INFORMATIKA Mulawarman | Februari 2014 Vol. 9 | No. 1
ISSN 1858-4853

KEAMANAN DATA DENGAN MENGGUNAKAN ALGORITMA *RIVEST CODE* 4 (RC4) DAN STEGANOGRAFI PADA CITRA DIGITAL

Hendrawati[1)], Hamdani[2)], Awang Harsa K[3)]

[1,2,3)]Program Studi Ilmu Komputer, FMIPA, Universitas Mulawarman
Email : aisah.hasbunallah@gmail.com[1)], hamdani@unmul.ac.id[2)], awangkid@gmail.com[3)]

ABSTRAK

Kriptografi merupakan ilmu dan seni yang mempelajari cara-cara untuk menyembunyikan informasi dengan menyamarkannya menjadi sandi yang sulit ditemukan maknanya. Kerahasiaan pesan sangatlah penting, tidak cukup hanya mengubah isi pesan untuk menjaga kerahasiaannya. Salah satu cara pengamanan data yang dapat dilkukan dengan mengkombinasikan kriptografi dan steganografi. Tujuannya adalah untuk merahasiakan pesan yang dikirim, serta menghindari pesan dari kecurigaan. Penelitian ini bertujuan untuk membuat suatu aplikasi yang mampu menyembunyikan data rahasia yang telah disandikan menggunakan algoritma *Rivest Code* 4 (RC4) ke dalam bentuk citra digital 24 bit dengan metode *Least Significant Bit* (LSB). Aplikasi ini melakukan enkripsi pada pesan teks dan penyisipan pesan dengan media penampung citra format *bmp dan *png. Hasil akhir berupa aplikasi yang menggunakan algoritma RC4 dan steganografi yang menggunakan metode LSB untuk meningkatkan keamanan data.

Kata kunci: kriptografi, *Least Significant Bit*, *Rivest Code* 4, steganografi.

PENDAHULUAN

Kriptografi adalah ilmu dan seni yang mempelajari cara-cara untuk menyembunyikan informasi dengan cara menyamarkannya menjadi sandi yang sulit ditemukan maknanya. Saat ini kriptografi modern dipicu oleh perkembangan peralatan komputer digital. Dengan komputer digital, cipher yang lebih kompleks menjadi sangat mungkin untuk dapat dihasilkan. Tidak seperti kriptografi klasik yang mengenkripsikan karakter per karakter (alfabet tradisional), kriptografi modern beroperasi pada *string biner*. Kriptografi modern tidak hanya memberikan aspek keamanan kerahasiaan pesan, tetapi juga aspek keamanan lain seperti otentikasi, integritas data dan penyangkalan.[1]

Proses pengamanan pesan dalam bentuk teks dapat dilakukan dengan mengenkripsi pesan menggunakan algoritma modern tertentu. Salah satu algoritma modern yang dapat digunakan algortima *rivest code* 4 (RC4). Yang melakukan proses enkripsi bit sebuah plainteks digabungkan dengan *pesudorandom bit stream* menggunakan *operator XOR*.[2]

Penyembunyian pesan tidak hanya dilakukan dengan menyamarkan pesan, melainkan dapat menggunakan media lain untuk menyisipkan pesan kedalamnya. Sehingga pihak ketiga tidak akan curiga terhadap pesan yang diberi, karena yang terlihat hanyalah media penampung pesan. Penyisipan pesan kedalam media penampung juga terdapat pada skripsi I Gede Wiryawan dengan judul Membangun Aplikasi Steganografi Dengan Metode *Least Significant Bit* (LSB).[3] Penyisipan pesan dilakukan dengan menggantikan bit-bit di dalam segmen citra dengan bit-bit pesan. Metode LSB digunakan karena hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya.[4] Sehingga penulis bermaksud untuk mengembangkannya dengan memberikan satu lapisan teknik dalam pengamanan data, yaitu dengan menggabungkan kedua pengamanan data untuk memperkuat kerahasiaan pesan.

