Jurnal Ilmiah
ILMU KOMPUTER
Universitas Udayana

Vol. 8, No. 2, September 2015 ISSN 1979-5661

IMPLEMENTASI ALGORITMA KRIPTOGRAFI AES 256 DAN METODE STEGANOGRAFI LSB PADA GAMBAR BITMAP

Gede Wisnu Bhaudhayana[1], I Made Widiartha[2]
Program Studi Teknik Informatika, Jurusan Ilmu Komputer,
Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Udayana.
Email: wisnu.bhaudhayana@gmail.com, madewidiartha@gmail.com

ABSTRAK

Dalam penggunaan teknologi sehari-hari, manusia tidak terlepas dari yang namanya internet sebagai kebutuhan untuk saling bertukar informasi. Orang-orang pada saat ini lebih banyak menghabiskan waktunya di depan layar komputer, laptop, bahkan smartphone untuk mengetahui informasi, mengirim data ke beberapa teman-teman mereka. Salah satu informasi yang sering dicari ataupun dikirim adalah file. Terlepas dari itu file gambar merupakan file yang banyak dicari dan dikirim dan banyak juga mengandung informasi-informasi penting di dalamnya. Keamanan file gambar tentu menjadi sangat penting agar tidak adanya pihak-pihak yang tidak berwenang meretas atau memanipulasi informasi dari gambar tersebut. Ada cara untuk mengamankan suatu informasi agar informasi itu tidak bocor kepada pihak yang tidak berwenang, yaitu dengan menggunakan kriptografi dan steganografi.

Kriptografi digunakan untuk mengubah pesan rahasia yang dapat dimengerti menjadi sebuah pesan yang tidak dapat dimengerti lagi. Sedangkan steganografi digunakan untuk menyisipkan sebuah pesan rahasia ke dalam media penampung sehingga seseorang tidak akan menyadari letak pesan rahasia tersebut. Dengan menggabungkan kedua cara ini, dapat menjaga kerahasiaan dan keamanan terhadap sebuah file terutama file gambar. Dalam penelitian ini algoritma dan metode yang digunakan adalah algoritma kriptografi AES (*Advanced Encryption Standard*) 256 dan metode steganografi LSB (*Least Significant Bit*). Integritas data perlu diuji untuk memastikan bahwa proses enkripsi dan dekripsi telah berjalan baik. Pengujian integritas data menggunakan metode SHA-1. Begitu pula kualitas gambar setelah penyisipan akan mengalami penurunan kualitas. Untuk mengevaluasi hal itu perlu dilakukan pengujian dengan menggunakan metode PSNR.

Dari hasil pengujian integritas data dengan membandingkan nilai hash file gambar hasil dekripsi dengan file gambar asli tidak ada perbedaan. Sehingga ini menunjukkan bahwa proses enkripsi dan dekripsi berhasil. Sedangkan pada pengujian menggunakan metode PSNR, rata-rata nilai PSNR adalah 44,14086 dB dan dengan nilai error rata-rata adalah 2,830403 dB yang berarti mengalami penurunan kualitas yang kecil. Dari hasil implementasi dan pengujian didapat kesimpulan bahwa algoritma kriptografi AES 256 dan metode steganografi LSB dapat diimplementasikan dalam menjaga kerahasiaan dan keamanan pesan rahasia.

Kata kunci: kriptografi, steganografi, algoritma AES, metode LSB, SHA-1, PSNR

ABSTRACT

In everyday use of technology, humans can not be separated from the name of the internet as the need to exchange information. People today spend more time in front of computer screens, laptops, and even a smartphone to find information, send data to some of their friends. One of the information sent is often sought or file. Regardless of the image file is a file that is much sought after and sent, and many also contain important information in it. Security image file would be very important that the absence of parties who are not authorized to hack or manipulate information from the image. There are ways to safeguard the information that it does not leak information to unauthorized parties, namely by using cryptography and steganography.

Cryptography is used to change the secret messages that can be understood into a message that can not be understood anymore. While steganography is used to insert a secret message into the media reservoir so that a person will not be aware of the location of the secret message. By combining both in this way, to protect confidentiality and security to a file especially image files. In this study the algorithms and methods used is a cryptographic algorithm AES (Advanced Encryption Standard) 256 and the method of steganography LSB (Least Significant Bit). Data integrity needs to be tested to ensure that the encryption and decryption process has

been going well. Data integrity testing using SHA-1. Similarly, the quality of the image after insertion will decrease the quality. To evaluate it needs to be tested using the method PSNR.

From the results of testing the integrity of the data by comparing the hash value of the decrypted image file with the original image file is no difference. So this shows that the encryption and decryption process successfully. While on testing using methods PSNR, the average PSNR value is 44,14086 dB and with the average value of error is 2,830403 dB, which means decreased quality small. From the results of the implementation and testing concluded that cryptographic algorithm AES 256 and LSB steganography method can be implemented in maintaining the confidentiality and security of confidential messages.

Keywords: cryptography, steganography, AES algorithm, LSB method, SHA-1, PSNR.

1. Pendahuluan

1.1 Latar Belakang

Seiring perkembangan zaman, saat ini manusia sedang memasuki era internet. Pertukaran informasi saat ini berlangsung tidak hanya dalam kehidupan sehari-hari tetapi juga di dunia maya. Informasi penting berupa pesan dapat dikirim dan diterima melalui internet pada saat ini. Kini pesan yang sering dikirim dan diterima melalui e-mail, jejaring sosial seperti Facebook, Twitter, dan lain sebagainya banyak berupa gambar. Gambar juga mengandung informasi yang penting pada sebagian orang untuk saling bertukar data.

Seiring dengan pesatnya kebutuhan informasi pada manusia seperti yang terjadi pada saat ini tentu diperlukan kemanan terhadap pesan yang dikirim maupun diterima. Keamanan tersebut diperlukan untuk menghindari adanya penyadapan atau pembajakan terhadap gambar yang mengandung informasi penting bagi penggunanya. Keamanan diperlukan untuk menjaga integritas gambar tersebut agar tetap aman.

Terdapat beberapa cara untuk mengamankan pesan gambar tersebut, yaitu dengan kriptografi dan steganografi. Kriptografi dan steganografi dapat diimplementasikan pada aplikasi untuk mengamankan gambar.

