PERANCANGAN PROGRAM APLIKASI KRIPTOGRAFI MENGGUNAKAN ALGORITMA MARS DENGAN MODUS ECB

Marzuki Silalahi, Tumpal P, Deby
Dosen FASILKOM - UIEU
Dosen Fakultas Teknologi Informatika Universitas Tarumanagara, Jakarta
Mahasiswa Fakultas Teknologi Informatika Universitas Tarumanagara, Jakarta
marzuki_silalahi@plasa.com

Abstrak

Salah satu cara yang digunakan untuk pengamanan data adalah menggunakan sistem kriptografi. Namun sejalan dengan perkembangan ilmu penyandian atau kriptografi, usaha-usaha untuk memperoleh kunci tersebut dapat dilakukan oleh siapa saja, termasuk pihak yang tidak sah untuk memiliki informasi tersebut. Melalui perancangan ini, akan dibahas mengenai kriptografi menggunakan algoritma MARS dengan modus ECB. MARS sebagai salah satu kandidat AES, memiliki kelebihan mempunyai keamanan dan kecepatan yang tinggi. Hal ini menjadikan MARS sebagai pilihan terbaik untuk enkripsi yang diperlukan oleh dunia informasi menuju abad berikutnya. Algoritma MARS menggunakan kunci 128 bit dan proses enkripsinya terdiri dari 32 round. Waktu proses enkripsi pada algoritma ini berjalan pada 85 Mbit per detik. Perancangan program ini juga menyediakan unit sarana pengiriman *file*, baik untuk file yang telah dienkripsi maupun jenis *file* biasa. Namun ruang lingkup pengirimannya terbatas pada jaringan *Local Area Network*. Pengujian telah dilakukan dan program MARS ini dapat dijamin dan dipercaya keamanannya.

Kata Kunci: Kriptografi, MARS, Modus ECB

Pendahuluan

Pengetahuan teknologi informasi dan komputerisasi sangat berkembang saat ini namun kebutuhan akan keamanan serta kerahasiaan data merupakan hal yang sangat penting. Untuk menghindari terjadinya kejahatan komputer, maka diperlukan suatu sekuriti (keamanan) yang baik, sehingga data yang terdapat pada komputer menjadi lebih aman. Salah satu cara yang paling baik adalah dengan menggunakan kriptografi. Terdapat berbagai macam algoritma dalam kriptografi, namun tidak semua mampu memberikan jaminan keamanan dan kerahasiaan yang baik terhadap pesan. Salah satu algoritma yang mampu memberikan jaminan keamanan data adalah algoritma MARS, untuk enkripsi data dalam berbagai aplikasi.

Tujuan dari perancangan ini adalah untuk merancang suatu program aplikasi kriptografi dengan menggunakan algoritma MARS yang dapat memberikan suatu sekuriti

yang lebih baik pada sebuah komputer, dapat menjamin keamanan dan kerahasiaan data, dan memberikan jaminan keamanan pengiriman data dalam jaringan.

Rancangan yang dibuat adalah sebuah program aplikasi komputer yang dapat memberikan keamanan serta kerahasiaan data pada komputer. Program ini memiliki kendali pengamanan data berupa sistem enkripsi dan dekripsi yang menggunakan *symmetric key*. Program ini dirancang dengan menggunakan bahasa pemrograman Visual Basic 6.0

Tinjauan Teori Perancangan

Dalam perancangan ini, dirancang suatu program yang terdiri dari beberapa unit seperti : unit *text editor*, unit kriptografi (unit enkripsi dan dekripsi) dan unit pengiriman *file* dalam jaringan. Spesifikasi dari perancangan program aplikasi kriptografi dengan menggunakan algoritma MARS adalah:

1. Unit *Text Editor*
2. Unit Kriptografi
   a. unit enkripsi
   b. Unit dekripsi
   c. Unit kunci *(key)*
3. Unit Pengiriman *File* di dalam Jaringan

Unit pengiriman *file* digunakan untuk mengirim atau menerima suatu pesan yang telah dienkripsi atau didekripsi, dari satu komputer ke komputer lainnya. Hanya saja pengiriman *file* ini terbatas dalam ruang *lingkup Local Area Network*(LAN).