METODE PENELITIAN

Kriptografi terdiri dari proses enkripsi dan dekripsi. Enkripsi merupakan proses penyandian pesan asli (*plaintext*) menjadi pesan terenkripsi (*ciphertext*) menggunakan kunci tertentu. Dekripsi merupakan kebalikan dari proses enkripsi, yaitu penyandian kembali pesan terenkripsi (*cipher*text) menjadi pesan asli (*plaintext*) menggunakan kunci yang sama saat melakukan proses enkripsi. Dalam kriptografi terdapat dua algoritma yaitu algoritma klasik dan modern. Jika algoritma sandi memiliki kekuatan enkripsi pesan rahasia maka algoritma dapat digunakan untuk pengamanan informasi. Pada implementasinya algoritma sandi harus memperhatikan kualitas layanan dari keseluruhan sistem yang diimplementasikan.[1]

Algortima sandi yang memiliki kekuatan pengamanan yang baik terletak pada kunci dan tidak pada kerahasiaan algoritmanya, kekuatannya diukur dari tingkat kesulitan kunci pendekripsian dari sandi, tidak pada rahasia algoritma sandi. Algoritma yang kuat memiliki kemungkinan jangkauan kunci yang besar sehingga sistem tidak mungkin dipecahkan dengan mencoba semua kemungkinan kunci secara

brute force. Algoritma yang kuat juga akan menciptakan *ciphertext* yang acak untuk semua standar tes statistik. Algoritma modern merupakan sistem kriptografi yang kekuatannya terletak pada kunci yang dijaga kerahasiaannya.[1]

Berdasarkan kesamaan kuncinya, algoritma sandi terbagi atas algoritma simetris dan algoritma asimetris. Algoritma simetris melakukan proses enkripsi dan dekripsi dengan menggunakan kunci rahasia untuk setiap bit (*cipher stream*) dan bit per blok (*cipher block*). Dalam sebuah *Stream cipher*, digit dari plainteks akan dienkripsi persatuan data contohnya adalah : bit, byte, nible atau 5 bit dan setiap mengenkripsi satu satuan data, digunakan kunci yang merupakan hasil pembangkitan dari kunci sebelumnya. Salah satu algoritma dengan proses kunci *stream cipher* adalah *rivest code 4* (RC4). [2]

Algoritma *Rivest code 4* (RC4) ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA, digunakan secara luas pada beberapa aplikasi dan umumnya dinyatakan sangat aman. RC4 merupakan metode penyandian pesan teks yang melakukan enkripsi per bit sehingga kelebihan dari metode ini kerusakan pada satu bit tidak mempengaruhi keseluruhan isi pesan. Pada RC4 dihasilkan pseudo random stream bit. Seperti halnya stream cipher lainnya, algoritma RC4 ini dapat digunakan untuk mengenkripsi dengan mengkombinasikannya dengan plainteks menggunakan *Exclusive-or* (Xor). Untuk proses dekripsi dilakukan cara yang sama dan dengan kunci yang sama, karena Xor merupakan fungsi simetrik. Secara garis besar proses algoritma RC4 dibagi menjadi dua bagian yaitu, *Key Scheduling Algorithm* (KSA) dan *Pseudo Random Generation Algorithm* (PRGA).[2]

Key Scheduling Algorithm (KSA) digunakan untuk menginisialisasikan permutasi di array 'S'. Panjang kunci didefinisikan sebagai jumlah byte dikunci dan mempnyai rentang panjang kunci dari 1 sampai 256. Kemudian array 'S' di proses ke 256 iterasi dengan cara yang sama dengan PRGA utama, juga di kombinasikan dalam byte dari kunci pada waktu yang bersamaan. Berikut adalah tiga tahapan algoritma KSA :

Proses inisialisasi S-Box (*Array* S)

```
-----------------------------------------------------------
--
for i = 0 to 255
    S[i] := i
-----------------------------------------------------------
--
```

Dari algoritma tersebut didapatkan urutan nilai S-Box sebagai berikut:

Tabel 1. Inisialisasi S-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

Proses inisialisasi S-Box (*Array* K)

```
-----------------------------------------------------------
--
Array kunci / Array dengan panjang kunci "length"
For i = 0 to 255
K[i] = Kunci [i mod length]
-----------------------------------------------------------
--
```

Kemudian melakukan langkah pengacakan S-Box dengan langkah berikut:

```
-----------------------------------------------------------
--
i = 0 ; j = 0
for i = 0 to 255
{
    j = ( j + S[i] + K[i] ) mod 256
    swap S[i] dan S[j]
-----------------------------------------------------------
--
```

