Hamid Rosyadi, 1610652008 TI SORE

Studi dan Implementasi Steganografi Metode LSB dengan *Preprocessing* Kompresi data dan Ekspansi Wadah

Abstract – Teknik Steganografi dengan menggunakan metode modifikasi LSB *(Least Significant Bit Modification)* adalah teknik yang paling sederhana. Oleh karena itu steganografi dengan metode ini sangat sesuai bagi para pemula dibidang steganografi. Metode modifikasi LSB melakukan penyimpanan data dengan cara mengganti bit – bit yang tidak signifikan (*least significant pixel*) pada berkas (*file*) wadah (*cover*) dengan bit – bit berkas yang akan disimpan.

Salah satu kelemahan dari metode modifkasi LSB adalah ketidakmampuannya dalam menyimpan data dengan ukuran yang besar. Rata – rata teknik steganografi dengan metode modifikasi LSB hanya mampu menyimpan data berukuran seperdelapan dari ukuran wadah (untuk wadah berupa citra 24-bit), tentu saja hal ini tidak efisien. Untuk mengatasi hal tersebut maka dalam makalah ini dikemukakan beberapa metode untuk memperbesar kemampuan teknik steganografi metode modifikasi LSB dalam menyimpan data. Metode yang pertama adalah melakukan preprosessing terhadap berkas data yang akan disimpan yaitu dengan jalan memampatkan (*compression*) data tersebut, kedua adalah melakukan preprosessing terhadap berkas wadah (*cover*) dengan cara memperbesar berkas wadah (*stretching image*), dan yang ketiga adalah menggabungkan metode pertama dan kedua sekaligus. Dalam makalah ini juga akan dilakukan analisis terhadap proses dan hasil dari masing – masing metode tersebut. Untuk meningkatkan keamanan data yang akan disimpan, data yang disimpan juga dienkripsi terlebih dahulu.

Kata kunci : *LSB (Least Significant Bit), stretching image, compression, preprocessing berkas wadah (cover), preprocessing berkas data*

1. PENDAHULUAN

Steganografi (*covered writing*) didefinisikan sebagai ilmu dan seni untuk menyembunyikan pesan rahasia (*hiding message*) sedemikian sehingga keberadaan (eksistensi) pesan tidak terdeteksi oleh indera manusia[4]. Steganografi telah dikenal semenjak tahun 500 SM, dimana Herodotus (sejarawan Yunani) menuliskan pesan pada kepala budak dan menunggu sampai rambut kepalanya tumbuh kembali sehingga pesan tidak terlihat dan selanjutnya dia diutus untuk menyampaikan pesan tersebut tanpa menimbulkan kecurigaan oleh bangsa Persia.

Saat ini dalam dunia digital, teknik steganografi banyak digunakan untuk menyembunyikan informasi rahasia dengan berbagai maksud. Salah satu tujuan dari steganografi adalah mengirimkan informasi rahasia melalui jaringan tanpa menimbulkan kecurigaan. Disamping itu steganografi juga dapat digunakan untuk melakukan autentikasi terhadap suatu hasil karya sebagaimana pemanfaatan watermarking.

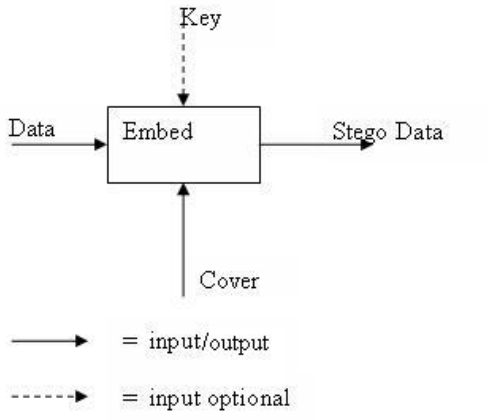
Steganografi memerlukan setidaknya dua properti. Properti pertama adalah wadah penampung (*cover*) dan yang kedua adalah data atau pesan yang disembunyikan. Untuk meningkatkan tingkat keamanan data yang disimpan, dapat dilakukan dengan menambahkan properti kunci (key) rahasia. properti kunci ini dapat berupa kunci simetris maupun kunci public atau privat. Berkas hasil dari proses steganografi sering disebut sebagai berkas stego (*stego file*) atau stego objek.

Properti wadah (cover) yang mungkin digunakan untuk menyimpan pesan dalam steganografi sangat beragam. Medium wadah tersebut antara lain citra, suara, video ataupun teks. Adapun data yang disimpan juga dapat berupa audio, citra, video maupun teks. Pertimbangan pemilihan penggunaan kunci dari segi tipe (simetris, public/privat) serta panjang kunci adalah suatu hal yang juga berperan penting dalam pengamanan data yang tersimpan dalam steganografi, disamping menjadi pertimbangan tingkat kemudahan saat ekstraksi data.

