Makalah IF3058 Kriptografi

Pemanfaatan Utilitas Komputer dalam Penentuan Fungsi Hash

Rheno Manggala Budiasa and 13506119
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if16119@students.if.itb.ac.id

***Abstrak*—Penerapan Fungsi Hash sudah banyak digunakan pada saat ini. Para pengembang perangkat lunak menggunakan fungsi Hash untuk berbagai kebutuhan. Mulai dari sebagai tanda tangan digital sampai pengecekan terhadap perubahan suatu file. Salah satu fungsi Hash cukup terkenal adalah MD5. Contoh penggunaannya adalah proses enkripsi pada sandi lewat (*password*) pada bahasa pemrograman PHP. Ukuran yang tetap (*fixed*) ditambah lagi kemampuan enkripsi yang cukup sulit ditebak membuat fungsi ini banyak digunakan dalam bidang Keamanan Informasi. Beberapa fungsi hash juga digunakan sebagai ..**

***Kata Kunci*—Hash, perangkat lunak, tanda tangan, digital, file, MD5, enkripsi, *password*, PHP, *fixed*, Keamanan.**

I. PENDAHULUAN

Keamanan informasi menjadi penting saat ini. Keamanan data dan menghindari program jahat (SPAM, virus, Trojan dll) menjadi kebutuhan penting dalam Teknologi Informasi. Beberapa perangkat digital dan perangkat lunak (*software*) mengamankan sistemnya dengan cara manipulasi bit. Salah satu metode yang biasa digunakan dalam Kemaman Informasi adalah Kriptografi. Beberapa metode enkripsi yang kita kenal dalam kriptografi (sha-0, sha-1, md5 dll) membuat data atau file yang terenkripsi tidak akan mudah dirusak. Beberapa bahasa pemrograman juga sudah menggunakan fungsi kriprografi misalnya Java, Visual Basic, dan C#.

Fungsi Hash merupakan salah satu aplikasi enkripsi dari kriptografi yang dapat digunakan untuk mengamankankan file atau data. Algoritma hash dalam kriptografi (sebagai contoh sha-1, sha-2, md4, md5 dll) dapat mengenkripsi data atau file secara tetap setiap bit-nya. Md5 misalnya menggunakan 128 bit sebagai nilai tetap enkripsi pesannya.

MD5 adalah salah satu dari serangkaian algortima *message digest* yang didesain oleh Profesor Ronald Rivest dari MIT (Rivest, 1994). Saat kerja analitik menunjukkan bahwa pendahulu MD5 — MD4 — mulai tidak aman, MD5 kemudian didesain pada tahun 1991 sebagai pengganti dari MD4 (kelemahan MD4 ditemukan oleh Hans Dobbertin).

Pada tahun 1993, den Boer dan Bosselaers memberikan awal, bahkan terbatas, hasil dari penemuan *pseudo-collision* dari fungsi kompresi MD5. Dua vektor inisialisasi berbeda *I* dan *J* dengan beda 4-bit diantara keduanya.

$$MD5compress(I,X) = MD5compress(J,X)$$

Pada tahun 1996 Dobbertin mengumumkan sebuah kerusakan pada fungsi kompresi MD5. Dikarenakan hal ini bukanlah serangan terhadap fungsi *hash* MD5 sepenuhnya, hal ini menyebabkan para pengguna kriptografi menganjurkan pengganti seperti WHIRLPOOL, SHA-1 atau RIPEMD-160.

Ukuran dari *hash* — 128-bit — cukup kecil untuk terjadinya serangan *brute force birthday attack*. MD5CRK adalah proyek distribusi mulai Maret 2004 dengan tujuan untuk menunjukka kelemahan dari MD5 dengan menemukan kerusakan kompresi menggunakan *brute force attack*.

Bagaimanapun juga, MD5CRK berhenti pada tanggal 17 Agustus 2004, saat kerusakan *hash* pada MD5 diumumkan oleh Xiaoyun Wang, Dengguo Feng, Xuejia Lai dan Hongbo Yu. Serangan analitik mereka dikabarkan hanya memerlukan satu jam dengan menggunakan IBM P690 cluster.

