Aplikasi Pengamanan Data Menggunakan Metode Skipjack

Hartono
STMIK IBBI
Jl. Sei Deli No. 18 Medan, 061-4567111
e-mail: hartonoibbi@gmail.com

Abstrak

Kerahasiaan data adalah suatu hal yang sangat penting dan harus dijaga dengan baik. Hal tersebut dilakukan dengan tujuan untuk melindungi data dari pihak – pihak yang tidak mempunyai wewenang untuk membaca data tersebut. Apabila data yang sudah mendapatkan pengamanan dicuri ataupun terkirim secara tidak sengaja kepada pihak atau tujuan yang salah, maka pihak lain yang mendapatkan data tersebut tidak akan mudah untuk mengetahui isi dari data tersebut. Salah satu cara yang dapat dilakukan untuk mengamankan data adalah dengan menggunakan suatu metode enkripsi dalam ilmu kriptografi. Metode Skipjack adalah salah satu dari sekian banyak metode enkripsi yang ada dan merupakan metode yang sederhana karena hanya melibatkan dua buah operasi matematik kriptografi yakni XOR dan permutasi. Metode enkripsi yang baik adalah metode yang mempunyai struktur yang sederhana, tidak rumit namun harus tahan terhadap setiap jenis serangan. Metode Skipjack merupakan salah satu metode enkripsi yang cukup aman dari segala jenis serangan untuk saat ini.

Kata kunci : Enkripsi, Dekripsi, Kriptografi, Skipjack, *Permutation*, *XOR*, *cryptanalysis*

Abstract

Confidentiality of data is a very important and should be maintained properly. This is done in order to protect the data of the parties - parties that do not have the authority to read the data. If data security is getting stolen or accidentally sent to the wrong party or purpose, then the other party who obtained the data will not be easy to know the contents of the data. One way that can be done to secure the data is by using an encryption method in the science of cryptography. Skipjack method is one of the many existing encryption methods and is the simplest method because it involves only two cryptographic operation mathematical namely XOR and permutation. Good encryption method is a method that has a simple structure, it is not complicated but it should be resistant to any kind of attack. Skipjack is one method of encryption method that is safe from all types of attacks for now.

Keywords : Enkripsi, Dekripsi, Kriptografi, Skipjack, *Permutation*, *XOR*, *cryptanalysis*

1. Pendahuluan

Keamanan data adalah suatu hal yang sangat penting dan harus diperhatikan jika kita akan melakukan pengiriman data dari satu pihak ke pihak lainnya. Hal tersebut dilakukan karena mungkin saja data yang dikirimkan tersebut dicuri oleh pihak lain sebelum data tersebut sampai ke pihak yang dituju, ataupun salah kirim secara tidak sengaja ke tujuan yang salah. Oleh sebab itu, data harus diamankan terlebih dahulu sebelum dikirimkan agar data terlindungi dari pihak yang tidak memiliki izin untuk membaca dan mengetahui isi dari data yang dikirimkan tersebut. Salah satu teknik pengamanan data yang umum dilakukan adalah pengamanan dengan cara mengubah data asli ke dalam bentuk yang tidak terbaca dengan menggunakan seperangkat aturan tertentu yang hanya diketahui oleh pihak pengirim dan penerima, sehingga pihak lain yang tidak berhak atas data tersebut akan menemui kesulitan untuk mengetahui isi dari data yang sudah diamankan tersebut.

Metode Skipjack merupakan salah satu metode pengamanan data yang dikembangkan oleh National Security Agency (NSA) di Amerika Serikat yang digunakan untuk menjamin keamanan (*security*) dan privasi komunikasi via telepon. Metode Skipjack merupakan suatu metode yang sederhana dimana implementasinya tidak memerlukan perhitungan – perhitungan yang rumit dan hanya melibatkan 2 buah operasi matematik kriptografi yaitu XOR dan permutasi. Dari evaluasi yang dilakukan oleh para pakar atas undangan pemerintah Amerika Serikat ditemukan beberapa kehandalan metode Skipjack yang diantaranya sebagai berikut :

1. Resiko kalau metode Skipjack dapat dibobol melalui metode potong kompas (cara pintas / *shortcut method*) adalah sangat kecil,
2. Walaupun struktur internal algoritma Skipjack dirahasiakan, kekuatan Skipjack terhadap usaha-usaha analisis kriptografi (*cryptanalysis*) tidak bergantung kepada kerahasiaan algoritmanya.

2. Metode Penelitian

Adapun langkah – langkah yang dilakukan untuk menyelesaikan permasalahan yang telah disebutkan sebelumnya adalah sebagai berikut :

1. Mengumpulkan berbagai informasi yang berhubungan dengan teknik kriptografi metode Skipjack yang akan diteliti, yakni landasan teori dan algoritma. Selain informasi yang berhubungan dengan teknik kriptografi metode Skipjack, dikumpulkan juga informasi mengenai bahasa pemrograman yang digunakan di dalam perancangan aplikasi untuk mengetahui fasilitas – fasilitas yang disediakan oleh bahasa pemrograman tersebut sehingga akan mempermudah di dalam perancangan dan pembuatan perangkat lunak. Informasi tersebut diperoleh melalui buku – buku, jurnal, dan internet.
2. Mempelajari dan menganalisa informasi yang telah dikumpulkan untuk selanjutnya diterapkan di dalam perancangan perangkat lunak.
3. Melakukan perancangan perangkat lunak berdasarkan informasi tersebut.
4. Melakukan pengujian terhadap rancangan aplikasi, apabila terdapat kesalahan maka akan dilakukan perbaikan terhadap kesalahan yang terjadi.
5. Jika pada pengujian tidak dijumpai kesalahan lagi maka aplikasi tersebut akan dikompilasikan menjadi satu paket yang siap diinstalasi ke dalam komputer pemakai.
6. Menganalisa kecepatan proses untuk masing – masing modus operasi.

