

IMPLEMENTASI STEGANOGRAFI DENGAN METODE END OF FILE PADA TEKS YANG TERENKRIPSI MENGGUNAKAN BLOCK CIPHER RIVEST CODE-6 KE DALAM CITRA

Andrew Chandra¹
22094659@students.ukdw.ac.id

Willy Sudiarto Raharjo²
willysr@ti.ukdw.ac.id

Junius Karel Tampubolon³
karel@ukdw.ac.id

Abstract

Datas or informations are very important assets nowadays, that is why we have to keep it safe so it will not fall into one who is do not have rights. There are several methods to maintain the confidentiality of datas/informations, which are cryptography and steganography. By using cryptographic Rivest Code-6 algorithm then the datas/informations is converted into a form that cannot be undestood, and then steganographic End of File algorithm the datas/informations is inserted into another media so that ordinary people do not know the existence of the datas/informations. The results of this research tell that by using both algorithms the secret datas/informations can be hidden and read back properly. In addition there will be increase in file size reservoirs, where the increase in dependent on the size of the secret datas/informations and file size of the reservoir, while the length of the key to be isnerted into reservoir will remain at 16 bytes.

Keywords: *End of File, RC6*

1. Pendahuluan

Cepatnya pertumbuhan teknologi informasi saat ini membuat sangat banyak orang bergantung padanya. Hal ini membuat banyak orang melakukan komunikasi dengan media elektronik, tentunya ada juga informasi yang bersifat rahasia atau hanya ditujukan pada orang tertentu saja. Namun zaman sekarang sudah terdapat banyak ancaman di dunia maya, misalnya saja seorang hacker yang mampu mengambil data atau informasi orang lain tanpa diketahui. Hal ini menimbulkan kekhawatiran bagi pemilik informasi rahasia tersebut.

Untuk mengatasi hal tersebut, informasi ini biasanya di-enkripsi atau disamarkan menjadi informasi yang berbeda, namun informasi yang sudah dienkripsi tidak jarang menimbulkan rasa curiga bagi sekelompok orang sehingga mudah dipecahkan. Untuk itu selain informasi tersebut di-enkripsim sebaiknya informasi itu juga disembunyikan keberadaannya, salah satu teknik untuk menyembunyikan informasi ini adalah steganografi. Steganografi merupakan teknik menyembunyikan informasi di dalam informasi lainnya yang tidak bersifat rahasia. Salah satu metode untuk melakukan steganografi ini adalah End of File (EOF).

Diharapkan dengan melakukan enkripsi dengan metode Rivest Code 6 informasi rahasia akan menjadi sebuah informasi lain dan informasi ini dapat disembunyikan ke dalam sebuah image dengan metode End of File.

¹ Mahasiswa Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana

² Dosen Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana

³ Dosen Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana

2. Landasan Teori

2.1 Konsep Dasar Steganografi

Menurut Wijaya, dan Prayudi (2004) steganografi adalah ilmu pengetahuan dan seni dalam menyembunyikan komunikasi. Suatu sistem steganografi sedemikian rupa menyembunyikan isi suatu data di dalam suatu sampul media yang tidak dapat diduga oleh orang biasa sehingga tidak membangunkan suatu kecurigaan kepada orang yang melihatnya.

Ada tiga aspek yang berbeda satu dengan lainnya dalam sebuah sistem penyembunyian informasi. Hal tersebut dikatakan berbeda karena jika salah satu aspek ditingkatkan maka bisa mengakibatkan aspek lain mengalami penurunan kualitas. Ketiga aspek tersebut adalah :

1. Kapasitas

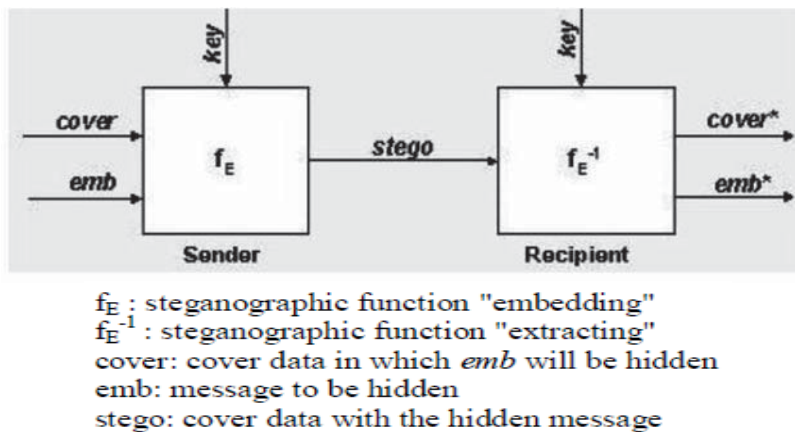
Kapasitas berarti kemampuan sistem untuk menyimpan dan mengolah seberapa banyak informasi untuk disembunyikan.

2. Keamanan

Keamanan adalah kemampuan sistem untuk mencegah agar orang lain yang tidak berhak untuk mengakses dan mendeteksi informasi tersembunyi.

3. Ketahanan

Ketahanan berarti kemampuan sistem untuk dapat bertahan dari berbagai jenis serangan yang dapat menghancurkan informasi yang disembunyikan.