Pada tanggal 1 Maret 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger mendemontrasikan konstruksi dari dua buah sertifikat X.509 dengan *public key* yang berbeda dan *hash* MD5 yang sama, hasil dari demontrasi menunjukkan adanya kerusakan. Konstruksi tersebut melibatkan *private key* untuk kedua *public key*

tersebut. Dan beberapa hari setelahnya, Vlastimil Klima menjabarkan dan mengembangkan algortima, mampu membuat kerusakan Md5 dalam beberapa jam dengan menggunakan sebuah komputer notebook. Hal ini menyebabkan MD5 tidak bebas dari kerusakan.

Dikarenakan MD5 hanya menggunakan satu langkah pada data, jika dua buah awalan dengan *hash* yang sama dapat dibangun, sebuah akhiran yang umum dapat ditambahkan pada keduanya untuk membuat kerusakan lebih masuk akal. Dan dikarenakan teknik penemuan kerusakan mengijinkan pendahuluan kondisi *hash* menjadi arbitari tertentu, sebuah kerusakan dapat ditemukan dengan awalan apapun. Proses tersebut memerlukan pembangkitan dua buah file perusak sebagai file templat, dengan menggunakan blok 128-byte dari tatanan data pada 64-byte batasan, file-file tersebut dapat mengubah dengan bebas dengan menggunakan algoritma penemuan kerusakan

Selain fungsi hash yang sudah kita ketahui dalam kriptografi. Ada beberapa parameter lain yang dapat kita gunakan sebagai penambah fungsi Hash yang sudah ada. Kita dapat memanfaatkan utilitas pada komputer seperti tanggal dan waktu dari computer kita sebagai fungsi Hash.

II. METODE

Pada kriptografi dikenal beberapa Algoritma Hash seperti GOST, HAVAL, MD2, MD4, MD5, PANAMA, RadioGatun, RIPEMD, RIPEMD-128/256, RIPEMD-160/320, SHA-0, SHA-1, SHA-256/224, SHA-512/384, Tiger(2)-192/160 /128, WHIRLPOOL.

Algorithm	Output size (bits)	Internal state size	Block size	Length size	Word size	Collision attacks (complexity)	Preimage attacks (complexity)
GOST	256	256	256	256	32	Yes 2^{105}	Yes 2^{192}
HAVAL	256/224/192/160/128	256	1024	64	32	Yes	
MD2	128	384	128	No	32	Almost	
MD4	128	128	512	64	32	Yes 3	Yes $2^{97.4}$
MD5	128	128	512	64	32	2^{16}	Yes 2^{127}
PANAMA	256	8736	256	No	32	Yes	
RadioGatún	Arbitrarily long	58 words	3 words	No	1-64	With flaws (2^{352} or 2^{704})[10]	
RIPEMD	128	128	512	64	32	Yes 2^{13}	
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 $2^{33.6}$	
SHA-1	160	160	512	64	40	Proven (2^{80}), With flaws (2^{63})[11]	No
SHA-256/224	256/224	256	512	64	32	No	No
SHA-512/384	512/384	512	1024	128	64	No	No
Tiger(2)-192/160/128	192/160/128	192	512	64	64	2^{62} [15]	Yes $2^{184.3}$
WHIRLPOOL	512	512	512	256	8	No	

Algoritma Hash

Fungsi Hash menyimpan beberapa bit tetap sebagai enkripsi terhadap chipper teksnya. Ciri khusus dari fungsi hash yaitu pesan yang telah dienkripsi tidak dapat dibalikkan (meskipun ada beberapa algoritma yang bisa dengan kunci tertentu). Karena sifatnya yang khusus ini Hash sering digunakan sebagai pengecekan terhadap integritas suatu file.

II.1 GOST

GOST merupakan blok cipher dari bekas Uni Sovyet, yang merupakan singkatan dari "Gosudarstvennyi Standard" atau Standar Pemerintah, standar ini bernomor 28147-89 oleh sebab itu metoda ini sering disebut sebagai GOST 28147-89. GOST secara struktural mirip dengan DES.

