merubah data tanpa mengubah *signature*-nya. Hal ini akan dijelaskan lebih lanjut pada bagian kelemahan fungsi hash.

2.2 Sifat fungsi hash

2.2.1 Sifat baku fungsi hash

Fungsi hash h memiliki sifat-sifat sebagai berikut :

1. *Preimage resistant* : jika diberikan suatu nilai hash y, akan sulit mencari x sedemikian sehingga h(x)=y
2. Second preimage resistant : jika diberikan sebuah masukan x, akan sulit mencari x' sedemikian sehingga h(x)=h(x')
3. Collision resistant : akan sulit untuk mencari x dan x' sedemikian sehingga h(x)=h(x')

2.2.2 Pemanfaatan fungsi hash lemah

Sebuah fungsi hash yang lemah adalah fungsi hash yang dapat ditemukan cara untuk mematahkan satu atau lebih sifat diatas. Kelemahan fungsi hash tersebut dapat dimanfaatkan, diantaranya :

1. Memanfaatkan *preimage resistance* yang lemah : jika kita bisa 'bekerja mundur' atau menghasilkan data dari suatu nilai hash, maka kita dapat menggunakannya untuk mengalahkan sistem *hash password.* Kita tidak perlu tahu *password* sebenarnya, karena cukup dengan memiliki data dengan nilai hash yang sesuai, kita bisa mendapat akses. Disini bentrokan dapat diartikan "lebih dari satu *password* yang bisa diterima".
2. Memanfaatkan *second preimage resistance* yang lemah : jika dapat menghasilkan data baru yang memiliki nilai hash yang sama dengan data asli, maka kita dapat membuat seolah-olah data kita adalah data asli. Umumnya dilakukan untuk menyebarkan perangkat lunak yang bermasalah.
3. Memanfaatkan *collision resistance* yang lemah : jika kita dapat membuat dua buah data memiliki nilai hash yang sama, maka kita dapat mengganti data yang asli dengan data lain. Misalnya data berisi perjanjian setelah disetujui kita ganti dengan data perjanjian yang berbeda.

2.2.3 *Digest* bukan *encryption*

Fungsi hash adalah penyingkatan atau *digest*, bukan enkripsi. Pada umumnya orang salah mengelompokan fungsi hash sebagai fungsi enkripsi, tapi penyingkatan dan enkripsi memiliki perbedaan yang cukup besar, seperti yang terdapat pada tabel 2.

Digest	Encryption
Umumnya merupakan fungsi banyak-ke-satu	Merupakan fungsi satu-ke-satu
Mungkin terjadi bentrokan	Tidak mungkin terjadi bentrokan
Merupakan operasi satu arah, sehingga tidak bisa dibalik	Merupakan operasi dua arah, dapat dibalik (disebut dekripsi)
Menghasilkan suatu nilai *digest* yang lebih kecil dari masukan	Merubah masukan menjadi kode rahasia
Panjang nilai keluaran tidak tergantung pada masukan	Panjang nilai keluaran selalu sama dengan nilai masukan
Pada pemeriksaan melakukan pembandingan nilai *digest*	Pada pemeriksaan mengubah kode rahasia menjadi data asli (proses dekripsi)

Tabel 2 Perbedaan *digest* dan *encryption*

Agar lebih terlihat perbedaannya, perhatikan contoh berikut :
Andaikan kita memiliki data sebuah kalimat :
"this is cleartext that anybody can easily read without the key used by encryption. It's also bigger than the box of text above."
Kalimat ini memiliki nilai digest (misalnya MD5) :
`bd18d50263b01456f22e3ff0d003bf66`
Sedangkan dengan fungsi enkripsi tertentu, kalimat ini memiliki *chipertext* :
"`guvf vf pyrnegrkg gung nalobql pna rnfvyl ernq jvgubhg gur xrl hfrw ol rapelcgvba. Vg'f nyfb ovttre guna gur obk bs grkg nobir.`"
Dari contoh diatas terlihat jelas perbedaan pada panjang kode dan jenis keluaran antara fungsi *digest* dan enkripsi.

2.2.4 *Avalanche Effect*

Avalanche effect adalah sebuah fenomena dimana perubahan terkecil tehadap sebuah data dapat membuat nilai hash-nya berbeda jauh. Contoh :
Kalimat A "This is a very small file with a few characters" mempunyai nilai hash:
`75cdbfeb70a06d42210938da88c42991`
Sedangkan kalimat B "this is a very small file with a few characters" mempunyai nilai hash:
`6fbe37f1eea0f802bd792ea885cd03e2`
Satu-satunya perbedaan pada dua kalimat tersebut adalah huruf pertamanya.
`T > 0x54 > 0 1 0 1 0 1 0 0`
`t > 0x74 > 0 1 1 1 0 1 0 0`
Artinya perbedaan kedua kalimat diatas dalam representasi biner hanya satu bit.

