APLIKASI PENYEMBUNYIAN PESAN PADA CITRA JPEG DENGAN ALGORITMA F5 DALAM PERANGKAT MOBILE BERBASIS ANDROID

Derwin Suhartono[1], Afan Galih Salman[2], Rojali[3], Christian Octavianus[4]
[1,2,3,4] *Computer Science Department, School of Computer Science, BINUS University*
Jl. K.H. Syahdan No. 9, Palmerah, Jakarta 11480, Indonesia
Telp. (021) 5345830 ext. 2259, Faks. (021) 5300244
E-mail: dsuhartono@binus.edu

ABSTRAK
Perkembangan dunia saat ini tidak lepas dari perkembangan teknologi komunikasi yang semakin berkembang dengan pesat. Karena perkembangan ini, maka dibutuhkan kemampuan untuk dapat mengakses informasi dengan cepat. Perkembangan ini sangat terlihat khususnya pada media elektronik, dimana salah satu faktor penting yang sangat berperan didalamnya adalah internet. Dengan internet, manusia dapat dengan mudah bertukar informasi dengan menggunakan media elektronik, seperti PC (Personal Computer) maupun dengan perangkat mobile, seperti handphone, maupun tablet PC. Karena semakin banyaknya orang yang menggunakan media internet, maka kebutuhan akan keamanan dalam berkomunikasi semakin diperlukan. Karena hal inilah diperlukan adanya cara untuk dapat mengirimkan data dengan aman. Salah satu caranya adalah dengan menyembunyikan data sebelum data tersebut dikirimkan. Pada penelitian ini akan dirancang suatu model penyembunyian pesan atau steganografi pada image berbasis android platform. Algoritma F5 merupakan salah satu dari algoritma yang digunakan dalam keperluan steganografi pada citra JPEG. Aplikasi steganografi ini dibangun dengan menggunakan bahasa pemrograman Java Android. Berdasarkan hasil perancangan dan implementasi, didapat hasil citra yang tidak jauh berbeda dengan citra aslinya, sehingga keamanan data yang dikirimkan dengan menggunakan program aplikasi terjamin. Berdasarkan hasil pengecekan error dengan menggunakan PSNR didapat hasil yang baik untuk setiap citra steganogram, yaitu lebih dari 70 dB.

Kata Kunci: Steganografi, Citra JPEG, Aplikasi Mobile, Algoritma F5

1. PENDAHULUAN

Perkembangan teknologi sekarang ini memungkinkan orang untuk dapat melakukan komunikasi maupun pertukaran data secara mudah. Karena itu tentunya keamanan data sangatlah penting, terutama dalam bisnis komersil maupun tradisional. Sebagai contoh, dalam pengiriman data-data krusial perusahaan, dimana diperlukan adanya pengamanan pada data, agar hanya beberapa orang yang dapat mengakses atau mendapatkan data tersebut.

Salah satu cara untuk mengamankan data yang akan dikirimkan seperti pada contoh diatas adalah dengan menggunakan algoritma steganografi yang sudah banyak berkembang untuk menyembunyikan data pada suatu media. Steganografi merupakan teknik yang mempelajari penyembunyian data di dalam data induk sehingga keberadaan data tidak bisa atau sulit untuk diketahui. Semua *file* umum yang sudah ada, secara teori, dapat digunakan sebagai media pembawa, seperti *file* gambar yang berformat JPEG, BMP, GIF, atau dalam *file* musik berformat MP3, dan bahkan dalam *file* video yang berformat AVI.

Proses steganografi sendiri secara garis besar akan dimulai dengan penyisipan data ke dalam media pembawa, yang dalam hal ini adalah *file image* dengan format JPEG (*Joint Photographic Experts Group*), proses penyisipan data dilakukan secara algoritmik berdasarkan dari kata kunci yang sudah ditentukan sebelumnya. Untuk dapat melihat data yang ada di dalam file *image* tersebut, penerima file tersebut harus memasukkan kata kunci yang sama dengan kata kunci pada saat data disisipkan. Jika kata kunci yang dimasukkan berbeda, maka data yang didapat akan berbeda dengan data yang sebenarnya yang dikirimkan.