Gambar 1. Model dasar embedding
(Kumar, 2008)

2.2 Algoritma End of File

Aditya, (2010) dalam jurnal yang berjudul “*Studi Pustaka untuk Steganografi dengan Beberapa Metode*” menyatakan bahwa teknik EOF atau *End Of File* merupakan salah satu teknik yang digunakan dalam steganografi. Teknik ini menggunakan cara dengan menyisipkan data pada akhir file. Teknik ini dapat digunakan untuk menyisipkan data yang ukurannya sesuai dengan kebutuhan. Ukuran file yang telah disisipkan data sama dengan ukuran file sebelum disisipkan data ditambah dengan ukuran data yang disisipkan ke dalam file tersebut. Dalam teknik ini, data disisipkan pada akhir file dengan diberi tanda khusus sebagai pengenalan start dari data tersebut dan pengenalan akhir dari data tersebut.

Kelebihan dari metode *End of File* adalah tidak ada batasan dalam menambahkan informasi yang ingin disembunyikan, bahkan jika ukuran informasi itu melebihi ukuran citra penampung. Data informasi akan disembunyikan/disisipkan di akhir *file* sehingga *file image* mungkin akan tampak ada perubahan dengan aslinya. Jika dapat dilihat mata, maka perubahan ini akan tampak di baris bawah dari *image*. Dalam metode *End of File* data yang disisipkan akan diberi penanda khusus untuk menandakan awal dan akhir dari data tersebut.

Menurut Wicaksono, (2007) metode *EOF* merupakan sebuah metode yang diadaptasi dari metode penanda akhir *file (end of file)* yang digunakan oleh sistem operasi Windows. Dalam sistem operasi Windows, jika ditemukan penanda *end of file* pada sebuah

file, maka sistem akan berhenti melakukan pembacaan pada file tersebut. Prinsip kerja EOF menggunakan karakter / simbol khusus ctrl-z yang diberikan pada setiap akhir file.

Contoh perhitungan yang dilakukan metode EOF adalah sebagai berikut. Misalkan Pada sebuah citra grayscale 6x6 piksel disisipkan pesan yang berbunyi "aku".

Kode ASCII dari pesan diberikan sebagai berikut:

97 07 117 35

Misalkan matriks tingkat derajat keabuan citra sebagai berikut :

196	10	97	182	101	40
67	200	100	50	90	50
25	150	45	200	75	28
176	56	77	100	25	200
101	34	250	40	100	60
44	66	99	125	190	200

Gambar 2. Matriks derajat keabuan citra (sebelum dilakukan steganografi)

Kode biner pesan disisipkan diakhir citra sehingga citra menjadi:

196	10	97	182	101	40
67	200	100	50	90	50
25	150	45	200	75	28
176	56	77	100	25	200
101	34	250	40	100	60
44	66	99	125	190	200
97	107	117	35		

Gambar 3. Matriks derajat keabuan citra (setelah dilakukan steganografi)

2.3 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani : "*cryptos*" yang artinya "*secret*" (rahasia), dan "*graphein*" yang artinya "*writing*" (tulisan), jadi kriptografi berarti "*secret writing*" (tulisan rahasia). Kriptografi adalah ilmu dan seni untuk menjaga keamanan dan kerahasiaan pesan dengan cara menyandikan ke dalam bentuk yang tidak dapat dimengerti lagi maknanya (Munir, 2006).

Menurut A. Menezes, P. van Oorschot and S. Vanstone (2010) kriptografi secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita. Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data.

Menurut Alfred J. Menezes, Paul C. van Oorschot, dan Scott A. Venstone (1996), ada beberapa tujuan dari kriptografi, antara lain :

- Kerahasiaan (*secrecy*) adalah layanan yang digunakan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berkepentingan. Di dalam kriptografi layanan ini direalisasikan dengan menyandikan pesan menjadi *ciphertext*.
- Integritas data (*data integrity*) adalah layanan yang menjamin bahwa pesan masih asli atau belum pernah di manipulasi selama pengiriman. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak.
- Otentikasi (*authentication*) adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication*) maupun mengidentifikasi kebenaran sumber pesan (*data origin*)

authentication). Di dalam kriptografi layanan ini direalisasikan dengan menggunakan tanda tangan digital yang menyatakan sumber pesan.

- Nirpenyangkalan (*non-repudiation*) adalah layanan untuk mencegah entitas yang berkomunikasi penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal bahwa telah menerima pesan.

Kriptografi sendiri mempunyai komponen-komponen untuk mencapai tujuan kriptografi. Menurut Ariyus (2009), pada dasarnya kriptografi terdiri dari beberapa komponen seperti :

- **Enkripsi**

Enkripsi merupakan hal yang sangat penting dalam kriptografi sebagai pengamanan atas data yang dikirimkan agar rahasianya terjaga. Pesan aslinya disebut *plaintext* yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan sebagai *cipher* atau kode. Seperti ketika kita tidak mengerti akan arti sebuah kata, kita bisa melihatnya di dalam kamus atau daftar istilah. Berbeda dengan enkripsi, untuk mengubah *plaintext* ke bentuk *ciphertext* digunakan algoritma yang bisa mengkodekan data yang diinginkan.

- **Dekripsi**

Dekripsi merupakan kebalikan dari enkripsi, pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (*plaintext*), yang disebut dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan yang digunakan untuk enkripsi.

- **Kunci (*key*)**

Kunci yang dimaksud di sini adalah kunci yang dipakai untuk melakukan proses enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, yakni kunci pribadi (*private key*) dan kunci umum (*public key*).

- ***Ciphertext***

Ciphertext merupakan suatu pesan yang sudah melalui proses enkripsi. Pesan yang ada pada *ciphertext* tidak bisa dibaca karena berisi karakter-karakter yang tidak memiliki makna (arti).