1.2 Identifikasi Masalah

1. Pertukaran informasi berupa pesan gambar yang terjadi di dunia maya semakin berkembang seiring dengan kebutuhan informasi manusia.
2. Pertukaran informasi yang tak terbatas berdampak terhadap adanya tindak kejahatan dunia maya berupa penyadapan dan pembajakan.
3. Pentingya sebuah keamanan terhadap informasi yang dikirim maupun diterima dengan cara kriptografi dan steganografi.

2. Landasan Teori

2.1 Kriptografi

Kriptografi berasal dari dua kata Yunani, yaitu *Crypto* yang berarti rahasia dan *Grapho* yang berarti menulis. Secara umum kriptografi dapat diartikan sebagai ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu data. Kriptografi mendukung kebutuhan dari dua aspek keamanan informasi, yaitu *secrecy* (perlindungan terhadap kerahasiaan data informasi) dan *authenticity* (perlindungan terhadap pemalsuan dan pengubahan informasi yang tidak diinginkan). Kriptografi tidak berarti hanya memberikan keamanan informasi saja, namun lebih ke arah teknik-tekniknya. (Wahyudi & DP, 2008)

Algoritma-algoritma kriptogrfi dapat dibedakan menjadi dua macam yaitu simetrik dan asimetrik. Algoritma simetrik (model enkripsi konvensional) merupakan algoritma yang menggunakan satu kunci untuk proses enkripsi dan deskripsi data. Sedangkan algoritma asimetrik (model enkripsi kunci publik) menggunakan kunci yang berbeda dalam proses enkripsi dan deskripsi pesan.

2.2 Algoritma AES

Algoritma AES merupakan algoritma *chiper* yang aman untuk melindungi data atau informasi yang bersifat rahasia. AES dipublikasikan oleh NIST (National Institute of Standard and Technology) pada tahun 2001 yang digunakan untuk menggantikan algoritma DES yang sudah dianggap kuno dan mudah dibobol.

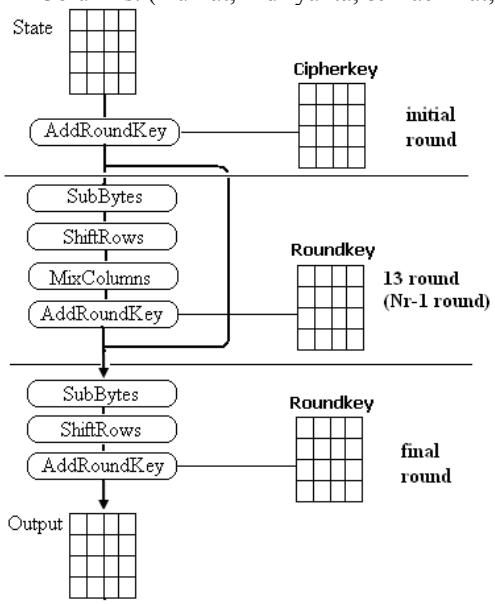
Input dan output dari algoritma AES terdiri dari urutan data sebesar 128 bit. Urutan data dalam satu kelompok 128 bit tersebut disebut juga sebagai blok data atau plaintext yang nantinya akan dienkripsi menjadi chipertext. Panjang kunci dari AES terdiri dari panjang kunci 128 bit, 192 bit, dan 256 bit. Perbedaan panjang kunci ini yang nantinya mempengaruhi jumlah putaran pada algoritma AES ini. Input dan output dari algoritma AES terdiri dari urutan data sebesar 128 bit. Urutan data dalam satu kelompok 128 bit tersebut disebut juga sebagai blok data atau plaintext yang nantinya akan dienkripsi menjadi chipertext. Panjang kunci dari AES terdiri dari panjang kunci 128 bit, 192 bit, dan 256 bit. Perbedaan panjang kunci ini yang nantinya mempengaruhi jumlah putaran pada algoritma AES ini. Jumlah putaran yang digunakan algoritma ini ada tiga macam seperti pada tabel di bawah.

Tabel 1 Jumlah Putaran pada Algoritma AES (Sumber: Yuniat, Indriyanta, & Rachmat, 2009)

Tipe	Panjang Kunci	Panjang Blok Input	Jumlah Putaran
AES-128	128 bit	128 bit	10
AES-192	192 bit	128 bit	12
AES-256	256 bit	128 bit	14

2.2.1 Proses Enkripsi AES

Proses enkripsi algoritma AES terdiri dari 4 jenis transformasi bytes, yaitu SubBytes, ShiftRows, Mixcolumns, dan AddRoundKey. Pada awal proses enkripsi, input yang telah dicopykan ke dalam state akan mengalami transformasi byte AddRoundKey. Setelah itu, state akan mengalami transformasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey secara berulang-ulang sebanyak Nr. Proses ini dalam algoritma AES disebut sebagai round function. Round yang terakhir agak berbeda dengan round-round sebelumnya dimana pada round terakhir, state tidak mengalami transformasi MixColumns. (Yuniat, Indriyanta, & Rachmat, 2009)



Gambar 1 Proses Enkripsi AES (Yuniat, Indriyanta, & Rachmat, 2009)

a. AddRoundKey
Dalam initial round, transformasi AddRoundKey() dilakukan terhadap kunci utama. Sedangkan dalam 10 round yang lain, proses AddRoundKey dilakukan terhadap kunci putaran (round key). Proses AddRoundKey didefinisikan sebagai operasi XOR antara array state dengan round key. Operasi XOR dilakukan pada masing-masing byte dalam array sehingga menghasilkan nilai baru pada array hasil dengan ukuran array hasil sama dengan ukuran array state awal dan array key, yaitu sebesar 4x4. Hasil untuk masing-masing baris dan kolom pada array state hasil diperoleh dari hasil operasi XOR antara array state awal dengan array key untuk baris dan kolom yang sama.

b. SubBytes
Transformasi SubBytes() memetakan setiap byte dari array state dengan menggunakan tabel substitusi S-Box. Tabel S-Box dapat dilihat pada berikut.