Media *file image* yang berformat JPEG ini juga memiliki beberapa keuntungan untuk dijadikan media pembawa data (*steganogram*). Pertama adalah karena tipe *file* ini merupakan tipe *file* yang sudah umum digunakan, dan sudah banyak digunakan untuk pertukaran gambar pada *internet*. Alasan kedua adalah karena tipe *file* JPEG ini juga memiliki banyak algoritma penyisipan *file* atau data yang memiliki tingkat kesulitan untuk dilacak yang tinggi.

Metode yang digunakan pada aplikasi ini adalah metode F5, yang merupakan pengembangan dari metode steganografi sebelumnya yaitu metode F3 dan F4. Algoritma F5 ini menyisipkan bit data pesan kedalam bit koefisien DCT kemudian membuat matriks *encoding* untuk mengurangi atau meminimalkan jumlah perubahan-perubahan yang diperlukan untuk menyisipkan suatu pesan dengan panjang tertentu (Fridrich, 2002). Metode F5 ini mempunyai tingkat efisiensi enkripsi yang lebih baik dibanding metode steganografi pendahulunya, karena menggunakan permutasi sehingga

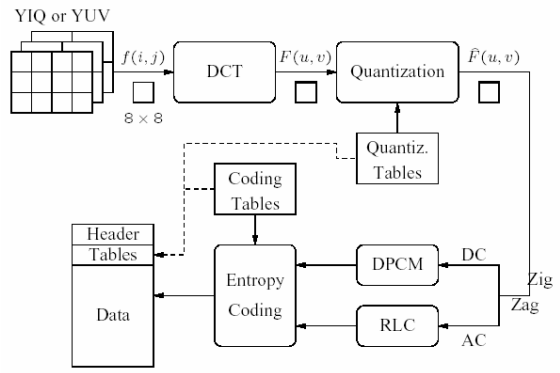
penyebaran pesan lebih seragam. Implementasi dari metode ini diharapkan dapat memenuhi kebutuhan akan tingkat keamanan data perangkat *mobile*.

2. LANDASAN TEORI

Kata steganografi (*Steganography*) berasal dari kata Yunani. *Steganos* yang artinya 'tersembunyi/terselubung', dan *graphien*, 'menulis' sehingga kurang lebih artinya "menulis (tulisan) terselubung". (Budi, 2002).

Steganografi adalah suatu teknik untuk menyembunyikan informasi yang bersifat pribadi dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya. Media yang digunakan umumnya merupakan suatu media yang berbeda dengan media pembawa informasi rahasia, dimana disinilah fungsi dari teknik steganografi yaitu sebagai teknik penyamaran menggunakan media lain yang berbeda sehingga informasi rahasia dalam media awal tidak terlihat secara jelas. Steganografi juga berbeda dengan kriptografi yaitu terletak pada hasil dari prosesnya. Hasil dari kriptografi biasanya berupa data yang berbeda dari bentuk aslinya dan biasanya datanya seolah-olah berantakan namun dapat dikembalikan ke data semula. Sedangkan hasil dari keluaran steganografi memiliki bentuk yang sama dengan data aslinya, tentu saja persepsi ini oleh indra manusia, tetapi tidak oleh komputer atau pengolah data digital lainnya.

Dimulai sejak beberapa tahun lalu, JPEG (*Joint Photographic Experts Group*) membuat teknik kompresi international pertama untuk format *file* citra. Pada tahun 1992 teknik kompresi ini mulai diterima secara formal sebagai standar internasional. (Leung, 2005). Standar ini ditetapkan oleh JPEG agar dapat memenuhi kebutuhan berbagai aplikasi yang bekerja dengan *file image*. Kompresi yang diajukan oleh JPEG ini dapat bekerja dengan citra berwarna maupun *greyscale*. Berikut akan dijelaskan secara lebih lanjut untuk tahapan pada kompresi JPEG:



Gambar 1. Tahapan dalam kompresi JPEG (Leung, 2005)

3. METODOLOGI

Tahap pertama dari kompresi JPEG adalah konversi dari RGB ke YcbCr. Dengan basis RGB, kita bisa mengubah warna ke dalam kode-kode angka sehingga warna tersebut akan tampil universal.

