Analisis dan Perbandingan *Message Digest* dengan SHA dan Usulan Fungsi Hash Baru

Edwin Romelta / 13508052
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
edro_ooo@yahoo.co.id

***Abstract*—Fungsi hash adalah suatu fungsi yang digunakan untuk mengubah suatu masukan menjadi suati nilai yang mempresentasikan seluruh nilai masukannya dengan panjang yang jumlahnya telah ditentukan sebelumnya walaupun jumlah inputan yang dimasukan lebih kecil daripada besar keluaran yang ditentukan sebelumnya. Sehingga dengan fungsi ini harusnya dapat menghasilkan nilai yang unik bedasarkan nilai yang dimasukan. Selain itu dengan perbuahan yang sangat kecil dapat mengubah nilai yang dimasukan dengan perubahan yang sangat signifikan.**

Fungsi hash dikenal juga dengan fungsi kriptografi satu arah karena tidak dapat dikembalikan lagi kebentuk nilai awal dari nilai hasil fungsi ini. Namun dengan hal ini, fungsi hash menjadi sebuah kriptografi yang powerfull, karena tidak bisa dipecahkan.

Fungsi ini banyak digunakan dalam kehidupan sehari – hari. Fungsi ini digunakan untuk memverifikasi sebuah file atau dokumen. Sehingga dokumen yang diterima oleh penerima berupa file yang benar. Karena dengan perbahan sedikit saja nilai hashnya berubah.

Makalah ini akan menganalisis 2 buah fungsi hash yang ada yaitu SHA-1 dan MD5. Karena kedua fungsi tersebut adalah fungsi hash yang masih populer dan dipakai oleh banyak orang meskipun sudah terbukti ada collision walaupun sudah ada lagi fungsi hash yang lebih baik dari pada kedua fungsi tersebut yaitu MD6 dan SHA-2. Setelah dianalisi, kedua buah fungsi itu akan dibandingkan satu sama lainnya. Selain itu bedasarkan hasil analisis dan perbandingan tersebut akan dibuat sebuah fungsi hash baru yang mungkin lebih baik daripada kedua fungsi hash tersebut.

***Index Terms*— fungsi hash, SHA-1, MD5**

1. PENDAHULUAN

1.1 LatarBelakang

Keamanan dalam dunia internet sangat diperlukan. Karena banyak sekali pemalsuan isi file. Sebelum adanya fungsi hash sangat sulit untuk mengetahui apakah isi dalam file itu adalah benar. Karena user hanya dapat melihat nama file dan besar filenya saja. Karena satu – satunya cara agar file yang di unduh itu benar user harus mendownload langsung dari sumbernya atau dari mirror yang diberikan oleh official server. Namun jika servernya itu bukan server local biasanya orang yang mengunduh tidak mau mengunduhnya karena waktu pengunduhan sangat lama. Dan ketika user tersebut mencoba mengunduh dari suatu web local rupanya file tersebut adalah virus yang sengaja orang buat. Tentu saja hal ini membuat susah user.

Sejak fungsi hash muncul server biasanya melampirkan besar hash yang di unduh dan fungsi hash apa yang digunakan untuk mendapatkan nilai tersebut. Meskipun kadang – kadang fungsi hashnya tidak diketahui tapi user masih dapat mengetahui fungsi hash yang digunakan dengan melihat jumlah karakter pada nilai hash yang diberikan. Karena kemungkinan server menggunakan fungsi hash yang sangat banyak digunakan orang yaitu fungsi SHA-1. Dengan demikian user dapat mengecek nilai hashfile tersebut sebelum file yang diunduh itu dijalankan. Sehingga user tidak lagi harus mengunduh dari official server.

Masalah yang lama memang berhasil di atasi, namun sekarang muncul lagi masalah baru. Apakah dengan fungsi hash ini benar menghasilkan nilai yang unik, yaitu nilai yang hanya dimiliki oleh satu buah file. Karena jika nilai ini tidak unik maka tujuan dari fungsi itu sendiri tidak tercapai yaitu dapat membedakan isi sebuah file dengan sebuah nilai yang lebih pendek namun mepresentasikan seluruh isi file tersebut. Dan sudah banyak fungsi hash yang terbukti collision. Untuk itu kita perlu untuk menganalisis kembali fungsi hash.

1.2 Tujuan

- Memahami fungsi hash secara garis besar
- Mengetahui perbedaan dan persamaanfungsi hash MD5 dan SHA-1
- Mengusulkan sebuah algotima fungsi hash yang baru
- Mengimplementasikan fungsi hash yang diusulkan sebelumnya

1.3 Rumusan Masalah

Pada Makalah ini membahas:

1. Menganalisi fungsi hash yang ada yaitu fungsi hash MD5 dan SHA-1

2. Membanding fungsi hash setelah melakukan analisis pada tiap – tiap fungsi
3. Mencoba mengusulkan sebuah algoritma fungsi hash yang baru
4. Melakukan percobaan dan implementasi pada fungsi hash yang diusulkan

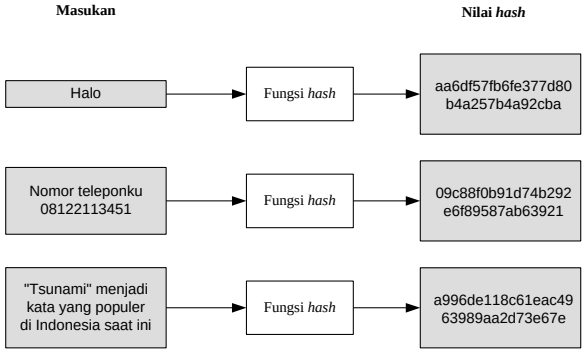
1.4 Batasan Masalah

Pada pengimplementasian dan percobaan usulan fungsi hash yang baru , percobaan collision test tidak akan dilakukan. Karena keterbatasan kemampuan dan waktu. Jika tidak sengaja ditemukan akan disertakan di dalam makalah ini.

