APLIKASI FUNGSI HASH KRIPTOGRAFI PADA MESSAGE DIGEST 5

Nessya Callista – NIM: 13505119

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

E-mail : if15119@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang penerapan fungsi hash kriptografi pada Message Digest 5 (MD5). Sewaktu kita mengirim atau menerima pesan pada jaringan, terdapat empat buah persoalan yang sangat penting, yaitu : kerahasiaan, autentikasi, keutuhan dan *non-repudiation*. Kriptografi dapat disebut juga sebagai seni untuk menyembunyikan informasi. Perkembangan komunikasi telah mendorong manusia untuk menyembunyikan informasi yang dimilikinya dari orang lain demi alasan keamanan dan privasi. Kriptografi telah dikenal sejak 4000 tahun yang lalu.
Message Digest 5 (MD5) adalah salah satu alat untuk memberi garansi bahwa pesan yang dikirim akan sama dengan pesan yang diterima. Hal ini didapat dengan membandingkan 'sidik jari' atau 'intisari pesan' kedua pesan tersebut. MD5 merupakan pengembangan dari MD4 dimana telah terjadi penambahan satu putaran. MD5 memproses teks masukan ke dalam blok-blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD5 adalah berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai hash. Makalah ini akan menganalisis proses keutuhan atau perubahan pesan dengan menggunakan MD5 dan juga menganalisis hasil keluaran dari MD5 yang berupa kecepatan dari proses aplikasi yang dibuat. MD5 digunakan secara luas dalam dunia perangkat lunak untuk menyediakan semacam jaminan bahwa file yang diambil belum terdapat perubahan. MD5 hanya menggunakan satu langkah pada data.

Kata kunci : MD5, kriptografi, hash.

1. Pendahuluan

Secara matematis kriptografi membawa proses perhitungan matematis sebagai sarana kriptografi pesan. Metode yang paling sering dipakai adalah *hashing*. *Hashing* adalah proses untuk menghitung *hash value* dari pesan.
Ada empat hal penting saat menerima dan mengirim pesan pada jaringan yaitu : kerahasiaan, autentikasi, keutuhan dan *non repudiation*. Kerahasiaan adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapa pun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah di sandi. Autentikasi berhubungan dengan identifikasi dan pengenalan baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informai yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman dan lain-lain. Keutuhan memberi garansi bahwa data kita tidak mengalami perubahan selama dalam proses pengiriman. Dengan kata lain data yang dikirim sama dengan data yang diterima. Dan *non repudiation* adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman informasi.

Salah satu dari bagian kriptografi adalah fungsi hash satu arah. Fungsi hash satu arah adalah dimana kita dengan mudah melakukan enkripsi untuk mendapatkan *chiper* tetapi sangat sulit untuk mendapatkan *plaintext*-nya. Salah satu

fungsi hash yang paling banyak digunakan adalah *Message Digest 5* (MD5).

MD5 merupakan fungsi hash satu arah yang diciptakan oleh Ronald Rivest pada tahun 1991 untuk menggantikan fungsi hash sebelumnya, MD4. MD5 adalah salah satu aplikasi yang digunakan untuk mengetahui bahwa pesan yang dikirim tidak mengalami perubahan selama berada di jaringan.

Algoritma MD5 secara garis besar adalah mengambil pesan yang mempunyai panjang variabel kemudian 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 'sidik jari' Md5.

Message digest atau inti sari pesan harus mengandung tiga sifat penting yaitu :

- Bila P diketahui, maka MD(P) akan dengan mudah dapat dihitung
- Bila MD(P) diketahui, maka tidak mungkin menghitun g P
- Tidak seorang pun dapat memberi dua pesan yang mempunyai intisari pesan yang sama
  H(M) tidak sama dengan H(M').

2. Kriptografi dan Fungsi Hash Satu Arah

Bagian ini menerangkan tentang kriptografi secara umum yaitu mengenai enkripsi dan dekripsi penggunaan kunci simetrik dan asimetrik, juga tujuan dari kriptografi.

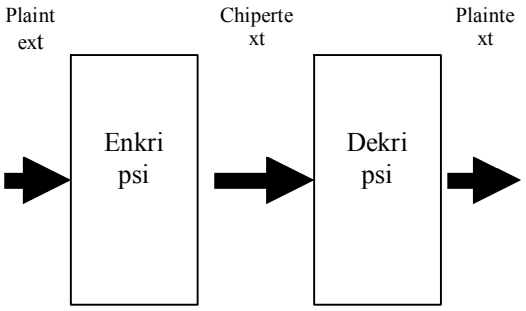
2.1 Prinsip Dasar Kriptografi

Kriptografi secara umum adalah ilmu yang menjaga kerahasiaan suatu berita atau pesan. Selain itu kriptografi dapat juga diartikan sebagai ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi. Kriptografi telah dipakai sejak jaman Julius Caesar sewaktu mengirimkan pesan kepada panglimanya dimana ia tidak ingin pesan itu diketahui oleh orang lain.

Kriptografi mempunyai dua bagian yang penting, yaitu : enkripsi dan dekripsi. Enkripsi adalah proses penyandian pesan asli menjadi simbol-simbol pesan yang tidak dapat diartikan seperti pesan aslinya. Dekripsi adalah merubah pesan yang sudah disandikan menjadi pesan aslinya. Pesan asli ini biasa disebut *plaintext* sedangkan pesan yang sudah disandikan disebut dengan *chipertext*.
Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan berisi elemen teks terang / *plaintext* dan yang berisi elemen teks sandi / *chipertext*. Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. Apabila elemen-elemen teks terang dinotasikan dengan P, elemen-elemen teks sandi dinotasikan dengan C, sedangkan untuk proses enkripsi dinotasikan dengan E dan dekripsi dengan notasi D.