Setelah itu *Pseudo Random Generation Algorithm* (PRGA). Memodifikasi state dan *output* sebuah *byte* dari *key stream*. Hal ini penting karena banyaknya dibutuhkan iterasi. Dalam setiap ietrasi, PRGA menginkremen i, menambahkan nilai S yang ditunjuk oleh i sampai j, kemudian menukar nilai S[i] dan S[j], lalu mengembalikan elemen dari S di tempat S[i] + S[j](modulo 256). Setiap elemen S ditukar dengan elemen lainnya paling tidak satu kali setiap 256 iterasi. Realisasi algoritma sebagai berikut :

```
-----------------------------------------------------------
--
i = ( i + 1 ) mod 256
j = ( j + S[i] ) mod 256
swap S[i] dan S[j]
t = ( S[i] + S[j] ) mod 256
y = S(t)
-----------------------------------------------------------
--
```

Byte y di XOR-kan dengan plainteks untuk menghasilkan cipherteks pada proses enkripsi, sedangkan proses dekripsi *byte* y di XOR dengan cipherteks untuk mendapatkan plainteks. Berikut adalah implementasi RC4 dengan mod 5 byte untuk lebih menyederhanakan. Pertama inisialisasi S-Box dengan panjang 5 byte, dengan S[0]=0, S[1]=1, S[2]=2, S[3]=3, S[4]=4 sehingga array S menjadi :

0	1	2	3	4

Inisialisasi 5 byte kunci array, K[i]. Misalkan kunci terdiri dari 3 byte yaitu 114. Ulangi kunci sampai memenuhi seluruh array K, karena kunci hanya terdiri dari 3 byte maka 2 byte berikutnya diisi dengan pengulangan kunci dimulai dari 1 kembali, sehinggan array K menjadi :

1	1	4	1	1

Melakukan proses pengacakan pada S[i] dan K[i], karena pada contoh menggunakan array dengan panjang 5 byte maka proses iterasi akan dilakakukan sebanyak 5 kali. Berdasarkan algoritma pengacakan S-Box nilai awal i=0 sampai i=4 akan menghasilkan array S sebagai berikut :

Iterasi Pertama :
i=0, maka
j=(j + S[i] + K[i]) mod 5
=(0 + S[0] + K[0]) mod 5
=(0 + 0 + 1) mod 5
= 1
Swap S[0] dan S[1] sehingga menghasilkan array S:

1	0	2	3	4

Iterasi kedua:
i=1, maka
j=(j + S[i] + K[i]) mod 5
=(1 + S[1] + S[1]) mod 5
=(1 + 0 + 1) mod 5
= 2
Swap S[1] dan S[2] sehingga menghasilkan array S:

1	2	0	3	4

Iterasi ketiga:
i=2, maka
j=(j + S[i] + K[i]) mod 5
=(2 + S[2] + S[2]) mod 5
=(2 + 0 + 4) mod 5
= 1
Swap S[2] dan S[1] sehingga menghasilkan array S:

1	0	2	3	4

Iterasi keempat:
i=3, maka
j=(j + S[i] + K[i]) mod 5
=(1 + S[3] + S[3]) mod 5
=(1 + 3 + 1) mod 5
= 0
Swap S[3] dan S[0] sehingga menghasilkan array S:

3	0	2	1	4

Iterasi kelima:
i=4, maka
j=(j + S[i] + K[i]) mod 5
=(0 + S[4] + S[4]) mod 5
=(0 + 4 + 1) mod 5
= 0
Swap S[4] dan S[0] sehingga menghasilkan array S:

4	0	2	1	3

Berikutnya adalah proses enkripsi yaitu meng-XOR-kan PRGA dengan plainteks, contoh plainteks “HI”. Karena plainteks terdiri dari dua karakter maka terjadi dua iterasi.

Iterasi 1:
i=0; j=0;
i=(i + 1) mod 5
=(0 + 1) mod 5
=1
j=(j +S[i]) mod 5
=(0 + S[1]) mod 5
=(0 + 0) mod 5
=0
Swap S[1] dan S[0]

0	4	2	1	3

t=(S[i] + S[j]) mod 5
=(S[1] + S[0]) mod 5
=(4 + 0) mod 5
=4
y=S[t]
=S[4] = 3 = 0000 0011
H =72= 0100 1000 xor
0100 1011 = 75 = K
*untuk mencari nilai karakter dapat dilakukan konversi dari nilai desimal ke biner.

