SISTEM ENKRIPSI DAN DEKRIPSI SERTA PENYEMBUNYIAN FILE TEKS MENGGUNAKAN ALGORITMA RC4 DAN *END OF FILE (EOF)* PADA CITRA DIGITAL

Romanus Damanik

Universitas Katolik Santo Thomas Sumatera Utara, Jl Setiabudi No. 479 F Tanjungsari Medan

Email : *rdfikom@gmail.com*

ABSTRACT

Basically, confidential data needs to be stored or delivered in a certain way so as not to be known by unauthorized foreign parties. And to overcome the problem then created the science of cryptography and steganography. One example of a cryptographic algorithm is the Rivest Code 4 (RC4) algorithm, and one example of the steganographic algorithm is End Of File (EOF). Cryptographic algorithms can be combined with steganographic algorithms so that the secret messages we have are more secure than foreign interference. First of all secret messages that we have first encrypted using the Rivest Code 4 (RC4) algorithm, then the encrypted ciphertext is hidden / inserted in the image / image media with the End Of File (EOF) steganography method.

Keyword: *Kriptografi, steganografi, rivest code 4, end of file, cipherteks*

ABSTRAK

Pada dasarnya data rahasia perlu disimpan atau disampaikan melalui suatu cara tertentu agar tidak diketahui oleh pihak asing yang tidak berhak. Dan untuk mengatasi masalah tersebut maka terciptalah ilmu kriptografi dan steganografi. Salah satu contoh algoritma kriptografi adalah algoritma Rivest Code 4 (RC4), dan salah satu contoh algoritma steganografi adalah End Of File (EOF). Algoritma kriptografi dapat dikombinasikan dengan algoritma steganografi agar pesan rahasia yang kita miliki lebih terjamin keamanannya dari gangguan pihak asing. Pertama-tama pesan rahasia yang kita miliki terlebih dahulu dienkripsi dengan menggunakan algoritma Rivest Code 4 (RC4), kemudian cipherteks hasil enkripsi tersebut disembunyikan/disisipkan di dalam media gambar/citra dengan metode steganografi End Of File (EOF).

Kata Kunci : *Kriptografi, steganografi, rivest code 4, end of file, cipherteks*

PENDAHULUAN

Kriptografi merupakan ilmu sekaligus seni untuk menjaga kerahasiaan pesan dengan cara menyamarkannya menjadi bentuk tersandi yang tidak mempunyai makna. Sementara steganografi adalah seni dan ilmu untuk menyembunyikan pesan rahasia di dalam pesan lain sehingga keberadaan pesan rahasia tersebut tidak dapat diketahui. Berbeda dengan kriptografi yang merahasiakan makna pesan namun keberadaan pesan tetap ada, steganografi merahasiakan dengan menutupi atau menyembunyikan pesan. Implementasi steganografi saat ini telah menggunakan media digital sebagai media penampung atau penyembunyi pesan, salah satunya media gambar (citra digital). Steganografi menyisipkan atau menyembunyikan pesan di dalam sebuah gambar (covertext), agar pihak lain tidak menyadari keberadaan informasi yang ada di dalam gambar tersebut.
Salah satu contoh algoritma kriptografi adalah algoritma Rivest Code 4 (RC4), dan salah satu contoh algoritma steganografi adalah End Of File (EOF).

Algoritma RC4 merupakan salah satu algoritma kunci simetris berbentuk stream cipher yang memproses unit atau input data, pesan atau informasi pada satu saat. Unit atau data pada umumnya sebuah byte atau bahkan bit (byte dalam hal RC4).[1]

Teknik EOF atau *End Of File* merupakan salah satu teknik yang digunakan dalam steganografi. Teknik ini digunakan dengan cara menambahkan data atau pesan rahasia pada akhir *file*. Teknik ini dapat digunakan untuk menambahkan data yang ukurannya sesuai dengan kebutuhan. Perhitungan kasar ukuran *file* yang telah disisipkan data sama dengan ukuran *file* sebelum disisipkan data ditambah ukuran data rahasia yang telah diubah menjadi *encoding file*.[2]