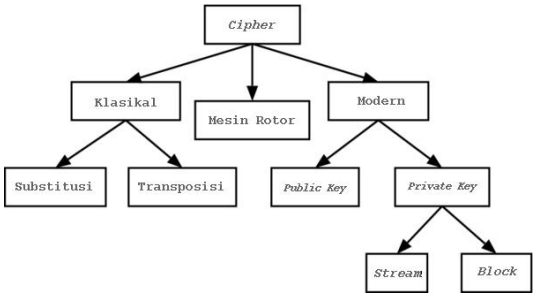
Enkripsi : E(P) = C
Dekripsi : D(C) = P atau D(E(P)) = P

Pada gambar dibawah ini dapat dilihat bahwa masukan berupa *plaintext* akan masuk ke dalam blok enkripsi dan keluarannya berupa *chipertext,* kemudian *chipertext* akan masuk ke dalam blok dekripsi dan keluarannya akan kembali menjadi *plaintext* semula.



Gambar 2.1

Ada banyak variasi pada tipe enkripsi yang berbeda. Algoritma yang digunakan pada awal sejarah kriptografi sudah sangat berbeda dengan metode modern, dan *chiper* modern diklasifikasikan berdasarkan bagaimana *chiper* tersebut beroperasi dan *chiper* tersebut menggunakan satu atau dua buah kunci.



Sejarah *chiper* pena dan kertas pada waktu dulu sering disebut sebagai *chiper* klasik. *Chiper* klasik termasuk juga *chiper* pengganti dan *chiper* transposisi. Pada awal abad 20, mesin-mesin yang lebih mutakhir digunakan untuk kepentingan enkripsi, mesin rotor yang merupakan skema awal yang lebih kompleks.

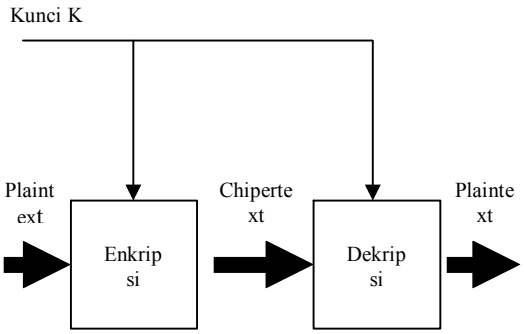
Ada dua model algoritma enkripsi yang menggunakan kunci yaitu : kunci simetrik dan kunci asimetrik.

Enkripsi kunci simetrik biasanya disebut enkripsi konvensional adalah enkripsi yang menggunakan kunci yang sama untuk enkripsi maupun dekripsi. Sandi ini biasanya diterapkan dalam sandi klasik.

Rumus:

$$ {}_nC_2 = \frac{n \,.\, (n-1)}{2} $$

Dengan n adalah banyak pengguna.



Gambar 2.2

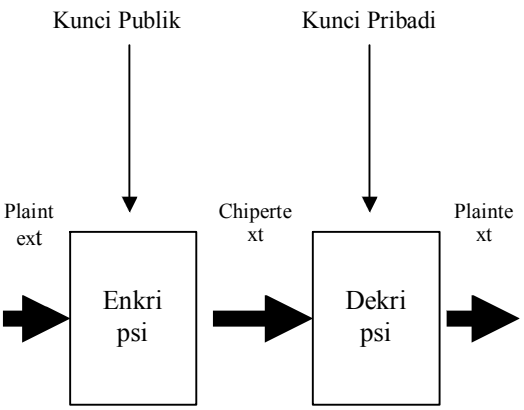
Pada gambar 2.2. terlihat bahwa untuk mengenkripsi maupun mendekripsi pesan hanya menggunakan satu buah kunci (K) saja. Penggunaan metode ini membutuhkan persetujuan antara pengirim dan penerima tentang kunci sebelum mereka melakukan proses pengiriman dan penerimaan pesan. Keamanan dari kunci simetrik tergantung pada kerahasiaan kunci. Apabila orang lain dapat menemukan kunci tersebut maka dengan mudah ia dapat membaca pesan yang sudah dienkripsi.

Enkripsi kunci simetrik dapat dibagi kedalam dua kelompok yaitu : metode *stream chiper* dan metode *block chiper*.

Enkripsi kunci asimetrik yang biasanya disebut dengan kunci publik dibuat sedemikian rupa sehingga kunci yang dipakai untuk enkripsi berbeda dengan kunci yang dipakai untuk dekripsi. Enkripsi ini disebut juga dengan kunci publik karena kunci untuk enkripsi boleh di beritahukan kepada umum sedangkan kunci untuk mendekripsi hanya disimpan oleh orang yang bersangkutan. Enkripsi asimetrik ini dapat ditulis sebagai berikut :

$$ E_k(P) = C $$
$$ D_k(C) = P $$

Contohnya seperti pada gambar 2.3. Bila seseorang ingin mengirim pesan kepada orang lain maka orang tersebut menggunakan kunci publik untuk mengenkripsi pesan yang dikirim kepadanya. Kemudian yang bersangkutan akan mendekripsi pesan tersebut dengan kunci pribadi miliknya.



Gambar 2.3

Cara membuat kunci privat dan kunci publik :

Pilih dua bilangan prima p dan q secara acak. Bilangan ini harus cukup besar (minimal 100 digit). Hitung n = pq. Bilangan n ini disebut parameter sekuriti. Pilih bilangan k secara acak tetapi k tidak boleh mempunyai faktor pembagi yang sama (selain bilangan 1) dengan (p-1) (q-1). Bilangan k ini kemudian kita jadikan sebagai kunci privat kita. Hitung h sedemikian sehingga kh mod (p-1) (q-1) =1. Ada algoritma yang disebut algoritma Euclid untuk menghitung h sangat efesien. Bilangan n dan h ini kita sebar ke publik. h adalah yang menjadi kunci publik. Sementara itu bilangan p dan q boleh dibuang, dan jangan pernah sampai tersebar ke publik.