2. DASAR TEORI

2.1 Fungsi Hash

Fungs hash adalah algoritma kriptografi satu arah, yaitu masukan yang sudah di ubah menjadi suatu nilai hash tidak dapat lagi dikembalikan ke bentuk inputan.



Jumlah keluaran dari fungsi hash sudah tetap tetap namun jumlah masukannya sembarang.

Nama lain fungsi hash adalah:

- fungsi kompresi (compression function)
- cetak-jari (fingerprint)
- cryptographic checksum
- message integrity check (MIC)
- manipulation detection code (MDC)

Berikut beberapa contoh fungsi hash :

Algoritma	Ukuran *message digest* (bit)	Ukuran blok pesan	Kolisi
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD	128	512	Ya
RIPEMD-128/256	128/256	512	Tidak
RIPEMD-160/320	160/320	512	Tidak
SHA-0	160	512	Ya
SHA-1	160	512	Ada cacat
SHA-256/224	256/224	512	Tidak
SHA-512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

Fungsi hash sering sekali digunakan di kehidupan sehari – hari. Beberapa aplikasi fungsi hash dalam kehidupan nyata adalah :

1. Menjaga integritas data
   Fungsi hash sangat peka terhadap perubahan 1 bit pada pesan. Pesan berubah 1 bit, nilai hash berubah sangat signifikan. Bandingkan nilai hash baru dengan nilai hash lama. Jika sama, pesan masih asli. Jika tidak sama, pesan sudah dimodifikasi
2. Menghemat waktu pengiriman.
   Misal untuk memverifikasi sebuah salinan arsip dengan arsip asli. Salinan dokumen berada di tempat yang jauh dari basisdata arsip asli Ketimbang mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih mangkus mengirimkan message digest-nya. Jika message digest salinan arsip sama dengan message digest arsip asli, berarti salinan arsip tersebut sama dengan arsip master. Contoh lainnya adalah dalam peniriman hash 1 block pada bit torrent.
3. Menormalkan panjang data yang beraneka ragam.
   Misalkan *password* panjangnya bebas (minimal 8 karakter) *Password* disimpan di komputer *host* (*server*) untuk keperluan otentikasi pemakai komputer. *Password* disimpan di dalam basisdata..Untuk menyeragamkan panjang *field password*. di dalam basisdata, *password* disimpan dalam. bentuk nilai *hash* (panjang nilai

hash tetap). Selain memendekan panjang data yang beragam

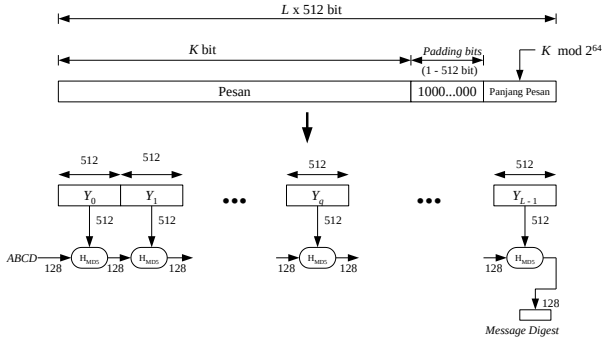
2.2 MD5

MD pada MD5 merupakan pendekan dari *message digest.* Dengan algoritma hash ini akan menghasilkan panjang hash 128 bit atau 16 karakter.

Algoritma MD5:

1. Ubah masukan ke bentuk bit
   Masukan dengan bentuk apapun seperti string atau number, terlebih dahulu diubah ke bentuk biner.
2. Penambahan Bit-bit Pengganjal
   Pertama – tama pesan di tambah dengan bit pengganjal. Ditambahkan terus hingga panjang pesan sekarang sama dengan 448 modulo 512. Karena pengolahan pesan dilakukan dalam blok sebesar 512 bit sedangkan sisa 64 bit akan digunakan untuk penambahan jumlah panjang pesan sebelum ditambah bit pengganjal .Jika panjang pesan tepat konkruen dengan 448 modulo 512 maka pesan tetap ditambah dengan sehingga ditambah sejumlah 512 bit. Jadi panjang bit pengganjal paling tidak ditambahkan 1 buah. Bit pengganjal pertama adalah 1 setelah itu ditambahkan dengan bit 0.
3. Penambahan nilai panjang pesan semula.
   Nilai yang ditambahkan adalah jumlah bit pesan sebelum ditambahkan dengan padding bit. Jika tidak sampai dengan 64bit maka pesan yang ditambah awalnya diisi dengan bit 0 sampai pas 64 bit setelah ditambah dengan panjang pesan semula. Jumlah pesan yang dimasukan juga dalam bentuk biner.