- ***Plaintext***

Plaintext sering juga disebut *cleartext*, merupakan suatu pesan bermakna yang ditulis atau diketik dan *plaintext* itulah yang akan diproses menggunakan algoritma kriptografi agar menjadi *ciphertext*.

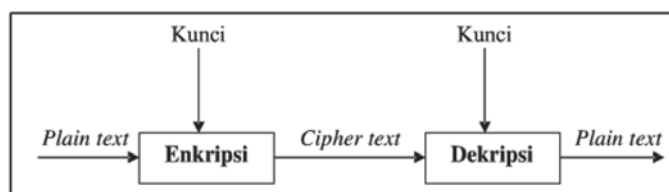
- **Pesan**

Pesan bisa berupa data atau informasi yang dikirim (melalui kurir, saluran komunikasi data, dan sebagainya) atau yang disimpan di dalam media perekaman (kertas, *storage*, dan sebagainya).

- ***Cryptanalysis***

Cryptanalysis bisa diartikan sebagai analisis sandi atau suatu ilmu untuk mendapatkan *plaintext* tanpa harus mengetahui kunci yang digunakan dalam proses enkripsi. Jika suatu *ciphertext* berhasil menjadi *plaintext* tanpa menggunakan kunci yang sah, maka proses tersebut dinamakan *breaking code* yang dilakukan oleh para *cryptanalysts*. Analisis sandi juga mampu menemukan kelemahan dari suatu algoritma kriptografi dan akhirnya bisa menemukan kunci atau *plaintext* dari *ciphertext* yang dienkripsi menggunakan algoritma tertentu.

Disamping itu kriptografi mempunyai komponen utama yaitu proses enkripsi dan dekripsi. Kriptografi membutuhkan sebuah kunci untuk mengubah *plaintext* menjadi *ciphertext* atau sebaliknya. Aspek kerahasiaan kunci sangatlah penting untuk diperhatikan karena apabila kunci tersebut diketahui oleh pihak yang tidak bersangkutan maka mereka bisa membongkar pesan yang sudah dilakukan proses enkripsi. Berikut ini adalah skema yang menggambarkan proses enkripsi dan dekripsi pada umumnya.



Gambar 4. Proses Enkripsi dan Dekripsi

Secara matematis, proses enkripsi merupakan pengoperasian fungsi E (enkripsi) menggunakan k (kunci) pada M (*plaintext*) sehingga dihasilkan C (*ciphertext*). Notasi dari proses enkripsi seperti dibawah ini :

$$E_k(M) = C \quad [1]$$

Sedangkan untuk proses dekripsi, merupakan pengoperasian fungsi D (dekripsi) menggunakan k (kunci) pada C (*ciphertext*) sehingga dihasilkan M (*plaintext*). Notasi dari proses dekripsi seperti dibawah ini :

$$D_k(C) = M \quad [2]$$

Sehingga dari dua hubungan diatas berlaku :

$$D_k(E_k(M)) = M \quad [3]$$

2.4 Algoritma Rivest Code 6 (RC-6)

Algoritma RC6 merupakan salah satu kandidat Advanced Encryption Standard (AES) yang diajukan oleh RSA Security Laboratories kepada NIST. Dirancang oleh Rivest (2001) algoritma ini merupakan pengembangan dari algoritma sebelumnya yaitu RC5 dan telah memenuhi semua kriteria yang diajukan oleh NIST. RC6 adalah algoritma yang menggunakan ukuran blok hingga 128 bit, dengan ukuran kunci yang digunakan bervariasi antara 128, 192 dan 256 bit.

Algoritma RC6 dilengkapi dengan beberapa parameter, sehingga dituliskan sebagai RC6-w/r/b. Parameter w merupakan ukuran kata dalam satuan bit, parameter r merupakan bilangan bukan negatif yang menunjukkan banyaknya iterasi selama proses enkripsi dan parameter b menunjukkan ukuran kunci enkripsi dalam byte. Setelah algoritma ini masuk dalam kandidat AES, maka ditetapkan bahwa nilai w = 32, r=20 dan b bervariasi antara 16, 24 dan 32 byte.

RC6-w/r/b memecah blok 128 bit menjadi 4 buah blok 32-bit, dan mengikuti aturan enam operasi dasar sebagai berikut :

1. $a + b$ operasi penjumlahan bilangan integer
2. $a - b$ operasi pengurangan bilangan integer
3. $a \oplus b$ operasi exclusive-OR (XOR)
4. $a \times b$ operasi perkalian bilangan integer
5. $a \lll b$ a dirotasikan ke kiri sebanyak variabel kedua (b)
6. $a \ggg b$ a dirotasikan ke kanan sebanyak variabel kedua (b)

(Rivest, 1998)

a. Penjadwalan Kunci (*Key Schedule*) RC-6

Key schedule digunakan untuk menginisialisasi kunci yang diinputkan oleh user sebelum digunakan untuk proses enkripsi dan dekripsi RC6. Jadwal kunci RC6-w/r/b sangat mirip dengan RC5-w/r/b. Satu-satunya perbedaannya adalah RC6-w/r/b lebih banyak menyediakan kata-kata untuk proses enkripsi dan dekripsi.

User memasukkan *key* sebanyak b *bytes* ($0 \leq b \leq 255$) dari *key* ini. *Byte-byte* 0 kemudian ditambahkan untuk membuat panjang kunci sama dengan sebuah bilangan integral tidak 0 *word*. *Byte-byte* kunci ini kemudian dimuat pada *little-endian fashion* ke dalam sebuah array c *w-bit word* L[0], ..., L[c-1]. Jadi *byte* pertama kunci disimpan sebagai *low-order byte* dari L[0], dan seterusnya, dan L[c-1] di-pad dengan *high order byte*. Jumlah *w-bit word* yang akan di-generate untuk kunci *round* adalah $2r+4$ words(masing-masing w *bits*) dan disimpan ke dalam array S[0; :::, 2r+3]. Array ini akan digunakan untuk proses enkripsi dan dekripsi.

