ANALISIS DAN IMPLEMENTASI VMPC (VARIABLY MODIFIED PERMUTATION COMPOSITION) STREAM CIPHER SEBAGAI MAC (MESSAGE AUTHENTICATION CODE)

A. Thoriq Abrowi Bastari - 13508025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18025@students.if.itb.ac.id

***Abstrak*— Makalah ini akan membahas penggunaan *stream cipher* sebagai MAC (Message Authentication Code). *Stream cipher* (*cipher* aliran) adalah salah satu jenis algoritma *cipher* berbasis bit. Pada makalah ini, *stream cipher* yang digunakan adalah VMPC (Variably Modified Permutation Composition), sebuah teknologi enkripsi yang didesain oleh Bartosz Zoltak dan dipublikasikan pada tahun 2004. VMPC Stream Cipher cukup efisien untuk diimplementasikan pada perangkat lunak dan diklaim menawarkan keamanan yang lebih baik dibandingkan RC4, yang masih populer hingga saat ini, baik dalam proses enkripsi maupun dari algoritma penjadwalan kunci.**

MAC (Message Authentication Code) digunakan untuk menjaga keaslian dari suatu pesan/arsip terhadap perubahan yang terjadi akibat virus atau perbuatan sengaja dari orang lain. Nilai MAC di-*generate* dengan menggunakan algoritma enkripsi tertentu, salah satunya adalah dengan *stream cipher*. Untuk menganalisis dan mengetahui lebih dalam penggunaan *stream cipher* dalam *message authentication* ini, tentunya akan dilakukan pengujian terhadap algoritma tersebut. Pengujian akan dilakukan dengan menggunakan program tertentu. Terdapat beberapa parameter yang nantinya menjadi dasar penilaian. Beberapa di antaranya adalah performansi (ditentukan dari waktu yang diperlukan untuk membangkitkan MAC) dan keamanan (resistansi terhadap serangan *message-forgery*).

Tujuan dilakukannya studi analisis dan pengujian ini adalah untuk mengetahui karakteristik, terutama kelebihan dan kekurangan, dari penggunaan *stream cipher*, khususnya VMPC, sebagai algoritma enkripsi untuk menghasilkan MAC. Dengan mengetahui hal tersebut diharapkan baik penulis maupun pembaca makalah ini dapat mengenal dengan lebih baik konsep dari MAC dan macam-macam penggunaannya.

Kata kunci*—VMPC, MAC, *stream cipher

I. PENDAHULUAN

Dengan semakin berkembangnya teknologi informasi pada saat ini, informasi sudah bukan barang mahal lagi. Informasi dapat diperoleh dari mana saja, terutama melalui internet. Hampir sebagian besar informasi yang diperlukan dapat diperoleh melalui internet dengan menggunakan berbagai jenis perangkat yang mempermudah pencarian informasi, seperti Google, Yahoo!, dan lain-lain.

Kemudahan dalam memperoleh informasi ini tentunya ditanggapi dengan sangat positif oleh banyak orang. Namun, timbul masalah-masalah baru yang mengiring. Salah satunya adalah isu keamanan dalam berinteraksi, atau keamanan informasi.

Seseorang yang ingin mengirimkan atau menyimpan informasi yang bersifat *confidential* menjadi ragu tentang keamanan informasinya miliknya tersebut. Pada era teknologi yang sudah maju seperti sekarang ini, hal tersebut dapat dimaklumi. Tidak ada yang dapat menjamin bahwa suatu informasi penting yang dikirimkan dari suatu komputer, telepon genggam, *smart phone*, atau pun alat-alat elektronik lainnya tidak dapat dilihat oleh orang lain yang tidak mempunyai hak. Tidak ada juga yang dapat menjamin seratus persen kalau informasi yang diperoleh atau diberikan oleh seseorang tidak melalui proses *forgery* atau pemalsuan.

Isu tersebut lah yang menyebabkan ilmu kriptografi semakin berkembang luas dan umum dalam masyarakat. Kriptografi adalah keilmuan dan praktek dalam menyembunyikan informasi. Sebenarnya kriptografi sendiri bukan ilmu yang baru muncul pada era modern ini. Sudah dari jauh hari orang-orang berusaha menjaga keamanan informasi yang *confidential*. Namun, seperti yang telah disebutkan sebelumnya, kemudahan dalam memperoleh informasi pada era modern ini lah yang menyebabkan kriptografi semakin populer, diperlukan, dan banyak digunakan.

