Perbandingan Steganografi pada Citra Gambar *Graphics Interchange Format* dengan Algoritma *Gifshuffle* dan Metode *Least Significant Bit*

Septu Jamasoka (13509080)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509080@std.stei.itb.ac.id

Abstract*—Graphics Interchange Format (GIF) merupakan format grafis yang paling sering digunakan untuk keperluan desain *website*. GIF memiliki kombinasi warna lebih sedikit dibanding JPEG, namun mampu menyimpan grafis dengan latar belakang (*background*) transparan ataupun dalam bentuk animasi sederhana. Karena kemampuan yang dimiliki oleh GIF yaitu sanggup membuat seolah animasi dari sejumlah gambar, GIF terkadang masih dianggap bagus untuk digunakan ketika ingin men-*load* sumber dalam ukuran besar, maka gambar GIF bisa ditampilkan terlebih dahulu agar pengguna tidak merasa aneh ketika load terjadi. Gambar GIF memiliki format khusus yaitu memiliki *colormap* yang mengandung entri sebanyak 255 entri dan terdiri dari sejumlah *image* apabila GIF tersebut berupa animasi atau satu *image* jika bukan merupakan animasi. Algoritma *gifshuffle* secara umum menggunakan *colormap* tersebut yang diacak sesuai dengan source teks yang ingin di-embed pengguna. Pengubahan susunan pada *colormap* tersebut tentunya tidak akan mengubah tampilan citra secara umum. Pada steganografi dengan metode *Least Significant Bit* (LSB), untuk citra gambar GIF pada dasarnya terdiri dari bagian *image* sehingga metode LSB dapat diterapkan dengan menyisipkan pesan pada bagian LSB dari tiap *byte image* dari citra gambar tersebut. Kedua metode tersebut tentunya memiliki kelebihan dan kekurangan masing-masing. Salah satu kelebihan dari masing-masing adalah pada algoritma *gifshuffle* adalah perubahan pada citra gambar tidak dapat terlihat dengan jelas, sedangkan algoritma *Least Significant Bit* (LSB) adalah memungkinkan ukuran pesan yang di-*embed* dinamis tergantung pada ukuran dari citra GIF tersebut. Selain itu, dari segi perhitungan *Peak-to-Signal Noise Ratio* dan kecepatan *embed* tentunya juga akan berbeda pada penggunaan kedua metode tersebut.

Index Terms*— GIF, steganografi, *gifshuffle*, metode *least significant bit.

I. PENDAHULUAN

Kriptografi, secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita[3]. Kriptografi merupakan salah satu cabang ilmu yang cukup penting terutama dalam keamanan terhadap data. Selain itu, kriptografi saat ini juga sudah banyak digunakan untuk keamanan penyaluran informasi selama proses pengiriman data dalam jaringan seperti jaringan komputer, jaringan komunikasi, dan lainnya.

Selain kriptografi, terdapat juga sebuah cabang ilmu yang cukup berkaitan dengan keamanan data, yaitu steganografi. Dengan steganografi, penyembunyian pesan lebih baik karena disimpan dalam media lain sehingga orang lain tidak merasa curiga daripada kriptografi dengan hasil penyembunyiannya berupa teks yang teracak dan memungkinkan orang lain mencoba untuk melakukan serangan kepada cipherteks tersebut.

Banyak media dapat digunakan untuk steganografi seperti teks atau artikel, gambar, audio, dan juga video. Salah satu yang cukup menarik adalah penyembunyian pada media citra *Graphics Interchange Format* (GIF). Media citra GIF banyak digunakan terutama pada halaman suatu web untuk memperingan pengaksesan video karena video memiliki ukuran yang cukup besar dan media GIF sanggup menciptakan animasi dengan ukuran yang cukup ringan. Berbagai alternatif dapat digunakan untuk melakukan steganografi pada media GIF yaitu dengan menggunakan metode *gifshuffle* dan metode *least significant bit*.

II. DASAR TEORI

A. Steganografi

Steganografi adalah seni dan ilmu menulis pesan tersembunyi atau menyembunyikan pesan dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Sebaliknya, kriptografi menyamarkan arti dari suatu pesan, tapi tidak menyembunyikan bahwa ada suatu pesan. Kata "steganografi" berasal dari bahasa Yunani *steganos*, yang artinya “tersembunyi atau terselubung”, dan *graphein*, “menulis”[4].