Konstanta $P_{32} = B7E15163$ dan $Q_{32} = 9E3779B9$ (*hexadecimal*) adalah konstanta yang sama, disebut juga konstanta ajaib, sama seperti yang digunakan di RC5. Nilai P_{32} turunan dari ekspansi biner $e-2$, dimana e adalah dasar dari fungsi logaritma alami. Nilai Q_{32} berasal dari ekspansi biner $\phi -1$, dimana ϕ adalah *Golden Ratio*. Mirip dengan ketentuan dari RC5 untuk P_{64} dan lainnya yang dapat digunakan untuk versi RC6 dengan ukuran kata lain. Nilai ini bisa berubah-ubah, nilai lain bisa dipilih untuk menjadikannya “*custom version of RC6*”.

Sebagai contoh, akan dilakukan enkripsi pada kata “AndrewChandra”, *plaintext* ini kemudian dijadikan nilai ASCII lalu dijadikan nilai biner dan dimasukkan ke dalam 4 buah register A, B, C, dan D.

Tabel 1.
Tabel perhitungan Register dari key “AndrewChandra”

Plaintext	ASCII	Bilangan Biner	Register
A	65	Blok Data	A 01000001011011100110010001110010
n	110	01101110	
d	100	01100100	
r	114	01110010	
e	101	01100101	B 01100101011101110100001101101000
w	119	01110111	
C	67	01000011	
h	104	01101000	
a	97	01100001	C 01100001011011100110010001110010
n	110	01101110	
d	100	01100100	
r	114	01110010	
a	97	01100001	D 01100001000000000000000000000000000000

Lalu masukkan *key* dari *user* ke dalam *array* L[0], L[1], L[2], dan L[3] dengan cara menempatkan *byte* pertama dari *key* ke *byte* paling kanan (least significant) dari L[0], lalu *byte* berikutnya ditempatkan di sisi kiri dari *byte* sebelumnya pada L[0] hingga terisi 4 *byte*., hal ini diteruskan hingga pada L[3]. Jika *key* yang dimasukkan kurang dari 16-*byte*, maka ditambahkan bit-bit 0 sehingga panjang *key* menjadi 16 *byte*.

Setelah itu inialisasi *key* S menggunakan *magic constant* P_w dan Q_w sesuai dengan algoritma di Gambar 1. Proses untuk membangun *key* inialisasi ini didapat dari 2 buah *magic constant* P_w dan Q_w . Beberapa nilai *magic constant* dalam *hexadecimal* pada beberapa ukuran register(w) adalah sebagai berikut :

Tabel 2.
Tabel nilai *magic constant*

w	P_w	Q_w
16	b7e1	9e37
32	b7e15163	9e3779b9
64	b7e151628aed2a6b	9e3779b9f4a7c15

Algoritma Inisialisasi *key*

```
Inisialisasi key S[i]}  
  
S[0]=Pw  
for i=1 to 2r+3  
    S[i]=S[i-1]+ Qw  
end
```

Gambar 5. Algoritma Inisialisasi *Key* Rivest Code-6

Kemudian dilakukan kombinasi *key* yang dimasukkan *user* dengan *key* inisialisasi S, dilakukan dengan algoritma *key scheduling* RC6 yang dapat dilihat pada algoritma di Gambar 5 menghasilkan 44 *round key* yang ditempatkan pada array S[0], S[1], S[2],..., S[43].

Proses kombinasi *key* dengan menggunakan *key* inisialisasi menjadi *round key* ditampung pada array S[0, 1, 2, ..., 2r+3] dengan panjang masing-masing *round key* 32-bit. *Key* inisialisasi S[i] ditambahkan dengan nilai A dan B. Pada langkah awal, nilai A dan B adalah 0, lalu hasilnya digeser sebesar 3 bit ke kiri, kemudian nilai S[i] yang baru disimpan di A. L[j] ditambahkan dengan nilai A dan B, lalu hasilnya digeser ke kiri sebesar 5 bit. Langkah ini memperlihatkan *key* dari *user* telah dikombinasikan dengan *key* inisialisasi. Proses ini dilakukan terus sebanyak 3 kali nilai maksimum {c, 2r + 4}.

b. Sekilas Mengenai Enkripsi dan Dekripsi

RC6 bekerja dengan 4 *w-bit register* A, B, C, dan D yang berisi input awal *plaintext* beserta *ciphertext output* pada akhir enkripsi. *Byte* pertama dari *plaintext* atau *ciphertext* ditempatkan di *Least-Significant Byte* dari A, sedangkan *byte* terakhir dari *plaintext* atau *ciphertext* ditempatkan di *Most-Significant Byte* dari register D. Kami menggunakan (A, B, C, D) = (B, C, D, A) berarti tugas paralel nilai di sebelah kanan untuk register di sebelah kiri.

c. Detail Enkripsi RC-6

Enkripsi RC6 menerima 4 *w-bit register* yang terdiri dari A, B, C, dan D yang berisi inisial input *plaintext* begitu juga pada akhir proses enkripsi. *Byte plaintext* yang pertama diletakkan pada *byte* awal dari A, dan *byte plaintext* terakhir diletakkan pada *byte* yang paling akhir dari register D.