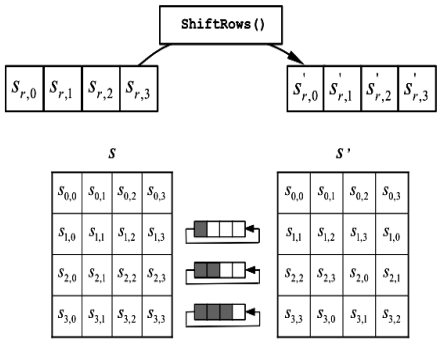
HEX	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

x

Gambar 2 S-Box (Yuniat, Indriyanta, & Rachmat, 2009)

Cara pensubstitusian adalah sebagai berikut: untuk setiap byte pada array, misalkan S[r,c] = xy, yang dalam hal ini xy adalah digit heksadesimal dari nilai S[r,c], maka nilai substitusinya, yang dinyatakan dengan S'[r,c], adalah elemen di dalam S-Box yang merupakan perpotongan baris x dengan kolom y.

c. ShiftRows
Transformasi ShiftRows() melakukan pergeseran secara wrapping (siklik) pada 3 baris terakhir dari array state. Jumlah pergeseran bergantung pada nilai baris (r). Baris r = 1 digeser sejauh 1 byte, baris r = 2 digeser sejauh 2 byte, dan baris r = 3 digeser sejauh 3 byte. Baris r = 0 tidak digeser.



Gambar 3 Transformasi ShiftRows (Yuniat, Indriyanta, & Rachmat, 2009)

d. MixColumns

Transformasi MixColumns() dilakukan setelah transformasi ShiftRows, merupakan sumber utama dari difusi pada algoritma AES. Difusi merupakan prinsip yang menyebarkan pengaruh satu bit plaintext atau kunci ke sebanyak mungkin ciphertext.

Transformasi MixColumns() mengalikan setiap kolom dari array state dengan polinom a(x) mod (x4 + 1). Setiap kolom diperlakukan sebagai polinom 4 suku pada GF (28). Polinom a(x) yang ditetapkan pada persamaan 1

$a(x) = \{03\}x^3 + \{01\}\ x^2 + \{01\}\ x + \{02\}$ (1)

Transformasi ini dinyatakan sebagai perkalian matriks seperti pada persamaan 2

$s'(x) = a(x) \otimes s(x)$ (2)

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Hasil dari perkalian matriks tersebut, setiap *byte* dalam kolom *array state* akan digantikan dengan nilai baru. Persamaan matematis untuk setiap *byte* tersebut pada persamaan 3

$s'0,c = (\{02\} \bullet s0,c) \oplus (\{03\} \bullet s1,c) \oplus s2,c \oplus s3,c$

$s'1,c = s0,c \oplus (\{02\} \bullet s1,c) \oplus (\{03\} \bullet s2,c) \oplus s3,c$

$s'2,c = s0,c \oplus s1,c \oplus (\{02\} \bullet s2,c) \oplus (\{03\} \bullet s3,c)$

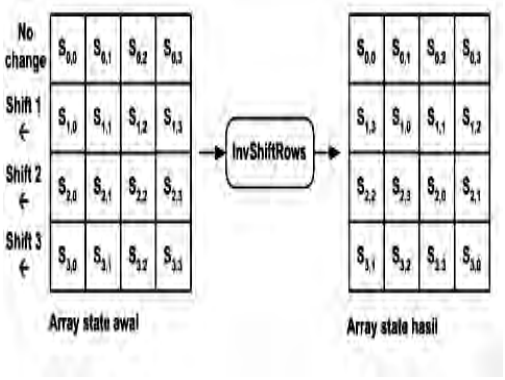
$s'3,c = (\{03\} \bullet s0,c) \oplus s1,c \oplus s2,c \oplus (*02\} \bullet s3,c)$ (3)

2.2.2 Proses Dekripsi AES

Transformasi cipher dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan inverse cipher yang mudah dipahami untuk algoritma AES. Transformasi byte yang digunakan pada invers cipher adalah *InvShiftRows*, InvSubBytes, InvMixColumns, dan AddRoundKey. Algoritma dekripsi dapat dilihat pada skema berikut ini :

a. InvShiftRows

InvShiftRows adalah transformasi byte yang berkebalikan dengan transformasi ShiftRows. Pada transformasi *InvShiftRows*, dilakukan pergeseran bit ke kanan sedangkan pada ShiftRows dilakukan pergeseran bit ke kiri. Ilustrasi transformasi *InvShiftRows* terdapat pada gambar berikut.



Gambar 4 Transformasi InvShiftRows (Yuniat, Indriyanta, & Rachmat, 2009)

b. InvSubBytes

InvSubBytes juga merupakan transformasi bytes yang berkebalikan dengan transformasi SubBytes. Pada *InvSubBytes*, tiap elemen pada state dipetakan dengan menggunakan tabel Inverse S-Box. Tabel Inverse S-Box akan ditunjukkan dalam tabel berikut.

HEX	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	D5	30	36	A5	38	Bf	40	A3	9e	81	F3	D7	Fb
1	7c	E3	39	82	9b	2f	Ff	87	34	8e	43	44	C4	De	E9	Cb
2	54	7b	94	32	A6	C2	23	3d	Ee	4c	95	0b	42	Fa	C3	4e
3	08	2e	A1	66	28	D9	24	B2	76	5b	A2	49	6d	8b	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5c	Cc	5d	65	B6	92
5	6c	70	48	50	Fd	Ed	B9	Da	5e	15	46	57	A7	8d	9d	84
6	90	D8	Ab	00	8c	Bc	D3	0a	F7	E4	58	05	B8	B3	45	06
7	d0	2c	1e	8f	Ca	3f	0f	02	C1	Af	Bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	Dc	Ea	97	F2	Cf	Ce	F0	B4	E6	73
9	96	Ac	74	22	E7	Ad	35	85	E2	F9	37	E8	1c	75	Df	6e
a	47	F1	1a	71	1d	29	C5	89	6f	B7	62	0e	Aa	18	Be	1b
b	fc	56	3e	4b	C6	D2	79	20	9a	Db	C0	Fe	78	Cd	5a	F4
c	1f	Dd	A8	33	88	07	C7	31	B1	12	10	59	27	80	Ec	5f
d	60	51	7f	A9	19	B5	4a	0d	2d	E5	7a	9f	93	C9	9c	Ef
e	A0	E0	3b	4d	Ae	2a	F5	B0	C8	Eb	Bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	D6	26	E1	69	14	63	55	21	0c	7d

