FUNGSI HASH PADA KRIPTOGRAFI

Aridarsyah Eka Putra

Program Studi Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha 10, Bandung, 40132
e-mail: if17058@students.if.itb.ac.id, black_crystae@yahoo.co.id

ABSTRAK

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan data. Algoritma kriptografi modern merupakan algoritma yang beroperasi dalam metode bit. Algoritma enkripsi dan dekripsi memproses semua data dan informasi dalam bentuk rangkaian bit. Rangkaian bit yang menyatakan plainteks dienkripsi menjadi cipherteks dalam bentuk rangkaian bit, demikian sebaliknya.
Algoritma kriptografi modern terbagi ke dalam tiga, yaitu algoritma simetri (block cipher dan stream cipher), algoritma asimetri dan *fungsi hash*. *Fungsi hash* dalam dunia ilmu komputer digunakan untuk penyimpanan data pada database dan digunakan untuk enkripsi dan dekripsi data dalam kriptografi. Makalah ini membahas tentang *fungsi hash* kriptografis, sifat-sifat *fungsi hash* dan penerapannya pada dunia kriptografi.

Kata kunci: hash, kriptografi.

1. PENDAHULUAN

Sejak digunakan dalam perang dunia, kriptografi terus mengalami perkembangan. Jika dahulu kriptografi menyamarkan teks asli (plainteks) menjadi teks sandi (cipherteks) dengan keluaran berupa karakter huruf yang dikenal dengan kriptografi klasik, kini teks sandi yang dihasilkan ialah berupa mode bit yang dikenal dengan kriptografi modern. Perkembangan ini tidak lepas dari penggunaan komputer digital yang merepresentasikan data dalam bentuk biner [1]. Kriptografi modern kini lebih banyak digunakan, karena cenderung lebih aman daripada kriptografi klasik.

Algoritma kriptografi modern terbagi ke dalam algoritma simetri, algoritma asimetri dan *fungsi hash*. Algoritma simetri ialah algoritma yang mempergunakan kunci yang sama pada enkripsi dan dekripsinya. Algoritma ini merupakan algoritma yang paling umum digunakan. Algoritma asimetri ialah algoritma yang mempergunakan kunci yang berbeda pada enkripsi dan dekripsinya [2]. *Fungsi hash* adalah fungsi yang menerima masukkan string yang panjangnya sembarang dan mentransformasikannya menjadi string keluaran yang panjangnya tetap (nilai hash), umumnya berukuran jauh lebih kecil daripada string masukkannya [3]. Ide dasar dari *fungsi hash* adalah menghitung nilai hash dari kunci atau nilai asli, kemudian membandingkan kunci atau nilai asli dengan isi pada memori yang beralamat nomor hashnya tanpa harus memeriksa isi tabel satu per satu sehingga lebih efisien.

2. PEMBAHASAN

Funsgi hash adalah sebuah fungsi matematika yang menggabungkan operasi logika untuk biner, teori bilangan, dan komposisi fungsi.
Fungsi hash Kriptografis adalah *fungsi hash* yang memiliki beberapa sifat keamanan tambahan sehingga dapat dipakai untuk tujuan keamanan data [4]. Umumnya digunakan untuk keperluan autentikasi dan integritas data. Persamaan *fungsi hash* secara matematis
ialah h = H(M); M = pesan ukuran sembarang, h = nilai hash (hash value) atau pesan-ringkas (message-*digest*); h <<<< M. Misalkan: size(M) = 1 MB, size(h) = 128 bit.

Nama lain *fungsi hash* adalah:
- *fungsi kompresi (compression function)*
- *cetak-jari (fingerprint)*
- *cryptographic checksum*
- *message integrity check (MIC)*
- *manipulation detection code (MDC)*

(Rinaldi Munir).