Ilmu kriptografi sudah banyak berkembang dari yang awalnya hanya berkutat dalam enkripsi dan dekripsi karakter kini menjadi enkripsi dan dekripsi dalam basis bit atau byte. Ini lah yang memungkinkan teknologi kriptografi pada zaman ini untuk melakukan enkripsi dan dekripsi pada dokumen atau *file* elektronis. Kemampuan ini digunakan untuk banyak hal dalam bidang keamanan informasi, salah satunya adalah untuk menjaga keaslian pesan.

Untuk menjaga keaslian dari suatu pesan atau dokumen, salah satu teknik yang umum digunakan adalah Message Authentication Code atau lebih dikenal dengan MAC saja. Fungsi dari MAC sendiri tidak jauh berbeda dibandingkan dengan *hash* dan *digital signature*.

Mengenai perbedaan mendasar dari ketiga teknik tersebut akan dibahas lebih lanjut pada bab berikutnya.

Terdapat banyak jenis algoritma enkripsi yang digunakan dalam MAC untuk menghasilkan nilai MAC itu sendiri. Beberapa yang umum digunakan adalah fungsi *hash* dan *block cipher*. Namun, pada makalah ini penulis akan mencoba membahas dan menganalisis penggunaan fungsi *stream cipher* dalam menghasilkan nilai MAC. Salah satu *stream cipher* yang dapat digunakan dalam MAC adalah VMPC (Variably Modified Permutation Composition) Stream Cipher. Algoritma *stream cipher* ini diklaim sangat efisien untuk diimplementasikan pada perangkat lunak.

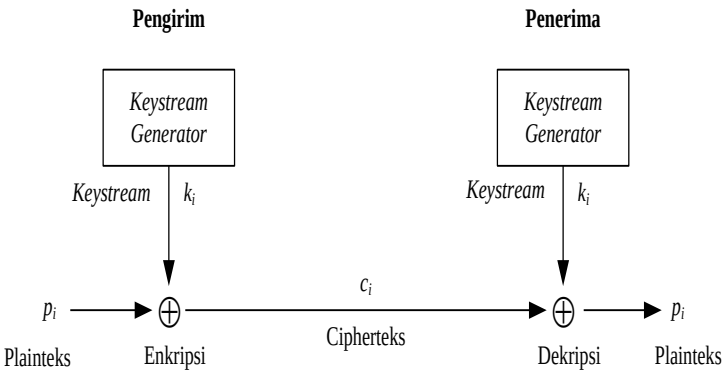
II. TEORI DASAR

Pada bab ini akan dijelaskan secara singkat beberapa teori-teori yang diperlukan dalam penyelesaian makalah ini. Terdapat empat sub bab mengenai teori dasar dalam VMPC-MAC (Message Authentication Code), yaitu *stream cipher*, MAC, VMPC Stream Cipher (algoritma enkripsi yang digunakan dalam membuat VMPC-MAC), dan juga VMPC-MAC itu sendiri.

A. Stream Cipher

Stream cipher adalah salah satu jenis algoritma kriptografi modern yang mengenkripsi plainteks menjadi cipherteks bit per bit atau byte per byte. *Stream cipher* atau *cipher* aliran adalah sebuah *symmetric key cipher* yang menggunakan *keystream* untuk kemudian digabungkan dengan *plaintext* untuk menghasilkan sebuah *ciphertext*. Operasi yang umum digunakan pada *stream cipher* adalah operasi XOR. Algoritma *stream cipher* pertama kali diperkenalkan oleh Vernam lewat algoritma yang diciptakannya, yaitu Vernam Cipher.

Skema pada *cipher* aliran pada umumnya adalah sebagai berikut:



Gambar 1. Skema *stream cipher*

Dari skema di atas dapat diperoleh informasi bahwa keamanan *stream cipher* sepenuhnya bergantung pada *keystream* yang digunakan. *Keystream* pada *stream cipher* dihasilkan oleh *keystream generator*. Oleh karena itu, *keystream generator* merupakan elemen yang paling penting dalam *cipher* aliran.

Beberapa kelebihan dari *stream cipher* di antaranya adalah kecepatannya eksekusinya oleh *hardware* dan *software*, sederhana untuk dibuat, dan kemampuannya untuk mengenkripsi plainteks dengan panjang yang tidak diketahui sebelumnya.

B. MAC (Message Authentication Code)

Sebelum membahas definisi dari MAC, terlebih dahulu akan diberikan penjelasan mengenai perbedaan antara MAC dengan *hash* dan *digital signature*. Ketiganya memiliki fungsi dan kegunaan yang relatif sama, yaitu menjaga keaslian data. Perbedaan utama dari ketiganya dapat lebih mudah dimengerti dari contoh proses aktualnya.