x

Gambar 5 Inverse S-Box (Yuniat, Indriyanta, & Rachmat, 2009)

c. InvMixColumns

Setiap kolom dalam state dikalikan dengan matrik perkalian dalam AES. Perkalian dalam matrik dapat dituliskan :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0B & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Hasil dari perkalian matriks tersebut, setiap byte dalam kolom array state akan digantikan dengan nilai baru. Persamaan matematis untuk setiap byte tersebut pada persamaan 2.9

s'0,c =({0E}• s0,c)⊕({0B}• s1,c)⊕({0D}•s2,c) ⊕({09}• s3,c)s'1,c = ({09}•s0,c)⊕ ({0E} • s1,c) ⊕ ({0B} • s2,c) ⊕ ({0D}•s3,c)s'2,c = ({0D}•s0,c)⊕ ({09}•s1,c)⊕ ({0E}• s2,c)⊕ ({0B}• s3,c)s'3,c = ({0B}•s0,c) ⊕ ({0D}•s1,c)⊕ ({09}•s2,c) ⊕(*0E} • s3,c) (2.9)

2.3 Steganografi

Steganografi (steganography) berasal dari bahasa Yunani yaitu "steganos" yang berarti "tersembunyi" atau "terselubung", dan "graphein" yang artinya "menulis". Steganografi dapat diartikan "tulisan tersembunyi" (covered writing). Steganografi adalah ilmu dan seni menyembunyikan pesan rahasia di dalam pesan lain sehingga keberadaan pesan rahasia tersebut tidak dapat diketahui. Steganografi membutuhkan dua properti, yaitu media penampung dan pesan rahasia. Media penampung yang umum digunakan adalah gambar, suara, video, atau teks. Pesan yang disembunyikan dapat berupa sebuah artikel, gambar, kode program, atau pesan lain. Proses penyisipan pesan ke dalam media covertext dinamakan encoding, sedangkan ekstraksi pesan dari stegotext dinamakan decoding. Kedua proses ini mungkin memerlukan kunci rahasia (yang dinamakan stegokey) agar hanya pihak yang berhak saja yang dapat melakukan penyisipan pesan dan ekstraksi. (Rakhmat & Fairuzabadi, 2010).

2.4 Metode LSB

Metode LSB merupakan metode steganografi yang paling sederhana dan mudah diimplementasikan. Metode ini menggunakan citra digital sebagai covertext. Pada susunan bit di dalam sebuah byte (1 byte = 8 bit), ada bit yang paling depan (most significant bit atau MSB) dan bit yang paling akhir (least significant bit atau LSB). Sebagai contoh byte 11010010, angka bit 1 (pertama, digaris-bawahi) adalah bit MSB, dan angka bit 0 (terakhir, digaris-bawahi) adalah bit LSB. Bit yang cocok untuk diganti adalah bit LSB, sebab perubahan tersebut hanya mengubah nilai byte satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan byte tersebut menyatakan warna merah, maka perubahan satu bit LSB tidak mengubah warna merah tersebut secara berarti. Mata manusia tidak dapat membedakan perubahan kecil tersebut.

2.5 SHA (Secure Hashing Algorythm)-1

SHA-1 adalah pengembangan dari SHA-0 dimana SHA-1 memperbaiki kelemahan yang ada di SHA-0. SHA-1 merupakan fungsi hash yang paling populer dibandingkan dengan fungsi hash SHA lainnya. SHA-1 memproduksi 160bit digest berdasarkan prinsip yang sama dengan algoritma MD4 dan MD5 namun dengan design yang berbeda. SHA-1 mempunyai kapasitas input message 2^{64}-1, dengan hasil hash 160 bits dan evaluasi kekuatan hash 2^{80} Misal SHA-1 digunakan untuk meng-hash sebuah pesan, M, yang mempunyai panjang maksimum 2^{64}-1 bits. Algoritma ini menggunakan urutan dari 80 kali 32-bit kata, dengan menggunakan 5 variabel yang menampung 32 bits per variabel, dan hasil hashnya. (Huda W, 2003)

2.6 PSNR (*Peak Signal to Noise Ratio)*

Kualitas media penampung setelah ditambahkan pesan rahasia tidak jauh berbeda dengan kualitas media penampung sebelum ditambahkan pesan. Setelah penambahan pesan rahasia, kualitas citra penampung tidak jauh berubah, masih terlihat dengan baik. Untuk mengukur kualitas citra steganografi diperlukan suatu pengujian secara obyektif. Pengujian secara objektif adalah dilakukan dengan menghitung nilai PSNR.

Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR diukur dalam satuan desibel. Pada penelitian ini, PSNR digunakan untuk mengetahui perbandingan kualitas gambar sebelum dan sesudah disisipkan pesan. Untuk menentukan PSNR, terlebih dahulu harus ditentukan MSE (*Mean Square Error*). MSE secara matematis dapat dirumuskan sebagai berikut:

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2$$

Dimana :

MSE = Nilai Mean Square Error citra steganografi
m = Panjang citra stego (dalam pixel)
I(i,j) = nilai piksel dari citra cover
n = Lebar citra stego (dalam pixel)
K(i,j) = nilai piksel pada citra stego

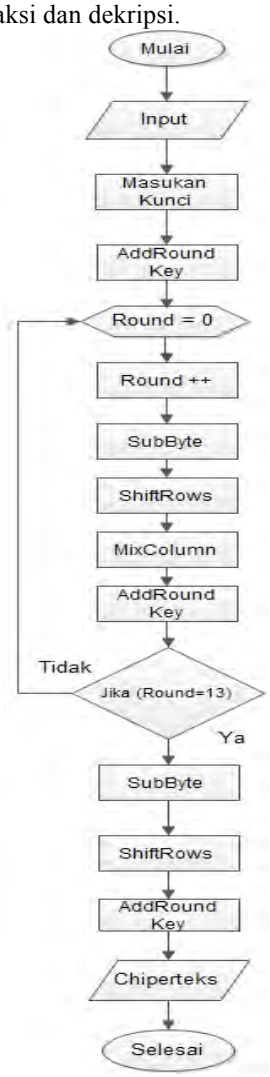
Setelah diperoleh nilai MSE maka nilai PSNR dapat dihitung dari kuadrat nilai maksimum dibagi dengan MSE. Secara matematis, nilai PSNR dirumuskan sebagai berikut :

