Aplikasi Steganography pada File dengan Menggunakan Teknik *Low Bit Encoding* dan *Least Significant Bit*

Hendra Bunyamin, Andrian
Jurusan Teknik Informatika
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri no. 65 Bandung 40164
Email: hendra.bunyamin@itmaranatha.org, nextarone@yahoo.com

Abstract

With the more popular multimedia world, security are the main concern at this time. Data theft occurred in popularity virtual world (internet). One important issue is the level of information security. This can be done by using encryption or steganography.

Steganography is a method to insert a snippet of information in a confidential other multimedia object. In the data hiding, steganography or data hiding embedding the data seems very familiar with the encryption. However, data hiding by changing the order of characters in a multimedia same. While in steganography, data hiding is done in a way to change or shift some of the information that is not visible in the media of important message.

The proposed methods in this paper are Least Significant Bit (LSB) and Low Bit Encoding (LBE). These methods will be applied to the message which is inserted into multimedia images (bmp, jpeg), audio (wav, mpeg), and other files (exe, pdf). Share media with the image data entered in the frame, this technique is expected to be able to insert information in a single frame maximum of 1 bit, so changes are not visible light. Method end of the project is proving a secret message hiding techniques in multimedia. Result output file generated by this scientific report changes in the quality of which is the lower of the original file.

In the comparison of image files, image files which have experienced steganography will be look brighter if due to changes in contrast. While the comparison of media files, media files which have experienced will have been damaged in audio, there is little noise in the audio. Then the last is a pdf or exe files, if the file has steganography, the file size will be changed in size due to a significant size of the file is added to the original size of the message to be hidden.

Keywords *: encryption, data hiding, steganography*

1. Latar Belakang

Saat ini Internet sudah berkembang menjadi salah satu media yang sangat populer di dunia. Namun dengan semakin berkembangnya Internet dan aplikasi menggunakan Internet, semakin berkembang pula kejahatan dalam penyalahgunaan informasi di suatu sistem informasi. Dengan menggunakan berbagai teknik, banyak orang yang mencoba untuk mengakses informasi yang bukan haknya. Oleh karena itu, keamanan suatu sistem informasi juga mesti diutamakan seiring dengan berkembangnya aplikasi menggunakan Internet.

Berbagai macam teknik dapat digunakan untuk melindungi kerahasiaan informasi dari orang yang tidak berhak. Salah satu teknik yang populer adalah teknik *steganography*. Teknik ini sudah dipakai lebih dari 2500 tahun yang lalu untuk menyembunyikan pesan rahasia. Berbeda dengan teknik *cryptography*, *steganography* digunakan untuk menyembunyikan pesan rahasia dan orang awam tidak menyadari keberadaan bahwa terdapat pesan yang disembunyikan. Teknik ini sering digunakan untuk menghindari kecurigaan orang dan menghindari keinginan orang untuk mengetahui isi pesan rahasia tersebut.

Pada umumnya, teknik *steganography* dikenakan pada *file-file* multimedia. Bentuk-bentuk *file* multimedia yang sering digunakan adalah video, *audio* dan gambar.

2. Tujuan

Tujuan dari pembuatan artikel ini adalah :

1. Memberikan pandangan bahwa *steganography* memiliki tingkat keamanan yang cukup tinggi.
2. Menerapkan teknik *steganography* dalam *file audio*, video, gambar dan *file* lainnya seperti *exe* dan *pdf*.
3. Memaparkan cara kerja dari aplikasi *steganography*, serta menunjukkan sebagaimana kuat ketahanan dari *file* hasil proses *steganography*.

3. Pembatasan Masalah Aplikasi

Aplikasi yang akan dibuat mempunyai batasan–batasan sebagai berikut:

1. Objek penelitian difokuskan pada kualitas dari *file* dan besar ukuran *file* yang telah disisipkan. Dengan kata lain apabila kualitas gambar pada gambar asli adalah 100% dengan ukuran *file* 65 kb dan *file* pesan yang akan disisipkan adalah 22,7 kb maka hasil *file* yang telah disisipkan mengalami kemunduran kualitas kurang lebih menjadi 88.5% dengan ukuran *file* akan menjadi besar.