Tujuan sebenarnya dari steganografi adalah untuk menyembunyikan pesan atau informasi pada berbagai media. Perubahan yang terjadi pada media tempat penyimpanan biasanya bersifat sedikit sehingga tidak dapat terlihat secara kasat mata oleh orang lain dan hanya orang yang diberitahu dapat mengekstrak pesan dari media tersebut.

Pada metode steganografi cara ini sangat berguna jika

digunakan pada cara steganografi komputer karena banyak format berkas digital yang dapat dijadikan media untuk menyembunyikan pesan. Format yang biasa digunakan di antaranya:

- Media citra: bitmap (bmp), gif, pcx, jpeg, dll.
- Media suara: wav, voc, mp3, dll.
- Media video: avi, wmv, dll.
- Media lainnya: teks file, html, pdf, dll.

Kelebihan steganografi jika dibandingkan dengan kriptografi adalah pesan-pesannya tidak menarik perhatian orang lain. Pesan-pesan berkode dalam kriptografi yang tidak disembunyikan, walaupun tidak dapat dipecahkan, akan menimbulkan kecurigaan. Seringkali, steganografi dan kriptografi digunakan secara bersamaan untuk menjamin keamanan pesan rahasianya.

Beberapa metode yang dapat digunakan untuk melakukan steganografi, yaitu

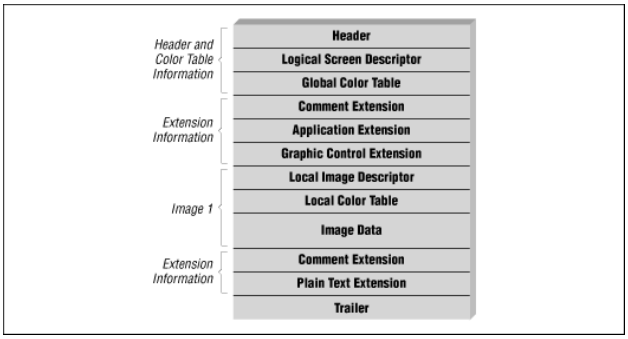
- Metode *Least Significant Bit*
- *Algorithm and Transformation*
- *Redundant Pattern Method*
- *Spread Spectrum Method*

B. Format Citra Graphics Interchange Format *(GIF)*

GIF pertama kali dikembangkan oleh CompuServe sebagai media citra online pertama karena pada masa itu hanya ada video 8 bit, sebelum JPG dan warna 24 bit digunakan. GIF menggunakan warna yang terindeks yang terbatas pada 256 warna yang boleh dimiliki oleh sebuah citra digital. GIF merupakan format grafik yang tergolong cukup bagus dan kegunaan untuk sekarang ini terletak pada web. Hal ini karena banyak gambar berukuran kecil seperti logo hanya memerlukan jumlah warna yang sedikit sehingga dengan penggunaan GIF, ukuran citra digital akan lebih kecil dibandingkan format JPG sehingga cocok untuk web yang memungkinkan penampilan citra lebih cepat. Kegunaan lain dari citra GIF adalah dapat digunakan untuk menyimpan sprite dari games yang berwarna sedikit dan menyimpan animasi ringan atau clip video dengan resolusi kecil.

Secara umum, format utama GIF terdiri dari GIF *header*, *global color table*, *image block*, dan diakhiri dengan trailer. Akan tetapi, dapat juga ditambahkan data-data tertentu berupa *comment extension* dan *application extension* sesudah *global color table*, dan *comment extension* dan *plain text extension* sesudah *image block*. Format citra GIF dapat dilihat pada gambar 1 di bawah ini.

Bagian GIF header berisi tandatangan berukuran 6 byte yang biasa berisi “GIF89a” atau “GIF87a”, ukuran layar (lebar dan panjang yang masing-masing berukuran 2 byte, *background* berupa indeks warna pada bagian tabel warna, aspek rasio, dan flags. Bagian global color table berisi 256 pilihan warna pada citra GIF yang masing-masing warna dilambangkan dengan 3 byte yang terdiri dari byte pertama mewakili komponen warna merah, byte kedua mewakili komponen warna hijau, dan byte ketiga mewakili komponen warna biru.