GOST merupakan blok cipher 64 bit dengan panjang kunci 256 bit. Algoritma ini mengiterasi algoritma enkripsi sederhana sebanyak 32 putaran (round). Untuk mengenkripsi pertama-tama plainteks 64 bit dipecah menjadi 32 bit bagian kiri, L dan 32 bit bagian kanan, R. Subkunci (subkey) untuk putaran i adalah Ki. Pada satu putaran ke-i operasinya adalah sebagai berikut :

Li = Ri-1

Ri = Li-1 xor f(Ri-1,Ki)

Sedangkan pada fungsi f mula-mula bagian kanan data ditambah dengan subkunci ke-i modulus 232. Hasilnya dipecah menjadi delapan bagian 4 bit dan setiap bagian menjadi input s-box yang berbeda. Di dalam GOST terdapat 8 buah s-box, 4 bit pertama menjadi s-box pertama, 4 bit kedua menjadi s-box kedua, dan seterusnya. Output dari 8 s-box kemudian dikombinasikan menjadi bilangan 32 bit kemudian bilangan ini dirotasi 11 bit kekiri. Akhirnya hasil operasi ini di-xor dengan data bagian kiri yang kemudian menjadi bagian kanan dan bagian kanan menjadi bagian kiri (swap). Pada implementasinya nanti rotasi pada fungsi f dilakukan pada awal saat inisialisasi sekaligus membentuk s-box 32 bit dan dilakukan satu kali saja sehingga lebih menghemat operasi dan dengan demikian mempercepat proses enkripsi/dekripsi.

Subkunci dihasilkan secara sederhana yaitu dari 256 bit kunci yang dibagi menjadi delapan 32 bit blok : k1, k2, …, k8. Setiap putaran menggunakan subkunci yang berbeda. Dekripsi sama dengan enkripsi dengan urutan ki dibalik. Standar GOST tidak menentukan bagaimana menghasilkan s-box sehingga ada spekulasi bahwa sebagian organisasi di (eks) Sovyet mempunyai s-box yang baik dan sebagian diberi s-box yang buruk sehingga mudah diawasi/dimata-matai. Kelemahan GOST yang diketahui sampai saat ini adalah karena key schedule-nya yang sederhana, sehingga pada keadaan tertentu menjadi titik lemahnya terhadap metoda kriptanalisis seperti Related-key Cryptanalysis. Tetapi hal ini dapat di atasi dengan melewatkan kunci kepada fungsi hash yang kuat

secara kriptografi seperti SHA-1, kemudian menggunakan hasil hash untuk input inisialisasi kunci. Kecepatan dari metoda ini cukup baik, tidak secepat Blowfish tetapi lebih cepat dari Idea. Untuk kecepatan enkripsi di memori sebesar 5MB data adalah : 3.492,620 Kbyte/detik pada Pentium Pro 200 MHz dan 2.466,700 Kbyte/detik pada Pentium MMX 200 MHz.

Pada metoda blok cipher ada yang dikenal sebagai mode operasi. Mode operasi biasanya mengkombinasikan cipher dasar, feedback dan beberapa operasi sederhana. Operasi cukup sederhana saja karena keamanan merupakan fungsi dari metoda cipher yang mendasarinya bukan pada modenya. Mode pertama adalah ECB (Electronic Codebook) dimana setiap blok dienkrip secara independen terhadap blok lainnya. Dengan metoda operasi ini dapat saja sebuah pesan disisipkan di antara blok tanpa diketahui untuk tujuan tertentu, misalnya untuk mengubah pesan sehingga menguntungkan si pembobol. Mode lainya adalah CBC (Cipher Block Chaining) dimana plainteks dikaitkan oleh operasi xor dengan cipherteks sebelumnya, metoda ini dapat dijelaskan seperti pada Gambar 1. Untuk mode ini diperlukan sebuah Initialization Vector (IV) yang akan di-xor dengan plainteks yang paling awal. IV ini tidak perlu dirahasiakan karena bila kita perhatikan jika terdapat n blok maka akan terdapat n-1 IV yang diketahui. Metoda lain yang dikenal adalah CFB (Cipher Feedback), OFB (Output Feedback), Counter Mode, dll.