Algoritma kriptografi dapat dikombinasikan dengan algoritma steganografi agar pesan rahasia yang kita miliki lebih terjamin keamanannya dari gangguan pihak asing. Pertama-tama pesan rahasia yang kita miliki terlebih dahulu dienkripsi dengan menggunakan algoritma Rivest Code 4 (RC4), kemudian cipherteks hasil enkripsi tersebut disembunyikan/disisipkan di dalam media gambar/citra dengan metode steganografi End Of File (EOF).[3]

METODE PENELITIAN

Analisis Algoritma RC4 dan End Of File (EOF)

Pada tahap ini, akan dilakukan analisis pada algoritma RC4 dalam melakukan proses enkripsi dan dekripsi pesan, serta analisis algoritma EOF dalam melakukan proses penyisipan dan ekstraksi pesan. Selain itu penulis juga akan memberikan sebuah contoh, dimana nantinya contoh tersebut akan menerangkan tahapan proses yang terjadi dalam melakukan enkripsi pesan, penyisipan, ekstraksi dan dekripsi pesan.

Analisis Algoritma RC4

Menurut (Sadikin, 2012) terdapat tiga operasi yang dilakukan pada algoritma RC4, yaitu penjadwalan kunci, enkripsi dan dekripsi. [4]
Berikut ini penulis akan menjelaskan seluruh langkah-langkah yang dilakukan pada algoritma RC4 mulai dari proses penjadwalan kunci, proses enkripsi dan juga proses dekripsi.

1. Proses Enkripsi
   Sebelum dilakukan proses enkripsi menggunakan algoritma RC4, terlebih dahulu yang harus dilakukan adalah melakukan penjadwalan kunci untuk menghasilkan *keystream*, setelah diperoleh keystream maka selanjutnya dapat dilakukan proses enkripsi dengan cara melakukan proses XOR antara *keystream* dengan plainteks.
   Langkah-langkah untuk melakukan proses penjadwalan kunci dan enkripsi pada algoritma RC4 adalah sebagai berikut:
   a. Input plainteks dan kunci.
   b. Inisialisasi array Sbox (S[0], S[1], …, S[255]), diisi dengan bilangan 0 samapi 255, sehingga array Sbox akan berbentuk S[0] = 0, S[1] = 1, …, S[255] = 255.
   c. Inisialisasi array kunci (array kunci K dengan panjang 256). Jika panjang kunci < 256 maka dilakukan padding, yaitu penambahan byte sehingga panjang kunci menjadi 256 byte.

Sehingga array kunci K berbentuk K[0], K[1], …, K[255].

d. Lakukan permutasi terhadap nilai-nilai didalam array Sbox dengan cara menukarkan isi array Sbox[i] dengan Sbox[j].
e. Bangkitkan Keystream K dengan mengambil nilai Sbox[i] dan Sbox[j] lalu dijumlahkan dan di modulo 256.
f. Kemudian lakukan operasi XOR antara Keystream dengan plainteks.

Terakhir, hasil operasi XOR diubah menjadi bilangan hexadesimal untuk menghasilkan cipherteks.

2. Proses Dekripsi

Proses dekripsi pada algoritma RC4 sama dengan proses enkripsi, dimana sebelum melakukan proses dekripsi terlebih dahulu dilakukan proses penjadwalan kunci untuk menghasilkan *keystream*, setelah diperoleh *keystream* maka selanjutnya dapat dilakukan proses dekripsi dengan cara melakukan proses XOR antara *keystream* dengan cipherteks.

Langkah-langkah untuk melakukan proses penjadwalan kunci dan dekripsi pada algoritma RC4 adalah sebagai berikut:

a. Input cipherteks dan kunci.
b. Inisialisasi array Sbox (S[0], S[1], …, S[255]), diisi dengan bilangan 0 samapi 255, sehingga array Sbox akan berbentuk S[0] = 0, S[1] = 1, …, S[255] = 255.
c. Inisialisasi array kunci (array kunci K dengan panjang 256). Jika panjang kunci < 256 maka dilakukan padding, yaitu penambahan byte sehingga panjang kunci menjadi 256 byte. Sehingga array kunci K berbentuk K[0], K[1], …, K[255].
d. Lakukan permutasi terhadap nilai-nilai didalam array Sbox dengan cara menukarkan isi array Sbox[i] dengan Sbox[j].
e. Bangkitkan Keystream K dengan mengambil nilai Sbox[i] dan Sbox[j] lalu dijumlahkan dan di modulo 256.
f. Setelah itu dilakukan operasi XOR antara Keystream dengan cipherteks.