Setiap sistem kriptografi yang baik harus memiliki karakteristik sebagai berikut :

- Keamanan sistem terletak pada kerahasiaan kunci dan bukan pada algoritma yang digunakan
- Memiliki ruang kunci (keyspace) yang besar
- Menghasilkan *chipertext* yang terlihat acak dalam seluruh tes statistik yang dilakukan terhadapnya
- Mampu menahan seluruh gangguan yang telah dikenal sebelumnya

2.2 Proses Enkripsi Pesan

Misalkan seseorang ingin mengirim pesan *m.* Ia akan mengubah *m* menjadi angka *n* < N, menggunakan protokol yang sebelumnya telah disepakati dan dikenal sebagai *padding scheme.*

Pengirim akan memiliki *n* dan mengetahui N dan *e,* yang telah diberitahukan oleh penerima. Pengirim kemudian menghitung *chipertext c* yang terkait pada n:

$$c = n^e \bmod N$$

Perhitungan tersebut dapat diselesaikan dengan cepat menggunakan metode eksponensial. Pengirim kemudian mengirimkan *c* pada penerima.

2.3 Proses Dekripsi Pesan

Penerima menerima pesan dari pengirim dan mengetahui kunci privat yang digunakan oleh penerima sendiri. Penerima kemudian memulihkan *n* dari *c* dengan langkah-langkah berikut :

$$n = c^d \bmod N$$

Perhitungan di atas akan menghasilkan n, dengan begitu penerima dapat mengembalikan pesan semula *m.* Prosedur dekripsi bekerja karena :

$$c^d \equiv (n^e)^d \equiv n^{ed} \pmod N$$

Kemudian, dikarenakan $ed \equiv 1 \pmod{p-1}$ dan $ed \equiv 1 \pmod{q-1}$, hasil dari *Fermat's little theorem*:

$$n^{ed} \equiv n \pmod p$$

Dan

$$n^{ed} \equiv n \pmod q$$

Dikarenakan *p* dan *q* merupakan bilangan prima yang berbeda, mengaplikasikan *Chinese remainder theorem* akan menghasilkan dua macam kongruen :

$$n^{ed} \equiv n \pmod{pq}$$

dan

$$c^d \equiv n \pmod N$$

2.4 Tipe Penyerangan / Gangguan pada Sistem Kriptografi

Suatu penyerangan pasif atas sistem kriptografi adalah semua metode yang bertujuan untuk mengungkapkan informasi tentang *plaintext* dan *chipertext*-nya tanpa mengetahui kuncinya.

Secara sistematis :
Diberikan fungsi F, G, dan H yang terdiri dari n variabel.
Diberikan sistem enkripsi E.
Diberikan suatu distribusi *plaintext* dan kunci.

Suatu penyerangan atas E dengan menggunakan G dan dengan mengasumsikan F membagi H dengan probabilitas p adalah suatu algoritma A dengan sepasang input f, g dan satu buah output h sedemikian sehingga terdapat probabilitas p atas h = $H(P_1,...,Pn)$, jika kita memiliki f = $F(P_1,..., Pn)$ dan g = G (Ek(P_1), ... , Ek(Pn)). Perlu diperhatikan bahwa probabilitas ini tergantung pada distribusi vektor-vektor (K, P_1, ..., Pn).

Penyerangan akan merupakan suatu trivial bila terdapat probabilitas paling sedikit p untuk h = $H(P_1, ... , Pn)$ jika f = $F(P_1, ... , Pn)$ dan g = $G(C_1,..., Cn)$. Disini $C_1,..., Cn$ terletak pada *chipertext* yang mungkin, dan tidak memiliki hubungan tertentu dengan $P_1, ... ,Pn$. Dengan kata lain, suatu serangan akan merupakan trivial bila ia tidak benar-benar menggunkan enkripsi $Ek(P_1), ... , Ek(Pn)$.

Dengan merumuskan penyerangan secara sistematis, kita dapat secara tepat memformulasikan dan bahkan membuktikan pernyataan bahwa suatu sistem kriptografi itu kuat. Kita katakan sebagai contoh bahwa suatu kriptosistem adalah aman terhadap seluruh penyerangan pasif jika sembarang penyerangan nontrivial terhadapnya tidak praktis. Jika kita dapat membuktikan pernyataan ini maka kita akan yakin bahwa sistem kriptografi kita akan bertahan terhadap seluruh teknik *cryptanaltic* pasif. Jika kita dapat mereduksi pernyataan ini hingga pada beberapa masalah yang tidak terpecahkan maka kita masih tetap memiliki keyakinan bahwa sistem kriptografi kita tidak mudah dibuka.

Chipertext-Only Attack

Dengan menggunakan notasi diatas suatu *chipertext only attack* adalah suatu penyerangan dengan F adalah konstanta. Diberikan hanya beberapa informasi $G(Ek(P_1), … , Ek(Pn))$ tentang n *chipertext*, penyerangan harus memiliki kesempatan memiliki kesempatan menghasilkan beberapa informasi $H(P_1, …, Pn)$ tentang *plaintext*. Penyerangan akan merupakan suatu trivial bila ia hanya menghasilkan $H(P_1, ..., Pn)$ ketika diberikan $G(C_1, ... , Cn)$ untuk $C_1, ... , Cn$ acak.

Sebagai contoh, misalkan G (C) = C dan misalkan H(P) adalah bit pertama P. Kita dapat secara mudah menulis suatu penyerangan yang menduga bahwa H(P) adalah 1. Penyerangan ini adalah trivial karena tidak menggunakan *chipertext,* probabilitas keberhasilannya adalah 50%. Di pihak lain, terdapat penyerangan atas RSA yang memproduksi 1 bit informasi tentang P, dengan keberhasilan probabilitas 100%, menggunakan C. Jika diberikan suatu C acak maka tingkat kesuksesan turun menjadi 50%. Inilah yang disebut penyerangan nontrivial.

