IMPLEMENTASI ALGORITMA KOMPRESI PADA STEGANOGRAFI CITRA DIGITAL DENGAN METODE MODIFIKASI LSB

Andi Setiadi Manalu
Teknik Komputer, Politeknik Bisnis Indonesia
email: andi.manaloe@gmail.com

Abstract

Steganography technique using the LSB (Least Significant Bit) modification method is the simplest technique. Therefore steganography with this method is very suitable for beginners in the field of steganography. The LSB modification method stores data by replacing insignificant bits in the file cover with the bits of the file to be stored. One of the weaknesses of modified LSB method is its inability to store large size data. On average, the steganography technique with the modified LSB method is only able to store data in the size of one eighth of the size of the cover (for the cover in the form of 24-bit image), of course this is not efficient. To overcome this problem, in this research several methods were proposed to enlarge the capabilities of the modified LSB steganography technique in storing data. The first method is preprocessing the data files to be stored, namely by compression the data, the second is to prepare the file cover (stretch) by enlarging the file cover (stretching image), and the third is combining the first method and second at the same time. In this research, an analysis of the processes and results of each method will also be carried out. To increase the security of the data to be stored, the data stored is also encrypted first.

Keywords: *Digital image, deflate algorithm, encode and decode, lsb modification, steganography*

1. PENDAHULUAN

1.1. Latar Belakang

Kriptografi adalah cara yang paling umum untuk penyampaian pesan rahasia dari pihak pertama ke pihak ke dua, meskipun kekuatan keamanannya tergantung algoritma kriptografi apa yang digunakan. Meskipun kriptografi memiliki kekuatan keamanan tertentu dalam menyampaikan pesan rahasia namun hal tersebut mampu menimbulkan kecurigaan dari pihak yang tidak lain juga ingin mengetahui isi pesan tersebut.

Hal ini disebabkan karena secara umum prinsip kriptografi adalah mengubah suatu pesan yang bisa dipahami secara langsung menjadi kode-kode yang tidak bisa dipahami secara langsung, sehingga pihak lain yang melihat adanya pengiriman kode-kode tersebut akan mencurigai bahwa kode-kode tersebut adalah suatu pesan rahasia. Mereka akan berusaha dengan segala cara untuk mengetahui isi dari pesan tersebut.

Steganografi adalah jawaban terhadap tantangan di atas. Steganografi tidak mengubah pesan menjadi kode-kode tertentu melainkan menyisipkan pesan pada suatu objek, dimana objek yang telah disisipi pesan tidak akan mengalami perubahan fisik secara kasat mata. Hal ini menguntungkan karena tidak akan menimbulkan kecurigaan pada pihak lain.

1.2. Rumusan Masalah

Rumusan masalah pada penelitian ini adalah bagaimana cara menyembunyikan informasi rahasia ke dalam media digital sehingga informasi rahasia tersebut benar–benar terjaga kerahasiannya.

1.3. Tujuan Penelitian

Tujuan yang akan dicapai dari penelitian ini adalah membuat aplikasi penyembunyian informasi ke dalam citra digital dengan memberikan satu alternatif metode steganografi untuk menyembunyikan pesan ke dalam berkas gambar dan memberi pesan pada gambar tentang keaslian dari gambar tersebut.

2. DASAR TEORI

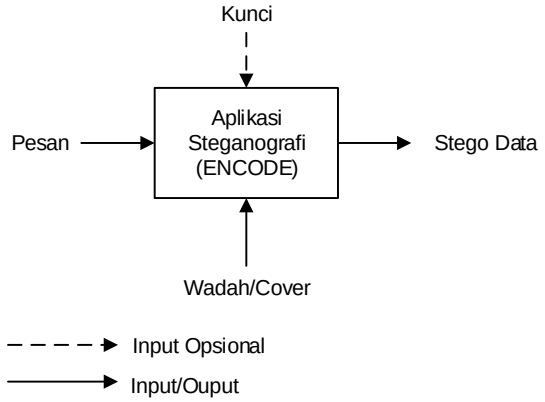
2.1. Steganografi

E-ISSN:2621: 3079
ISSN: 2621:1556

Steganografi merupakan seni penyembunyian pesan ke dalam pesan lainnya sedemikian rupa sehingga orang lain tidak menyadari ada sesuatu di dalam pesan tersebut. Kata steganografi (*steganography*) berasal dari bahasa Yunani yaitu *steganos* yang artinya tersembunyi atau terselubung dan *graphein* yang artinya menulis, sehingga kurang lebih artinya adalah "menulis tulisan yang tersembunyi atau terselubung".

Steganografi seperti disebutkan diatas melibatkan dua komponen pokok yaitu pesan dan wadah. Pesan yaitu sesuatu yang akan disisipkan pada wadah sedangkan wadah adalah media tempat pesan disisipkan. Pesan maupun wadah bisa berupa citra, *audio*, *video*, teks maupun jenis data yang lain. Untuk meningkatkan tingkat keamanan data yang disimpan, dapat dilakukan dengan menambahkan properti kunci rahasia (*secret key*). Properti kunci ini dapat berupa kunci simetris maupun kunci umum (*public*) atau tersendiri (*private*). Berkas hasil dari proses steganografi sering disebut sebagai berkas *stego* (*stego file*) atau *stego* objek.

