Studi dan Pengujian Algoritma Steganografi pada Aplikasi Steghide

Ratna Mutia S / 13503086

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl Ganesha 10, Bandung
E-mail : if13086@students.if.it.ac.id

Abstrak

Makalah ini membahas mengenai studi dan pengujian aplikasi Steghide. Studi dilakukan terhadap algoritma Steganografi yang diimplementasikan dalam aplikasi Steghide. Steganografi merupakan ilmu atau seni dalam menyembunyikan informasi dengan memasukkan informasi tersebut ke dalam pesan lain. Dalam perkembangannya steganografi memanfaatkan media dijital seperti *file* gambar, musik, film, dan sebagainya. Sedangkan Steghide merupakan aplikasi untuk menyembunyikan pesan ke dalam media dijital berformat jpeg, bmp, wav, dan au. Steghide dirancang oleh Stefan Hetzl untuk mengimplementasikan algoritma *graph-theoretic approach*.
Algoritma *graph-theoretic approach* digunakan pada Steghide dengan sebuah prinsip dasar bahwa mengganti sebuah pixel lebih dipilih daripada menimpa pixel tersebut. Sebuah graf kemudian dibangun dari *cover-data* dan pesan rahasia. Pada graf tersebut, pixel direpresentasikan sebagai sebuah verteks sedangkan pasangan setiap pixel dihubungkan melalui sisi. Melalui proses studi diketahui bahwa algoritma ini tahan terhadap serangan *first-order statistical test.*
Pengujian dilakukan terhadap Steghide melalui tiga jenis pengujian. Pertama, Steghide diuji terhadap *fidelity*, *robustness*, dan *recovery* Kedua, Steghide diuji langsung dengan program yang mampu melakukan steganalisis yaitu Stegdetect. Ketiga, Steghide diuji melalui program kecil CDiff yang dibuat penulis untuk mengetahui posisi bit yang berubah pada *cover-data*.
Hasil pengujian menunjukan bahwa Steghide tidak memenuhi kriteria *fidelity* dan *robustness*. Kemudian Steghide terbukti tahan terhadap serangan yang dilakukan oleh Stegdetect. Terakhir, CDiff telah menunjukan bahwa posisi bit yang berubah pada *stego-data* Steghide tersebar hampir ke semua pixel.

Kata kunci : *graph-theoretic approach, Steghide, Steganograf, Stegdetect, first-order statistical test*

1. Pendahuluan

Tujuan dari steganografi adalah menyembunyikan pesan sedemkian rupa sehingga keberadaan pesan tersebut tidak dapat dikenali. Oleh karena itu terdapat kriteria tertentu yang perlu dipenuhi setiap algoritma steganografi agar dapat mencapai tujuan tersebut.

Seiring perkembangan ilmu steganografi, para steganografer terus melakukan perbaikan terhadap algoritma steganografi. Di sisi lain para steganalisis terus mencari kelemahan setiap algoritma yang muncul. Kedua aktifitas ini secara tidak langsung semakin mengembangkan teknik steganografi.

Steghide mengimplementasikan sebuah algoritma yang tergolong baru. Oleh karena itu, sebagai sebuah aplikasi, steghide perlu diuji sehingga Steghide dapat terus dikembangkan dan tahan terhadap serangan-serangan baru yang dimunculkan para steganalis. Sebagai sebuah *open sou*rce, Steghide memiliki potensi untuk itu.

Studi terhadap algoritma kemudian melakukan pengujian terhadap aplikasi diharapkan dapat menjadi dasar pengembangan Steghide menjadi aplikasi steganografi yang lebih baik dan aman.

1. Steganografi

Diilustrasikan Alice dan Bob berada sama-sama di penjara. Pada suatu saat keduanya akan menyusun rencana untuk kabur dari penjara. Alice harus mengirimkan pesan tersebut melalui kurir. Jika Alice mengenkripsi pesan rahasianya, kurir tentu akan curiga. Oleh karena itu Alice dan Bob perlu menggunakan suatu teknik sehingga kurir tidak dapat mendeteksi adanya pesan rahasia. Teknik tersebut dikenal dengan sebutan steganografi.

1.1 Pengertian Steganografi

Steganografi merupakan suatu ilmu atau seni dalam menyembunyikan informasi dengan memasukkan informasi tersebut ke dalam pesan lain. Dengan demikian keberadaan informasi tersebut tidak diketahui oleh orang lain. [2]

Tujuan dari steganografi adalah menyembunyikan keberadaan pesan dan dapat dianggap sebagai pelengkap dari kriptografi yang bertujuan untuk menyembunyikan isi pesan. Oleh karena itu, berbeda dengan kriptografi, dalam steganografi pesan disembunyikan sedemikian rupa sehingga pihak lain tidak dapat mengetahui adanya pesan rahasia. Pesan rahasia tidak diubah menjadi karakter 'aneh' seperti halnya kriptografi. Pesan tersebut hanya disembunyikan ke dalam suatu media berupa gambar, teks, musik, atau media digital lainnya dan terlihat seperti pesan biasa.

1.2 Terminologi

Dalam steganografi dikenal beberapa terminologi. *Cover-data* atau *cover-text* merupakan media penyembunyi pesan. Sedangkan hasil penggabungan antara *coverdata* dengan pesan yang disembunyikan disebut *stego -text, stego-data,* atau *stego-object.* Algoritma yang menghasilkan *stego text* disebut *stegosystem*. Pihak yang menciptakan *stegosystem* disebut steganografer. [2]

Secara umum *stegosystem* terdiri dari tiga tahap yaitu algoritma untuk mendapatkan kunci, mengkodekan pesan, dan men-*decode* pesan. Algoritma tersebut dibungkus dalam suatu teknik tehnik penyembunyian pesan yang bermacam-macam [2]

Gambar 1 Cover-data

Gambar 2 Stego-data

Gambar 1 merupakan *file* gambar lena.jpg yang dijadikan sebagai cover-data. Sedangkan gambar 2 merupakan *file* gambar lena.jpg yang telah dimasukan pesan rahasia berupa teks melalui aplikasi steganografi. Terlihat bahwa dengan mata manusia yang terbatas, perbedaan kedua gambar tersebut tidak terlihat. Keberadaan pesan rahasia di dalam gambar 2 pun tidak dapat diketahui keberadaannya oleh pihak lain .

Bandingkan apabila pesan rahasia tersebut dikenakan teknik kriptografi. Misal :

Pesan rahasia : kabur melalui ruang A
Pesan hasil enkripsi : qwertyuiwirotrwbfm

Pesan hasil enkripsi menjadi serangkaian huruf yang tidak bermakna sehingga keberadaan pesan rahasia mudah dikenali.

1.3 Sejarah dan Perkembangan

Istilah steganografi berasal dari bahasa Yunani yaitu *steganos* yang artinya adalah penyamaran atau penyembuyian dan *graphein* yang artinya adalah tulisan.

Pada tahun 400 SM di Yunani, dikenal istilah *wax tablet*. Pesan ditulis di atas media kayu kemudian disembunyikan dengan melapisi lilin sebagai penutupnya. [6]

Pada Perang Dunia II, pesan diperkecil menjadi sebuah titik yang ditaruh di bawah perangko dan penyembunyian pesan dengan menggunakan tinta transparan. [6] Steganografi mengalami kemajuan yang sangat pesat sejak tahun 1990-an ketika pemerintah, industri, warga negara biasa bahkan organisasi teroris mulai menggunakan aplikasi perangkat lunak steganografi untuk menyembunyikan pesan-pesan atau foto ke dalam beberapa tipe media.

Akhir-akhir ini kata steganografi menjadi sering disebut di masyarakat bersama-sama dengan kata kriptografi setelah pemboman gedung WTC di AS, ketika para pejabat AS mengkalim bahwa para teroris menyembunyikan pesan-pesan kegiatan terornya dalam berbagai gambar porno, *file* MP3 dan web site tertentu. Novel Da Vinci Code pun turut mempopulerkan steganografi dan kriptografi. Saat ini penyembunyian pesan memang telah banyak menggunakan media dijital seperti MP3, gambar digital berformat JPG, film, arsip, dan lain-lain.