2. Kecepatan dalam pemrosesan *encoding* dan *decoding* belum menjadi pokok penelitian pada *paper* ini.

4. Mekanisme JNI (*Java Native Interface*)

Dalam *framework* JNI, fungsi native diimplementasikan secara terpisah .C atau .CPP *file*. (C++ menyediakan antarmuka yang sedikit baik dengan JNI). Berikut adalah contoh JNI:

```
JNIEXPORT void JNICALL Java_ClassName_MethodName
   (JNIEnv *env, jobject obj)
 {
     /*Implement Native Method Here*/
 }
```

Env pointer merupakan struktur yang berisi antarmuka ke JVM. Termasuk semua fungsi yang diperlukan untuk berinteraksi dengan JVM dan untuk bekerja dengan objek-objek JAVA. Contoh fungsi-fungsi JNI mengkonversi *native array* ke JAVA *array*, mengkonversi *native string* ke *java string*, *instantiating objects, throwing exceptions*, etc. Pada dasarnya, apapun yang dapat dilakukan oleh kode JAVA dapat dilakukan dengan menggunakan JNIEnv, walaupun tidak mudah. Berikut adalah contoh mengkonversi JAVA *string* ke *native string*:

```
//C++ code
 JNIEXPORT void JNICALL Java_ClassName_MethodName
   (JNIEnv *env, jobject obj, jstring javaString)
 {
     //Get the native string from javaString
     const char *nativeString =
               env->GetStringUTFChars(javaString, 0);

     //Do something with the nativeString
      env->ReleaseStringUTFChars(javaString, nativeString);
 }
```

```
/*C code*/
 JNIEXPORT void JNICALL Java_ClassName_MethodName
   (JNIEnv *env, jobject obj, jstring javaString)
 {
     /*Get the native string from javaString*/
     const char *nativeString =
                 (*env)->GetStringUTFChars(env, javaString, 0);

     /*Do something with the nativeString*/
     (*env)->ReleaseStringUTFChars(env, javaString, nativeString);
 }
```

Perlu diketahui bahwa C++ JNI adalah sintak yang lebih baik daripada C JNI disebabkan lebih cocok pada pemrograman JAVA, C++ menggunakan *object method invocation semantics*. Dalam C++, yang merupakan *parameter env* dan *parameter* adalah mutlak sebagai bagian dari *metode obyek invocation semantics*. Jenis data *native* dapat dipetakan dari jenis data JAVA. Untuk jenis kompleks seperti *objek array* dan *string native* harus secara *eksplisit* mengkonversi data dengan memanggil *methods* pada JNIEnv.

Untuk menjalankan mekanisme JNI (JAVA *Native Interface*), terdapat beberapa hal yang harus dilakukan terlebih dahulu. Pada saat instalasi JDK, Netbeans IDE dan Netbeans C/C++ *pack* harus dimasukkan. Selain itu dibutuhkan juga Cygwin

sebagai *complier* dari C/C++ tersebut. Setelah perangkat lunak tersebut di-install, diperlukan sebagian kecil perubahan pada konfigurasi sistem harus dilakukan.

5. Pengujian Aplikasi

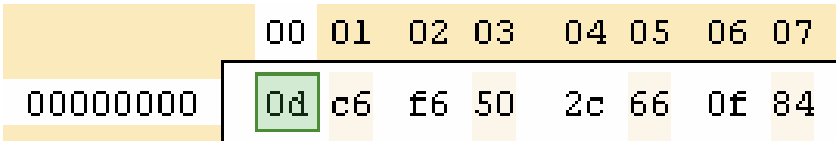
Tabel 1 menunjukkan perbandingan dan perhitungan gambar dengan format bmp dengan menggunakan perhitungan LSB.