Misalkan ada dua pasang manusia yang ingin bertukar pesan melalui jaringan internet, sebut saja Bob dan Alice. Bob ingin mengirimkan suatu pesan pada Alice. Alice yang menerima pesan tersebut ingin membuktikan keaslian pesan tersebut karena tidak ingin pesan yang dia terima ternyata berbeda dari pesan yang dikirimkan oleh Bob entah karena data tersebut rusak atau sengaja diubah oleh orang lain. Lalu bagaimana cara Alice membuktikan itu?

Cara pertama yang dapat dilakukan adalah dengan *hash*. Selain mengirimkan pesan kepada Alice, Bob juga membuat sebuah *hash code* dari pesan tersebut dan mengirimkannya kepada Alice bersama dengan pesan yang ingin dikirim. Alice yang mendapatkan pesan tersebut beserta *hash code*-nya juga membuat *hash code* sendiri dari pesan yang diterimanya. *Hash code* tersebut lalu dibandingkan dengan *hash code* yang diperoleh dari Bob. Apabila diperoleh kecocokan maka bisa disimpulkan bahwa pesan tersebut asli dan tidak mengalami perubahan sejak dikirim oleh Bob.

Namun, proses pengecekan dengan *hash* sangat rentan terhadap serangan *man-in-the-middle*. Seseorang bisa saja melakukan *intercept* terhadap pesan tersebut beserta *hash code*-nya lalu mengganti keduanya dengan sesuatu yang berbeda. Solusi dari permasalahan ini lah yang menjadi tujuan dibuatnya MAC (Message Authentication Code).

Pada contoh kejadian yang sama, selain membuat *hash code* dari pesan yang akan dikirimkan, Bob juga mengenkripsi *hash code* tersebut dengan suatu kunci simetris. Alice yang menerima pesan tersebut beserta *hash code* yang telah dienkripsi juga melakukan hal yang sama, yaitu membangkitkan *hash code* dari pesan yang diterima lalu mengenkripsi *hash code* tersebut menggunakan kunci simetris yang sama. Pengecekan dilakukan dengan membandingkan kedua hasil enkripsi dari *hash code* tersebut.

MAC dapat membuktikan kalau suatu pesan adalah asli dan tidak mengalami perubahan sejak dikirimkan namun tidak bisa menjamin pesan tersebut adalah pesan yang dikirimkan oleh orang tertentu. Hal ini lah yang

mendasari lahirnya *digital signature*.

Lagi-lagi dengan contoh yang sama, Bob juga mengenkripsi *hash code* dari pesan yang akan dikirimkannya dengan suatu kunci privat. Hasilnya adalah sebuah *digital signature* yang dikirimkan kepada Alice bersama dengan pesannya. Alice yang ingin memeriksa keaslian pesan dan keaslian pengirimnya harus memiliki kunci publik dari Bob. Sama seperti langkah sebelumnya, Alice membandingkan *digital signature* tersebut lalu dapat menentukan keaslian dan identitas pengirim pesan tersebut.

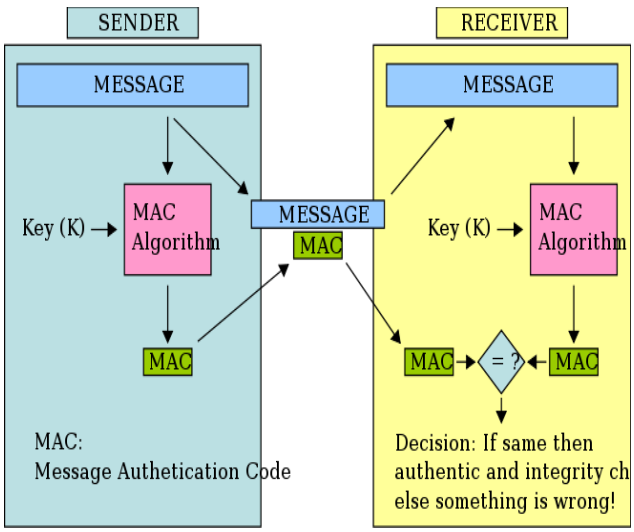
Dari contoh-contoh proses tersebut, bisa dibilang perbedaan mendasar antara ketiganya hanyalah dari ada tidaknya proses enkripsi *hash code* dan juga perbedaan jenis kunci yang digunakan. MAC menggunakan kunci simetris sementara *digital signature* menggunakan kunci asimetris, yaitu kunci publik dan privat. MAC dapat dilekatkan langsung di dalam pesan atau di luar pesan (dalam *file* yang berbeda).

MAC secara matematis:

$$MAC = C_K(M)$$

MAC = nilai *hash*
C = fungsi *hash* (atau algoritma MAC)
K = kunci rahasia

Berikut adalah skema umum dari MAC (Message Authentication Code):



Gambar 3. Skema umum MAC

C. VMPC (Variably Modified Permutation Composition) Stream Cipher

VMPC Stream Cipher adalah perluasan dari VMPC *one-way function* yang dikembangkan menjadi sebuah algoritma enkripsi berbasis byte.