1.4 Teknik Steganografi

Walaupun steganografi telah berada dalam berbagai media, pola dasar dari steganografi tidak pernah berubah yaitu bahwa pesan rahasia disembunyikan ke dalam *cover-data* menjadi *stego-data*. Terdapat tiga kriteria yang menentukan apakah metode yang digunakan untuk steganografi itu baik atau tidak. Pertama, *stego-data* harus dapat mencapai tujuan dari *steganografi* yaitu bahwa pesan rahasia tidak dapat difilter atau dirusak. Kedua, keberadaan pesan rahasia dalam *stego-data* tidak dapat dikenali oleh penyerang. Ketiga, penerima dapat membaca pesan rahasia tersebut.

Walaupun *file* bertipe gambar menjadi *cover-data* yang paling banyak digunakan saat ini, masih terdapat sejumlah strategi untuk menyembunyikan pesan rahasia dijital. Spasi pada sebuah *file* teks dapat digantikan oleh karakter lain yang merupakan ejaan dari pesan rahasia. *File* suara dapat meyembunyikan pesan rahasia dalam *audio noise*. *File* bertipe exe mempunyai area untuk menyimpan pesan rahasia. Bahkan sebuah jaringan pun dapat menyimpan pesan rahasia di belakangnya.

Keragaman strategi tersebut dibutuhkan karena bagaimanapun dalam steganografi terdapat sekelompok orang yang akan terus mencari kelemahan dan kesalahan dari setiap teknik. Sekelompok orang tersebut disebut steganalis. Dan seperti halnya kriptografi, belum ada penyembunyian pesan yang sempurna dan tidak dapat dipecahkan.

Terdapat metode-metode pendeteksi pesan dalam steganografi yang dibuat para steganalis. Seiring ditemukannya kelemahan dari stegosystem yang diajukan para steganografer, bermunculanlah teknik-teknik penyembunyian pesan yang baru.

Salah satu teknik steganografi dijital yang sederhana adalah LSB. Pada susunan bit di dalam sebuah byte ada bit yang paling berarti dan bit yang paling kurang berarti. Misalkan pada byte 11010010, bit 0 yang terakhir adalah bit LSB. Bit yang cocok untuk diganti adalah bit LSB, sebab penggantian hanya mengubah nilai byte tersebut satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Jika byte tersebut di dalam gambar menyatakan warna tertentu, maka perubahan satu bit LSB tidak mengubah warna tersebut secara berarti. Mata manusia pun tidak dapat membedakan perubahan warna yang kecil. [1]

Untuk memperkuat penyembunyian data, bit-bit data tidak digunakan untuk mengganti byte-byte yang berurutan, namun dipilih susunan byte secara acak. Misalnya terdapat 50 byte dan 6 bit data yang akan disembunyikan, maka byte yang diganti bit LSB-nya dipilih secara acak, misalkan byte nomor 36, 5, 21, 10, 18, 49.

1. 5 Aplikasi Steganografi

Program untuk melakukan steganografi telah banyak dibangun. Di antaranya jsteg, outguess, jphide, MP3stego, S-Tool dan lain-lain. Pada tahun 2001 ditemukan steganalisis untuk mendeteksi keberadaan sebuah pesan dalam gambar berformat JPG dan diimplementasikan dalam sebuah program bernama stegdetect.

Salah satu teknik steganalisis yang digunakan Stegdetect adalah *linear discriminant analysis*. [3] Dengan demikian antara bagian pixel asli dengan pixel yang telah dikenakan pesan rahasia dapat dibedakan. Stegdetect telah berhasil mendeteksi keberadaan pesan rahasia pada *stego-data* hasil jsteg, jphide, dan outguess. Karena ketidakamanan itulah teknik steganografi yang lebih baik diajukan para steganografer.

2. Steghide

2.1 Deskripsi Program

Steghide merupakan salah satu program yang mengimplementasikan teknik penyembunyian pesan yang tahan terhadap Stegdetect. Program ini dirancang oleh Stefan Hetzl untuk menyembunyikan beragam *file* dalam format gambar dan audio. Program ini

mengimplementasikan *stegosystem* yang tahan terhadap serangan *first order statistical test.* [5]

Steghide dibangun sebagai *open source* oleh karena itu dapat secara bebas dimodifikasi dan didistribusikan di bawah GNU *General Public Licence.* [5] Dengan demikian Steghide berpotensi untuk terus dikembangkan oleh siapa saja.

Saat ini Steghide telah mencapai versi 0.5.1 dan dapat diperoleh melalui http://steghide.sourceforge.net/ Steghide telah dapat dijalankan pada sistem operasi Unix dan Windows.

2.1 Fitur

Secara rinci fitur yang dimiliki oleh Steghide adalah : [5]
- Mengkompresi data yang disembunyikan
- Mengenkripsi data yang disembunyikan
- Mendukung arsip berformat JPEG, BMP, WAV, dan AU
- Melakukan *checksum*

Dibanding dengan kakas steganografi yang telah ada seperti jsteg ataupun MP3stego, fitur yang dimiliki Steghide lebih banyak. Akan tetapi Steghide belum memiliki GUI dan masih dijalankan melalui *command prompt*.

2.2 Implementasi

Dalam implementasinya Steghide dibangun berorientasi objek dan menggunakan bahasa C++. Total terdapat 48 kelas yang diimplementasikan. Dalam menerapkan algoritma enkripsi dan kompresi, Steghide dibantu *library* yang telah ada seperti :[5]
- libmhash
  Library yang menghasilkan beragam algoritma Hash dan algoritma untuk mengolah kunci kriptografi. Steghide menggunakan *library* ini untuk mengubah password ke dalam format yang dapat digunakan langsung oleh algoritma steganografi dan kriptografi yang digunakan
- libmcrypt
  Library yang menghasilkan beragam algoritma enkripsi kunci simetri. Tanpa *library* ini, pengguna tidak dapat menggunakan Steghide untuk mengenkripsi data sebelum menyembunyikan data rahasia atau membuka data terenkripsi walaupun pengguna mengetahui *password*.
- libjpeg
  Library ini mengimplementasikan kompresi terhadap *file* JPEG. Tanpa *library* ini, pengguna tidak dapat menggunakan Steghide untuk menyembunyikan pesan dalam format JPEG.
- zlib
  Library yang menangani kompresi data *lossless*. Tanpa *library* ini, pengguna tidak dapat mengkompresi data sebelum menyembunyikan data rahasia ataupun membuka *stegodata* yang terkompresi.
- libmhash
  Digunakan untuk mengkompilasi Steghide.

Tanpa *library-library* tersebut, kemampuan Steghide menjadi terbatas. Akan tetapi, bagi pengguna Unix, *library* tersebut biasanya telah terinstalasi pada sistem. Sedangkan bagi pengguna Windows, *library* telah disediakan dalam paket instalasi.

2.3 Proses

Melalui manual Steghide diketahui bahwa Steghide menggunakan *graph-theoretic approach* sebagai *stego-system*. Penjelasan detail mengenai *graph-theoretic approach* akan dijelaskan bab 3. Secara umum tahap steganografi pada Steghide adalah [5]:
- Pesan rahasia dienkripsi dan dikompresi.
- Melalui *password* dihasilkan angka secara random. Angka tersebut menandakan posisi dari *pixel* pada *cover-data.* Pesan rahasia akan ditempatkan pada posisi *pixel* tersebut. Posisi yang tidak perlu diubah dikeluarkan dari list
- Algoritma *graph-theoretic matching* akan menemukan pasangan posisi yang cocok sehingga apabila nilai pixelnya diganti akan tersusun sebagai bagian dari pesan rahasia
- Apabila algoritma *graph-theoretic matching* sudah tidak dapat menemukan pasangan posisi pixel maka posisi *pixel* yang tidak termasuk ke dalam pasangan yang ditemukan algoritma akan dimodifikasi. Modifikasi akan dilakukan dengan tidak mengganti pixel akan tetapi menimpa *pixel* tersebut dengan pesan rahasia tersisa.

Dikarenakan proses steganografi lebih banyak dilakukan dengan mengganti nilai dari *pixel* dibandingkan dengan menimpa *pixel* maka *stego-data* dari Steghide tahan terhadap steganalisis yang menggunakan prinsip *first-order statistic. First-order statistic* merupakan

perhitungan kuantitas kemunculan warna pada gambar. Dengan menimpa pixel maka nilai kuantitas ini akan berubah. Sedangkan dengan hanya mengganti nilai pixel, *first-order statistic* antara *stego-data* dengan *cover-data* tidak akan berubah banyak.
Untuk *file* berformat audio seperti WAV dan AU, proses yang diterapkan Steghide terhadap *cover-data* dan pesan rahasia adalah sama. Perbedaaanya adalah proses dilakukan terhadap sampel bukan terhadap pixel.