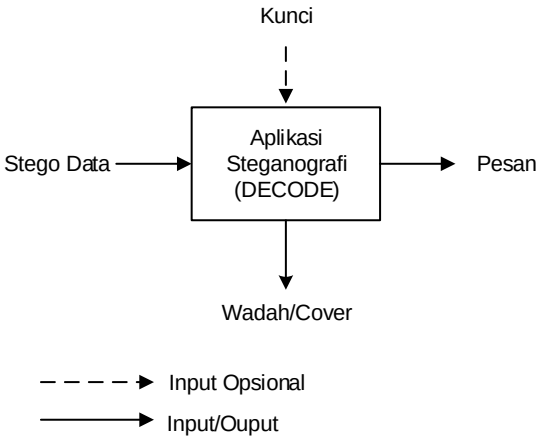
Blok diagram penyembunyian data dalam steganografi secara umum adalah sebagai berikut :



Gambar 1. Block diagram penyembunyian data

Data atau informasi yang akan disembunyikan disimpan dalam sebuah wadah (*cover*) melalui suatu algoritma steganografi tertentu (misalnya LSB). Untuk menambah tingkat keamanan data dapat diberikan kunci agar tidak semua orang mampu mengungkapkan data yang disimpan dalam berkas wadah (*cover*). Hasil akhir dari proses penyembunyian data ini adalah sebuah berkas *stego* (*stego data*/*stego file*).

Sedangkan blok diagram pengungkapan informasi dari berkas stego adalah sebagai berikut :



Gambar 2. Block diagram pengungkapan data

Berkas *stego* diekstrak setelah memasukkan kunci yang dibutuhkan. Hasil ekstraksi ini adalah informasi atau data yang disimpan beserta berkas *stego*. Dalam kebanyakan teknik steganografi, ekstraksi pesan tidak akan mengembalikan berkas *stego* tepat sama dengan berkas wadah (*cover*) saat pesan disimpan, karena saat penyimpanan pesan tidak dilakukan pencatatan kondisi awal dari berkas wadah yang digunakan untuk menyimpan pesan. Dengan demikian jika diingikan penghilangan pesan dari berkas *stego* maka yang dapat dilakukan diantaranya adalah dengan melakukan pengubahan nilai piksel secara acak dari tempat piksel–piksel pesan disimpan dalam berkas wadah (*cover*).

2.2. Metode Modifikasi LSB

Metode modifikasi LSB (*Least Significant Bit modification*) tergolong metode yang menggunakan teknik substitusi. Metode LSB menyembunyikan data rahasia dalam piksel–piksel tak signifikan (*least significant pixel*) dari berkas wadah (*cover*). Pengubahan nilai piksel–piksel tak signifikan pada dasarnya memberikan pengaruh terhadap berkas wadah, tetapi karena perubahan yang terjadi sangat kecil, sehingga tidak tertangkap oleh indra manusia. Kenyataan inilah yang akhirnya dimanfaatkan sebagai teknik penyembunyian data atau pesan (steganografi). Sebagai ilustrasi cara

penyimpanan data dengan metode LSB, misalnya piksel–piksel wadah berikut :

(10010101 11000101 00101010) → piksel pertama (R, G ,B)
(00011100 10000110 01100110) → piksel kedua (R, G ,B)
(10000111 10010100 00100010) → piksel ketiga (R, G ,B)

Digunakan untuk menyimpan karakter "A" (01000001), maka piksel–piksel wadah tersebut akan dirubah menjadi :

(10010100 11000101 00101010) → piksel pertama (R, G ,B)
(00011100 10000110 01100110) → piksel kedua (R, G ,B)
(10000110 10010101 00100010) → piksel ketiga (R, G ,B)

Perubahan yang tidak signifikan ini tidak akan tertangkap oleh indra manusia (jika media wadah adalah gambar, suara atau video).

Dalam contoh diatas penggantian piksel tak signifikan dilakukan secara terurut. Penggantian piksel tak signifikan juga dapat dilakukan secara tak terurut, bahkan hal ini dapat meningkatkan tingkat keamanan data (*imperceptibility*). Disamping itu juga mungkin melakukan pengubahan piksel tidak pada bagian awal berkas wadah. Pengubahan piksel juga dapat dipilih mulai dari tengah, atau dari titik lain dari berkas wadah yang dimungkinkan untuk menyimpan seluruh informasi rahasia tanpa menimbulkan permasalahan saat pengungkapan data.

Steganografi dengan metode LSB hanya mampu menyimpan informasi dengan ukuran yang sangat terbatas. Misalnya suatu citra 24-bit (R=8-bit, G=8-bit, B=8-bit) digunakan sebagai wadah untuk menyimpan data berukuran 100 bit, jika masing–masing komponen warnanya (RGB (*Red Green Blue*)) digunakan satu piksel untuk menyimpan informasi rahasia tersebut, maka setiap pikselnya disimpan 3 bit informasi, sehingga setidaknya dibutuhkan citra wadah berukuran 34 piksel atau setara 34 x 3 x 8 = 816 bit (8 kali lipat). Jadi suatu citra 24-bit jika digunakan untuk menyimpan informasi rahasia hanya mampu menampung informasi maksimum berukuran 1/8 (seperdelapan) dari ukuran citra penampung tersebut.