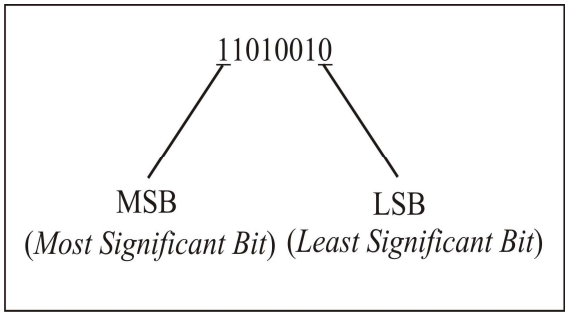
Iterasi 2:
i=(i + 1) mod 5
=(1 + 1) mod 5
=2
j=(j +S[i]) mod 5
=(0 + S[2]) mod 5
=(0 + 2) mod 5
=2
Swap S[2] dan S[2]

0	4	2	1	3

t=(S[i] + S[j]) mod 5
=(S[2] + S[2]) mod 5
=(2 + 2) mod 5

=4
y=S[t]
=S[4] = 3 = 0000 0011
I =73= 0100 1001 xor
0100 1010 = 74 = J

Pesan yang akan diterima adalah "KJ" untuk mendapatkan isi pesan yang asli maka dilakukan proses Xor terhadap cipherteks dengan memasukkan kunci yang sama saat melakukan enkripsi. Pesan yang telah dienkripsi akan terlihat sangat tidak lazim oleh pihak lain dan dapat mendatangkan kecurigaan. Untuk itu penting menyembunyikan pesan rahasia dengan segala cara sehingga selain pengguna yang dituju, pihak lain tidak akan menyadari keberadaan dari pesan. Steganografi mempelajari cara menempatkan pesan rahasia ke dalam suatu media penampung. Penyembunyian pesan dilakukan dengan menggantikan bit-bit pesan di dalam segmen citra dengan bit-bit pesan. Metode yang paling sederhana adalah metode modifikasi *Least Significant Bit* (LSB). Pada susunan bit di dalam sebuah *byte* (1 *byte* = 8 bit), ada bit paling berarti yang disebut *Most Significant Bit* (MSB) dan yang paling kurang berarti (LSB). Bit pada LSB sangat cocok untuk diganti karena hanya mengubah satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Berikut contoh susunan bit pada byte dan metode penyisipan pesan, [1]

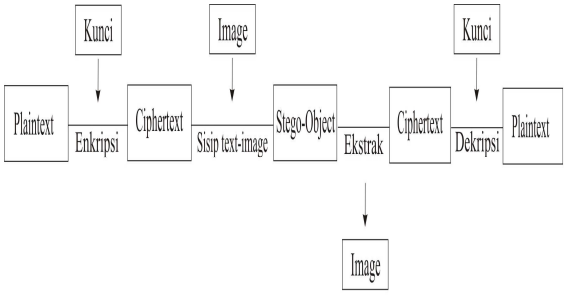


Gambar 1. Metode Penyembunyian Data

Gambar 1 sebagai contoh susunan bit pada *byte* yang menjelaskan bit yang cocok untuk digantikan adalah bit LSB, sebab penggantian hanyan mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Pada citra 24 bit setiap data piksel sudah mengandung komponen warna merah, hijau dan biru (RGB). Nilai dari bit-bit yang kurang berarti atau LSB dari setiap *byte* didalam citra digantikan dengan bit-bit pesan yang akan disembunyikan. Jika *byte* merupakan komponen hijau (G), maka penggantian satu bit LSB-nya hanya mengubah sedikit kadar warna hijau dan perubahannya tak terdeteksi oleh mata manusia.[4]

HASIL DAN PEMBAHASAN

Penerapan algoritma RC4 dan steganografi pada citra digital merupakan sistem penggabungan dua teknik pengamanan data yaitu kriptografi dan steganografi untuk meningkatkan keamanan dari kerahasiaan suatu pesan. Pada sistem ini kriptografi menggunakan algoritma *Rivest Code 4* (RC4) untuk proses enkripsi dan dekripsi, pesan rahasia yang telah dienkripsi di proses ke steganografi dengan metode *Least Significant Bit* (LSB). Berikut gambaran keseluruhan proses kerja sistem, [5]



