Implementasi dan Analisis Perbandingan Algoritma MAC Berbasis Fungsi Hash Satu Arah Dengan Algoritma MAC Berbasis *Cipher Block*

Pudy Prima – 13508047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18047@students.if.itb.ac.id

***Abstract*—*Message Authentication Code* (MAC) adalah suatu teknik yang dapat digunakan untuk melakukan otentikasi pengirim pesan serta menjamin integritas data. MAC membutuhkan kunci rahasia yang disepakati antara pihak pengirim dan penerima pesan. MAC dapat diimplementasi menggunakan dua pendekatan umum, yaitu algoritma MAC berbasis fungsi hash satu arah dan algoritma MAC berbasis *cipher block*. Pada penelitian ini, algoritma MAC berbasis fungsi hash satu-arah diimplementasi menggunakan fungsi SHA-1, sedangkan algoritma MAC berbasis *cipher block* diimplementasi menggunakan mode operasi *Electronic Code Book* (ECB) dengan algoritma *cipher block* sederhana yang tidak memanfaatkan jaringan feistel. Aspek yang dibandingkan pada penelitian ini adalah kecepatan dalam pembangkitan nilai MAC.**

Index Terms*—MAC, fungsi hash, *cipher block

I. PENDAHULUAN

Dalam kriptografi, *Message Authentication Code* (MAC) dapat digunakan untuk menjaga integritas pesan serta untuk melakukan otentikasi pengirim pesan. MAC dapat diimplementasi menggunakan dua pendekatan, yaitu dengan algoritma MAC berbasis fungsi hash satu arah serta dengan algoritma MAC berbasis *cipher block*. Setiap pendekatan tersebut memiliki kelebihan serta kekurangan masing-masing.

Aspek kecepatan kadang dirasa cukup penting untuk mengukur efektifitas serta efisiensi suatu algoritma, walaupun aspek keamanan masih tetap paling dipentingkan.

Pada penelitian kali ini, setiap pendekatan algoritma MAC ini akan diimplementasi. Algoritma MAC berbasis fungsi hash satu-arah akan diimplementasi menggunakan fungsi SHA-1 (*Secure Hash Algorithm*), sedangkan algoritma MAC berbasis *cipher block* akan diimplementasi dengan mode operasi ECB (*Electronic Code Book*) denga algoritma blok *cipher* sederhana yang memanfaatkan fungsi *exclusive or* (XOR).

Aspek yang akan dibandingkan antara kedua pendekatan tersebut adalah kecepatan dalam pembangkitan nilai MAC.

II. DASAR TEORI

A. *Message Authentication Code* (MAC)

Dalam kriptografi, *Message Authentication Code* (MAC) merupakan informasi yang digunakan untuk menjaga integritas pesan serta melakukan otentikasi pengirim pesan. Dalam algoritma MAC, masukan yang diterima adalah pesan yang akan diautentikasi serta kunci rahasia. Adapun keluarannya adalah nilai MAC (kadang disebut tag).

Seperti tanda tangan digital, MAC dilekatkan (embed) pada pesan, dan digunakan untuk otentikasi tanpa perlu merahasiakan pesan. Namun demikian, MAC berbeda dari tanda tangan digital. Pada MAC, nilai dihasilkan dengan menggunakan kunci rahasia yang sama antara pengirim dan penerima pesan, atau sama dengan penggunaan enkripsi simetri, sedangkan tanda tangan digital dihasilkan dengan menggunakan kunci privat dari pasangan kunci rahasia (kunci privat dan kunci publik), atau sama dengan enkripsi asimetri.

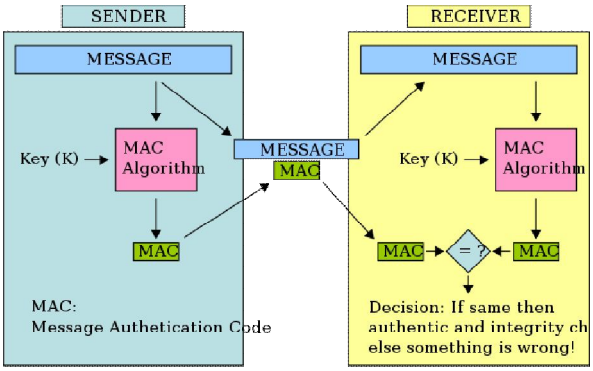
Secara matematis, MAC dinyatakan sebagai