VMPC *one-way function* adalah transformasi dari permutasi n-elemen yang didefinisikan sebagai berikut:

for x from 0 to n-1:
g(x) = VMPC(f(x)) = f(f(f(x))+1)

Operasi penambahan pada *pseudo-code* di atas adalah operasi penambahan dengan modulo n. Pada penggunaannya pada VMPC Stream Cipher dihasilkan sebuah 8-bit *stream* dari permutasi 256 elemen. *Initial state* dari permutasi tersebut dikalkulasi dalam VMPC Key Scheduling Algorithm.

Serupa dengan skema *stream cipher* pada umumnya, VMPC Stream Cipher membangkitkan *keystream* dengan sebuah *keystream generator* lalu dilakukan algoritma enkripsi dengan *key* tersebut dan plainteks.

Skema dari VMPC Key Scheduling Algorithma kurang lebih seperti ini:

1. Set s to 0
2. Set i-th element of P to i for i ∈ {0,1,...,255}

3. Set m to 0

4. Add modulo 256 (m modulo 256)-th element of P to s
5. Add modulo 256 (m modulo c)-th element of K to s
6. Set s to s-th element of P
7. Swap (m modulo 256)-th element of P with s-th element of P
8. Increment m

9. Go to step 4 if m is lower than 768

10. If Initialization Vector is not used: terminate the algorithm

11. Set m to 0

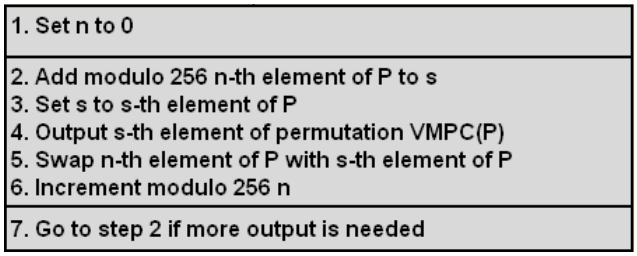
12. Add modulo 256 (m modulo 256)-th element of P to s
13. Add modulo 256 (m modulo z)-th element of V to s
14. Set s to s-th element of P
15. Swap (m modulo 256)-th element of P with s-th element of P
16. Increment m

17. Go to step 12 if m is lower than 768

Gambar 4. Skema umum VMPC Key Scheduling Algorithm

K : *Kunci rahasia*
C : *Panjang kunci rahasia*
V : *Initialization vector*
Z : *Panjang initialization vector, z* ∈ *{16...64}*
M : *Variabel 16-bit*

Sementara, skema dari VMPC Stream Cipher sendiri adalah sebagai berikut:



Gambar 5. Skema umum VMPC Stream Cipher

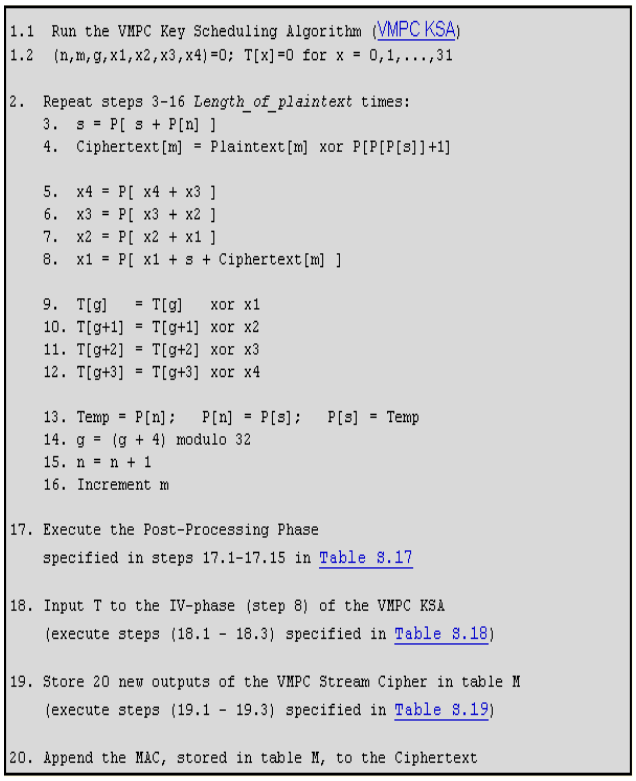
P : *Tabel permutasi 256-byte*
S : *Variabel 8-bit dari key scheduler*
n : *Variabel 8 bit*

VMPC Stream Cipher dijalankan setelah VMPC Key Scheduling Algorithm dijalankan terlebih dahulu. Perlu diperhatikan bahwa *initialization vector* bersifat opsional. Namun, dengan adanya *initialization vector*, permutasi yang dihasilkan akan semakin acak.