Untuk mengenkripsi dan mendekripsi data, kriptografi menggunakan suatu algoritma (*cipher*) dan kunci *(key).*

Element Pembangun Algoritma MARS

a. Tipe-3 *Feistal Network*
b. Operasi yang digunakan algoritma MARS
   1. Penjumlahan, pengurangan, perkalian dan XOR.
   2. *Fixed Rotation*
   3. *Data Dependent Rotation*
c. S-Box

Algoritma Enkripsi dan Dekripsi MARS

1. *forward mixing,*
2. *cryptographic core* (transformasi kunci utama),
3. *backward mixing.*

Proses dekripsi MARS dilakukan kebalikan dari proses enkripsinya.

Struktur Cipher Algoritma MARS

Struktur *cipher* pada MARS dibagi dalam 3 tahap yakni :

1. *forward mixing,*
2. *"cryptographic core",*
3. *backward mixing,*

Notasi yang digunakan dalam cipher

1. D[ ] adalah sebuah *array* untuk 4 word 32-bit data. Inisial D berisi *plaintext* dan pada akhir proses enkripsi berisi *ciphertext.*
2. K[ ] adalah *array* untuk *expanded key*, terdiri dari 40 word 32 bit
3. S[ ] adalah sebuah *S-box*, terdiri dari 512 word 32-bit

Perluasan Kunci

Perluasan kunci berfungsi untuk membangkitkan sub kunci dari kunci yang diberikan oleh pemakai

yaitu *array k*[ ] yang terdiri dari n 32-*bit word* (dimana *n* adalah jumlah *words* dari 4 sampai 14) ke dalam *array* K[] sebanyak 40 *words.*

MARS menerima satu blok *key* awal sebesar 128 bit untuk mendapatkan *key* lain yang digunakan dalam proses enkripsi dan dekripsi. Untuk panjang kunci yang kurang dari 128, maka dilakukan *padding bits* yaitu proses penambahan untuk menambahkan satu dan sisanya ditambahkan dengan nol sampai mencapai panjang kunci yang seharusnya.

Key expansion menyediakan 40 *words key* (K[0],.....,K[39]) yang terdiri dari 32-bit. Empat kunci pertama digunakan untuk proses *forward mixing* dan empat kunci terakhir digunakan pada proses *backwards mixing.*

Pembahasan

Hasil Perancangan dan Pembahasan

Proses Pembentukan Kunci

Pada bagian ini, kunci yang dimasukkan pengguna akan diolah menjadi 40 *words* yang akan digunakan pada algoritma MARS. Tahap pertama, kunci awal ini dipisahkan menjadi 4 bagian @ 32bit.

Misalkan kuncinya :

ABCDEFGHIJKLMNOP

k[0]= ABCD (01000001 01000010 01000011 01000100)

k[1]= EFGH (01000101 01000110 01000111 01001000)

k[2]= IJKL (01001001 01001010 01001011 01001100)

k[3]= MNOP (01001101 01001110 01001111 01010000)

Lalu inisialisasi T[] dengan kunci data, dimana T[] adalah *array* sementara yang terdiri dari 15 *words* dan *n* adalah jumlah words di kunci *buffer* k[], ($4 \leq n \leq 14$) Misalkan n = 4

T[0*n* - 1] = k[0..*n* - 1],

T[*n*] = *n*, T[*n* + 1..14]= 0

T[0..3] = k[0..3],

T[4] = 4, T[5..14] = 0//Empat iterasi, menghitung 10 *words* untuk tiap K[]

for *j* = 0 to 3 do

for *i* = 0 to 14 do

//Transformasi Linear *T*[*i*] = ((*T*[*i*-7 mod 15]⊕ T[*i*-2 mod 15])<<< 3) ⊕ (4*i* + *j*) Untuk i = 0 to 14 dan j = 0

T[0] = ((*T*[0-7 mod 15]⊕ T[0-2 mod 15])<<< 3)⊕(4.0 + 0)

= ((T[7] ⊕ T[2])<<< 3) ⊕ 0

= ((0 ⊕ k[2])<<< 3) ⊕ 0

= ((00000000 00000000 00000000 00000000 ⊕ 01001001 01001010 01001011 01001100)<<< 3)⊕ 0

