STUDI ALGORITMA ADLER, CRC, FLETCHER DAN IMPLEMENTASI PADA MAC

Andi Setiawan – NIM : 13506080

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16080@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang algoritma checksum yang terdiri dari algoritma Adler, CRC, dan Fletcher. Checksum adalah adalah data yang biasanya berbentuk string yang panjangnya tetap, yang digunakan untuk mengecek apakah file yang diperiksa tidak berubah. Pada makalah ini akan dibahas tentang bagaimana pembuatan checksum dari masing-masing algoritm. Selain studi algoritma, makalah ini juga berisi implementasi algoritma pada Message Authentication Code (MAC).

MAC adalah suatu cara untuk menguji keabsahan suatu pesan tanpa merahasiakan pesannya. Dengan mekanisme ini, pesan tersebut dapat diketahui apakah ia masih asli dan tidak diubah atau pesan tersebut sudah diubah sebelumnya.

***Kata kunci**: Adler, Fletcher, CRC, MAC, checksum, hashsum*

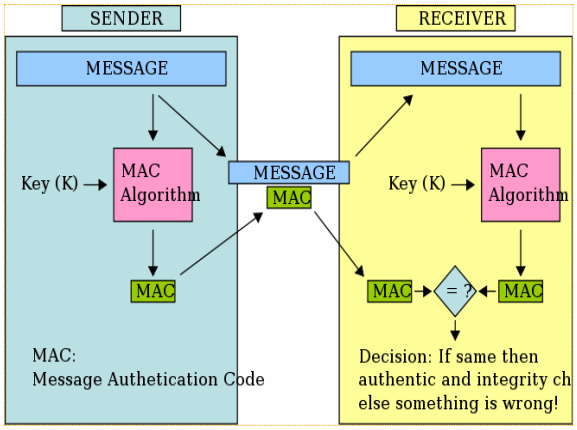
1. Pendahuluan

Message Authentication Code (MAC) adalah adalah suatu kode yang dibangkitkan dari pesan dengan menggunakan kunci tertentu dan fungsi tertentu.. Fungsi tersebut dapat berupa fungsi hash ataupun fungsi enkripsi yang dioperasikan dengan mode CBC. Jika fungsi yang digunakan adalah fungsi hash, maka MAC tersebut dinamakan ***keyed-Hash Message Authentication (KHMAC).*** Pada KHMAC, kunci digunakan oleh penerima pesan untuk memverifikasi nilai *hash*. Mekanisme dari KHMAC secara sederhana adalah, pengguna menggabungkan kunci dengan pesan yang ingin diotentifikasi, lalu menghitung nilai hash-nya dengan menggunakan suatu fungsi hash tertentu.

Selain KHMAC, terdapat juga MAC yang menggunakan algoritma kriptografi simpetri, yaitu algortima blok aliran yang berkaitan (Cipher Block Chaining), yang disebut CMAC (Cipher Message Authentication Code). Misalnya CMAC yang menggunakan algoritma enkripsi DES, maka kuncinya adalah kunci DES yang panjangnya 54 bit, dan panjang dari bloknya adalah 64 bit. MAC-nya adalah hasil enkripsi dari blok terakhir dengan menggunakan sistem CBC.

.

MAC digunakan untuk otentikasi pesan tanpa perlu merahasiakan isi pesannya. Gambar dibawah menunjukkan mekanisme dari MAC., Pihak pengirim akan membangkitkan MAC menggunakan algoritma MAC dengan kunci yang dirahasiakan.Kunci tersebtu diasumsikan telah dikirimkan melalui jalur yang aman, sehingga tidak diketahui orang lain selain pihak pengirim dan penerima. Setelah MAC ini didapatkan, lalu MAC ini dilekatkan pada pesan, dan dikirim ke penerima. Pihak penerima mendapatkan pesan beserta MAC-nya. Setelah mendapatkan kirimannya, pengirim akan menghitung MAC dari pesan menggunakan algoritma MAC yang sama, dan kunci yang sama, lalu membandingkan hasilnya dengan MAC yang dikirim oleh pengirim. Jika hasilnya sama, maka kiriman tersebut masih asli, belum berubah dan kiriman tersebut dikirimkan oleh pengirim sesungguhnya. Jika hasilnya berbeda, maka kemungkinan besar pesan tersebut telah berubah selama proses pengiriman atau pesan tersebut tidak dikirim oleh pengirim yang kita maksudkan. Perbubahan pesan ini bisa terjadi mungkin karena pesan tersebut sengaja diubah oleh pihak lain yang tidak berwenang, ataupun karena terjadi kerusakan pada saat proses pengiriman.