D. VMPC-MAC

VMPC-MAC adalah salah satu algoritma untuk menghitung MAC (Message Authentication Code). VMPC-MAC menggunakan beberapa *state* internal yang terdapat pada VMPC Stream Cipher, seperti tabel permutasi P. Hal ini membantu menyederhanakan desain dan meminimalisasi komputasi yang dilakukan sehingga menghasilkan performansi yang tinggi dalam implementasinya pada perangkat lunak.

Berikut adalah skema yang dipakai dalam VMPC-MAC:



Gambar 6. Skema VMPC-MAC

P : *Tabel 256-byte dari tabel permutasi, dihasilkan pada VMPC Key Scheduling Algorithm*
S : *Variabel 8-bit, dihasilkan pada VMPC Key Scheduling Algorithm*
T : *Tabel 32-byte*
x1, x2 ,x3 ,x4 : *Variabel 8-bit*
M : *Tabel MAC berukuran 20-byte*
n, m, g, R : *Variabel integer sementara*
+ : *Penambahan dengan modulo 256*

Dari skema VMPC-MAC di atas dapat dilihat bahwa VMPC-MAC menggunakan VMPC Stream Cipher dan VMPC Key Scheduling Algorithm yang digabungkan dengan algoritma lain untuk menambahkan efek difusi pada hasil MAC yang dikeluarkan.

III. PERANCANGAN VMPC-MAC

Untuk lebih memahami VMPC-MAC ini, penulis mengimplementasikan MAC yang menggunakan VMPC dalam bahasa pemrograman Java. Berikut adalah penjelasan umum mengenai skema dari program dan pengujian-pengujian yang dilakukan menggunakan program ini.

A. Penjelasan Umum Program

Terdapat dua fungsi utama dalam program yang telah dirancang. Fungsi pertama adalah fungsi keyScheduler yang menjalankan VMPC Key Scheduling Algorithm dan fungsi yang kedua adalah encryptMessage yang melakukan enkripsi dan meng hasilkan nilai MAC dari pesan yang menjadi masukannya. Panjang kunci dan *initial vector* ditetapkan menjadi 16-byte. Sementara ukuran MAC juga ditentukan sebesar 20-byte. Berikut adalah *pseudo-code* dari program uji yang dibuat:

Pseudo-code keyScheduler:

```
s = 0
for i from 0 to 255:
  P[i]=i

for m from 0 to 767:
  n = m and 255
  s = P[(s + P[n]+ K[m mod c])mod 256]
  Temp = P[n]
  P[n] = P[s]
  P[s] = Temp

If Initialization Vector is used:
```

```
for m from 0 to 767:
  n = m and 255
  s = P[(s + P[n]+ V[m mod z])mod 255]
  Temp = P[n]
  P[n] = P[s]
  P[s] = Temp
```

Pseudo-code encryptMessage:

```
(n,m,g,x1,x2,x3,x4)=0;

for x from 0 to 31:
  T[x] = 0

for m from 0 to plaintext.Length-1:
  s = P[s + P[n]]
  Ciphertext[m] = Plaintext[m] xor
P[P[P[s]]+1]

  x4 = P[x4 + x3]
  x3 = P[x3 + x2]
  x2 = P[x2 + x1]
  x1 = P[x1 + s + Ciphertext[m]]

  T[g]   = T[g]   xor x1
  T[g+1] = T[g+1] xor x2
  T[g+2] = T[g+2] xor x3
  T[g+3] = T[g+3] xor x4

  Temp = P[n];   P[n] = P[s]
  P[s] = Temp
  g = (g + 4) mod 32
  n = n + 1

for R from 1 to 24:
  s = P[s + P[n]]

  x4 = P[x4 + x3 + R]
  x3 = P[x3 + x2 + R]
  x2 = P[x2 + x1 + R]
  x1 = P[x1 + s + R]

  T[g]   = T[g]   xor x1
  T[g+1] = T[g+1] xor x2
  T[g+2] = T[g+2] xor x3
  T[g+3] = T[g+3] xor x4

  Temp = P[n];   P[n] = P[s]
  P[s] = Temp
  g = (g + 4) mod 32
  n = n + 1

for m from 0 to:
  n = m mod 256
  s = P[s + P[n] + T[m mod 32]]
  Temp = P[n];   P[n] = P[s]
  P[s] = Temp

for n from 0 to:
  s = P[s + P[n]]
  M[n] = P[P[P[s]]+1]
  Temp = P[n];   P[n] = P[s]
  P[s] = Temp
```