MAC = CK(M)

dengan,

MAC = nilai hash
C = fungsi hash (atau algoritma MAC)
K = kunci rahasia
M = pesan yang akan dihitung nilai MACnya

Prosedur penggunaan MAC dalam pengiriman pesan adalah seperti ditunjukkan pada gambar di bawah ini. Pertama-tama, pengirim pesan akan menghitung nilai MAC dari pesan yang akan dikirim berdasarkan kunci rahasia tertentu. Setelah itu, pesan akan dikirim beserta nilai MAC yang telah dihasilkan tadi. Pihak penerima pesan, menggunakan kunci rahasia yang sama dengan kunci rahasia yang dimiliki pihak pengirim pesan, akan menghitung nilai MAC dari pesan yang diterima. Jika nilai MAC yang dihitung oleh penerima pesan sama dengan nilai MAC yang dilekatkan pada pesan, maka dapat disimpulkan bahwa pesan yang diterima tersebut masih terjaga keasliannya.



Gambar 1 Prosedur *Message Authentication Code*

Algoritma MAC dapat dibangun menggunakan dua pendekatan, yaitu algoritma MAC berbasis fungsi hash satu arah dan algoritma MAC berbasis *cipher block*.

B. Fungsi Hash

Fungsi hash adalah fungsi yang mentransformasikan suatu masukan string dengan panjang sembarang menjadi keluaran berupa string yang panjangnya tetap (fixed) dan umumnya berukuran jauh lebih kecil daripada ukuran string semula. Nilai hash suatu string sering disebut sebagai nilai *message digest*. Fungsi hash biasanya digunakan untuk melakukan verifikasi terhadap keaslian suatu arsip atau dokumen.

Secara umum, persamaan fungsi hash adalah sebagai berikut :

$$\mathbf{h = H(M)}$$

dengan,

M = pesan ukuran sembarang

h = nilai hash (*message digest*)

Fungsi hash satu-arah (*One-Way Hash*) adalah fungsi hash yang hanya bisa menghasilkan nilai *message digest* dari suatu pesan, namun tidak dapat mengembalikan nilai *message digest* menjadi pesan awal. Dengan kata lain, fungsi hash ini bekerja satu arah, enkripsi saja tanpa dekripsi.

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 $y \neq x$ sedemikian sehingga *H*(*y*) = *H*(*x*).
6. Tidak mungkin mencari pasangan *x* dan *y* sedemikian sehingga *H*(*x*) = *H*(*y*).

Fungsi hash satu-arah yang umum digunakan adalah *Secure Hash Algorithm* (SHA). Algoritma SHA menerima masukan berupa pesan dengan ukuran maksimum 2^{64} bit (2.147.483.648 gigabyte) dan menghasilkan *message digest* yang panjangnya 160 bit. SHA-1 adalah salah satu varian SHA yang banyak digunakan. Langkah-langkah pembuatan *message digest* dengan SHA-1 adalah sebagai berikut :

1. Penambahan bit-bit pengganjal (*padding* bits).
   Algoritma SHA-1 melakukan pemrosesan pesan dalam blok-blok berukuran 512 bit. 64 bit dari blok tersebut digunakan untuk menyimpan panjang pesan, sehingga pesan disimpan dalam 448 bit pada blok. Bit-bit pengganjal diperlukan agar panjang pesan pada blok terakhir tetap mencapai 448 bit. Bit-bit pengganjal terdiri dari satu buah bit 1 diikuti dengan sisanya bit 0.
2. Penambahan nilai panjang pesan semula.
   Nilai panjang pesan semula ditambahkan pada 64 bit di setiap blok. Nilai panjang pesan ini digunakan sebagai penanda bit-bit pengganjal.
3. Inisialisasi penyangga (buffer) MD.
   SHA membutuhkan lima buah penyangga (buffer) yang masing-masing panjangnya 32 bit. Kelima penyangga ini akan menampung hasil akhir dan hasil antara, karena SHA terdiri atas 80 putaran (untuk SHA-1). Kelima penyangga tersebut diberi nama A, B, C, D, dan E. Setiap penyangga diinisialisasi dengan nilai sebagai berikut :
   A = 67452301
   B = EFCDAB89
   C = 98BADCFE
   D = 10325476
   E = C3D2E1F0
4. Pengolahan pesan dalam blok berukuran 512 bit.
   Setiap blok pesan yang berukuran 512 bit diproses bersama dengan penyangga MD menjadi keluaran 128-bit. Proses ini disebut proses HSHA dan terdiri dari 80 putaran. Masing-masing putaran menggunakan bilangan penambah Kt, yaitu :
   Putaran $0 \le t \le 19$ K_t = 5A827999
   Putaran $20 \le t \le 39$ K_t = 6ED9EBA1
   Putaran $40 \le t \le 59$ K_t = 8F1BBCDC
   Putaran $60 \le t \le 79$ K_t = CA62C1D6