Kalimat A	Kalimat B	Bit
01110101110011 01 (75cd)	01101111101111 10 (6fbe)	8
1011111111101011 (bfeb)	0011011111110001 (37f1)	5
0111000010100000 (70a0)	1110111010100000 (eea0)	5
0110110101000010 (6d42)	1111100000000010 (f802)	5
0010000100001001 (2109)	1011110101111001 (bd79)	7
0011100011011010 (38da)	0010111010101000 (2ea8)	7
1000100011000100 (88c4)	1000010111001101 (85cd)	5
0010100110010001 (2991)	0000001111100010 (03e2)	8

Tabel 3 Perbandingan per bit antar kalimat A dan B

Dari tabel diatas terlihat bahwa pebedaan satu bit pada dua buah data dapat menghasilkan nilai hash yang berbeda hingga 50-bit. Perbedaan ini akan semakin besar sesuai dengan panjang data masukan, persis dengan efek longsoran salju.
Hal ini merupakan salah satu keunggulan fungsi hash, yaitu peka terhadap perubahan sekecil apapun. Dalam kasus ini, perubahan satu bit.

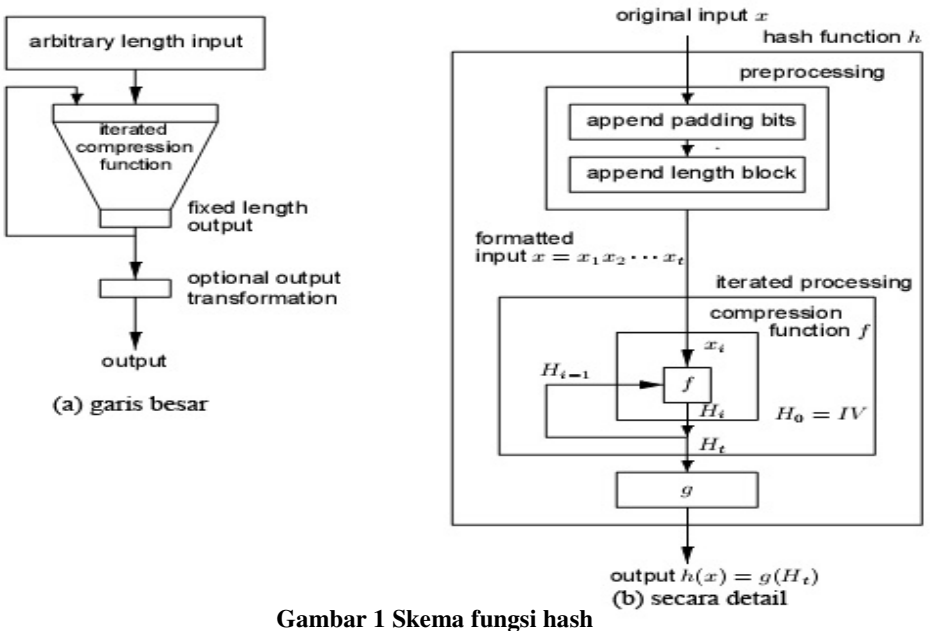
2.3 Klasifikasi fungsi hash

Berdasarkan penggunaan kunci hash, fungsi hash dibagi menjadi dua :
1. Fungsi hash tanpa kunci (*unkeyed hash function*). Contohnya MD4, MD5, SHA-1, dan RIPEMD.
2. Fungsi hash dengan kunci (*keyed hash function*). Contohnya MAC dan HMAC.

Fungsi hash yang biasa digunakan umumnya merupakan fungsi hash tanpa kunci. Namun untuk menghasilkan *digital signature* dapat digunakan fungsi dengan atau tanpa kunci, walau jika menggunakan fungsi hash tanpa kunci kita membutuhkan fungsi enkripsi lain untuk meng-enkripsi nilai hash dengan suatu kunci (bukan kunci untuk fungsi hash).

2.4 Algoritma hash secara umum

Fungsi hash sebenarnya adalah sebuah fungsi matematika yang menggabungkan operasi logika untuk biner, teori bilangan, dan komposisi fungsi. Langkah-langkah dasar pada fungsi hash (untuk lebih jelasnya dapat dilihat pada gambar 1) adalah sebagai berikut :
1. Menambahkan *padding bits* berupa bit 0 secukupnya agar panjang masukan menjadi 64-bit kurang dari kelipatan 512-bit.
2. Menambahkan *length block* sepanjang 64-bit berisi informasi panjang masukan.



