IMPLEMENTASI ALGORITMA TEA DAN FUNGSI HASH MD4 UNTUK ENKRIPSI DAN DEKRIPSI DATA

Oleh : Nurdin [5]

Abstrak

*Keamanan data merupakan salah satu aspek terpenting dalam teknologi informasi. Dengan tingkat keamanan yang tinggi, diharapkan informasi yang disajikan dapat terjaga keasliannya. Pada penelitian ini menghasilkan suatu sistem yang mengamankan data dan informasi yang tersimpan pada komputer dari gangguan para kriptanalis. Proses penyandian yang dilakukan hanya pada pesan dalam format teks (yaitu dalam format *.txt). Sistem ini dibangun menggunakan perangkat lunak Borland Delphi 7.0. Tahapan yang dilakukan untuk melakukan proses pembentukan sistem tersebut meliputi tahapan analisis permasalahan, perancangan aplikasi yang melibatkan diagram alir data, algoritma dan flowchart beserta pemodelan struktur program dan desain antarmuka aplikasi, sehingga aplikasi yang terbentuk menjadi mudah dipergunakan dan memiliki fungsi yang optimal. Dengan menggunakan Algoritma TEA yang merupakan algoritma kriptografi kunci rahasia, permasalahan tersebut dapat diatasi. Kekuatan algoritma ini terletak pada jaringan feistel (meliputi operasi subtitusi, permutasi dan modular arithmetic) dan bilangan delta yang berasal dari golden number. Hasil dari proses enkripsi adalah ciperteks yang berukuran 16 byte atau kelipatannya. Algoritma kriptografi TEA lebih aman dikarenakan jumlah round serta panjang kuncinya yang lebih panjang.*

Kata Kunci: *Kriptografi, Algoritma, Tea, Ciperteks, Enkripsi, Dekripsi*

[5] Program Studi Teknik Informatika Fakultas Teknik Universitas Malikussaleh
E-Mail: nurdin_um@ymail.com

JT-FTI
V2,N1
97-112

PENDAHULUAN

Kriptografi merupakan salah satu komponen yang tidak dapat diabaikan dalam membangun keamanan komputer. Kriptografi bertujuan agar pesan atau data yang dikirim tidak dibaca oleh pihak yang tidak berhak dan kalaupun data itu bisa dibaca maka tidak bisa dimengerti oleh pihak tersebut, karena data sudah dienkripsi dan belum didekripsikan. Data yang akan dikirimkan dan belum mengalami penyandian dikenal dengan istilah plainteks dan setelah disamarkan dengan suatu cara penyandian, maka *plainteks* ini akan berubah menjadi *cipherteks*.

Sudah banyak algoritma kriptografi yang ditemukan dan banyak juga kriptanalisnya. Semakin tinggi tingkat keamanan suatu algoritma kriptografi biasanya disertai dengan meningkatnya beban proses enkripsi dan dekripsinya. Hal ini akan menjadi masalah besar jika diaplikasikan pada perangkat elektronik seperti *handphone* di mana penggunaan daya listrik dan *memory* sangat diperhatikan. TEA dan turunannya diklaim mempunyai tingkat keamanan yang tinggi jika dibandingkan dengan algoritma kriptografi sejenis. Keunggulan utama dari TEA adalah keringanan prosesnya, operasi-operasi yang digunakan hanya berupa operasi bit biasa, tanpa substitusi, permutasi, ataupun operasi matrix.

Berdasarkan uraian diatas, dapat diambil kesimpulan bahwa sistem pengamanan sangat dibutuhkan untuk menjaga kerahasiaan suatu data maupun informasi. Algoritma TEA memiliki tingkat keamanan yang tinggi jika dibandingkan dengan algoritma kriptografi sejenis, selain itu juga algoritma TEA juga memiliki banyak keunggulan

TINJAUAN PUSTAKA

Definisi Kriptografi

Menurut Munir (2006), kriptografi (*cryptography*) berasal dari kata '**kryptos**' yang artinya tersembunyi dan '**grafia**' yang artinya sesuatu yang tertulis (bahasa Yunani) sehingga kriptografi dapat juga disebut sebagai sesuatu yang tertulis secara rahasia (tersembunyi), Schneier(1996).

Menurut Menezes *et al* (1996), kriptografi adalah ilmu yang mempelajari teknik – teknik matematika yang berhubungan dengan aspek-aspek pada keamanan informasi misalnya kerahasiaan, integritas data, otentikasi pengirim / penerima data, dan otentikasi data. Dengan pengembangan bidang kriptografi, pembagian antara apa yang termasuk kriptografi dan apa yang tidak telah menjadi kabur. Dewasa ini, kriptografi dapat dianggap sebagai perpaduan antara studi teknik dan aplikasi yang tergantung kepada keberadaan masalah – masalah sulit.