Known-Plaintext Attack

Penyerangan ini memiliki $F(P_1,P_2) = P_1$, $G(C_1,C_2) = (C_1,C_2)$, dan $H(P_1,P_2)$ tergantung hanya pada P_2. Dengan kata lain bila diberikan dua *chipertext* C_1 dan C_2 dan satu dekripsi P_1, penyerangan ini seharusnya menghasilkan informasi tentang dekripsi P_2.

Brute-Force Attack

Contohnya, kita diberikan sejumlah *plaintext* P_1, $... , P_{n-1}$ dan *chipertext* $C_1, ... , C_{n-1}$. Kita juga diberikan sebuah *chipertext* Cn. Kita jalankan seluruh kunci K. Bila kita temukan K sedemikian sehingga $Ek(P_1) = Ci$ untuk setiap $i < n$, kita cetak Dk(Cn).
Jika n cukup besar sehingga hanya satu kunci yang bekerja, penyerangan ini akan sukses untuk seluruh input yang valid pada setiap waktu. Sementara ia akan menghasilkan hasil yang tepat hanya sekali untuk input acak. Penyerangan ini adalah nontrivial, masalahnya adalah sangat lambat bila terdapat banyak kemungkinan kunci.

Saat ini dapat diketahui dengan beberapa jam kerja bagaimana proses pembangkitan kerusakan pada MD5. Yaitu dengan membangkitkan dua byte *string* dengan hash yang sama. Dikarenakan terdapat bilangan yang terbatas pada keluaran MD5 (2128), tetapi terdapat bilangan yang tak terbatas sebagai masukannya. Hal ini harus dipahami sebelum kerusakan dapat ditimbulkan, walaupun diyakini bahwa menemukannya adalah hal yang sulit.

Sebagai hasilnya bahwa hash MD5 dari informasi tertentu tidak dapat lagi mengenalinya secara berbeda. Jika ditunjukkan informasi dari sebuah *public key,* hash MD5 tidak mengenalinya secara berbeda jika terdapat *public key* selanjutnya yang mempunyai hash MD5 yang sama.

Bagaimanapun juga, penyerangan tersebut memerlukan kemampuan untuk memilih kedua pesan kerusakan. Kedua pesan tersebut tidak dengan mudah untuk memberikan serangan *pre-image* menemukan pesan dengan hash MD5 yang sudah ditentukan, ataupun serangan *preimage* kedua, menemukan pesan dengan hash MD5 yang sama sebagai pesan yang diinginkan.

Hash MD5 lama yang dibuat sebelum serangan-serangan tersebut diungkap masih dinilai aman untuk saat ini. Khusunya pada *digital signature* lama masih dianggap layak pakai. Seorang *user* boleh saja tidak ingin membangkitkan atau mempercayai *signature* baru menggunakan MD5 jika masih ada kemungkinan kecil pada teks (kerusakan dilakukan dengan melibatkan pelompatan beberapa bit pada bagian 128 byte pada masukan *hash*) akan memberikan perubahan yang berarti.

Penjaminan ini berdasar pada posisi saat ini dari kriptoanalisis. Situasi bisa saja berubah secara tiba-tiba tetapi kemungkinan kerusakan dengan beberapa data yang belum ada adalah permasalahannya yang lebih susah lagi dan akan selalu butuh waktu untuk terjadinya transisi.

2.5 Tujuan dari Kriptografi

Seperti perkembangan ilmu kriptografi tujuan-tujuan dari kriptografi terus berkembang.

Bila pertama kali kriptografi dibuat hanya untuk keamanan data saja maka sekarang semakin banyak tujuan-tujuan yang ingin dicapai yaitu :

- Privasi
  Dimana orang lain tidak dapat mengetahui isi pesan yang kita kirim
- Autentikasi
  Penerima pesan dapat meyakinkan dirinya bahwa pesan yang diterima tidak mengalami perubahan dan memang berasal dari orang yang diinginkan
- Penerima pesan dapat meyakinkan pihak ketiga bahwa pesan yang diterima berasal dari orang yang diinginkan
- Tidak ada yang dapat berkomunikasi dengan pihak lain. Hanya dapat berkomunikasi dengan pihak yang diinginkan.
- Pertukaran bersama suatu nilai tidak akan dikeluarkan sebelum nilai lainnya diterima
- Koordinasi
  Dalam komunikasi dengan banyak pihak setiap pihak dapat berkoordinasi untuk tujuan yang sama walaupun terdapat pihak lain.

2.6 Prinsip Dasar Fungsi Hash Satu Arah

Fungsi hash satu arah adalah fungsi hash yang memiliki beberapa sifat keamanan tambahan sehingga dapat dipakai untuk tujuan keamanan data. Fungsi hash adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut dengan nilai hash.

Sifat-sifat fungsi hash kriptografi :

- Tahan *pre-image* : bila diketahui nilai hash *h* maka sulit (secara komputasi tidak layak) untuk mendapatkan *m* dimana *h* = hash (*m*)
- Tahan *pre-image* kedua: bila diketahui input m_1 maka sulit mencari input m2 (tidak sama dengan m1) yang menyebabkan hash (m1) = hash (m2)
- Tahan tumbukan : sulit menacari dua input yang berbeda m1 dan m2 yang menyebabkan hash (m1) = hash (m2)

Fungsi hash satu arah memiliki banyak nama fungsi seperti : fungsi pembanding, fungsi penyusutan, intisari pesan, sidik jari, *message integrity check* (MIC) atau pemeriksaan keutuhan pesan dan *manipulation detection code* (MDC) atau pendeteksi penyelewengan kode.