Tabel 1 Perbandingan Gambar

Ukuran *File* (UF)	Pixel (P)	Ukuran Pixel (UP) P * P	Total Ukuran Byte (TUB) UP * 3	Maksimum Ukuran *File* Byte (MUFB) TUB / 8	Ukuran *File* Pesan Byte (UFPB)	Kualitas (K)
192 kb	256	65596	196608	24576	24576	87,5%
					19660	90,0%
					8848	95,5%
Ukuran *File* (UF)	Pixel (P)	Ukuran Pixel (UP)	Total Ukuran Byte (TUB)	Maksimum Ukuran *File* Byte (MUFB)	Ukuran *File* Pesan Byte (UFPB)	Kualitas (K)
258 kb	512	262144	786432	98304	98304	87,5%
					78643	90,0%
					35389	95,5%
2305 kb	1024	1048576	3145728	393216	393216	87,5%
					314573	90,0%
					141558	95,5%

Dari Tabel 1 dapat disimpulkan bahwa proses terbaik adalah dengan menggunakan kualitas 95,5% sedangkan paling buruk adalah 87,5%. Untuk kualitas di bawah 87,5%, aplikasi tidak mengijinkan pengguna melakukan proses. Hal ini disebabkan oleh ukuran *file* pesan *byte* telah melebihi maksimum ukuran *file byte*.

Pada aplikasi ini, data pesan rahasia yang akan disembunyikan pada *file audio* akan dienkripsi terlebih dahulu, sehingga keamanannya menjadi lebih terjamin. Dengan adanya proses enkripsi, data pesan rahasia pun akan berubah, sehingga nilai-nilai *bit* yang akan disembunyikan akan berubah juga. Gambar 1 merupakan *hexadecimal* dari nilai A yang telah mengalami enkripsi.

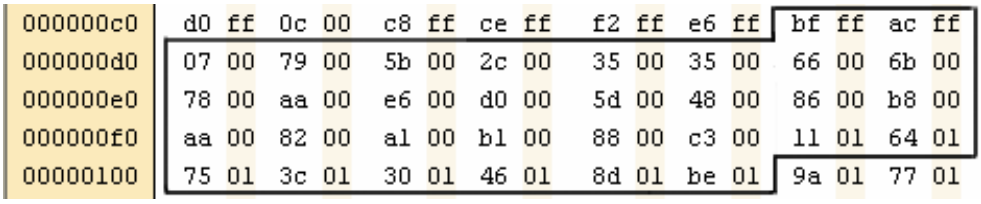


	00	01	02	03	04	05	06	07
00000000	0d	c6	f6	50	2c	66	0f	84

Gambar 1 *Hexadecimal* Nilai A

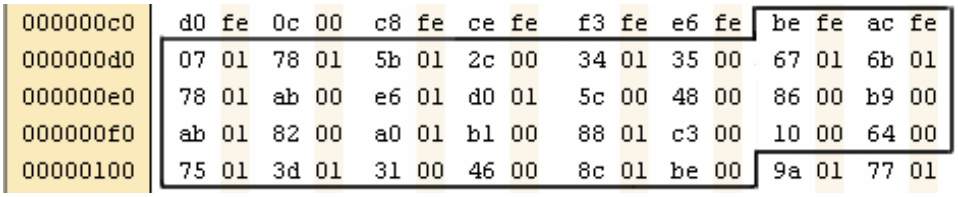
Pada Gambar 1, nilai A yang telah mengalami enkripsi memiliki 8 *hexadecimal*, setiap *hexadecimal* memiliki 8 bit. Maka terdapat 64 bit yang akan disembunyikan pada 64 *hexadecimal* file *audio*.

Perbandingan antara file *audio* sebelum disembunyikan nilai A ditunjukkan pada Gambar 2 dan file *audio* setelah disembunyikan nilai A ditunjukkan oleh Gambar 3



000000c0	d0	ff	0c	00	c8	ff	ce	ff	f2	ff	e6	ff	bf	ff	ac	ff
000000d0	07	00	79	00	5b	00	2c	00	35	00	35	00	66	00	6b	00
000000e0	78	00	aa	00	e6	00	d0	00	5d	00	48	00	86	00	b8	00
000000f0	aa	00	82	00	a1	00	b1	00	88	00	c3	00	11	01	64	01
00000100	75	01	3c	01	30	01	46	01	8d	01	be	01	9a	01	77	01

Gambar 2 *File Audio* Sebelum Disembunyikan Nilai A



000000c0	d0	fe	0c	00	c8	fe	ce	fe	f3	fe	e6	fe	be	fe	ac	fe
000000d0	07	01	78	01	5b	01	2c	00	34	01	35	00	67	01	6b	01
000000e0	78	01	ab	00	e6	01	d0	01	5c	00	48	00	86	00	b9	00
000000f0	ab	01	82	00	a0	01	b1	00	88	01	c3	00	10	00	64	00
00000100	75	01	3d	01	31	00	46	00	8c	01	be	00	9a	01	77	01

Gambar 3 *File Audio* Setelah Disembunyikan Nilai A

Tabel 2 menunjukkan perbandingan dan perhitungan data pesan rahasia yang telah dienkripsi dan akan disisipkan pada *file audio* dengan menggunakan teknik *Low Bit Encoding*.