Gambar 1 Skema fungsi hash

3. Memecah masukan menjadi blok-blok dengan panjang tertentu (misalnya 512 untuk MD5)
4. Melakukan proses kompresi iteratif untuk setiap blok masukan
5. Menggabungkan hasil kompresi setiap blok menjadi suatu nilai hash

2.5 Berbagai fungsi hash yang terkenal

Fungsi-fungsi hash yang terkenal diantaranya :
- MD4 (128-bit), sudah tidak dipakai
- MD5 (128-bit)
- RIPEMD-160 (160-bit)
- SHA-1 (160-bit)

Fungsi hash selalu berkebang seiring perkembangan jaman, karena belum ada fungsi hash yang 100% tidak memiliki bentrokan. Karena dengan metode komputasi tertentu, bentrokan dapat dihasilkan. Semua algoritma hash yang sudah 'terpecahkan' umunya sudah tidak digunakan karena sudah tidak aman lagi menggunakannya. Dari fungsi diatas, semua sudah terpecahkan kecuali SHA-1. Lihat Tabel 4.Salah satu contoh kasus bentrokan pada algoritma MD5 adalah sebagai berikut :

Data 1 :
```
d131dd02c5e6eec4693d9a0698aff95c
2fcab587 12467eab4004583eb8fb7f89
55ad340609f4b30283e4888325 71 415a
085125e8f7cdc99fd91dbd f2 80373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e2 b4 87da03fd02396306d248cda0
e99f33420f577ee8ce54b67080 a8 0d1e
c69821bcb6a8839396f965 2b 6ff72a70
```

Data 2 :
```
d131dd02c5e6eec4693d9a0698aff95c
2fcab507 12467eab4004583eb8fb7f89
55ad340609f4b30283e4888325 f1 415a
085125e8f7cdc99fd91dbd 72 80373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e234 87da03fd02396306d248cda0
e99f33420f577ee8ce54b6708028 0d1e
c69821bcb6a8839396f965 ab 6ff72a70
```

Keduanya memiliki nilai hash :
```
79054025255fb1a26e4bc422aef54eb4
```

2.6 Aplikasi fungsi hash

Pada bagian awal sudah dijelaskan bahwa pada makalah ini hanya akan dijelaskan mengenai aplikasi fungsi hash dalam *digital signature*, tapi tidak adal salahnya kita melihat aplikasi lain dari fungsi hash :
1. Memeriksa integritas data : seperti dijelaskan sebelumnya, fungsi hash dapat digunakan untuk memeriksa integritas data. Contohnya pada FTP server ProFTPD (www.proftpd.org), setiap file yang bisa di-*download* selalu disertai dengan nilai hash MD5-nya. Jadi jika kita mendapat file dengan nilai hash MD5 yang berbeda artinya terjadi kesalahan saat transfer data.
2. *Hash password* : karena menyimpan *password* dalam bentuk *cleartext* berbahaya, maka banyak sistem komputer menyimpan *password* dalam bentuk nilai hash-nya. Cara ini lebih aman karena walaupun seseorang dapat menjebol database *password* sebuah sistem, namun untuk mendapatkan *password*-nya akan sulit. Dalam proses pemeriksaan *password*, masukan dari pengguna akan dicari

Algorithm	Output size	Internal state size	Block size	Length size	Word size	Collision
HAVAL	256/224/192/160/128	256	1024	64	32	Yes
MD2	128	384	128	No	8	Almost
MD4	128	128	512	64	32	Yes
MD5	128	128	512	64	32	Yes
PANAMA	256	8736	256	No	32	With flaws
RIPEMD	128	128	512	64	32	Yes
RIPEMD-128/256	128/256	128/256	512	64	32	No
RIPEMD-160/320	160/320	160/320	512	64	32	No
SHA-0	160	160	512	64	32	Yes
SHA-1	160	160	512	64	32	With flaws
SHA-256/224	256/224	256	512	64	32	No
SHA-512/384	512/384	512	1024	128	64	No
Tiger(2)-192/160/128	192/160/128	192	512	64	64	No
VEST-4/8 (hash mode)	160/256	176/304	8	80	1	No
VEST-16/32 (hash mode)	320/512	424/680	8	88	1	No
WHIRLPOOL	512	512	512	256	8	No

Tabel 4 Perbandingan fungsi-fungsi hash

nilai hash-nya untuk sebandingkan dengan nilai hash yang terdapat pada database.

3. Error correction : hanya dapat diterapkan untuk fungsi hash yang dua arah. Prinsip kerjanya sama dengan enkripsi-dekripsi, yaitu mengembalikan nilai hash menjadi data asli. Data asli akan dehasilkan jika dalam pemeriksaan nilai hash setelah pengiriman (*redundancy test*) ditemukan perbedaan nilai hash.