Sifat-Sifat *Fungsi hash* Kriptografi
· Tahan preimej (*Preimage resistant*): bila diketahui nilai hash h maka sulit (secara komputasi tidak layak) untuk mendapatkan m dimana h = hash(m).
· Tahan preimej kedua (*Second preimage resistant*): bila diketahui input m1 maka sulit mencari input m2 (tidak sama dengan m1) yang menyebabkan hash(m1) = hash(m2).
· Tahan tumbukan (*Collision-resistant*): sulit mencari dua input berbeda m1 dan m2 yang menyebabkan hash(m1) = hash(m2). [5]

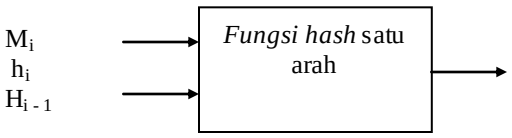
Fungsi hash merupakan fungsi yang bersifat satu arah dimana jika kita masukkan data, maka dia akan menghasilkan sebuah "checksum" atau "fingerprint" dari data tersebut dan tidak dapat diubah kembali menjadi pesan semula. Sebuah pesan yang dilewatkan ke *fungsi hash* akan menghasilkan keluaran yang disebut Message Authenticated Code (MAC). Dilihat dari sisi matematik, *fungsi hash* memetakan satu set data ke dalam sebuah set yang lebih kecil dan terbatas ukurannya [6].

Sifat-sifat *fungsi hash* satu-arah adalah sebagai berikut:
1.Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2.H menghasilkan nilai (h) dengan panjang tetap (*fixed-length output*).
3.H(x) mudah dihitung untuk setiap nilai x yang diberikan.
4.Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga H(x) = h. Itulah sebabnya fungsi H dikatakan *fungsi hash* satu-arah (*one-way hash function*).
5.Untuk setiap x yang diberikan, tidak mungkin mencari y ¹ x sedemikian sehingga H(y) = H(x).
6.Tidak mungkin mencari pasangan x dan y sedemikian sehingga H(x) = H(y).

Masukan *fungsi hash* adalah blok pesan (M) dan keluaran dari hashing blok pesan sebelumnya,

$$hi = H(Mi, hi - 1)$$

Skema *fungsi hash* ditunjukkan pada Gambar 2.1.



Gambar 1. *Fungsi hash* satu arah

Aplikasi *fungsi hash* satu arah bertujuan untuk:
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

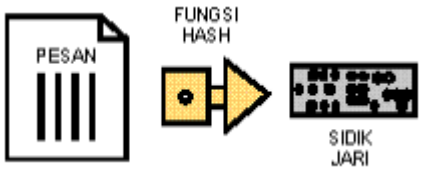
2. Menghemat waktu pengiriman.
· Misal untuk memverifikasi sebuah salinan arsip dengan arsip asli.
· Salinan dokumen berada di tempat yang jauh dari basisdata arsip asli
· Daripada mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih mangkus mengirimkan *message digest*-nya.
· Jika *message digest* salinan arsip sama dengan *message digest* arsip asli, berarti salinan arsip tersebut sama dengan arsip master.
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). [3]

Kelebihan fungsi hash dibandingkan fungsi enkripsi pada umumnya :
1. Hasil dari fungsi hash panjangnya tetap, panjang masukan tidak akan mempengaruhi panjang nilai hash.
2. Karena tidak merubah data asli, tidak diperlukan proses dekripsi
3. Perubahan sekecil apapun pada data asli akan membuat nilai hash yang sangat jauh berbeda, sehingga cukup mudah untuk memeriksa keaslian data
Kekurangan fungsi hash diantaranya :
1. Memiliki kemungkinan untuk terjadi bentrokan. Hal ini tidak dapat dihindari untuk semua fungsi hash, namun ada beberapa fungsi hash dibuat khusus untuk menghindari terjadinya bentrokan.
2. Fungsi hash adalah fungsi satu arah, jadi jika kita hanya mendapat sebuah nilai hash, kita tidak bisa mengembalikannya menjadi data yang asli. Hal ini dipersulit dengan kemungkinan terjadinya bentrokan.
3. Tingkat keamanan suatu fungsi hash dinilai berdasarkan jumlah kemungkinan nilai hash, yaitu 2^n, dengan n adalah panjang nilai hash dalam bit. Jadi semakin panjang nilai hash semakin aman.