Karena mata manusia lebih sensitif pada warna *luminance* (Y) dari pada warna *chrominance* (Cb,Cr), sehingga informasi warna *chrominance* tidak diikutsertakan pada proses kompresi dan hanya warna Y yang diproses sebagai masukan gambar untuk proses selanjutnya. Warna YCbCr diperoleh dengan mentransformasikan RGB dengan rumus (Gunawan, 2003):

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.159 & -0.332 & 0.050 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Tahap kedua dari kompresi JPEG adalah tahap DCT (*Discrete Cosine Transform*). Hal yang pertama kali dilakukan pada tahap DCT ini adalah membagi keseluruhan gambar menjadi 8 x 8 *pixel*. Kemudian setiap blok-blok *pixel* tersebut diproses satu persatu menjadi 64 keofisien DCT melalui rumus :

$$F(u,v) = \frac{C(u)C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)u\pi}{16} f(i,j) \quad C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & jika\ \xi = 0 \\ 1 & lainnya \end{cases}$$

Tujuan dari tahap ini adalah karena pada gambar yang belum terkompresi nilai koefisien DCT rata-rata berukuran amat kecil dan banyak yang dapat dihilangkan dengan tetap mempertahankan keakuratan gambar.

Dibanding nilai 63 koefisien DCT lainnya, koefisien pertama dari tiap blok pasti memiliki nilai yang paling besar karena merupakan nilai rata-rata dari keseluruhan blok, koefisien pertama disebut koefisien DC dan 63 koefisien lainnya disebut koefisien AC. Untuk mengembalikan kembali koefisien DCT yang didapat kedalam 64 nilai *pixel* sebelumnya harus dilakukan tahap Invers DCT, tetapi hasil yang didapat akan sedikit mengalami perubahan sehingga tahap ini dinamakan tahap *lossy*. Rumus Invers DCT adalah sebagai berikut :

$$\tilde{f}(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)C(v)}{4} \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} F(u,v), \qquad C(\xi) \begin{cases} \frac{\sqrt{2}}{2} & jika\ \zeta = 0 \\ 1 & lainnya \end{cases}$$

Tahap selanjutnya adalah tahap kuantisasi dari koefisien-koefisien DCT yang didapat sebelumnya. Proses kuantisasi merupakan proses untuk mengurangi jumlah bit yang diperlukan untuk menyimpan suatu data gambar. Karena mata manusia lebih peka terhadap frekuensi rendah dari pada frekuensi tinggi dan karena frekuensi tinggi tidak merubah data gambar secara signifikan, maka pada proses kuantisasi frekuensi tinggi ini dipotong dengan cara, matriks koefisien hasil DCT dibagi

dengan matriks *quantum*. Matriks quantum ini ditentukan oleh faktor kualitas yang dipilih antara 1 sampai 100 yang nantinya dipakai untuk menentukan kualitas dari suatu gambar JPEG.

Tahap kuantisasi juga merupakan tahap *lossy* dalam kompresi JPEG karena kuantisasi melakukan pembagian antara setiap koefisien DCT dengan koefisien dari matriks *quantum* yang ditentukan dan melakukan pembulatan setelahnya.

Berikut adalah tabel kuantisasi untuk koefisien *luminance* dan *chrominance* yang sudah ditetapkan sebagai standar oleh JPEG dengan rasio kompresi paling baik dan penurunan kualitas gambar paling rendah :

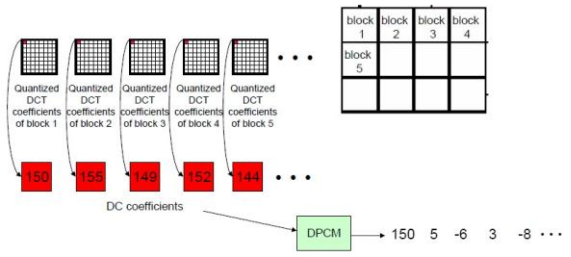
Tabel 1. Kuantisasi *Luminence* dan Tabel Kuantisasi *Chrominence*

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Keragaman warna pada suatu blok ternyata juga sangat berpengaruh pada penurunan kualitas yang disebabkan oleh dua tahap *lossy* ini (DCT dan Kuantisasi). Kesalahan yang terjadi pada blok dengan tekstur warna yang beragam lebih besar dibanding dengan blok yang mempunyai tekstur warna relatif sama.