= ((01001001 01001010 01001011 01001100)<<< 3) ⊕ 0

= (10001001 00101001 01001001 01101001) ⊕ 0

= 10001001 00101001 01001001 01101001

T[1] = ((*T*[1-7 mod 15]⊕ T[1-2 mod 15])<<< 3)⊕(4.1 + 0)

= ((T[6] ⊕ T[1])<<< 3) ⊕ 4

= ((0 ⊕ k[1])<<< 3) ⊕ 4

= ((00000000 00000000 00000000 00000000 ⊕ 01000101 01000110 01000111 01001000)<<< 3) ⊕ 4

= ((01000101 01000110 01000111 01001000)<<< 3) ⊕ 4

= (00001000 10101000 11001000 11101001) ⊕ 00000000 00000000 00000000 00000100

= 00001000 10101000 11001000 11101101

T[2] = ((*T*[2-7 mod 15]⊕ T[2-2 mod 15])<<< 3)⊕(4.2 + 0)
= ((T[5] ⊕ T[0])<<< 3) ⊕ 8
= ((0 ⊕ k[0])<<< 3) ⊕ 8
= ((00000000 00000000 00000000 00000000 ⊕ 01000001 01000010 01000011 01000100)<<< 3) ⊕ 8
= ((01000001 01000010 01000011 01000100)<<< 3) ⊕ 8
= (10001000 00101000 01001000 01101000) ⊕ 00000000 00000000 00000000 00001000
= 10001000 00101000 01001000 01100000

T[3] = ((*T*[3-7 mod 15]⊕ T[3-2 mod 15])<<< 3)⊕(4.3 + 0)
= ((T[4] ⊕ T[1])<<< 3) ⊕ 12
= ((4 ⊕ k[1])<<< 3) ⊕ 12
= ((00000000 00000000 00000000 00000100 ⊕ 01000101 01000110 01000111 01001000)<<< 3) ⊕ 12
= ((01000101 01000110 01000111 01001100)<<< 3) ⊕ 12
= (10001000 10101000 11001000 11101001) ⊕ 00000000 00000000 00000000 00001100
= 10001000 10101000 11001000 11100100

Untuk i dari 4 sampai dengan 14 dan j dari 1 sampai dengan 3 dilakukan dengan cara yang sama untuk mendapatkan T[] sebanyak 15 *words.*

Setelah itu dilakukan empat pengulangan lagi untuk mendapatkan T[i] baru.

For i = 0 to 14 do
T[*i*] =(*T*[*i*] + S[low 9 bits of *T*[*i* - 1 mod 15]])<<< 9

Dimulai dari i = 0
T[0] =(*T*[0] + S[low 9 bits of *T*[0 - 1 mod 15]])<<< 9
=(10001001 00101001 01001001 01101001 + S[low 9 bits of *T*[1]]) <<< 9
=(10001001 00101001 01001001 01101001 + S[361]) <<< 9
=(10001001 00101001 01001001 01101001 + 0x000399bd) <<< 9
=(10001001 00101001 01001001 01101001 + 00000000 00000011 10011001 10111101) <<< 9
= 10001001 00101100 11100011 00100110 <<< 9
= 10010011 01000100 10010110 01110001

T[1] =(*T*[1] + S[low 9 bits of *T*[1 - 1 mod 15]])<<< 9
=(00001000 10101000 11001000 11101101 + S[low 9 bits of T[0]]) <<< 9
=(00001000 10101000 11001000 11101101 + S[361]) <<< 9
=(00001000 10101000 11001000 11101101 + 0x000399bd) <<< 9
=(00001000 10101000 11001000 11101101 + 00000000 00000011 10011001 10111101) <<< 9

= 00001000 10101100 01100010 10101010 <<< 9
= 01010101 00000100 01010110 00110001

T[2] =(*T*[2] + S[low 9 bits of *T*[2 - 1 mod 15]])<<< 9
=(10001000 00101000 01001000 01100000 + S[low 9 bits of T[1]]) <<< 9
=(10001000 00101000 01001000 01100000 + S[361]) <<< 9
=(10001000 00101000 01001000 01100000 + 0x000399bd) <<< 9
=(10001000 00101000 01001000 01100000 + 00000000 00000011 10011001 10111101) <<< 9
= 10001000 00101011 11100010 00011101 <<< 9
= 00001110 11000100 00010101 11110001