Yang terakhir, ubah hasil operasi XOR menjadi karakter ASCII untuk menghasilkan plainteks.

Pada algoritma End Of File (EOF), pesan disisipi pada akhir file citra, sehingga ukuran citra yang disisipi akan semakin bertambah besar dari citra aslinya. Pada algoritma ini citra yang disisipi akan terlihat seperti memiliki garis hitam. Semakin banyak karakter yang disisipi maka semakin terlihat jelas garis pada akhir file citra.[5]

Berikut ini penulis akan menjelaskan seluruh langkah-langkah yang dilakukan pada algoritma steganografi End Of File (EOF), mulai dari proses penyisipan pesan dan juga proses ekstraksi pesan. [6]

1. Proses Penyisipan Pesan

Langkah-langkah dalam melakukan proses penyisipan pesan menggunakan algoritma EOF adalah sebagai berikut:

a. Input file citra bitmap yang akan disisipi (*cover image*) dan teks yang akan disisipkan kedalam citra.
b. Baca setiap piksel dari file citra mulai dari awal hingga akhir.
c. Berikan penanda "#" di awal dan di akhir teks yang akan disisipkan ke citra.
d. Setelah itu ubah pesan teks yang telah diberi penanda menjadi bilangan desimal ASCII lalu sisipkan kedalam citra.

Setelah pesan disisipkan semua, selanjutnya jadikan file citra baru (*stego image*).

2. Proses Ekstraksi Pesan

Langkah-langkah dalam melakukan proses ekstraksi pesan menggunakan algoritma EOF aalah sebagai berikut:

a. Input file citra (*stego image*) yang akan diekstraksi.
b. Baca pixel dari ujung (akhir) file citra.
c. Ambil nilai piksel citra, lalu ubah menjadi karakter.
d. Bila ketemu penanda (karakter #) lanjutkan ke piksel selanjutnya hingga ketemu karakter # lagi.

Hapus karakter penanda (#), supaya diperoleh teks yang sebenarnya.

HASIL DAN PEMBAHASAN

Analisis Algoritma End Of File (EOF)

Berikut ini penulis akan memberikan contoh proses enkripsi, penyisipan, ekstraksi dan dekripsi menggunakan algoritma RC4 dan algoritma End Of File (EOF). Pertama-tama pesan rahasia (pesan asli) di enkripsi menggunakan algoritma RC4 untuk menghasilkan cipherteks, selanjutnya cipherteks tersebut disisipkan kedalam citra menggunakan algoritma EOF sehingga diperoleh stego image. Setelah itu stego image di ekstraksi untuk memperoleh pesan yang disisip (cipherteks), terakhir cipherteks di dekripsi untuk memperoleh plainteks (pesan asli).

1. Proses Enkripsi

   Misalkan pesan yang akan dienkripsi adalah kata "RIDHO", dan kunci yang digunakan adalah "FIKOM" maka proses enkripsinya adalah sebagai berikut:

   a. Inisialisasi array Sbox

Tabel 1. Inisialisasi Array Box

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

   b. Inisialisasi array kunci (array kunci K dengan panjang 256)

Tabel 2. Inisialisasi array kunci

Iterasi- i	Key Char	Key [i]	Sbox [i]
0	F	70	0
1	I	73	1
2	K	75	2
3	O	79	3
4	M	77	4
Iterasi- i	Key Char	Key [i]	Sbox [i]
0	F	70	0
1	I	73	1
2	K	75	2
3	O	79	3
4	M	77	4
5	F	70	5
6	I	73	6
7	K	75	7
8	O	79	8
9	M	77	9

   c. Lakukan permutasi terhadap nilai-nilai didalam array Sbox dengan cara menukarkan isi array Sbox[i] dengan Sbox[j].