2.1. Perancangan Sistem

2.1.1. Algoritma

Algoritma Skipjack mengenkripsi *plaintext* 64 bit menjadi ciphertext 64 bit dengan jumlah putaran sebanyak 32 putaran yang menggunakan kunci rahasia yang berukuran 80 bit. Modus enkripsi dan dekripsi dari metode Skipjack mirip dengan metode DES yakni : *Electronic Code Book Mode*, *Cipher Block Chaining Mode*, *Cipher FeedbackMode*, dan *Output FeedbackMode*. Proses enkripsi dalam metode Skipjack terhadap suatu blok data dilakukan dengan menggunakan dua buah *rule* secara bergantian yakni *rule A* dan *rule B*. Sedangkan pada proses dekripsi, *rule* yang digunakan merupakan kebalikan (*inverse*) dari *rule A* dan *rule B* yakni *rule* A^{-1} dan *rule* B^{-1}. Operasi matematik kriptografi yang digunakan di dalam *rule A, rule B, rule* A^{-1}, dan *rule* B^{-1} tersebut adalah XOR dan permutasi. Operasi permutasi dilakukan dengan menggunakan sebuah tabel substitusi yang disebut dengan *F-Table* (Tabel 2.1) dan kunci rahasia. Nilai – nilai dalam *F-Table* diberikan dalam nilai Heksadesimal. Baris pertama pada tabel (x0…xF) menunjukkan kolom sedangkan kolom pertama pada tabel (0x…Fx) menunjukkan baris. Sebagai contoh F(7A) = D6. Nilai – nilai *F-Table* adalah sebagai berikut :

Tabel 1 : *F-Table*

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	A3	d7	09	83	f8	48	f6	f4	b3	21	15	78	99	b1	af	f9
1x	e7	2d	4d	8a	ce	4c	ca	2e	52	95	D9	1e	4e	38	44	28
2x	0a	df	02	a0	17	f1	60	68	12	b7	7a	c3	e9	fa	3d	53
3x	96	84	6b	ba	f2	63	9a	19	7c	ae	e5	f5	f7	16	6a	a2
4x	39	b6	7b	0f	c1	93	81	1b	ee	b4	1a	ea	d0	91	2f	b8
5x	55	b9	da	85	3f	41	bf	e0	5a	58	80	5f	66	0b	d8	90
6x	35	d5	c0	a7	33	06	65	69	45	00	94	56	6d	98	9b	76
7x	97	fc	b2	c2	b0	fe	db	20	e1	eb	D6	e4	dd	47	4a	1d
8x	42	ed	9e	6e	49	3c	cd	43	27	d2	07	d4	de	c7	67	18
9x	89	cb	30	1f	8d	c6	8f	aa	c8	74	dc	c9	5d	5c	31	a4
Ax	70	88	61	2c	9f	0d	2b	87	50	82	54	64	26	7d	03	40
Bx	34	4b	1c	73	d1	c4	fd	3b	cc	fb	7f	ab	e6	3e	5b	a5
Cx	Ad	04	23	9c	14	51	22	f0	29	79	71	7e	ff	8c	0e	e2
Dx	0c	ef	bc	72	75	6f	37	a1	ec	d3	8e	62	8b	86	10	e8
Ex	08	77	11	be	92	4f	24	c5	32	36	9d	cf	f3	a6	bb	ac
Fx	5e	6c	a9	13	57	25	b5	e3	bd	a8	3a	01	05	59	2a	46

2.1.2. Algoritma Pengolahan Kunci

Kunci rahasia dalam algoritma Skipjack memiliki panjang 80 bit. Pengolahan kunci yang dilakukan dalam metode Skipjack sangatlah sederhana karena suatu kunci rahasia hanya dibagi – bagi menjadi 10 buah subkunci dengan panjang masing – masing 8 bit yang akan digunakan dalam proses enkripsi dan dekripsi. Subkunci tersebut disebut dengan *cryptovariable* yang dinyatakan sebagai cv_0, cv_1, cv_2, cv_3, cv_4, cv_5, cv_6, cv_7, cv_8, dan cv_9.

Adapun algoritma pengolahan kunci adalah sebagai berikut :

1. Ubah kunci rahasia ke dalam bentuk heksadesimal.
2. Bagi kunci rahasia tersebut menjadi 10 bagian yang masing – masing berukuran 8 bit sebagai berikut:
   cv_0 : bit 1 sampai bit 8 ; cv_1 : bit 9 sampai bit 16
   cv_2 : bit 17 sampai bit 24 ; cv_3 : bit 25 sampai bit 32
   cv_4 : bit 33 sampai bit 40 ; cv_5 : bit 41 sampai bit 48
   cv_6 : bit 49 sampai bit 56 ; cv_7 : bit 57 sampai bit 64
   cv_8 : bit 65 sampai bit 72 ; cv_9 : bit 73 sampai bit 80
   Catatan : Bit 1 dimulai dari posisi bit paling tinggi (MSB, *Most Significant Bit*).

2.1.3. Algoritma Permutasi

Fungsi permutasi pada metode Skipjack disebut dengan permutasi G yang merupakan 4 *round* dari struktur *Feistel*. Fungsi *round* tersebut merupakan tabel substitusi *byte* yang *fixed*, yang dinamakan *F-Table* (Tabel 2.1). Masing-masing *round* dari permutasi G juga memasukkan sebuah *cryptovariable*. Permutasi G dilakukan pada proses enkripsi di awal setiap *rule* yakni *rule A dan rule B*. Sedangkan pada proses dekripsi, permutasi yang dilakukan merupakan kebalikan (*inverse*) dari permutasi G yang disebut dengan permutasi G^{-1} yang dilakukan di awal setiap *rule* A^{-1} dan *rule* B^{-1}. Sebagai masukan (*input*) untuk melakukan proses permutasi adalah seperempat bagian dari blok *plaintext* ataupun blok *ciphertext* dalam bentuk heksadesimal yang berukuran 16 bit.

Berikut ini adalah langkah – langkah dari permutasi G dan permutasi G^{-1} :

1. Untuk permutasi G, G(*Word = g1 || g2*) = $g_5 \parallel g_6$ di mana g_1 adalah byte pertama dari *Word* (*high byte*) dan g_2 adalah byte kedua dari *Word* (*low byte*) dan sebagai hasilnya (*output*) adalah gabungan antara g_5 dengan g_6. Untuk g_3, g_4, g_5, dan g_6 , rumus yang berlaku adalah sebagai berikut :

   $$g_i = F(g_{i-1} \oplus cv_{4k+i-3}) \oplus g_{i-2} ,$$

   di mana $3 \le i \le 6$ (i awal = 3), k pada proses enkripsi putaran pertama adalah 0, F merupakan tabel substitusi atau *F-Table*, dan cv_{4k+i-3} adalah *cryptovariable* dengan indeks (4k+i-3) dalam *cryptovariable schedule*. Sesuai dengan rumus : $g_i = F(g_{i-1} \oplus cv_{4k+i-3}) \oplus g_{i-2}$ maka :

   $$g_3 = F(g2 \oplus cv_{4k}) \oplus g_1 \quad ; \quad g_4 = F(g3 \oplus cv_{4k+1}) \oplus g_2$$
   $$g_5 = F(g4 \oplus cv_{4k+2}) \oplus g_3 \quad ; \quad g_6 = F(g5 \oplus cv_{4k+3}) \oplus g_4$$