Kaitan fungsi hash dan matematika diskrit :
Seperti sudah disebutkan bahwa fungsi hash adalah sebuah fungsi matematika yang menggabungkan operasi logika untuk biner, teori bilangan, dan komposisi fungsi.
Teori bilangan digunakan dalam fungsi kompresi dalam fungsi hash. Contoh fungsi kompresi paling sederhana adalah modulo. Dengan modulo atau mod, kita dapat mengkompresi data dengan panjang berapapun menjadi panjang yang kita inginkan. Contoh : 1568 mod 10 = 8, kompresi dari empat digit menjadi satu digit.
Operasi logika dan aljabar boolean digunakan dalam fungsi terkecil dalam hash. Yang berguna untuk melakukan pengkodean atau pengacakan. Fungsi dan komposisi fungsi digunakan untuk membentuk sebuah funshi hash, karena sebenarnya fungsi hash terdiri dari beberapa fungsi yang lebih sederhana.

2.1 Sidik Jari

Kini akan dibahas mengenai keutuhan pesan saat dikirimkan. Saat pengirim pesan hendak mengirimkan pesannya, dia harus membuat sidik jari dari pesan yang akan dikirim untuk penerima pesan. Pesan (yang besarnya dapat bervariasi) yang akan di-hash disebut preimej, sedangkan outputnya yang memiliki ukurannya tetap, disebut nilai hash. Kemudian, melalui saluran komunikasi yang aman, dia mengirimkan sidik jarinya kepada kepada penerima. Setelah penerima menerima pesan si pengirim – tidak peduli lewat saluran komunikasi yang mana – penerima kemudian juga membuat sidik jari dari pesan yang telah diterimanya dari pengirim. Kemudian kedua sidik jari dibandingkan. Jika kedua sidik jari itu identik, maka penerima dapat yakin bahwa pesan itu utuh tidak diubah-ubah sejak dibuatkan sidik jari yang diterimanya.
Jika pesan sudah diubah, tentunya akan menghasilkan nilai hash yang berbeda.
Fungsi hash untuk membuat sidik jari tersebut dapat diketahui oleh siapapun, tak terkecuali, sehingga siapapun dapat memeriksa keutuhan dokumen atau pesan tertentu.

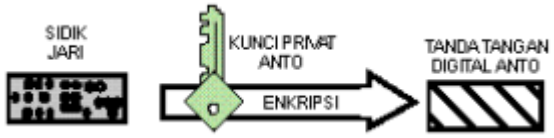


Gambar 2. Membuat sidik jari pesan

Tak ada algoritma rahasia dan umumnya tak ada pula kunci rahasia.
Jaminan dari keamanan sidik jari berangkat dari kenyataan bahwa hampir tidak ada dua pre-image yang memiliki nilai hash yang sama. Inilah yang disebut dengan sifat bebas *collision* dari suatu *fungsi hash* yang baik Selain itu,sangat sulit untuk membuat suatu preimej jika hanya diketahui hash-valuenya saja.