Steganografi berbeda dengan kriptografi. Jika dalam kriptografi pesan yang dirahasiakan tidak disembunyikan, seorang kriptanalis dapat membaca pesan dalam format yang terenkripsi dan juga melakukan dekripsi data, maka dalam steganografi yang pertama kali harus dilakukan oleh seorang steganalis adalah menemukan stego objek terlebih dahulu, hal ini karena pesan yang dirahasiakan disembunyikan (tidak nampak) dalam medium lain (*cover*).

Skema penyembunyian data dalam steganografi secara umum adalah sebagaimana ditunjukkan pada Gambar 1. Pada Gambar 1 data atau informasi yang ingin 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 penyimpanan data ini adalah sebuah berkas stego (*stego data/stego file*).



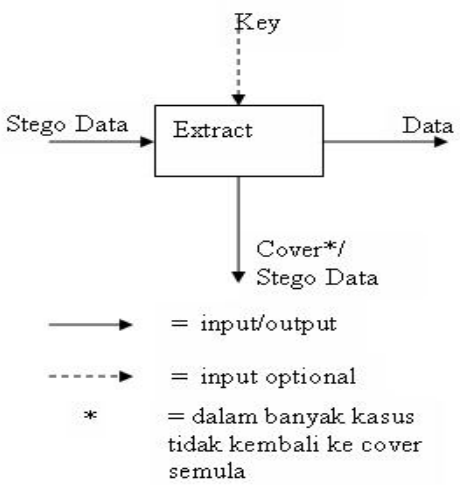
Gambar 1 Penyembunyian Data

Sedangkan proses pengungkapan informasi dari berkas stego digambarkan pada Gambar 2. 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, hal ini 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 pixel secara acak dari tempat pixel – pixel pesan disimpan dalam berkas wadah (cover).

Dalam pengungkapan data atau informasi dari berkas stego terkadang dibutuhkan berkas wadah yang asli. Hal ini contohnya dilakukan untuk pengungkapan data pada stego file yang dibuat dengan algoritma DCT (*Discrete Cosine Transform*).

Saat ini setidaknya terdapat 6 teknik steganografi[1], yaitu:

1. Teknik substitusi (*Substitution Techniques*), yaitu dengan melakukan substitusi pixel – pixel tertentu dari berkas wadah (*cover*) dengan pixel – pixel informasi yang disimpan. Contohnya adalah metode LSB
2. Teknik Domain Transformasi (*Domain Transform Techniques*), yaitu dengan cara menyimpan informasi rahasia pada transformasi ruang (misalnya domain frekuensi) dari berkas wadah (*cover*). Contoh algoritma yang tergolong teknik ini adalah steganografi pada domain DCT (*Discrete Cosine Transform*).
3. Teknik Spread Spectrum (*Spread Spectrum Techniques*), ide dari teknik ini mengadopsi teknik *spread spectrum* pada saluran komunikasi. Dalam teknik ini informasi rahasia yang disimpan disebarkan pada suatu frekuensi tertentu dari berkas wadah (*cover*).
4. Teknik Statistik (*Statistical Techniques*), Dengan teknik ini data diencoding melalui pengubahan beberapa informasi statistik dari berkas wadah (*cover*). Berkas wadah di bagi dalam blok – blok dimana setiap blok tersebut menyimpan satu pixel informasi rahasia yang disembunyikan. Jika pixel yang ditemukan pada suatu blok untuk menyimpan data adalah pixel '1' maka tidak dilakukan perubahan nilai pixel, jika sebaliknya maka dilakukan perubahan nilai pixel. Meskipun secara teoritis mungkin untuk dilakukan, pada kenyataannya teknik ini agak sulit untuk diimplementasikan.
5. Teknik Distorsi (*Distortion Techniques*), informasi yang hendak disembunyikan disimpan berdasarkan distorsi sinyal.
6. Teknik Pembangkitan Wadah (*Cover Generation Techniques*), Teknik ini menyembunyikan informasi rahasia dengan jalan (atau sejalan) dengan pembangkitan wadah (*cover)*. Misalnya aplikasi pembangkitan teks / paragraf bahasa inggris otomatis.



Gambar 2 Pengungkapan Data

2. STEGANOGRAFI DENGAN METODE MODIFIKASI LSB

Salah satu metode steganografi yang banyak digunakan adalah metode modifikasi LSB (*Least significant pixel modification*). Metode modifikasi LSB tergolong metode yang menggunakan teknik substitusi. Metode LSB terutama digunakan untuk steganografi berbasis media (*media-based steganography*) [6].