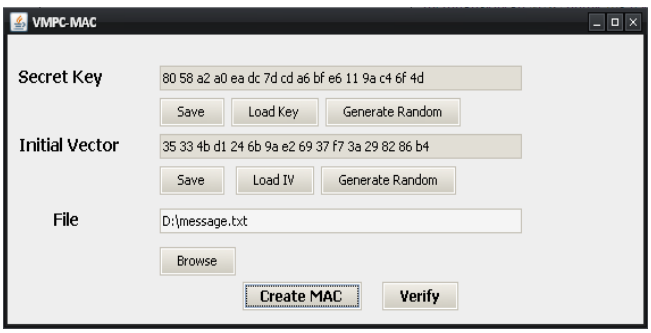
Fungsi keyScheduler selalu dijalankan terlebih dahulu sebelum fungsi encryptMessage. Nilai MAC yang dihasilkan lalu akan disimpan dalam *file* terpisah. Pembangkitan MAC dan verifikasi pesan mengharuskan pengguna untuk terlebih dahulu memasukkan kunci rahasia dan *initial vector* (opsional).

Hasil dari MAC disimpan dalam bentuk *array of byte* saja sehingga tidak bisa dibaca dengan jelas. Kunci rahasia dan *initial vector* juga dapat disimpan dalam *file*, juga dalam byte. Namun, pada antarmuka, kunci dan *initial vector* ditampilkan dalam bentuk hexadecimal. Pengguna juga memiliki pilihan untuk membangkitkan kunci rahasia dan *initial vector* secara acak.

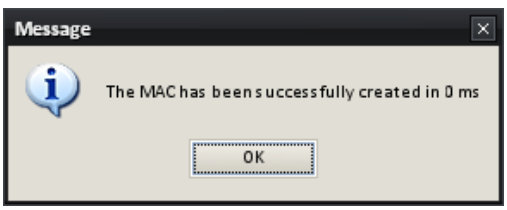
Dalam melakukan verifikasi pesan/*file*, program akan otomatis mencari *file* dengan nama yang sama namun dengan ekstensi .mac pada direktori *file* tersebut untuk mendapatkan nilai MAC untuk dijadikan pembanding. Setelah itu melakukan proses yang hampir sama dengan membangkitkan MAC lalu membandingkannya. Beberapa kemungkinan penyebab verifikasi gagal antara lain adalah *file* yang berubah, kunci rahasia yang salah, *initial vector* yang salah, dan nilai MAC yang berubah. Setelah selesai dibandingkan, hasil verifikasi akan ditampilkan kepada pengguna.

B. Pengujian Program

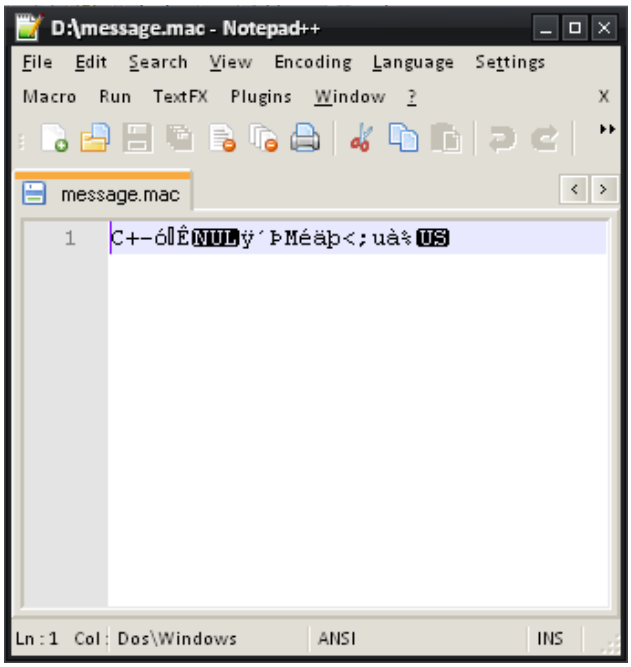
Pengujian pertama dilakukan dengan mencoba membangkitkan MAC untuk *file* teks biasa.