2.2 Tanda Tangan Digital

Penerima pesan dapat merasa yakin bahwa sidik jari yang datang bersama pesan yang diterimanya memang berkorelasi. Namun bagaimana penerima pesan dapat merasa yakin bahwa pesan itu beasal dari pengirim tanpa ada gangguan dari yang tidak diinginkan pada saat pesan melintas melalui saluran komunikasi?
Untuk mencegah terjadinya pemalsuan sidik jari, pengirim pesan membubuhkan tanda tangannya pada pesan tersebut. Dalam dunia elektronik, pengirim pesan membubuhkan tanda tangan digitalnya pada pesan yang akan dikirimkan untuk penerima sehingga penerima dapat merasa yakin bahwa pesan itu memang dikirim oleh pengirim.
Sifat yang diinginkan dari tanda tangan digital diantaranya adalah:
1. Tanda tangan itu asli (otentik), tidak mudah ditulis /ditiru oleh orang lain. Pesan dan tanda tangan pesan tersebut juga dapat menjadi barang bukti, sehingga penandatangan tak bisa menyangkal bahwa dulu ia tidak pernah menandatanganinya.
2. Tanda tangan itu hanya sah untuk dokumen (pesan) itu saja. Tanda tangan itu tidak bisa dipindahkan dari suatu dokumen ke dokumen lainnya. Ini juga berarti bahwa jika dokumen itu diubah, maka tanda tangan digital dari pesan tersebut tidak lagi sah.
3. Tanda tangan itu dapat diperiksadengan mudah.
4. Tanda tangan itu dapat diperiksa oleh pihak-pihak yang belum pernah bertemu dengan penandatangan.
5. Tanda tangan itu juga sah untuk kopi dari dokumen yang sama persis.
Meskipun ada banyak skenario, ada baiknya kita perhatikan salah satu skenario yang cukup umum dalam penggunaan tanda tangan digital. Tanda tangan digital memanfaatkan *fungsi hash* satu arah untuk menjamin bahwa tanda tangan itu hanya berlaku untuk dokumen yang bersangkutan saja. Bukan dokumen tersebut secara keseluruhan yang ditandatangani, namun biasanya yang ditandatangani adalah sidik jari dari dokumen itu beserta *timestamp*-nya dengan menggunakan kunci privat. *Timestamp* berguna untuk menentukan waktu pengesahan dokumen.



Gambar 3. Pembuatan tanda tangan digital

Keabsahan tanda tangan digital dapat diperiksa oleh penerima pesan. Pertama-tama penerima membuat kembali sidik jari pesan yang diterimanya. Lalu penerima mendekripsikan tanda tangan digital pengirim untuk mendapatkan sidik jari yang asli. Penerima kemudian membandingkan kedua sidik jari tersebut. Jika kedua sidik jari tersebut sama, maka dapat diyakini bahwa pesan tersebut ditanda tangani oleh pengirim pesan.

2.3. Sertifikat Digital

Pengiriman pesan dengan tanda tangan digital dapat dilakukan jika pengirim dan penerima pesan sudah saling mengenal dan mengetahui kunci untuk ekripsi dan dekripsi pesan.
Jika pengirim dan penerima pesan belum saling mengenal atau tidak saling mengetahui kunci untuk enkripsi dan dekripsi, maka perlu adanya pihak ketiga. Setiap anggota jaringan diasumsikan telah memiliki saluran komunikasi pribadi yang aman dengan pihak ketiga. Saluran inilah yang dimanfaatkan untuk mengirim kunci publik penerima ke pengirim pesan (dan sebaliknya). Pihak ketiga ini menjadi penjamin keabsahan kunci jika pengirim dan penerima pesan sebelumnya tidak pernah

bertukar kunci publik. Skenario ini tetap membutuhkan kunci-kunci kriptografi (baik itu simetris ataupun asimetris) untuk pengamanan saluran komunikasi antara pihak ketiga dengan pengirim atau penerima pesan.
Masalah di atas dapat dipecahkan dengan penggunaan sertifikat digital. Jadi pihak ketiga sukup menanda tangani kunci publik milik setiap orang di jaringan tersebut. Sebenarnya dalam sertifikat tersebut tak hanya berisi kunci publik, namun dapat berisi pula informasi penting lainnya mengenai jati diri pemilik kunci publik, seperti misalnya nama, alamat, pekerjaan, jabatan, perusahaan dan bahkan hash dari suatu informasi rahasia. Semua orang mempercayai otoritas pihak ketiga dalam memberikan tanda tangan, sehingga orang-orang dalam jaringan itu merasa aman menggunakan kunci publik yang telah ditandatangani pihak ketiga.