Gambar 1 MAC

Aspek keamanan MAC

Ada beberapa hal yang perlu diperhatikan dalam aspek keamanan dari MAC, antara lain:

a. Keamanan kunci

Keamanan MAC sangat bergantung pada kerahasiaan kunci. Oleh karena itu, baik pengirim maupun penerima harus benar-benar menjaga kunci tersebut dari tindakan yang kurang bertanggung jawab.

b. Fungsi *hash*

Keamanan MAC bergantung pada fungsi *hash* atau teknik enkripsi yang digunakan.

c. Pemalsuan

Keamanan dari MAC berarti keamanannya dari usaha pemalsuan. MAC dikatakan gagal jika seorang penyusup yang tidak memiliki kunci K, bisa menemukan beberapa pesan bersama dengan nilai MAC-nya. Penyusup tersebut diasumsikan dapat mengumpulkan sejumlah contoh dari teks dan nilai MAC -nya yang valid dengan melakukan observasi terhadap jalur aliran data antara pengirim dan penerima.

Beberapa Serangan pada MAC

a. *Known-text attack*

Merupakan salah satu jenis serangan yang mungkin terjadi pada MAC dimana penyerang dapat menentukan pola *MAC*.

b. Chosen-text attack

Merupakan salah satu jenis serangan yang mungkin terjadi pada MAC dimana penyerang dapat menentukan pola *MAC* dari pasangan *(m,*Ck*(m))* yang dipilihnya sendiri.

c. *Adaptive chosen-text attack*

Merupakan salah satu jenis serangan yang mungkin terjadi pada MAC dimana penyerang dapat menentukan pola *MAC* dari pasangan *(m,*Ck*(m))* yang mengarah ke penemuan kunci.

d. *Brute Force attack*

Serangan ini adalah serangan yang sangat lazim digunakan pada kriptografi. Tipikal dari serangan *brute force* adalah mencoba semua kemungkinan yang ada untuk mendapatkan nilai tertentu. Serangan ini akan mencoba semua kemungkinan pesan yang benar untuk sebuah nilai *hash* yang diketahui.

2. Checksum/Hashsum

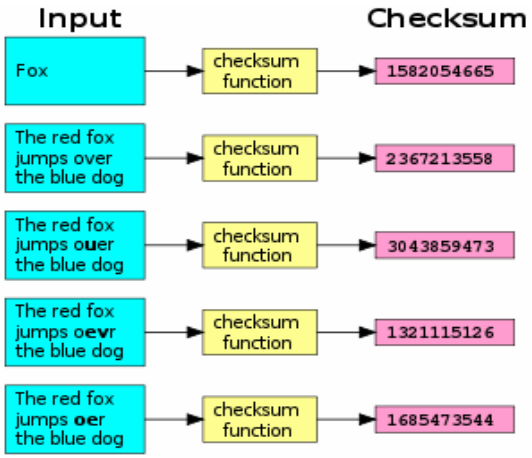
Checksum digunakan untuk memastikan integritas data untuk transmisi data atau penyimpanan. Checksum dasarnya merupakan perhitungan ringkasan dari data.

Jaringan pengiriman data sering terjadi kesalahan, bisa saja data yang dikirimkan hilang atau bit-nya terduplikasi. Akibatnya, data yang diterima mungkin tidak akan sama dengan data yang dikirim, dan jelas hal ini adalah hal yang buruk.

Akibat kesalahan transmisi, protokol jaringan sering menggunakan checksum untuk mendeteksi kesalahan seperti itu. Pemancar yang akan menghitung checksum dari data dan mengirimkan data bersama-sama dengan checksum. Penerima yang akan menghitung checksum dari data yang diterima dengan algoritma yang sama sebagai transmitter. Jika dihitung checksum yang diterima dan tidak sesuai dengan checksum pesan yang dihitung, maka kemungkinan telah terjadi kesalahan transmisi.

Menggunakan checksum secara drastis mengurangi jumlah transmisi diketahui kesalahan. Namun, algoritma checksum yang biasa tidak dapat menjamin deteksi error dari 100%, sehingga san-

gat kecil jumlah kesalahan transmisi dapat tetap terdeteksi.



Gambar 2 Mekanisme Checksum

Seperti yang dijelaskan pada gambar di atas, fungsi checksum akan selalu menghasilkan checksum dengan panjang yang tetap dan cukup identik satu sama lain. Bisa dibilang bahwa, bila pesan yang dimasukan berbeda, maka checksumnya juga akan berbeda.