Gambar 1. *Layout* Format Citra GIF

Gambar 2. Contoh citra GIF[11][12][13]

Bagian image block secara umum berupa sekumpulan dari *image descriptor*, *local color table*, dan data citra

yang sudah terkompresi. Sedangkan bagian trailer hanya berisi satu byte, yaitu ';' sebagai penanda akhir dari sebuah citra GIF. Untuk lebih jelasnya, dapat dilihat pada gambar 2 dan 3 berikut.

```
1)  01000111 -> 'G'
2)  01001001 -> 'I'
3)  01000110 -> 'F'
4)  00111001 -> '8'
5)  00111010 -> '9'
6)  01100001 -> 'a'
7)  01110100
8)  00000011 -> 884 pixels wide
9)  00101011
10) 00000100 -> 1067 pixels high
11) 11110111 -> Global colormap present, 256 colors, 24 bits per color.
12) 11111111 -> background is color index 255.
13) 00000000 -> zero
14) 00000000
15) 00000000
16) 00000000 -> color 0 in the colormap = (0, 0, 0) = black
17) 10000000
18) 00000000
19) 00000000 -> color 1 in the colormap = (128, 0, 0) = red
20) 00000000
21) 10000000
22) 00000000 -> color 2 in the colormap = (0, 128, 0) = green
```

Gambar 3. Contoh GIF *Header* dan *Global Color Table*[5]

```
1)  00101100 -> ","
2)  00000000
3)  00000000 -> 0 pixels offset to the right.
4)  00000000
5)  00000000 -> 0 pixels offset to the bottom.
6)  01110100
7)  00000011 -> 884 pixels wide
8)  00101011
9)  00000100 -> 1067 pixels high
10) 00000000 -> No local colormap, non-interlaced.
```

Gambar 4. Contoh *Local Image Descriptor*[5]

C. Metode Least Significant Bit

Metode *least significant bit* merupakan metode standar yang banyak digunakan untuk melakukan steganografi terutama pada media citra digital perubahan satu bit pada *least significant bit* tidak akan mengakibatkan perubahan warna yang cukup besar sehingga tidak dapat terlihat secara kasat mata oleh orang lain.

Metode ini menggunakan *least significant bit* dari byte-byte pada media digital yang disubstitusi dengan satu bit –satu bit pesan yang ingin disisipkan sehingga ukuran file yang bisa disimpan cenderung jauh lebih kecil dari ukuran asli dari citra digital tersebut. Secara umum algoritma untuk metode ini dapat dilihat pada gambar 5. Proses ektraksi dari metode ini dapat dilakukan dengan menyusun bit-bit yang disisipkan pada citra tersebut sehingga membentuk pesan yang telah disisipkan tersebut.

for i = 1 to length(m) **do**
    Compute index j_i where to store the ith message bit of *m*
    $Sj_i \leftarrow \text{LSB}(Cj_i) = m_i$
endfor

Gambar 5. Algoritma Metode *Least Significant Bit*

Akan tetapi, apabila penyisipan dilakukan pada byte-byte data secara berurutan, maka akan mudah bagi orang lain untuk mengekstraksi pesan yang disisipkan karena orang tersebut cukup menyusun bit-bit akhir dari semua byte dan kemungkinan diperolehnya teks semakin mudah. Oleh karena itu, terdapat metode baru yaitu penggunaan *pseudorandom* dalam penyisipan bit sehingga bit-bit akan terkesan disimpan secara acak dan orang lain yang mencoba tidak mungkin dapat menyusun pesan yang tersisip.

Saat ini, sudah banyak steganalisis terhadap metode ini. Salah satu steganalisis adalah dengan mengubah semua bit dari byte sesuai dengan mengikuti least significant bit. Hal ini dapat mengakibatkan apabila terdapat pesan yang tersimpan, akan muncul ketidakteraturan pada gambar yang telah diubah tersebut dan orang dapat mengetahui bahwa pada byte-byte tersebut tersisipkan pesan-pesan sehingga mudah bagi orang tersebut untuk membongkar pesan yang ada. Kekurangan dari penggunaan pseudorandom ini adalah penentuan tempat penyisipan data tergantung hasil bilangan acak sehingga terdapat kemungkinan terjadi tabrakan data dan memerlukan mekanisme khusus dalam menangani hal ini karena kesalahan satu bit dapat mengakibatkan pesan sulit untuk dibaca oleh penerima pesan.