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

Dimana:
MSE = nilai *MSE*, *MAXi* = nilai maksimum dari pixel citra yang digunakan. Semakin rendah Nilai MSE maka akan semakin baik, dan semakin besar nilai PSNR maka semakin baik kualitas citra steganografi. (Moenandar, Wirawan, & Setijadi, 2012)

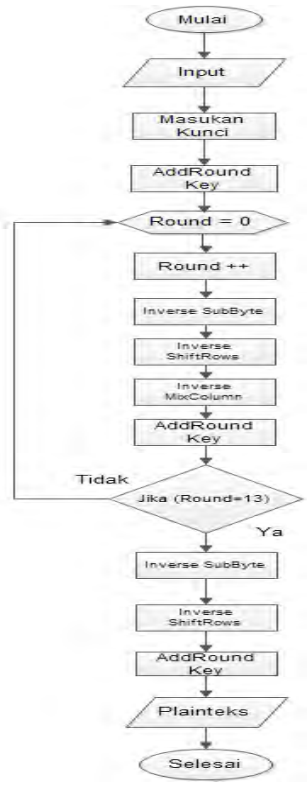
3. Perancangan

Untuk membuat sebuah aplikasi yang dapat mengamankan file gambar. Tentu perlu adanya rancangan dalam tahapan-tahapan proses. Tahapan-tahapan tersebut akan direpresentasikan dalam diagram alir atau yang biasa disebut *flowchart*. Pada flowchart menjelaskan proses pada tahapan enkripsi dan penyisipan. Begitu juga untuk proses ekstraksi dan dekripsi.



Gambar 6 Flowchart Enkripsi Algoritma AES 256

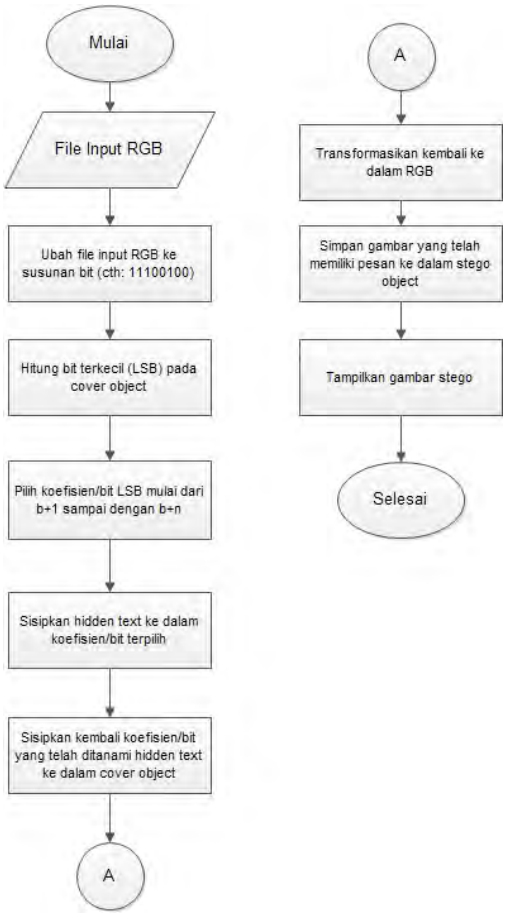
Diagram alir pada gambar di atas menjelaskan proses enkripsi menggunakan algoritma kriptografi AES. Pada langkah awal dalam melakukan proses enkripsi sebuah data harus melakukan input data yang akan dienkripsi. Pada penelitian ini data berupa gambar. Selanjutnya adalah melakukan proses masukan kunci. Karena yang digunakan dalam penelitian ini adalah algoritma AES 256 maka kunci sepanjang 256 bit. Langkah selanjutnya melakukan proses AddRoundKey. Proses AddRoundKey ini dilakukan operasi XOR antara plainteks dengan kunci. Proses enkripsi ini terdiri dari 4 jenis transformasi bytes, yaitu SubBytes, ShiftRows, Mixcolumns, dan AddRoundKey. Pada awal proses enkripsi atau Round = 0, input akan mengalami transformasi byte AddRoundKey. Setelah itu, state akan mengalami transformasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey secara berulang-ulang sebanyak Round = 13, Jika Round = 14 maka akan melakukan tiga proses transformasi SubBytes, ShiftRows, dan AddRoundKey dan akan menghasilkan cipherteks. Proses enkripsi selesai.



Gambar 7 Flowchart Deksripsi Algoritma AES 256

Diagram alir pada gambar di atas menjelaskan proses dekripsi menggunakan algoritma kriptografi AES. Pada langkah awal dalam melakukan proses dekripsi sebuah chiperteks harus melakukan input chiperteks yang akan didekripsi. Dari hasil cipherteks dan dari proses kunci random akan dilakukan penggabungan kunci dalam satu blok yaitu dengan melakukan proses AddRoundKey yaitu cipherteks ditambahkan pada state dengan operasi XOR dengan kunci. Proses

dekripsi ini terdiri dari 4 jenis transformasi bytes, yaitu InverseSubByte, InverseShiftRows, InverseMixcolumn, dan AddRoundKey. Pada awal proses dekripsi atau Round = 0, input yang telah state akan mengalami transformasi byte AddRoundKey. Setelah itu, state akan mengalami transformasi InverseSubBytes, InverseShiftRows, InverseMixColumns, dan AddRoundKey secara berulang-ulang sebanyak Round = 13, Jika Round = 14 maka akan melakukan tiga proses transformasi InverseSubBytes, InverseShiftRows, dan AddRoundKey dan akan menghasilkan plainteks atau data asli. Proses dekripsi selesai.