Tabel 2 Perbandingan Enkripsi Data

Jumlah Huruf (JH)	**Jumlah *Hexadecimal* Setelah Enkripsi (JHSE)**	**Jumah Bit (JB)** JHSE * 8
1	8	64
7	8	64
8	16	128
15	16	128
16	24	192
23	24	192
24	32	256

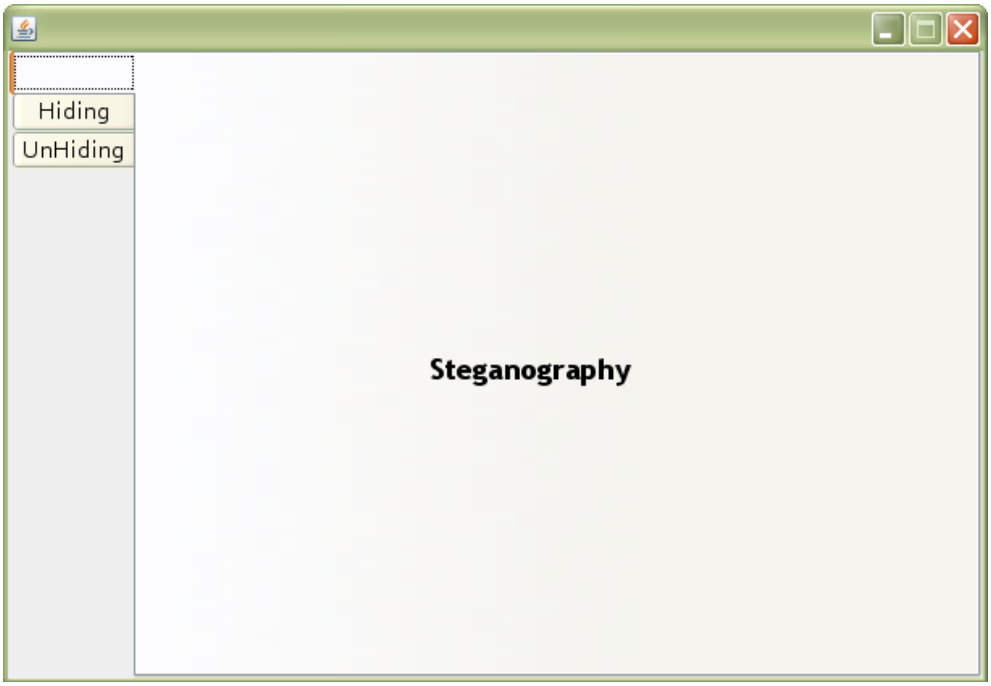
Jumlah Huruf (JH)	Jumlah *Hexadecimal* Setelah Enkripsi (JHSE)	Jumah Bit (JB)
31	32	256
39	40	320
47	48	384

Dari Tabel 2 dapat disimpulkan bahwa apabila jumlah huruf (JH) dari pesan rahasia yang akan disisipkan adalah 1, maka jumlah *hexadecimal* setelah enkripsi (JHSE) adalah 8, maka jumlah bit yang akan disembunyikan adalah 64 bit, sehingga dibutuhkan 64 nilai *hexadecimal* pada *audio* yang akan disisipkan dengan menggunakan teknik *Low Bit Encoding*. Untuk 7 JH, JHSE adalah 8 juga, dan untuk 8 JH, JHSE adalah 16, dan untuk 15 JH, JHSE adalah 16 juga, dan untuk 23 JH, JHSE adalah 24. Dari data pada Tabel 2 maka dapat disimpulkan bahwa proses enkripsi data adalah maksimal 7 JH untuk 8 *hexadecimal*, kemudian maksimal 15 JH untuk 16 *hexadecimal*, kemudian maksimal 23 JH untuk 24 *hexadecimal*. Maka proses enkripsi data setiap 8 *hexadecimal* adalah dimulai dari kelipatan 7, 8, 8, dan seterusnya.