T[3] =(*T*[3] + S[low 9 bits of *T*[3 - 1 mod 15]])<<< 9
=(10001000 10101000 11001000 11100100 + S[low 9 bits of T[2]]) <<< 9
=(10001000 10101000 11001000 11100100 + S[96]) <<< 9
=(10001000 10101000 11001000 11100100 + 0x5ded0ab8) <<< 9
=(10001000 10101000 11001000 11100100 + 01011101 11101101 00001010 10111000) <<< 9
= 11100110 10010101 11010011 10011100 <<< 9
= 11001110 01110011 01001010 11101001

Dilakukan proses untuk mendapatkan *expansion key* dari K[0] sampai dengan K[39]

for i = 0 to 9 do
Untuk i = 0.....9 dan j = 0
K[10j+i] = T[4i mod 15]
K[10.0 + 0] = T[4.0 mod 15]
K[0] = T[0]
K[1] = T[4]
K[2] = T[8]
K[3] = T[12]
K[4] = T[1]
K[5] = T[5]
K[6] = T[9]
K[7] = T[13]
K[8] = T[2]
K[9] = T[6]

Untuk i = 0.....9 dan j = 1
K[10j+i] = T[4i mod 15]
K[10.1 + 0] = T[4.0 mod 15]
K[10] = T[0]
K[11] = T[4]
K[12] = T[8]
K[13] = T[12]
K[14] = T[1]
K[15] = T[5]
K[16] = T[9]
K[17] = T[13]
K[18] = T[2]
K[19] = T[6]

Untuk i = 0.....9 dan j = 2
K[10j+i] = T[4i mod 15]
K[10.2 + 0] = T[4.0 mod 15]
K[20] = T[0]
K[21] = T[4]
K[22] = T[8]
K[23] = T[12]
K[24] = T[1]
K[25] = T[5]
K[26] = T[9]
K[27] = T[13]
K[28] = T[2]
K[29] = T[6]
Untuk i = 0.....9 dan j = 3
K[10j+i] = T[4i mod 15]
K[10.3 + 0] = T[4.0 mod 15]
K[30] = T[0]
K[31] = T[4]
K[32] = T[8]
K[33] = T[12]

K[34] = T[1]
K[35] = T[5]
K[36] = T[9]
K[37] = T[13]
K[38] = T[2]
K[39] = T[6]

Dari semua kunci K[0] sampai dengan K[39] digunakan dalam proses enkripsi dekripsi algoritma MARS yang terdiri dari tahap *forward mixing*, transformasi kunci utama dan *backward mixing*.

Kesimpulan dan Saran

Adapun kesimpulan yang diperoleh dari pembuatan program aplikasi ini adalah sebagai berikut :

1. Perancangan program aplikasi kriptografi dengan menggunakan algoritma MARS ini dapat memberikan sekuriti yang lebih baik terhadap sebuah komputer.
2. Data yang diproses dapat dipercaya kerahasiaannya karena telah diacak menggunakan iterasi sebanyak 32 round dan masing-masing round mengalami fungsi feistel, fungsi E dan S-box dengan panjang kunci 128 bit.

Sedangkan saran yang dapat diberikan untuk pengembangan lebih lanjut dari program aplikasi kriptografi ini adalah, bagi yang ingin mengembangkan program ini, kunci yang digunakan dapat dibuat lebih panjang dari 128 bit, misalnya 256 bit.

Daftar Pustaka

Andi, "Memahami Model Enkripsi dan Security Data", Wahana Komputer, Yogyakarta, 2003.

Anonimus, ***"****Randomness Testing of the Advanced Encryption Standard Finalist Candidates"*, http:// www.tropsoft. com/strongenc/mars.html, 14 Oktober 1999.

Burwick, Carolynn et al ., *"MARS : A 128-BitBlockCipher"*, http://researchweb.watson.ibm. com /security /mars. html, 22 September 1999.

Kusumo, Ario Suryo, "Buku Latihan, Microsoft Visual Basic 6.0", PT Elex Media Komputindo, Jakarta, 2000.

Pfleeger, Charles P, *"Security in Computing",* 2nd Edition. Upper Prentice Hall, Saddle River, 1997.

Stalling, William, *"Cryptography and Network Security",* 3rd Edition, Prentice Hall, Upper Saddle River 2003.