2.3. Kompresi Dengan Format GZIP (Algoritma Deflate)

Algoritma pemampatan data dengan format data GZIP termasuk dalam algoritma kompresi atau pemampatan yang bersifat *lossless*. Berbeda dengan algoritma pemampatan yang bersifat *lossy* yang menghilangkan sebagian informasi dari berkas yang di mampatkan untuk mendapatkan hasil yang optimum, algoritma kompresi yang bersifat *lossless* seperti GZIP tidak membuang sedikitpun informasi yang dimiliki oleh berkas asal. Algoritma kompresi yang bersifat *lossy* umumnya digunakan untuk memampatkan berkas–berkas gambar, video ataupun suara, hal ini menimbang perubahan (penghilangan) sedikit pada berkas asal tidak akan menimbulkan efek yang mampu ditangkap oleh indra manusia. Sedangkan algoritma kompresi yang bersifat *lossless* umumnya digunakan untuk berkas teks atau *binary* (*executable*).

Berkas termampatkan dengan format gzip dibuat dengan menggunakan algoritma kompresi *deflate*. Sebagaimana format zip berkas terkompresi dengan format gzip dibuat dengan algoritma *deflate* yang pertama kali didisain oleh Philip Katz (1962-2000), algoritma *deflate* sendiri merupakan algoritma yang berbasiskan algoritma LZ77 dan kode Huffman (*Huffman Codes*). Spesifikasi format kompresi gzip distandarisasi melalui RFC1952 yang ditulis oleh Peter Deutsch.

Meskipun algoritma *deflate* tidak dirancang untuk suatu tipe berkas secara spesifik, akan tetapi metode–metode pemampatan data yang dirancang khusus untuk tipe berkas tertentu, yang umumnya memiliki kerumitan yang lebih tinggi, umumnya memiliki performansi (dalam segi ukuran berkas hasil kompresi) yang lebih tinggi. Pada umumnya algoritma *deflate* (termasuk gzip) memiliki nilai faktor kompresi (*compression factor*) antara 2.5 sampai 3 untuk pemampatan berkas tipe teks dan memiliki nilai yang lebih kecil jika berkas yang dimampatkan adalah tipe *binary* (*executable*). Faktor kompresi (*compression factor*) merupakan *invers* dari nilai rasio kompresi (*compression ratio*) yang menunjukkan persentase ukuran berkas hasil pemampatan dibandingkan ukuran berkas sebelum dimampatkan.

E-ISSN:2621: 3079
ISSN: 2621:1556

Jadi rasio kompresi (*compression ratio*) ditunjukkan dengan persamaan :

$$RK = \frac{Ukuran\_file\_output}{Ukuran\_file\_input}$$

Sedangkan faktor kompresi (*compression factor*) memiliki persamaan :

$$FK = \frac{Ukuran\_file\_input}{Ukuran\_file\_output}$$

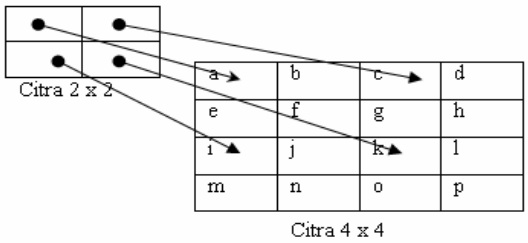
Dari persamaan tersebut terlihat, bahwa rasio kompresi akan selalu bernilai kurang dari 1, jadi jika suatu algoritma kompresi memiliki nilai rasio kompresi 0,5 maka algoritma ini mampu memampatkan berkas hingga menjadi separuh (50%) dari ukuran semula. Jadi semakin kecil nilai rasio kompresi dari suatu algoritma kompresi maka semakin bagus algoritma tersebut. Berkebalikan dengan nilai rasio kompresi adalah nilai faktor kompresi. Sehingga semakin besar nilai faktor kompresi dari suatu algoritma pemampatan data, maka algoritma tersebut berarti semakin baik. Pada umumnya nilai faktor kompresi lebih sering digunakan sebagai standar ukuran mengingat secara alamiah nilainya menunjukkan tingkat keandalan dari suatu algoritma (semakin besar nilai ≡ semakin bagus kualitas).

2.4. Tinjauan Umum Berkas Citra 24-bit

Citra digital merupakan suatu larik dua dimensi atau suatu matriks yang merepresentasikan intensitas cahaya yang bervariasi pada piksel. Piksel adalah titik di layar monitor yang dapat diatur untuk menampilkan warna tertentu. Setiap piksel terdiri dari susunan tiga warna merah, hijau dan biru (RGB). Piksel disusun di layar monitor dalam susunan baris dan kolom . Susunan piksel dalam baris dan kolom ini yang dinamakan resolusi monitor. Resolusi monitor yang sering dijumpai adalah 640x480, 800x600, 1024x768. Resolusi 640x480 akan menampilkan piksel sejumlah 640 baris dan 480 kolom, sehingga total piksel yang digunakan 640x480 = 307.200 piksel. Melalui piksel inilah suatu gambar dapat dimanipulasi untuk menyimpan informasi yang akan digunakan sebagai salah satu pengimplementasian steganografi.

2.5. Perbesaran Citra Dengan Model Interpolasi

Dalam melakukan perbesaran ukuran citra, maka setiap piksel pada citra asal harus dipetakan pada sekumpulan piksel yang ada pada citra yang berukuran lebih besar. Tingkat efektivitas dari algoritma untuk menghasilkan perbesaran citra tersebut sangat menentukan citra hasil perbesaran. Misalkan dikehendaki perbesaran citra yang berukuran 2x2 dengan faktor perbesaran citra sebesar 2, maka hasil perbesaran citra adalah citra baru berukuran 4x4.