Kebanyakan data tidak bulat dibagi kedalam 64 bit (8 byte), oleh sebab itu dalam blok cipher diperlukan data tambahan pada blok terakhir untuk menggenapi blok menjadi 64 bit, hal ini biasanya disebut padding. Ada beberapa cara untuk melakukan padding, salah satu caranya adalah yang disebut Ciphertext Stealing yang akan digunakan dalam aplikasi dari metoda GOST ini.

Subkunci dihasilkan secara sederhana yaitu dari 256 bit kunci yang dibagi menjadi delapan 32 bit blok : k1, k2, …, k8. Setiap putaran menggunakan subkunci yang berbeda. Dekripsi sama dengan enkripsi dengan urutan ki dibalik. Standar GOST tidak menentukan bagaimana menghasilkan s-box sehingga ada spekulasi bahwa sebagian organisasi di (eks) Sovyet mempunyai s-box yang baik dan sebagian diberi s-box yang buruk sehingga mudah diawasi/dimata-matai. Kelemahan GOST yang diketahui sampai saat ini adalah karena key schedule-nya yang sederhana, sehingga pada keadaan tertentu menjadi titik lemahnya terhadap metoda kriptanalisis seperti Related-key Cryptanalysis. Tetapi hal ini dapat diatasi dengan melewatkan kunci kepada fungsi hash yang kuat secara kriptografi seperti SHA-1, kemudian menggunakan hasil hash untuk input inisialisasi kunci. Kecepatan dari metoda ini cukup baik, tidak secepat Blowfish tetapi lebih cepat dari Idea. Untuk kecepatan enkripsi di memori sebesar 5MB data adalah : 3.492,620 Kbyte/detik pada Pentium Pro 200 MHz dan 2.466,700 Kbyte/detik pada Pentium MMX 200 MHz.

II.2 HAVAL

HAVAL adalah salah satu algoritma hash satu arah yang diciptakan oleh Yuliang Zheng, Josef Pieprzky dan Jennifer Seberry. Tujuan utama dari algoritma ini adalah untuk menghasilkan keluaran nilai dengan panjang digest yang dapat bervariasi. Hasil keluaran panjang nilai algoritma HAVAL dapat berupa 128, 160, 192, 224 dan 256 bit. Pada algoritma HAVAL ini, suatu pesan dapat diproses sebanyak 3, 4 atau 5 kali. Hal ini menambahkan eksibilitas pada algoritma HAVAL. Kombinasi antara panjang output (*digest*) dan jumlah proses menyediakan 15 versi HAVAL. Menurut eksperiman dari penciptanya, HAVAL lebih cepat 60 % dibandingkan dengan MD5 jika melalui 3 kali proses, 15% lebih cepat jika melalui 4 kali proses dan sama cepat dengan MD5 jika melalui 5 kali proses

II.3 MD2

MD2 adalah fungsi hash kriptografi yang dikembangkan oleh Ronald Rivest pada tahun 1989. Algoritma ini dioptimalkan untuk 8-bit komputer. MD2 dispesifikasikan dalam RFC 1319. Meskipun algoritma lainnya telah diajukan sejak, seperti MD4, MD5 dan SHA , bahkan pada 2009 MD2 tetap digunakan di infrastruktur kunci publik sebagai bagian dari sertifikat yang dihasilkan dengan MD2 dan RSA.