Gambar 8 Flowchart Proses Embed Metode LSB

Diagram alir pada gambar di atas merupakan diagram alir dari proses embed atau penyisipan pesan ke dalam gambar atau media penampung menggunakan metode LSB. Langkah awal dimulai dengan input file RGB (file gambar bitmap). Selanjutnya file input RGB tersebut diubah ke dalam bentuk susunan bit (contoh: 11100100). Selanjutnya menghitung bit terkecil (LSB) pada gambar cover. Proses selanjutnya adalah memilih koefisien bit LSB mulai dari b+1 sampai dengan b+n untuk disisipkan *hidden text* (pesan). Setelah itu sisipkan hidden text ke dalam koefisien bit terpilih dan disisipkan kembali ke dalam gambar cover. Setelah itu transformasikan kembali ke dalam nilai RGB yang baru dan simpan gambar yang telah memiliki pesan di dalamnya sebagai gambar stego.

Gambar 9 Flowchart Proses Ekstraksi Metode LSB

Diagram alir di atas merupakan diagram alir dari proses ekstraksi menggunakan metode LSB. Langkah awal dimulai dari input file RGB. Selanjutnya mengubah file RGB ke dalam bentuk susunan bit (contoh: 11100100). Langkah selanjutnya memilih koefisien/bit LSB terpilih dari b+1 sampai dengan b+n. Selanjutnya menghitung jumlah byte hidden text pada gambar cover. Proses selanjutnya adalah melakukan ekstrak hidden text bit terpilih dalam gambar object. Maka akan menghasilkan pesan yang sebelumnya disisipkan pada gambar cover.

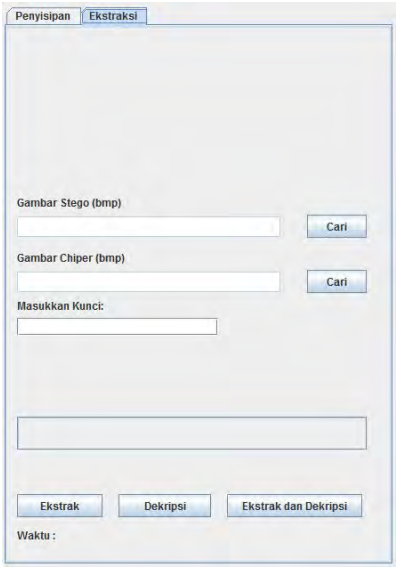
4. Implementasi dan Hasil

4.1 Implementasi

Untuk dapat membuat aplikasi pengamanan file gambar dengan algoritma kriptografi AES dan metode steganografi LSB, maka implementasinya dibuat dalam bentuk program dengan menggunakan bahasa *java* dan tools *netbean*.

Gambar 10 Tampilan Program Enkripsi Dan Penyisipan

Gambar di atas merupakan tampilan program untuk proses enkripsi dan penyisipan. Untuk melakukan proses enkripsi dan penyisipan pertama ambil gambar. Gambar asli sebagai pesan yang akan dienkripsi dan gambar penampung sebagai media untuk menampung pesan gambar. Kedua gambar berformat bmp. Setelah itu masukkan kunci sebagai chiperkey. Selanjutnya tekan tombol embed dan enkripsi untuk mendapatkan hasil program berupa gambar yang telah disisipkan pesan rahasia.



Gambar 11 Tampilan Program Dekripsi Dan Ekstraksi

4.2 Hasil

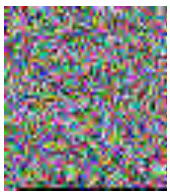
Untuk mengetahui hasil dari aplikasi pengamanan file gambar maka perlu dilakukan percobaan untuk mengetahui hasil dari setiap proses yang ada, yaitu dari proses enkripsi, penyisipan, ekstraksi, dan dekripsi.

4.2.1 Hasil Enkripsi

Setelah dilakukan percobaan terhadap sebuah file gambar berformat bmp untuk proses enkripsi maka didapatkan hasil sebagai berikut. Contoh dilakukan percobaan untuk proses enkripsi terhadap file airplane.bmp dengan resolusi 55 x 55 pixel dan size 9,07 kb.

Gambar 12 Gambar Asli Sebelum Dienkripsi

Gambar di atas merupakan gambar asli atau pesan sebelum dilakukan proses enkripsi.. Setelah dienkripsi gambar asli tersebut akan tampak seperti berikut.



Gambar 13 Gambar Setelah Dienkripsi

Gambar di atas merupakan tampilan gambar setelah dienkripsi. Tampak seperti gambar yang hancur dan tak berarti. Setelah dilakukan proses enkripsi gambar memiliki resolusi 55 x 56 pixel dengan ukuran file 9,24 kb.

Tabel 2 Tabel Hasil Enkripsi

Nama File	Size	Waktu Eksekusi
UNIX-BMP.bmp	3,05 kb	1359 ms
UNIX-BMP1.bmp	4,74 kb	2332 ms
Add_record2561.bmp	6, 02 kb	2562 ms
Add_record256.bmp	7,47 kb	3335 ms
airplane1.bmp	7,97 kb	3436 ms
airplane.bmp	9,07 kb	3718 ms
happy.bmp	10,5 kb	4803 ms
pepper.bmp	11,4 kb	4409 ms
happy1.bmp	12,0 kb	5392 ms
pepper1.bmp	14,5 kb	6341 ms

Dari percobaan enkripsi terhadap beberapa gambar, didapat hasil seperti tabel di atas. Pada tabel di atas terdapat beberapa gambar dengan ukuran yang berbeda-beda. Dilakukan penambahan 10% ukuran gambar dari gambar yang mempunyai ukuran atau *size* terkecil. Data dari tabel di atas menyatakan semakin besar ukuran file gambar, maka semakin besar pula waktu yang diperlukan dalam eksekusi.

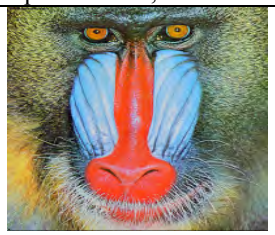
4.2.2 Hasil Penyisipan

Setelah melakukan percobaan untuk proses enkripsi maka selanjutnya dilakukan percobaan proses *embedding* atau penyisipan gambar pesan ke gambar penampung. Untuk melakukan penyisipan, gambar penampung harus memiliki resolusi dan *size* yang lebih besar terhadap gambar pesan. Setelah dilakukan percobaan proses penyisipan didapatkan hasil sebagai berikut.