Dari hal ini muncul ide untuk tidak menyisipkan pada *least significant bit*, tetapi pada bit kedua atau bit ketiga dari byte tersebut karena dengan begitu akan sulit untuk dilacak oleh orang lain dan perubahan dua atau empat ukuran warna tidak akan cukup signifikan dan cukup aman agar tidak bisa diserang dengan mengubah semua bit dari byte mengikuti *least significant bit* dari citra digital.

D. Algoritma Gifshuffle

Gifshuffle adalah algoritma yang digunakan untuk menyusun ulang *palette* warna yang bertujuan untuk menyembunyikan informasi. Algoritma *gifshuffle* ini memanfaatkan *colormap* pada citra GIF yang disusun ulang sehingga tidak dapat diketahui apabila terdapat pesan yang disembunyikan pada citra tersebut. Citra GIF terdiri dari sebuah *colormap* yang dapat menampung warna sebanyak 256 entri, hal ini memungkinkan untuk menyimpan pesan berukuran 1683 bit. Adapun langkah-langkah untuk menyisipkan pesan dengan metode ini adalah sebagai berikut.

1. Mulai dengan pesan yang ingin disembunyikan, pesan tersebut susun secara berurut sehingga terbentuk susunan 1's dan 0's.
2. Tambahkan '1' pada awal dari urutan tersebut, sehingga diperoleh bilangan biner *m* yang cukup besar.
3. Hitung jumlah warna unik yang tersimpan pada *colormap* citra GIF dan disebut dengan *n*. Jika *m* > *n!-1* maka pesan terlalu besar dan tidak dapat disisipkan.
4. Setiap warna pada *colormap* diubah dalam representasi integer dengan melakukan perhitungan:
   C = byte_red x 65536 + byte_green x 256 +

byte_blue

Kemudian disusun berdasarkan urutan.

5. Iterasi *i* dari nilai *1..n*. Setiap warna *n-i* dialokasikan pada posisi baru (*m mod i*), kemudian *m* dibagi dengan *i*.
6. Setiap warna *(n-1)..0* kemudian ditambahkan pada *colormap* baru sesuai posisi baru masing-masing, apabila tidak memiliki cukup warna untuk memenuhi *colormap*, maka diisi dengan warna terakhir dari *colormap* asli.
7. *Image block* kemudian didekompres untuk disusun urutan warna mengikuti urutan baru, kemudian dikompres kembali. Lakukan untuk setiap *image block* yang ada.

Algoritma *gifshuffle* memungkinkan penyembunyian pesan tanpa mengubah susunan bit-bit secara total dan hanya mengubah susunan pada *colormap* tanpa mengubah warna pada gambar sehingga tidak terdapat perubahan yang cukup berarti pada citra GIF. Oleh karena itu, algoritma *gifshuffle* cukup aman dalam penyembunyian data, tetapi pesan yang dapat disimpan terbatas dan cukup kecil sehingga terkadang jarang ada orang yang menggunakannya.

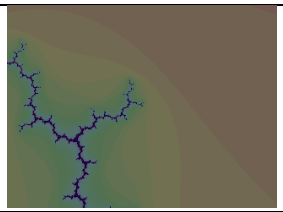
III. IMPLEMENTASI

A. Kakas dan File Uji

Implementasi yang dilakukan untuk melakukan pengujian terhadap citra GIF dilakukan dengan menggunakan bahasa Java dan kakas yang digunakan berupa NetBeans 7.0. Program yang dibuat hanya berupa program berbasis command prompt tanpa ada *graphical user interface* sehingga hasil waktu yang diperoleh lebih tepat dibandingkan dengan menggunakan *graphical user interface*. Selain itu, contoh citra GIF yang digunakan hanya berupa citra diam yang tidak memiliki animasi pada citra tersebut sehingga total *image block* yang dimiliki adalah satu buah.

Citra GIF yang digunakan terdiri atas 3 citra dengan ukuran yang berbeda-beda. Adapun file-file uji dapat dilihat pada tabel 1 berikut.

Tabel 1. Citra Uji

Nama File	Citra	Ukuran (kiloByte)
image1.gif		23
image2.gif		29
image3.gif		114

Adapun pesan-pesan yang ingin disisipkan berjumlah tiga buah dengan ukuran yang berbeda yang dapat digunakan untuk menguji ukuran penampung dari citra tersebut.