Gambar 3. Perbesaran citra 2x2 dengan faktor perbesaran 2

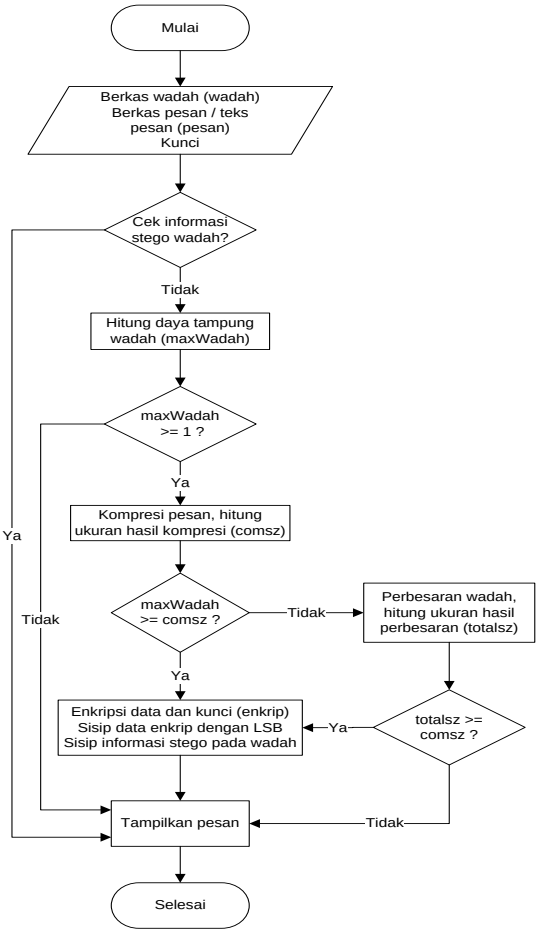
Pada perbesaran citra pada Gambar 3 diberikan ilustrasi perbesaran citra ukuran 2x2 dengan faktor perbesaran 2 sehingga menghasilkan citra baru berukuran 4x4. Terlihat pada Gambar 3 tersebut setiap piksel dari citra asal (citra ukuran 2x2) dipetakan pada piksel–piksel citra hasil (a,c,i, dan k).

Teknik termudah (tersederhana) dan tercepat dalam pengisian informasi setiap piksel yang masih kosong pada citra hasil perbesaran tersebut adalah dengan cara melakukan duplikasi citra asal. Sehingga jika cara ini yang dipilih, nilai piksel b diduplikasi (*copy*) dari piksel a sehingga nilai piksel b pada citra hasil akan memiliki nilai yang sama dengan nilai piksel a, nilai piksel d diduplikasi dari nilai piksel c, sedangkan nilai–nilai piksel pada baris kedua yaitu piksel e sampai h diperoleh dengan cara menduplikasi nilai–nilai pada baris pertama (piksel a sampai d).

3. ANALISIS DAN PEMBAHASAN

3.1. Diagram Alir Proses Penyembunyian (*Encode*) Data

E-ISSN:2621: 3079
ISSN: 2621:1556



Gambar 4. Diagram alir proses penyembunyian (*encode*) data

Aplikasi yang akan dibuat membutuhkan masukan berupa berkas wadah "2x2.jpg", pesan berupa teks "A" dan kunci berupa teks "OK". Kemudian berkas wadah dicek apakah ada informasi *stego* pada isi (*content*) berkas dibagian eof (*end of file*). Jika ada maka tampilkan pesan bahwa berkas wadah tersebut terdapat informasi *stego* dan selesai. Jika tidak maka mulai menghitung daya tampung berkas wadah sebagai berikut :

Jenis satuan yang dipakai adalah piksel, byte dan bits, yaitu :

1 Bits = 0 atau 1
1 Byte = 8 bits
1 Pixel = 3 byte
1 Word = 2 Bytes = 2 x (8 bits) = 16 bits
Double Word = 4 Bytes = 4 x (8 bits)= 32 bits

Dari satuan diatas didapat maksimal daya tampung berkas wadah :

- Ukuran : lebar piksel x tinggi piksel → 2 x 2 = 4 piksel
- RGB : total_piksel x 3 byte (RGB) → 4 x 3 = 12 byte
- Bits : total_RGB x 8 bits (total_bits) → 12 x 8 = 96 bits
- Tampung : total_RGB / 8 bits (total_bits) → 12 / 8 = 1 byte

Jadi maksimal daya tampung (maxWadah) berkas wadah gambar 24 bits "2x2.jpg" adalah 1 byte. Byte biasanya merepresentasikan sebuah karakter (misalkan seperti A, ?, -, dll). Karena minimal data pesan yang akan disisipkan adalah 1 byte maka berkas wadah memiliki daya tampung minimal 1 byte juga.

Setelah menghitung maxWadah kemudian maxWadah dicek apakah lebih besar sama dengan 1 byte. Jika tidak maka tampilkan pesan bahwa wadah tersebut memiliki maxWadah kurang dari 1 byte dan selesai. Jika ya maka lakukan pengkompresian pesan sebagai berikut : Kompresi data menggunakan format GZIP. Kompresi dilakukan jika nilai hasil kompresi lebih kecil dari nilai awal dan jika nilai hasil kompresi lebih besar dari nilai awal maka tidak dilakukan kompresi. Pesan teks "A" panjangnya 1 byte jika dikompresi akan menjadi "s " dengan panjang 3 byte. Nilai hasil kompresi lebih besar dari nilai pesan awal maka tidak dilakukan pengkompresian melainkan langsung menggunakan pesan awal tersebut yaitu "A" dengan panjang 1 byte. Maka nilai dari panjang pesan (comsz) adalah 1 byte.