Gambar 14 Gambar Penampung

Gambar di atas merupakan gambar penampung yang akan disisipkan pesan berupa gambar pesan atau gambar *chiper*. Gambar penampung tersebut memiliki resolusi 315 x 315 pixel dengan ukuran file 291. Setelah disisipkan pesan gambar *chiper* dengan nama file airplane.bmp maka didapatkan hasil sebagai berikut.

Nama File	Size	Waktu Eksekusi
UNIX-BMP.bmp	3,14 kb	1973 ms
UNIX-BMP1.bmp	4,85 kb	3062 ms
Add_record2561.bmp	6, 16 kb	3884 ms
Add_record256.bmp	7,62 kb	5021 ms
airplane1.bmp	8,12 kb	5247 ms
airplane.bmp	9,24 kb	5877 ms
happy.bmp	10,7 kb	6886 ms
pepper.bmp	11,6 kb	7306 ms
happy1.bmp	12,2 kb	7768 ms
pepper1.bmp	14,7 kb	9286 ms



Gambar 15 Gambar Penampung Setelah Penyisipan

Setelah dilakukan penyisipan gambar *chiper* terhadap gambar penampung maka tampak gambar penampung tidak mengalami perubahan jika dilihat dengan mata. Begitu pula dengan resolusi dan ukuran file gambar penampung yang tetap sama seperti sebelum dilakukan penyisipan.

4.2.3 Hasil Ekstraksi

Selain percobaan proses enkripsi dan penyisipan. selanjutnya melakukan percobaan untuk mengekstrak file yang telah dilakukan proses enkripsi dan *embedding*. Proses ekstraksi dilakukan untuk mendapatkan file yang tersembunyi dalam penampung. Setelah dilakukan percobaan maka berikut hasil dari proses ekstraksi.

Gambar 16 Gambar Hasil Proses Ekstraksi

Setelah dilakukan proses ekstraksi maka didapatkan hasil berupa gambar *chiper* yang tersembunyi pada gambar penampung. Gambar di atas merupakan gambar *chiper* dari file airplane.bmp.

4.2.4 Hasil Dekripsi

Untuk mendapatkan gambar asli yang telah dilakukan proses ekstraksi. Maka gambar *chiper* yang didapat dari proses ekstraksi perlu didekripsi. Proses dekripsi merupakan proses akhir untuk mendapatkan kembali informasi yang telah disembunyikan dan disamarkan. Berikut gambar setelah didekripsi.

Gambar 17 Gambar Setelah Didekripsi

Gambar di atas merupakan gambar chiper yang telah didekripsi. Terlihat gambar tidak ada perbedaan jika dibandingkan dengan gambar asli. Begitupun dari segi pixel dan ukuran file.

Sama seperti percobaan pada proses enkripsi. Pada proses dekripsi terhadap beberapa gambar didapatkan hasil seperti pada tabel di atas. Begitu juga dilihat dari waktu eksekusi, semakin besar ukuran file gambar maka semakin besar pula waktu yang dibutuhkan selama eksekusi.

4.2.5 Hasil Pengujian SHA-1

Untuk mengetahui integritas data pada hasil dekripsi terhadap file asli maka perlu dilakukan pengujian nilai hash menggunakan metode hashing SHA-1. Metode ini bertujuan untuk mengetahui apakah gambar hasil dekripsi memiliki nilai hash yang sama dengan gambar asli. Apabila terdapat kesamaan bilai hash antara gambar dekripsi dengan gambar asli maka dapat dipastikan bahwa proses enkripsi dan dekripsi telah berhasil. Berikut tabel nilai hasil pengujian menggunakan metode SHA-1.

Tabel 3 Tabel Hasil Pengujian SHA-1

File Asli/ File Dekripsi	SHA
Add_record256.bmp	73D2C00AD85A6CC33C19EB8A30B8FBF4A0ADCB41
Add_record256.bmp	73D2C00AD85A6CC33C19EB8A30B8FBF4A0ADCB41
Add_record2561.bmp	6D2EF4FDBF42844026832156C2782FE304F8938A
Add_record2561.bmp	6D2EF4FDBF42844026832156C2782FE304F8938A
airplane.bmp	81F9C2F4A795A2AD470B5D6F88294DA0EEC9AB97
airplane.bmp	81F9C2F4A795A2AD470B5D6F88294DA0EEC9AB97
airplane1.bmp	1B516B69DE67CD413ACAD6A8376CED78774E947F
airplane1.bmp	1B516B69DE67CD413ACAD6A8376CED78774E947F
happy.bmp	B24A2BCF6E67006E769874CB8C85E1E33C3D0D54
happy.bmp	B24A2BCF6E67006E769874CB8C85E1E33C3D0D54
happy1.bmp	31519A7351B7A6AEA9735E59DA8379F1BC2C61F9
happy1.bmp	31519A7351B7A6AEA9735E59DA8379F1BC2C61F9
pepper.bmp	6A2FEE1FDD9C86C46C8A414453BF5D57BAB1C952
pepper.bmp	6A2FEE1FDD9C86C46C8A414453BF5D57BAB1C952
pepper1.bmp	17867B551C5E26D50AF3AE967CECC4EFCDB694DE
pepper1.bmp	17867B551C5E26D50AF3AE967CECC4EFCDB694DE
UNIX-BMP.bmp	38DE09763D9A15BB7F9C244D877F1FF7C0C154B4
UNIX-BMP.bmp	38DE09763D9A15BB7F9C244D877F1FF7C0C154B4
UNIX-BMP1.bmp	0D9F8BB794A50852CA41B6107183F8C4BD2181D8
UNIX-BMP1.bmp	0D9F8BB794A50852CA41B6107183F8C4BD2181D8

Tabel di atas merupakan tabel nilai hash dari gambar dekripsi dan gambar asli. Pada tabel di atas menjelaskan nilai hash dari masing-masing file. Sebagai contoh terdapat airplane.bmp dengan warna kolom abu-abu yang merupakan file asli dan file airplane.bmp dengan kolom berwarna putih yang merupakan gambar dekripsi. Nilai hash kedua file menunjukkan nilai hash yang sama. Itu berarti bahwa setiap proses enkripsi dan dekripsi file gambar dengan algoritma kriptografi AES berhasil.