Tabel 2. File Pesan

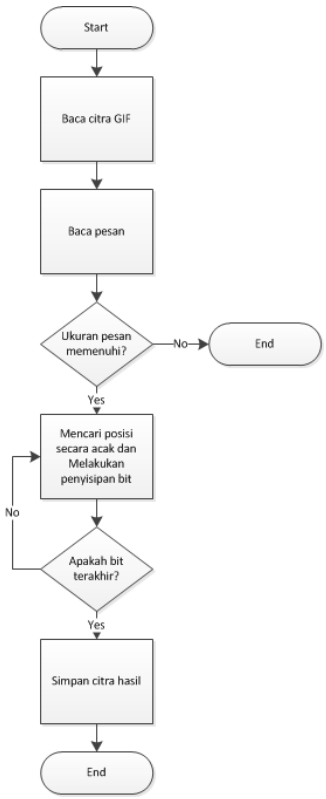
Nama File	Ukuran (byte)
tes1.txt	32
tes2.txt	119
tes3.txt	2334

B. *Metode* Least Significant Bit

Metode *least significant bit* yang digunakan adalah penyisipan bit-bit dari pesan yang ingin disimpan secara pseudorandom sehingga tempat penyisipan terletak pada posisi acak dan menyulitkan orang lain untuk mencoba melakukan serangan. Langkah-langkah dari metode *least significant bit* adalah:

1. Membaca file GIF.
2. Melakukan pengecekan untuk mencari posisi *image block.*
3. Menghasilkan nilai acak antara {1,5} sehingga posisi berikutnya berada antara 1 hingga 5 byte dari posisi byte sekarang.
4. Menyisipkan bit pada least significant bit pada byte tersebut.
5. Mengulangi langkah 3 dan 4 sehingga semua bit dari pesan yang ingin disisipkan sudah tersisip secara acak.
6. Menyimpan kembali ke dalam file baru.

Diagram alir untuk langkah-langkah tersebut dapat dilihat pada gambar 6 berikut.



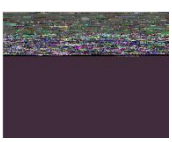
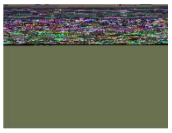
Gambar 6. Diagram Alir Metode LSB

Hasil uji yang dilakukan terhadap file uji dapat dilihat pada tabel 3 dan 4 di bawah ini.

Tabel 3. Hasil Uji dengan Metode *Least Significant Bit*

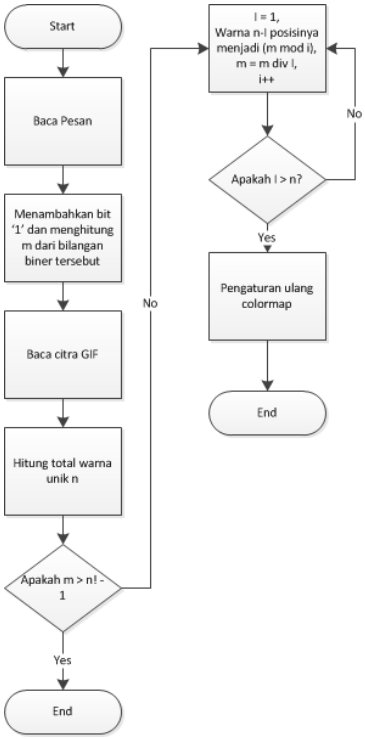
	image1	image2	image3
tes1	Sukses: sukses PSNR: 68.45 dB Waktu: 2.01 ms	Sukses: sukses PSNR: 69.18 dB Waktu: 2.04 ms	Sukses: sukses PSNR: 75.36 dB Waktu: 7.57 ms
tes2	Sukses: sukses PSNR: 62.69 dB Waktu: 2.72 ms	Sukses: sukses PSNR: 64.05 dB Waktu: 2.76 ms	Sukses: sukses PSNR: 70.06 dB Waktu: 7.84 ms
tes3	Sukses: gagal PSNR: - Waktu: -	Sukses: gagal PSNR: - Waktu: -	Sukses: sukses PSNR: 57.26 dB Waktu: 9.67 ms

Tabel 4. Citra Hasil Uji dengan Metode *Least Significant Bit*

	image1	image2	image3
tes1			
tes2			
tes3	---	---	

C. Algoritma Gifshuffle

Algoritma *Gifshuffle* yang diimplementasikan mengikuti langkah-langkah yang disebutkan dalam bagian Dasar Teori terkait algoritma *gifshuffle* ini. Diagram alir untuk langkah-langkah tersebut dapat dilihat pada gambar 7 berikut.