   Gambaran Pembuatan Messege fungsi MD5 sebagai berikut



4. Inisialisasi penyangga (*buffer*) *MD.*
   MD5 membutuhkan 4 buah penyangga yang harus diinisalisasi sebelum memulai proses pengolahan tiap bloknya. 4 buah penyangga karena jumlah hasil akhir dari MD5 adalah 16 karakter hexa atau 128 bit. Berikut adalah buffer yang diinisialisasi dalam bentuk hexa.
   A = 01234567
   B = 89ABCDEF
   C = FEDCBA98
   D = 76543210
5. Pengolahan pesan dalam blok berukuran 512 bit.
   Bentuk blok dengan besar 512bit tiap bloknya. Dalam pengolahan ini terdapat 4 putaran yang tiap putarannya dilakukan sebanyak 16 kali. Tiap putaran memiliki proses yang berbeda – beda. Sebelum putaran pertama dilakukan inisialisasi 4 buah variable dengan besar 32 bit yang menampung buffer inisialisasi.
   Operasi dasar pada MD5 adalah ;

   a = b + CLSs(a + g(b, c, d) + X[k] + T[i])

   keterangan :
   a, b, c, d = empat buah peubah penyangga 32-bit A, B, C, D
   CLSs = circular left shift sebanyak s bit
   X[k] = kelompok 32-bit ke-k dari blok 512 bit message ke-q. Nilai k = 0 sampai 15.
   T[i] = elemen Tabel T ke-i (32 bit)
   \+ = operasi penjumlahan modulo 232
   g = salah satu fungsi F, G, H, I
   Operasi fungi F, G, H, I :

Nama	Notasi	***g*(*b*, *c*, *d*)**
f_F	$F(b, c, d)$	$(b \wedge c) \vee (\sim b \wedge d)$
f_G	$G(b, c, d)$	$(b \wedge d) \vee (c \wedge \sim d)$
f_H	$H(b, c, d)$	$b \oplus c \oplus d$
f_I	$I(b, c, d)$	$c \oplus (b \wedge \sim d)$

Catatan: operator logika AND, OR, NOT, XOR masing-masing dilambangkan dengan $\wedge$, $\vee$, $\sim$, $\oplus$

Setelah selesai 1 blok nilai a,b,c,d akan di jumlahkan ke buffer inisialisasi. Dan jika seluruh proses sudah selesai maka hasil MD5 didapatkan dari penggabungan variable A, B, C,D

2.3 SHA-1

SHA atau *Secure Hash Algorithm* juga merupakan algoritma kriptografi satu arah. Hasil dari algoritma ini adalah bilangan hexa yang panjangnya 20 karakter atau

160 bit.

Algoritma SHA1:

1. Ubah masukan ke bentuk bit

   Masukan dengan bentuk apapun seperti string atau number, terlebih dahulu diubah ke bentuk biner.

2. Penambahan Bit-bit Pengganjal

   Pertama – tama pesan di tambah dengan bit pengganjal. Ditambahkan terus hingga panjang pesan sekarang sama dengan 448 modulo 512. Karena pengolahan pesan dilakukan dalam blok sebesar 512 bit sedangkan sisa 64 bit akan digunakan untuk penambahan jumlah panjang pesan sebelum ditambah bit pengganjal .Jika panjang pesan tepat konkruen dengan 448 modulo 512 maka pesan tetap ditambah dengan sehingga ditambah sejumlah 512 bit. Jadi panjang bit pengganjal paling tidak ditambahkan 1 buah. Bit pengganjal pertama adalah 1 setelah itu ditambahkan dengan bit 0.

3. Penambahan nilai panjang pesan semula.

   Nilai yang ditambahkan adalah jumlah bit pesan sebelum ditambahkan dengan padding bit. Jika tidak sampai dengan 64bit maka pesan yang ditambah awalnya diisi dengan bit 0 sampai pas 64 bit setelah ditambah dengan panjang pesan semula. Jumlah pesan yang dimasukan juga dalam bentuk biner.

4. Inisialisasi penyangga (*buffer*) *SHA*.

   SHA membutuhkan 5 buah penyangga yang harus diinisalisasi sebelum memulai proses pengolahan tiap bloknya. 5 buah penyangga karena jumlah hasil akhir dari SHA adalah 20 karakter hexa atau 160 bit. Berikut adalah buffer yang diinisialisasi dalam bentuk hexa.

   A = 67452301
   B = EFCDAB89
   C = 98BADCFE
   D = 10325476
   E = C3D2E1F0

5. Pengolahan pesan dalam blok berukuran 512 bit.

   Bentuk blok dengan besar 512bit tiap bloknya. Dalam pengolahan ini terdapat 4 putaran yang tiap putarannya dilakukan sebanyak 20 kali. Tiap putaran memiliki proses yang berbeda – beda. Sebelum putaran pertama dilakukan inisialisasi 5 buah variable dengan besar 32 bit yang menampung buffer inisialisasi.