Fungsi hash satu arah dibuat berdasarkan ide tentang fungsi pemampatan. Untuk memperoleh nilai hash merupakan hal yang mudah. Kita dapat memperolehnya dari *pre-image,* tetapi

sangat sulit untuk membangkitkan *pre-image* dari nilai hash-nya.
Metode fungsi hash satu arah berfungsi untuk melindungi data dari modifikasi. Apabila kita ingin melindungi data dari modifikasi yang tidak terdeteksi, kita dapat menghitung hasil fungsi hash dari data tersebut, selanjutnya dapat menghitung hasil fungsi hash lagi dan membandingkannya dengan hasil yang pertama. Apabila berbeda maka terjadi perubahan data selama proses pengiriman.

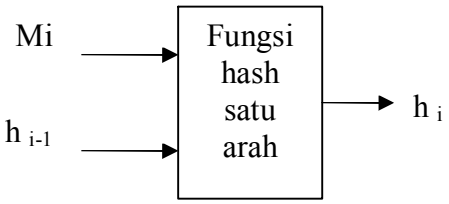
Sebagai contoh adalah bila si pengirim (A) akan mengirim pesan kepada penerima (B). Sebelum mengirim, A akan melakukan hash terhadap pesan yang akan dikirim untuk mendapatkan nilai hash kemudian ia akan mengirim pesan itu bersama dengan nilai hash nya. Lalu B

melakukan hash lagi untuk memperoleh nilai hash dari pesan yang telah diterimanya. Bila terjadi perbedaan nilai hash maka telah terjadi perubahan selama proses pengiriman.

Masukan dari fungsi hash satu arah adalah blok pesan dan keluaran dari blok text atau nilai hash sebelumnya ini dapat dilihat pada gambar 2.4 sehingga secara garis besar hash dari blok Mi adalah :

$$hi = f(Mi, h_{i-1})$$

Nilai hash ini bersama blok pesan berikutnya menjadi input berikutnya bagi fungsi pemampatan. Nilai hash keseluruhan adalah nilai hash dari blok paling akhir. *Pre-image* sedapatnya mengandung beberapa binary yang menggambarikan panjang dari masukan pesan Teknik ini digunakan untuk mengatasi masalah yang dapat terjadi bila pesan yang tidak sama memiliki nilai hash yang sama. Metode ini biasa disebut **MD-***streghtening* atau penguatan MD.



Gambar 2.4

2.7 Sistem Kriptografi MD5

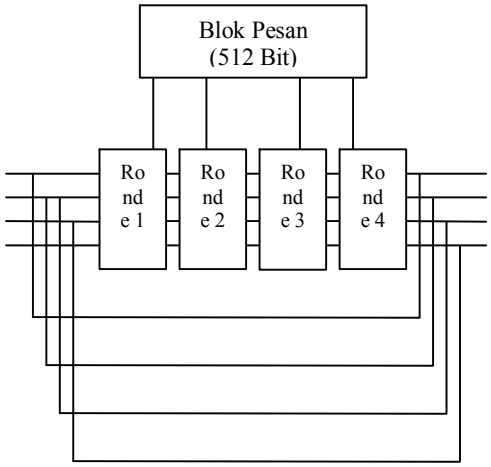
Pada bagian ini dijelaskan mengenai system kriptografi MD5 secara spesifik, yaitu sistem sistem kriptografi algoritma MD5 yang menjelaskan dari awal masukan hingga keluarannya.

2.7.1 Prinsip Dasar MD5

MD5 adalah salah satu penggunaan fungsi hash satu arah yang paling banyak digunakan. MD5 merupakan fungsi hash kelima yang dirancang oleh Ronald Rivest. MD5 merupakan pengembangan dari MD4 dimana terjadi penambahan satu putaran. MD5 memproses text masukan ke dalam blok-blok bit sebanyak 512

bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD5 berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128 bit yang di sebut nilai hash.

Pada gambar 2.5 terlihat simpul utama dari MD5. Simpul utama MD5 mempunyai blok pesan dengan panjang 512 bit yang masuk ke dalam 4 buah putaran. Hasil keluaran dari MD5 adalah berupa 128 bit dari byte terendah A dan tertinggi byte D.



Gambar 2.5

2.7.2 Algoritma MD5

Setiap pesan yang akan di enkripsi terlebih dahulu dicari berapa banyak bit yang terdapat pada pesan. Kita anggap sebanyak b bit. Disini adalah bit non negatif integer, b bisa saja nol dan tidak harus kelipatan delapan. Pesan dengan panjang b bit dapat digambarkan sebagai berikut:

m_0 m_1 ... m_(b-1)

Terdapat lima langkah yang dibutuhkan untuk menghitung intisari pesan. Adapun langkah-langkah tersebut dapat dijelaskan pada subbab berikut :

2.7.2.1 Menambahkan Bit

Pesan akan ditambahkan ke bit-bit tambahan sehingga panjang bit akan longruen dengan 448 mod 512. Hal ini berarti pesan akan mempunyai panjang yang kurang dari 64 bit dari kelipatan 512 bit. Penambahan bit selalu dilakukan walaupun panjang dari pesan sudah sama dengan 448, mod 512. Penambahan bit dilakukan dengan menambahkan ’1’ diawal dan diikuti ’0’

sebanyak yang diperlukan sehingga panjang pesan akan kongruen dengan 448, mod 512.

2.7.2.2 Penambahan Panjang Pesan

Setelah penambahan bit, pesan masih membutuhkan 64 bit agar sama dengan kelipatan 512 bit. 64 bit tersebut merupakan perwakilan dari b (panjang pesan sebelum penambahan bit dilakukan). Bit-bit ini ditambahkan ke dalam dua word (32 bit) dan ditambahkan dengan *low-order*

terlebih dahulu. Penambahan pesan ini biasa disebut juga **MD *Streghtening*** atau penguatan MD.