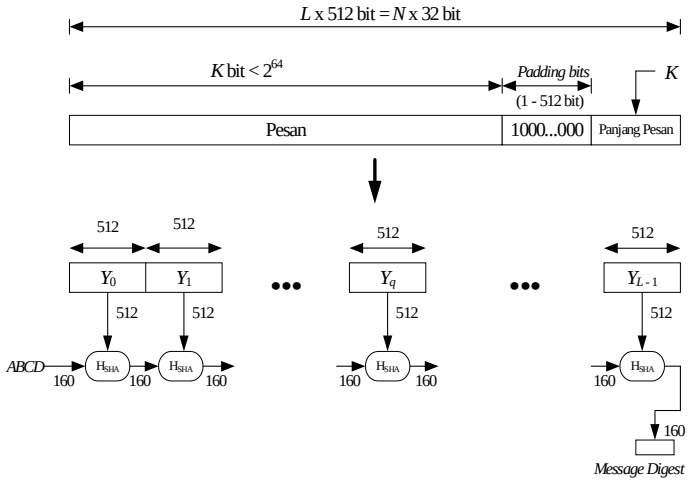
   Fungsi logika ft pada setiap putaran dapat dilihat dalam tabel berikut.

Putaran	$f_t(b, c, d)$
0 .. 19	$(b \wedge c) \vee (\sim b \wedge d)$
20 .. 39	$b \oplus c \oplus d$
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 .. 79	$b \oplus c \oplus d$

   Nilai W_1 sampai W_{16} berasal dari 16 *word* pada blok yang sedang diproses, sedangkan nilai W_t berikutnya didapatkan dari persamaan

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$$

Skema pembuatan *message digest* dengan SHA-1 dapat dilihat pada gambar berikut.

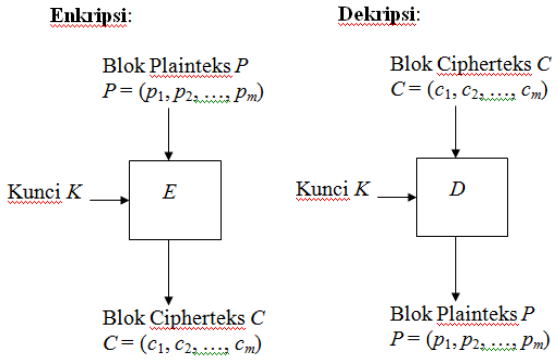


Gambar 2 Skema pembuatan *message digest* menggunakan SHA-1

C. Algoritma *Cipher Block*

Algoritma blok *cipher* melakukan operasi terhadap blok-blok bit yang panjangnya tetap. Algoritma ini tergolong ke dalam algoritma simetri. Blok *cipher* menggunakan kunci yang panjangnya sama dengan panjang blok yang digunakan. Algoritma blok *cipher* terdiri dari sepasang algoritma, yaitu algoritma untuk melakukan proses enkripsi (E) dan algoritma untuk melakukan proses dekripsi (D). Pada algoritma ini, bit-bit plainteks akan dibagi menjadi blok-blok bit dengan panjang sama, misalnya 64 bit. Enkripsi dilakukan terhadap blok bit plainteks menggunakan bit-bit kunci.

Skema enkripsi dekripsi algoritma blok *cipher* ditunjukkan dari gambar berikut.



Gambar 3 Skema enkripsi-dekripsi blok *cipher*

Pemrosesan algoritma blok *cipher* dapat dilakukan dalam berbagai mode operasi, yaitu *Electronic Code Book* (ECB), *Cipher Block* Chaining (CBC), *Cipher* Feedback (CFB), serta Output Feedback (OFB). Setiap mode operasi memiliki karakteristik tersendiri untuk menghindari serangan-serangan. Pemilihan mode yang akan digunakan mempengaruhi pemrosesan algoritma blok *cipher* ini.