   Untuk membangkitkan 64 buah 32-bit word lagi, diperlukan fungsi :

   w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

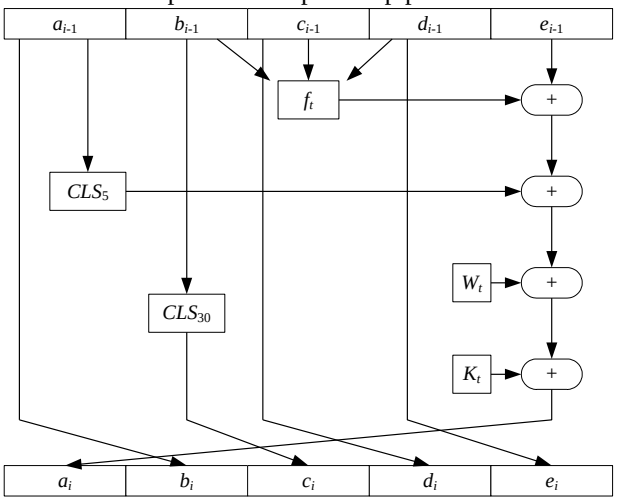
Operasi pada tiap putaran :

Putaran	$f_t(b, c, d)$	K
0 .. 19	$(b \wedge c) \vee (\sim b \wedge d)$	5A827999
20 .. 39	$b \oplus c \oplus d$	*6ED9EBA1*
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$	8F1BBCDC
60 .. 79	$b \oplus c \oplus d$	*CA62C1D6*

Operasi Dasar
e = d
d = c
c = b leftrotate 30
b =a
a = (a leftrotate 5) + f + e + k + w[i]

keterangan :
a, b, c, d, e = lima buah buffer
i = putaran, $0 \leq i \leq 79$
f = fungsi logika
w[i] = word 32-bit ke-i dari 512 bit message
K = bilangan penambah

Gambaran operasi dasar pada tiap putaran



Setelah selesai 1 blok nilai a,b,c,d,e akan di jumlahkan ke buffer inisialisasi. Dan jika seluruh proses sudah selesai maka hasil SHA-1 didapatkan dari penggabungan variable A, B, C, D, E

III. ANALISIS

3.1 MD5

Beberapa hasil analisis dengan algoritma fungsi hash MD5 yaitu :

1. Panjang Pesan
   Panjang pesan maksimum adalah 2^{64} karena jumlah panjang pesan maksimun yang disediakan untuk disimpan adalah 64bit.
2. Penambahan *padding* bit
   padding bit selalu ditambah dengan 1 terlebih dahulu lalu ditambahkan degan 0 sampai konkuren dengan 448 mod 512.
3. Inisialisasi Penyanga
   Menggunakan 4 buah buffer dengan nilai yang ditentukan. Penentuan nilai ini sepertinya asal saja karena bentuk angkanya yang berurutan
4. Pengolahan tiap Blok.
   Pengolahan tiap blok dengan besar blok 512 bit.
5. Pembentukan 32- bit word
   Cukup memotong – motong 512 bit menjadi 16 buah 32-but word setelah itu tidak dilakukan apa – apa lagi.
6. Pengolahan tiap putaran
   Setiap putaran akan melakukan 4 operasi logika biasa yang diulang 16 kali tiap operasi dengan penambaan suatu nilai yang sudah ditentukan di tiap putarannya
7. Operasi Logika
   Menggunakan operasi logika biasa Seperti AND,OR,XOR,implikasi, dan LeftShirt
8. Panjang Nilai Hash
   Panjang hash yang dihasilkan adalah 128 bit atau 16 karakter bilangan hexadecimal.

3.2 SHA-1

Beberapa hasil analisis dengan algoritma fungsi hash SHA1 yaitu :

1. Panjang Pesan
   Panjang pesan maksimum adalah 2^{64} karena jumlah panjang pesan maksimun yang disediakan untuk disimpan adalah 64bit.
2. Penambahan *padding* bit
   padding bit selalu ditambah dengan 1 terlebih dahulu lalu ditambahkan degan 0 sampai konkuren dengan 448 mod 512.
3. Inisialisasi Penyanga
   Menggunakan 5 buah buffer dengan nilai yang ditentukan. Penentuan nilai ini sepertinya asal saja karena bentuk angkanya yang berurutan
4. Pengolahan tiap Blok.
   Pengolahan tiap blok dengan besar blok 512 bit
5. Pembentukan 32- bit word
   Setelah memotong – motong 512 bit menjadi 16 buah 32-but word. Namun dibentuk lagi 64 buah 32-bit word dengan memanfaatkan 16 buah 32-bit word yang ada. Untuk menghasilkan 64 buah 32-bit word tersebut dipelukan sebuah fungsi operasi logika biasa.
6. Pengolahan tiap putaran
   Setiap putaran akan melakukan 4 operasi logika biasa dengan penambaan suatu nilai yang sudah ditentukan di tiap operasi putarannya. Dan tiap operasi dilakukan sebanyak 16 kali.
7. Operasi Logika
   Menggunakan operasi logika biasa Seperti AND,OR,XOR,implikasi, dan LeftShirt
8. Panjang Nilai Hash
   Panjang hash yang dihasilkan adalah 128 bit atau 16 karakter bilangan hexadecimal.