Proses enkripsi yang dilakukan pada awal proses terhadap pesan rahasia dilakukan untuk lebih menambah keamanan pada stego-data tersebut. Algoritma enkripsi yang diimplementasikan secara default pada Steghide adalah Rijndael dengan panjang kunci 128 bit. Sedangkan algoritma enkripsi lain yang juga dapat digunakan oleh pengguna adalah Cast dengan 128 atau 256 bit, Rijndael dengan 192 atau 256 bit, Gost, Twofish, Arcfour, LOKI, Saferplus, Wake, DES, Serpent, Xtea, Blowfish, Enigma, RC2, dan TripleDES. Algoritma enkripsi yang dilakukan menggunakan mode *chiper blok* seperti CBC, CFB, OFB, ataupun ECB

Proses kompresi dilakukan agar ukuran *file* pesan rahasia menjadi lebih kecil sehingga dapat menyesuaikan dengan kapasitas *stego-data* dalam menyembunyikan data. Sedangkan proses *checksum* dilakukan untuk menjaga integrity dari data. Checksum dihitung dengan menggunakan algoritma CRC32.

Steghide juga memasukan data nama *file* pada *stego-data.* Dengan demikian ketika penerima pesan mengektraksi *stego-data*, hasil ekstraksi akan langsung tersimpan dengan nama *file* tersebut.

Pada penggunaan Steghide, fitur *enkripsi*, *kompresi*, *checksum*, dan penyimpanan nama *file* dapat diatur sedemikian rupa. Pengguna dapat memilih algoritma enkripsi yang digunakan, apakah pesan akan dikompresi terlebih dahulu, apakah akan dilakukan proses *cheksum*, atau apakah nama *file* akan disimpan pada *stego-data*. Dikarenakan fleksibilitas tersebut, pilihan strategi pada Steghide lebih beragam dan akan menyulitkan steganalis untuk mendapatkan pesan rahasia.

Proses ektraksi pesan rahasia dari *stego-data* hanya dapat dilakukan oleh pemegang kunci dalam hal ini pihak pengirim dan penerima pesan. Steghide juga memfasilitasi bagi pengguna untuk mengetahui informasi *file stego-data* maupun *file* pesan yang disembunyikan. Dengan syarat pengguna tersebut harus mengetahui kunci. Kunci tidak disimpan dalam *file stego-data.* Kunci inilah yang menentukan proses ektraksi.

Seperti teknik *steganografi* pada umumnya, proses ekstraksi pesan rahasia dari *stego-data* merupakan kebalikan dari proses penyembunyian data. Berdasarkan kunci yang dimasukan pengguna, Steghide akan memparsing *file stego-data* untuk memperoleh pesan rahasia. Adapun proses ekstraksi secara umum adalah sebagai berikut :

- Membaca masukan kunci dari pengguna
- Membaca masukan *stego-data*
- Menghasilkan angka acak yang diturunkan dari kunci
- Membaca *pixel-pixel* pada posisi *pixel* tertentu berdasarkan angka acak
- Menyusun hasil pembacaan menjadi pesan rahasia
- Mengekstrak dan mendekripsi hasil pembacaan
- Menghilangkan *checksum*
- Menulis hasil akhir pada *file* yang telah didefinisikan pada *stego-data*

3. A Graph Theory Approach

3.1 Pendahuluan

A graph theory approach merupakan algoritma yang dikembangkan oleh Stefan Hetzl bersama Peter Mutzel. Latar belakang dari pembentukan algoritma baru ini adalah bahwa *stego-system* yang ada saat itu telah tidak aman karena pesan rahasia yang ada pada *stego-data,* berformat JPEG khususnya, telah dapat dikenali keberadaannya. Teknik steganalisis yang digunakan salah satunya menggunakan *first-order statistic.* Seperti telah dibahas sebelumnya bahwa perubahan *first-order statistic* menandakan bahwa terjadi perubahan dari gambar asli terhadap stego-data. Sehingga keberadaan pesan rahasia dalam *file* gambar berformat JEPG dapat diketahui. Melalui proses analisis selanjutnya misalnya melaui *brute-force dictionary attack* pesan rahasia pun dapat diketahui. Oleh karena itu, menurut pemikiran Stefan Hetzl dan Peter Mutzel, diperlukan algoritma baru yang tahan terhadap serangan dengan *first-order statistic test.* [4]

Paradigma *stego-system* yang ada saat itu adalah pesan rahasia disusun sedemikian rupa sehingga menimpa pixel pada *cover-data*. Cara ini mengakibatkan perubahan cukup besar pada *first-order statistic* gambar. *A Graph Theory Approach* kemudian dirancang dengan paradigma yang berbeda. Pixel-pixel pada gambar diganti dengan komponen data pesan rahasia, bukan ditimpa. Akibatnya tidak terjadi perubahan *first-order statistic* pada gambar.

Secara garis besar, proses yang terjadi adalah : [4]

- Pixel-pixel yang akan dimodifikasi direpresentasikan sebagai puncak atau verteks.
- Setiap pixel akan dipasangkan dengan pixel yang sesuai. Setiap pasangan yang mungkin dihubungkan oleh garis (sisi) dengan pixel tersebut.
- Pesan rahasia disembunyikan ke dalam pixel-pixel tersebut dengan cara penyelesaian komputasi kombinatorial dengan menghitung pencocokan kardinalitas maksimum.
- Hasil perhitungan digunakan sebagai penentu dimanakah pesan rahasia akan menggantikan pasangan pixel yang sesuai
- Untuk meminimalisasi perubahan secara visual, setiap sisi dikenai bobot tertentu.

Selain tahan terhadap *first-order statistical test,* kelebihan dari algoritma ini adalah tidak adanya kebergantungan sistem pada tipe *cover-data* yang digunakan (misalnya gambar, audio,..)

3.2 Terminologi Algoritma

Terminologi yang digunakan dalam algoritma in adalah : [4]

- Sampel
  Diistilahkan sebagai satuan data terkecil. Misalkan sampel pada *file* bertipe gambar adalah pixel.
- S
  Simbol dari himpunan nilai tiap sampel.
- $D = (s_1,\ldots,s_n)$
  Cover-data ataupun stego-data merupakan *array of* sampel. Disimbolkan dengan D
- $P \subseteq \{1,\ldots,N\}$
  Himpunan frekuensi nilai sampel pada S merupakan himpunan P dengan $|\{i \in P \mid s_i = x\}|$
- $v : S \rightarrow \{0, \ldots, m-1\}$
  Fungsi yang menandakan nilai tiap sampel diganti dengan nilai pesan rahasia
- $Vi(D) = v(sk\cdot(i-1)+1) \oplus m \ldots \oplus m\ v(sk\cdot(i-1)+k)$, dimana $1 \le i \le \left\lfloor \frac{N}{k} \right\rfloor$
  Suatu nilai disisipkan pada sejumlah atau himpunan sampel bukan pada sampel tunggal atau lebih tepatnya pada modulo m dari jumlah nilai yang akan disisipkan
- $Em = (e_1,\ldots e_n)$ dimana $e_1 \in \{0,\ldots,m\text{-}1\}$ untuk $i \in \{1,\ldots,n\}$
  Em merupakan simbol dari pesan rahasia
- *Graph* G terrdiri dari (V,E)
  V adalah himpunan *verteks* dan $E \subseteq VxV$ merupakan himpunan dari sisi. *Graph* yang digunakan merupakan *graph* tidak berarah sehingga $(x,y) \in E$ dan $(y,x) \in E$ merupakan sisi yang sama.
- c(e) untuk setiap $e \in E$
  c(e) merupakan simbol bobot pada setiap sisi G
- $M \subseteq E$
  M merupakan himpunan *verteks* dengan sisi
- sisi $e \in E$ dikatakan cocok dengan suatu M jika $e \in M$.
- *Maximum cardinality matching*
  Bagian graph G yang memiliki nilai kardinalitas maksimum
- *Maximum cardinality minimum weight matching*
- M merupakan *maximum cardinality matching* yang memiliki bobot paling minimum

3.3 Membangun Graf

Terdapat dua definisi dalam proses pembangunan sebuah graf [4]. Kedua definisi tersebut adalah :

Definisi 1 :
Misal $C = (s_1,\ldots,s_n)$ merupakan *cover data*
Verteks berupa struktur (P,T) dimana $P = (p_1,\ldots,p_k) \in \{1,\ldots,N\}^k$ dengan k tupel sampel pada *cover-data* dan $T = \{t_1,\ldots t_k) \in \{0,\ldots,m\text{-}1\}^k$ dengan k tupel dari nilai target.