Secara umum algoritma RC6 yang dapat dilihat pada algoritma di Gambar 5. memiliki langkah-langkah sebagai berikut :

1. *Whitening* awal
2. Transformasi
3. *Mixing*
4. *Swap register*
5. *Whitening* akhir

d. Detail Dekripsi RC-6

Data dari *ciphertext* yang tersimpan pada 4 *register* akan dikurangkan dengan hasil *key schedule* dan dirotasi sebanyak r sambil dilakukan operasi XOR terhadap data tersebut.

Hasil akhir adalah mendapatkan *plaintext* dengan melakukan proses pengurangan ke masing-masing bagian dengan hasil *key schedule*. Data-data ini kemudian digabungkan kembali membentuk *plaintext* sesuai dengan data awal sebelum dienkripsi.

Seluruh operasi pada dekripsi RC-6 merupakan kebalikan dari proses enkripsi, yaitu terdiri dari :

1. *Whitening* akhir
2. *Swap register*

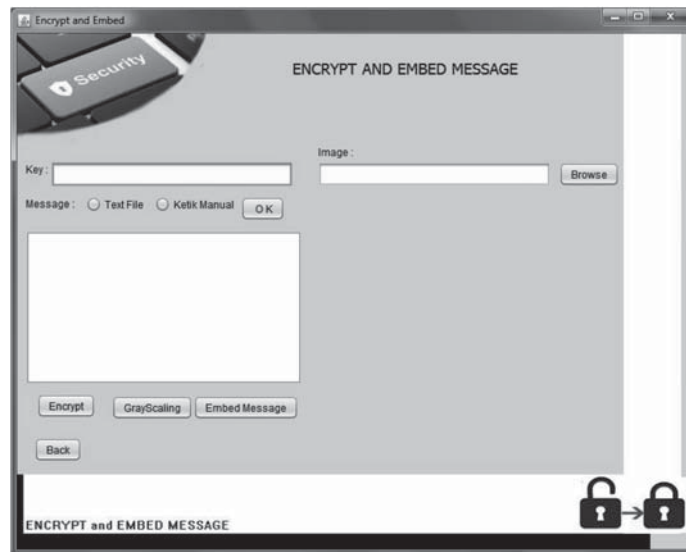
3. Transformasi
4. *Mixing*
5. *Whitening* awal

3. Hasil dan Pembahasan

3.1. Implementasi Sistem

3.1.1 Implementasi Input pada Proses Enkripsi dan Penyisipan

Input yang diperlukan dalam proses enkripsi dan penyisipan ada 3 *input* yaitu teks pesan yang ingin dienkripsi, *file* citra, kunci enkripsi, dan lokasi *folder* untuk menyimpan output sistem. *File* teks harus mempunyai format txt atau dengan melakukan pengetikan langsung sedangkan *file* citra berformat BMP. Tidak ada batasan maksimum untuk panjang kunci yang ingin dimasukkan. Sistem mengimplementasikan fungsi *validasi* format citra untuk memastikan bahwa inputan file citra yang dipilih pengguna adalah file citra BITMAP.



Gambar 6. Halaman Enkripsi dan Penyisipan

3.1.2 Implementasi Input pada Proses Dekripsi dan Ekstraksi

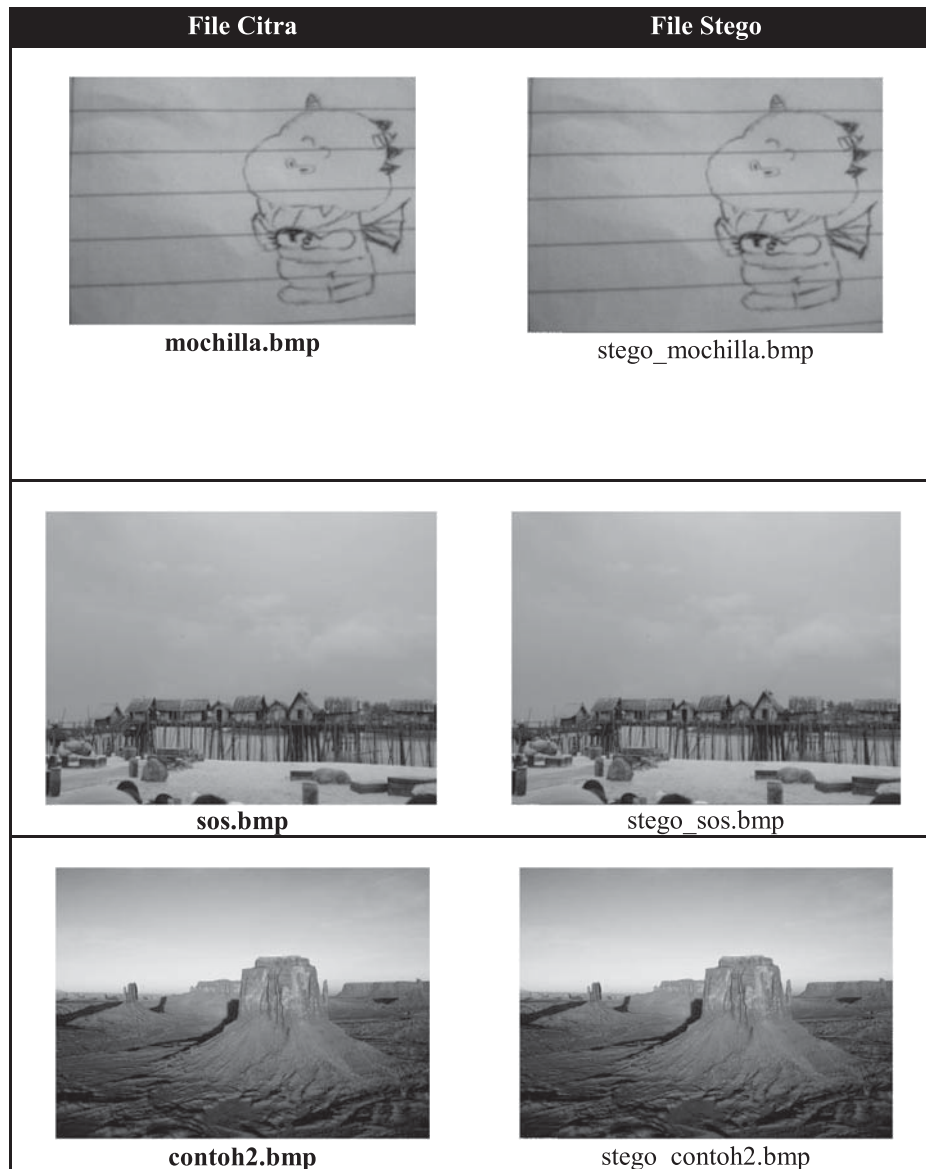
Input yang diperlukan dalam proses ekstraksi dan dekripsi ada 2 *input*, yaitu kunci skripsi dan *file* gambar yang sudah dilakukan steganografi. *File* gambar ini harus dengan format .BMP. Gambar 7 menunjukkan tampilan antarmuka dari sistem untuk proses ekstraksi dan dekripsi.