=====Iterasi 1 =====
i = 0
j = 0
j = (j + Sbox[i] + Key[i])Mod 256
j = (0 + 0 + 70) Mod 256 = 70
Swap (Sbox[0], Sbox[70])

=====Iterasi 2 ====
i = 1
j = 70
j = (j + Sbox[i] + Key[i])Mod 256
j = (70 + 1 + 73) Mod 256 = 144
Swap (Sbox[1], Sbox[144])

======Iterasi 3 ====
i = 2
j = 144
j = (j + Sbox[i] + Key[i])Mod 256
j = (144 + 2 + 75) Mod 256 = 221
Swap (Sbox[2], Sbox[221])

=====Iterasi 4 =====
i = 3
j = 221
j = (j + Sbox[i] + Key[i])Mod 256
j = (221 + 3 + 79) Mod 256 = 47
Swap (Sbox[3], Sbox[47])

===Iterasi 5 ====
i = 4
j = 47
j = (j + Sbox[i] + Key[i])Mod 256
j = (47 + 4 + 77) Mod 256 = 128
Swap (Sbox[4], Sbox[128])

==== Iterasi 6 ======
i = 5
j = 128
j = (j + Sbox[i] + Key[i])Mod 256
j = (128 + 5 + 70) Mod 256 = 203
Swap (Sbox[5], Sbox[203])

==== Iterasi 7 ====
i = 6
j = 203
j = (j + Sbox[i] + Key[i])Mod 256
j = (203 + 6 + 73) Mod 256 = 26
Swap (Sbox[6], Sbox[26])

==== Iterasi 8 =====
i = 7
j = 26
j = (j + Sbox[i] + Key[i])Mod 256
j = (26 + 7 + 75) Mod 256 = 108
Swap (Sbox[7], Sbox[108])

==== Iterasi 9 ====
i = 8
j = 108
j = (j + Sbox[i] + Key[i])Mod 256
j = (108 + 8 + 79) Mod 256 = 195
Swap (Sbox[8], Sbox[195])
==== Iterasi 10 ======
i = 9
j = 195
j = (j + Sbox[i] + Key[i])Mod 256
j = (195 + 9 + 77) Mod 256 = 25
Swap (Sbox[9], Sbox[25])

17	144	221	88	39	3	155	108	195	36	141	65	20	68	204	32
50	218	54	115	111	251	167	31	114	171	178	85	225	43	143	2
113	28	184	214	38	131	25	168	130	93	8	209	137	56	152	51
22	250	1	69	140	241	34	196	84	95	18	216	90	82	10	98
215	189	37	81	230	238	66	101	222	57	61	103	183	79	187	134
254	160	224	223	4	100	186	200	136	102	180	207	41	9	148	199
243	21	16	104	153	126	248	174	239	231	154	26	166	125	219	147
40	145	6	201	119	78	176	77	62	139	7	121	151	128	14	163
67	74	206	106	19	249	86	135	47	197	203	35	46	105	182	94
96	149	76	228	242	190	164	173	162	118	64	42	13	211	152	172
212	244	179	97	181	87	91	124	52	55	11	72	129	237	44	48
185	193	123	83	133	63	165	75	80	247	177	236	169	122	0	109
142	89	220	198	59	229	210	73	92	150	235	245	110	233	202	107
23	45	159	58	194	70	27	132	205	12	213	45	29	112	156	99
157	217	120	226	33	5	53	24	232	15	252	116	158	191	253	234
127	161	71	227	208	175	117	246	188	146	60	240	255	138	170	30

d. Bangkitkan Keystream K dengan mengambil nilai Sbox[i] dan Sbox[j] lalu dijumlahkan dan di modulo 256.

==== Iterasi 1 ====
i = 0
j = 0
i = (i + 1) Mod 256 = (0 + 1) Mod 256 = 1
j = (j + Sbox[i]) Mod 256 = (0 + 144) Mod 256 = 144
swap (Sbox[1],Sbox[144])
K1 = Sbox[(Sbox[1] + Sbox[144]) Mod 256]
K1 = Sbox[(96 + 144) Mod 256]
K1 = Sbox[240 Mod 256] = 127
K1 = Sbox[240] = 127

========== Iterasi 2 ==========
i = 1
j = 144
i = (i + 1) Mod 256 = (1 + 1) Mod 256 = 2
j = (j + Sbox[i]) Mod 256 = (144 + 221) Mod 256 = 109
swap (Sbox[2],Sbox[109])
K2 = Sbox[(Sbox[2] + Sbox[109]) Mod 256]
K2 = Sbox[(125 + 221) Mod 256]
K2 = Sbox[346 Mod 256] = 180
K2 = Sbox[90] = 180