Electronic Code Book (ECB) adalah mode operasi bplo *cipher* yang paling sederhana. Pada mode ini, plainteks akan dibagi menjadi beberapa blok berukuran sama, dan setiap blok plainteks Pi dienkripsi secara individual dan independen menjadi blok *cipher*teks Ci. Mode ECB memiliki keuntungan sebagai berikut :

1. Karena setiap blok plainteks dienkripsi secara independen, maka kita tidak perlu mengenkripsi file secara linear (enkripsi dapat dilakukan secara acak untuk setiap bloknya).
2. Kesalahan satu atau lebih bit pada blok *cipher*teks hanya mempengaruhi *cipher*teks yang bersangkutan pada waktu dekripsi.

Adapun kelemahan mode operasi ECB adalah sebagai berikut :

1. Karena bagian plainteks yang sering berulang (sehingga terdapat blok-blok plainteks yang sama), maka hasil enkripsinya menghasilkan blok *cipher*teks yang sama.
2. Pihak lawan dapat memanipulasi *cipher*teks untuk mengelabui penerima pesan.

III. IMPLEMENTASI

A. MAC Berbasis Fungsi Hash Satu Arah

Dalam kriptografi, HMAC (Hash-based *Message Authentication Code*) merupakan bentuk perhitungan nilai MAC yang melibatkan fungsi hash dan dikombinasikan dengan suatu kunci rahasia. Dalam penelitian kali ini, fungsi hash yang digunakan adalah SHA-1, karena dirasa cukup umum dan mudah dalam pengimplementasiannya.

Nilai hash biasanya tidak dibangkitkan menggunakan kunci rahasia. Namun pada *Message Authentication Code* berbasis fungsi hash satu-arah, nilai hash dibangkitkan dengan menggunakan suatu kunci rahasia tertentu. Proses pembangkitan nilai hash dilakukan dengan cara yang sama dengan proses pada fungsi hash SHA-1, hanya saja blok-blok yang dihitung nilai hashnya harus di*padding* terlebih dahulu dengan bit-bit kunci.

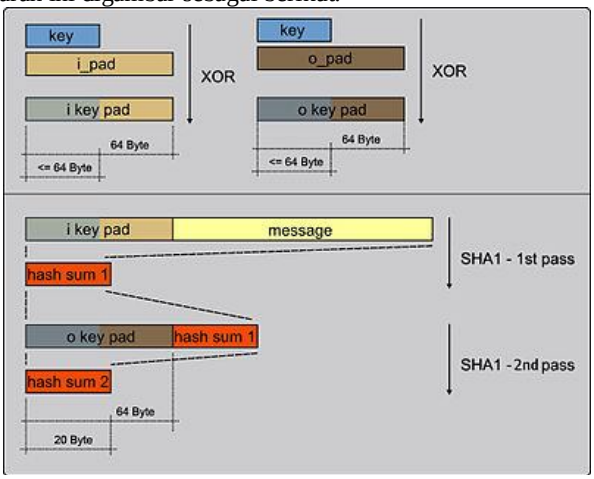
Secara umum, fungsi hash yang digunakan dalam algoritma MAC berbasis fungsi hash satu arah ini adalah sebagai berikut :

$$\mathbf{HMAC}(K,m) = \mathbf{H}((K \oplus \text{opad}) \parallel \mathbf{H}((K \oplus \text{ipad}) \parallel \text{m}))$$

- H : fungsi hash
- *K* : kunci rahasia
  Kunci ini ditambahkan ke sebelah kanan blok (jika panjang kunci lebih pendek dari panjang blok) atau dihitung nilai hashnya terlebih dahulu sebelum kemudian ditambahkan ke bagian kanan blok (jika panjang kunci lebih panjang dari panjang blok).
- *m* : pesan yang akan dibuat nilai MACnya
- ‖ : tanda konkatenasi
- ⊕ : tanda *exclusive or* (XOR)
- opad : outer *padding*
  0x5c5c5c…5c5c, konstanta heksadesimal sepanjang satu blok
- ipad : inner *padding*

0x363636…3636, konstanta heksadesimal sepanjang satu blok

Skema algoritma MAC berbasis fungsi hash satu-arah ini digambar sebagai berikut.