Maksud dari definisi 1 tersebut adalah : Terdapat satu sampel yaitu s_{p1}, $i \in \{1,\ldots,k\}$ yang perlu diganti dengan nilai lain dengan fungsi $v(s_i^*) = t_i$, untuk menyisipkan sebagian pesan rahasia. Angka k merupakan sesuatu yang bernilai tetap untuk sebuah graf dan disebut sebagai rasio sampel per verteks.

Gambar 3 menampilkan hasil pembentukan verteks dengan kasus sebagai berikut :

Terdapat cover-data C= ($s_1,\ldots s_n$) dengan nilai k=3, m=4, dan pesan rahasia $E_4 = (e_1,\ldots,e_n)$
Baris pertama pada Gambar 3 merupakan himpunan sampel pada *cover–data*. Baris kedua merupakan nilai yang akan disipkan pada *cover-data*. Setiap ketiga (sesuai nilai k=3) nilai tersebut dijumlahkan kemudian dimodulokan dengen nilai m. Hasilnya berupa $V_l(C)$. Kemudian dilakukan perbandingan antara nilai $V_l(C)$ dengan e_l yang merupakan ke-3 bagian dari pesan rahasia. Jika kedua nilai yaitu $V_l(C)$ dengan e_l tidak sama maka verteks baru didefinisikan pada baris terakhir gambar 3. Nilai target diperoleh melalui perhitungan :

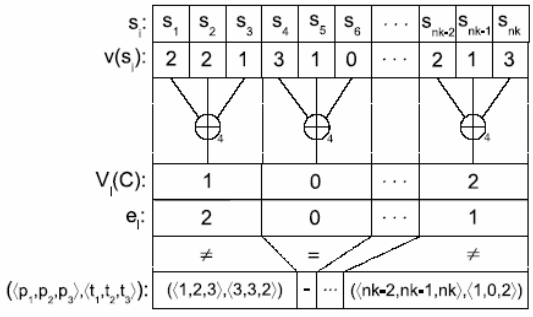
$$t_i = v(s_i) + e_l - V_l(C)$$

Contoh perhitungan nilai target berdasarkan Gambar 3 :
$t_1 = v(s_1) + e_l - V_l(C)$
$t_1 = 2 + 2 - 1$
$t_1 = 3$
Maka nilai sampel ke-1 akan diganti dengan nilai target ke-1 yaitu 3

$t_{nk} = v(s_{nk}) + e_l - V_l(C)$
$t_{nk} = 3 + 1 - 2$
$t_{nk} = 2$
Maka nilai sampel ke-nk akan diganti dengan nilai target ke-nk yaitu 2

Mengganti satu dari $v(s_i)$ yang berkaitan dengan t_1, t_2, atau t_3 menghasilkan e_l sebagai nilai dari verteks. Operasi ini merupakan penyisipan sebuah verteks.

Definisi 2:
Misalkan v dan w merupakan dua verteks dengan $v = (\{p_1, \ldots,p_k),(t_1,\ldots,t_2))$ dan $w = ((q_1,\ldots,q_k),(u_1,\ldots,u_2))$ dan misalkan $i,j \in \{1,\ldots,k\}$



Gambar 3 Hasil kontruksi verteks untuk k=3 dan m=4

Terdapat sebuah sisi yang menghubungkan ke-i nilai sampel dari v dengan ke-j nilai sampel dari w dan disimbolkan dengan $(v,w)_{i,j} \in E$ jika
$v(s_{pi}) = u_j$ and $v(s_{qj}) = t_i$

Sebuah sisi menghubungkan dua verteks dan diberi label berupa indeks dari salah satu nilai sampel dari setiap verteks. Pertukaran dari kedua nilai sampel dihasilkan dalam penyisipan kedua verteks. Sebagai catatan, memungkinkan kedua verteks dihubungkan oleh lebih dari dua sisi. Pada kasus tersebut, M hanya dapat terdiri dari salah satu sisi.

Definisi di atas tidak membatasi pertukaran yang menghasilkan distorsi yang terlihat jelas seperti penggantian pixel berwarna hitam dengan cyan. Oleh karena itu perlu didefinisikan sebuah pembatasan pada himpunan sisi yang memiliki kesamaan secara visual.

Didefinisikan sebuah fungsi $d : S \times S \rightarrow R$ dengan tujuan memperoleh jarak visual. Sebuah relasi dari kesamaan visual $\sim$ untuk $s_1, s_2 \in S$ disimbolkan sebagai :
$s_1 \sim s_2 \Leftrightarrow d(s_1, s_2) \leq r$ untuk sebuah radius ketetanggan sebesar r.

Dengan demikian definisi dari himpunan sisi dapat dilengkapi menjadi :

$$E^{\sim} = \{(v,w)_{i,j} \in E \mid s_{pi} \sim s_{qj}\}$$

Dimana $v = ((p_1,\ldots,p_k),(t_1,\ldots,t_2))$ dan $w = ((q_1,\ldots,q_k),(u_1,\ldots,u_2))$. Bobot sisi c: $E \rightarrow R$ merupakan jarak antara kedua nilai sampel $d(s_{p1},s_{p2})$.

3.4 Prinsip Penyisipan pada *Cover-data*

Tujuan dari proses penyisipan pesan adalah menemukan cara untuk memodifikasi *cover-data* sehingga dapat disisipkan keseluruhan bagian dari pesan rahasia. Untuk tujuan inilah, dilakukan perhitungan kecocokan pada setiap graf.

Setiap kecocokan yang ditemukan pada graf akan berkaitan dengan himpunan pengganti nilai sampel pada cover-data. Sehingga semua verteks yang cocok akan disisipkan oleh pesan rahasia. Tentu saja sulit untuk meraih kecocokan yang

sempurna pada keseluruhan *cover-data*. Untuk bagian verteks yang belum disisipkan pesan, terpaksa dilakukan penimpaan pixel terhadap sebuah sampel pada tiap verteks. Dan sebagai fitur juga, perlu dilakukan pencarian *minimal weight maximum matching* untuk meminimasi perubahan secara visual. [4]

3.5 Implementasi Stuktur Data

Algoritma *graph theory approach* telah diimplementasikan oleh Stefan Hetzl dalam pembangunan Steghide. Implementasi algoritma dan struktur data pada Steghide adalah sebagai berikut :

Jumlah pertambahan verteks dalam graf adalah O(n), n adalah ukuran dari pesan rahasia. Sednagkan jumlah pertambahan sisi sebesar $O(n^2)$. Sebagai contoh, sebuah graf untuk *file* gambar dengan data yang akan disisipkan sebesar 1 KB akan mempunyai rata-rata 3200 verteks dan 98000 sisi. Jika ukuran data yang disisipkan meningkat menjadi 4 KB, maka ukuran graf menjadi 12500 verteks dan 1470000 sisi. [4]

Bagaimanapun, tujuan awal adalah data berukuran besar dapat disisipkan pada sebuah *cover-data*. Besarnya ukuran graf tersebut mengakibatkan dua hal yaitu pertama, list terbatas tidak mungkin digunakan karena jumlah sisi akan sangat besar. Dan kedua, kalaupun terdapat algoritma yang sangat cepat untuk menyelesaikan masalah pencarian kecocokan verteks dengan waktu O(_|V | · |E|) maka perlu digunakan heuristic sederhana yang sama cepatnya.

Dengan demikian, bentuk implementasi yang diterapkan adalah graf tidak disimpan menggunakan list terbatas. Akan tetapi menggunakan struktur data yang memungkinkan dilakukannya *on-the-fly construction* pada seluruh sisi yang ada.