Ada beberapa macam algoritma checksum; contoh yang sering digunakan adalah algoritma checksum CRC32. Algoritma Checksum lainnya adalah adler dan flethcer yang akan dibahas pada makalah ini. Ketiga algoritma tersebut termasuk pada Position-dependent checksum.

Meskipun algoritma checksum ini lebih banyak digunakan pada pengiriman paket di jaringan internet. Namun penulis melihat bahwa algoritma ini dapat dapat diimplementasikan sebagai algoritma MAC. Karena fungsi perhitungan checksum ini bisa disebut sebagai fungsi hash karena selalu menghasilkan data yang reltif unik dan panjangnya tetap. Implementasi dari algoritma tersebut akan dibahas pada bagian selanjutnya.

3. Algoritma Checksum

3.1 Adler

Algoritma ini ditemukan oleh Mark Adler. Dibandingkan dengan algoritma Checksum lain, Adler memberikan kecepatan perhitungan yang lebih cepat. Adler juga lebih aman dibandingkan algoritma Fletcher. Pada makalah ini, penulis membahas tentang perhitungan Adler-32. Perhitungan awal dari algoritma adler-32 didapat dari dua bilangan yang panjangnya16-bit sebut saja bilangan tersebut A dan B. Bilangan A dan B ini kemudian akan dikonkatenasi menjadi bilangan integer 32 bit. A adalah jumlah dari semua byte di string ditambah dengan satu, sedangkan B adalah jumlah dari setiap nilai-nilai dari A dari setiap langkah. Pada permulaan algoritma Adler-32, nilai A diinisialisasi dengan nilai 1, sedangkan B diinisialisasi dengan nilai 0. Hasil dari penjumlahan kemudian akan dimodulo dengan 65521, yang merupakan bilangan prima terbesar yang lebih kecil dari 216.

Fungsi Adler-32 ini dapat diekspresikan sebagai berikut :

$$A = 1 + D1 + D2 + ... + Dn \pmod{65521}$$
$$B = (1 + D1) + (1 + D1 + D2) + ... + (1 + D1 + D2 + ... + Dn) \pmod{65521}$$
$$= n \times D1 + (n-1) \times D2 + (n-2) \times D3 + ... + Dn + n \pmod{65521}$$

$$\text{Adler-32}(D) = B \times 65536 + A$$

Dimana D adalah string byte masukan, dan n adalah panjang dari D.
Contoh algoritma Adler-32 ini pada saat menghitung checksum dari string : ”coba”

ascii	A	B
c = 99	1+99 = 100	0+100 =100
o = 111	100+111 = 211	100+211 = 311
b = 98	211+98 = 309	311+309 = 620
a = 97	309+97 = 406	628+406 = 1134

Maka nilai A dan B dalam basis 16 adalah :
A = 406 = 196 (hex)
B = 1134 = 46E (hex)

Dan nilai checksum-nya adalah A konkat B = 046E0196

3.2. Fletcher

Algoritma ini ditemukan oleh JG Flethcer lebih dahulu dibandingkan algoritma adler. Sebenarnya kedua algoritma adalah sama, hanya ada satu perbedaan yaitu pada jumlah modulo-nya. Pada adler-32, hasil perhitungan dimodulo 65521 yang didapat dengan mencari bilangan prima yang lebih kecil dari 2^{16}. Sedangkan pada algoritma Fletcher, hasil penjumlahan pada A dan B dimodulo 65535 atau 2^{16}-1.

Sehingga rumus algoritma Fletcher menjadi:

$$A = 1 + D1 + D2 + ... + Dn \pmod{65535}$$

$$B = (1 + D1) + (1 + D1 + D2) + ... + (1 + D1 + D2 + ... + Dn) \pmod{65535}$$
$$= n \times D1 + (n-1) \times D2 + (n-2) \times D3 + ... + Dn + n \pmod{65535}$$