4. *Audio identification* : pada program multimedia yang menggunakan *playlist* fungsi hash dapat digunakan untuk mencari data yang terduplikasi. Jika ada beberapa file yang sama, akan mudah untuk mencari tahu kesamaannya. Dengan mencari nilai hash-nya kita bisa mengetahui apakah file-file tersebut identik atau tidak.

2.7 Kelebihan dan kekurangan fungsi hash

Kelebihan fungsi hash dibandingkan fungsi enkripsi pada umumnya :

1. Hasil dari fungsi hash panjangnya tetap, panjang masukan tidak akan mempengaruhi panjang nilai hash.
2. Karena tidak merubah data asli, tidak diperlukan proses dekripsi
3. Perubahan sekecil apapun pada data asli akan membuat nilai hash yang sangat jauh berbeda, sehingga cukup mudah untuk memeriksa keaslian data

Kekurangan fungsi hash diantaranya :

1. Memiliki kemungkinan untuk terjadi bentrokan. Hal ini tidak dapat dihindari untuk semua fungsi hash, namun ada beberapa fungsi hash dibuat khusus untuk menghindari terjadinya bentrokan.
2. Fungsi hash adalah fungsi satu arah, jadi jika kita hanya mendapat sebuah nilai hash, kita tidak bisa mengembalikannya menjadi data yang asli. Hal ini dipersulit dengan kemungkinan terjadinya bentrokan.
3. Tingkat keamanan suatu fungsi hash dinilai berdasarkan jumlah kemungkinan nilai hash, yaitu 2^n, dengan n adalah panjang nilai hash dalam bit. Jadi semakin panjang nilai hash semakin aman.

2.8 Kaitan fungsi hash dan matematika diskrit

Seperti sudah disebutkan pada subbab 2.4 bahwa fungsi hash adalah sebuah fungsi matematika yang menggabungkan operasi logika untuk biner, teori bilangan, dan komposisi fungsi.

Teori bilangan digunakan dalam fungsi kompresi dalam fungsi hash. Contoh fungsi kompresi paling sederhana adalah modulo. Dengan modulo atau mod, kita dapat mengkompresi data dengan panjang berapapun menjadi panjang yang kita inginkan. Contoh : 1568 mod 10 = 8, kompresi dari empat digit menjadi satu digit.

Operasi logika dan aljabar boolean digunakan dalam fungsi terkecil dalam hash. Yang berguha untuk melakukan pengkodean atau pengacakan.

Fungsi dan komposisi fungsi digunakan untuk membentuk sebuah funshi hash, karena sebenarnya fungsi hash terdiri dari beberapa fungsi yang lebih sederhana.

3. Secured Hash Algorithm-256 / SHA-256

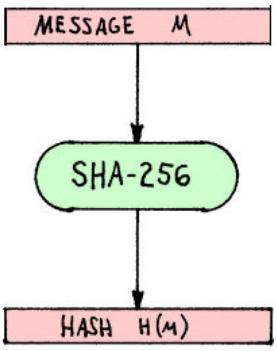
3.1 Sekilas mengenai SHA-256

Fungsi ini adalah salah satu jenis hash yang masih umum digunakan. Fungsi ini adalah varian dari SHA-1, yang dibuat karena

algoritma bentrokan dari *reduced* SHA-1 telah ditemukan. SHA-1 sendiri adalah pengganti dari SHA-0.
Dengan panjang nilai hash 256-bit, fungsi ini memiliki 2^{256} atau sekitar 1,16x $x10^{77}$. Sampai saat ini belum ada yang dapat memecahkan algoritma bentrokan untuk SHA-256.
SHA-256 umumnya digunakan sebagai fungsi antara untuk fungsi lain, termasuk fungsi hash MAC, HMAC, dan beberapa fungsi penghasil *digital signature*.

3.2 Skema dan algortima SHA-256

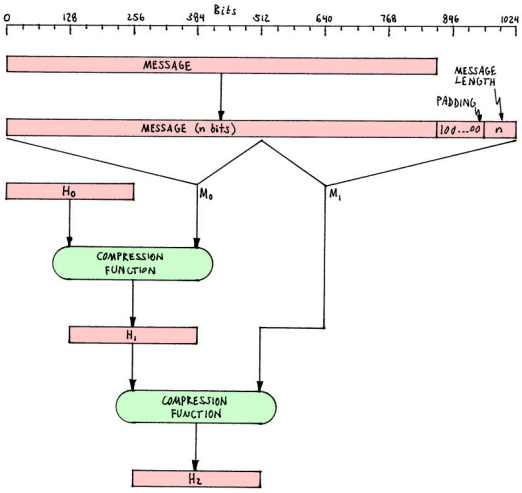
3.2.1 Fungsi utama



Gambar 2 Fungsi utama SHA-256

Fungsi utama SHA-256 menerima masukan berupa sata atau pesan M dengan panjang sembarang, lalu akan menghasilkan nilai hash h(M) dengan panjang 256-bit.