Gambar 7. Halaman Dekripsi dan Ekstraksi

3.1.3 Implementasi Output pada Proses Enkripsi dan Penyisipan

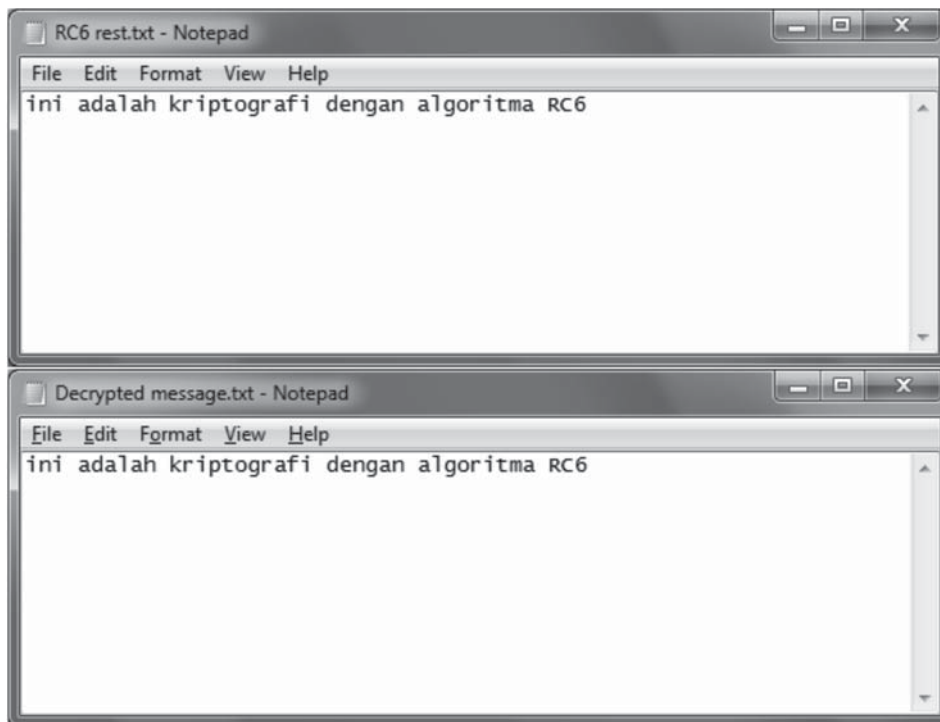
Output yang dihasilkan dari proses enkripsi dan penyisipan adalah sebuah citra (file stego) yang telah disisipkan pesan rahasia (file teks) yang sudah terenkripsi terlebih dengan algoritma RC6. User dapat menyimpan *file* citra hasil skripsi dan penyisipan dengan nama sesuai kebutuhan usir. Setelah melakukan proses ekstraksi dan deskripsi usir juga bisa menyimpan pesan rahasia tersebut dalam format .txt dan nama *file*-nya sesuai kebutuhan usir. Gambar 8 menunjukkan contoh file citra asli dan file citra yang sudah disisipi pesan terenkripsi (file stego).



Gambar 8. Citra asli dibandingkan dengan citra setelah disisipi pesan

3.1.4 Implementasi Output pada Proses Dekripsi dan Ekstraksi

Output yang dihasilkan dari proses ekstraksi dan dekripsi adalah sebuah file teks yang berisi pesan asli yang sebelumnya dienkripsi dan disisipkan pada citra BMP. Gambar 9 menunjukkan contoh pesan rahasia sebelum di enkripsi dan disisipkan dan setelah diekstraksi dan didekripsi :



Gambar 9. Contoh file teks sebelum di enkripsi dan disisipkan dan setelah diekstraksi dan didekripsi.

3.2. Analisis Sistem

3.2.1. Tujuan Analisis




Dalam penelitian ini, terdapat beberapa hal yang merupakan tujuan dari analisis dari sistem yang mengimplementasikan algoritma RC6 dan *End of File (EOF)*. Hal tersebut adalah untuk menganalisa apakah dengan menggunakan kedua metode di atas dapat menyembunyikan pesan dengan baik, menganalisa pengaruh panjang kunci dan panjang pesan terhadap citra yang disisipi.