Gambar 2. Sistem Algoritma RC4 dan Steganografi

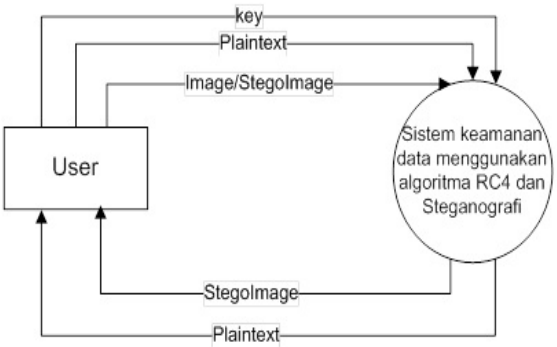
Pada gambar 2 merupakan gambaran keseluruhan sistem Aplikasi Algoritma RC4 pada Steganografi. Pihak pertama akan menginputkan pesan asli (*plaintext*), kunci dan gambar asli (*image*). Pesan asli (*plaintext*) dan kunci akan di enkripsi menggunakan algoritma RC4. Hasil dari enkripsi (*ciphertext*) disisipkan kedalam tiap bit LSB (*Least Significant Bit*) pada gambar asli (*image*). Gambar yang telah berisi pesan enkripsi RC4 (*stego-image*) akan diterima oleh pihak kedua yang memiliki hak akses pesan rahasia. Untuk mendapatkan kembali pesan asli (*plaintext*) pihak kedua dapat menginputkan *stego-image* yang diterima. *Stego Image* yang diterima pihak kedua kemudian diproses untuk mendapatkan kembali pesan (ekstrak) sehingga didapat pesan hasil enkripsi RC4 (*ciphertext*). Kemudian input kunci yang juga digunakan pada proses enkripsi, kunci yang diperoleh akan didekripsi dengan *ciphertext* menggunakan algoritma RC4.

Perancangan Sistem

Dalam tahap perancangan ini akan menjelaskan flowchart dari program, empat diagram konteks, yaitu level 0 menggambarkan garis besar program, level 1 pembagian tahapan kerja yang dapat dilakukan, level 2 tahapan kerja untuk Enkripsi RC4-Steganografi yang melakukan penyisipan pesan dan level 2 tahapan kerja Steganografi-Dekripsi RC4 untuk proses mengembalikan pesan rahasia dan dekripsi, terakhir perancangan desain program. Di bawah ini adalah penjelasan mengenai rancangan sistem pada Aplikasi algoritma *Rivest Code* 4 pada steganografi.

Diagram Konteks Level 0

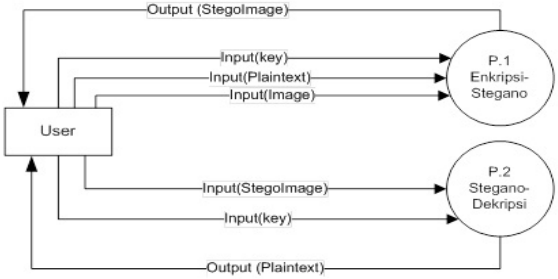
Diagram konteks level 0 merupakan sistem secara garis besar yang memeperlihatkan masukkan dan keluaran dari sistem yang dibuat. Pada level 0 pengguna akan memasukkan kunci, pesan dan gambar yang akan diguankan pada saat proses enkripsi dan steganografi sehingga keluarannya berupa *stegoimage*. Untuk membaca kembali pesan, pengguna dapat memasukkan kunci yang sama pada saat enkripsi dan gambar yang berisi pesan sehingga didapat keluaran berupa pesan asli atau plainteks.



Gambar 3. Diagram Konteks Level 0

Diagram Konteks Level 1

Pada diagram konteks level 1 menjabarkan pengembangan proses yang terjadi pada diagram konteks level 0. Dua tahap proses yang dilakukan, yaitu proses enkripsi-steganografi dan steganografi-dekripsi. Gambar diagram konteks level 1 sebagai berikut.



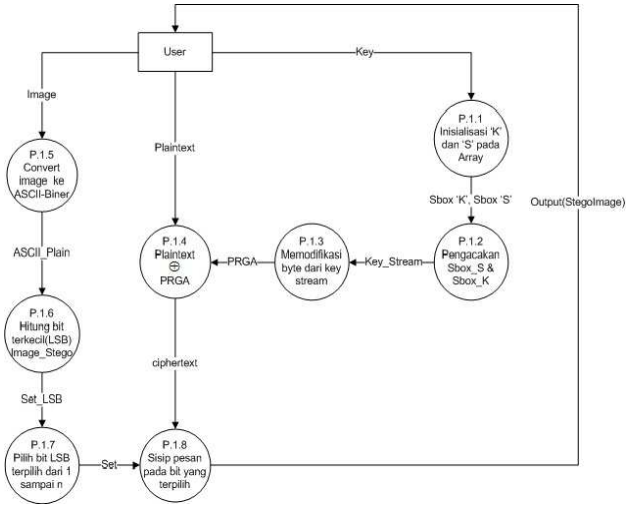
Gambar 4. Diagram Kontek Level 1

Pada gambar 4 terdapat 2 tahapan yang dapat dilakukan pada kriptosistem dan steganosistem yaitu:

1. Enkripsi_Steganografi
   Proses untuk mengubah pesan asli (*plaintext*) menjadi pesan rahasia (*ciphertext*). Hasil dari *ciphertext* akan disisipkan kedalam sebuah gambar (*Image*). Untuk melakukan proses enkripsi, *user* harus menginputkan *plaintext* , kunci (*key*) serta *image*.
2. Steganografi_Dekripsi
   Proses untuk memisahkan gambar berisi pesan (*StegoImage*) dari pesan rahasia (*ciphertext*) yang telah disisipkan dan mengubah *ciphertext* menjadi pesan asli (*plaintext*). Untuk melakukan proses dekripsi, *user* harus menginputkan gambar berisi pesan (*StegoImage*) dan kunci yang sama saat proses enkripsi.

Diagram Konteks Level2 (Enkripsi Steganografi)

Diagram konteks level 2 menjelaskan lebih rinci tahapan kerja enkripsi pesan menngunakan algoritma RC4. Pesan yang telah dienkripsi akan diproses melalui tahap penyisipan ke dalam suatu media penampung citra. Berikut rincian tahap kerja diagram konteks level 2 untuk proses enkripsi-steganografi.



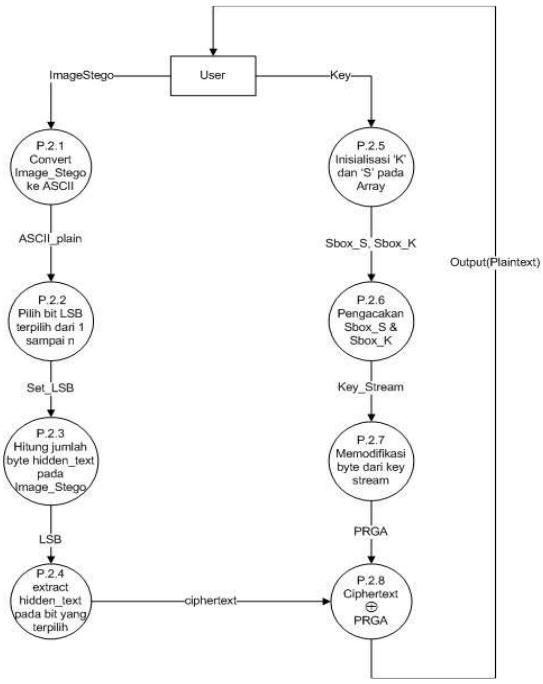
Gambar 5. Diagram Konteks Level 2 (Enkripsi-Steganografi)

Diagram Konteks Level2 (Steganografi Dekripsi)

Untuk mengembalikan cipherteks yang tersembunyi dalam media penampung menjadi plainteks, maka dilakukan proses ekstrak gambar dan dekripsi. Pada proses dekripsi digunakan kunci yang sama pada saat melakukan enkripsi. Penjelasan lebih rinci dapat dilihat pada diagram konteks level 2 steganografi-dekripsi sebagai berikut.

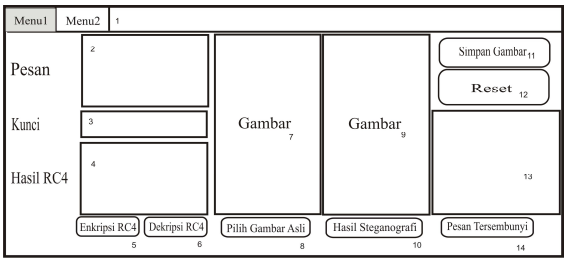
Perancangan Desain

Dalam tahap perencanaan desain, Aplikasi algoritma *Rivest Code* 4 pada Steganografi dibuat dalam dua *form*. Pertama menu1 yang akan menjalankan serangkaian proses aplikasi mulai dari pengen-kripsian *plaintext* hingga proses penyisipan pesan kedalam gambar. Dan sebaliknya, proses pendekripsian *ciphertext* dalam gambar. Menu 1 menyediakan keseluruhan proses kerja yang memudahkan pengguna untuk kembali mengecek hasil sistem sebelum menyimpan data. Desain menu1 dapat dilihat pada gambar 7 berikut.