Gambar 4. Contoh sertifikat digital

Jika seseorang berhasil mencuri sertifikat digital yang dipertukarkan antara pengirim dan penerima pesan, serta menggantinya dengan sertifikat digital milik dirinya sendiri, maka pengirim dan penerima pesan dapat segera melihat bahwa sertifikat digital yang diterimanya bukan 'lawan bicara' yang semestinya.
Serangan terhadap sistem yang memiliki pengamanan dengan sertifikat digital sulit dilakukan. Secara teoritis keunggulan dari tanda tangan digital adalah kemampuan untuk melakukan proses otentikasi secara off-line. Pemeriksa cukup memiliki kunci publik dari OS utama untuk mengetahui sah-tidaknya kunci publik dari lawan bicaranya. Selain itu untuk meningkatkan keamanan, kunci publik OS utama bisa saja diintegrasikan dalam program aplikasi. Namun kenyataannya, karena ada kemungkinan sertifikat digital tersebut hilang, tercuri atau identitas pemilik sertifikat berubah (perubahan alamat surat elektronik atau nomor KTP misalnya), maka sertifikat digital perlu diperiksa keabsahannya dengan melihat daftar sertifikat terbatalkan (certificate revocation list) yang disimpan oleh OS.[4]

3. METODE

Beberapa algoritma *fungsi hash* kriptografi :

Tabel 1 Algoritma *Fungsi hash* Kriptografi

Algoritma	Ukuran *message*	Ukuran blok pesan	Kolisi
	diggest (bit)		
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

Dalam makalah ini hanya akan diulas algoritma Secure Hash Algorithm (SHA), *Message digest* (MD5).

3.1. Secure Hash Algortihm (SHA)

NIST bersama NSA (*National Security Agency*) mendesain SHA untuk digunakan sebagai komponen *Digital Signature Standard* (DSS). Standar hash adalah *Secure Hash Standard* (SHS) dengan SHA sebagai algoritma yang digunakan. Dapat disimpulkan SHS adalah standar sedangkan SHA adalah algoritma.
Standar menetapkan SHA yang diperlukan untuk menjamin keamanan DSA. Ketika pesan dengan sembarang panjang < 264 bit dimasukkan, SHA menghasilkan 160 bit keluaran yang disebut *message digest* (MD). MD ini kemudian dimasukkan ke dalam DSA, yang menghitung tanda tangan digital untuk pesan tersebut. Penandatanganan MD (dan bukannya penandatanganan secara langsung) sering kali meningkatkan efisiensi proses, karena MD biasanya jauh lebih kecil dibanding pesan aslinya. MD pesan yang sama seharusnya dapat diperoleh oleh pemeriksa tanda tangan ketika menerima pesan dari pengirim dengan cara memasukkan pesan tersebut ke *fungsi hash* SHA. SHA dikatakan aman karena didesain supaya secara matematis tidak dimungkinkan untuk mendapatkan pesan aslinya bila diberikan hashnya atau tidak mungkin mendapatkan dua pesan yang berbeda yang menghasikan MD yang sama. SHA dibuat berdasarkan rancangan yang serupa dengan MD4 yang dibuat oleh Prof. Ronald L. Rivest dari MIT, SHA menghasilkan kelaran sidik jari 160 bit, lebih panjang dibanding MD5 [2].
Ketiga algoritma SHA adalah struktur yang berbeda dan unggul seperti SHA-0, SHA-1 dan SHA-2. SHA-2 menggunakan algoritma identik dengan variabel *digest* ukuran yang terkenal sebagai SHA-224, SHA-256, SHA 384, dan SHA-512 [7].