========== Iterasi 3 ==========
i = 2
j = 109
i = (i + 1) Mod 256 = (2 + 1) Mod 256 = 3
j = (j + Sbox[i]) Mod 256 = (109 + 88) Mod 256 = 197
swap (Sbox[3],Sbox[197])
K3 = Sbox[(Sbox[3] + Sbox[197]) Mod 256]
K3 = Sbox[(229 + 88) Mod 256]
K3 = Sbox[317 Mod 256] = 82
K3 = Sbox[61] = 82

========== Iterasi 4 ==========
i = 3
j = 197
i = (i + 1) Mod 256 = (3 + 1) Mod 256 = 4
j = (j + Sbox[i]) Mod 256 = (197 + 39) Mod 256 = 236
swap (Sbox[4],Sbox[236])
K4 = Sbox[(Sbox[4] + Sbox[236]) Mod 256]
K4 = Sbox[(158 + 39) Mod 256]
K4 = Sbox[197 Mod 256] = 88
K4 = Sbox[197] = 88

========== Iterasi 5 ==========
i = 4
j = 236
i = (i + 1) Mod 256 = (4 + 1) Mod 256 = 5
j = (j + Sbox[i]) Mod 256 = (236 + 3) Mod 256 = 239
swap (Sbox[5],Sbox[239])
K5 = Sbox[(Sbox[5] + Sbox[239]) Mod 256]

K5 = Sbox[(234 + 3) Mod 256]
K5 = Sbox[237 Mod 256] = 191
K5 = Sbox[237] = 191

e. Lakukan operasi XOR antara Keystream dengan plainteks, kemudian hasilnya diubah menjadi bilangan hexadesimal untuk menghasilkan cipherteks.

Key Stream	Plain teks	XOR	Hexa
01111 111	01010 010	00101 101	2D
10110 100	01001 001	11111 101	FD
01010 010	01000 100	00010 110	16
01011 000	01001 000	00010 000	10
10111 111	01001 111	11110 000	F0

Dengan demikian cipherteks yang diperoleh dari proses enkripsi adalah 2DFD1610F0

1. Proses Penyisipan Pesan
Proses yang terjadi dalam melakukan penyisipan pesan kedalam citra adalah sebagai berikut:
Misalkan citra bitmap berukuran 5x5 piksel yang dijadikan cover image dalam nilai RGB adalah sebagai berikut:

200,189, 203	194,185, 146	192,170, 87	198,168, 18	211,16 2,7
201,190, 204	199,190, 151	201,179, 96	198,168, 18	209,16 0,5
193,189, 203	189,190, 192	185,190, 170	196,170, 111	206,15 7,65
190,186, 200	188,189, 191	191,196, 176	212,106, 127	24,162, 70
196,190, 190	198,180, 178	223,182, 180	232,160, 148	182,87, 67

b. Karakter "#" dalam hexadecimal adalah "23". Sehingga pesan yang akan disisip menjadi 232DFD1610F023.
c. Setelah itu ubah pesan teks yang telah diberi penanda menjadi bilangan desimal ASCII lalu sisipkan kedalam citra.

Hexa	Desimal
23	35
2D	45
FD	253
16	22
10	16
F0	240
23	35

d. Setelah pesan disisipkan semua, selanjutnya jadikan file citra baru (*stego image*)

200,189, 203	194,185, 146	192,170, 87	198,168, 18	211,16 2,7
201,190, 204	199,190, 151	201,179, 96	198,168, 18	209,16 0,5
193,189, 203	189,190, 192	185,190, 170	196,170, 111	206,15 7,65
190,186, 200	188,189, 191	191,196, 176	212,106, 127	24,162, 70
196,190, 190	198,180, 178	223,182, 180	232,160, 148	182,87, 67
35,35,35	45,45,45	253,253, 253	22,22,22	16,16,1 6
240,240, 240	35,35,35	0,0,0	0,0,0	0,0,0