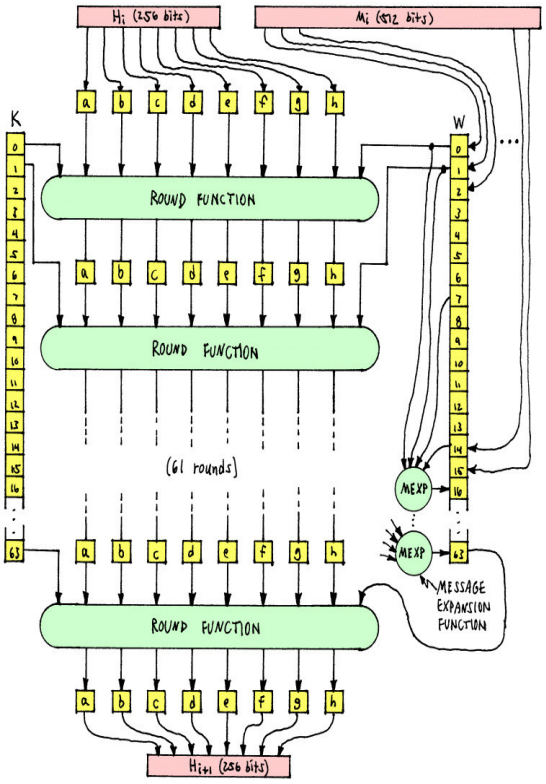
3.2.2 Struktur fungsi hash berulang



Gambar 3 Struktur fungsi hash berulang

Fungsi hash berulang ini merupakan langkah 1-4 pada fungsi hash umum (lihat subbab 2.4). Secara lengkap langkah-langkah yang dilakukan pada fungsi ini, seperti yang terdapat pada gambar 3, adalah sebagai berikut :
1. Menambahkan *padding bits* (bit 0) sehinga panjangnya menjadi 64 kurang dari kelipatan 512-bit. Hal ini dilakukan untuk menyisakan tempat untuk *length block*.
2. Menambahkan length block dengan panjang 64 bit sehingga total panjang pesan sekarang adalah kelipatan 512-bit.
3. Membagi-bagi pesan tadi menjadi b buah blok M_0, M_1, M_2, …, M_b. masing-masing sepanjang 512-bit.
4. Memanggil nilai H_0 (*initial value*) dengan panjang 256-bit yang sudah didefinisikan sebelumnya. Umumnya bilai H_0 dibuat seacak mungkin. Nilai ini adalah salah satu faktor yang menentukan hasil hash. Untuk membedakan fungsi SHA-256 kita dengan milik orang lain kita cukup mengganti nilai H_0 saja.
5. Memanggil fungsi kompresi dengan parameter masukan H_0, M_0, dan parameter keluaran H_1.
6. Terus memanggil fungsi kompresi dengan parameter masukan H_{b-1}, M_{b-1}, dan parameter keluaran H_b sebanyak b kali.
7. H_b merupakan nilai hash dari masukan.

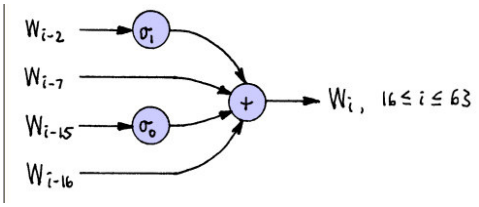
3.2.3 Fungsi kompresi



Gambar 4 Fungsi kompresi

Fungsi ini menerima masukan H_i dan M_i yang masing-masing memiliki panjang 256-bit dan 512-bit. H_i adalah nilai pembanding yang digunakan dalam kompresi, sedangkan M_i adalah satu blok dari pesan yang diterima fungsi utama.
Langkah-langkah yang dilakukan pada fungsi ini adalah :
1. Memecah H_i menjadi 8 blok a-h dengan panjang masing-masing blok 32-bit.
2. Memecah M_i menjadi 16 blok W_0-W_{15} dengan panjang masing-masing blok 32-bit.
3. Mengisi W_{16}-W_{63} dengan cara memanggil *message expansion function* sebanyak 48 kali dengan parameter masukan W_{n-2}, W_{n-7}, W_{n-15}, W_{n-16}, dan keluaran W_n (untuk $16 \leq n \leq 63$).
4. Mendefinisikan pencacah K_i dari 0-63.
5. Memanggil *round function* dengan parameter masukan K_n, a_n-h_n, W_n, dan parameter keluaran a_{n+1}-h_{n+1} sebanyak 64kali (untuk $0 \leq n \leq 63$).
6. Menggabungkan a_{n+1}-h_{n+1} menjadi H_{i+1} dengan panjang 256-bit.
7. Hi+1 adalah hasil atau keluaran dari fungsi kompresi ini.