Metode LSB menyembunyikan data rahasia dalam pixel – pixel tak signifikan (*least significant pixel*) dari berkas wadah (*cover*). Pengubahan nilai pixel – pixel 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 pixel-pixel wadah berikut

01001101 00101110 10101110 10001010 10101111 10100010 00101011 101010111

Digunakan untuk menyimpan karakter 'H' (**01001000**), maka pixel – pixel wadah tersebut akan dirubah menjadi

0100110**0** 0010111**1** 1010111**0** 1000101**0** 1010111**1** 1010001**0** 0010101**0** 101010110

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

Dalam contoh diatas penggantian pixel tak signifikan dilakukan secara terurut. Penggantian pixel tak signifikan juga dapat dilakukan secara tak terurut, bahkan hal ini dapat meningkatkan tingkat keamanan data (*imperceptability*). Disamping itu juga mungkin melakukan pengubahan pixel tidak pada bagian awal berkas wadah. Pengubahan pixel 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.

Meskipun metode LSB mudah diterapkan, akan tetapi steganografi dengan metode ini akan menghasilkan berkas stego yang mudah rusak (dirusak). Dengan menggunakan metode LSB sebagai teknik steganografi, perubahan sedikit pada berkas stego sangat mungkin juga akan merusak informasi rahasia yang tersimpan di dalamnya.

Disamping kemungkinan kerusakan informasi yang tersimpan dalam berkas stego akibat perubahan pada berkas stego, steganografi dengan metode LSB juga 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) digunakan satu pixel untuk menyimpan informasi rahasia tersebut, maka setiap pixelnya disimpan 3 bit informasi, sehingga setidaknya dibutuhkan citra wadah berukuran 34 pixel 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 dari ukuran citra penampung tersebut.

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*). Hal ini mengingat perubahan yang kecil pada berkas yang dikompresi akan memberi pengaruh besar pada berkas hasil kompresi saat di dekompresi ulang. Misalnya pada suatu berkas program computer (*source code*), perubahan yang terjadi walaupun sedikit akan berakibat pada kesalahan kode program tersebut saat di kompilasi setelah di dekompresi.

Berkas termampatkan dengan format gzip dibuat dengan menggunakan algoritma kompresi *deflate*. Sebagaimana format zip berkas terkompresi dengan format gzip dibuat dengan algoritma *deflate* yang pertamakali didisain oleh Philip Katz (1962-2000), algoritma deflate sendiri merupakan algoritma yang berbasiskan algoritma LZ77 dan kode Huffman (*Huffman Codes*). Spesifikasi format kompresi gzip distandardisasi 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 yang berkas dimampatkan adalah tipe binary (*executable*). Factor kompresi (*compression factor*) merupakan invers dari nilai rasio kompresi (*compression ratio*) yang menunjukkan persentase ukuran berkas hasil pemampatan dibandingkan ukuran berkas sebelum dimampatkan.

Jadi rasio kompresi ditunjukkan dengan persamaan:

$$RK = \frac{Ukuran\_file\_Output}{Ukuran\_File\_Input}$$

RK = rasio kompresi

Persamaan 1 Rasio Kompresi

Sedangkan faktor kompresi memiliki persamaan:

$$FK = \frac{Ukuran\_File\_Input}{Ukuran\_File\_Output}$$

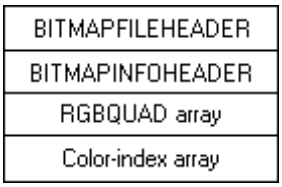
FK = faktor kompresi

Persamaan 2 Faktor Kompresi

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 standard ukuran mengingat secara alamiah nilainya menunjukkan tingkat keandalan dari suatu algoritma (semakin besar nilai ≡ semakin bagus kualitas).

4. Struktur berkas citra tipe bitmap 24-bit

Properti wadah (*cover*) yang digunakan dalam makalah ini adalah berupa citra bitmap 24-bit, struktur data citra tersebut adalah sebagai berikut:



Gambar 3 Struktur File Bitmap

a. BITMAPFILEHEADER

Merupakan header dari file, berisi informasi tipe file (”BM”), ukuran file dalam byte, dan informasi jumlah *offset* byte antara header dan data bitmap yang sebenarnya.

b. BITMAPINFOHEADER

Menyimpan informasi ukuran panjang dan lebar file dalam satuan pixel, format warna (jumlah bidang warna / *bits-per-pixel*), informasi apakah bitmap terkompresi atau tidak serta tipe kompresinya, jumlah data bitmap dalam byte, resolusi, dan jumlah warna yang digunakan.

c. RGBQUAD array

Berisi informasi intensitas RGB untuk setiap komponen warna pada *pallete*.

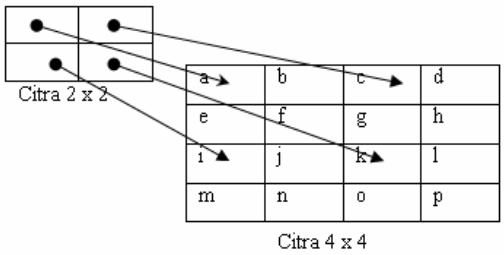
d. Color-index array

Merupakan data file bitmap yang sebenarnya, pada bagian inilah informasi dapat disimpan dengan teknik steganogeafi metode LSB.