Gambar 4 Skema algoritma HMAC

Pseudocode dari fungsi ini adalah sebagai berikut.

```
function hmac (key, message)
    if (length(key) > blocksize) then
        key = hash(key)
        //  keys  longer  than  blocksize  are
shortened
    end if
    if (length(key) < blocksize) then
        key  =  key  ||  [0x00  *  (blocksize  -
length(key))]
        // keys shorter than blocksize are zero-
padded ('||' is concatenation)
    end if

    o_key_pad = [0x5c * blocksize] ⊕ key
    // Where blocksize is that of the underlying
hash function
    i_key_pad = [0x36 * blocksize] ⊕ key
    // Where ⊕ is exclusive or (XOR)

    return  hash(o_key_pad  ||  hash(i_key_pad  ||
message))
    // Where '||' is concatenation
end function
```

Hasil implementasi algoritmanya adalah sebagai berikut.

```
// pembangkit MAC dengan fungsi hash
public static byte[] hmac(byte[] message, byte[] key)
{
    byte[] macVal = new byte[160];
    byte[] K0 = new byte[key.Length];
    byte[] temp = new byte[key.Length];
    byte[] temp2 = new byte[key.Length];
    byte[] o_key_pad = new byte[blocksize];
    byte[] i_key_pad = new byte[blocksize];

    if (key.Length > blocksize)
    {
        key = getMessageDigest(key);
    }

    for (int i = 0; i < i_key_pad.Length; i++)
    {
        if (i % 2 == 0)
        {
            i_key_pad[i] = 6;
        }
        else
        {
           i_key_pad[i] = 3;
        }
    }

    for (int i = 0; i < o_key_pad.Length; i++)
    {
        if (i % 2 == 0)
        {
            o_key_pad[i] = (byte)('c');
        }
        else
        {
            o_key_pad[i] = 5;
        }
    }

    temp = XOR(key, i_key_pad);
    temp = concat(temp, message);
    temp = getMessageDigest(temp);

    temp2 = XOR(K0, o_key_pad);
    macVal = concat(temp, temp2);
    macVal = getMessageDigest(macVal);
    return macVal;
}
```

B. MAC Berbasis *Cipher Block*

Pada penelitian kali ini, mode operasi blok *cipher* yang digunakan adalah mode *Electronic Code Book* (ECB). Mode ini dipilih karena implementasinya dirasa paling sederhana. Algoritma blok *cipher* yang digunakan adalah algoritma sederhana, yaitu dengan melakukan XOR antara plainteks dengan kunci.

Langkah-langkah yang dilakukan dalam membuat MAC dengan algoritma blok *cipher* adalah sebagai berikut :

1. Hitung panjang blok kunci, lalu pecah isi file pesan menjadi sejumlah blok dengan panjang setiap blok sama dengan panjang blok kunci. Jika panjang blok terakhir kurang dari panjang kunci, maka lakukan *padding* terhadap blok tersebut.
   Padding yang dilakukan adalah dengan menambahkan sejumlah bit 0 di akhir blok hingga panjang blok tersebut menyamai panjang blok kunci.

```
// Fungsi Padding
public int[] PaddingKey(int[] key)
        {
            int[] temp = new int[64];
            for (int i = 0; i < key.Length; i++)
            {
                temp[i] = key[i];
            }
            for (int i = key.Length; i < 64; i++)
            {
                temp[i] = 0;
            }
            return temp;
        }
```

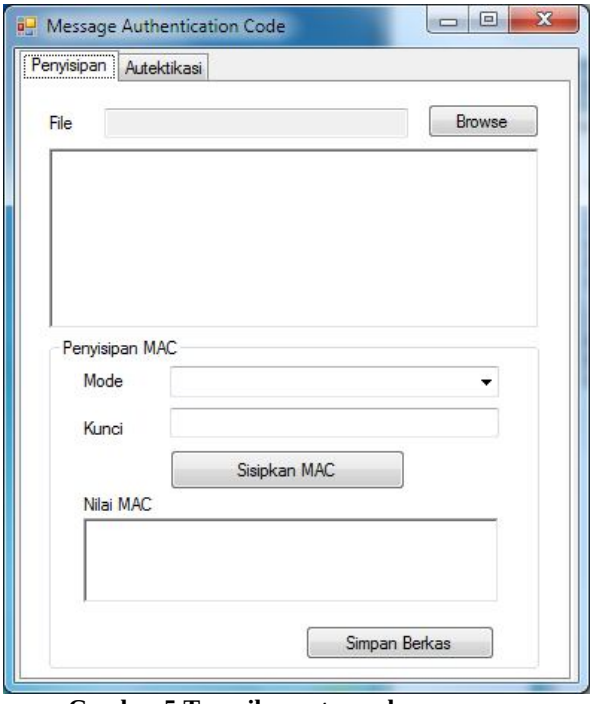
2. Untuk setiap blok pesan, operasikan algoritma sederhana blok *cipher*.

```
//Algoritma block cipher
```

```
        public BitArray cipherblockAlgo(BitArray
plain, BitArray key)
        {
            BitArray cipher = new BitArray(key.Length);
            int[] KeyAll = ConBitToInt(key);
            int[] plainAll = ConBitToInt(plain);
            int[] cipherAll = XOR(plainAll, KeyAll);
            for (int i = 0; i < key.Length; i++)
            {
                if (cipherAll[i] == 1)
                {
                    cipher.Set(i, true);
                }
            }
            return cipher;
        }
```