2. Untuk permutasi G^{-1}, G^{-1}(*Word* = $g_5 \parallel g_6$) = $g_1 \parallel g_2$di mana g_5 adalah byte pertama dari *Word* (*high byte*) dan g_6 adalah byte kedua dari *Word* (*low byte*) dan sebagai hasilnya (*output*) adalah gabungan antara g_1 dengan g_2. Untuk g_4, g_3, g_2, g_1 , rumus yang berlaku adalah sebagai berikut :

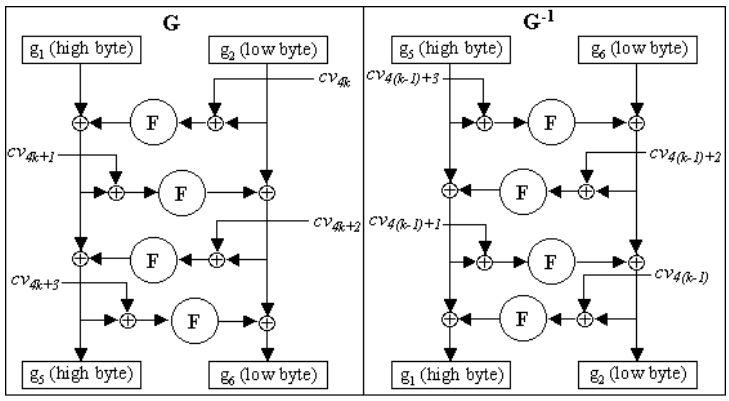
   $$g_{i-2} = F(g_{i-1} \oplus cv_{4(k-1)+i-3}) \oplus g_i ,$$

   di mana: $3 \le i \le 6$, (i awal = 6), k pada proses dekripsi putaran pertama adalah 32, F merupakan tabel substitusi atau *F-Table*, dan $cv_{4(k-1)+i-3}$ adalah c*ryptovariable* dengan indeks (4(k-1)+i-3) dalam *cryptovariable schedule*. Sesuai dengan rumus : $g_{i-2} = F(g_{i-1} \oplus cv_{4(k-1)+i-3}) \oplus g_i$maka :

   $$g_4 = F(g_5 \oplus cv_{4(k-1)+3}) \oplus g_6 \quad ; \quad g_3 = F(g_4 \oplus cv_{4(k-1)+2}) \oplus g_5$$
   $$g_2 = F(g_3 \oplus cv_{4(k-1)+1}) \oplus g_4 \quad ; \quad g_1 = F(g_2 \oplus cv_{4(k-1)}) \oplus g_3$$

   Panjang *cryptovariable* adalah 10 bytes dengan label $cv_0 \ldots cv_{10}$maka *schedule* yang diberikan pada defenisi permutasi G dan permutasi G^{-1} selalu dibuat modulo 10. Berikut ini adalah diagram

permutasi G dan permutasi G^{-1} :



Gambar 1 : Diagram Permutasi G dan Permutasi G^{-1}

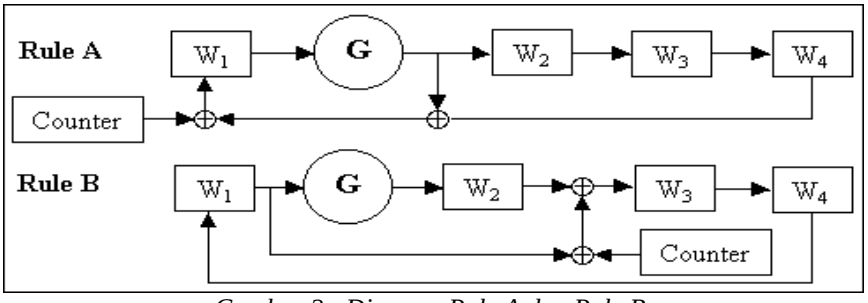
2.1.4. Algoritma Enkripsi Metode Skipjack

Pada metode Skipjack, sebuah blok *plaintext* yang hendak dienkripsi terlebih dahulu akan dikonversikan ke dalam bentuk heksadesimal. Nilai heksadesimal tersebut merupakan nilai ASCII (*American Standard Code for Information Interchange*) dari masing - masing karakter yang ada dalam blok *plaintext* tersebut. Setelah proses konversi dilakukan, blok *plaintext* yang sudah dalam bentuk heksadesimal tersebut akan dibagi menjadi 4 bagian yang disebut dengan *Word* yang dinyatakan dengan W_1^0, W_2^0, W_3^0, W_4^0 dengan masing-masing *Word* berukuran 16 bit. 4-*Word* blok data tersebut akan dienkripsi secara bergantian menggunakan dua buah *rule* yaitu *ruleA* dan *ruleB* sebanyak 32 putaran yakni : 8 putaran pertama dilakukan dengan *rule A*, kemudian 8 putaran kedua dilakukan dengan *rule B*, lalu 8 putaran ketiga dilakukan dengan *rule A* dan 8 putaran terakhir dengan *rule B*. *Ciphertext* adalah W_1^{32}, W_2^{32}, W_3^{32}, W_4^{32}. Terdapat dua variabel penting dalam proses enkripsi yaitu *counter* dan *k* dimana pada awal putaran pertama, *counter* = 1 dan *k* = 0.

Langkah – langkah dari *ruleA* adalah sebagai berikut :
1. Lakukan permutasi G dengan *input*W_1^k.
2. W_1^{k+1} merupakan hasil dari operasi XOR antara *output* permutasi G, W_4^k, dan *counter*.
3. W_2^{k+1} merupakan *output* dari permutasi G.
4. $W_3^{k+1} = W_2^k$.
5. $W_4^{k+1} = W_3^k$.
6. *counter* dan *k* ditambah satu.

Langkah – langkah dari *rule B* adalah sebagai berikut :
1. Lakukan permutasi G dengan *input*W_1^k.
2. $W_1^{k+1} = W_4^k$.
3. W_2^{k+1} merupakan *output* dari permutasi G.
4. W_3^{k+1} merupakan hasil dari operasi XORantara W_1^k, W_2^k, dan *counter*.
5. $W_4^{k+1} = W_3^k$.
6. *counter* dan *k* ditambah satu.