Gambar 7. Diagram Alir Algoritma *Gifshuffle*

Hasil uji yang dilakukan terhadap file uji dapat dilihat pada tabel 5 dan 6 di bawah ini.

Tabel 5. Hasil Uji dengan Algoritma *Gifshuffle*

	image1	image2	image3
tes1	Sukses: sukses PSNR: Infinity Waktu: 185.48 ms	Sukses: sukses PSNR: Infinity Waktu: 226.49 ms	Sukses: sukses PSNR: Infinity Waktu: 645.01 ms
tes2	Sukses:	Sukses:	Sukses:

	sukses PSNR: Infinity Waktu: 1320.19 ms	sukses PSNR: Infinity Waktu: 1324.99 ms	sukses PSNR: Infinity Waktu: 1373.11 ms
tes3	Sukses: gagal PSNR: - Waktu: -	Sukses: gagal PSNR: - Waktu: -	Sukses: gagal PSNR: - Waktu: -

Tabel 6. Citra Hasil Uji dengan Algoritma *Gifshuffle*

	image1	**image2**	**image3**
tes1			
tes2			
tes3	-	-	-

IV. ANALISIS

A. Analisis terhadap Metode Least Significant Bit

Dari hasil pengujian diperoleh bahwa dengan metode Least Significant Bit menyebabkan citra GIF yang dihasilkan cukup rusak karena parubahan satu nilai naik atau turun dapat mengakibatkan perubahan warna yang cukup signifikan karena karena image data pada GIF hanya menyimpan reference ke *colormap* yang ada sehingga mengakibatkan keseluruhan file citra GIF menjadi rusak. PSNR yang dihasilkan lumayan bagus meskipun pada kenyataannya file yang dihasilkan cukup rusak karena perhitungan PSNR yang digunakan hanya menghitung perubahan byte bukan perubahan warna secara mencolok.

Waktu yang dibutuhkan untuk menyisipkan pesan tergolong cepat karena hanya membutuhkan kurang dari 10 *milisecond* untuk melakukan penyisipan pesan karena cukup mencari posisi secara acak dan menyisipkan bit pesan pada *least significant bit* dari data gambar.

B. Analisis terhadap Algoritma Gifshuffle

Dari hasil pengujian diperoleh bahwa algoritma *gifshuffle* menghasilkan citra yang sama persis dengan citra sebelum disisipkan pesan karena citra. Hal ini karena *gifShuffle* tidak mengubah bit-bit warna dari sebuah *image*, tetapi *gifshuffle* hanya mengubah susunan warna dari *colormap* pada citra tersebut. Oleh karena itu hampir semua PSNR bernilai *Infinity* yang artinya tidak terdapat perubahan pada citra.

Waktu yang dibutuhkan untuk menyisipkan pesan tergolong cukup lama yaitu hampir membutuhkan waktu hingga 0,5 sampai 1 detik dalam penyisipan. Hal ini karena terdapat bilangan BigInteger yang mengakibatkan dalam pengoperasiannya membutuhkan waktu lebih banyak dibandingkan integer biasa. Selain itu, batas penyimpanan terhadap pesan maksimal sebesar 1683 bit sehingga ukuran maksimal pesan hanya sebesar 210 byte.

C. Perbandingan Metode Least Significant Bit *dan Algoritma* Gifshuffle

Perbandingan steganografi pada citra GIF dengan menggunakan metode least significant bit dan algoritma *gifshuffle* terhadap pengujian yang dilakukan dapat dilihat pada tabel 7.