4.2.6 Hasil Pengujian PSNR

Selain pengujian untuk mengetahui kesamaan gambar asli dan gambar dekripsi. Penting adanya untuk melakukan pengujian terhadap kualitas gambar stego. Metode yang digunakan untuk mengetahui kualitas suatu gambar stego adalah metode PSNR. Metode PSNR merupakan sebuah metode pengujian secara objektif. Berikut merupakan tabel pengukuran kualitas gambar stego dengan metode PSNR.

Tabel 4Tabel Uji PSNR

Penampung	File	MSE	PSNR
Babbon.bmp	Add_record256.bmp	2,78595	44,228764
	Add_record2561.bmp	2,741475	44,333744
	airplane.bmp	2,8389416	44,108204
	airplane1.bmp	2,809658	44,169483
	happy.bmp	2,8986585	43,97552
	happy1.bmp	2,95395	43,8743
	pepper.bmp	2,9187918	43,935772
	pepper1.bmp	3,0116584	43,752335
	UNIX-BMP.bmp	2,6432	44,592083
	UNIX-BMP1.bmp	2,70175	44,438427

Tabel di atas merupakan tabel pengujian kualitas salah satu gambar stego dengan nama file Babbon.bmp menggunakan metode PSNR. Pengujian dilakukan dengan menggunakan gambar penampung sebelum disisipkan pesan dan gambar penamping setelah disisipkan pesan (gambar stego). Sebelum mendapatkan nilai PSNR terlebih dahulu harus dicari nilai MSE (Mean Square Error) seperti yang telah dijelaskan sebelumnya. Setelah mendapatkan nilai MSE selanjutnya mencari nilai PSNR. Pada tabel di atas sebagai contoh file dengan nama airplane.bmp memiliki nilai MSE sebesar 2,8389416dB dan nilai PSNR sebesar 44,108204 dB. Itu menjelaskan berarti kualitas hasil penyisipan sudah baik dilihat dari nilai MSE dan nilai PSNR. Semakin kecil nilai MSE berarti semakin kecil nilai error dan semakin besar nilai PSNR berarti hasil penyisipan semakin baik.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan dari hasil dan pembahasan yang telah didapatkan maka dapat ditarik kesimpulan bahwa algoritma kriptografi AES 256 dan metode steganografi LSB baik untuk diimplementasikan dalam mengamankan file gambar yang kerahasiaannya sangat dijaga. Dari hasil proses enkripsi didapatkan gambar *chiper* yang tidak dapat dimengerti. Dengan semakin bertambahnya ukuran file semakin besar waktu eksekusi dari proses enkripsi. Seperti yang terlihat pada tabel percobaan yang mendapatkan hasil dari waktu eksekusi yang bertambah seiring dengan bertambahnya ukuran file gambar. Dari hasil proses dekripsi didapatkan file

gambar pesan asli. Begitu pula seperti proses enkripsi, pada proses dekripsi semakin besar ukuran file maka semakin besar pula waktu eksekusi proses dekripsi. Ini diperkuat dengan hasil uji SHA-1 yang menyatakan nilai hash file gambar hasil dekripsi sama dengan nilai hash file gambar asli. Ini berarti proses dari enkripsi maupun dekripsi telah berhasil. Selanjutnya dilihat dari hasil proses penyisipan dan ekstraksi. Gambar penampung sebelum disisipkan pesan (gambar cover) dan gambar penampung setelah disisipkan pesan (gambar stego) tidak ada perbedaan jika dilihat secara kasat mata. Pada saat dilakukan pengujian terhadap kualitas gambar stego, hasil nilai PSNR rata-rata memiliki nilai 44,14086 dB dengan nilai error rata-rata 2,830403 dB dan itu menyatakan kualitas hasil penyisipan yang baik. Jadi, aplikasi ini dapat diimplementasikan untuk mengamankan dan menjaga kerahasiaan file gambar.

5.2 Saran

Adapun saran yang dapat disampaikan untuk penelitian yang akan datang, yaitu:

1. Algoritma kriptografi AES 256 dan metode steganografi LSB baik untuk diimplementasikan pada file gambar, namun penelitian selanjutnya diharapkan dapat mengimplementasikan pada file-file multimedia seperti file audio, file vide, file berformat .gif dan lain sebagainya.
2. Metode steganografi LSB dapat digunakan untuk penyisipan file gambar. Namun perlu pula dipertimbangkan aspek-aspek seperti perbandingan resolusi antara gambar cover dan gambar stego.

DAFTAR PUSTAKA

[1] Huda W, M. (2003). Perkembangan Fungsi Hash pada SHA (Secure Hash Algorithm). *Jurnal Ilmu Komputer Dan Teknologi Informasi, Vol III N0.2, Oktober 2003* .

[2] Moenandar, G., Wirawan, & Setijadi, E. (2012). Analisa Kualitas Citra Pada Steganografi Untuk Aplikasi E-Goverment. *Seminar Nasional Manajemen Teknologi XV* .

[3] Putra, H., Santoso, E., & Muflikhah, L. (2013). Implementasi Algoritma Kriptografi Advance Encryption Standard (AES) Pada Kompresi Data Teks. *Jurnal Ilmu Komputer Universitas Brawijaya* .

[4] Rakhmat, B., & Fairuzabadi, M. (2010). Steganografi Menggunakan Metode Least Significant Bit Dengan Kombinasi Algoritma Kriptografi Vigenere Dan RC4. *Jurnal Dinamika Informatika* , 1-17.

[5] Wahyudi, K., & DP, S. (2008). Aplikasi Kriptografi Untuk Pertukaran Pesan Menggunakan Teknik Steganografi Dan Algoritma AES. *Prosiding Seminar Nasional Teknoin 2008* .

[6] Yuniat, V. i., Indriyanta, G., & Rachmat, A. (2009). Enkripsi Dan Dekripsi Dengan Algoritma AES 256 Untuk Semua Jenis File. *Jurnal Informatika Volume 5* .