MD2 adalah fungsi hash cryptographic yang dikembangkan oleh Ronald Rivest pada tahun 1989. Algoritma dioptimalkan untuk komputer 8-bit. MD2 yang ditetapkan dalam RFC 1319. Meskipun algoritma lainnya telah diusulkan sejak dulu, seperti MD4, MD5 dan SHA, bahkan sampai dengan 2004 [update] MD2 tetap digunakan dalam infrastruktur kunci publik sebagai bagian dari sertifikat yang dihasilkan dengan MD2 dan RSA

II.4 MD4

MD4 didesain untuk sebagai algoritma fungsi hash yang memiliki kemangkusan dalam segi waktu. MD4 dibuat oleh Ronald Rivest pada Oktober 1990. Pertama-tama kita misalkan pesan memiliki sejumlah b-bit pesan sebagai input, dan kita menginginkan untuk menghasilkan message digest dari pesan tersebut. Dalam hal ini, b merupakan sebuah bilangan bulat positif, mungkin nol, dan bilangan tersebut harus kelipatan dari 8. Kita membagi pesan tersebut sebagai berikut : m_0 m_1 . . . m_{b-1} Di bawah ini akan dijelaskan lima

tahap proses menghasilkan message digest untuk algoritma MD4.

1. Memasukkan Padding Bit Pertama-tama dilakukan padding pada pesan awal sehingga panjangnya (dalam satuan bit) kongruen dengan 448 modulo 512. Maka pesan tersebut kekurangan 64 bit untuk mencapai kelipatan 512. Padding selalu dilakukan sehingga pesan tersebut kongruen dengan 512. Padding bit awal yaitu bit "1", selanjutnya ditambahkan bit "0" sehingga pesan tersebut memiliki panjang yang kongruen dengan 448 modulo 512. Jadi, panjang padding paling sedikit adalah satu bit hingga 512 bit.

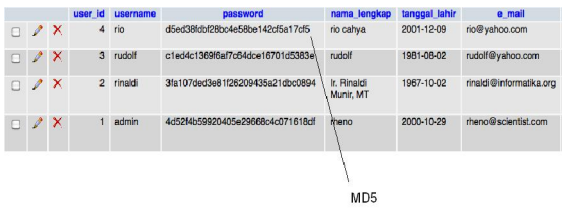
2. Memasukkan Panjang Pesan Sebuah 64-bit yang direpresentasikan oleh b (panjang pesan awal sebelum dilakukan padding bit) dimasukkan pada hasil pesan untuk tahap satu di atas. Panjang pesan yang digunakan selalu lebih kecil dari 264. Bila panjang pesan melebihi 264, maka order terendah yang direpresentasikan oleh 264 yang akan digunakan. Oleh karena itu, setelah proses ini, pesan akan memiliki panjang kelipatan dari 512 bit. Dan pesan dibagi dalam kelipatan 32 bit dalam M[0 . . . N-1] yang menotasikan pesan, dimana N adalah kelipatan dari 16.

3. Inisialisasi Penyangga Ada empat peubah penyangga yang digunakan, yaitu A, B, C, D yang digunakan untuk menghasilkan message digest. Masing-masing variable ini merepresentasikan sebuah nilai 32- bit register. Register tersebut diinisialisasi dengan nilai dalam bentuk heksadesimal, dalam order terendah terlebih dahulu) :

4. Proses Pesan Dalam Blok 32-bit Untuk tahap ini pertama-tama definisikan terlebih dahulu fungsi-fungsi pelengkap yang dibutuhkan untuk menghasilkan message digest, Terdapat tiga fungsi pelengkap yang dalam hal ini tiap-tiap fungsi ini menerima input 32-bit dan menghasilkan output 32 bit juga.

II.5 MD5

Md5 digunakan oleh beberapa bahasa pemrograman sebagai enkripsi terhadap pengujian suatu file.



Enkripsi password dengan md5

MD5 ialah fungsi hash kriptografik yang digunakan secara luas dengan hash value 128-bit. Pada standard Internet (RFC 1321), MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah file.

MD5 di desain oleh Ronald Rivest pada tahun 1991 untuk menggantikan hash function sebelumnya, yaitu MD4 yang berhasil diserang oleh kriptanalis. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan message digest yang panjangnya 128 bit.