3.3 Usulan Algoritma Baru

Bedasarkan hasil analisis diatas maka untuk membentuk fungsi hash yang baik diperlukan beberapa hal, yaitu:

1. Panjang Pesan
   Panjang pesan sebaiknya memang disimpan di dalam hash namun sebaiknya tidak dibatasi. Sesuatu saat pasti ada pesan yang nilainya lebih dari 2^{64} meskipun masih sangat lama lagi hal itu terjadi.
2. Penambahan *padding* bit
   SHA-1 dan MD5 menggunakan penambahan *padding bit* dan paling tidak bit 1 harus ditambah. Hal ini digunakan agar panjang pesan dapat habis dibagi 512 karena besar tiap blok harus 512 tidak boleh kurang dan tidak boleh lebih.
3. Inisialisasi Penyanga
   SHA-1 dan MD5 menggunakan inisialisasi penyangga. Nilai yang dipilih terlihat asal. Tidak dijelaskan mengapa pembuat fungsi hash tersebut mengambil nilai seperti. Sepertinya tanpa penggunaan inisialisasi penyangga tidak akan berpengaruh besar atau menginisalisasinya dengan 00000000.
4. Pengolahan tiap Blok.
   SHA-1 dan MD5 mengolahan tiap blok dengan besar blok 512 bit. Dengan mengurangi panjang blok akan membuat jumlah operasi yang akan dilakukan lebih banyak sehingga kemungkinan kolisi pun berkurang. Namun waktu proses pembentukan nilai hash pun bertambah.

Sehingga ada baiknya memikirkan penentuan yang prioritas mana yang lebih dibutuhkan untuk saat ini, kemungkinan kolisi atau fungsi hash yang cepat. Namun sepertinya lebih baik mengurangi kemungkinan kolisi karena kecepatan computer bekerja akan lebih baik tiap tahunnya karena perkembangan teknologi yang sangat cepat.

5. Pembentukan 32-bit Word

   SHA-1 membangkitkan 64 buah lagi 32-bit word dengan mengolah 16 buah 32-bit word yang didapat dari 512 panjang blok sedangkan MD5 hanya menggunakan 16 buah 32-bit word. Dengan mengunakan pembangkitan 32-bit word lagi untuk diproses akan mengurangi kemungkinan kolisi juga karena angka yang berbeda tiap putarannya.

6. Pengolahan tiap putaran

   SHA-1 dan MD5 menggunakan 4 buah operasi yang berbeda yang dilakukan beberapa kali. Operasi pada SHA-1 20 kali sedangkan MD5 16 kali. Sepertinya jumlah operasi yang digunakan bedasarkan panjang hasil akhir dari fungsi hash tersebut. Namun dengan penambahan jumlah operasi mungkin tidak akan berpengaruh secara signifikan dengan kemungkinan kolisi. Lebih baik lagi banyak nilai yang akan di olah. Dengan banyak nilai yang diolah dan unik kemungkinan kolisipun mungkin akan lebih baik lagi dibandingkan penambahan jumlah operasi. Ditambah lagi dengan memperbanyak jumlah operasi akan menyulitkan ketika akan mengimplementasikan fungsi tersebut.

7. Operasi Logika

   Menggunakan operasi logika biasa Seperti AND,OR,XOR,implikasi, dan LeftShirt karena sepertinya tidak perlu operasi yang komplek agar tidak kolisi.

8. Panjang Nilai Hash

   Dengan panjang hasil akhir yang panjang tentu akan mengurangi kemungkinan kolisi. Namun tidak menjamin tidak mungkin terjadi kolisi.

IV. PERBANDINGAN

4.1 SHA-1 dengan MD5

1. Panjang Pesan

   Sama – sama hanya bisa menerima pesan dengan panjang maksimum 2^{64} bit

2. Penambahan *padding* bit

   Sama – sama melakkan penambahan padding bit dengan ditambah dengan 1 terlebih dahulu lalu ditambahkan dengan 0 sampai konkuren dengan 448 mod 512.

3. Inisialisasi Penyanga

   SHA – 1menggunakan 5 buah buffer sedangkan MD5 menggunakan 4 buah buffer namun kedua tidak menggunakan alasan yang jelas dalam pemilihan nilai inisialisasinya

4. Pengolahan tiap Blok.

   SHA1 dam MD5 sama – sama mengolah dengan tiap blok dengan besar blok 512 bit.

5. Pembentukan 32- bit word

   MD5 Cukup memotong – motong 512 bit menjadi 16 buah 32-bit word setelah itu tidak dilakukan apa – apa lagi. Sedangkan SHA1 seleah membentuk 16 buah 32-bit word masih membentuk 64 lagi 32-bit word untuk digunakan pada operasi logikanya dalam pengolahan tiap blok.

6. Pengolahan tiap putaran

   SHA1 dan MD5 pada tiap putaran akan melakukan 4 operasi logika biasa, namun SHA1 tiap operasinya akan melakukan 20 kali putaran sedangkan MD5 hanya 16 kali putaran. Tetapi MD 5 dalam 64 putarannya mengunakan sesuatu nilai yang sudah fix sedangkan pada SHA1 hanya pada tiap operasi

7. Operasi Logika

   SHA1 dan MD5 menggunakan operasi logika biasa Seperti AND,OR,XOR,implikasi, dan LeftShirt

8. Panjang Nilai Hash

   Panjang hash yang dihasilkan oleh MD5 adalah 128 bit atau 16 karakter bilangan hexadecimal. Sedangkan pada SHA1 adalah 160 bit.

8.1 SHA1 & MD5 dengan Usulan Algoritma Baru

1. Panjang Pesan

   Panjang pesan yang di simpan adalah 2^{128} bit sepertinya masih tidak begitu perlu untuk tidak dibatasi jumlah maksimal pesan yang data disimpan. 2 kali lipat pesan yang dapat ditampung oleh SHA-1 dan MD5

2. Penambahan *padding* bit

   Sama – sama melakkan penambahan padding bit dengan ditambah dengan 1 terlebih dahulu namun ditambahkan dengan 0 sampai konkuren dengan 128 mod 256 karena pesan yang disimpan lebih banyak.