5. Perbesaran citra dengan model interpolasi

Dalam melakukan perbesaran ukuran citra, maka setiap pixel pada citra asal harus dipetakan pada sekumpulan pixel yang ada pada citra yang berukuran lebih besar. Tingkat efektivitas dari algoritma untuk menghasilkan perbesaran citra tersebut sangat menentukan citra hasil perbesaran. Pada umumnya algoritma yang mampu menghasilkan perbesaran citra dengan kualitas terbaik membutuhkan waktu pemrosesan dan sumber daya (*resource*) lain yang lebih besar pula.

Misalkan dikehendaki perbesan citra yang berukuran 2 x 2 dengan faktor perbesan citra sebesar 2, maka hasil perbesaran citra adalah citra baru berukuran 4 x 4.



Gambar 4 Perbesan Citra 2 x 2 dengan Faktor Perbesan 2.

Pada perbesaran citra pada Gambar 4 diberikan ilustrasi perbesaran citra ukuran 2 x 2 dengan faktor perbesaran 2 sehingga menghasilkan citra baru berukuran 4 x 4. Terlihat pada gambar tersebut setiap pixel dari citra asal (citra ukuran 2 x 2) dipetakan pada pixel – pixel citra hasil (**a**,**c**,**i**, dan **k**). Adapun pixel – pixel yang tersisa pada citra hasil juga harus diisi dengan nilai yang sesuai, sehingga tidak merusak hasil dari pemetaan ini.

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

Algoritma yang lebih baik untuk melakukan pengisian nilai dari pixel – pixel kosong pada berkas hasil perbesaran adalah dengan cara melakukan interpolasi nilai pixel – pixel kosong tersebut dari nilai pixel – pixel yang ada di sampingnya (*nearest neighbors*). Pada dasarnya terdapat banyak algoritma yang mengimplementasikan teknik ini, akan tetapi kemampuan algoritma tersebut bervariasi. Tantangan terbesar dalam pengimplementasian algoritma ini terdapat pada kecepatan pemrosesan. Salah satu teknik yang cukup sederhana dengan menggunakan metode

nearest neighbors adalah dengan menghitung nilai rata – rata dari pixel – pixel yang terdapat pada sebelah kanan dan kiri pixel yang hendak diberi nilai. Dengan metode ini nilai – nilai pada pixel – pixel **b** dan **j** pada Gambar 4 diperoleh dengan menghitung masing – masing nilai rata – rata pada pixel **a** dan **c** serta **i** dan **k**. Nilai – nilai warna dari pixel **e** diperoleh dengan cara menghitung nilai rata – rata dari nilai warna pada pixel **a** dan **i** begitu juga untuk nilai – nilai warna pada pixel **f** dan **g**, yaitu nilai rata – rata dari **b** dan **j** serta **c** dan **k**. adapun cara penentuan nilai – nilai pixel yang tidak memiliki dua tetangga (kiri-kanan) yaitu pixel – pixel yang terletak di sudut gambar seperti pixel **d** ditentukan dengan cara melakukan duplikasi dari nilai yang ada pada tetangganya yang dalam hal ini adalah pixel **c**, sedangkan nilai pixel **h** diperoleh dari nilai rata – rata pada nilai warna pixel **d** dan **l**. Proses yang dilakukan saat penggunaan algoritma ini adalah dengan cara:

1. duplikasi nilai – nilai pixel citra asal pada citra hasil sesuai dengan tempatnya (misal: a, c, i , dan k)
2. lakukan iterasi pada setiap baris tempat pixel – pixel pada langkah pertama diduplikasikan (missal: baris 1 dan 3) dan isi nilai – nilai pixel yang masih kosong dengan nilai baru yang diperoleh dengan menghitung nilai rata – rata dari pixel – pixel yang bertetanggan dengan pixel tersebut.
3. lakukan iterasi pada baris – baris yang masih belum memiliki nilai dan isi nilai pada setiap pixelnya dengan cara menghitung nilai rata – rata dari nilai warna pixel pada bagian atas dan bawahnya.
4. baris terakhir (baris ke-4) diisi dengan cara menyalin dari nilai yang ada pada baris ketiga.