Steganography file audio ini juga mendukung teknik *End Of File (EOF)* yang merupakan salah satu teknik *steganography* dengan menyisipkan data pada akhir *file.* Apabila kapasitas pada *audio file* sudah tidak dapat menampung data pesan rahasia, maka teknik *EOF* akan digunakan. Data pesan rahasia yang sudah tidak dapat ditampung pada *audio file* akan disisipkan pada akhir dari *audio file*, sehingga secara otomatis akan terjadi perbesaran kapasitas yang cukup signifikan dari *file* asli dan *file* hasil *steganography.* Untuk teknik ini, selanjutnya akan dijelaskan pengujian yang lebih detail dengan menggunakan teknik *EOF*

6. Hasil dan Implementasi

- *Main Form*

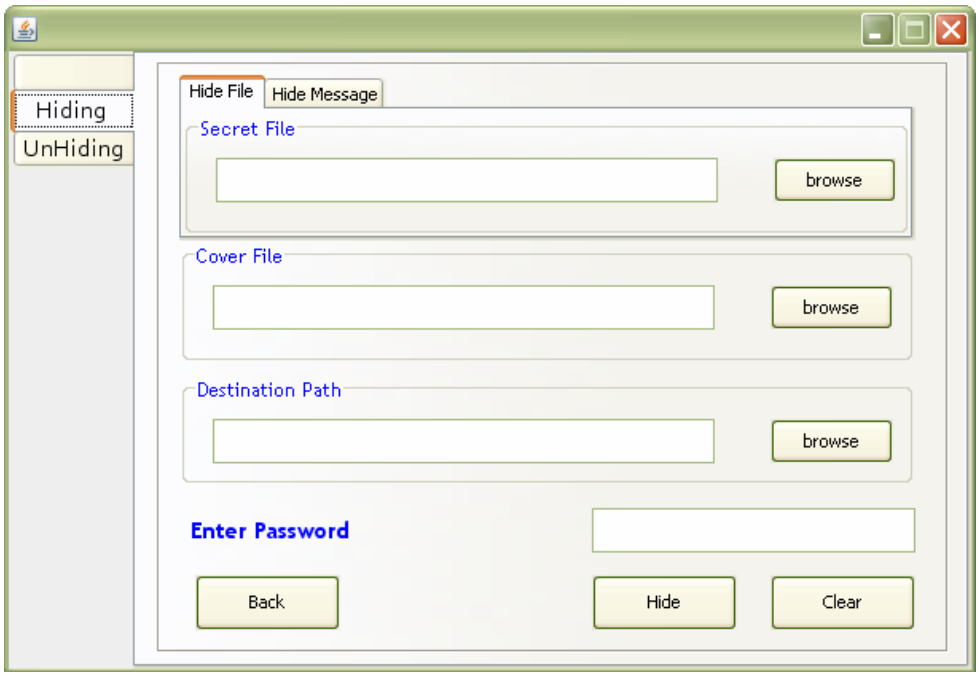


Gambar 4 Main Form

Gambar 4 merupakan realisasi dari rancangan antarmuka pengguna untuk *MainForm*. *Form* ini adalah tampilan utama yang muncul ketika pengguna menjalankan aplikasi. *Form* ini terdiri dari:

- *Tab Panel* yang terdiri dari *Hide* dan *Unhide* untuk menampilkan *panel Hide* atau *Unhide*.