Terdapat dua proses penting di dalam kriptografi yang berperan dalam merahasiakan suatu informasi yakni enkripsi (*encryption*) dan

dekripsi (*decryption*). Enkripsi adalah transformasi data (*plaintext*) ke dalam bentuk yang hampir tidak dapat dibaca (*ciphertext*) tanpa pengetahuan yang cukup. Tujuan dari enkripsi adalah untuk menjamin kerahasiaan dengan menjaga informasi tersembunyi dari siapapun yang bukan pemilik atau yang berkepentingan dengan informasi tersebut, bahkan bagi orang yang memiliki akses terhadap data yang telah dienkripsi. Sedangkan dekripsi adalah kebalikan dari enkripsi, yakni transformasi dari data yang telah dienkripsi (*ciphertext*) kembali ke bentuk semula (*plaintext*). Proses enkripsi dan dekripsi pada umumnya membutuhkan penggunaan sejumlah informasi yang rahasia, yang sering disebut kunci (*key*).

Keamanan data merupakan salah satu aspek terpenting dalam teknologi informasi.

Nurdin

Tujuan Kriptografi

Menurut Stalling (1998), ada beberapa tuntutan yang terkait dengan isu keamanan data yaitu :

1. *Confidentiality*
   Menjamin bahwa data-data tersebut hanya bisa diakses oleh pihak-pihak tertentu saja.
2. *Authentication*
   Baik pada saat mengirim atau menerima informasi, kedua belah pihak perlu mengetahui bahwa pengirim dari pesan tersebut adalah orang yang sebenarnya seperti yang diklaim.
3. *Integrity*
   Tuntutan ini berhubungan dengan jaminan setiap pesan yang dikirim pasti sampai pada penerimanya tanpa ada bagian dari pesan tersebut yang diganti, diduplikasi, dirusak, diubah urutannya, dan ditambahkan.
4. *Nonrepudiation*
   Nonrepudiation mencegah pengirim maupun penerima mengingkari bahwa mereka telah mengirimkan atau menerima suatu pesan/informasi. Jika sebuah pesan dikirim, penerima dapat membuktikan bahwa pesan tersebut memang dikirim oleh pengirim yang tertera. Sebaliknya, jika sebuah pesan diterima, pengirim dapat membuktikan bahwa pesannya telah diterima oleh pihak yang ditujunya.
5. *Access Control*
   Membatasi sumber-sumber data hanya kepada orang-orang tertentu.
6. *Availability*
   Jika diperlukan setiap saat semua informasi pada sistem komputer harus tersedia bagi semua pihak yang berhak atas informasi tersebut.

TINY ENCRIPTION ALGORITMA (TEA)

Tiny Encription Algorithm (TEA) merupakan suatu algoritma sandi yang diciptakan oleh David Wheeler dan Roger Needham dari Computer Laboratory, Cambridge University, England pada bulan November 1994.

JT-FTI
V2,N1
97-112

Algoritma ini merupakan algoritma penyandian *block cipher* yang dirancang untuk penggunaan memory yang seminimal mungkin dengan kecepatan proses yang maksimal.

System penyandian TEA menggunakan proses *feistel network* dengan menambahkan fungsi matematik berupa penambahan dan pengurangan sebagai operator pembalik selain XOR. Hal ini dimaksudkan untuk menciptakan sifat non-linearitas. Pergeseran dua arah (ke kiri dan ke kanan) menyebabkan semua bit kunci dan data bercampur secara berulang ulang.

Algoritma TEA merupakan algoritma kriptography simeteris atau disebut juga algoritma kriptography konvensional yaitu algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi.

Cara Kerja Algoritma TEA

1. Pergeseran *(shift)*

Blok teks terang pada kedua sisi yang masing masing sebanyak 32-bit akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.

2. Penambahan

Langkah selanjutnya setelah digeser kekiri dan kekanan, maka Y dan Z yang telah digeser akan ditambahkan dengan kunci k[0]-k[3]. Sedangkan Y dan Z awal akan ditambahkan dengan sum (delta).