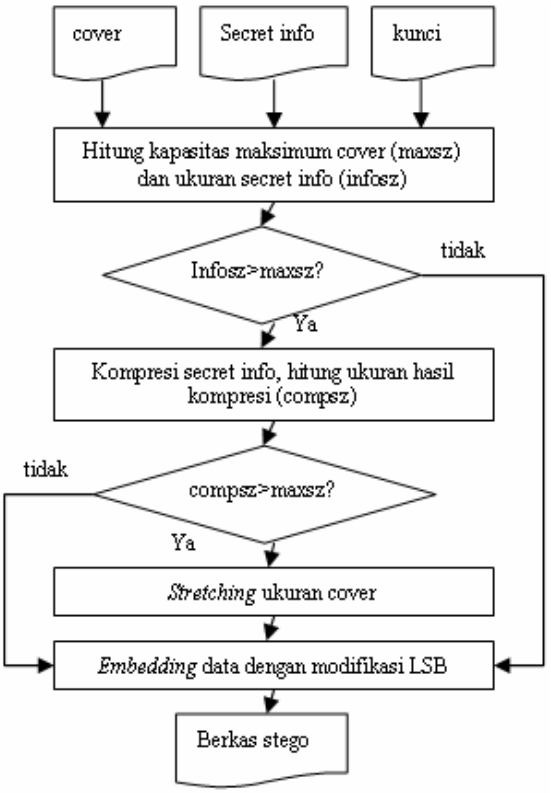
Adapun salah satu cara penghitungan nilai rata – rata dari dua pixel adalah dengan menghitung nilai rata – rata dari komponen warna (*red, green, blue*) dari kedua pixel dan melakukan pencarian nilai yang paling mendekati dengan nilai tersebut pada *pallete*. Meskipun teknik penghitungan rata – rata ini cukup masuk akal akan tetapi pada dasarnya teknik ini bukanlah teknik yang paling baik dalam menghitung nilai rata – rata dari dua buah pixel.

6. Pemanfaatan kompresi dan perbesaran citra untuk meningkatkan kemampuan metode LSB

Steganografi dengan metode modifikasi LSB (*least significant bits modification*) terbukti hanya mampu menyembunyikan informasi yang berukuran minimal, hal ini mendorong dikembangkannya suatu teknik tambahan untuk meningkatkan kemampuan suatu aplikasi steganografi yang menggunakan metode modifikasi LSB. Teknik ini pada dasarnya tidak mengubah proses penyimpanan datanya sendiri (tidak merubah teknik modifikasi LSB), akan tetapi teknik ini mengubah atau menambah proses sebelum proses penyimpanan informasi dilakukan (*preprocessing*).

Pengubahan atau penambahan proses ini dapat dilakukan baik secara nyata terlihat oleh user (*visible*) maupun tanpa disadari oleh user (*invisible*). Pemilihan penambahan proses tambahan yang dilakukan ketika dilakukan penyimpanan data dalam suatu berkas wadah (*cover*) sebaiknya diproses ketika aplikasi berjalan (*run time*), hal ini mengingat tidak semua kasus penyimpanan informasi melalui steganografi teknik modifikasi LSB ini membutuhkan proses tambahan. Dengan kata lain, jika diketahui bahwa berkas atau informasi yang hendak disimpan memiliki ukuran yang cukup untuk disimpan dalam berkas wadah (*cover*) tanpa melalui proses tambahan maka tidak perlu dilakukan proses tambahan. Tidak dilakukannya proses tambahan tentu saja akan meningkatkan performansi dari aplikasi secara keseluruhan.

Proses tambahan yang mungkin dilakukan yang dibahas dalam makalah ini adalah proses pemampatan (*compression*) data atau informasi yang hendak disimpan serta pemrosesan berkas wadah melalui perbesaran ukuran berkas wadah tersebut (*stretching*). Pemrosesan ini dilakukan setelah dilakukan analisis terhadap ukuran informasi yang akan disimpan (ukuran data/berkas), serta ukuran dari berkas wadah tempat menyimpan informasi. Diagram alir berikut menggambarkan proses penyimpanan data tahap demi tahap.



Gambar 5 Diagram alir proses steganografi.

Diagram alir yang digambarkan pada Gambar 4 diatas menggambarkan proses steganografi yang baru, yang memanfaatkan proses pemampatan data dan perebesaran ukuran wadah, jika diperlukan, untuk menyimpan suatu informasi rahasia dalam suatu berkas wadah (*cover*) yang berupa citra.