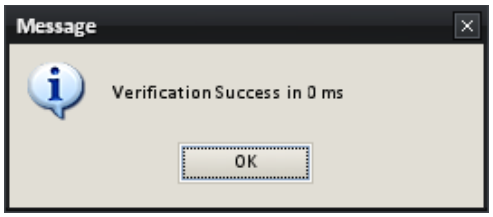
Gambar 7. Screenshot aplikasi saat akan membangkitkan MAC



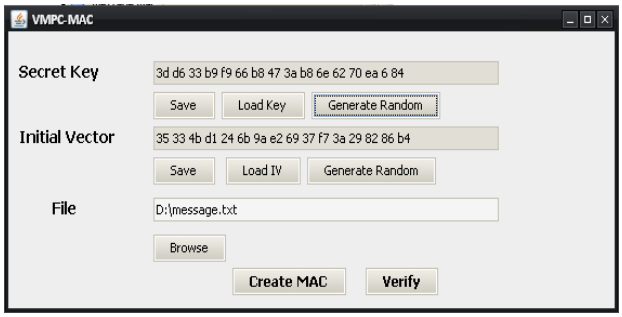
Gambar 8. Pesan keberhasilan membangkitkan nilai MAC



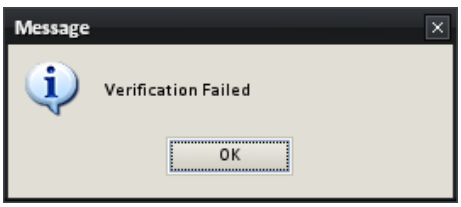
Gambar 9. MAC dalam bentuk *file*



Gambar 10. Pesan keberhasilan melakukan verifikasi dengan kunci dan IV dari hasil yang sebelumnya

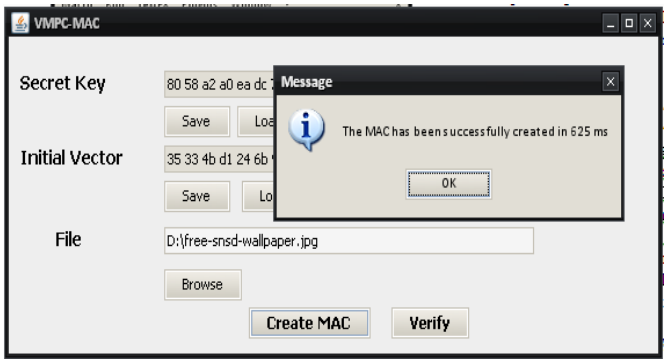


Gambar 11. *Screenshot* aplikasi saat akan melakukan verifikasi dengan kunci yang berbeda

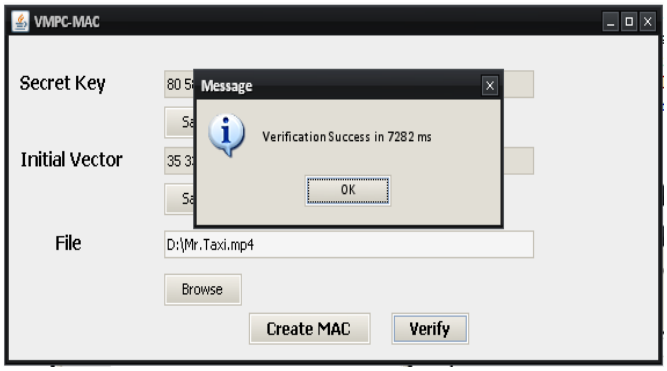


Gambar 12. Pesan kegagalan yang muncul

Selanjutnya dilakukan pengujian dengan berbagai variasi ukuran *file*. Berikut adalah tabel dan *screenshot* beberapa pengujian lainnya:



Gambar 13. Pembangkitan MAC untuk *file* jpg berukuran 1,67 MB



Gambar 14. Verifikasi MAC *file* mp4 berukuran 19,0 MB

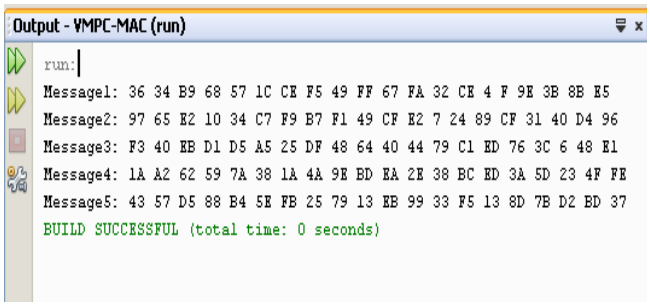
Nama File	Ukuran (KB)	Waktu (ms)
free-snsd-wallpaper	1718	625
aimp_2.61.583.zip	7789	2734
Mr. Taxi.mp4	19525	7282
Alvaro Recoba – Magic Left Foot.flv	28758	10563
xampp-tomcat-addon-win32-6.0.20.exe	30349	12531

Tabel 1. Hasil percobaan untuk beberapa *file* (berdasarkan ukuran *file*)

Untuk pengujian berikutnya, akan coba dibandingkan hasil MAC dari beberapa *file* yang berbeda namun dengan isi yang mirip. Pengujian ini dilakukan untuk membuktikan efek difusi yang dihasilkan. Kunci dan *initial vector* yang digunakan adalah hasil dari pembangkitan kunci acak. Semua bahan uji menggunakan kunci dan *initial vector* yang sama.