Setelah menghitung comsz kemudian maxWadah dicek apakah lebih besar sama dengan comsz. Jika ya maka lakukan proses enkripsi.

3.2. Proses Enkripsi (*Encryption*) Data

Mode enkripsi data yang digunakan adalah mode operasi enkripsi *block cipher* mode ECB (*Elecktronik Code Book*). Pada mode ini, setiap blok plainteks dienkripsi secara individual dan independen. Secara matematis, enkripsi mode ECB dinyatakan sebagai $Ci = EK(Pi)$ dan dekripsi sebagai $Pi = DK(Ci)$ yang dalam hal ini, *Pi* dan *Ci* masing-masing blok plainteks dan cipherteks ke-*i*. Operasi *block cipher* mode ECB termasuk jenis kriptografi simetri yaitu menggunakan kunci enkripsi yang sama dengan kunci dekripsinya.

E-ISSN:2621: 3079
ISSN: 2621:1556

Proses enkripsi yang dilakukan adalah dengan meng-XOR-kan *Pi* dengan *K1* yang akan menghasilkan *C1*, kemudian meng-XOR-kan kembali *C1* dengan *K2* yang nantinya akan menghasilkan *C2*. Hasil dari *C2* ini akan digeser secara *wrapping* bit-bit satu posisi ke kiri. Hasil pergeseran ini adalah merupakan hasil Enkripsi (*cipherteks*) antara teks-asli (*plainteks*) dengan kunci (*key*). Prosesnya adalah :

Lambang XOR adalah ⊕

A
Pi = 0100 0001
K1 = 0100 1111 ⊕
C1 = 0000 1110
K2 = 0100 1011 ⊕
C2 = 0100 0101

Hasil *C2* kemudian digeser 1 bit ke kiri menjadi "1000 1010" dalam notasi HEX adalah "8a".

Hasil keluaran dari proses enkripsi kemudian dilakukan proses penyembunyian atau penyisipan (*encode*) data menggunakan metode modifikasi LSB.

3.3. Proses Penyembunyian (*Encode*) Data

Isi gambar dari berkas wadah "2x2.jpg" adalah (merah), hasil pengkonversian dari tiap-tiap piksel ke biner adalah :

00000000.00000000. 11111110	00000000.00000000. 11111110
00000000.00000000. 11111110	00000000.00000000. 11111110

Proses penyembunyian/penyisipan (*encode*) bit-bit pesan hasil enkripsi adalah "10001010" ke tiap bit-bit piksel gambar wadah yaitu :

Melakukan dua kali perulangan sebanyak lebar dan tinggi piksel. Misalkan pada perulangan pertama dilakukan pada bit-bit "00000000.00000000.11111110". Pada bit-bit ini tergabung tiga macam warna dari RGB dalam biner. Jika dibilah berdasarkan satu warna dalam biner adalah R = 00000000, G = 00000000 dan B = 11111110. Kemudian bit-bit dari R diambil satu bit paling akhir yaitu "0" untuk digantikan dengan satu bit pertama dari bit pesan yaitu "1" maka akan menjadi "0000000**1**", begitu juga dengan satu bit paling akhir dari G yaitu "0" diganti dengan satu bit berikutnya dari bit pesan yaitu "0" menjadi "0000000**0**", dan satu bit paling akhir dari B yaitu "0" diganti dengan satu bit berikutnya dari bit pesan yaitu "0" menjadi "1111111**0**". Jika digabungkan ketiga biner tersebut dari RGB akan menjadi "00000000. 00000000. 11111110".

Begitu juga dengan perulangan terhadap bit-bit berikutnya sama dengan penjelasan proses diatas adalah sebagai berikut :

Bit-bit berikutnya "00000000.00000000.11111110" yaitu :
R = 00000000 → 0 diganti 0 → 00000000
G = 00000000 → 0 diganti 1 → 00000001
B = 11111110 → 0 diganti 0 → 11111110

Bit-bit berikutnya "00000000.00000000.11111110" yaitu :
R = 00000000 → 0 diganti 1 → 00000001
G = 00000000 → 0 diganti 0 → 00000000
B = 11111110 → 0 tidak diganti → 11111110

Bit-bit berikutnya "00000000.00000000.11111110" yaitu :
R = 00000000 → 0 tidak diganti → 00000000
G = 00000000 → 0 tidak diganti → 00000000
B = 11111110 → 0 tidak diganti → 11111110

Proses-proses diatas menghasilkan bit-bit data baru sebagai berikut :

00000001.00000000. 11111110	00000000.00000001. 11111110
00000001.00000000. 11111110	00000000.00000000. 11111110

Data inilah yang akan menjadi hasil keluaran (*output*) dari proses penyembunyian data antara berkas wadah dengan pesan berupa berkas *stego* (*stego file*) gambar 24 bits berekstensi png.