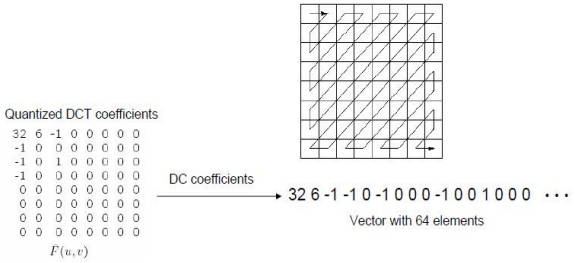
Tahapan selanjutnya dari kompresi JPEG adalah DCPM (*Differential Pulse Code Modulation*). Pada tahap ini, koefisien DC dari tiap blok disatukan untuk memasuki tahap *Entropy Coding*, teknik DPCM digunakan karena nilai-nilai koefisien DC antar blok tidak berbeda jauh. (Leung, 2005).



Gambar 2. *Differential Pulse Code Modulation*

Koefisien DC yang sudah melalui tahap DPCM kemudian dikompresi menggunakan metode *Huffman*, tetapi sebelumnya deretan angka tersebut akan dirubah bentuknya menjadi pasangan-pasangan (*size, amplitude*) dimana *size* menyatakan jumlah bit yang diperlukan untuk merepresentasikan jumlah angka DPCM dan *amplitude* menyatakan angka tersebut dalam bit. Dalam *Entropy coding* yang mengalami kompresi *huffman* hanya *size*-nya saja, karena perubahan *size* tidak terlalu jauh sedangkan *amplitude*-nya bervariasi.

Tahapan selanjutnya adalah RLC (*Run Length Coding*). *Run Length Coding* memakai metode *zig-zag scannin*, yaitu proses yang merubah matriks 8 x 8 hasil proses kuantisasi ke dalam vektor 1 x 2^8 , dengan pembacaan secara *zig-zag scanning*. Pada proses *zig-zag scanning* ini koefisien DCT terkuantisasi yang bernilai nol cenderung terbaca secara berurutan.



Gambar 3. Proses *Zig-Zag Scan*

RLC (*Run-Length Code*) yaitu proses serangkaian simbol yang berurutan dikodekan menjadi suatu kode yang terdiri dari simbol tersebut dan jumlah pengulangannya. RLC efektif karena hasil keluaran matriks setelah proses kuantisasi pada frekuensi tinggi cenderung nol (0) dan berurutan, Karena hampir setengahnya lebih adalah nol, maka nilai 0 inilah yang disimbolkan menjadi 0 dan jumlah pengulangannya. Untuk proses dekompresi, dilakukan proses sebaliknya yaitu hasil pengkodean RLC di-*scan* dan diuraikan kembali, kemudian kode hasil penguraian dibaca sebagai blok. Berikut ini adalah contoh perubahan proses setelah proses *zig-zag scanning* ke dalam proses RLC.

Setelah diubah urutannya, niai AC kemudian diubah bentuknya menjadi pasangan-pasangan (*runlength,value*), dimana *runlength* adalah jumlah 0 yang berurutan dan *value* adalah nilai non – 0 yang terletak sesudahnya. Dalam hal ini koefisien DC tidak diperhitungkan dalam RLC.

Koefisien AC yang sudah melalui tahap RLC juga dikompresi menggunakan kompresi *huffman*, pasangan-pasangan sebelumnya diubah lagi menjadi pasangan-pasangan (*runlength, size, value*). Dalam hal ini yang mengalami kompresi *huffman* hanya *runlength* dan *size*-nya seperti pada koefisien DC.