3.2.4 Message expansion function

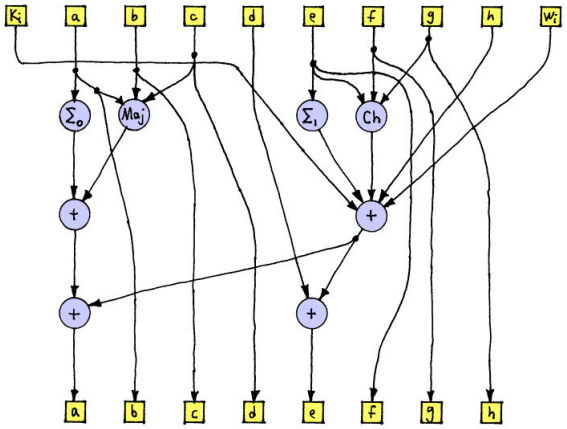


Gambar 5 ***Message expansion function***

Fungsi ini adalah fungsi antara dalam fungsi kompresi yang menerima 4 masukan masing-masing dengan panjang 32-bit lalu menghasilkan keluaran dengan panjang 32-bit. Langkah-langkahnya adalah :
1. Memanggil fungsi σ_1 untuk merubah W_{i-2}.
2. Memangil fungsi σ_0 untuk mengubah W_{i-15}.
3. Melakukan operasi XOR terhadap W_{i-2}, W_{i-7}, W_{i-15}, dan W_{i-16}. Hasil yang didapat adalah W_i.

3.2.5 *Round function*

Fungsi ini adalah fungsi antara dalam fungsi kompresi, sama seperti *message expansion function*. Menerima masukan K_i, a_i-h_i, W_i, dan menghasilkan a_{i+1}-h_{i+1}.
Keluaran yang dihasulkan dapat ditulis sebagai (tanda '+' berarti XOR) :
1. $a_{i+1}=((\Sigma_0(a_i) + Maj(a_i,b_i,c_i)) + (K_i + \Sigma_1(e_i) + Ch(e_i,f_i,g_i) + h_i + W_i))$
2. $b_{i+1}=a_i$
3. $c_{i+1}=b_i$
4. $d_{i+1}=c_i$
5. $e_{i+1}=(d_i + (K_i + \Sigma_1(e_i) + Ch(e_i,f_i,g_i) + h_i + W_i))$
6. $f_{i+1}=e_i$
7. $g_{i+1}=f_i$
8. $h_{i+1}=g_i$



Gambar 6 ***Round function***

3.2.6 Fungsi-fungsi lain

Fungsi-fungsi ini adalah fungsi terkecil yang menyusun fungsi hash. Fungsi-fungsi tersebut adalah :
1. $\sigma_0(X)$ = (X right-rotate 7) xor (X right-rotate 18) xor (X right-shift 3)
2. $\sigma_1(X)$ = (X right-rotate 17) xor (X right-rotate 19) xor (X right-shift 10)
3. $\Sigma_0(X)$ = (X right-rotate 2) xor (X right-rotate 13) xor (X right-rotate 22)
4. $\Sigma_1(X)$ = (X right-rotate 6) xor (X right-rotate 11) xor (X right-rotate 25)
5. Ch(X,Y,Z) = (X and Y) xor ((not X) and Z)
6. Maj(X,Y,Z) = (X and Y) xor (X and Z) xor (Y and Z)

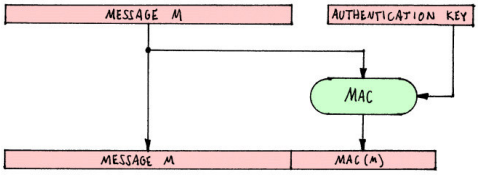
4. MAC dan HMAC

4.1 *Message Authentification Code* (MAC)

MAC adalah salah satu fungsi hash dengan kunci yang dibuat khusus untuk melakukan pemeriksaan keaslian pesan atau *message authentification*. Algoritma MAC mulai berkembang sekitar tahun 1995 dan banyak digunakan secara umum karena memiliki tingkat keamanan yang layak untuk masa itu.
Pada awalnya MAC menggunakan kunci dengan panjang 32-bit atau 56-bit, namun seiring perkembangan jaman, panjang kunci MAC mengalami penyesuaian.
Algoritma MAC ada dua jenis, yaitu :
1. MAC yang mengunakan *cipher-block-chaining*. Algoritma ini menggunakan algoritma *data encryption standard* (DES) sebagai dasarnya, sehingga kunci MAC tersebut adalah kunci DES dengan panjang bit yang sesuai.

2. MAC yang menggunakan fungsi hash tanpa kunci. Algoritma ini menggunakan fungsi hash tanpa kunci yang sudah ada, misalnya MD5, SHA-1, atau SH-256. Algoritma ini biasa disebut sebagai Hash Message Authentification Code (HMAC).