3. Inisialisasi Penyanga

   Mengunakan 8 buah buffer yang akan di inisialisasi. Lebih banyak daripada SHA1 dan MD5 namun angka yang dipilih juga sembarang.

4. Pengolahan tiap Blok.

   Pengolahan tiap blok sebesar 256 bit tidak seperti SHA-1 dan MD5 yang 512 bit.

5. Pembentukan 32- bit word

Dari panjang blok akan dipotong – potong menjadi 8 buah 32-bit word tidak seperti SHA-1 dan MD5 yang 16 32-bit word. Setelah itu akan dibangkitkan 120 buah lagi 32-bit word tidak seperti SHA1 yang hanya memabangktkan 64 32-bit word. Namun disini mengunakan 128 nilai yangsu dah diinisalisasi sebelumnya dan tiap niali akan dipakai di tiap putaran ke-i.

6. Pengolahan tiap putaran
   Pada tiap putarannya sama seperti SHA1 dan MD5 yang melakukan 4 buah operasi namun tiap operasi akan diulang sebanyak 32 kali.
7. Operasi Logika
   Sama seperti SHA1 dan MD5 yang melakukan operasi logika sederhana seperti AND, OR, XOR, implikasi, dan CircullarLeftShirt karena sepertinya tidak perlu operasi yang komplek agar tidak terdapat kolisi.
8. Panjang Nilai Hash
   Panjang Nilai hash adalah 256 bit atau 32 buah 32-bit word. Tidak seperti SHA1 yang 160 bit atau MD5 yang 128 bit.

V. IMPLEMENTASI

Algoritma usulan :

1. Ubah masukan ke bentuk bit
   Masukan dengan bentuk apapun seperti string atau number, terlebih dahulu diubah ke bentuk biner.
2. Penambahan Bit-bit Pengganjal
   Pertama – tama pesan di tambah dengan bit pengganjal. Ditambahkan terus hingga panjang pesan sekarang sama dengan 128 modulo 256. Karena pengolahan pesan dilakukan dalam blok sebesar 256 bit sedangkan sisa 128 bit akan digunakan untuk penambahan jumlah panjang pesan sebelum ditambah bit pengganjal .Jika panjang pesan tepat konkruen dengan 128 modulo 256 maka pesan tetap ditambah dengan sehingga ditambah sejumlah 256 bit. Jadi panjang bit pengganjal paling tidak ditambahkan 1 buah. Bit pengganjal pertama adalah 1 setelah itu ditambahkan dengan bit 0.
3. Penambahan nilai panjang pesan semula.
   Nilai yang ditambahkan adalah jumlah bit pesan sebelum ditambahkan dengan padding bit. Jika tidak sampai dengan 128bit maka pesan yang ditambah awalnya diisi dengan bit 0 sampai pas 128 bit setelah ditambah dengan panjang pesan semula. Jumlah pesan yang dimasukan juga dalam bentuk biner.
4. Inisialisasi penyangga (*buffer*).
   Algoritma ini membutuhkan 8 buah penyangga yang harus diinisalisasi sebelum memulai proses pengolahan tiap bloknya. 8 buah penyangga karena jumlah hasil akhir dari algoritma ini adalah 32 karakter hexa atau 256 bit. Berikut adalah buffer yang diinisialisasi dalam bentuk hexa.
   A = 01234567
   B = 89ABCDEF
   C = 67452301
   D = EFCDAB89
   E = 10325476
   F = 98BADCFE
   G = 76543210
   H = FEDCBA98
   Tidak ada alasan dalam pemilihan nilai untuk inisialisasi ini.
5. Pengolahan pesan dalam blok berukuran 256 bit.
   Bentuk blok dengan besar 256 tiap bloknya. Dalam pengolahan ini terdapat 4 putaran yang tiap putarannya dilakukan sebanyak 32 kali. Tiap putaran memiliki proses yang berbeda – beda. Sebelum putaran pertama dilakukan inisialisasi 8 buah variable dengan besar 32-bit word yang menampung buffer inisialisasi. Lalu pecah 256 bit menjadi 8 buah 32-bit word. Misalnya a, b, c, d, e, f, g, h.

   Untuk membangkitkan 120 buah 32-bit word lagi digunakan fungsi:

   w[i] = (w[i-1] xor w[i-3] xor w[i-5] xor w[i-7]) leftrotate 1

   Sedangkan 4 buah operasi yang dilakukan apda tiap blok adalah :

Putaran	Operasi
0 … 31	(b AND ~(c) AND d) OR (c AND ~(d) AND e) OR (d AND ~(e) AND f) OR (e AND ~(f) AND g) OR (f AND ~(g) AND h)
32 … 63	b XOR c XOR d XOR e XOR f XOR g XOR h
64 … 95	(b AND c AND d) OR (c AND d AND e) OR (d AND e AND f) OR (e AND f AND g) OR (f AND g AND h)
96 … 127	(b XOR ~(c) XOR d) OR (~(c) AND d AND ~(e)) OR (d XOR ~(e) XOR f) OR (~(e) AND f AND ~(g)) OR (f XOR ~(g) ^ XOR h);