Cara Kerja SHA

Mula-mula pesan diberi tambahan untuk membuat panjangnya menjadi kelipatan 512 bit (L x 512). Jumlah bit data asal adalah K bit. Tambahkan bit '1' kemudian tambahkan bit '0' secukupnya sampai 64 bit kurangnya dari kelipatan 512(512-64=448), yang disebut juga sebagai kongruen dengan 448 (mod 512). Akhirnya tambahkan 64 bit yang mrnyatakan panjang pesan sebelum diberi tambahan. Pesan dibagi-bagi menjadi blok-blok berukuran 512 bit dan setiap blok diolah. Keluaran setiap blok digabungkan dengan keluaran blok berikutnya. Sehingga akhirnya diperoleh *digest*. Pengubahan satu huruf dapat menghasilkan cipher yang jauh berbeda [2].

Contoh enkripsi SHA:

Plainteks: "I heard you crying loud all the way accross the town"
Cipherteks:
eaeb1b86f6e41c1b40d7c288f6d7fbff3f948a6e
Menghilangkan huruf "d" pada kata "heard
Plainteks: "I hear you crying loud all the way accross the town"
Cipherteks:
d9aee3365c0ef380f4021fd618b4d4ea3ad9e5a4

3.2. *Message digest* 5 (md5)

Salah satu *fungsi hash* yang banyak digunakan dalam keamanan jaringan komputer dan internet adalah Message *Digest* (MD) versi 5 yang dirancang oleh Ron Rivest.

MD5 di desain oleh Ronald Rivest, salah satu pembuat algoritma RSA, pada tahun 1991 untuk menggantikan hash function sebelumnya, MD4. Pada tahun 1996, sebuah kecacatan ditemukan dalam desainnya, walau bukan kelemahan fatal, pengguna kriptografi mulai menganjurkan menggunakan algoritma lain, seperti SHA-1 (klaim terbaru menyatakan bahwa SHA-1 juga cacat). Pada tahun 2004, kecacatan-kecacatan yang lebih serius ditemukan menyebabkan penggunaan algoritma tersebut dalam tujuan untuk keamanan jadi makin dipertanyakan [8]. MD5 merupakan kelanjutan MD4 yang dirancang dengan tujuan:

· **Keamanan**. Secara perhitungan, ate, atis tidak dimungkinkan untuk mendapatkan dua pesan yang memiliki hash yang sama. Tidak ada seangan yang lebih efisien dibanding brute force.

· **Keamanan secara langsung**. Keamanan MD5 tidak didasarkan pada suatu asumsi, seperti kesulitan pemfaktoran.

· **Kecepatan**. MD5 sesuai untuk diimplementasikan dengan perangkat lunak yang berkecepatan tinggi, karena berdasar pada sekumpulan manipulasi operan 32-bit.

· **Kesederhanaan dan Kompak**. MD5 sederhana tanpa struktur data atau program yang komplek. Hash-hash MD5 sepanjang 128-bit (16-byte), yang dikenal juga sebagai ringkasan pesan (MD), secara tipikal ditampilkan dalam bilangan heksadesimal 32-digit.

Cara Kerja MD5

Cara kera MD5 serupa dengan SHA. MD5 mengolah masukan yang berupa blok 512 bit, dibagi kedalam 16 subblok berukuran 32-bit. Keluaran algoritma diset menjadi 4 blok yang masing-masing berukuran 32 bit, dan setelah digabungkan akan membentuk nilai hash 128 bit. Pengubahan satu huruf dapat menghasilkan cipher yang jauh berbeda [2].