Algoritma F5 bekerja setelah tahap kuantisasi dari kompresi JPEG. Algoritma steganografi F5 diperkenalkan oleh Pfitzmann peneliti Jerman dan Westfeld pada tahun 2001. Tujuan dari mereka penelitian ini adalah untuk mengembangkan konsep dan metode *embedding* praktis untuk gambar JPEG yang akan memberikan steganografi tinggi kapasitas tanpa mengorbankan keamanan. Daripada mengganti LSB dari koefisien DCT terkuantisasi dengan bit pesan, nilai absolut dari koefisien menurun satu. Para penulis berpendapat bahwa jenis *embedding* tidak dapat dideteksi menggunakan serangan statistik $\chi 2$. Algoritma F5 meng-*embed* bit pesan ke koefisien DCT yang dipilih secara acak dan

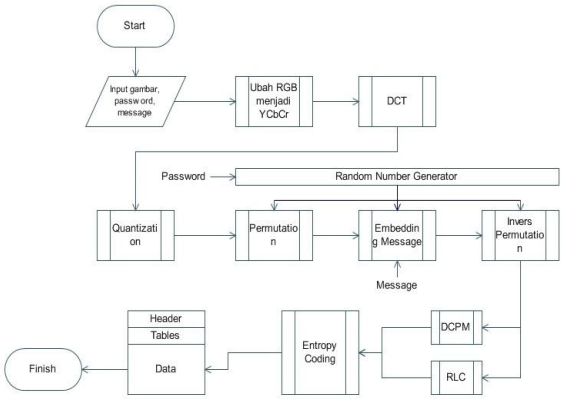
menggunakan matriks *embedding* yang meminimalkan jumlah perubahan yang perlu untuk menanamkan panjang pesan tertentu.

Proses *embedding* dimulai dengan menurunkan benih untuk PRNG (*Pseudo Random Number Generator*) dari kata sandi pengguna dan menghasilkan "*random walk*" koefisien DCT dari *cover image* tersebut. PRNG juga digunakan untuk mengenkripsi nilai *k* menggunakan *stream cipher* dan menanamkannya dalam cara yang teratur bersama-sama dengan panjang pesan di awal aliran pesan. Tubuh pesan tertanam menggunakan *embedding* matriks, menyisipkan *k* bit pesan ke satu kelompok *2k-1* koefisien dengan menurunkan nilai absolut paling banyak satu koefisien dari masing-masing kelompok satu. Proses *embedding* terdiri dari langkah-langkah berikut:

a. Ambil nilai RGB dari gambar input
b. Hitung tabel kuantisasi yang sesuai dengan faktor kualitas Q dan kompres gambar saat menyimpan DCT terkuantisasi koefisien.
c. Hitung perkiraan kapasitas tanpa *embedding* matriks $C = h_{DCT} - h_{DCT} / 64 - h(0) - h(1) + 0.49h(1)$, di mana h_{DCT} adalah jumlah semua koefisien DCT, *h (0)* adalah jumlah koefisien DCT AC bernilai nol, *h (1)* adalah jumlah dari AC Koefisien DCT dengan nilai absolut 1, $h_{DCT}/64$ adalah jumlah dari DC koefisien. Parameter *C* dan panjang pesan yang digunakan untuk menentukan matriks *embedding* terbaik.
d. Password yang ditentukan pengguna digunakan untuk menghasilkan benih untuk PRNG juga digunakan menentukan jalur acak untuk *embedding* bit-bit pesan. PRNG juga digunakan untuk menghasilkan *pseudo-random bit-stream* yang diXOR dengan pesan untuk membuatnya *bit-stream* teracak. Selama *embedding*, koefisien DC dan koefisien sama dengan nol dilewati.
e. Pesan dibagi menjadi segmen-segmen dari *k* bit yang tertanam ke dalam kelompok 2^k-1 koefisien sepanjang jalur acak. Jika hash dari kelompok yang tidak cocok dengan bit-bit pesan, nilai absolut dari salah satu koefisien dalam kelompok diturunkan satu untuk mendapatkan nilai yang cocok. Jika koefisien menjadi nol, kejadian ini disebut sebagai penyusutan, dan *k* bit pesan yang sama di*embed* ulang dalam kelompok berikutnya dari koefisien DCT.
f. Jika ukuran pesan sesuai dengan perkiraan kapasitas, maka proses *embed* berlanjut, lain daripada itu *error* yang menunjukkan panjang maksimal yang mungkin akan ditampilkan.