3. Peng-XOR-an

Proses selanjutnya setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng-XOR-an dengan rumus untuk satu *round* adalah sebagai berikut:

y = y + (((z<<4)+k[0])^z+sum^((z>>5)+k[1])) ..... (2.1)

z = z + (((y<<4)+k[2]^y+sum^((y>>5)+k[3])) ..... (2.2)

Hasil penyandian dalam satu *cycle* satu blok teks terang 64-bit menjadi 64-bit teks sandi adalah dengan menggabungkan Y dan Z. Untuk penyandian pada *cycle* berikutnya Y dan Z ditukar posisinya, sehingga Y_1 menjadi Z_1 dan Z_1 menjadi Y_1 lalu dilanjutkan proses seperti langkah-langkah diatas sampai dengan *16 cycle (32 round).*

4. Key Schedule

Algoritma TEA menggunakan *key schedule*-nya sangat sederhana. Yaitu kunci k[0] dan k[1] konstan digunakan untuk *round* ganjil sedangkan kunci k[2] dan k[3] konstan digunakan untuk *round* genap.

5. Dekripsi dan Enkripsi

Proses dekripsi sama halnya seperti pada proses penyandian yang berbasis *feistel cip*her lainnya. Yaitu pada prinsipnya adalah sama pada saat proses enkripsi. Hal yang berbeda adalah penggunaan teks sandi sebagai *input* dan kunci yang digunakan urutannya dibalik. Proses dekripsi semua *round* ganjil menggunakan k[1] terlebih dahulu kemudian k[0], demikian juga dengan semua *round* genap digunakan k[3] terlebih dahulu kemudian k[2]. Rumus untuk enkripsi dekripsi seperti dibawah ini:

Proses enkripsi digunakan rumus :

$L_0 = L_0 + f (R_0 , k[0], k[1], sum)$ ..... (2.3)

$R_0 = R_0 + f (L_0, k[2], k[3], sum)$ ..... (2.4)

Jadi L_0 merupakan hasil penjumlahan dari L_0 ditambahkan dengan f (R_0 , k[0], k[1], sum).

Proses enkripsi untuk satu round digunakan rumus:

y = y + (((z<<4)+k[0])^z+sum^((z>>5)+k[1])) ..... (2.5)

z = z + (((y<<4)+k[2]^y+sum^((y>>5)+k[3])) ..... (2.6)

Proses dekripsi digunakan rumus :

$L_0 = L_0 + f (R_0 , k[1], k[0], sum)$ ..... (2.7)

$R_0 = R_0 + f (L_0, k[3], k[2], sum)$ ..... (2.8)

Jadi L_0 merupakan hasil penjumlahan dari L_0 ditambahkan dengan f (R_0 , k[0], k[1], sum).

Proses dekripsi untuk satu round digunakan rumus:

y = y + (((z<<4)+k[1])^z+sum^((z>>5)+k[0])) ..... (2.9)

z = z + (((y<<4)+k[3]^y+sum^((y>>5)+k[2])) ..... (2.10)

Rumus Y diatas menjelaskan bahwa Y merupakan hasil dari Y yang ditambahkan dengan Z yang yang digeser kekiri sebanyak empat kali dengan penambahan kunci k[1]. Kemudian hasil penjumlahan tadi di XOR kan dengan Z yang dijumlahkan dengan sum(delta). Hasil dari peng-XOR-an dari kedua penjumlahan tadi di XORkan lagi dengan Z yang digeser kekanan sebanyak lima kali dengan penambahan kunci k[0]. Demikian juga

dengan rumus Z sama halnya dengan rumus Y, hanya kunci yang digunakan menggunakan kunci k[3] dan k[2].

JT-FTI
V2,N1
97-112

FUNGSI HASH MD4

MD4 didesain untuk sebagai algoritma fungsi hash yang memiliki kecepatan dalam segi waktu. MD4 dibuat oleh Ronald Rivest pada Oktober 1990. Pertama-tama kita misalkan pesan memiliki sejumlah b-bit pesan sebagai input, dan kita menginginkan untuk menghasilkan message digest dari pesan tersebut. Dalam hal ini, b merupakan sebuah bilangan bulat positif, mungkin nol, dan bilangan tersebut harus kelipatan dari 8. Kita membagi pesan tersebut sebagai berikut :

m_0 m_1 . . . m_{b-1}

Di bawah ini akan dijelaskan lima tahap proses menghasilkan message digest untuk algoritma MD4.

Memasukkan Padding Bit

Pertama-tama dilakukan padding pada pesan awal sehingga panjangnya (dalam satuan bit) kongruen dengan 448 modulo 512. Maka pesan tersebut kekurangan 64 bit untuk mencapai kelipatan 512. Padding selalu dilakukan sehingga pesan tersebut kongruen dengan 512. Padding bit awal yaitu bit "1", selanjutnya ditambahkan bit "0" sehingga pesan tersebut memiliki panjang yang kongruen dengan 448 modulo 512. Jadi, panjang padding paling sedikit adalah satu bit hingga 512 bit.