Contoh enkripsi MD5:

Plainteks: "I heard you crying loud all the way accross the town"
Cipherteks:
add0135ce7fce6bbf36d240887470789

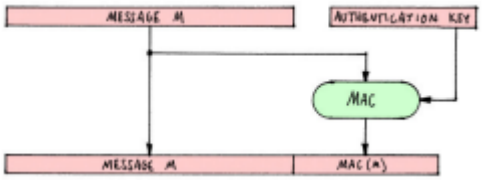
Menghilangkan huruf "d" pada kata "heard"
Plainteks: "I hear you crying loud all the way accross the town"
Cipherteks:
515c93f6d4692a2df6181702b7415c95

3.3. MAC (*Message Authentification Code*)

MAC adalah salah satu fungsi hash dengan kunci yang dibuat khusus untuk melakukan pemeriksaan keaslian pesan atau message authentification. Algoritma MAC mulai berkembang sekitar tahun 1995 dan banyak digunakan secara umum karena memiliki tingkat keamanan yang layak untuk masa itu. Pada awalnya MAC menggunakan kunci dengan panjang 32-bit atau 56-bit, namun seiring perkembangan jaman, panjang kunci MAC mengalami penyesuaian.

Algoritma MAC ada dua jenis, yaitu :

1. MAC yang mengunakan *cipher-block-chaining*. Algoritma ini menggunakan algoritma *data encryption standard* (DES) sebagai dasarnya, sehingga kunci MAC tersebut adalah kunci DES dengan panjang bit yang sesuai.
2. MAC yang menggunakan fungsi hash tanpa kunci. Algoritma ini menggunakan fungsi hash tanpa kunci yang sudah ada, misalnya MD5, SHA-1, atau SH-256. Algoritma ini biasa disebut sebagai *Hash Message Authentification Code* (HMAC).



Gambar 5 Skema Umum MAC

Fungsi MAC secara garis besar serupa dengan fungsi hash tanpa kunci pada umumnya, tapi terdapat perbedaan pada cara penggunaannya. Pada fungsi hash tanpa kunci, pengirim pesan dan penerima pesan hanya perlu memiliki fungsi hash yang sama, yang sebaiknya dirahasiakan. Pada saat pengiriman pesanpengirim cukup mengirimkan pesan asli yang telah ditambahkan nilai hash, pesan ini tidak perlu dirahasiakan. Namun

pada penggunaan MAC, selain mengirimkan pesan yang telah ditambahkan kode MAC (tidak perlu dirahasiakan), pengirim juga harus mengirim kunci pemeriksaan keaslian (atuhentification key) atau disebut juga kunci MAC secara rahasia. Dengan begitu, proses pengiriman menjadi dua tahap, tetapi dengan kenaikan tingkat keamanan.

4. KESIMPULAN

Fungsi hash merupakan fungsi yang bersifat satu arah yang akan menghasilkan sebuah "checksum" atau "fingerprint" dari data tersebut dan tidak dapat diubah kembali menjadi pesan semula. Umumnya digunakan untuk keperluan autentikasi dan integritas data. Kelebihan *fungsi hash* ialah menjaga integritas data, menghemat waktu pengiriman dan menormalkan panjang data yang beraneka ragam. Kelebihan inilah yang menjadi alasan mengapa saat ini *fungsi hash* banyak digunakan, khususnya dalam bidang keamanan data. Pemanfaatannya digunakan dalam aplikasi algoritma SHA dan MD5 yang kini banyak digunakan oleh kriptografer.

REFERENSI

[1] RJP, Rizky MT. Diktak Kuliah Algoritma Kriptografi Modern. Program Studi Ilmu Komputer Universitas Pendidika Indonesia. 2009

[2] Kurniawan, Yusuf Ir. MT. KRIPTOGRAFI Keamanan Internet dan Jaringan Komunikasi. Penerbit Informatika. Bandung. 2004

[3] Munir, Rinaldi. Diktat Kuliah IF5054 Kriptografi. Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika , Institut Teknologi Bandung. 2006

[4] http://www.geocities.com/amwibowo/resource /komparasi/bab3.html

[5] http://id.wikipedia.org/wiki/Kriptografi

[6] http://yurindra.wordpress.com/about/hash -function-integrity-checking/

[7] http://en.wikipedia.org/wiki/SHA_hash_functions

[8] http://id.wikipedia.org/wiki/MD5