- *Hide Panel*

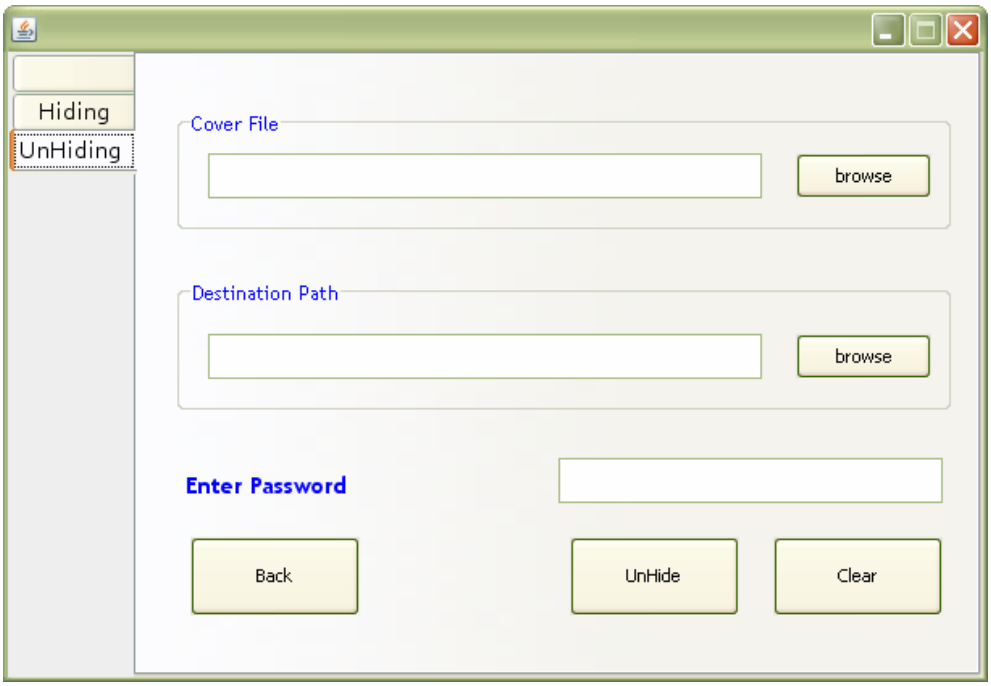


Gambar 5 Hide Panel

Gambar 5 merupakan relasasi dari rancangan antarmuka pengguna untuk penyisipan pesan rahasia. *Form* ini muncul ketika pengguna menekan *tab Hiding*. *Form* ini terdiri dari:

- *Tab Panel* yang terdiri dari *Hide File* untuk menyisipkan pesan rahasia dalam format txt dan *Hide Message* untuk menyisipkan pesan rahasia yang dituliskan langsung pada aplikasi ini.
- *TextField* untuk menuliskan alamat *directory* dari *Secret Message/Secret File, Cover File, Destination Path,* dan *Password*.
- *Button browse* untuk mendapatkan alamat *directory Secret File*, *Cover File* dan *Destination Path*. *Button Hide* untuk proses penyisipan *file*. *Button Clear* untuk membersihkan *field*. *Button Back* untuk kembali ke *panel* sebelumnya.

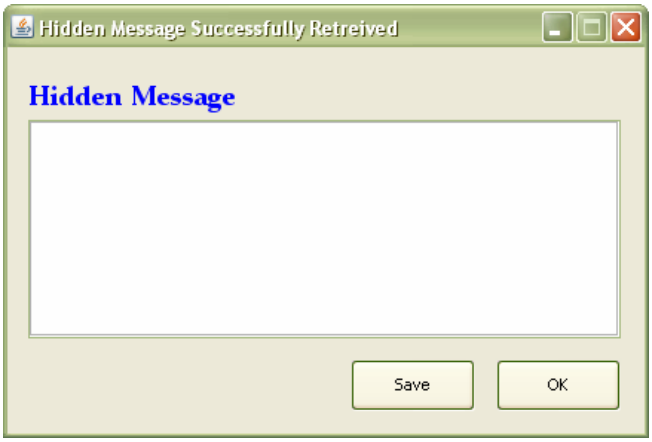
- *Unhide Panel*



Gambar 6 Unhide Panel

Gambar 6 merupakan realisasi dari rancangan antarmuka pengguna untuk ekstraksi pesan rahasia. *Form* ini muncul ketika pengguna menekan *tab Unhiding*. *Form* ini terdiri dari:

- *TextField* untuk menuliskan alamat *directory dari Cover File*, *Destination Path*, dan *Password*.
- *Button browse* untuk mendapatkan alamat *directory Cover File* dan *Destination Path*. *Button Unhide* untuk proses ekstraksi *file*. *Button Clear* untuk membersihkan *field*. *Button Back* untuk kembali ke *panel* sebelumnya.
- *Secret Message Frame*