Memasukkan Panjang Pesan

Sebuah 64-bit yang direpresentasikan oleh b (panjang pesan awal sebelum dilakukan padding bit) dimasukkan pada hasil pesan untuk tahap satu di atas. Panjang pesan yang digunakan selalu lebih kecil dari 2^{64}. Bila panjang pesan melebihi 2^{64}, maka order terendah yang direpresentasikan oleh 2^{64} yang akan digunakan.

Oleh karena itu, setelah proses ini, pesan akan memiliki panjang kelipatan dari 512 bit. Dan pesan dibagi dalam kelipatan 32 bit dalam M[0 . . . N-1] yang menotasikan pesan, dimana N adalah kelipatan dari 16.

Inisialisasi Penyangga

Ada empat peubah penyangga yang digunakan, yaitu A, B, C, D yang digunakan untuk menghasilkan message digest.Masing-masing variable ini merepresentasikan sebuah nilai 32-bit register.

Register tersebut diinisialisasi dengan nilai dalam bentuk heksadesimal, dalam order terendah terlebih dahulu.

Proses Pesan Dalam Blok 32-bit

Untuk tahap ini pertama-tama kita definisikan terlebih dahulu fungsi-fungsi pelengkap yang dibutuhkan untuk menghasilkan message digest, Terdapat tiga fungsi pelengkap yang dalam hal ini tiap-tiap fungsi ini menerima input 32-bit dan menghasilkan output 32 bit juga.

Dalam tiap bit posisi F pada fungsi berlaku hubungan kondisional berikut : if X then Y else Z. Fungsi F seharusnya didefinisikan menggunakan atau tanpa v selama XY dan not(X)Z tidak memiliki bit "1" pada posisi yang sama. Dalam tiap bit posisi G berlaku hubungan berikut : jika paling tidak dua dari X, Y, Z sedang digunakan, maka G memiliki sebuah bit "1" dalam posisi tersebut, dan jika tidak G memilii sebuah bit "0". Ini menarik untuk diperhatikan bahwa jika bit-bit X, Y, dan Z saling bebas tidak saling mempengaruhi satu sama lain, tiap-tiap bit dari f(X,Y,Z) akan menjadi saling bebas pula. Fungsi H merupakan operasi bit XOR atau fungsi parity. Dia memiliki atribut yang mirip dengan F dan G.