   Sedangkan tiap putaran akan melakukan :

   b = c leftrotate 10
   c = d
   d = e leftrotate 15
   e = f

f = g leftrotate 20
g = h
h = a leftrotate 25
a = a leftrotate 5 + F + k[i] + w[i]

Berikut adalah table k[i]

0	DC52D9FA	43	A2145DFF	86	7A25FE12
1	84D9CA81	44	C782C9AD	87	9126D3FF
2	76AD109D	45	3F8CDD9F	88	1952BDED
3	87162AAF	46	ED35B521	89	95DF2D5D
4	A01F38DE	47	EF0918DC	90	63D3A3DD
5	10A703D3	48	AF01BD89	91	80FE782F
6	178AD07E	49	8921DF7D	92	BA24D92D
7	298FD389	50	92DF887A	93	1067DE21
8	16DC2731	51	67D29FFE	94	AC76DE21
9	4789DC12	52	F82921DD	95	187D31DE
10	582DE124	53	87DF5C4D	96	7625DF12
11	21258124	54	84DE3D4A	97	DA28E3E8
12	ADFE9BBC	55	C7AA26DC	98	4A26D6E3
13	BBE39D22	56	98A5DEC7	99	16F72D35
14	1287D11A	57	C8B2AA65	100	BE7128D3
15	7269DA7B	58	1286D3CA	101	2297A8C3
16	3670AC5E	59	37F3E4C5	102	1D8E27CE
17	F28AD249	60	A6B3C8D6	103	19DE28DD
18	280FB37A	61	19D6A5E3	104	B716DE34
19	67AAC728	62	96A32564	105	DB382931
20	1894FD82	63	A4D3E6E8	106	CC125931
21	97A6D268	64	C3D6B112	107	B921A2D3
22	1978DF27	65	A5D623E7	108	1F109212
23	1972BDEF	66	E98F5EA4	109	F4F8FDFF
24	18BBFA35	67	12D4C761	110	B13D3DFF
25	1892D7AE	68	163F3531	111	F1A34DB3
26	18D7EB6F	69	25D39BF1	112	98F72FD2
27	178FB38F	70	87F54C54	113	39D3FF76
28	18AD72BD	71	987C6BDA	114	17FA3EE3
29	179D3E5E	72	AE73BD22	115	BD829A21
30	12B7D3DA	73	AB2E8FC0	116	652A2411
31	27D38F7F	74	E462D73E	117	7182DA28
32	10DDF3A7	75	A38C72BB	118	AD823BFF
33	76D910FF	76	AA224F2F	119	FA24A367
34	21EF24F2	77	BC65AD12	120	FB29A8D2
35	123EFF12	78	1264DD7F	121	D820A882
36	524D6576	79	63AD26DE	122	4820A3F1
37	82DF278A	80	821FBCCB	123	23DE98D1
38	AD6284FD	81	DE271F28	124	52AD45DE
39	1298F3D5	82	F34DD21A	125	83BD29DE
40	902176DF	83	D2971D35	126	267DE8F2
41	87AD09CE	84	EE298AC1	127	129F51D2
42	DF80A8DD	85	991827AD		

Tidak ada alasan dalam pemilihan angka dalam table diatas.

Setelah 1 blok selesai maka masukan nilai a, b, c, d, e, f, g, h kedalam nilai buffer yaitu A, B, C, D, E, F, G, H. Lalu dilanjutkan ke blok berikutnya seperti SHA1 dan MD5.

Jika seluruh blok sudah selesai dijalankan maka hasil dari nilai hash yang didapat merupakan gabungan dari nilai a, b,c ,d ,e ,f ,g ,h yang hasilnya adalah 256 bit atau 32 buah 32-bit word.

VI. PERCOBAAN

Jika pesan yang dimasukan adalah " " atau spasi maka nilai hash yang didapatkan adalah :

15C79E3345C419EA8F0A80CEB6A7134E3F5B3C07A0E7F183EDD1EFEFFEDCBA98

Sedangkan jika yang dimasukan adalah pesan berikut :

> Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa lebih panas dari hari-hari biasanya. Menurut laporan Dinas Meteorologi Kota Bandung, suhu tertinggi kota Bandung adalah 33 derajat Celcius pada Hari Rabu, 17 Oktober yang lalu. Suhu terseut sudah menyamai suhu kota Jakarta pada hari-hari biasa. Menurut Kepala Dinas Meteorologi, peningkatan suhu tersebut terjadi karena posisi bumi sekarang ini lebih dekat ke matahari daripada hari-hari biasa.
>
> Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak lama lagi akan tinggal kenangan. Disamping karena faktor alam, jumlah penduduk yang padat, polusi dari pabrik di sekita Bandung, asap knalpot kendaraan, ikut menambah kenaikan suhu udara kota.

Hasil dari nilai hash adalah:

7885A0DD473E94C7D8012FE361EB3D0ED673E05F68D1EA15635EEC34FEDCBA98

Sedangkan dengan mengubah angka "33" menjadi "34" nilai hash menjadi :

57D7908644F0BB450D876F9992E0C5F70A376DF29D35A37C4FBB1DA5FEDCBA98

Dapat dilihat bahwa dengan mengubah satu buah karakter pada pesan dapat mengubah nilai hash secara keseluruhan dan ini sesuai dengan tujuan keamanan dari fungsi hash itu sendiri.