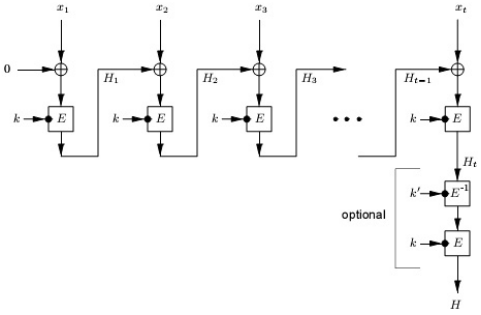
4.2 Skema umum MAC



Gambar 7 Skema umum MAC

Fungsi MAC secara garis besar serupa dengan fungsi hash tanpa kunci pada umumnya, tapi terdapat perbedaan pada cara penggunaannya. Pada fungsi hash tanpa kunci, pengirim pesan dan penerima pesan hanya perlu memiliki fungsi hash yang sama, yang sebaiknya dirahasiakan. Pada saat pengiriman pesan, pengirim cukup mengirimkan pesan asli yang telah ditambahkan nilai hash, pesan ini tidak perlu dirahasiakan. Namun pada penggunaan MAC, selain mengirimkan pesan yang telah ditambahkan kode MAC (tidak perlu dirahasiakan), pengirim juga harus mengirim kunci pemeriksaan keaslian (*atuhentification key*) atau disebut juga kunci MAC secara rahasia. Dengan begitu, proses pengiriman menjadi dua tahap, tetapi dengan kenaikan tingkat keamanan.

4.3 Algoritma CBC-MAC



Gambar 8 Skema fungsi CBC-MAC

Fungsi CBC-MAC adalah fungsi MAC yang memanfaatkan algoritma CBC (*chiper-block-chaining*), yaitu metode enkripsi yang menerapkan mekanisme umpan balik atau *feedback*. Dalam metode ini, blok hasil enkripsi diumpanbalikkan ke enkripsi blok selanjutnya. Dengan metode ini, setiap blok *chipertext* tidak hanya bergantung pada *plaintext*-nya, tapi juga spada seluruh blok *plaintext* sebelumnya.

Algoritma CBC-MAC :
Input : data x, *block chiper* E, kunci MAC k
Output : kode MAC dengan panjang n-bit, n adalah panjang blok dalam *block chiper* E
1. *Padding and blocking*. Tambahkan padding bits pada x secukupnya, lalu pecah x menjadi t buah blok yang masing-masing berukuran n-bit.
2. *CBC processing*. Misalkan E_k adalah enkripsi dengan E menggunakan kunci k, dan nilai H_0 ditentukan, hitung blok Ht dengan rumus :

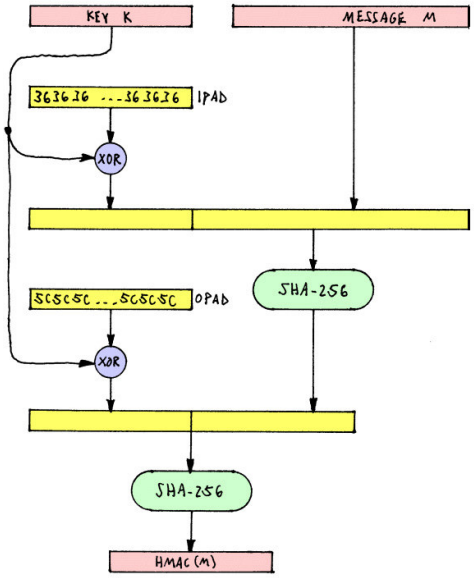
$$H_i = E_k(H_{i-1} \text{ XOR } x_i),\ 2 \le i \le t.$$

3. *Optional process*. Setelah mendapatkan Ht, dapat dilakukan proses tambahan untuk memperkuat keamanan MAC. Menggunakan kunci $k' \neq k$ (k' bisa didapat dari k dengan algoritma lain), hitung

$$H_t' = E_{k'}^{-1}(H_t)$$
$$H_t = E_k(H_t')$$

4. *Completion*. H_t adalah kode MAC dengan panjang n-bit.