3.4. Penyisipan Informasi Steganografi

Agar berkas *stego* dapat dikenali atau diidentifikasi maka dilakukan penyisipan data secara langsung berupa teks informasi steganografi pada bagian akhir dari isi (*content*) berkas wadah gambar disebut eof (*end of file*) dengan format sebagai berikut :

1. Teks:
   tipe_stegopanjang_data|status_kompresi.
   Contoh: tsteg8|0 → tidak dilakukan kompresi
2. Berkas:
   tipe_stegonama_file|ekstensi|lebar_tinggi|panjang_data |status_kompresi.
   Contoh: fsteg2x2|JPG|2_2|7|1 → kompresi: fsteg3□0□□ p□1□7□1□1□

Pada contoh diatas, pesan yang akan disembunyian berupa teks "A" panjangnya 8 bit. Teks "A" ini tidak dilakukan kompresi. Maka

E-ISSN:2621: 3079
ISSN: 2621:1556

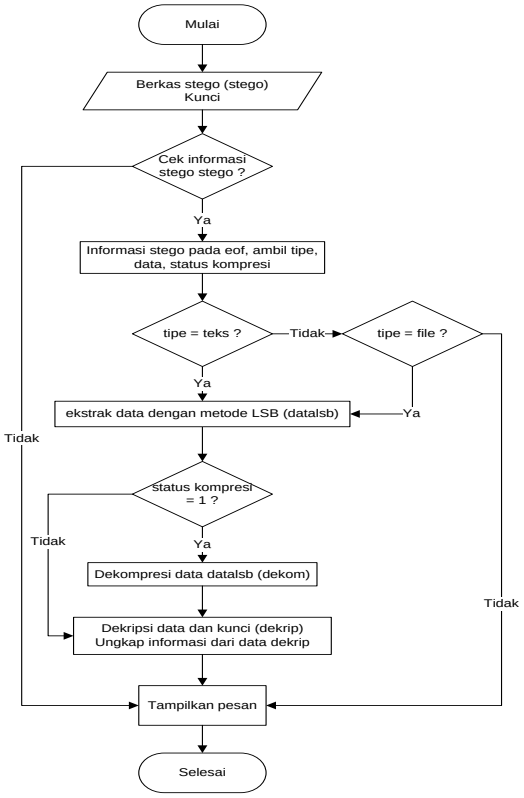
informasi *stego* yang akan disisipkan pada berkas wadah adalah "teks8|0".
Setelah melakukan proses enkripsi, proses penyembunyian data dengan metode modifikasi LSB dan proses penyisipan informasi *stego* pada berkas maka proses *flowchart* selanjutnya adalah menampilkan hasil keluran (*output*) berupa berkas *stego* (*stego file*).

3.5. Perbesaran Citra Wadah

Nilai faktor perbesaran adalah 2 (dua) yang akan dikalikan dengan tinggi dan panjang piksel gambar yaitu tinggi piksel = 2 dan panjang piksel = 2 maka hasil pembesaran adalah (tinggi x 2) dan (panjang x 2) menjadi (2 x 2 = 4) dan (2 x 2 = 4), maka berkas gambar baru hasil pembesaran adalah 4x4 piksel. Kemudian menghitung kembali hasil pembesaran tersebut (totalsz) sebagai berikut :

- Ukuran : lebar piksel x tinggi piksel → 4 x 4 = 16 piksel
- RGB : total_ piksel x 3 byte (RGB) → 16 x 3 = 48 byte
- Bits : total_RGB x 8 bits (total_bits) → 48 x 8 = 384 bits
- Tampung : total_RGB / 8 bits (total_bits) → 48 / 8 = 6 byte

Jadi maksimal daya tampung (totalsz) berkas wadah gambar 24 bits "2x2.jpg" adalah 6 byte (6 karakter). Kemudian totalsz dicek apakah lebih besar sama dengan comsz. Jika tidak maka tampilkan pesan bahwa berkas wadah tersebut tidak memiliki daya tampung yang besar untuk menampung pesan dan selesai. Jika ya maka lakukan enkripsi pesan dengan kunci dimana proses enkripsi ini telah dijelaskan pada sub **3.2** kemudian hasil enkripsi tersebut disembunyikan atau disisipkan ke dalam piksel wadah menggunakan metode modifikasi LSB dimana proses penyembunyian atau penyisipan ini telah dijelaskan pada sub **3.3** dan dilakukan penyisipan informasi *stego* pada wadah dibagian eof (*end of file*) dari isi (*content*) wadah dimana proses penyisipan informasi *stego* ini telah dijelaskan pada sub **3.4**. Setelah itu tampilkan hasil keluaran (*output*) berupa berkas *stego* (*stego file*).



Gambar 5. Diagram alir proses pengungkapan (*decode*) data

Aplikasi yang akan dibuat membutuhkan masukan berupa berkas *stego* "2x2.png" dan kunci berupa teks "OK". Kemudian berkas *stego* dicek apakah ada informasi *stego* pada isi (*content*) berkas dibagian eof (*end of file*). Jika tidak maka tampilkan pesan bahwa berkas *stego* tersebut tidak terdapat informasi *stego* dan selesai. Jika ya maka mulai mengambil potongan informasi *stego* yang terdapat pada berkas dibagian akhir atau eof (*end of file*) dari isi (*content*) berkas *stego* yaitu "teks8|0". Potongan isi ini menjelaskan bahwa berkas *stego* menyimpan informasi bertipe teks dengan panjang 8 bits atau 1 byte (1 karakter) dan tidak dikompresi.
Kemudian tipe dari potongan informasi *stego* dicek apakah bertipe teks atau bertipe file. Jika ya maka lakukan pengekstrakan data dengan metode modifikasi LSB.