MD-5 adalah salah satu aplikasi yang digunakan untuk mengetahui bahwa pesan yang dikirim tidak ada perubahan sewaktu berada di jaringan. Algoritma MD-5 secara garis besar adalah mengambil pesan yang mempunyai panjang variable diubah menjadi 'sidik jari' atau 'intisari pesan' yang mempunyai panjang tetap yaitu 128 bit. 'Sidik jari' ini tidak dapat dibalik untuk mendapatkan pesan, dengan kata lain tidak ada orang yang dapat melihat pesan dari 'sidikjari'MD-5

Message Digest 5 (MD-5) adalah salah satu penggunaan fungsi hash satu arah yang paling banyak digunakan. MD-5 merupakan fungsi hash kelima yang dirancang oleh Ron Rivest dan didefinisikan pada RFC 1321[10]. MD-5 merupakan pengembangan dari MD-4 dimana terjadi penambahan satu ronde[1,3,10]. MD-5 memproses teks masukan ke dalam blok-blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD-5 berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai hash[3,10]. Simpul utama MD5 mempunyai blok pesan dengan panjang 512 bit yang masuk ke dalam 4 buah ronde. Hasil keluaran dari MD-5 adalah berupa 128 bit dari byte terendah A dan tertinggi byte D.

Cara Kerja Algoritma MD5 Langkah-langkah pembuatan message digest secara garis besar:
a.Penambahan bit-bit pengganjal(paddingbits).
b.Penambahan nilai panjang pesan semula.
c.Inisialisasi penyangga(buffer) MD.
d.Pengolahan pesan dalam blok berukuran 512 bit.

II.6 PANAMA

Desain dari Panama didasarkan pada mesin state berhingga dengan state 544-bit dan buffer atau penyangga sepanjang 8192 bit. State dan penyangga dapat diubah dengan melakukan sebuah iterasi. Ada dua macam mode pada fungsi iterasi. Yang pertama adalah mode Push, yang dapat memasukkan sebuah masukan dan tidak menghasilkan keluaran, dan yang kedua adalah mode Pull, yang tidak mengambil input dan menghasilkan keluaran. Sebuah iterasi Pull kosong adalah sebuah iterasi Pull yang keluarannya diabaikan.

Perubahan transformasi pada state mempunyai sifat difusi yang tinggi dan nonlinear yang terdistribusi. Desainnya ditujukan pada penyediaan kenon-linearan yang sangat tinggi dan difusi yang cepat untuk iterasi yang banyak. Hal ini disadari dari kombinasi dari empat transformasi masing-masing dengan kontribusi sendiri. Ada satu untuk kenon-linearan, satu untuk dispersi bit, satu untuk difusi inter-bit, dansatu untuk pemasukan penyangga dan bit-bit input.

Penyangga bertindak sebagai LFSR (Linear Feedback Shift Register) yang memastikan bahwa bit masukan telah dimasukkan ke dalam state melalui iterasi dengan interval yang panjang. Dalam mode Push, input ke shift register dibentuk oleh masukan eksternal, dan pada mode Pull, dibentuk oleh bagian dari state.Fungsi hash Panama didefinisikan sebagai melakukan iterasi-iterasi Push dengan input blok-blok pesan. Jika semua blok pesan telah dimasukkan, sejumlah iterasi Pull kosong dilakukan agar blok-blok pesan akhir terdifusi kedalam penyangga dan state. Hal ini diikuti oleh sebuah iterasi Pull akhir untuk mendapatkan hasil hash-nya.

Skema enkripsi aliran dari Panama diinisialisasi dengan melakukan dua iterasi Push untuk memasukkan kunci dan parameter diversifikasi diikuti sejumlah iterasi Pull kosong agar kunci dan parameter terdifusi ke dalam penyangga dan state. Setelah inisialisasi ini, skema tersebut siap untuk menghasilkan bit-bit keystream dengan melakukan iterasi-iterasi Pull.

II.7 SHA

Secure Hash Algorithm (SHA) merupakan salah satu algoritma fungsi hash yang digunakan. SHA adalah fungsi *hash* satu-arah yang dibuat oleh NIST dan digunakan bersama DSS (*Digital Signature Standard*). Oleh NSA, SHA dinyatakan sebagai standard fungsi *hash* satu-arah. SHA didasarkan pada MD4 yang dibuat oleh Ronald L. Rivest dari MIT. SHA disebut aman (*secure*) karena ia dirancang sedemikian sehingga secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan. Dengan mengubah sedikit pada pesan dengan probabilitas sangat tinggi akan menghasilkan message digest yang berbeda. Hal ini akan menjadikan kegagalan dalam verifikasi ketika secure hash algorithm ini digunakan dengan digital signature algorithm atau sebuah keyed-hash message authentication algorithm. SHA mempunyai empat spesifikasi, yaitu SHA-1, SHA-256, SHA-384 dan SHA-512.