Berikut ini adalah diagram dari *rule A* dan *rule B* :



Gambar 2 : Diagram Rule A dan Rule B.

2.1.5. Algoritma Dekripsi Metode Skipjack

Sama halnya dengan proses enkripsi, sebuah blok *ciphertext* yang hendak didekripsi terlebih dahulu akan dikonversikan ke dalam bentuk heksadesimal sesuai dengan nilai ASCII dari masing - masing karakter yang ada dalam blok *ciphertext* tersebut. Setelah proses konversi dilakukan, blok *ciphertext* tersebut akan dibagi menjadi 4 bagian yang disebut dengan *Word* yang dinyatakan dengan W_1^{32}, W_2^{32}, W_3^{32}, W_4^{32} dengan masing-masing *Word* berukuran 16 bit. 4-*Word* blok data tersebut akan didekripsi secara bergantian menggunakan dua buah *rule* yaitu *rule*A^{-1} dan *rule*B^{-1} sebanyak 32 putaran yakni : 8 putaran pertama dilakukan dengan *rule* B^{-1}, kemudian 8 putaran kedua dilakukan dengan *rule* A^{-1}, lalu 8 putaran ketiga dilakukan dengan *rule* B^{-1} dan 8 putaran terakhir dengan *rule* A^{-1}. *Plaintext* adalah W_1^0, W_2^0,W_3^0, W_4^0. Terdapat dua variabel penting dalam proses dekripsi yaitu *counter* dan *k* dimana pada awal putaran pertama, *counter* = 32 dan *k* = 32.

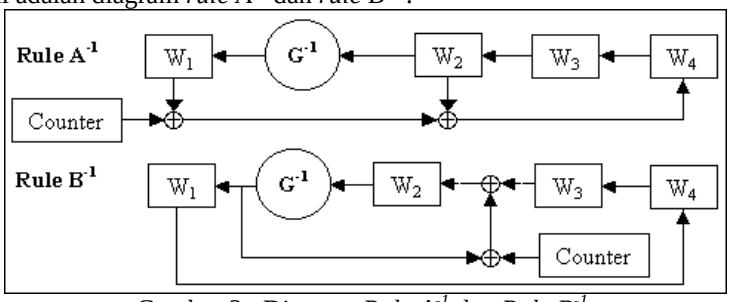
Langkah – langkah dari *rule*A^{-1} adalah sebagai berikut :

1. Lakukan permutasi G^{-1} dengan *input*W_2^k.
2. W_1^{k-1} merupakan *output* dari permutasi G^{-1}.
3. $W_2^{k-1} = W_3^k$.
4. $W_3^{k-1} = W_4^k$.
5. W_4^{k-1}= merupakan hasil dari operasi XOR antara W_1^k, W_2^k dan *counter*.
6. *counter* dan *k* dikurangi satu.

Langkah – langkah dari *rule* B^{-1} adalah sebagai berikut :

1. Lakukan permutasi G^{-1} dengan *input*W_2^k.
2. W_1^{k-1} merupakan output dari permutasi G^{-1}.
3. W_2^{k-1} merupakan hasil dari operasi XOR antara *output* permutasi G^{-1}, W_3^k, dan *counter*.
4. $W_3^{k-1} = W_4^k$.
5. $W_4^{k-1} = W_1^k$.
6. *counter* dan *k* dikurangi satu.

Berikut ini adalah diagram *rule* A^{-1} dan *rule* B^{-1} :



Gambar 3 : Diagram *Rule* A^{-1} dan *Rule* B^{-1}.

3. Pembahasan dan Hasil

3.1. Pembahasan

3.1.1. Pengolahan Kunci

Proses pengolahan kunci pada metode Skipjack adalah proses yang dilakukan sebelum melakukan proses enkripsi maupun dekripsi. Berikut ini adalah contoh pelaksanaan proses pengolahan kunci yang bertujuan untuk menghasilkan 10 buah subkunci (*cryptovariable*).

Kunci = "*CRYPTOLOGY*"

Kunci dalam bentuk heksadesimal = 43525950544F4C4F4759

Bagi kunci menjadi 10 subkunci (*cryptovariable*) masing – masing 8 bit sebagai berikut :

cv[0] = 43;cv[1] = 52;cv[2] = 59;cv[3] = 50;cv[4] = 54
cv[5] = 4F; cv[6] = 4C; cv[7] = 4F; cv[8] = 47;cv[9] = 59

3.1.2. Proses Permutasi

Permutasi G dilakukan pada saat proses enkripsi yaitu pada awal setiap pelaksanaan *rule A* dan

rule B. Contoh dari pelaksanaan permutasi G tersebut dengan *input* berupa suatu string dengan panjang 16 bit dalam bentuk heksadesimal adalah sebagai berikut (contoh *cryptovariable* yang digunakan adalah seperti pada contoh pengolahan kunci sebelumnya) :

Input : 434F ; K = 0
g1 = Mid(*Input*,1,2) = 43
g2 = Mid(*Input*,3,2) = 4F
g3 = F(g2 ⊕ cv[(4*k)mod 10]) ⊕ g1 = DA
cv[(4*0)mod 10] = cv[0] = 43
g2 = 4F : 0100 1111
cv[0] = 43 : 0100 0011 ⊕
0000 1100
F(0000 1100) = F(0C) = 99
F(0C) = 99 : 1001 1001
g1 = 43 : 0100 0011 ⊕
g3 = 1101 1010 (DA)
g4 = F(g3 ⊕ cv[((4*k)+1)mod 10]) ⊕ g2 = 68
cv[((4*0)+1)mod 10] = cv[1] = 52
g3 = DA : 1101 1010
cv[1] = 52 : 0101 0010 ⊕
1000 1000
F(1000 1000) = F(88) = 27
F(88) = 27 : 0010 0111
g2 = 4F : 0100 1111 ⊕
g4 = 0110 1000 (68)
g5 = F(g4 ⊕ cv[((4*k)+2)mod 10]) ⊕ g3 = 5E
cv[((4*0)+2)mod 10] = cv[2] = 59
g4 = 68 : 0110 1000
cv[2] = 59 : 0101 1001 ⊕
0011 0001
F(0011 0001) = F(31) = 84
F(31) = 84 : 1000 0100
g3 = DA : 1101 1010 ⊕
g5 = 0101 1110 (5E)
g6 = F(g5 ⊕ cv[((4*k)+3)mod 10]) ⊕ g4 = C7
cv[((4*0)+3)mod 10] = cv[3] = 50
g5 = 5E : 0101 1110
cv[3] = 50 : 0101 0000 ⊕
0000 1110
F(0000 1110) = F(0E) = AF
F(0E) = AF : 1010 1111
g4 = 68 : 0110 1000 ⊕
g6 = 1100 0111 (C7)