Proses penyimpanan informasi dimulai dengan menghitung ukuran dari berkas informasi yang akan disimpan, serta dilakukan perhitungan pula kapasitas maksimum (*steganographic capacity*) dari berkas wadah yang diberikan. Setelah masing – masing keterangan tentang ukuran berkas informasi yang hendak disimpan dan daya tampung dari wadah tersebut diperoleh, langkah selanjutnya adalah membandingkan kedua variabel tersebut untuk menentukan langkah selanjutnya yang perlu dilakukan. Jika daya tampung berkas wadah memungkinkan untuk menyimpan berkas data maka dilakukan penyimpanan secara langsung dari berkas informasi tersebut, sebaliknya jika ternyata berkas yang ditampung mempunyai ukuran yang lebih besar dari daya tampung berkas penampung atau wadah, maka langkah yang dilakukan pertama kali adalah melakukan pemampatan berkas informasi yang hendak disimpan. Selanjutnya dilakukan penghitungan ulang ukuran berkas informasi rahasia yang hendak disimpan tersebut setelah dilakukan pemampatan dan selanjutnya diperiksa apakah ukurannya lebih kecil dari daya tampung yang tersedia di berkas wadah. Jika daya tampung wadah lebih besar dari ukuran berkas informasi rahasia yang hendak disimpan, maka proses penyimpanan berkas informasi rahasia tersebut dapat langsung dilakukan dengan menggunakan metode modifikasi LSB. Jika ternyata ukuran berkas informasi yang telah dimampatkan tersebut masih lebih besar dari daya tampung berkas wadah (*cover*) maka langkah yang dilakukan adalah dengan memperbesar ukuran berkas wadah (*cover*) dengan suatu faktor perbesaran n. setelah dilakukan perbesarak ukuran berkas wadah maka dilakukan proses penyimpanan informasi rahasia ke dalam berkas wadah dengan menggunakan metode modifikasi LSB untuk menghasilkan berkas stego.

Pemilihan pemrosesan terhadap berkas informasi yang hendak disimpan lebih dahulu dilakukan dari pada pemrosesan berkas wadah (*cover*) jika berkas informasi rahasia tidak dapat ditampung langsung oleh berkas wadah karena terlalu besar ukuran data sebab pemrosesan terhadap berkas informasi rahasia (pemampatan) dimungkinkan lebih efisien. Disamping itu pemilihan ini juga dilakukan dengan pertimbangan adanya batasan yang jelas dari ukuran berkas yang akan dihasilkan dari proses pemampatan. Selain itu dalam proses pemampatan juga dimungkinkan untuk dilakukan proses – proses yang lain untuk menambah tingkat keamanan data diantaranya adalah dengan menambahkan kunci rahasia untuk proses enkripsi. Proses pemapatan itu sendiri pada dasarnya juga merupakakan teknik untuk mengaburkan pesan rahasia yang hendak disimpan tersebut.

Fakor n dalam proses perbesaran berkas wadah diperoleh setelah memperhitungkan jumlah ukuran bit yang masih belum dapat tertampung. Ukuran ini diperoleh dengan menghitung selisih dari ukuran berkas yang akan di simpan dengan daya tampung dari berkas wadah.

sisa = infosz – dayatampung.
C’ = C + sisa x 8
$n = \frac{C'}{C}$
C = ukuran cover semula.
C’ = ukuran cover setelah perbesaran.
Infosz = ukuran berkas yang akan disimpan (bit).

Proses perbesaran ini hampir selalu dapat menghasilkan berkas wadah yang mampu menampung segala ukuran berkas informasi. Hal ini mengingat faktor perbesaran diperoleh dari hasil perhitungan kebutuhan dari ukuran berkas informasi yang akan ditampung.

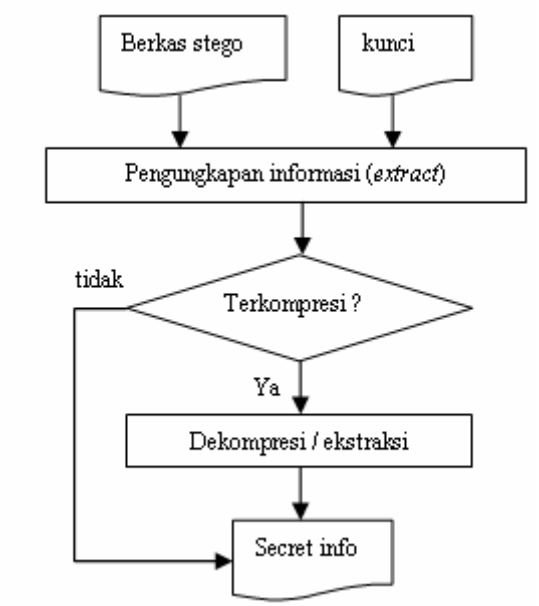
Perbesaran dengan nilai faktor n yang tidak dibatasi bukanlah pilihan yang baik. Hasil perhitungan untuk nilai n dapat menghasilkan nilai yang sangat besar jika ukuran berkas yang disimpan juga besar, semakin besar ukuran fila informasi yang akan disimpan maka semakin besar perbesaran berkas penampung yang mungkin dilakukan. Hal ini sangatlah tidak baik karena disamping proses perbesaran ukuran berkas penampung yang membutuhkan banyak sumberdaya, juga karena ukuran berkas stego yang besar dari hasil perbesaran berkas cover tersebut umumnya tidak dikehendaki oleh seluruh pihak yang berkepentingan dalam kerahasiaan informasi tersebut baik pihak pengirim informasi rahasia maupun penerima informasi. Oleh Karena itu nilai n harus dibatasi dengan suatu margin nilai tertentu.