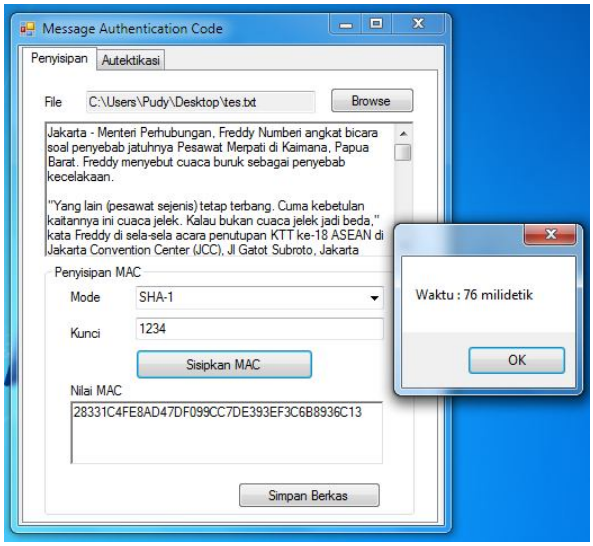
3. Gabungkan seluruh blok *cipher* hasil enkripsi, lalu ubah menjadi string heksadesimal. Inilah yang merupakan nilai MAC dari file pesan tadi.
4. Sisipkan nilai MAC di bagian akhir file pesan.

IV. PENGUJIAN



Gambar 5 Tampilan antarmuka program

Pengujian penyisipan pesan ini dilakukan terhadap berbagai pesan dengan berbagai ukuran. Pengujian ini dilakukan untuk membandingkan kecepatan pembuatan nilai MAC antara algoritma MAC berbasis fungsi hash satu-arah dengan algoritma MAC berbasis *cipher block*. Pada program yang digunakan, telah ditambahkan fitur penghitung waktu sehingga dapat diketahui waktu yang dibutuhkan untuk membuat nilai MAC dari suatu pesan.

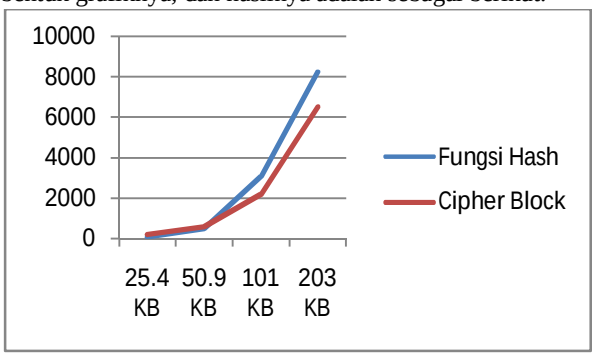


Gambar 6 Tampilan pembangkitan nilai MAC

Hasil pengujiannya dapat dilihat dari tabel berikut.

Ukuran File	Fungsi hash	*Cipher Block*
25.4 KB	76 ms	189 ms
50.9 KB	491 ms	577 ms
101 KB	3078 ms	2187 ms
203 KB	8244 ms	6521 ms

Data pada tabel tersebut kemudia dibuat menjadi bentuk grafiknya, dan hasilnya adalah sebagai berikut.



Gambar 7 Grafik data hasil pengujian

V. ANALISIS

Berdasarkan data yang diperoleh dari penelitian, kita dapat membandingkan penggunaan algoritma MAC berbasis fungsi hash satu-arah dengan algoritma MAC berbasis *cipher block* dengan berpedoman pada kecepatan pembuatan nilai MAC menggunakan setiap algoritma tersebut. Dari grafik yang telah dibuat, terlihat bahwa waktu yang diperlukan untuk membuat nilai MAC dari suatu pesan dengan menggunakan pendekatan fungsi hash cenderung lebih lama, dan menjadi semakin lama ketika file pesan yang digunakan berukuran lebih besar.