Isi Pesan	MAC
diegomiliti	36 34 B9 68 57 1C CE F5 49 FF 67 FA 32 CE 4 F 9E 3B 8B E5
diegomilito	97 65 E2 10 34 C7 F9 B7 F1 49 CF E2 7 24 89 CF 31 40 D4 96
diegomilitu	F3 40 EB D1 D5 A5 25 DF 48 64 40 44 79 C1 **ED** 76 3C 6 48 E1
diegomilita	1A A2 62 59 7A 38 1A 4A 9E BD EA 2E 38 BC **ED** 3A 5D 23 4F FE
diegomiliti	43 57 D5 88 B4 5E FB 25 79 13 EB 99 33 F5 13 8D 7B D2 BD 37

Tabel 2. Hasil percobaan untuk beberapa *file* (berdasarkan isi pesan)



Gambar 15. Screenshot hasil pengujian 5 *file* yang berbeda

IV. ANALISIS

Pada pengujian yang pertama, dengan ukuran *file* yang hanya 14 bytes, pembangkitan MAC tidak sampai bulat 1 ms. Waktu eksekusi dalam melakukan pembangkitan MAC dan verifikasi memang hampir sama karena proses yang dilakukan juga serupa.

Dari hasil percobaan yang telah dilakukan, waktu eksekusi memang berbanding lurus dengan ukuran *file* yang akan dibangkitkan nilai MAC-nya. Waktu yang dibutuhkan untuk membangkitkan nilai MAC atau verifikasi *file* relatif cepat. Bahkan jauh lebih cepat dibandingkan dengan waktu untuk membaca *file* yang akan digunakan.

VMPC *one-way function*, inti dari semua yang berhubungan dengan VMPC, termasuk VMPC-MAC ini, adalah sebuah fungsi yang mendekati *true one-way function*. Diperlukan sekitar 2^{260} operasi untuk mendapatkan *state* awal dari *state* akhir fungsi tersebut. Hal ini lah yang membuat VMPC-MAC dan VMPC Stream Cipher menawarkan keamanan yang jauh lebih baik dibandingkan kebanyakan fungsi lainnya.

Dari hasil pengujian yang kedua, terbukti bahwa efek difusi yang ditimbulkan oleh VMPC-MAC sudah cukup baik. Untuk 5 pesan yang berukuran sama dan hanya berbeda satu byte, hanya terdapat satu kasus terdapat hasil yang sama untuk indeks keluaran MAC (tabel M), yaitu pada pesan ke-3 dan ke-4 (dapat dilihat pada Tabel 2). Ini berarti algoritma *key scheduler* yang digunakan dan beberapa permutasi tambahan pada VMPC-MAC menghasilkan keluaran yang acak.

Penggunaan 2 fase pada VMPC Key Scheduling Algorithm menghasilkan nilai permutasi yang acak karena setiap fasenya melibatkan 768 tahap substitusi. Tingkat keamanan yang lebih baik bahkan dapat diperoleh dengan menambahkan satu fase lagi pada VMPC KSA setelah fase permutasi dengan *initial vector*. Dengan begitu, dapat meningkatkan *diffusion effect* secara signifikan pada MAC yang dihasilkan.

V. KESIMPULAN

Dari percobaan-percobaan dan studi literatur yang telah dilakukan, penulis dapat mengambil kesimpulan sebagai berikut:

1. *Stream cipher* dapat digunakan sebagai algoritma enkripsi pada MAC.
2. VMPC-MAC memiliki performansi yang cukup baik dan membuktikan bahwa VMPC Stream Cipher efektif dan efisien untuk diterapkan dalam MAC.
3. VMPC-MAC memiliki tingkat keamanan yang tinggi.

Pengembangan dari VMPC-MAC ini bisa dilakukan dengan mengganti VMPC Key Scheduler Algorithm dengan *key scheduler* baru yang menambahkan satu tahap permutasi lagi setelah permutasi dengan *initial vector*.

REFERENSI

[1] Bartosz Zoltak. *VMPC-MAC: A Stream Cipher Based Authenticated Encryption Scheme*. Cryptology ePrint Archive, Report 2004/301, 2004.

[2] Kai-Thorsten Wirt. *ASC – A Stream Cipher with Built–In MAC Functionality*. World Academy of Science, Engineering and Technology 29, 2007.

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

[4]http://en.wikipedia.org/wiki/Variably_Modified_Permutation_Composition

[5]http://dotnetslackers.com/articles/security/Hashing_MACs_and_Digital_Signatures_in_NET.aspx

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, 8 Mei 2011

A Thoriq Abrowi Bastari (13508025)