2.7.2.3 Inisialisasi MD5

Pada MD5 terdapat empat buah word 32 bit register yang berguna untuk menginisialisasi *message digest* pertama kali. Register-register ini di inisialisasikan dengan bilangan hexadesimal.

Word A : 01 23 45 67

Word B : 89 AB CD EF

Word C : FE DC BA 98

Word D : 76 54 32 10

Register-register ini biasa disebut dengan *Chain Variabel* atau variabel rantai.

2.7.2.4 Proses Pesan di Dalam Blok 16 Word

Pada MD5 juga terdapat empat buah fungsi nonlinier yang masing-masing digunakan pada tiap operasinya (satu fungsi untuk satu blok) yaitu:

F (X,Y, Z) = (X ^ Y) V ((~X) ^ Z)

G (X,Y, Z) = (X ^ Z) V (Y ^ (~Z))

H (X, Y, Z) = X ⊕ Y ⊕ Z

I (X, Y, Z) = Y ⊕ (X V (~Z))

(⊕ untuk XOR, ^ untuk AND, V untuk OR dan ~ untuk NOT).

Pada gambar 3.2 dapat dilihat satu buah operasi dari MD5 dengan operasi yang dipakai sebagai contoh adalah FF (a, b, c, d, Mj, s, ti) menunjukkan a = b + ((a + F(b,c,d) + Mj + ti) <<< s)

Bila Mj menggambarkan pesan ke – j dari sub blok (dari 0 sampai 15) dan <<< s menggambaekan bit akan digeser ke kiri sebanyak s bit maka keempat operasi dari masing-masing ronde adalah :

FF(a,b,c,d,Mj,s,ti) menunjukkan a=b + ((a + F(b,c,d) + Mj + ti) <<< s)

GG(a,b,c,d,Mj,s,ti) menunjukkan a=b + ((a + G(b,c,d) + Mj + ti) <<< s)

HH(a,b,c,d,Mj,s,ti) menunjukkan a=b + ((a + H(b,c,d) + Mj + ti) <<< s)

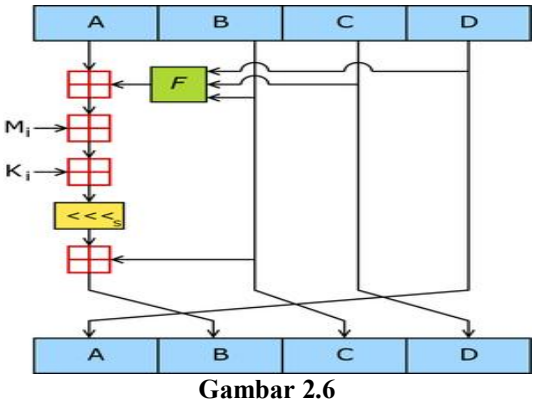
II(a,b,c,d,Mj,s,ti) menunjukkan a=b + ((a + I(b,c,d) + Mj + ti) <<< s)

Konstanta ti didapat dari integer 2^{32} , abs (sin (i)) dimana i dalam radian.

2.7.2.5 Keluaran MD5

Keluaran MD5 adalah 128 bit dari word terendah A dan tertinggi word D masing-masing 32 bit.

Secara umum algoritm dari struktur kriptografi pada MD5 adalah seperti yang digambarkan pada gambar 2.6



Gambar 2.6

MD5 terdiri atas 64 operasi yang dikelompokkan dalam empat putaran dari 16 operasi. F adalah fungsi non linier; satu fungsi digunakan pada

tiap-tiap putaran. Mi menunjukkan blok 32 bit dari masukan pesan, dan Ki menunjukkan konstanta 32 bit, berbeda untuk tiap-tiap operasi. ⋘ s menunjukkan perputaran bit kiri oleh s. s bervariasi untuk tiap-tiap operasi. ⊞ menunjukkan tambahan modulo 2^{32}. MD5 memproses variasi panjang pesan ke dalam keluaran 128 bit dengan panjang yang tetap. Pesan masukan dipecah menjadi dua gumpalan blok 512 bit. Pesan ditata sehingga panjang pesan dapat dibagi 512. Penataan bekerja sebagai berikut : bit tunggal pertama, 1, diletakkan pada akhir pesan. Proses ini diikuti dengan serangkaian nol (0) yang diperlukan agar

penjang pesan lebih dari 64 bit dan kurang dari kelipatan 512. Bit-bit sisa diisi dengan 64 bit integer untuk menunjukkan panjang pesan yang asli. Sebuah pesan selalu ditata setidaknya dengan 1 bit tunggal, seperti jika panjang pesan adalah kelipatan 512 dikurangi 64 bit untuk informasi panjang (panjang mod (512) = 448), sebuah blok baru dari 512 bit ditambahkan dengan 1 bit diikuti dengan 447 bit-bit nol (0) diikuti dengan panjang 64 bit.

Algoritma MD5 yang utama beroperasi pada kondisi 128 bit, dibagi menjadi empat *word* 32 bit, menunjukkan A, B, C, dan D. Operasi tersebut diinisialisasi juga untuk tetap konstan. Algoritma utama kemudian beroperasi pada masing-masing blok pesan 512 bit, masing-masing blok melakukan pengubahan terhadap kondisi. Pemrosesan blok pesan terdiri atas empat tahap, batasan putaran, dimana tiap putaran membuat 16 operasi serupa berdasar pada fungsi non linier F, tambahan modular dan rotasi ke kiri. Gambar 2.5 mengilustrasikan satu operasi dalam putaran. Ada empat macam kemungkinan fungsi F, berbeda dari yang digunakan pada tiap-tiap putaran.