$$\text{Fletcher-32}(D) = B \times 65536 + A$$

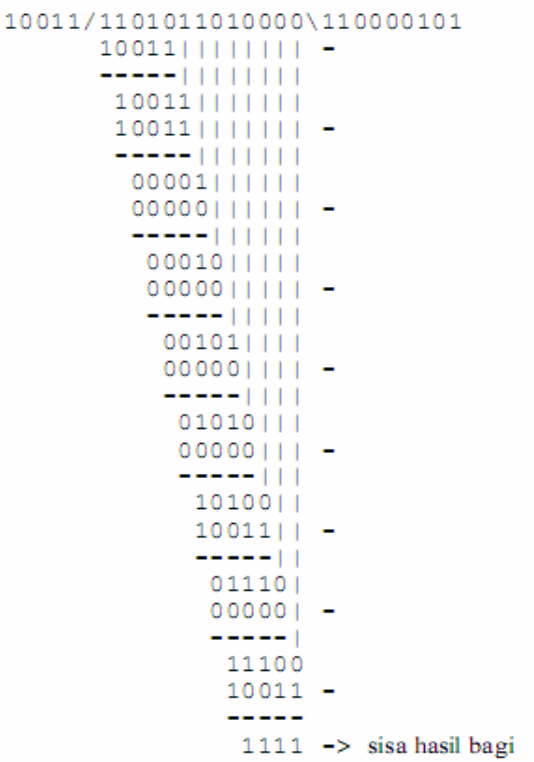
Menurut beberapa literatur, perbedaan modulo ini cukup berpengaruh dalam menghasilkan hasil checksum yang baik. Dalam hal ini, modulo bilangan prima memungkinkan algoritma Adler menangkap perbedaan pada beberapa perubahan bit yang kecil. Selain itu perbedaan algoritma adler dan fletcher juga terletak pada kecepatan kalkulasinya. Beberapa sumber mengatakan bahwa algoritma Adler dapat dikomputasi dengan lebih cepat dibandingkan algoritma Fletcher.

3.3 CRC (Cyclic Redudancy Check)

CRC-32 algoritma Cyclic Redundancy Check yang menghasilkan checksum sebesar 32 bit. (CRC). CRC adalah salah satu fungsi hash yang masih kurang aman, yang berguna untuk menghitung checksum pada saat pengiriman data melalui jaringan. CRC cukup terkenal dan juga mudah diterapkan pada perangkat keras. Prinsip utama yang digunakan adalah dengan melakukan pembagian polinomial dengan mengabaikan bit-bit carry.

Untuk menghitung CRC, file direpresentasikan menjadi serangkaian bit yang besar, kemudian bit-bit tersebut dioperasikan sebagai suatu bilangan polynomial yang sangat besar. CRC didapatkan dengan membagi bilangan polinomial tersebut dengan sebuah divisor/ pembagi,

Setiap operasi pembagian pasti akan menghasilkan suatu sisa hasil bagi (meskipun ada kemungkinan bernilai 0), tetapi ada perbedaan dalam melakukan pembagian pada penghitungan CRC ini. Pada aljabar yang dipelajari di sekolah, pembagian dapat dilakukan dengan mengurangi suatu bilangan dengan kelipatan pembagi secara terus-menerus sampai menghasilkan suatu sisa hasil bagi (yang lebih kecil dari bilangan pembagi). Dari nilai hasil bagi, sisa hasil bagi, dan bilangan pembagi kita bisa mendapat bilangan yang dibagi dengan mengalikan bilangan pembagi dengan hasil bagi dan menambah dengan sisa hasil bagi. Dalam penghitungan CRC, operasi pengurangan dan penjumlahan dilakukan dengan melakukan operasi XOR pada bit-bit, jika operasi tersebut ekivalen dengan operasi pengurangan pada aljabar biasa. Perhitungan CRC juga mengabaikan bit carry setelah bit tersebut melewati suatu operasi. Agar lebih jelas, dapat dilihat pada gambar di bawah, berikut ini adalah contoh perhitungan checksum pada CRC:



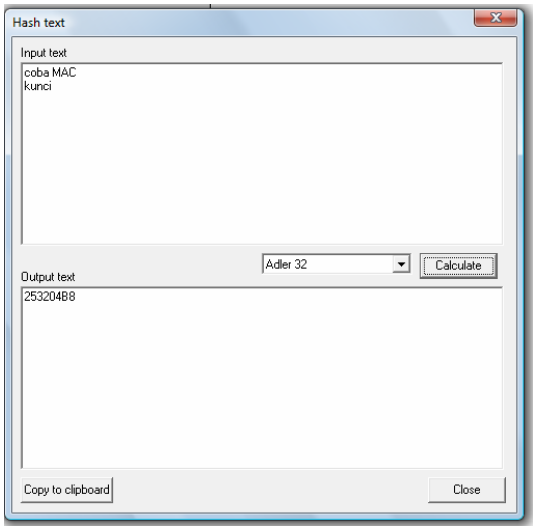
Gambar 3 Menghitung Checksum pada CRC

Gambar diatas menjelaskan tentang cara mencari sisa hasil bagi. Bit yang mengalami operasi pengurangan akan di-XOR=kan. Sisa hasil bagi ini adalah checksum yang dihasilkan pada algoritma CRC, namun pada algoritma CRC-32 sisa hasil bagi yang diharapkan adalah 32 bit. Hal yang sangat mempengaruhi panjang dari sisa hasil bagi ini adalah bit-bit pembagi. Bit pembagi harus lebih panjang satu bit dari bit sisa hasil bagi dan pada most significant bit nilainya satu. Hal ini akan memastikan panjang bit sisa hasil bagi. Misalkan pada contoh diatas panjang bit pembagi adalah lima bit, maka panjang bit sisa hasil bagi adalah empat bit. Pada contoh di atas, penulis hanya memberikan ilustrasi cara pencarian nilai CRC dengan cara aljabar biasa. Masih ada metode lain untuk mencari nilai CRC

yaitu dengan pendekatan tabel CRC yang tidak dibahas pada makalah ini.

4. MAC Menggunakan Adler-32

Pada makalah ini penulis mencoba mengimplementasikan MAC dengan algoritma Checksum adler-32. Untuk membantu membuat MAC, penulis menggunakan perangkat lunak EasyHash v1.1. Sebuah pesan string "coba MAC" ingin dikirimkan oleh pengirim. Pengirim dan penerima telah mengetahui sebelumnya bahwa kunci rahasia yang mereka gunakan adalah "kunci". Adler-32 kemudian menghitung nilai checksum dari gabungan dan kunci dan mengeluarkan string berbasis 16 yaitu 253204B8.

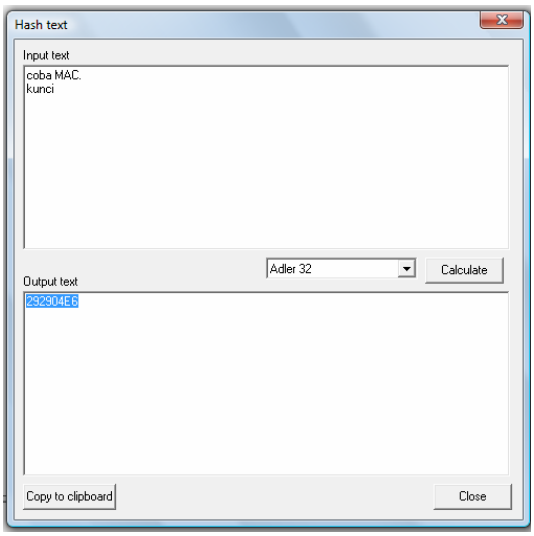


Gambar 4 Penghitugan Adler

Checksum dan pesan ini kemudian digabungkan dan dikirim ke penerima.

cobaMAC
253204B8

Penerima menerima pesan "coba MAC". Ia masih belum mengetahui keaslian dari pesan ini. Maka ia mencoba menghitung checksum dengan menggabungkan string tersebut dengan string "kunci". Dan mendapatkan hasilnya sama yaitu 253204B8. Maka terbutki pesan tersebut otentik dari pengirim yang ia maksudkan. Apabila terjadi perubahan sedikit saja pada pesan berupa penambahan tanda titik "coba MAC." Nilai checksum menjadi 292904E6 dan berbeda dari checksum yang dikirim oleh pengirim, maka pesan tersebut tidaklah otentik.



Gambar 5 Perhitungan Adler jika ada perubahan

5. Kesimpulan

Algoritma checksum dapat digunakan untuk membangkitkan message authentication code, namun keamanan algoritma masih kurang baik disbanding fungsi hash lain.

DAFTAR PUSTAKA

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

[2]Wikipedia. *http://en.wikipedia.org/.* Diakses 20 Mei 2009.

[3]Software EasyHash. *http://www.download3k.com/DownloadLink1-Easy-Hash.htm.* Diakses 20 Mei 2008.

[4]Message Authentication Code. http://www.ittelkom.ac.id/library/index.php?view=article&catid=20%3Ainformatika&id=522%3Amessage-authentication-code-mac&option=com_content&Itemid=15. Tanggal akses: 20 Mei 2009

[5]Ditto Narapratama. Perbandingan Performansi Algoritma Adler-32 dan CRC-32 pada Library ZLib. 2006. Institut Teknologi Bandung.

[6]Wijayanto, Indra Sakti. Penggunaan CRC32 dalam Integritas Data. 2006. Institut Teknologi Bandung