VII. KESIMPULAN

SHA1 dan MD5 sama – sama memiliki kekurangan dan kelebihan sehingga tidak ada yang terbaik dari kedua algoritma tersebut. Namun tergantung tujuan yang ingin dicapai dari penggunaan hash baru bisa dikatakan yang mana yang lebih baik.

Namun MD5 sudah terbukti terbukti memiliki kolisi sedangkan SHA1 baru diperkirakan terdapat kolisi maka bisa dibilang saat ini SHA1 lebih baik daripada MD5. Karena SHA1 masih belum bisa dikatakan memiliki kolisi, namun jika nanti SHA1 terbukti memiliki kolisi maka kembali keawal lagi yaitu tidak ada dari kedua fungsi hash ini yang lebih baik.

Meskipun MD5 sudah terbukti terdapat kolisi dan SHA1 diperkirakan ada kolisi, kedua fungsi hash ini masih dipakai banyak orang.

Dalam pembuatan suatu fungsi hash yang baru dibutuhkan beberapa hal yang harus dipertimbangkan agar fungsi hash tersebut dapat mengurangi kemungkinan kolision dan juga cepat bekerjanya yaitu :

1. Panjang Pesan
   Panjang pesan sebaiknya disimpan dalam fungsi hash karena jika ada satu bit yang hilang maka seluruh nilai hash akan berubah. Namun sebaiknya tidak usah menyimpan panjang pesan maksimum yang dapat di simpan pada suatu fungsi hash.
2. Penambahan *padding bit*
   Penambahan padding bit diperlukan untuk agar jumlahnya tepat sama dengan besar blok yang diinginkan. Mungkin jika fungsi hash tersebut tidak mengunakan panjang pesan yang ditetapkan fungsi padding ini baru akan berubah tidak seperti SHA1 dan MD5. Karena kita harus menghitung perkiraan jumlah panjang pesan dahulu. Sehingga akan membuat fungsi itu sendiri menjadi lebih rumit.
3. Inisialisasi *buffer*
   Hal ini sebenarnya tidak begitu perlu dilakukan karena dengan mengisi inisialisasi *buffer* dengan 00000000 hasil akhir dari nilai hash tersebut masih berubah cukup signifikan jika diubah salah satu bitnya namun ada baiknya kita menginisalisasinya terlebih dahulu diabndingkan tidak.
4. Besar Blok
   Besar blok sebaiknya lebih sedikit karena dengan besar blok yang lebih sedikit akan menghasilkan blok yang lebih banyak untuk di lakukan operasi tiap bloknya dengan begitu harapanya adalah kemungkinan kolisi menjadi lebih sedikit. Namun dengan jumlah blok yang sangat banyak akan mengakibatkan fungsi hash itu sendiri menjadi lebih lama. sehingga harus dipikirkan prioritas mana yang lebih diperlukan. Apakah kecepatan berjalannya fungsi hash tersebut atau mengurangi kemungkinan kolisi.
5. Inisialisasi Pecahan tiap Blok
   Sebaiknya setelah kita beach blok tersebut menjadi beberapa bagian. Kita gunakan kembali pecahan tersebut untuk membangkitkan suatu nilai yang akan dioperasikan selanjutnya seperti yang dilakukan oleh SHA1. Dengan begitu nilai pada tiap putaran akan selalu unit tidak seperti MD5 yang hanya memanfaatkan 16 buah nilai dari hasil pecahan tiap blok tersebut.
6. Penggunaan Tabel nilai tetap
   MD5 menggunakan sebuah tabel yang nilainya sudah tetap. Dengan tabel itu setiap putarannya akan menggunakan nilai yang unik tidak seperti SHA1 yang mengunakan sebuah nilai tetap pada tiap operasi. SHA1 hanya memiliki 4 buah nilai tetap sedangkan MD5 menggunakan 64 buah nilai tetap yang akan digunakan pada putaran sehingga ada nilai yang unik pada tiap putarannya.
7. Jumlah Operasi pada tiap putaran
   Pemilihan operasi logika pada tiap blok tidak perlu rumit – rumit. Cukup dengan banyak operasi yang cukup. Dengan jumlah operasi yang banyak kemungkinan kolisipun juga dapat dikurangi namun akan mengakibatkan fungs itu sendiri sulit di implementasikan.
8. Operasi Logika
   Pada operasi *bitwise* pada tiap operasi, tidak perlu menggunakan operasi yang rumit – rumit yang penting dapat mencangkup seluruh buffer yang ada.
9. Panjang nilai hash
   Dengan panjang nilai hash yang panjang tentu saja akan mengurangi kemungkinan kolisi karena dapat menampung jumlah yang sangat banyak. MD5 menghasilkan 128 bit sedankan SHA1 menghasilkan 160 bit.

Fungsi hash yang saya usulkan telah memenuhi salah satu keamanan dari fungsi hash yaitu dapat berubah secara signifikan jika terjadi perubahan pada salah satu bitnya.

Namun belum diketahui apakah ada kolisi atau kecacatan pada fungsi yang saya usulkan karena keterbatasan kemampuan.

VIII. REFERENCES

[1] Slide Fungsi Hash
[2] Slide Fungsi Hash MD5
[3] Slide Fungsi Hash SHA-1
[4] http://en.wikipedia.org/wiki/SHA-1
[5] http://en.wikipedia.org/wiki/MD5

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Maret 2011

Edwin Romelta / 13508052