G = g5 + g6 = 5EC7

Permutasi G^{-1} merupakan kebalikan (*inverse*) dari permutasi G yang dilakukan pada saat proses dekripsi yaitu pada awal setiap pelaksanaan *rule* A^{-1} dan *rule* B^{-1}. Contoh dari pelaksanaan permutasi G^{-1} tersebut dengan *input* berupa suatu string dengan panjang 16 bit dalam bentuk heksadesimal adalah sebagai berikut (contoh *cryptovariable* yang digunakan adalah seperti pada contoh pengolahan kunci sebelumnya) :

Input = 31EE ; k = 32
g5 = Mid(*Input*,1,2) = 31
g6 = Mid(*Input*,3,2) = EE
g4 = F(g5 ⊕ cv[(4*(k-1)+3)mod 10]) ⊕ g6 = A4
cv[(4*(32-1)+3)mod 10] = cv[7] = 4F
g5 = 31 : 0011 0001
cv[7] = 4F : 0100 1111 ⊕
0111 1110
F(0111 1110) = F(7E) = 4A
F(7E) = 4A : 0100 1010
g6 = EE : 1110 1110 ⊕
g4 = 1010 0100 (A4)
g3 = F(g4 ⊕ cv[(4*(k-1)+2)mod 10]) ⊕ g5 = 03
cv[(4*(32-1)+2)mod 10] = cv[6] = 4C
g4 = A4 : 1010 0100
cv[6] = 4C : 0100 1100 ⊕
1110 1000
F(1110 1000) = F(E8) = 32
F(E8) = 32 : 0011 0010

g5 = 31 : 0011 0001 ⊕
g3 = 0000 0011 (03)
g2 = F(g3 ⊕ cv[(4*(k-1)+1)mod 10]) ⊕ g4 = 74
cv[(4*(32-1)+1)mod 10] = cv[5] = 4F
g3 = 03 : 0000 0011
cv[5] = 4F : 0100 1111 ⊕
0100 1100
F(0100 1100) = F(4C) = D0
F(4C) = D0 : 1101 0000
g4 = A4 : 1010 0100 ⊕
g2 = 0111 0100 (74)
g1 = F(g2 ⊕ cv[(4*(k-1))mod 10]) ⊕ g3 = 09
cv[(4*(32-1))mod 10] = cv[4] = 54
g2 = 74 : 0111 0100
cv[4] = 54 : 0101 0100 ⊕
0010 0000
F(0010 0000) = F(20) = 0A
F(20) = 0A : 0000 1010
g3 = 03 : 0000 0011 ⊕
g1 = 0000 1001 (09)

G^{-1} = g1 + g2 = 0974

3.1.3. Proses Enkripsi

Proses enkripsi dalam metode Skipjack memiliki 32 putaran dengan menggunakan 10 buah subkunci yang merupakan hasil pembagian dari sebuah kunci rahasia. Berikut ini adalah contoh proses enkripsi metode Skipjack :
*Plaintext =**COMPUTER***
Kunci =***CRYPTOLOGY***

Pengolahan Kunci :
Ubah kunci ke dalam bentuk heksadesimal :43525950544F4C4F4759
Bagi *key* menjadi 10 bagian sebagai berikut :
cv(0) = 43; cv(1) = 52; cv(2) = 59; cv(3) = 50; cv(4) = 54
cv(5) = 4F; cv(6) = 4C; cv(7) = 4F; cv(8) = 47; cv(9) = 59

Proses Enkripsi :
Ubah *plaintext* ke dalam bentuk heksadesimal : 434F4D5055544552
Bagi *plaintext* menjadi 4 bagian(W1,W2,W3,W4) sebagai berikut :
W1(0) = 434F; W2(0) = 4D50; W3(0) = 5554; W4(0) = 4552

Putaran ke-1 (Rule A, K = 0, Counter = 1)
G(W1(0)) = G(434F) = g5 + g6 = 5EC7
g1 = Mid(W1(0),1,2) = 43
g2 = Mid(W1(0),3,2) = 4F
g3 = F(g2 ⊕ cv[(4*k) mod 10]) ⊕ g1 = DA
cv[(4*0)mod 10] = cv[0] = 43
g2 = 4F : 0100 1111
cv[0] = 43 : 0100 0011 ⊕
0000 1100
F(0000 1100) = F(0C) = 99
F(0C) = 99 : 1001 1001
g1 = 43 : 0100 0011 ⊕
g3 = 1101 1010 (DA)
g4 = F(g3 ⊕ cv[((4*k)+1)mod 10]) ⊕ g2 = 68
cv[((4*0)+1)mod 10] = cv[1] = 52
g3 = DA : 1101 1010
cv[1] = 52 : 0101 0010 ⊕
1000 1000
F(1000 1000) = F(88) = 27
F(88) = 27 : 0010 0111
g2 = 4F : 0100 1111 ⊕
g4 = 0110 1000 (68)
g5 = F(g4 ⊕ cv[((4*k)+2)mod 10]) ⊕ g3 = 5E

cv[((4*0)+2)mod 10] = cv[2] = 59
g4 = 68 : 0110 1000
cv[2] = 59 : 0101 1001 ⊕
0011 0001
F(0011 0001) = F(31) = 84
F(31) = 84 : 1000 0100
g3 = DA : 1101 1010 ⊕
g5 = 0101 1110 (5E)
g6 = F(g5 ⊕ cv[((4*k)+3)mod 10]) ⊕ g4 = C7
cv[((4*0)+3)mod 10] = cv[3] = 50
g5 = 5E : 0101 1110
cv[3] = 50 : 0101 0000 ⊕
0000 1110
F(0000 1110) = F(0E) = AF
F(0E) = AF : 1010 1111
g4 = 68 : 0110 1000 ⊕
g6 = 1100 0111 (C7)
W1(1) = G(W1(0)) ⊕ W4(0) ⊕ Counter = 1B94
G(W1(0)) = 5EC7 : 0101 1110 1100 0111
W4(0) = 4552 : 0100 0101 0101 0010 ⊕
0001 1011 1001 0101
Counter = 1 : 0000 0000 0000 0001 ⊕
0001 1011 1001 0100 (1B94)
W2(1) = G(W1(0)) = G(434F) = 5EC7
W3(1) = W2(0) = 4D50
W4(1) = W3(0) = 5554
K = 1 ; Counter = 2
Ciphertext : W1(1) + W2(1) + W3(1) + W4(1) = 1B94 5EC7 4D50 5554