Tabel 7. Perbandingan Steganografi dengan Metode *Least Significant Bit* dengan Algoritma *Gifshuffle*

Pembanding	**Metode LSB**	**Algoritma *Gifshuffle***
PSNR	Berkisar antara 50 hingga 80 dB	Tidak terbatas (*Infinity*)
Waktu penyisipan	Kurang dari 10 ms	Mencapai 0,5 hingga 1 s
kualitas hasil citra GIF	Sangat rusak	Tidak mengalami perubahan dibandingkan dengan *file* asli
ukuran pesan	Bergantung dengan ukuran citra GIF	Terbatas pada 210 byte dan tidak mungkin dapat bertambah.

Dari tabel 7 diperoleh bahwa dengan metode LSB, hasil citra GIF akan rusak dan dapat mengakibatkan orang lain merasa file citra tersebut sudah dilakukan steganografi sehingga akan mencoba untuk melakukan steganalisis, sedangkan dengan menggunakan algoritma *gifshuffle*, citra yang dihasilkan sama persis dengan citra asli sebelum dilakukan penyisipan sehingga tidak mungkin menimbulkan kecurigaan bagi orang lain. Dari segi waktu untuk menyisipkan pesan, metode LSB lebih cepat dibandingkan dengan algoritma *gifshuffle* karena *gifshuffle* membutuhkan operasi dalam BigInteger yang memakan waktu yang cukup lama dibandingkan dengan metode LSB yang hanya menggunakan tipe-tipe dasar yang berukuran kecil.

V. KESIMPULAN

Algoritma *gifshuffle* menghasilkan kualitas citra yang sama persis dengan kualitas dari citra asli karena secara umum algoritma ini hanya melakukan penyusunan ulang warna-warna pada *colormap*. Sedangkan dengan menggunakan metode *least significant bit* dapat mengakibatkan kerusakan pada citra GIF karena perubahan satu digit dapat mengakibatkan perubahan warna yang cukup signifikan karena *image* data pada GIF hanya menyimpan reference ke *colormap* yang ada. Dari segi PSNR algoritma *gifshuffle* jauh lebih bagus

dibandingkan dengan metode LSB, tetapi dari segi waktu LSB lebih bagus dibandingkan dengan *gifshuffle*. Oleh karena itu, lebih baik digunakan algoritma *gifshuffle* untuk menyisipkan pesan pada citra GIF, akan tetapi ukuran pesan yang dapat disimpan sangat kecil sekali dan tidak bergantung pada ukuran citra GIF tersebut.

REFERENCES

[1] http://en.wikipedia.org/wiki/Graphics_Interchange_Format, diakses pada tanggal 18 Maret 2012, 20.14 WIB.
[2] http://ha.hn.web.id/2011/10/26/steganografi-menyembunyikan-pesan-rahasia/, diakses pada tanggal 18 Maret 2012, 20.12 WIB.
[3] http://id.wikipedia.org/wiki/Kriptografi, diakses pada tanggal 18 Maret 2012, 20.13 WIB.
[4] http://id.wikipedia.org/wiki/Steganografi, diakses pada tanggal 18 Maret 2012, 20.12 WIB.
[5] http://ptolemy.eecs.berkeley.edu/eecs20/sidebars/images/gif.html, diakses pada tanggal 18 Maret 2012, 20.13 WIB.
[6] http://www.darkside.com.au/gifshuffle/, diakses pada tanggal 18 Maret 2012, 21.20 WIB.
[7] http://www.jsums.edu/cms/tues/docs/Steganography/LSB-Steganography.pdf, diakses pada tanggal 18 Maret 2012, 21.01 WIB.
[8] http://www.nikis.de/181/gif89a.htm#establish_code_size, diakses pada tanggal 18 Maret 2012, 20.20 WIB.
[9] http://www.onicos.com/staff/iz/formats/gif.html#ib, diakses pada tanggal 18 Maret 2012, 20.14 WIB.
[10] http://www.scantips.com/basics9g.html, diakses pada tanggal 18 Maret 2012, 21.20 WIB.
[11] https://ccse.lbl.gov/Research/Combustion/CombustPics/HflameVort_still.gif, diakses pada tanggal 18 Maret 2012, 22.10 WIB.
[12] http://www.nahee.com/spanky/www/fractint/dz/top1500.gif, diakses pada tanggal 18 Maret 2012, 22.10 WIB.
[13] http://burritojustice.files.wordpress.com/2012/03/wall-of-salsa.gif, diakses pada tanggal 18 Maret 2012, 22.10 WIB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Maret 2012

Septu Jamasoka
13509080