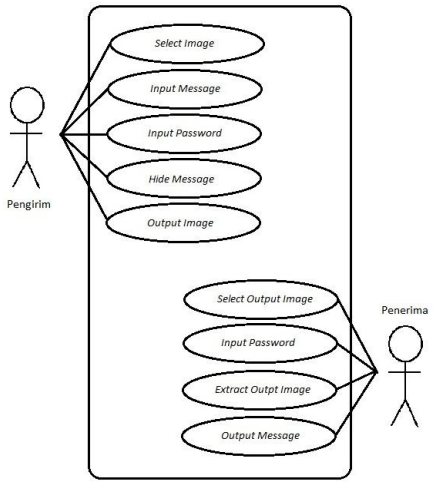
Algoritma F5 ini tidak memodifikasi histogram koefisien DCT, Algoritma ini menunjukkan bahwa beberapa karakteristik penting dari histogram tetap dipertahankan, seperti yang kemonotonan dan kemonotonan dari kenaikan. Algoritma F5 tidak dapat dideteksi dengan menggunakan serangan $\chi 2$ karena *embedding* tidak didasarkan pada penggantian bit maupun pertukaran nilai tetap apapun.

Secara garis besar proses berjalannya algoritma steganografi F5, mulai dari tahapan kompresi JPEG pertama sampai akhir, dapat dilihat pada gambar di bawah berikut:



Gambar 4. *Flow Chart* Algoritma

Perancangan program aplikasi pada penelitian ini menggunakan metode *Linear Sequential (Waterfall)*. Metode *Waterfall* ini memiliki lima tahapan yaitu, *requirement analysis*, *system design*, *implementation*, *integration*, dan *maintenance*.



Gambar 5. *Use Case Diagram*

Pada *use case diagram*, pengirim atau *user* dapat menentukan gambar mana yang akan disisipi oleh pesan atau *message*. Kemudian pengirim dapat menuliskan pesan yang akan disembunyikan kedalam gambar yang telah dipilih sebelumnya. Pengirim kemudian harus menentukan kata kunci atau *password* yang akan digunakan. Dan yang terakhir pengirim dapat menekan tombol *insert message* untuk memulai proses penyisipan pesan kedalam gambar. Hasil atau *ouput* dari proses ini

berupa gambar yang telah disisipi oleh pesan (*stego image*).

Dari sisi *user* penerima, yang dapat dilakukan adalah, memilih gambar yang telah disisipi pesan, kemudian memasukkan *password* yang sama dengan *password* pada saat pesan disisipkan ke dalam gambar. Kemudian dapat melakukan proses ekstraksi pesan yang ada didalam gambar, dan kemudian mendapati hasil dari proses ekstraksi yang berupa pesan yang disisipkan ke dalam gambar.

4. SISTEM DAN IMPLEMENTASI

Dalam perancangan program digunakan komputer dengan spesifikasi sebagai berikut:

a. *Processor:* 2*nd generation Intel Core i5 – 2430M CPU @ 2.4GHz*
b. *Memory*: *4GB (2,8 usable)*
c. Sistem Operasi: *Windows 7 Home Premium 64-bit (6.1, build 7601)*

Sedangkan untuk kebutuhan perangkat lunak (*software*), digunakan beberapa perangkat lunak pendukung. Perangkat lunak tersebut antara lain :

a. *Platform: Java SE version 1.6, Android SDK, Eclipse 4.0, ADT version 16*
b. Bahasa Pemrograman: *Android*

Dalam menjalankan aplikasi, *user* dapat melakukan dua proses, yaitu proses memasukkan pesan dan proses pengeluaran pesan. Pada proses pemasukkan pesan, *user* memasukkan pesan rahasia berupa teks atau tulisan ke dalam *image* atau gambar yang telah dipilih sebelumnya dan kemudian menentukan kata kunci atau *password* dari gambar. Untuk proses pengeluaran pesan, *user* pertama kali memilih gambar yang sebelumnya telah disisipi pesan dan kemudian memasukkan kata kunci atau *password* yang sesuai dengan gambar yang telah dipilih tersebut, setelahnya akan muncul pesan yang tersisip atau tersembunyi didalam gambar.

5. ANALISA DAN PEMBAHASAN

Berikut adalah beberapa hasil dari steganografi dengan menggunakan program aplikasi.