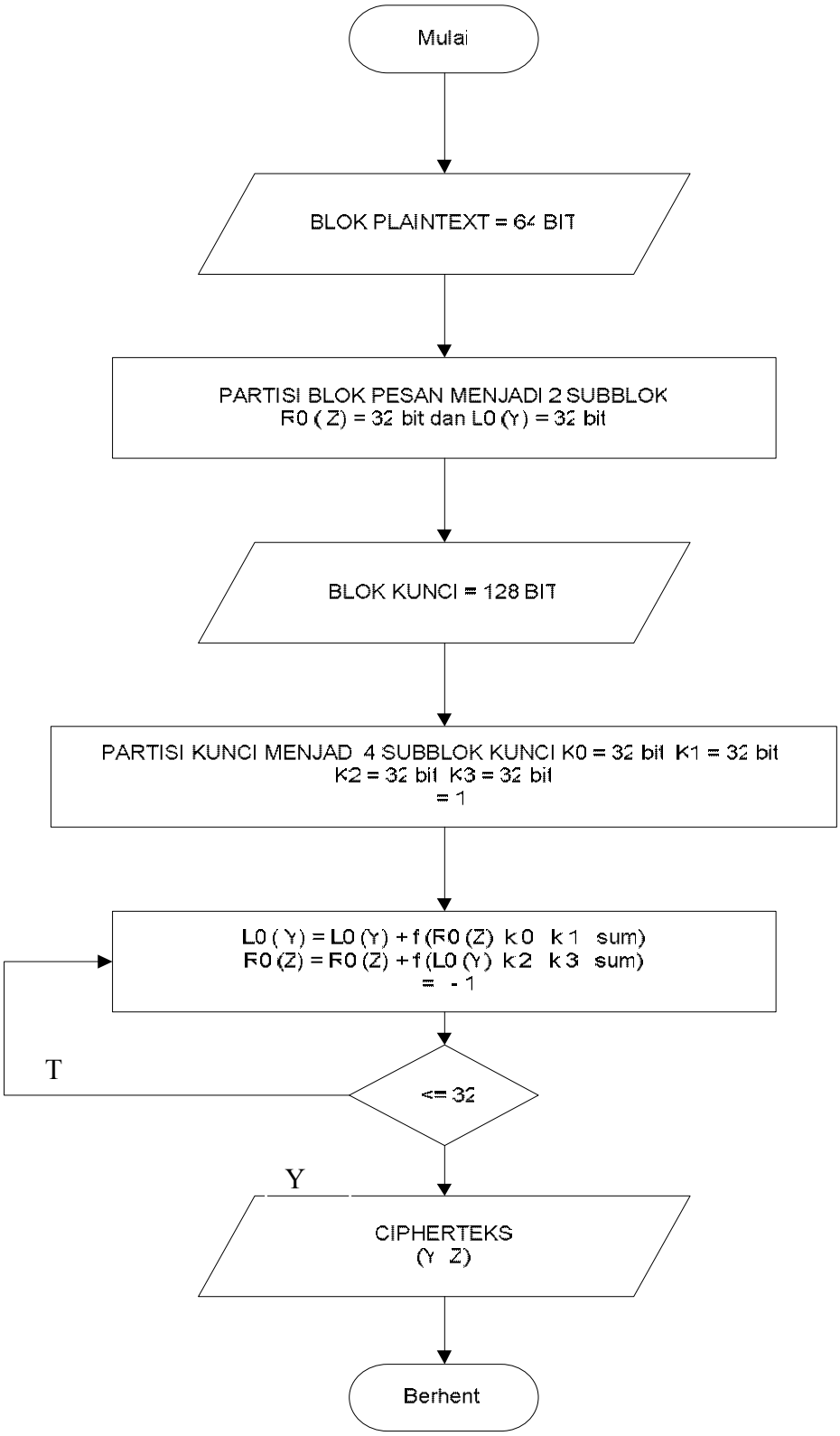
FLOWCHART SISTEM PROSES ENKRIPSI DAN DEKRIPSI

Pada perangkat lunak Kriptografi algoritma TEA, terdapat dua prosedur utama, yaitu prosedur Encryption dan prosedur *Descryption*. Adapun algoritma dan flowchart untuk Kriptografi algoritma TEA adalah sebagai berikut :

Flowchart Enkripsi

Gambaran aliran proses kerja dari prosedur enkripsi dapat ditunjukkan pada flowchart dibawah ini:

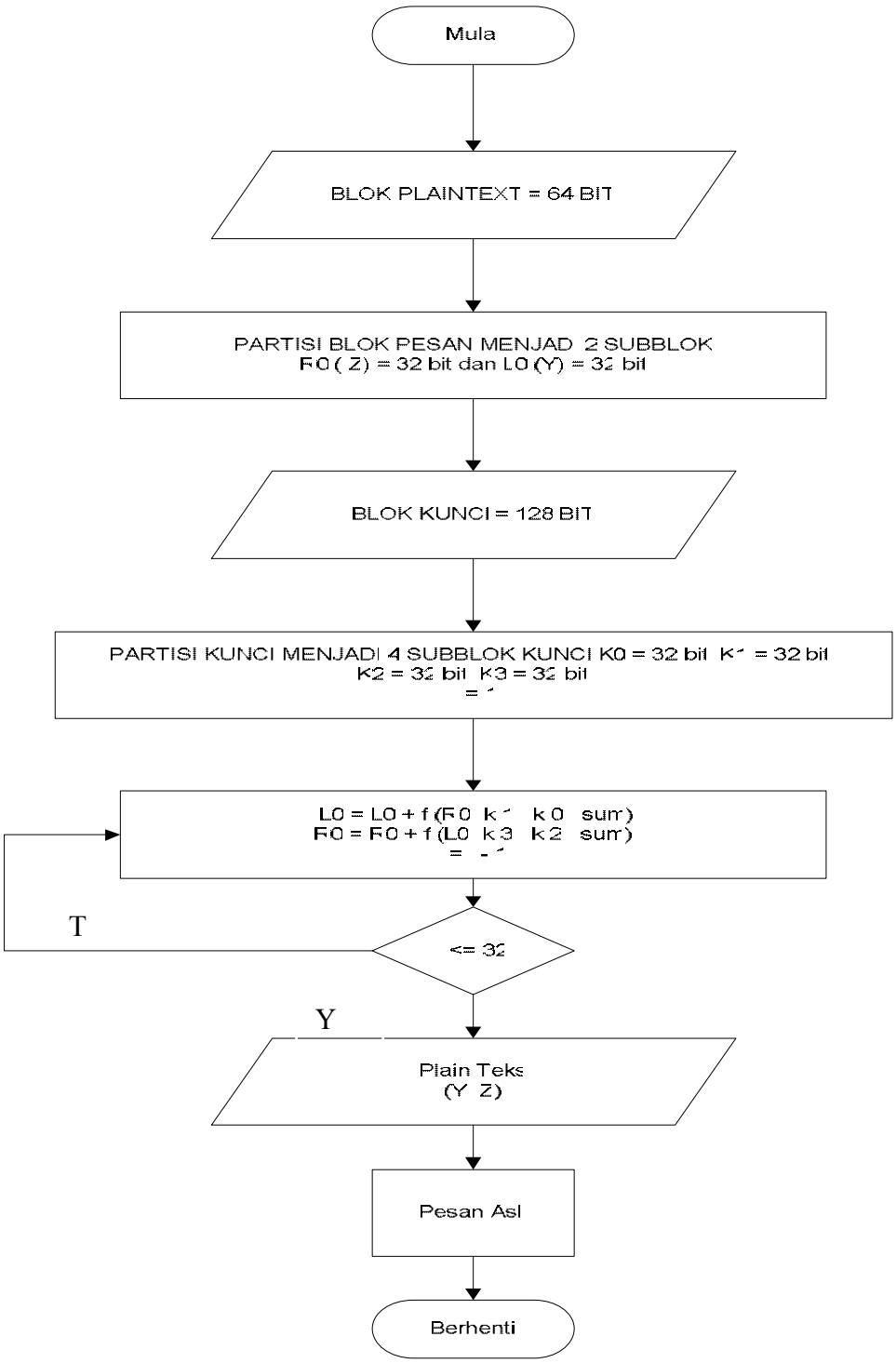
JT-FTI
V2,N1
97-112



Gambar Flowchart Enkripsi

Flowchart Dekripsi

Gambaran aliran proses kerja dari prosedur dekripsi dapat ditunjukkan pada flowchart dibawah ini :



Gambar Flow Chart Dekripsi

JT-FTI
V2,N1
97-112

ANALISA SISTEM

Untuk melakukan *enkripsi*, proses diawali dengan *input*-bit teks terang sebanyak 64-bit. Kemudian 64-bit teks tersebut dibagi menjadi dua bagian, yaitu sisi kiri (L0) sebanyak 32-bit dan sisi kanan (R0) sebanyak 32-bit. Setiap bagian teks terang akan dioperasikan sendiri-sendiri. R0 (z) akan digeser kekiri sebanyak empat (4) kali dan ditambahkan dengan kunci k[0]. Sementara itu z ditambah dengan sum (delta) yang merupakan konstanta. Hasil penambahan ini di-XOR-kan dengan penambahan sebelumnya. Kemudian di-XOR-kan dengan hasil penambahan antara z yang digeser kekanan sebanyak lima (5) kali dengan kunci k[1]. Hasil tersebut kemudian ditambahkan dengan L (y) yang akan menjadi R .

Sisi sebelah kiri akan mengalami proses yang sama dengan sisi sebelah kanan. L0 (y) akan digeser kekiri sebanyak empat (4) kali lalu ditambahkan dengan kunci k[2]. Sementara itu, Y ditambah dengan sum (delta). Hasil penambahan ini di-XOR-kan dengan penambahan sebelumnya. Kemudian di-XOR-kan dengan hasil penambahan antara Y yang digeser ke kanan sebanyak lima (5) kali dengan kunci k[3]. Hasil tersebut kemudian ditambahkan dengan R0 (Z) yang akan menjadi L1.

Untuk proses deskripsi pada algoritma TEA sama halnya dengan proses enkripsinya. Hanya saja terjadi perbedaan pada penjadwalan kuncinya yaitu pada proses enkripsi untuk cipher R yang mengalami pergeseran bit ke kiri sebanyak 4 bit digunakan kunci k[0] pada proses deskripsi digunakan kunci k[1], untuk cipher R yang mengalami pergeseran ke kanan sebanyak 5 bit menggunakan kunci [1] pada proses deskripsi menggunakan kunci k[0]. Begitu juga halnya dengan cipher L, pada proses enkripsi untuk cipher L yang mengalami pergeseran ke kiri sebanyak 4 bit menggunakan kunci k[2] pada proses deskripsi digunakan kunci k[3]. Untuk cipher L yang mengalami pergeseran ke kanan sebanyak 5 bit digunkan kunci k[3] pada proses deskripsi digunakan kunci k[2].

HASIL DAN PEMBAHASAN

Proses Enkripsi

Pada proses ini string yang diinputkan untuk dienkripsikan adalah “ENKRIPSI” . Berikut ini adalah proses perhitungan manualnya :

Konversi ke biner :

Kode ascii dari 'E' = 01000101
Kode ascii dari 'N' = 01001110
Kode ascii dari 'K' = 01001011
Kode ascii dari 'R' = 01010010
Kode ascii dari 'I' = 01001001
Kode ascii dari 'P' = 01010000

Kode ascii dari 'S' = 01010011
Kode ascii dari 'I' = 01001001

Plain Text (dalam biner) =
01000101010011100100101101010010010010010101000001010011010010
01
R0(Z) = 01000101010011100100101101010010
L0(Y) = 01001001010100000101001101001001
KUNCI = TEK. INFORMATIKA