1. 3. Proses Ekstraksi Pesan
Proses yang terjadi dalam melakukan ekstraksi pesan dari citra (stego image) adalah sebagai berikut:
a. Citra stego image yang akan diekstraksi adalah:

200,189, 203	194,185, 146	192,170, 87	198,168, 18	211,16 2,7
201,190, 204	199,190, 151	201,179, 96	198,168, 18	209,16 0,5
193,189, 203	189,190, 192	185,190, 170	196,170, 111	206,15 7,65
190,186, 200	188,189, 191	191,196, 176	212,106, 127	24,162, 70
196,190, 190	198,180, 178	223,182, 180	232,160, 148	182,87, 67
35,35,35	45,45,45	253,253, 253	22,22,22	16,16,1 6
240,240, 240	35,35,35	0,0,0	0,0,0	0,0,0

b. Baca nilai piksel citra dari ujung (akhir), lalu ubah menjadi karakter. Bila ketemu penanda (karakter #) lanjutkan ke piksel selanjutnya hingga ketemu karakter # lagi. Lalu hapus karakter penanda (#), supaya diperoleh teks yang sebenarnya.

Nilai 35 yang berwarna ungu merupakan nilai ASCII dari karakter penanda "#", setelah penanda dihapus maka nilai yang diperoleh

adalah 45 253 22 16 240, dan bila diubah menjadi hexadecimal maka pesan yang disisipkan adalah 2DFD1610F0

4. Proses Dekripsi

Tahapan proses dekripsi adalah sebagai berikut:

a. Karena proses dekripsi dan ekripsi pada algoritma RC4 adalah sama, maka proses ini sialisasi array Sbox, inisialisasi array Kunci, permutasi nilai Sbox dan pembankitan Keystream pada proses dekripsi juga sama dengan proses enkripsi. Dengan demikian operasi XOR antara Keysteram dan cipherteks menjadi seperti berikut:

Key Stream	Cipherteks	XOR	Desimal	Karakter
01111 111	001011 01	01010 010	82	R
10110 100	111111 01	01001 001	73	I
01010 010	000101 10	01000 100	68	D
01011 000	000100 00	01001 000	72	H
10111 111	111100 00	01001 111	79	O

Berdasarkan proses dekripsi yang terjadi, maka plainteks (pesan asli) yang diperoleh adalah kata “RIDHO”.

KESIMPULAN

Berdasarkan pembahasan dan hasil dari penelitian, maka diperoleh beberapa kesimpulan sebagai berikut:

1. Kombinasi dari algoritma RC4 dan metode EOF mampu mengamankan file teks dan memberikan keamanan ganda.
2. Ukuran file citra setelah dilakukan penyisipan (*stego image*) memiliki ukuran yang lebih besar dibandingkan dengan ukuran file citra asli (*cover image*).

DAFTAR PUSTAKA

[1] Haji Wachyu Hari, Mulyono Slamet, 2012, Implementasi RC4 Stream CipherUntuk Keamanan Basis Data, Seminar Nasional Aplikasi Teknologi Informasi 2012 (SNATI 2012) , Diakses 11 April 2016.

[2]Wasino, Rahayu Tri Puji, Setiawan, Implementasi Steganografi Teknik End Of File Dengan Enkripsi Rijndael, Seminar Nasional Teknologi Informasi danKomunikasi 2012, Diakses 26 April 2016

[3]Krisnawati, 2008, Metode Least Significant Bit (LSB) dan End Of File (EOF) Untuk Menyisipkan Teks ke Dalam Citra Grayscale, Seminar Nasional Informatika 2008 (semnasIF 2008), Diakses 12 April 2016

[4]Sadikin R, 2012, *Kriptografi Untuk Keamanan Jaringan*, Andi Offset,Yogyakarta

[5]File Teks Dengan Algoritma Rsa (*Rivest Shamir Adleman*), Seminar Nasional Teknologi Informasi dan Komunikasi Terapan 2012 (Semantik2012), Diakses 11 April 2016.

[6]Zain, Ruri Hartika, 2012, Perancangan dan Implementasi Cryptography Dengan Metode Algoritma RC4 Pada Type File Document Menggunakan Bahasa Pemrograman Visual Basic 6.0, Jurnal Momentum. Vol 12 No. 1, Diakses 11 April 2016.