Jika nilai faktor perbesaran (n) dibatasi dengan suatu batas atas tertentu, maka berkas – berkas informasi yang memiliki ukuran melebihi ukuran daya tampung maksimum berkas wadah dengan perbesaran maksimum yang diizinkan akan ditolak untuk dilakukan operasi penyimpanan. Hal ini lebih baik dilakukan mengingat berbagai pertimbangan yang telah diuraikan terdahulu.

Proses pengungkapan data dari berkas stego yang telah dilakukan proses steganografi dengan metode modifikasi LSB dengan preprosessing dilakukan dengan melalui dua tahapan. Tahapan – tahapan tersebut adalah pengambilan informasi melalui bit – bit tak signifikan berkas stego, proses dekompresi jika diperlukan, dan (pada aplikasi yang kami buat) dilakukan dekripsi data hasil ekstraksi.

Dari diagram alir proses pengungkapan informasi dari

berkas stego yang ditunjukkan pada Gambar 5. Pemeriksaan apakah hasil pengungkapan informasi pada proses pertama dalam keadaan termampatkan atau tidak dapat diketahui dari format berkas hasil pengungkapan pertama. Jika hasil pengungkapan pertama kali berkas yang dihasilkan tidak dalam keadaan terkompresi maka hasil pengungkapan tersebut dapat langsung ditampilkan kepada pengungkap informasi setelah dilakukan dekripsi. Jika ternyata berkas yang dihasilkan pada proses ektraksi yang pertama masih dalam keadaan terkompresi maka berkas tersebut harus diekstraksi terlebih dahulu agar dapat dibaca informasi yang terkandung di dalamnya oleh penerima pesan.



Gambar 6 Diagram Alir Proses Ekstraksi (Decoding)

7. Implementasi

Untuk melakukan pengujian dari semua teori yang telah dikemukakan pada penjeasan sebelumnya, maka dilakukan implementasi metode steganografi diatas pada sebuah program yang kami bangun dengan menggunakan kakas Microsoft Visual Studio .NET 2005 dengan bahasa pemrograman C# .NET.

Lingkungan implementasi Microsoft Visual Studio .NET ini dipilih karena alasan – alasan pragmatis dalam pengembangan aplikasi.

8. Pengujian

Dalam melakukan pengujian kami memanfaatkan beberapa berkas cover dan beberapa berkas informasi yang akan disimpan dalam cover. Berkas – berkas yang akan di gunakan tesebut adalah sebagai berikut :

No	Nama	Ukuran	Tipe
1	Agung.bmp	452x668 pixels (884 kB)	Berkas bitmap 24-bit
2	Agung1.bmp	209 x 327 pixels (200 kB)	Berkas bitmap 24-bit
3	Family.bmp	131x102 (39,4 kB)	Berkas bitmap 24-bit

Tabel 1 Berkas Wadah

No	Nama	Ukuran	Tipe
1	Al-fatihah.mp3	206.498 byte	Berkas audio
2	putty.exe	335.872 byte	Berkas binary
3	I'tibar ramadhan.txt	695 bytes	Berkas teks

Tabel 2 Berkas Data Uji

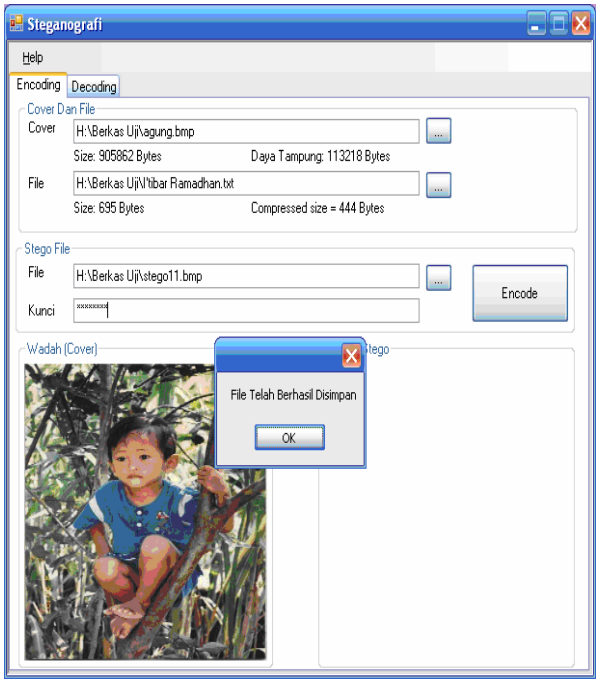
No	Berkas	Cover	embed	extract
1	1	1	2.424 det.	0.301 det.
2	1	2	1.632 det	0.210 det
3	1	3	0.140 det.	0.020 det
4	2	1	3.976 det.	0.261 det.
5	2	2	3.235 det.	0.201 det.
6	2	3	0.040 det.	0.020 det
7	3	1	4.767 det.	0.260 det
8	3	2	4.076 det.	0.230 det.
9	3	3	0.020 det.	0.020 det.