Konversi biner
01010100010001010100101100101110001000000100100101001110010000
11001001111010100100100110101000001010101000100100101001011010
00001

Kunci dibagi menjadi empat blok :
K(0) = 01010100010001010100101100101110
K(1) = 00100000010010010100111001000110
K(2) = 01001111010100100100110101000001
K(3) = 01010100010010010100101101000001

Lakukan proses proses berikut sampai nilai i = 32, proses yang dilakukan adalah sebagai berikut :
L0(Y) = L0(Y) + f(R0(Z),K(0), K(1), sum)
R0(Z) = R0(Z) + f(L0(Z),K(2), K(3), sum)

Disaat nilai i = 32, kemudian satukan kembali biner dari L0(Y) dan R0(Z) Kemudian pra chippertext diproses dengan algoritma MD4 untuk mendapatkan chippertext, proses yang dilakukan adalah sebagai berikut :

$F(x,y,z) = (x \cap y) \cup (\neg x \cap z)$
$G(x,y,z) = (x \cap z) \cup (y \cap \neg z)$
$H(x,y,z) = x \text{ XOR } y \text{ XOR } z$
$I(x,y,z) = Y \text{ XOR } (X \cup \neg z)$

00111000001001010001001000101000001101000011101100001010001100
11

Chippertext dalam bentuk Hexadesimal adalah sebagai berikut :
Kode ascii dari 00111000 = 38
Kode ascii dari 00100101 = 25
Kode ascii dari 00100101 = 12

JT-FTI
V2,N1
97-112

Kode ascii dari 00101000 = 28
Kode ascii dari 00110100 = 34
Kode ascii dari 00111011 = 3B
Kode ascii dari 00001010 = 0A
Kode ascii dari 00110011 = 33

Jadi hasil dari proses enkripsi adalah sebagai berikut :
38251228343B0A33

Proses Deskripsi

Pada proses ini string yang diinputkan untuk didekripsikan adalah "38251228343B0A33 " Berikut ini adalah proses perhitungan manualnya :
Konversi ke biner :
Kode ascii dari 38 = 00111000
Kode ascii dari 25 = 00100101
Kode ascii dari 12 = 00100101
Kode ascii dari 28 = 00101000
Kode ascii dari 34 = 00110100
Kode ascii dari 3B = 00111011
Kode ascii dari 0A = 00001010
Kode ascii dari 33 = 00110011

Chipertext dalam biner :
00111000001001010001001000101000001101000011101100001010001100
11
R0(Z) = 00111000001001010001001000101000
L0(Y) = 00110100001110110000101000110011
KUNCI = TEK. INFORMATIKA

Konversi biner
01010100010001010100101100101110001000000100100101001110010000
11001001111010100100100110101000001010101000100100101001011010
00001

Kunci dibagi menjadi empat blok :
K(0) = 01010100010001010100101100101110
K(1) = 00100000010010010100111001000110
K(2) = 01001111010100100100110101000001
K(3) = 01010100010010010100101101000001

Lakukan proses proses berikut sampai nilai i = 32, proses yang dilakukan adalah sebagai berikut :

L0(Y) = L0(Y) + f(R0(Z),K(1), K(0), sum)

R0(Z) = R0(Z) + f(L0(Z),K(3), K(2), sum)

Disaat nilai i = 32, kemudian satukan kembali biner dari L0(Y) dan R0(Z)

Kemudian pra plaintext di proses dengan algoritma MD4 untuk mendapatkan plaintext, proses yang dilakukan adalah sebagai berikut :

$F(x,y,z) = (x \cap y) \cup (\neg x \cap z)$

$G(x,y,z) = (x \cap z) \cup (y \cap \neg z)$

$H(x,y,z) = x \text{ XOR } y \text{ XOR } z$

$I(x,y,z) = Y \text{ XOR } (X \cup \neg z)$

Hasil dekripsi adalah sebagai berikut :

0100010101001110010010110101001001001001010100000101001101001
001

Kode ascii dari 01000101 = 'E'

Kode ascii dari 01001110 = 'N'

Kode ascii dari 01001011 = 'K'

Kode ascii dari 01010010 = 'R'

Kode ascii dari 01001001 = 'I'

Kode ascii dari 01010000 = 'P'

Kode ascii dari 01010011 = 'S'