Pseudocode pada algoritma MD5

```
//Catatan: Seluruh variable tidak pada
32-bit dan dan wrap modulo 2^32 saat
melakukan perhitungan

//Mendefinisikan r sebagai berikut
var int[64] r, k

r[ 0..15] := {7, 12, 17, 22,  7, 12, 17,
22,  7, 12, 17, 22,  7, 12, 17, 22}

r[16..31] := {5,  9, 14, 20,  5,  9, 14,
20,  5,  9, 14, 20,  5,  9, 14, 20}

r[32..47] := {4, 11, 16, 23,  4, 11, 16,
23,  4, 11, 16, 23,  4, 11, 16, 23}



r[48..63] := {6, 10, 15, 21,  6, 10, 15,
21,  6, 10, 15, 21,  6, 10, 15, 21}

//Menggunakan bagian fraksional biner
dari integral sinus sebagai konstanta:
for i from 0 to 63
    k[i] := floor(abs(sin(i + 1)) × 2^32)

//Inisialisasi variabel:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE

var int h3 := 0x10325476

//Pemrosesan awal:
append "1" bit to message
append "0" bits until message length in
bits ≡ 448 (mod 512)
append bit length of message as 64-bit
little-endian integer to message

//Pengolahan pesan paada kondisi gumpalan
512-bit:
for each 512-bit chunk of message
    break chunk into sixteen 32-bit
little-endian words w(i), 0 ≤ i ≤ 15

    //Inisialisasi nilai hash pada
gumpalan ini:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3

    //Kalang utama:
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            f := (b and c) or ((not b)
and d)
            g := i
        else if 16 ≤ i ≤ 31
            f := (d and b) or ((not d)
and c)
            g := (5×i + 1) mod 16
        else if 32 ≤ i ≤ 47
            f := b xor c xor d
            g := (3×i + 5) mod 16
        else if 48 ≤ i ≤ 63
            f := c xor (b or (not d))
            g := (7×i) mod 16

        temp := d
        d := c
        c := b
        b := ((a + f + k(i) + w(g))
leftrotate r(i)) + b
        a := temp


 //Tambahkan hash dari gumpalan sebagai
hasil:
    h0 := h0 + a
```

```
h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
```

```
var int digest := h0 append h1 append h2
append h3 //(diwujudkan dalam little-
endian)
```

Catatan: Meskipun rumusan dari yang tertera pada RFC 1321, berikut ini sering digunakan untuk meningkatkan efisiensi:

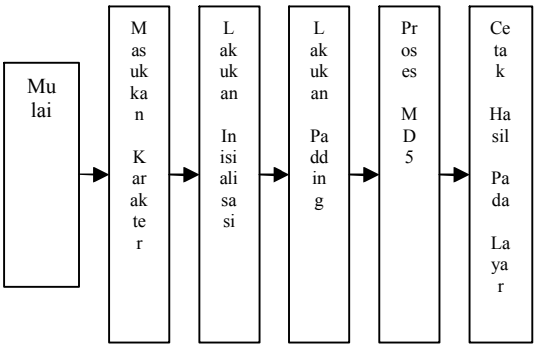
```
(0  ≤ i ≤ 15): f := d xor (b and (c xor
d))
(16 ≤ i ≤ 31): f := c xor (d and
(b xor c))
```

3. Aplikasi

Aplikasi yang digunakan adalah aplikasi 32 bit yang berjalan pada sistem operasi *Windows* 98 ke atas.

3.1 Proses MD5 dengan Masukan Berupa String

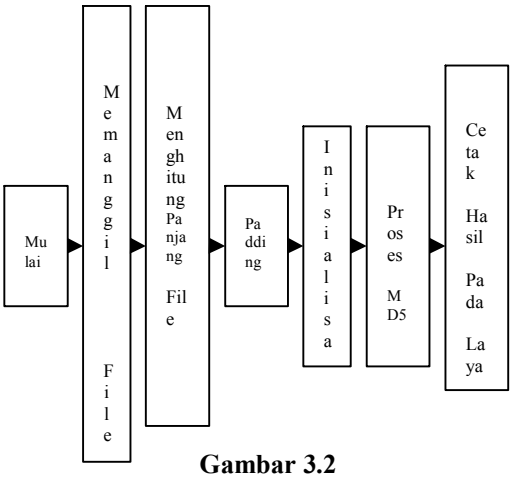
Proses MD5 dengan masukan berupa string adalah proses yang masukannya berupa karakter-karakter yang dimasukkan melalui *keyboard*. Hal ini dapat dilihat dari gambar 3.1



Gambar 3.1

3.2 Proses MD5 dengan Masukan Berupa File

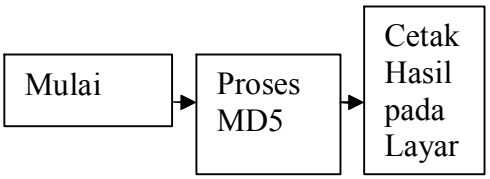
Proses MD5 dengan masukan berupa file adalah proses MD5 yang masukannya memanggil file yang kemudian dihitung berapa panjang bit nya, file ini dianggap sebagai bit memori sehingga masukannya tidak dipengaruhi oleh eksistensinya. Kemudian dilakukan proses MD5. Dapat dilihat pada gambar 3.2



Gambar 3.2

3.3 Proses MD5 sebagai Test Suite

Test suite dilakukan untuk mengetahui apakah program yang dibuat ini sudah benar atau tidak. Sebagai perbandingannya digunakan hasil yang telah dibuat oleh Ronald Rivest yang telah didefenisikan pada RFC 1321. Pada gambar 3.3 dapat dilihat bahwa masukan dari MD5 telah ditentukan sehingga tinggal membandingkan hasil pada layar dengan yang tercantum pada RFC 1321.