Dari data pengujian, ketika file yang digunakan berukuran cukup kecil (kurang dari 100 KB), ternyata waktu pembuatan nilai MAC antara pendekatan *cipher*

block dengan pendekatan fungsi hash tidak terlalu jauh berbeda. Ini berarti ukuran file pesan mempengaruhi kecepatan pembuatan nilai MAC.

Untuk data selanjutnya, yaitu dengan ukuran file pesan yang lebih besar dari 100 KB, pendekatan fungsi hash memakan waktu yang jauh lebih lama dibandingkan penggunaan pendekatan *cipher block*. Dengan demikian, dapat diketahui bahwa algoritma yang digunakan berpengaruh terhadap kecepatan pembuatan nilai MAC.

Saat menggunakan pendekatan fungsi hash, terdapat 80 putaran dengan proses yang berbeda-beda di setiap 20 putaran. Jumlah putaran yang cukup banyak inilah yang menyebabkan waktu pembuatan nilai MAC dengan algoritma MAC berbasis fungsi hash satu-arah menjadi relatif lebih lama. Untuk pembuatan nilai MAC terhadap file yang berukuran relatif besar, pendekatan fungsi hash satu-arah ini akan memakan waktu yang cukup lama, sehingga dirasa kurang efektif. Namun jika dilihat dari nilai *message digest* yang dihasilkan, pendekatan ini cukup efektif karena nilai *message digest* yang dihasilkan selalu memiliki panjang yang sama.

Adapun pada pembuatan nilai MAC dengan pendekatan *cipher block*, proses pada algoritma ini memang jauh lebih sederhana, yaitu dengan menghitung nilai XOR antara blok pesan dengan blok kunci, serta tidak menggunakan jaringan feistel yang cukup rumit. Algoritma blok *cipher* yang sederhana inilah yang membuat pembuatan nilai MAC menjadi relatif lebih cepat. Untuk target waktu yang tidak terlalu lama, pendekatan ini dapat digunakan. Namun demikian, algoritma ini dirasa terlalu sederhana, dan kemungkinan akan lebih mudah terbongkar polanya, sehingga mungkin ditemukan nilai data yang sama dengan hasil pembuatan nilai MAC menggunakan pendekatan *cipher block* ini.

VI. KESIMPULAN

- Implementasi pembangkitan nilai MAC dengan algoritma MAC berbasis fungsi hash satu-arah dan algoritma MAC berbasis *cipher block* berhasil dilakukan.
- Aspek yang dibandingkan terhadap penggunaan kedua pendekatan algoritma tersebut adalah kecepatan pembangkitan nilai MACnya.
- Berdasarkan data hasil pengujian, waktu yang digunakan untuk membangkitkan nilai MAC dengan algoritma MAC berbasis fungsi hash satu-arah relatif lebih lama dibandingkan dengan penggunaan algoritma MAC berbasis *cipher block*.
- Untuk file pesan yang relatif kecil (kurang dari 100 KB), pemilihan pendekatan algoritma yang digunakan tidaklah terlalu berarti, karena waktu yang digunakan tidak berbeda jauh.
- Untuk file pesan yang relatif besar (lebih dari 100 KB), pemilihan pendekatan algoritma cukup signifikan. Jika yang diinginkan adalah kecepatan pembangkitan nilai MAC, maka pendekatan *cipher block* dapat digunakan (dengan asumsi algoritma blok *cipher* yang digunakan relatif sederhana dan tidak mengandung jaringan feistel), sedangkan jika menginginkan keamanan pesan, maka lebih cocok digunakan pendekatan fungsi hash, karena nilai MAC yang dihasilkan berukuran sama panjang untuk setiap pesan.

REFERENSI

http://en.wikipedia.org/wiki/Message_authentication_code, diakses pada 26 April 2011

http://en.wikipedia.org/wiki/HMAC, diakses pada 26 April 2011

http://upe.acm.jhu.edu/member_sites/zarfoss/HMAC.html#HMAC, diakses pada 26 April 2011

http://en.wikipedia.org/wiki/SHA-1, diakses pada 26 April 2011

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, 9 Mei 2011

Pudy Prima
13508047