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security[2] (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

spesifikasi jenis-jenis SHA

Komputer yang kita kenal sekarang (PC, Desktop, Mac, dan lain-lain) merupakan perangkat digital yang juga mengimplementasikan fungsi hash sebagai keamanannya.

Selain algoritma yang kita ketahui sebelumnya, fungsi hash dapat kita bentuk sendiri melalui utilitas komputer yang ada misalnya jam atau tanggal dari sebuah perangkat komputer. Caranya adalah dengan menambahkan *header* atau informasi tambahan pada suatu pesan, sehingga jika pesan tersebut disimpan maka akan terdiri dari header dan isi pesan.

III. PROSES

Seperti yang telah dijelaskan sebelumnya bahwa beberapa bahasa pemrograman sudah mengimplementaskan fungsi hash. Dengan fasilitas ini kita tinggal memanggil fungsi hash yang ada dan menambahkannya dengan informasi tambahan (*header*).

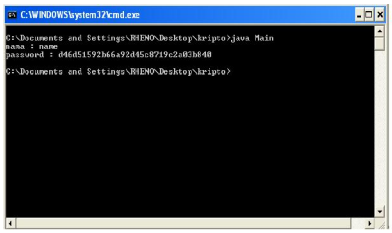
Bahasa yang akan digunakan adalah java. Alasan penggunaannya karena bahasa tersebut sudah mengimplementasikan fungsi hash. Fungsi Hash tersebut diimplementasikan melalui kode :

```
import java.security.MessageDigest;
```

Berikut adalah contoh kode program untuk fungsi hash pada java :

```
public void hash()
{
try{
          String name = "name";
           String passwd = "rheno";

           MessageDigest m =
MessageDigest.getInstance("MD5");

m.update(passwd.getBytes("UTF8"));

              byte s[] = m.digest();
           String result = "";
           for (int i = 0; i < s.length;
i++) {
           result +=
Integer.toHexString((0x000000ff & s[i]) |
0xffffff00).substring(6);
           }


System.out.println("nama : "+name);
System.out.println("password : "+result);

}catch(Exception E){}
  }
```

Hasil yang didapat sebagai berikut :



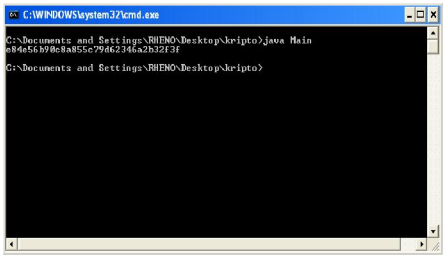
Sekarang kita masukkan suatu pesan dengan format :

-Header-
-isi pesan-

Contoh pesan teks :

17/05/2010 5:12:20
Hello Rheno

Maka setelah dienkripsi menjadi :



Hasil Hash dengan MD5

IV. KESIMPULAN

- Algoritma-algoritma pada fungsi hash sangat cocok digunakan untuk tanda tangan digital
- Jumlah bloknya yang tetap dan tidak mudah ditebak membuat algoritma ini sangat baik digunakan sebagai pengecekan suatu pesan.
- Beberapa Aplikasi E-mail juga sudah banyak yang menggunakan fungsi ini

V. REFERENSI

[1] www.wikipedia.com

[2] Staphen Brown and Zvonko Vranesic, *Fundamentals of Digital Logic with VHDL Design*, *Second Edition*, 2005, McGraw Hill Higher Education

[3] Rinaldi Munir, Slide Kuliah Kriptografi, Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB 2006

[4] www.java2s.com

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, 29 April 2010

Nama dan NIM