Gambar 3.3

3.4 Contoh Aplikasi

Pada contoh aplikasi ini terdapat dua buah aplikasi yang dipakai yaitu simpan dan tampilkan. Simpan adalah menyimpan data berupa nama, nilai ujian dan nilai hash-nya. Proses akhir akan melakukan perhitungan MD5 dari data yang disimpan untuk mendapatkan nilai hash-nya yang kemudian membandingkan nilai hash-nya dengan nilai hash dari data semula, apabila nilai hash yang didapat sama maka data akan ditampilkan. Tetapi bila nilai hash yang didapat berbeda maka pada form akhir akan ditampilkan pesan bahwa terdapat kesalahan atau perubahan.

3.5 Pengukuran Kecepatan Aplikasi

Pengukuran kecepatan aplikasi merupakan sebuah analisa yang akan dipakai untuk mengukur tingkat kecepatan dari proses mencari nilai hash dari file dengan menggunakan aplikasi MD5.

Adapun rumus yang dipakai dalam aplikasi untuk menghitung kecepatan mencari nilai hash tersebut adalah :

$$\text{Kecepatan} = \frac{\text{Besar Ukuran File}}{\text{Lama Waktu Proses}}$$

Satuan dari kecepatan enkripsi ini adalah Mbytes/detik.
Dalam analisa kecepatan ini, akan dilakukan sebanyak lima kali pengambilan waktu terbaik yang diperlukan untuk enkripsi dalam setiap filenya kemudian dicari waktu rata-ratanya.

Besar file dan tabel perbandingan kecepatan maksimum dengan kecepatan rata-rata terhadap besar file.

4. Hash-Hash MD5

Hash-hash MD5 sepanjang 128 bit (16 byte), yang dikenal juga sebagai ringkasan pesan, secara tipikal ditampilkan dalam bilangan heksadesimal 32 digit. Berikut ini merupakan contoh pesan ASCII sepanjang 43 byte sebagai masukan dan hash MD5 terkait :

MD5 (”The quick brown fox jump over the lazy dog”) = 9e107d9d372bb6826bd81d3542a419d6

Bahkan perubahan yang kecil pada pesan akan (dengan probabilitas lebih) menghasilkan hash benar-benar berbeda, misalnya pada kata ”dog” , huruf d diganti menjadi c :

MD5(”The quick brown fox jump over the lazy cog”) = 1055d3e698d289f2af8663725127bd4b

Hash dari panjang nol adalah :

MD5(“”)=d41d8cd98f00b204e9800998ecf8427e

5. Analisis Kecepatan MD5

Analisis kecepatan disini adalah analisis tentang kecepatan aplikasi dalam mengenkrip file untuk mencari nilai hash. Analisis dilakukan untuk mencari kecepatan aplikasi dengan masukan file yang mempunyai besar yang berbeda-beda.

Pengujian ini dilakukan dengan cara mengenkrip file sebanyak 31 buah file dengan besar file yang berbeda-beda. Setiap file ini dilakukan pengambilan waktu eksekusi sebanyak lima kali kemudian dicari waktu rata-ratanya.

6. Pengujian Integritas

Ringkasan MD5 digunakan secara luas dalam dunia perangkat lunak untuk menyediakan semacam jaminan bahwa file yang diambil (*download*) belum terdapat perubahan. Seorang *user* dapat membandingkan MD5 sum yang di publikasikan dengan *checksum* dari file yang diambil. Dengan asumsi bahwa *checksum* yang dipublikasikan dapat dipercaya akan keasliannya. Seorang *user* dapat secara yakin bahwa file tersebut adalah file yang sama dengan file yang dirilis oleh para developer, jaminan perlindungan dari virus komputer yang ditambahkan pada perangkat lunak. Bagaimana pun juga seringkali kasus yang terjadi bahwa *checksum* yang dipublikasikan tidak dapat dipercaya. Dalam hal ini MD5 hanya mampu melakukan *error-checking*. MD5 akan mengenali file yang didownload tidak sempurna atau tidak lengkap.
Hasil pengujian digambarkan dengan tabel hasil pengujian, yang kemudian dijabarkan dengan grafik hasil uji coba terhadap file yaitu grafik kecepatan aplikasi terhadap besar file, grafik rata-rata waktu eksekusi terhadap besar file dan tabel perbandingan kecepatan maksimum dengan kecepatan rata-rata terhadap besar file.

7. Kesimpulan

Beberapa kesimpulan yang dapat diperoleh adalah :

- *Message Digest 5* (MD5) adalah sebuah fungsi hash satu arah yang mengubah masukan dengan panjang variabel menjadi keluaran dengan panjang tetap yaitu 128 bit

- Rata-rata kecepatan dari program aplikasi MD5 adalah 7,1633 Mbytes/detik
- Apikasi yang dibuat hanya efektif digunakan untuk ukuran file kurang dari 40 Mbytes
- Sumber daya komputer berpengaruh terhadap kecepatan enkripsi

DAFTAR PUSTAKA

[1] Cryptographi. (2006). http://en.wikipedia.org/wiki/Cryptography. Tanggal Akses 31 Desember 2006 pukul 15.30

[2] Kriptografi. (2006). *http://id.wikipedia.org/wiki/Kriptografi*. Tanggal Akses: 31 Desember 2006 pukul 14.00

[3] Kriptografi. (2006). *http://nanangblog.inc.md/?p=31*. Tanggal Akses : 31 Desember 2006 pukul 15.00

[4] Message Digest 5. (2006). http://en.wikipedia.org/wiki/MD5. Tanggal Akses 28 Desember 2006 pukul 17.00

[5] Munir, Rinaldi. 2004. Diktat Kuliah IF2153 Matematika Diskit. Deparetemen Teknik Informatika, Istitut Teknologi Bandung.

[6] RSA SecurID Authentication in Action. (2006). http://www.rsasecurity.com/. Tanggal Akses 28 Desember 2006 pukul 16.30

[7] The MD5 Message-Digest Algorithm. (2006). http://www.ietf.org/rfc/rfc1321.txt. Tanggal Akses 29 Desember 2006 pukul 17.00