Tabel 2. Perbandingan Gambar

No	Gambar Awal	Gambar Hasil	Nama File
1			Supporter.jpg
2			Camp_Nou.jpg
3			Ball.jpg
4			Anfield.jpg

Citra atau gambar hasil dari proses steganografi dengan menggunakan aplikasi ini tidak begitu terlihat perbedaannya secara kasat mata.

Karena itulah untuk mengetahui tingkat kesalahan atau *error* dari gambar asli dengan gambar hasil steganografi dilakukan analisis PSNR (*Peak Signal to Noise Ratio*). *Peak Signal to Noise Ratio* adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut.Untuk menentukan *PSNR*, terlebih dahulu harus ditentukan nilai rata-rata kuadrat dari error (*MSE - Mean Square Error*).

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

Dan PSNR sendiri didefinisikan sebagai :

$$PSNR = 10.\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20.\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

Dimana, MAX_I = nilai maksimum piksel

Tabel 3. Tabel Pengukuran PSNR

No	Nama File	Ukuran Piksel	Ukuran Sebelum	Ukuran Sesudah	Nilai PSNR
1	Supporter.jpg	300 x 200 px	27,2 KB	21,3 KB	63,03195
2	Camp_Nou.jpg	300 x 200 px	23,3 KB	12,7 KB	72,32379
3	Ball.jpg	314 x 235 px	34,7 KB	12,2 KB	72,53886
4	Anfield.jpg	314 x 208 px	19,1 KB	17,0 KB	71,83927
5	Celtic_match.jpg	314 x 207 px	16,7 KB	12,7 KB	71,37714

Hasil perhitungan PSNR yang didapat berkisar antara 63 dB sampai dengan 72dB. Kualitas gambar yang dihasilkan cukup baik, karena standar nilai PSNR yang baik adalah diatas 30dB – 40dB. Semakin tinggi nilai PSNR yang didapat maka kualitas gambar stego yang dihasilkan semakin menyerupai gambar aslinya.

6. PENUTUP

Algoritma Steganografi F5 dapat dimplementasikan pada *mobile device* berbasis Android untuk menyisipkan pesan kedalam media gambar dengan *format* tipe data JPEG tanpa mengubah isi pesan. Algoritma Steganografi F5 ini juga dapat meningkatkan keamanan atas pencurian data dari pihak luar yang tidak berkepentingan. Hasil dari pengujian dengan menggunakan *Peak Signal to Noise Ratio* atau PSNR, ternyata gambar yang dihasilkan memiliki kualitas yang tidak jauh berbeda dengan gambar sebelum disisipi pesan. Dengan kisaran rata-rata nilai PSNR lebih dari 70 dB.

Dalam proses steganografi dengan metode F5 ini masih terdapat beberapa aspek yang dapat dikembangkan lebih lanjut. Beberapa diantaranya yaitu :

1. Pengembangan lebih lanjut untuk memakai tipe data yang lain selain tipe data JPEG.
2. Proses penyebaran data dalam *image* dapat lebih ditingkatkan.
3. Program aplikasi dapat dikembangkan lebih lanjut agar dapat langsung melakukan pengiriman pesan yang telah tersisipi, tanpa harus menutup program aplikasi terlebih dahulu.
4. Mengembangkan perangkat keras (*hardware*) yang lebih mendukung kecepatan proses steganografi pada perangkat *mobile*.

PUSTAKA

Fridrich, Jessica, Miroslav Goljan, Dorin Hogea. (2002). *Steganalysis of JPEG Images: Breaking the F5 Algorithm.* Diakses pada 22 Desember 2011 dari ws2.binghamton.edu/fridrich/Research/f5.pdf

Rahardjo, Budi. (2005). *Keamanan Sistem Informasi*. Diakses pada 28 Desember 2011 dari http://budi.insan.co.id/books/handbook.pdf

Leung. (2005). *Image Compression Standards*. Diakses pada 26 Desember 2011 dari http://www2.it.lut.fi/kurssit/06-07/Ti5312400/Materiaali/Paiva_2/2-2-3.pdf

Wibisono, Gunawan. (2003). *Implementasi Kompresi Gambar dengan Format JPEG*. Diakses pada 19 Desember 2011 dari http://elib.unikom.ac.id/download.php?id=257