Putaran ke-2 (Rule A, K = 1, Counter = 2)

G(W1(1)) = G(1B94) = g5 + g6 = 21A7
g1 = Mid(W1(1),1,2) = 1B
g2 = Mid(W1(1),3,2) = 94
g3 = F(g2 ⊕ cv[(4*k) mod 10]) ⊕ g1 = B6
cv[(4*1)mod 10] = cv[4] = 54
g2 = 94 : 1001 0100
cv[4] = 54 : 0101 0100 ⊕
1100 0000
F(1100 0000) = F(C0) = AD
F(C0) = AD : 1010 1101
g1 = 1B : 0001 1011 ⊕
g3 = 1011 0110 (B6)
g4 = F(g3 ⊕ cv[((4*k)+1)mod 10]) ⊕ g2 = 3C
cv[((4*1)+1)mod 10] = cv[5] = 4F
g3 = B6 : 1011 0110
cv[5] = 4F : 0100 1111 ⊕
1111 1001
F(1111 1001) = F(F9) = A8
F(F9) = A8 : 1010 1000
g2 = 94 : 1001 0100 ⊕
g4 = 0011 1100 (3C)
g5 = F(g4 ⊕ cv[((4*k)+2)mod 10]) ⊕ g3 = 21
cv[((4*1)+2)mod 10] = cv[6] = 4C
g4 = 3C : 0011 1100
cv[6] = 4C : 0100 1100 ⊕
0111 0000
F(0111 0000) = F(70) = 97
F(70) = 97 : 1001 0111
g3 = B6 : 1011 0110 ⊕
g5 = 0010 0001 (21)
g6 = F(g5 ⊕ cv[((4*k)+3)mod 10]) ⊕ g4 = A7
cv[((4*1)+3)mod 10] = cv[7] = 4F
g5 = 21 : 0010 0001
cv[7] = 4F : 0100 1111 ⊕
0110 1110
F(0110 1110) = F(6E) = 9B
F(6E) = 9B : 1001 1011
g4 = 3C : 0011 1100 ⊕
g6 = 1010 0111 (A7)
W1(2) = G(W1(1)) ⊕ W4(1) ⊕ Counter = 74F1
G(W1(1)) = 21A7 : 0010 0001 1010 0111

W4(1) = 5554 : 0101 0101 0101 0100 ⊕
0111 0100 1111 0011
Counter = 2 : 0000 0000 0000 0010 ⊕
0111 0100 1111 0001 (74F1)
W2(2) = G(W1(1)) = G(1B94) = 21A7
W3(2) = W2(1) = 5EC7
W4(2) = W3(1) = 4D50
K = 2 ; Counter = 3
Ciphertext : W1(2) + W2(2) + W3(2) + W4(2) = 74F1 21A7 5EC7 4D50
Dan hasil akhir pada putaran ke-32 adalah sebagai berikut :**8AF0 31EE 1104 C2C2 (Šð1î□□ÂÂ)**.

3.1.4. Proses Dekripsi

Proses dekripsi merupakan kebalikan dari proses enkripsi yang mentransformasikan *ciphertext* menjadi *plaintext*. Berikut ini adalah contoh proses dekripsi :
Ciphertext =**Šð1î□□ÂÂ**
Kunci = ***CRYPTOLOGY***

Pengolahan Kunci :
Ubah kunci ke dalam bentuk heksadesimal :43525950544F4C4F4759
Bagi *key* menjadi 10 bagian sebagai berikut :
cv(0) = 43; cv(1) = 52; cv(2) = 59; cv(3) = 50; cv(4) = 54
cv(5) = 4F; cv(6) = 4C; cv(7) = 4F; cv(8) = 47; cv(9) = 59

Proses Dekripsi :
Ubah *Ciphertext* ke dalam bentuk heksadesimal :8AF031EE1104C2C2
Bagi *Ciphertext* menjadi 4 bagian(W1,W2,W3,W4) sebagai berikut :
W1(32) = 8AF0; W2(32) = 31EE; W3(32) = 1104; W4(32) = C2C2

Putaran ke-1 (Rule B^{-1}, K = 32, Counter = 32)
G^{-1}(W2(32)) = G^{-1}(31EE) = g1 + g2 = 0974
g5 = Mid(W2(32),1,2) = 31
g6 = Mid(W2(32),3,2) = EE
g4 = F(g5 ⊕ cv[(4*(k-1)+3)mod 10]) ⊕ g6 = A4
cv[(4*(32-1)+3)mod 10] = cv[7] = 4F
g5 = 31 : 0011 0001
cv[7] = 4F : 0100 1111 ⊕
0111 1110
F(0111 1110) = F(7E) = 4A
F(7E) = 4A : 0100 1010
g6 = EE : 1110 1110 ⊕
g4 = 1010 0100 (A4)
g3 = F(g4 ⊕ cv[(4*(k-1)+2)mod 10]) ⊕ g5 = 03
cv[(4*(32-1)+2)mod 10] = cv[6] = 4C
g4 = A4 : 1010 0100
cv[6] = 4C : 0100 1100 ⊕
1110 1000
F(1110 1000) = F(E8) = 32
F(E8) = 32 : 0011 0010
g5 = 31 : 0011 0001 ⊕
g3 = 0000 0011 (03)
g2 = F(g3 ⊕ cv[(4*(k-1)+1)mod 10]) ⊕ g4 = 74
cv[(4*(32-1)+1)mod 10] = cv[5] = 4F
g3 = 03 : 0000 0011
cv[5] = 4F : 0100 1111 ⊕
0100 1100
F(0100 1100) = F(4C) = D0
F(4C) = D0 : 1101 0000
g4 = A4 : 1010 0100 ⊕
g2 = 0111 0100 (74)
g1 = F(g2 ⊕ cv[(4*(k-1))mod 10]) ⊕ g3 = 09
cv[(4*(32-1))mod 10] = cv[4] = 54
g2 = 74 : 0111 0100
cv[4] = 54 : 0101 0100 ⊕
0010 0000