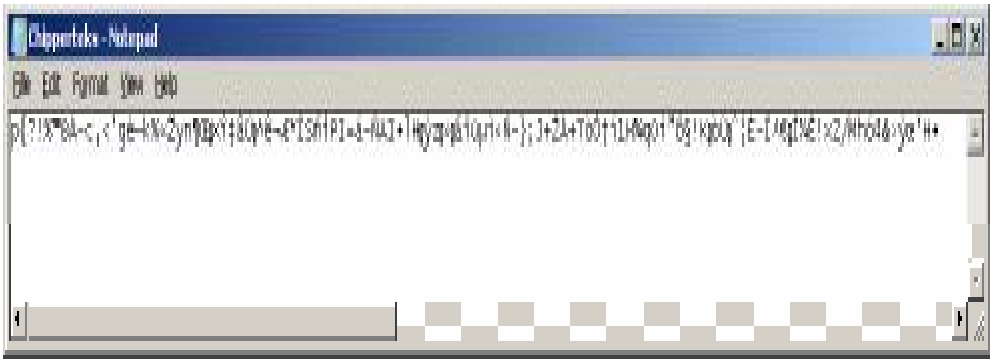
Kode ascii dari 01001001 = 'I'

Jadi hasil dari proses dekripsi adalah sebagai berikut :

"ENKRIPSI"

HASIL ENKRIPSI

Berikut ini adalah contoh *file* hasil dienkripsi dengan algoritma TEA dan fungsi Hash MD4.

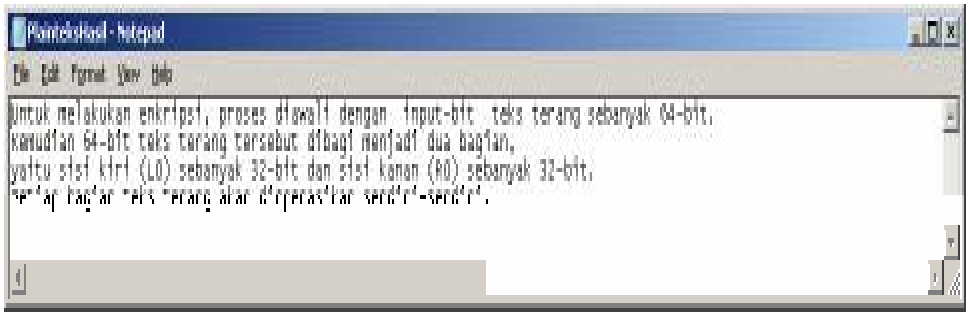


Gambar Hasil Enkripsi

JT-FTI
V2,N1
97-112

HASIL DEKRIPSI

Berikut ini adalah contoh *file* hasil dekripsi dengan algoritma TEA dan fungsi Hash MD4.



Gambar Hasil Dekripsi

KESIMPULAN

1. Algoritma kriptografi TEA lebih aman, hal ini dikarenakan jumlah *round* serta panjang kuncinya yang lebih panjang dan prosedur algoritma *enkripsi*nya yang dirancang lebih kuat daripada algoritma DES serta tidak membutuhkan S-Box dan P-Box dalam proses *enkripsi* dan *deskripsi*nya sehingga meminimalkan penggunaan memori dan dapat memaksimalkan proses.
2. Kapasitas *file plaintext* sama dengan kapasitas *ciphertext,* sehingga dapat dijadikan sebagai otentikasi data.
3. Dalam melakukan proses *enkripsi*, sebaiknya usahakan untuk tidak menggunakan kunci yang sama untuk meng *enkripsi file* yang berbeda. Selanjutnya, agar kinerja algoritma lebih optimal sebaiknya algoritma TEA digabungkan dengan algoritma kriptografi *asimetris* (*hybrid cryptography*) dimana algoritma TEA hanya digunakan untuk proses *enkripsi* dan *deskripsi* pesan, sementara untuk pembentukan kunci digunakan algoritma asimetri agar kunci yang didistribusikan tetap aman.

DAFTAR PUSTAKA

Bruce Schneier, ***Applied Crytography***, *Second Edition*, John Willey and Sons Inc., 1996.

David Khan, ***The Codebreaker***, The Macmillan Company,1973.

Jusuf Kurniawan, Kriptografi, **Keamanan Internet dan Jaringan Komunikasi**,
Penerbit Informatika Bandung, 2004.

Jennifer Seberpy, Jojef Pieprzyk, ***Cryptography : An Introduction to Computer***
Security.

Menezes, A. et al, ***Handbook Of Applied Cryptography*** , IBCRC Press, 1996.

Rinaldi Munir, ***Kriptografi,*** Penerbit Informatika, 2006.

William Stallings, " Cryptography and Network Security: Principles and Practices " ,
3rd edition, Pearson Education International, 2003.

http://edipermadi.files.wordpress.com/2008/06/tea-spec.pdf . Di akses pada tanggal 9 Februari 2012 14.00 WIB

&

JT-FTI
V2,N1
97-112