4.4 Algoritma HMAC



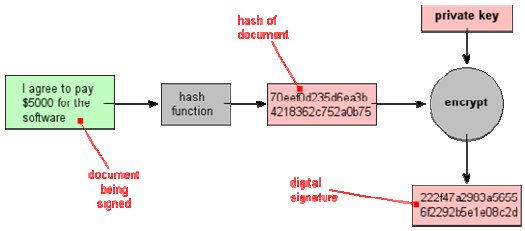
Gambar 9 Skema HMAC

Fungsi ini menggunakan fungsi hash tanpa kunci sebagai dasarnya, disini dipakai contoh SHA-256. Untuk meningkatkan keamanan, digunakan dua blok tambahan yaitu ipad dan opad. Ipad adalah blok yang berisi bit 363636…3636 dengan panjang 256-bit. Sedangkan opad berisi 5c5c5c…5c5c dengan panjang 256-bit pula. Kedua blok ini digunakan sebagai kunci enkripsi untuk kunci MAC (disini enkripsi mengunakan XOR, enkripsi paling sederhana).
Algoritma HMAC :
Input : pesan M, kunci MAC k, ipad, dan opad
Output : nilai HMAC

1. Kunci k dienkripsi dengan kunci ipad, lalu ditambahkan ke bagian depan pesan M.
2. Pesan M diproses dengan fungsi hash SHA-256, dan menghasilkan nilai hash M'.
3. Kunci k dienkripsi kembali dengan kunci opad, lalu ditambahkan ke bagian depan M'.
4. M' diproses kembali dengan SHA-256 utuk menghasilkan nilai hash M''.
5. Nilai HMAC dari M dengan kunci k adalah M''.

5. Digital Signature

Digital signature adalah sebuah nilai yang unik yang dimiliki suatu dokumen. Untuk menghasilkan sebuah *digital signature* dibutuhkan sebuah nilai kunci yang bersifat rahasia dan hanya diketahui oleh pembuat dokumen. *Digital signature* dapat digunakan sebagai bukti bahwa dokumen tersebut asli.



Gambar 10 Proses pembuatan *digital signature*

Digital signature dibuat dalam dua tahap :
1. Mencari nilai hash dengan algoritma hash yang aman (misalkan SHA-256)
2. Mengenkripsi nilai hash tadi dengan suatu nilai rahasia (disebut kunci pribadi atau *private key*)

Keuntungan menggunakan dua tahap di dalam digital signature adalah meningkatkan tingkat keamanan, karena akan semakin sulit dipecahkan. Keuntungan lainnya adalah dapat menghasilkan digital signature yang panjangnya tidak tergantung dengan panjang dokumen.
Contoh penggunaan digital signature:
Misalnya kita membuat suatu dokumen penting dan memutuskan untuk membuat *digital signature*-nya dengan menggunakan kunci pribadi kita. Sewaktu-waktu kita dapat membuktikan bahwa dokumen itu memang buatan kita dengan cara mendekripsi *digital signature*-nya dengan kunci publik milik kita. Lalu kita tinggal membuktikan kalau nilai hash hasil dekripsi sama dengan nilai hash dokumen kita.

6. Kesimpulan

Dari studi mengenai *cryptographic hash function* dan penggunaanya dalam *digital signature* dapat ditarik beberapa kesimpulan :
1. *Cryptographic hash function* adalah salah satu algoritma yang cukup baik untuk digunakan dalam bidang keamanan sistem informasi. Namun dengan catatan bahwa algoritma yang digunakan adalah algoritma yang belum dipecahkan bentrokannya.
2. Fungsi hash adalah salah satu aplikasi bab-bab dalam Mata Kuliah Matematika Diskrit, yaitu Operasi Logika, Aljabar Boolean, Relasi dan Fungsi, dan Teori Bilangan.
3. Fungsi hash yang baik adalah fungsi hash yang memiliki sifat preimage resistant, second preimage resistant, dan collision resistant. Dan secara tidak langsung, fungsi hash yang baik adalah fungsi hash yang menggunakan panjang nilai hash lebih banyak.
4. MAC dan HMAC digunakan untuk pemeriksaan keaslian pesan atau message authentification.
5. Digital signature adalah suatu mekanisme untuk menjaga keaslian suatu dokumen, yang merupakan gabungan fungsi hash dan enkripsi.

DAFTAR PUSTAKA

[1] Friedl, Steve. (2005). An Illustrated Guide to Sryptographic Hashes. *http://www.unixwiz.net/techtips/iguide-crypto-hashes.html.* Tanggal akses : 2 Januari 2007 pukul 09:30.

[2] Kaminsky, Alan. (2004). Cryptographic One-way functions. *http://www.csrc.rit.gov/onewayhash.shtml.* Tanggal akses : 2 Januari 2007 pukul 09:30.

[3] Menezes, A. *et al*.(1997). Handbook of Applied Cryptography. CRC Press Inc.

[4] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[5] Wikipedia. (2006). Cryptographic hash function. *http://en.wikipedia.org/wiki/Cryptographic_hash_function.* Tanggal akses : 2 Januari 2007 pukul 09:30.

[6] Wikipedia. (2006). Hash Function. *http://en.wikipedia.org/wiki/Hash_function.* Tanggal akses : 2 Januari 2007 pukul 09:30.