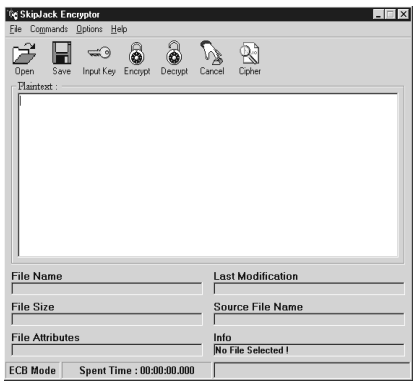
F(0010 0000) = F(20) = 0A
F(20) = 0A : 0000 1010
g3 = 03 : 0000 0011 ⊕
g1 = 0000 1001 (09)
W1(31) = G^{-1}(W2(32)) = G^{-1}(31EE) = 0974
W2(31) = G^{-1}(W2(32)) ⊕ W3(32) ⊕ Counter = 1850
G^{-1}(W2(32))= 0974 : 0000 1001 0111 0100
W3(32) = 1104 : 0001 0001 0000 0100 ⊕
0001 1000 0111 0000
Counter = 32 : 0000 0000 0010 0000 ⊕
0001 1000 0101 0000 (1850)
W3(31) = W4(32) = C2C2
W4(31) = W1(32) = 8AF0
K = 31 ; Counter = 31
Ciphertext : W1(31) + W2(31) + W3(31) + W4(31) = 0974 1850 C2C2 8AF0

Putaran ke-2 (Rule B^{-1}, K = 31, Counter = 31)
G^{-1}(W2(31)) = G^{-1}(1850) = g1 + g2 = 92A6
g5 = Mid(W2(31),1,2) = 18
g6 = Mid(W2(31),3,2) = 50
g4 = F(g5 ⊕ cv[(4*(k-1)+3)mod 10]) ⊕ g6 = BE
cv[(4*(31-1)+3)mod 10] = cv[3] = 50
g5 = 18 : 0001 1000
cv[3] = 50 : 0101 0000 ⊕
0100 1000
F(0100 1000) = F(48) = EE
F(48) = EE : 1110 1110
g6 = 50 : 0101 0000 ⊕
g4 = 1011 1110 (BE)
g3 = F(g4 ⊕ cv[(4*(k-1)+2)mod 10]) ⊕ g5 = DD
cv[(4*(31-1)+2)mod 10] = cv[2] = 59
g4 = BE : 1011 1110
cv[2] = 59 : 0101 1001 ⊕
1110 0111
F(1110 0111) = F(E7) = C5

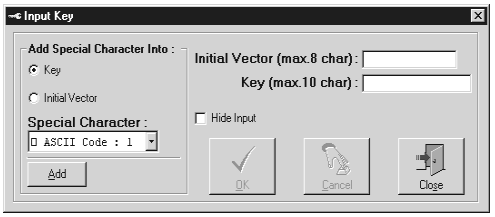
F(E7) = C5 : 1100 0101
g5 = 18 : 0001 1000 ⊕
g3 = 1101 1101 (DD)
g2 = F(g3 ⊕ cv[(4*(k-1)+1)mod 10]) ⊕ g4 = A6
cv[(4*(31-1)+1)mod 10] = cv[1] = 52
g3 = DD : 1101 1101
cv[1] = 52 : 0101 0010 ⊕
1000 1111
F(1000 1111) = F(8F) = 18
F(8F) = 18 : 0001 1000
g4 = BE : 1011 1110 ⊕
g2 = 1010 0110 (A6)
g1 = F(g2 ⊕ cv[(4*(k-1))mod 10]) ⊕ g3 = 92
cv[(4*(31-1))mod 10] = cv[0] = 43
g2 = A6 : 1010 0110
cv[0] = 43 : 0100 0011 ⊕
1110 0101
F(1110 0101) = F(E5) = 4F
F(E5) = 4F : 0100 1111
g3 = DD : 1101 1101 ⊕
g1 = 1001 0010 (92)
W1(30) = G^{-1}(W2(31)) = G^{-1}(1850) = 92A6
W2(30) = G^{-1}(W2(31)) ⊕ W3(31) ⊕ Counter = 507B
G^{-1}(W2(31)) = 92A6 : 1001 0010 1010 0110
W3(31) = C2C2 : 1100 0010 1100 0010 ⊕
0101 0000 0110 0100
Counter = 31 : 0000 0000 0001 1111 ⊕
0101 0000 0111 1011 (507B)
W3(30) = W4(31) = 8AF0
W4(30) = W1(31) = 0974
K = 30 ; Counter = 30
Ciphertext : W1(30) + W2(30) + W3(30) + W4(30) = 92A6 507B 8AF0 0974

Dan hasil akhir pada putaran ke-32 adalah sebagai berikut : **434F 4D50 5554 4552 (*COMPUTER*)**.

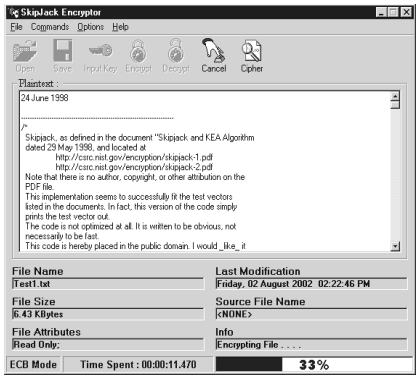
3.2. Hasil



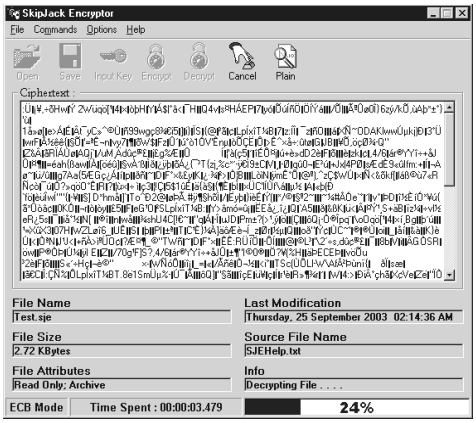
Tampilan *Form* Utama



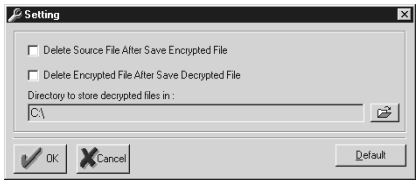
Tampilan *Form* Pengisian Kunci



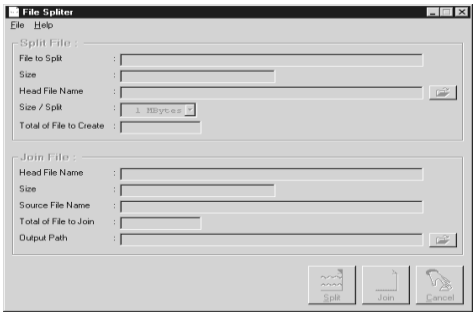
Form Utama Saat
Menjalankan Proses Enkripsi