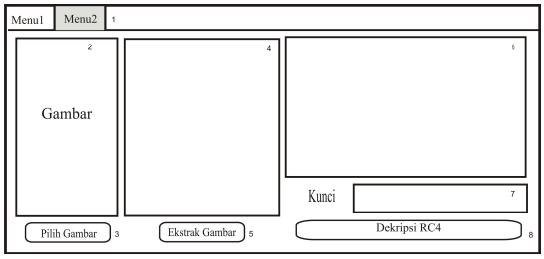
Gambar 6. Diagram Konteks Level 2 (Steganografi-Dekripsi)

Keterangan:
1. Nama *Form* Menu 1
2. *Box* untuk menuliskan pesan rahasia
3. *Input* kunci RC4
4. Hasil enkripsi atau dekripsi
5. Tombol proses enkripsi
6. Tombol proses dekripsi
7. Menampilkan gambar asli atau *cover text*
8. Tombol pilih gambar asli
9. Menampilkan gambar yang telah disisipkan pesan
10. Tombol proses penyisipan pesa ke dalam gambar
11. Tombol *save*
12. Tombol *reset*
13. Menampilakn pesan yang tersisip pada gambar
14. Tombol untuk melihat pesan yang tersembunyi di dalam gambar



Gambar 7. Desain form menu 1

Sedangkan Menu2 di desain hanya melakukan proses membaca pesan yang terdapat di gambar dan dekripsi pesan yang tersimpan. Menu2 digunakan untuk mengungkapkan kembali pesan yang telah disimpan sebelumnya pada tempat penyimpanan data. Desain menu2 dapat dilihat pada gambar 8 berikut.

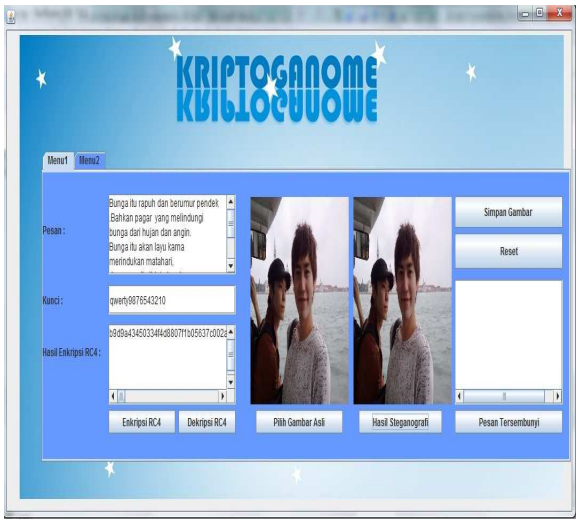


Gambar 8. Desain *form* menu 2

Gambar 8 desain *form* Menu2 terdiri dari beberapa komponen yang sama dengan Menu1. Di Menu2 terdapat *box* gambar yang akan menampilkan gambar yang telah dipilih melalui tombol pilih gambar. Gambar yang telah pilih akan diproses pada tombol ekstrak gambar untuk menampilkan *ciphertext* atau pesan yang tersembunyi. Pada *box* kunci dimasukkan kunci yang sama pada proses enkripsi, dan tombol dekripsi untuk mengembalikkan pesan yang terenkripsi.

PENGUJIAN SISTEM

Untuk mengetahui sistem telah bekerja dengan baik, maka perlu dilakukan pengujian terhadap sistem. Proses pengujian akan dibagi menjdi dua tahap yaitu, tahap pertama proses enkripsi RC4 dan steganografi penyisipan pesan sedangkan tahap kedua proses steganografi untuk mengembalikkan pesan dan dekripsi RC4.