3.6. Proses Pengungkapan (*Decode*) Data

Isi gambar dari berkas *stego* "2x2.png" adalah (merah) memiliki lebar piksel sama dengan 2 dan tinggi piksel sama dengan 2, hasil

E-ISSN:2621: 3079
ISSN: 2621:1556

pengkonversian dari tiap-tiap piksel ke biner adalah :

00000001.00000000.	00000000.00000001.
11111110	11111110
00000001.00000000.	00000000.00000000.
11111110	11111110

Melakukan dua kali perulangan sebanyak lebar dan tinggi piksel. Misalkan pada perulangan pertama dilakukan pada bit-bit "00000001.00000000.11111110". Pada bit-bit ini tergabung tiga macam warna dari RGB dalam biner. Jika dibilah berdasarkan satu warna dalam biner adalah R = 00000001, G = 00000000 dan B = 11111110. Kemudian bit-bit dari R diambil satu bit paling akhir yaitu "1" untuk disimpan sebagai data pesan maka akan menjadi "1", begitu juga dengan satu bit paling akhir dari G yaitu "0" disimpan sebagai data pesan maka akan menjadi "10", dan satu bit paling akhir dari B yaitu "0" disimpan sebagai data pesan maka akan menjadi "100". Jika digabungkan ketiga biner tersebut dari RGB akan menjadi "100".

Begitu juga dengan perulangan terhadap bit-bit berikutnya sama dengan proses penjelasan diatas adalah sebagai berikut :

Bit-bit berikutnya "00000000.00000001.11111110" yaitu :

R = 00000000 → 0 → 1000
G = 00000001 → 1 → 10001
B = 11111110 → 0 → 100010

Bit-bit berikutnya "00000001.00000000.11111110" yaitu :

R = 00000001 → 1 → 1000101
G = 00000000 → 0 → 10001010
B = 11111110 → 0 → 100010100

Bit-bit berikutnya "00000000.00000000.11111110" yaitu :

R = 00000000 → 0 → 1000101000
G = 00000000 → 0 → 10001010000
B = 11111110 → 0 → 100010100000

Proses-proses diatas menghasilkan bit-bit data baru yaitu "10001010000" dengan panjang 12 bit. Karena potongan informasi *stego* diatas menjelaskan bahwa informasi data yang disimpan memiiliki panjang 8 bit, maka bit-bit pesan yang diambil dari bit-bit data baru (datalsb) adalah sepanjang 8 bit yaitu "10001010" dalam notasi HEX adalah "8a".

Hasil dari datalsb kemudian dicek apakah pengkompresan data dilakukan. Jika ya maka lakukan pendekompresan data kemudian pada hasil pendekompresan data lakukan pendekripsian data.

3.7. Proses Dekripsi (*Decryption*) Data

Proses dekripsi yang dilakukan adalah *Pi* digeser secara *wrapping* bit-bit satu posisi ke kanan. Hasil pergeseran ini kemudian di-XOR-kan dengan *K1* yang akan menghasilkan *C1*, kemudian meng-XOR-kan kembali *C1* dengan *K2* yang nantinya akan menghasilkan *C2*. Hasil *C2* ini adalah merupakan hasil Dekripsi (*plainteks*) antara teks-tersandi (*cipherteks*) dengan kunci (*key*). Prosesnya adalah :

Lambang XOR adalah ⊕. *Pi* adalah "10001010" digeser 1 bit ke kanan menjadi "01000101" dalam notasi HEX adalah 45.

45
Pi = 0100 0101
K1 = 0100 1111 ⊕
C1 = 0000 1010
K2 = 0100 1011 ⊕
C2 = 0100 0001

Hasil *C2* adalah "0100 0001" dalam notasi HEX adalah "A". Data inilah yang akan menjadi hasil keluaran (*output*) dari proses pengungkapan data antara berkas stego dengan kunci berupa berkas pesan berekstensi txt.

3.8. Pengujian Steganografi Untuk Proses *Encode*

Berikut ini adalah hasil pengujian untuk proses enkode :

Tabel 1. Hasil Pengujian Proses Enkode

Berkas awal
cth_800x600.jpg

Berkas/Teks Pesan
cth_798x563.PNG

E-ISSN:2621: 3079
ISSN: 2621:1556

Berkas Stego
cth_800x600.jpg

Dari Tabel 1. berkas awal yang dijadikan sebagai wadah adalah "cth_800x600.jpg" dengan ukuran piksel 800x600. Kemudian berkas pesan yang akan disisipkan adalah "cth_798x563.PNG" dengan ukuran piksel 798x563. Maka hasil yang didapatkan dari proses enkode adalah berkas *stego* yaitu "cth_800x600.jpg" dengan ukuran piksel 800x600. Dari hasil ini dapat dilihat bahwa perubahan warna pada gambar *stego* secara kasat mata manusia tidak terjadi perubahan atau dengan kata lain warna pada gambar awal dengan warna gambar *stego* adalah sama.