3.2.2. Data Analisis

Subbab data analisis berisi data file citra dan file teks yang digunakan sebagai pengujian sistem. Tabel 3 menunjukkan citra yang akan digunakan sebagai penampung pesan rahasia yang terenkripsi dalam analisis sistem yang berupa citra BMP.

Untuk data pesan rahasia atau file teks yang akan digunakan dalam pengujian sistem ini terdapat 3 buah pesan yang mempunyai panjang pesan yang berbeda. Tabel 4 menunjukkan data pesan rahasia / file teks yang akan dienkripsi dan disisipkan ke dalam citra dalam analisis sistem.

Tabel 3.
Data File Citra penampung pesan terenkripsi.

No	Nama File	Ukuran File	Dimensi	Citra
1	mochilla.bmp	263.054 bytes	350 x 250 pixels 24 bit	
2	food.bmp	920.094 bytes	680 x 451 pixels 24 bit	
3	valley.bmp	2.359.350 bytes	1024 x 768 pixels 24 bit	

Tabel 4.
Data file teks yang akan dienkripsi dan disisipkan.

No	Nama File	Ukuran File	Isi Pesan
1	Pesan 16byte.txt	14 bytes	Andrew Chandra
2	Pesan 32byte.txt	25 bytes	Panjang Kunci + padding: 32 Karakter (256 bit) Algoritma Kriptografi RC6
3	Pesan Panjang.txt	152 bytes	Ini adalah sebuah program berbasis Java untuk mengimplementasikan kriptografi dengan algoritma Rivest Code 6 dan steganografi dengan metode End of File.

Tabel 5.
Data kunci yang akan digunakan

No	Ukuran Kunci	Isi Kunci
1	16 bytes	Kunci 16 bytes
2	32 bytes	Teknik Informatika 2009
3	>32 bytes	Implementasi EOF dan RC6 di Netbeans

Tabel 6.
Hasil pengujian enkripsi dan steganografi terhadap file citra dengan menggunakan panjang kunci 16 byte.

No	File Citra	File Teks	Ukuran Citra Sebelum Enkripsi	Resolusi Sebelum Enkripsi	Ukuran Citra Sesudah Enkripsi	Resolusi Setelah Enkripsi	Sisip Ekstraksi
1	mochilla.bmp	Pesan 16byte.txt	263.054 bytes	350 x 250	263.054 bytes	350 x 251	V
2	mochilla.bmp	Pesan 32byte.txt	263.054 bytes	350 x 250	264.106 bytes	350 x 251	V
3	mochilla.bmp	Pesan Panjang.txt	263.054 bytes	350 x 250	264.106 bytes	350 x 251	V
4	food.bmp	Pesan 16byte.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
5	food.bmp	Pesan 32byte.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
6	food.bmp	Pesan Panjang.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
7	valley.bmp	Pesan 16byte.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V
8	valley.bmp	Pesan 32byte.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V
9	valley.bmp	Pesan Panjang.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V

Tabel 7.

Hasil pengujian enkripsi terhadap file teks dengan menggunakan panjang kunci 32 byte.

No	File Citra	File Teks	Ukuran Citra Sebelum Enkripsi	Resolusi Sebelum Enkripsi	Ukuran Citra Sesudah Enkripsi	Resolusi Setelah Enkripsi	Sisip Ekstraksi
1	mochilla.bmp	Pesan 16byte.txt	263.054 bytes	350 x 250	263.054 bytes	350 x 251	V
2	mochilla.bmp	Pesan 32byte.txt	263.054 bytes	350 x 250	264.106 bytes	350 x 251	V
3	mochilla.bmp	Pesan Panjang.txt	263.054 bytes	350 x 250	264.106 bytes	350 x 251	V
4	food.bmp	Pesan 16byte.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
5	food.bmp	Pesan 32byte.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
6	food.bmp	Pesan Panjang.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
7	valley.bmp	Pesan 16byte.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V
8	valley.bmp	Pesan 32byte.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V
9	valley.bmp	Pesan Panjang.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V

Tabel 8.

Hasil pengujian enkripsi dan steganografi terhadap file citra dengan menggunakan panjang kunci >32 byte.

No	File Citra	File Teks	Ukuran Citra Sebelum Enkripsi	Resolusi Sebelum Enkripsi	Ukuran Citra Sesudah Enkripsi	Resolusi Setelah Enkripsi	Sisip Ekstraksi
1	mochilla.bmp	Pesan 16byte.txt	263.054 bytes	350 x 250	263.054 bytes	350 x 251	V
2	mochilla.bmp	Pesan 32byte.txt	263.054 bytes	350 x 250	264.106 bytes	350 x 251	V
3	mochilla.bmp	Pesan Panjang.txt	263.054 bytes	350 x 250	264.106 bytes	350 x 251	V
4	food.bmp	Pesan 16byte.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
5	food.bmp	Pesan 32byte.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
6	food.bmp	Pesan Panjang.txt	920.094 bytes	680 x 451	922.134 bytes	680 x 452	V
7	valley.bmp	Pesan 16byte.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V
8	valley.bmp	Pesan 32byte.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V
9	valley.bmp	Pesan Panjang.txt	2.359.350 bytes	1024 x 768	2.362.422 bytes	1024 x 769	V

Catatan :

- V = Berhasil, X = Gagal

Dari hasil pengujian diatas, dapat disimpulkan bahwa proses enkripsi dengan algoritma RC6 dan steganografi dengan algoritma *End of File* sedikit mengubah ukuran file teks. Dari data diatas perubahan yang terjadi hanya sangat kecil sekali sehingga tidak terlalu

mempengaruhi kapasitas yang akan dibutuhkan pada file citra yang menjadi wadah steganografi.