Struktur data yang dimaksud adalah pertama, list terbatas untuk nilai sampel dalam bentuk sebuah array. Array diindeks berdasarkan nilai sampel dengan setiap sample s mengandung list yang memuat seluruh nilai sample yang memiliki jarak $\leq$ r terhadap s. List kemudian diurut berdasarkan jarak terhadap s secara *ascending.* Kedua, sebuah array berindeks untuk menyimpan kejadian pada nilai sampel. Setiap sample s akan mengandung *list of pointer* terhadap verteks yang mengandung s. Dengan menggunakan kedua struktur data tersebut, iterasi terhadap verteks dapat dilakukan tanpa menyimpannya ke dalam sebuah list terbatas.

Seperti telah dijelaskan sebelumnya, heuristik akan digunakan untuk mempercepat proses kalkulasi dalam menemukan *maximum cardinality minimum weight matching.* Heuristik tersebut mengakibatkan verteks tersusun berdasarkan jumlah sisi yang ada pada tiap verteks. Proses penyusunan ini mengakibatkan nilai kardinalitas dibandingkan secara acak karena verteks dengan jumlah pasangan yang lebih kecil akan dicocokan terlebih dahulu.

Input : Graph $G = (V, E)$
Output : Matching M on G

Sort all vertices by degree in ascending order into $\langle v_1, \ldots, v_n \rangle$;
Initialize $M = \emptyset$;
Mark all vertices as active;
for $i = 1, \ldots, n$ **do**
  if v_i *is active and* $degree(v_i) > 0$ **then**
    Set $e = (v_i, w)$ = shortest edge of v_i;
    Set $M = M \cup \{e\}$;
    Mark v_i and w as inactive and delete all of their edges from E
  end
end
return M;

Algoritma 1 ***Static minum degree construction heuristic***

Setelah proses heuristic selesei, untuk sejumlah format data, dilakukan heuristic secara *Dept-first Search* untuk mengaugmentasi path. Proses ini dilakukan untuk meningkatkan nilai kardinalitas.

3.6 Hasil Pengujian Algoritma

Pengujian dilakukan dengan menerapkan algoritma terhadap beragam *file* gambar dan audio. Algoritma diterapkan dengan parameter yang berbeda-beda sehingga dapat ditemukan parameter yang sesuai dengan format pesan. Paramater yang dimaksud adalah rasio jumlah sampel per verteks k, modulus m, dan radius r. Hasil pengujian ditampilkan pada tabel 1[4]:

Tabel 1 Hasil Uji 1

	palette	true-color	jpeg	waveform	μ-law
radius r	20	10	1	20	1
samples/vertex k	3	2	3	2	2
modulus m	4	4	2	2	2
embedding rate	8.33%	4.16%	*5.86%*	3.13%	6.25%
algorithms	SMD	SMD	SMD	SMD,DFS	SMD,DFS

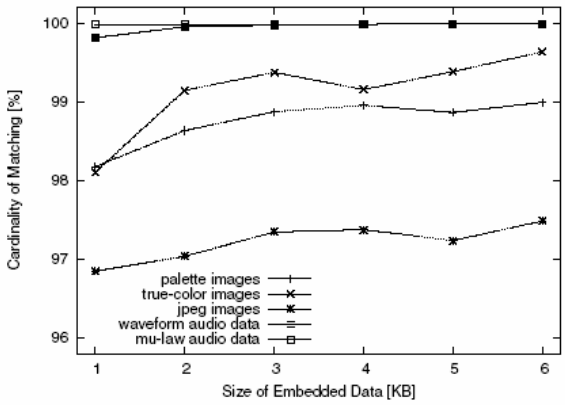
Fungsi jarak menggunakan Euclidean pada gambar bermode RGB dan untuk format lain,

fungsi yang digunakan adalah fungsi perhitungan jarak antar nilai sampel. Kemudian terdapat parameter lain yang disebut *embedding rate* atau tingkat penyisipan. Tingkat penyisipan adalah ukuran perbadingan pesan rahasia terhadap ukuran *cover-data* dengan rumus :

$$r = \frac{1}{\frac{2}{m} \cdot k \cdot s}$$

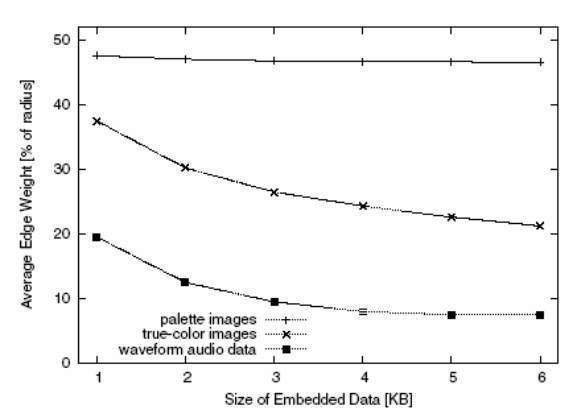
Nilai s merupakan simbol yang menyatakan ukuran sampel per bit. Sampel pada *file* JPEG adalah nilai koefisien dari transformasi diskrit *cosine* . Koefisien dengan nilai 0 tidak digunakan untuk menyisipkan data karena nilai 0 pada *cover-data* dan nilai 1 pada *stego-data* akan menghasilkan distorsi visual pada gambar. Oleh karena itu, tingkat penysipan pada *file* gambar diberikan secara empiris sehingga nilai yang tertera pada tabel hanya bersifat valid untuk gambar yang digunakan saat pengujian, tidak untuk gambar lain.

Parameter lain yang digunakan saat pengujian adalah algoritma heuristic yang digunakan. SMD merujuk pada *static minimum degree construction heuristic* sedangkan DFS merujuk pada *heuristic dept first search* untuk mengaugmentasi path. Khusus untuk data audio, DFS lebih dipilih sebagai proses akhir karena DFS lebih cepat dengan kualitas yang tetap baik dan memberikan solusi yang lebih baik dari SMD.



Gambar 4 Kardinalitas pencocokan

Gambar 4 menunjukan nilai kardinalitas dari sebuah perhitungan pencocokan. Nilai kardinalitas bernilai tinggi yaitu lebih dari 97% verteks memiliki kecocokan. Sedangkan untuk *file* audio , tingkat kardinalitas mencapai hampir 100%. Berbeda dengan *file* JPEG ynga hanya mampu mencapai angka 97%. Format JPEG memang lebih sulit ditangani karena nilai koefisian transformasi diskrit *cosine* tidak memiliki distribusi yang seragam sedangkan nilai sampelnya yang memiliki nilai absolute lebih kecil akan lebih sering muncul . Hasil tersebut menunjukan bahwa penyisipan dapt dilakukan tanpa mengubah nilai first order statistic.



Gambar 5 Rata-rata bobot sisi

Hasil pengujian lain direpresentasikan ke dalam Gmabr 5. Gambar 5 menunjukan bahwa rata-rata bobot sisi dalam bentuk opersen terhahadap radius. Algoritma yang bersofat netral alan memilih sisi secara random dan enghasilkan rata-rata bobot sisi sebesar 50%. Sedangkan algoritma lain akan memilih sisi dengan bobot terkecil sehingga akan memberikan efek yang signifikan terhadap *file* gambar true-color dengan audio waveform.

Hasil lain yang diperoleh melalui pengujian adalah bahwa gambar true-color akan membutuhkan lebih banyak waktu daripada format data yang lain. Misal untuk menyisipkan data 6KB heuristic akan dilalui selama 12 detik Akibatnya, menimbulkan perbedaan nilai sampel yang tinggi. Masalah lain yang timbul pada saat pengujian adalah ditemukan fakta bahwa dengan menggunakan atruktur data berupa list terbatas untuk nilai sample akan membutuhkan waktu $O(|S'|^2)$ dengan S' merupakan nilai sampel yang ada pada graf.

3.7 Steganalysis

Jumlah dari verteks yang tidak memiliki kecocokan akan dianggap sama dengan jumlah *first-order statistic*. Pengujian menunjukan bahwa kecocokan akan banyak diemukan pada *cover-data* yang natural. Sehingga pendekatan

yang digunakan pada algoritma ini sampai sekarang tidak dapat dideteksi oleh test steganalis yang menggunakan *first-order statistic* seperti serangan X^2 . [4]

Steganalisis lain diuji coba dengan menjalankan skema *blind* steganalisis maupun pengukuran *blockiness*. Hasilnya, algoritma tahan terhdaap serangan-serangan demikian.