Tabel 3 Hasil Pengujian

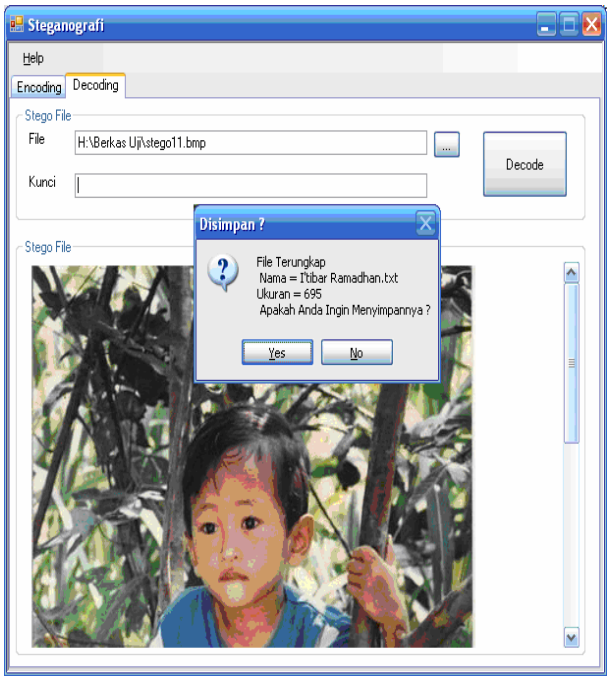
dari tabel hasil pengujian terlihat bahwa secara keseluruhan proses ekstraksi data jauh lebih cepat dari proses *embedding* data. Disamping itu proses steganografi dengan ukuran data yang lebih besar dari daya tampung maksimum wadah akan membutuhkan waktu yang relatif lebih lama (karena melalui proses tambahan) dibandingkan jika ukuran data tidak melebihi daya tampung. Misalnya pengujian no. 5 dibandingkan pengujian no. 9.

7. Kesimpulan

Steganografi dengan teknik modifikasi LSB merupakan teknik yang relatif mudah dimengerti dan mudah diimplementasikan. Dengan melakukan *preprocessing* terlebih dahulu terhadap berkas data maupun berkas wadah yang akan digunakan, teknik steganografi ini mampu "menyimpan data yang lebih besar" dari kapasitas maksimum (*steganographic capacity*) wadah yang sebenarnya.



Gambar 7 Proses Encoding



Gambar 8 Proses Decoding

Hasil pengujian yang kami lakukan menunjukkan, bahwa teknik ini hampir dapat dipastikan akan selalu berhasil untuk menyimpan data berukuran berapapun. Hal ini mengingat kemungkinan untuk memperbesar ukuran wadah secara dinamis sesuai ukuran data yang akan disimpan.

Teknik yang dibahas disini menambahkan beberapa proses selain proses modifikasi LSB sendiri (*preprocess*). Oleh karena itu dibutuhkan sumber daya yang lebih besar untuk setiap proses *encoding* (*embedding*) dan *decoding* (*extracting*) meskipun tidak terlalu signifikan. Salah satu sumber daya yang diukur dalam makalah ini adalah waktu pemrosesan, dan dari data yang diperoleh terlihat adanya perbedaan waktu pemrosesan tersebut meskipun tidak terlalu signifikan.

Daftar Pustaka

[1] Al-Mualla, Dr. Muhammed, Al – Ahmad, Prof. Husein, *Information Hiding: Steganography and Watermarking*,Etisalat College of Engineering, UAE, 2003

[2] Deutsch, Peter, RFC 1952 *GZIP berkas format specification version 4.3*, http://rfc-editor.org/

[3] Kharrazi, Mehdi, Sencar, Husrev T, Memon, Nasir, *Image Steganography: Concept and Practice*, Polytechnic University, New York, 2004

[4] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Teknik Informatika ITB, Bandung, 2005

[5] noname,*How to:Use Interpolation Mode to Control Image Quality During Scaling*, MSDN Library for Visual Studio 2005, Microsoft Corporation, 2005

[6] Rude, Thomas, Johnson, Neil F, *Introduction To Steganography: Hidden Information*, George Manson University, 2001

[7] Salomon, David, *Data Compression the complete reference 3rd ed, Springer- Verlag, New York, 2004*

[8] Thompson, Nigel, *Stretching 256-Color Images Using Interpolation*, MSDN Library for Visual Studio 2005, Microsoft Corporation, 2005