3.9. Pengujian Steganografi Untuk Proses *Decode*

Berikut ini adalah hasil pengujian untuk proses dekode :

Tabel 2. Hasil Pengujian Proses Dekode

Berkas awal
cth_800x600.jpg

Berkas/Teks Pesan
cth_798x563.PNG

Berkas Stego
cth_798x563.PNG

Dari Tabel 2. berkas awal yang dijadikan sebagai wadah adalah "cth_800x600.jpg" dengan ukuran piksel 800x600 merupakan berkas *stego*. Kemudian berkas pesan yang akan diungkapkan adalah "cth_798x563.PNG" dengan ukuran piksel 798x563. Maka hasil yang didapatkan dari proses dekode adalah berkas pesan yaitu "cth_798x563.PNG" dengan ukuran piksel 798x563. Dari hasil ini dapat dilihat bahwa gambar pesan yang disisipkan dapat

E-ISSN:2621: 3079
ISSN: 2621:1556

diungkapkan sesuai dengan gambar awal pesan sebelum disisipkan ke berkas wadah.

3.10. Perubahan Terhadap Berkas Citra

a. Ukuran Berkas

Perubahan ukuran berkas wadah tergantung pada pesan yang ditampung, semakin besar pesan maka semakin besar kapasitas wadah dan semakin kecil pesan maka semakin kecil penambahan kapasitas wadah. Faktor–faktor terjadinya penambahan kapasitas :

- Penyisipan informasi pada EOF (*End Of File*) berkas
- Perubahan bit-bit warna asli ketika terjadi proses enkode dan atau dekode pesan yang tidak signifikan
- Pengkonversian terhadap header berkas asli yaitu berkas bertipe DLL (*Dynamic Link Library*) asli dengan berkas bertipe DLL standar PHP saat proses penyimpanan berkas

b. Warna Gambar

Perubahan bit-bit warna terhadap berkas wadah tergantung pada perubahan bit-bit warna pesan yang disisipkan. Faktor–faktor terjadinya perubahan warna :

- Penyisipan dan penempatan bit-bit warna pesan pada wadah
- Tipe atau format (*extension*) berkas gambar
- Perubahan konversi berkas bertipe DLL (*Dynamic Link Library*) asli dengan DLL standar PHP saat proses penyimpanan berkas

3.11. Pengujian Wadah Citra Dengan Banyak Kata

Hasil pengujian wadah citra dengan banyak kata ditujunkkan pada Table 3.

4. KESIMPULAN

Berdasarkan dari hasil analisis dan pengujian yang telah dilakukan sebelumnya, dapat diambil beberapa kesimpulan diantaranya adalah :

1. Teknik modifikasi LSB yang dilakukan dalam pengujian, penyisipan bit-bit pesan dimulai dari bit terakhir pada bit-bit warna wadah secara terurut.
2. Pengujian pesan lebih besar dari wadah dapat dilakukan dengan perbesaran wadah dengan faktor perbesaran 2 (dua) untuk menampung semua data pesan yang akan disimpan.
3. Pengujian hasil keluaran dari proses enkode disimpan dalam format (tipe) png karena format png tidak membuang sedikitpun informasi yang dimiliki oleh berkas asal.

Tabel 3. Hasil Pengujian Dengan Banyak Kata

Berkas Awal	Berkas Stego
a. Dengan 673 Kata (4704 Karakter)	
cth_1600x1000.jpg 	cth_1600x1000_1.png 
b. Dengan 1028 Kata (7695 Karakter)	
cth_1600x1000.jpg	cth_1600x1000_2.png

	
c. Dengan 2583 Kata (18983 Karakter)	
cth_1600x1000.jpg	cth_1600x1000_3.png
	

Dari Tabel 3. Bagian a berkas awal yang dijadikan sebagai wadah adalah "cth_1600x1000.jpg" dengan ukuran piksel 1600x1000. Kemudian pesan yang akan disisipkan berupa teks sebanyak 673 kata (4704 karakter). Maka hasil yang didapatkan dari proses enkode adalah berkas *stego* yaitu "cth_1600x1000_1.png" dengan ukuran piksel 1600x1000. Dari hasil ini dapat dilihat bahwa perubahan warna pada gambar *stego* secara kasat mata manusia tidak terjadi perubahan atau dengan kata lain warna pada gambar awal dengan warna gambar *stego* adalah sama.
Begitu juga untuk bagian b dan c pada Tabel 3. yaitu berkas awal yang dijadikan sebagai wadah adalah "cth_1600x1000.jpg" dengan ukuran piksel 1600x1000. Kemudian pesan yang akan disisipkan berupa teks sebanyak 1028 kata (7695 karakter) dan 2583 kata (18983 karakter). Maka hasil yang didapatkan dari proses enkode adalah berkas *stego* yaitu "cth_1600x1000_2.png" dan "cth_1600x1000_3.png" dengan ukuran piksel 1600x1000.

5. REFERENSI

1. Gusti N. 2007. Steganografi Pada File Gambar. Skripsi. Teknik Informatika STMIK AKAKOM. Yogyakarta.
2. Hakim M. 2007. Studi dan Implementasi Steganografi Metode LSB dengan Preprocessing Kompresi data dan Ekspansi Wadah.
3. Deutsch P. RFC 1952 GZIP berkas format specification version 4.3. http://rfc-editor.org/. Diakses tanggal 20 Desember 2018.
4. Wikipedia. 2009. Steganografi. http://id.wikipedia.org/wiki/Steganografi. Diakses tanggal 20 Desember 2018.
5. Kurniawan Y. 2004. KRIPTOGRAFI Keamanan Internet dan Jaringan Komunikasi. Penerbit Informatika. Bandung