Gambar 9. Proses Enkripsi RC4 – Steganografi

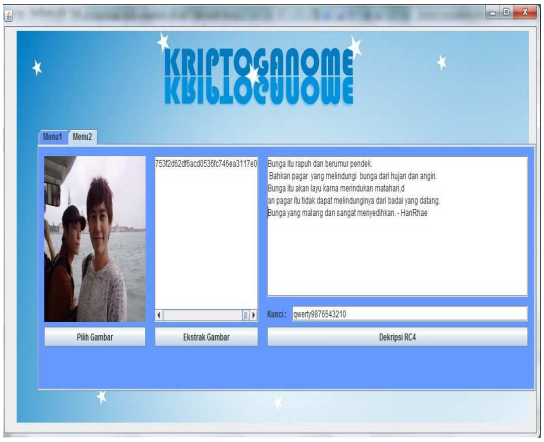
Enkripsi RC4 dan Steganografi

Pada pengujian enkripsi RC4 dan steganografi akan menggunakan *form* Menu1. *user* diminta untuk memasukkan kunci yang akan digunakan untuk enkripsi dan pesan yang ingin disampaikan. Setelah kunci dan pesan teks dimasukkan, klik tombol enkripsi rc4 dan sistem akan mulai melakukan proses algoritma RC4. Pesan yang telah dienkripsi

akan di tampilkan, untuk memastikan pesan dapat dibaca kembali atau didekripsi, dapat menekan tombol dekripsi. Pesan yang telah terenkripsi kemudian akan disisipkan kedalam media penampung citra. *user* dapat mengambil gambar berformat *bmp atau *png sebagai media penampung. Gambar yang telah terpilih akan dimasukkan hasil enkripsi RC4 dengan menekan tombol hasil stegnaografi. Gambar 9 menujukkan proses enkripsi rc4 dan steganografi.

Steganografi dan Dekripsi RC4

Proses pemisahan atau ekstraksi pesan rahasia (cipherteks) dengan *file* gambar (stegoimage) akan ditampilkan pada *form* Menu2. Pada menu2 hanya menyediakan proses ekstrak dan dekripsi pesan. Untuk memulai proses *user* dapat mengambil gambar yang akan di ekstrak, gambar yang berisi hasil enkripsi. Gambar yang telah dipilih akan di ekstrak dan di dekripsi dengan memasukkan kunci yang sama saat proses enkripsi. Hasil ekstrak dan dekripsi rc4 dapat dilihat pada gambar 10.



Gambar 10. Proses Steganografi-Dekripsi RC4

PEMBAHASAN

Pembahasan ini berisi hasil pengujian aplikasi keamanan data dengan menggunakan algoritma RC4 dan steganografi pada citra digital. Pengujian dilakukan untuk membandingkan hasil gambar yang telah disispkan cipherteks dan perubahan pesan dari dalam gambar setelah dimanipulasi. Pengujian dilakukan dengan menggunakan plainteks yang sama dan kunci "qwerty9876543210", sedangkan media penampung dengan format gambar *bmp. Hasil pengujian beberapa citra dapat dilihat pada tabel 2.

Dari hasil pengujian sistem di tabel 2, dilakukan beberapa proses manipulasi terhadap gambar untuk mengetahui tingkat ketahanan gambar. Dan hasil yang diperoleh gambar yang telah tersisip pesan rentan terhadap operasi manipulasi.

KESIMPULAN

Kesimpulan yang dapat diambil berdasarkan penelitian mengenai penerapan algoritma RC4 pada steganografi citra digital yaitu penggunaan dua teknik pengamanan data pada kriptografi algoritma RC4 dan steganografi yang menggunakan metode LSB, dapat digunakan untuk meningkatkan keamanan data.

DAFTAR PUSTAKA

[1] Munir, R. 2004. *Bahan Kuliah IF5054 Kriptografi.* Institut Teknologi Bandung: Departemen Teknik Informatika.

[2] Sutiono, A. P. 2010. *Algoritma RC4 Sebagai Perkembangan Metode Kriptografi*. Bandung: Institut Teknologi Bandung.

[3] Wiryawan, I. G. 2011. *Membangun Aplikasi Steganografi Dengan Metode Least Significant Bit (LSB).* Skripsi Sarjana Sains Bidang Ilmu Komputer, Universitas Mulawarman.

[4] Rahim, M. 2006. *Teknik Penyembunyian Data Rahasia Dengan Menggunakan Citra Digital Sebagai Berkas Penampung*. Semarang : Universitas Diponegoro.

[5] Hendrawati. *Keamanan Data Dengan Menggunakan Algoritma Rivest Code 4 (RC4) Dan Steganografi Pada Citra Digital*. Penelitian Ilmu Komputer Universitas Mulawarman. Samarinda