Gambar 7 Secret Message Frame

Gambar 7 merupakan realisasi dari rancangan antarmuka pengguna untuk membaca pesan rahasia, dengan asumsi pada saat penyisipan *file* proses yang digunakan adalah *hide message*. *Form* ini muncul ketika pengguna menekan *button unhide* pada saat proses ekstraksi *file* telah selesai. *Form* ini terdiri dari:

- *TextArea* untuk menuliskan pesan rahasia yang telah disisipkan pada *cover file*.
- *Button save* untuk menyimpan pesan rahasia dalam format txt. *Button* ok untuk keluar dari *frame* ini.

7. Simpulan

Kesimpulan yang dapat ditarik dari hasil evaluasi yaitu secara umum aplikasi ini menghasilkan nilai guna yang cukup tinggi. Aplikasi ini memberikan solusi pada masalah keamanan. Beberapa hal yang ditawarkan aplikasi ini terhadap pengguna di antaranya penyisipan pesan rahasia dengan menggunakan enkripsi *password* serta ekstraksi kembali pesan rahasia pada *file-file* seperti gambar (*bmp*, *jpeg*), media (*wav*, *mpeg*), dan lain-lain (*exe*, *pdf*).

Tujuan dari aplikasi ini yaitu membantu pengguna dalam mengamankan suatu pesan yang akan disampaikan kepada orang yang berhak membacanya. Penggunaan media multimedia dalam dunia internet sudah berkembang sangat pesat sehingga kecurigaan akan adanya pesan rahasia pada *file* multimedia sangat kecil sekali.

Setiap pengujian *file* mengalami perubahan yang berbeda-beda. Pada *file* gambar yang telah mengalami *steganography* akan mengalami perubahan kontras pada gambar, gambar akan tampak lebih terang.

Untuk Pengujian *file* media, *file* media akan mengalami kerusakan pada *audio*. Kerusakan yang dimaksud adalah adanya *noise* pada *audio*. Kerusakan pada *audio*

tersebut sesuai dengan ukuran dari pesan rahasia yang disembunyikan. Apabila ukuran pesan rahasia kecil, *noise* pada *audio* tidak akan terlalu banyak.
Untuk pengujian *file* lainnya seperti *exe* atau *pdf*, *file* yang telah disisipi informasi mengalami perubahan ukuran yang cukup signifikan. Besar ukuran *file* yang telah disisipi informasi menjadi ukuran *file* asli ditambah dengan ukuran pesan rahasia yang akan disembunyikan.

Daftar Pustaka

[Bri09] Brittnee Morgan, uri.com-index. Retrieved February 10, 2009 Available:http://www.uri.edu/personal2/love0945/stegdetection3.htm

[Cyg09] Cygwin, inonit.com-index. Retrieved April 17, 2009 Available:http://www.inonit.com/cygwin/jni/helloWorld/load.html

[Eve09] Evergreen (BCD), academic.evergreen.edu-index. Retrieved April 17, 2009 Available:http://academic.evergreen.edu/projects/biophysics/technotes/program/bcd.htm

[Geo98] George Mason University, jjtc.com:*File*s. Retrieved February 1998 Available:http://www.jjtc.com/pub/r2026.pdf

[Inf04] InformIT, informit.com-index. Retrieved Maret 01,2004 Available:http://www.informit.com/guides/content.aspx?g=security&seqNum=103

[Jav05] Java Sound Resources, jsresources.org-index. Retrieved February 17, 2005 Available:http://www.jsresources.org/

[Net09] Netbeans, java.sun.com-index. Retrieved April 15, 2009 Available:http://java.sun.com/developer/onlineTraining/Programming/JDCBook/jni.html

[Ste09] StegoArchive, home.comcast.net-index. Retrieved February 05, 2009 Available:http://home.comcast.net/~ebm.md/stego.html

[Sun09] SunSteganography, 123eng.com-index. Retrieved June 23, 2009 Available:http://www.123eng.com/sourcecode/java/java-swing/sun-stegnography.html

[Tec09] TechFaq, tech-faq.com-index. Retrieved February 05, 2009 Available:http://www.tech-faq.com/steganography.shtml