4. Pengujian Steghide

Pengujian terhadap Steghide akan dilakukan melalui tiga jeni pengujian. Pertama, Steghide diuji apakah *stego-data* yang dihasilkan telah memenuhi kriteria. Kedua, Steghide diuji langsung dengan program yang mampu malakukan steganalysis seperti Stegdetect. Ketiga, Steghide diuji melalui program kecil CDiff yang dibuat penulis untuk mengetahui posisi bit yang berubah pada *cover-data*. [4]

4.1 Pengujian terhadap Kriteria Steganografi

Penyembunyian pesan rahasia ke dalam *stego-data* akan mengubah kualitas data tersebut. Kriteria yang harus diperhatikan dalam hasil penyembunyian data adalah [1]:

- *Fidelity*.
  Mutu *cover-data* tidak jauh berubah. Setelah penambahan pesan rahasia, *stego-data* masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam *stego-data* tersebut terdapat pesan rahasia.
- *Robustness.*
  Pesan yang disembunyikan harus tahan (robust) terhadap berbagai operasi manipulasi yang dilakukan pada *stego-data*, seperti pengubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan *cropping*, enkripsi, dan sebagainya. Bila pada citra penampung dilakukan operasi-operasi pengolahan citra tersebut, maka pesan yang disembunyikan seharusnya tidak rusak (tetap valid jika diekstraksi kembali).
- *Recovery.*
  Data yang disembunyikan harus dapat diungkapkan kembali (reveal). Karena tujuan stenografi adalah penyembunyian informasi, maka sewaktu-waktu pesan rahasia di dalam *stego-data* harus dapat diambil kembali untuk digunakan lebih lanjut.

4.1.1 Pengujian terhadap *Fidelity*

Pengujian terhadap *fidelity* dapat dianggap sebagai pengujian apakah Steghide tahan terhadap *detection attack. Detection attack* merupakan serangan yang bertujuan untuk mengetahui keberadaan pesan rahasia pada suatu *file*. *Detection attack* terdiri dari RQP *attack*, RS *attack*, serta *known-cover attack*. Pengujian terhadap *fidelity* ini lebih khusus pada *known-cover attack* yang artinya pihak penyerang diasumsikan mempunyai gambar asli sesuai *cover-data* yang digunakan.
Pengujian dilakukan dalam tiga tahap pada berbagai format *file* yaitu JPG, BMP, dan WAV.

Salah satu contoh perintah yang dilakukan dalam proses pengujian adalah sebagai berikut:

```
steghide   embed –ic PH034251.JPG –ef
daftarpustaka.txt –sf tesF2.JPG –verbose
```

Maksud dari perintah di atas adalah bahwa *file* teks daftarpustaka.txt disembunyikan ke dalam *cover-data* berupa *file* gambar PH034251.JPG.

--verbose digunakan agar Steghide menampilkan informasi proses secara detail. Keluaran yang diperoleh melalui perintah tersebut adalah :

```
reading secret file
“daftarpustaka.txt”…done
reading cover file “PH034251.JPG”…done
creating the graph… 148 sample values,
2119 vertices, 2138251 edges
executing Static Minimum Degree
Construction heuristic…. 1000.0% <1.0>
done
writing stego file “tesF2.JPG”… done
```

Keluaran tersebut menggambarkan graf yang dibentuk, proses pembacaan, serta penulisan *file*. Proses steganografi dibantu dengan proses enkripsi rijndael-128 dengan mode CBC serta proses kompresi terhadap *file* teks. Sedangkan kunci yang digunakan memiliki panjang 4 karakter.

Uji 1 : Membandingkan tampilan visual atau audio *stego-data* dan *cover-data*

Perbandingan tampilan gambar pada antara *Stego-data* (Gambar 6) dengan *Cover-data* (Gambar 7) adalah :

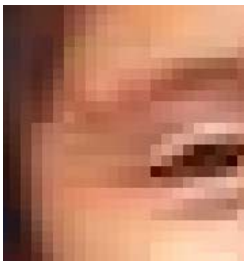
Gambar 6 *Stego-data*

Gambar 7 *Cover-data*

Secara kasat mata, kedua gambar tersebut tidak memiliki perbedaan.

Apabila gambar pada *Stego-data* dan *cover-data* dibesar sampai 8x pada area yang sama, hasilnya akan terlihat seperti pada gambar 8 dan gambar 9.

Melalui gambar 8 dan 9, terlihat bahwa perbedaan antara kedua gambar masih sulit untuk dikenali.

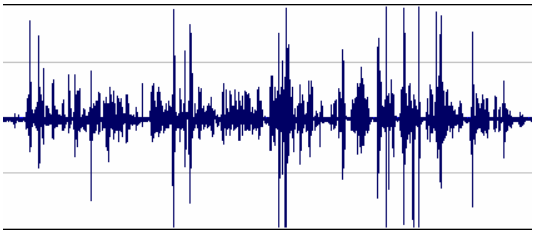


Gambar 8 *Stego-data* 8x

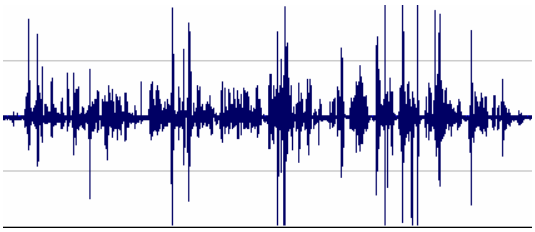
Gambar 9 Cover-data 8x

Percobaan untuk uji 1 juga telah dilakukan terhadap *file* WAV. *Cover-data* masih berupa *file* gambar JPG sedangkan *stego-data* berupa *file* WAV. Hasilnya, tetap tidak terdapat perbedaan secara visual antara *stego-data* dan *cover-data*.

Sedangkan pada percobaan antar *file* WAV, yaitu proklamasiRI.wav sebagai *cover-data* dan vibrate.wav sebagai pesan rahasia, didapatkan hasil yang sama. Bahwa tidak terdapat perubahan yang dapat ditangkap telinga manusia antara *stego-data* dan *cover-data* Keluaran suara *stego-data* sama persis dengan *cover-data*. Hasil percobaan berupa gambar sinyal suara pada *file* wav yang diujikan terdapat pada gambar 10 dan 11. Pada gambar sinyal suara pun, sulit untuk mengenali perbedaan antara *stego-data* dan *cover-data*.



Gambar 10 *Cover-data* proklamasiRI.wav



Gambar 11 *Stego-data* tesF7.wav

Oleh karena itu, berdasarkan percobaan yang telah dilakukan, dapat disimpulkan bahwa Steghide telah lulus uji 1. Artinya, melalui tampilan visual dan audio, keberadaan pesan rahasia pada *steg-data* sulit dikenali.

Uji 2 : Membandingkan antara ukuran *file stego-data* dengan *cover-data*

Setelah menguji *fidelity* pada tampilan visual dan audio *Stego-data* tesF2.JPG memiliki ukuran *file* sebesar 49,168 bytes. Sedangkan *cover-data* memiliki ukuran sebesar 48,558 bytes. Hal ini menunjukan bahwa terdapat perubahan atau manipulasi terhadap *cover-data* sehingga terdapat pertambahan sebesar 610 bytes pada *stego-data*. Selama pihak lain memiliki *file* gambar yang asli maka hal ini akan menimbulkan kecurigaan.

Akan tetapi, perbedaan besar ukuran *file* ini tidak berlaku bagi *file* tesF7.wav dan proklamasiRI.wav. tesF7.wav memiliki ukuran *file* yang sama dengan proklamasiRI.wav yaitu 217,672 bytes. Hal serupa terjadi untuk *cover-data* berformat bmp. Ukuran *stego-data* dan *cover-data* sama besar.

Hasil uji 2 secara lengkap terdapat pada tabel 2.

Tabel 2 Hasil uji 2

Cover-data	*Pesan rahasia*	Ukuran *Stego-data*	Ukuran *cover-data*

wav	jpg	217,672	217,672
wav	wav	217,672	217,672
jpg	txt	49,168	48,558
jpg	jpg	49,288	48,558
jpg	wav	49,320	48,558
bmp	jpg	1,368,890	1,368,890

Melalui tabel tersebut dapat terlihat bahwa perbedaan ukuran *file* hanya terjadi pada kasus ketika *cover-data* yang digunakan berformat jpg. Oleh karena itu dapat dsimpulkan bahwa Steghide tidak lulus uji 2 hanya untuk *cover-data* berformat jpg.