Perubahan ukuran *file* citra di sini terjadi karena penyisipan pesan dilakukan dengan cara melakukan penambahan baris sesuai yang dibutuhkan. Misalnya citra yang menjadi penampung berukuran 50 x 50 pixel, panjang pesan yang ingin disisipkan adalah 20 byte, dan panjang kunci setelah dilakukan proses MD5 adalah 16byte. Dalam kasus ini akan dilakukan penambahan sebanyak 1 baris pada citra penampungnya sehingga menjadi berukuran 50 x 51 pixel. Bila dengan kasus serupa namun panjang pesan yang disisipi adalah 40 byte, maka akan dilakukan penambahan sebanyak 2 baris pada citra penampungnya sehingga berukuran 50 x 52 pixel. Oleh karena itu bisa dikatakan bahwa panjang pesan akan mempengaruhi besarnya penambahan ukuran citra penampung, sedangkan panjang kunci tidak berpengaruh terhadap penambahan ukuran citra penampung. Hal ini disebabkan karena sebesar apapun kunci yang akan disisipkan, saat dilakukan proses enkripsi MD5 maka hasilnya akan tetap menjadi sebesar 16 byte.

4. Kesimpulan

Berdasarkan hasil analisis dan implementasi sistem, maka diperoleh kesimpulan sebagai berikut :

1. Proses enkripsi RC6 dan steganografi *End of File* dapat melakukan penyisipan pesan dengan baik karena ukuran citra penampung dapat bertambah sesuai dengan pesan yang ingin disisipkan.
2. Dengan menggunakan metode *End of File* maka pesan yang ingin disisipkan tidak dibatasi.
3. Perubahan ukuran *file* citra ini tergantung dari besarnya citra yang digunakan dan juga besarnya pesan yang disisipkan.
4. Panjang kunci yang ingin disisipkan pasti sebesar 16 byte dikarenakan melalui proses enkripsi dengan algoritma MD5.

Daftar Pustaka

- Aditya, Yogie, (2010), Studi Pustaka untuk Steganografi dengan Beberapa Metode, Yogyakarta : Universitas Islam Indonesia
- Cormen, Thomas H., Leiserson, Charles E, Rivest, Ronald L., Stein, Clifford, (2009), Introduction to Algorithm Third Edition, London
- Dony, Ariyus. (2005), Kriptografi Keamanan Data dan Komunikasi. Yogyakarta: Penerbit Andi Offset.
- Edisuryana, Mukharrom, (2013), Aplikasi Steganografi pada Citra Berformat Bitmap dengan Menggunakan Metode End of File, Semarang : Universitas Diponegoro
- Krisnawati, (2008), Metode Least Significant Bit(LSB) dan End of File (EOF) untuk Menyisipkan Teks ke dalam Citra Grayscale, Medan : Universitas Sumatera Utara
- Lu, Mingming, (2002), RC6 Encryption and Decryption, diakses pada tanggal 3 Maret 2014 dari <http://www.codeproject.com/Articles/2545/RC6-encryption-and-decryption>
- Muharini, Anisah, (2012), Aplikasi Algoritma Rivest Code 6 dalam Pengamanan Citra Digital, Jakarta : Universitas Indonesia
- Permana, Rangga Wisnu Adi, (2008), Implementasi Algoritma RC6 untuk Enkripsi SMS pada Telepon Seluler, Bandung : Institut Teknik Bandung
- Prayudi, Yudi, Idham, Malik, (2005). Studi dan Analisis Algoritma RIVEST CODE 6(RC6) Dalam Enkripsi/Denkripsi Data. Yogyakarta : Universitas Islam Indonesia

- Rachmawanto, Eko Hari, (2010), Teknik Keamanan Data Menggunakan Kriptografi dengan Algoritma Vernam Cipher dan Steganografi dengan Metode End of File(EOF), Semarang : Universitas Dian Nuswantoro
- Rivest, R.L., Robshaw, M.J.B., Sidney, R., dan Yin, Y.L, (2001). The RC6 Block Cipher. USA. MIT Laboratory for Computer Science, Cambridge
- Sejati, Adiputra, (2010), Studi dan Perbandingan Steganografi Metode EOF(End of File) dengan DCS(Dynamic Cell Spreading), Bandung : Institut Teknologi Bandung
- Susanto, Hari, (2013), Fungsi Hash(Teknik Kriptografi), diakses pada tanggal 7 Mei 2014 dari <http://hari-cio-8a.blog.ugm.ac.id/2013/03/22/fungsi-hash-teknik-kriptografi>
- Tiberiu, Popa, (2011), AES(Rijndael) Implementation, diakses pada tanggal 8 Februari 2014 dari <http://n3vrax.wordpress.com/2011/08/14/aesrijndael-java-implementation>
- Wandani, Henny, (2012), Implementasi Sistem Keamanan Data dengan Menggunakan Teknik Steganografi End of File (EOF) dan Rabin Public Key Cryptosystem, Medan : Universitas Sumatera Utara
- Wasino, (2012), Implementasi Steganografi Teknik End of File dengan Enkripsi Rijndael, Tangerang : STMIK Dharma Putra
- Yudi Triyanto, Fransiskus. (2010). Analisis Perbandingan Algoritma Enkripsi AES-128 dengan Algoritma RC6. Yogyakarta : Universitas Kristen Duta Wacana