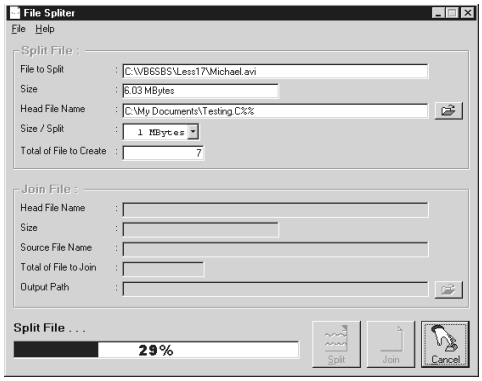
Form Utama Saat
Menjalankan Proses Dekripsi



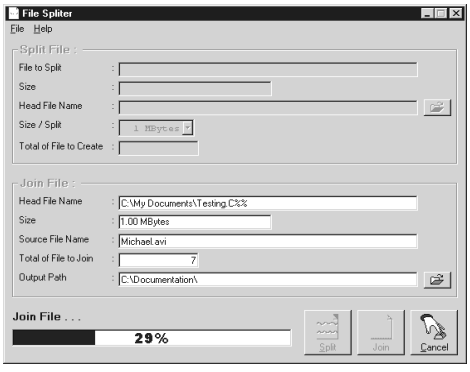
Tampilan *Form* Konfigurasi



Tampilan *FormFile Spliter*



Tampilan *FormFile Spliter*
Saat Memecah Berkas



Tampilan *FormFile Spliter*
Saat Menggabung Berkas

Gambar 4 Tampilan hasil

4. Kesimpulan

Setelah perangkat lunak pengamanan data dirancang dan dilakukan pengujian maka kesimpulan yang dapat diambil adalah sebagai berikut :

1. Pada proses permutasi G, kunci yang dimasukkan oleh pengguna dan blok *plaintext* yang akan dienkripsi akan menentukan nilai – nilai *F-Table* yang akan digunakan karena proses pemilihan nilai – nilai *F-Table* dilakukan dengan cara meng-XOR-kan bagian – bagian dari kunci dengan bagian – bagian dari suatu blok *plaintext* kemudian hasil dari operasi XOR tersebut digunakan untuk mengindeks nilai – nilai dari *F-Table*.
2. Modus operasi *Electronic Code Book* memiliki waktu proses tercepat karena algoritma Skipjack digunakan secara murni tanpa adanya tambahan operasi. Sedangkan pada 3 modus operasi lainnya diperlukan waktu proses yang lebih lama karena ada satu operasi tambahan yang harus dilakukan diluar algoritma Skipjack yakni operasi XOR.
3. Waktu proses dari setiap pengujian yang dilakukan terhadap berkas yang sama dalam suatu modus operasi selalu berbeda-beda. Hal tersebut disebabkan oleh sistem operasi Microsoft Windows yang bersifat *multitasking*, maksudnya dua atau lebih program dapat dijalankan sekaligus sehingga mungkin saja sewaktu proses enkripsi atau dekripsi sedang berjalan, ada program lain yang sedang berjalan juga tanpa sepengetahuan kita. Hal itulah yang menyebabkan perbedaan waktu proses antara suatu pengujian dengan pengujian berikutnya.
4. Modus operasi *Cipher Feedback* dan *Output Feedback* masing-masing mempunyai struktur enkripsi dan dekripsi yang sama, sehingga jika proses enkripsi atau dekripsi dilakukan lebih dari 1 kali maka salah satu dari kunci dan vektor inisialisasi harus diganti. Melakukan suatu proses lebih dari 1 kali dengan kunci dan vektor inisialisasi yang sama dalam kedua modus operasi tersebut adalah sama dengan melakukan proses kebalikan (*inverse*) terhadap proses yang telah dilakukan sebelumnya.

5. Saran

Adapun saran yang dapat diberikan adalah sebagai berikut :

1. Untuk lebih meningkatkan keamanan, suatu berkas dapat dienkripsi sebanyak 2 kali atau lebih dalam beberapa jenis modus operasi dan untuk setiap proses enkripsi dapat digunakan kunci ataupun vektor inisialisasi yang berbeda-beda.
2. Kecepatan algoritma Skipjack dapat ditingkatkan dengan mengimplementasikan algoritma tersebut dalam bahasa pemrograman yang lebih mendekati perangkat keras (*hardware*).
3. Untuk berkas dengan ukuran yang besar dapat dibelah atau dibagi dahulu menjadi beberapa berkas yang lebih kecil sebelum dienkripsi. Hal tersebut dilakukan agar proses enkripsi terhadap suatu berkas yang besar dapat dilakukan secara bertahap tanpa harus membatalkan keseluruhan proses enkripsi yang sudah dilakukan karena ada kegiatan lain yang harus dilakukan pada saat itu juga dan kegiatan tersebut dapat mengganggu atau menghentikan proses enkripsi yang sedang berjalan, sehingga pengguna tidak akan mengalami kerugian waktu yang banyak karena harus membatalkan suatu proses enkripsi yang sudah berlangsung lama.

Referensi

Buku Teks :

[1] A.Menezes, P.Van Oorschot, S.Vanstone, ***Handbook of Applied Cryptography***, CRC Press Inc, Canada, 1997

[2] Bruce Schneier, ***Applied Cryptography: Protocols, Algorithms, and Source Code in C***, Second Edition, Jon Wiley & Sons, Inc., America, 1996.

[3] Patterson, Wayne, ***Mathematical Cryptology : for Computer Scientists and Mathematicians***, Rowman&Littlefield, New Jersey, 1987

[4] Halvorson, Michael, ***Microsoft Visual Basic 6.0 Professional***, Terjemahan Adi Kurniadi, P.T. Elex Media Komputindo, Jakarta, 2000