Jpeg merupakan *file* gambar terkompresi. Setelah dilakukan proses *stego-system* dan kemudian hasil proses dituliskan pada *file,* Steghide perlu melakukan kompresi sehingga didapatkan ukuran *file* yang mendekati *file* aslinya, sesuai karakteristik dari *file* jpg. Hanya saja, proses kompresi tidak dilakukan secara sempurna sehingga terjadi perubahan ukuran *file*.

Uji 3 : Membandingkan ukuran *file* terkompresi

Teknik ini dikenalkan oleh Hansman pada [3]. *File stego-data* dan *cover-data* dikompresi melalui program kompresi seperti WinRar atau WinZip kemudian ukuran *file* hasil kompresi dibandingkan.

Apabila *Stego-data* memiliki rasio kompresi yang berbeda dengan *cover-data* maka keberadaan pesan rahasia pada suatu *file* dapat dikenali.

Uji 3 dilakukan untuk mengimplementasikan teknik Hansman tersebut. Program kompresi yang digunakan adalah WinZip. Dan hasil yang diperoleh sebagai berikut (tabel 3) :

Tabel 3 Hasil uji 3

Jenis *file*	*Cover-data* setelah kompresi	*Stego-data* setelah kompresi
wav	93,793	97,815
bmp	25,767	66,810
jpg	48,629	49,129

Melalui tabel 3 terlihat bahwa terdapat perbedaan antara ukuran *file* terkompresi pada *stego-data* dengan *cover data*. Padahal ukuran *file cover-data* dan *stego-data* berformat wav maupun bmp sebelum kompresi adalah sama. Oleh karena itu Steghide tidak lulus uji 3. Sehingga dapat disimpulkan bahwa melalui teknik Hansman, keberadaan pesan rahasia pada *stego-data* dapat diketahui.

4.1.2 Pengujian terhadap *robustness* dan *recovery*

Pengujian terhadap *robustness* dilakukan bersamaan dengan pengujian terhadap *recovery*. Karena kedua hal tersebut saling berkaitan. Proses dilakukan dengan melakukan tiga jenis manipulasi. *File* dengan format yang berbeda dikenakan proses manipulasi yang berbeda pula. Pengujian terbatas dilakukan terhadap *file* berformat jpg dan wav.

Uji 4: Pengujian terhadap *file* jpeg.

Pengujian dilakukan terhadap *file* tesF2.jpg. Proses ekstraksi *file* sebelum dikenakan manipulasi akan menghasilkan *file* teks bernama daftarpustaka2.txt. *File* teks tersebut seharusnya memiliki isi yang sama dengan *file* daftarpustaka.txt.
Berikut perintah yang digunakan pada steghide untuk melakukan ekstraksi :

```
steghide extract –sf tesF2.jpg –xf
daftarpustaka2.txt -v
```

Maksud perintah tersebut adalah *file* tesF2.jpg diektrak untuk mendapatkan pesan rahasia. Pesan tersebut disimpan ke dalam sebuah *file* teks bernama daftarpustaka2.txt

-v digunakan agar Steghide menampilkan informasi proses secara detail. Keluaran yang diperoleh melalui perintah tersebut adalah :

```
reading stego files “tesF2.jpg”…done
extracting data …done
checking crc32 checksum… ok
writing extracted data to
“daftarpustaka2.txt”…done
```

Berikut isi *file* daftarpustaka2.txt yang sama dengan pesan rahasia yang disembunyikan pada *file* tesF2.jpg:

1. Ross J. Anderson. Stretching the Limits of Steganography. In Ross J. Anderson, editor, Information Hiding, First International Workshop, volume 1174 of Lecture
Notes in Computer Science, pages 39–48. Springer, 1996.
2. Rainer B¨ohme and Andreas Westfeld. Exploiting Preserved Statistics for Steganalysis.
In Jessica J. Fridrich, editor, Information Hiding, 6th International
Workshop, volume 3200 of Lecture Notes in Computer Science. Springer, 2004.
3. Elke Franz. Steganography Preserving Statistical Properties. In F.A.P. Petitcolas, editor, Information Hiding, 5th International Workshop, volume 2578 of Lecture
Notes in Computer Science, pages 278–294. Springer, 2003.
4. Jessica Fridrich, Miroslav Goljan, and David Soukal. Higher–order statistical steganalysis
of palette images. In Proceedings of the Electronic Imaging SPIE Santa
Clara, CA, January 2003, pages 178–190, 2003.

Hasil uji tersebut menunjukan bahwa *stego-data* dapat di*recover*.
Proses pengujian dilanjutkan dengan melakukan manipulasi terhadap *stego-data.*

Uji 4.a : mengubah nilai kontras *stego-data*.

Gambar 12 tesF2.jpg sebelum dimapulasi

Gambar 13 tesF2.jpg setelah dimanipulasi

File hasil manipulasi diekstraksi. Hasil yang diperoleh berupa pesan *error* sebagai berikut :

```
extracting data…steghide : could not
extract any data with that passphrase
```

Kemungkinan pesan rahasia yang terdapat pada *file* hasil manipulasi rusak sehingga tidak dapat diekstraksi. Dengan demikian dapat disimpulkan bahwa Steghide tidak memenuhi kriteria *robustness* pada tahap uji 4.a.

Uji 4.b : merotasi *stego-data*

Gambar 14 tesF2.jpg sebelum rotasi

Gambar 15 tesF2.jpg setelah rotasi

File hasil rotasi kemudian diekstraksi. Hasil yang diperoleh sama seperti hasil uji 4.a, berupa pesan *error* sebagai berikut :

```
extracting data…steghide : could not
extract any data with that passphrase
```

Kemungkinan pesan rahasia yang terdapat pada *file* hasil rotasi rusak sehingga tidak dapat diekstraksi dengan kunci semula. Dengan demikian dapat disimpulkan bahwa Steghide tidak memenuhi kriteria *robustness* pada tahap uji 4.b.

Uji 4.c : menambah gambar kotak pada *stego-data*

Pada pengujian 4.c, penambahan kotak dilakukan melalui program Paint[tm].

Gambar 16 tesF2.jpg sebelum diberi kotak

Gambar 17 tesF2.jpg Setelah diberi kotak

File hasil manipulasi diekstraksi. Hasil yang diperoleh masih berupa pesan *error* sebagai berikut :

```
extracting data…steghide : could not
extract any data with that passphrase
```

Pesan rahasia yang terdapat pada *file* hasil manipulasi rusak sehingga tidak dapat diekstraksi. Dengan demikian dapat disimpulkan bahwa Steghide tidak memenuhi kriteria *robustness* pada tahap uji 4.c.

Karena tidak memenuhi ketiga jenis manipulasi, dapat disimpulkan bahwa untuk gambar jpeg, Steghide tidak memenuhi kriteria *robustness*

Uji 5: Pengujian terhadap *file* wav.

Pengujian dilakukan terhadap *file* tesF7.wav. Proses ekstraksi *file* sebelum dikenakan manipulasi akan menghasilkan *file* teks bernama README2.txt. *File* teks tersebut memiliki isi yang sama dengan *file* README.txt.
Perintah yang digunakan pada steghide untuk melakukan ekstraksi *file* wav:

```
steghide extract –sf tesF7.jpg –xf README2.txt -v
```

Keluaran yang diperoleh melalui perintah tersebut adalah :

```
reading stego files "tesF7.wav"…done
extracting data …done
checking crc32 checksum… ok
writing extracted data to "README2.txt"…done
```

Cuplikan isi *file* README2.txt yang sesuai dengan pesan rahasia pada *file* tesF7.wav adalah:

Introduction :
==========

Steghide is a steganography program that is able to hide data in various kinds of image- and audio-files. The color- respectivly sample-frequencies are not changed thus making the embedding resistant against first-order statistical tests.

The current version of steghide is 0.5.1

Features:
*) compression of embedded data
*) encryption of embedded data
*) embedding of a checksum to verify the integrity of the extraced data
*) support for JPEG, BMP, WAV and AU files

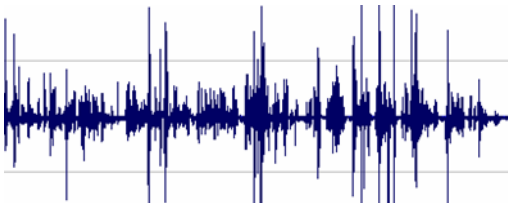
Steganography :
==========

Steganography literally means covered writing. Its goal is to hide the fact large) cover file and embedding the (rather ..

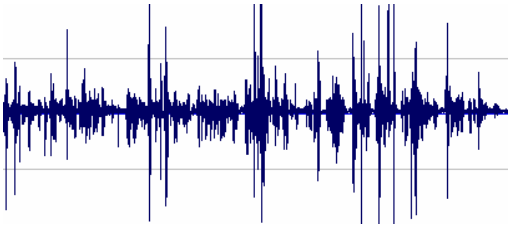
Hasil uji tersebut menunjukan bahwa *stego-data* dapat di*recover*.

Selanjutnya proses manipulasi *file* wav dilanjutkan dengan bantuan program Sony Sound Forge versi 7.0

Uji 5.a : manipulasi dengan metode Flip/Rotate



Gambar 18 Gambar sinyal sebelum manipulasi



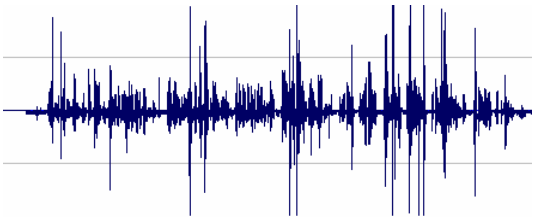
Gambar 19 Gambar sinyal setelah manipulasi

Ektraksi *file* wav yang telah dimanipulasi menghasilkan pesan *error* :

```
steghide: error -5 while calling zlib's uncompress
```

Kemungkinan penyebabnya adalah terdapat kerusakan pada *file* ataupun pesan rahasia.

Uji 5.b : manipulasi dengan penambahan *silence*

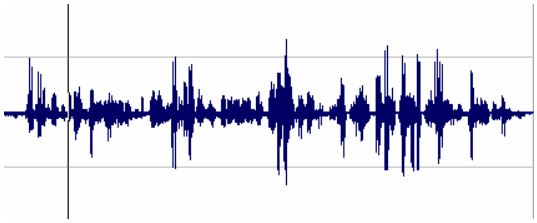


Gambar 20 Gambar sinyal setelah penambahan silence

Ektraksi *file* wav yang telah dimanipulasi menghasilkan pesan *error* :

```
extracting data…steghide : could not extract any data with that passphrase
```

Uji 5.c : memberi efek *echo*



Gambar 21 Gambar sinyal dengan efek echo

Ektraksi *file* wav yang telah diberi efek *echo* menghasilkan pesan *error* :

```
extracting data…steghide : could not
extract any data with that passphrase
```

Hal ini menambah kepastian bahwa Steghide tidak memenuhi kriteria robustness baik terhadap *file* gambar jpeg maupun *file* wav

Setiap manipulasi yang dilakukan terhadap *stego-data* mengakibatkan pesan rahasia tidak dapat diekstraksi. Manipulasi yang dilakukan baik ringan maupun kompleks sangat mempengaruhi proses ekstraksi. Hal ini disebabkan posisi pesan rahasia pada *stego-data* yang tersebar dan kompleksnya proses penyembunyian pesan pada Steghide sehingga sangat bergantung pada keberadaan informasi tambahan yang bisa jadi rusak ketika dikenakan proses manipulasi.

4.1 Pengujian dengan *Stegdetect*

Stegdetect merupakan program hasil steganalisis terhadap *stego-system* pada jsteg, outguess, jphide, dan invisible [7]. Stegdetect menggunakan tes statistik untuk menganalisis suatu gambar jpeg. Analisis dilakukan untuk mengetahui keberadaan suatu pesan rahasia pada suatu gambar serta informasi program yang melakukan penyembunyian pesna pada gambar tersebut.

Sesuai dengan tujuan awal Steghide yaitu *stego-system* yang tahan terhadap serangan *first-order statistical test,* pesan pada *stego-data* Steghide seharusnya tidak dapat terdeteksi oleh Stegdetect. Pengujian ini dilakukan untuk membuktikan hal tersebut.

Pengujian dilakukan menggunakan program stegdetect yang berbasis *command line*. Perintah yang digunakan adalah :

```
stegdetect tesF2.jpg
```

Dengan perintah tersebut, stegdetect akan mengetes keberadaan pesan pada tesF2.jpg. Keluaran perintah tersebut berupa pesan :

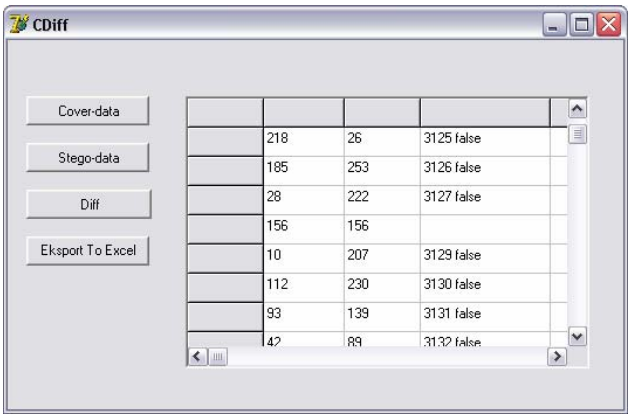
```
tesF2.jpg : negative
```

Dengan demikian terbukti bahwa *stego-data* Steghide dengan format jpeg tahan terhadap serangan secara statistic oleh Stegdetect.

4.2 Pengujian dengan Program Kecil CDiff

CDiff dibangun penulis dengan kakas Borland Delphi 7.0 Program ini dapat menampilkan perbedaan bit antara kedua buah *file* yaitu cover-data dengan stego-data. Informasi yang ditampilkan berupa bit awal, bit hasil perubahan, dan posisi bit. Diharapkan program kecil ini nantinya dapat membantu steganalisis terhadap Steghide.

Pengujian dilakukan terhadap *file* tesF2.jpg. *File* tesF2.jpg dibandingkan dengan *cover-data* PH034251.jpg. Tujuan dari pengujian ini hanya untuk mengetahui posisi bit-bit yang berubah pada *stego-data*.



Gambar 21 Tampilan CDiff

Kolom pertama menandakan nilai byte pada *cover-data,* kolom kedua menandakan nilai byte pada *stego-data*. Sedangkan kolom ketiga akan memberi informasi berupa posisi bit dan *false* jika antara byte *cover-data* dan *stego-data* memiliki nilai yang berbeda.

Hasil pengujian menandakan bahwa perubahan nilai *byte* terjadi mulai pada byte ke-21.Nilai byte diatur semikian rupa secara acak sehingga

stego-data tidak menampilkan distorsi secara visual maupun audio.

5. Kesimpulan

1. Steghide menggunakan algoritma *graph-theoretic approach* yang tahan terhadap serangan *first-order statistical test*
2. Steghide tidak memenuhi kriteria *fidelity* karena keberadaan pesan dapat dianalis melalui teknik Hansman yang memanfaatkan program kompresi untuk mengetes perbedaan ukuran *file stego-data* dan *cover-data* setelah kompresi. Akan tetapi secara visual amupun audio, perbedaan antara *stego-data* dan *cover-data* sulit dibedakan.
3. Steghide tidak memenuhi criteria *robustness* karena manipulasi yang dilakukan terhadap *stego-data* mengakibatkan pesan rahasia tidak dapat diekstraksi.
4. Steghide memenuhi kriteria *recovery*. Pesan dapat diekstraksi dari *stego-data* dan seuai dengan pesan yang dimaksud pengirim.
5. Keberadaan pesan pada *stego-data* Steghide tidak dapat dideteksi oleh Stegdetect
6. Posisi bit yang berubah pada *stego-data* tersebar. Hal ini diketahui melalui program kecil CDiff

6. Referensi

[1] Ary Ary, Steganografi, 2006
[2] Cachin Christian, *Digital Steganography*
[3] C Raphael. *Steganaysis of Random LSB Insertion Using Discrete Logarithms*
[4] Hetzl Stefan, Mutzel Petra. *A Graph Theoretic Approach To Steganogrphy*
[5] Hetzl Stefan, *Steghide Manual*
[6] Munir, Rinaldi. (2006). Diktat Kuliah IF5054 Kriptografi. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.
[7] Niels Provos, *Stegdetect Manual*