TUGAS AKHIR –SM141501

KEAMANAN PESAN RAHASIA PADA STEGANOGRAFI CITRA MENGGUNAKAN KODE HAMMING (15,11)

RIDWANUL FATA
NRP 06111240007002

Dosen Pembimbing
Dr. Darmaji S.Si, MT

DEPARTEMEN MATEMATIKA
Fakultas Matematika Komputasi dan Sains Data
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT -SM141501

SECRET MESSAGE SECURITY ON IMAGE STEGANOGRAPHY USING (15,11) HAMMING CODE

RIDWANUL FATA
NRP 061112 4000 7002

Supervisor
Dr. Darmaji S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Computational Mathematics and Data Science
Sepuluh Nopember Institute of Technology
Surabaya 2018

LEMBAR PENGESAHAN

KEAMANAN PESAN RAHASIA PADA STEGANOGRAFI CITRA MENGGUNAKAN KODE HAMMING (15,11)

SECRET MESSAGE SECURITY ON IMAGE STEGANOGRAPHY USING (15,11) HAMMING CODE

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Sains
Pada bidang studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
RIDWANUL FATA
NRP.06 11240007002

Menyetujui,
Dosen Pembimbing

Dr. Darmaji, S.Si, MT
NIP.19691015 199412 1 001

Mengetahui,
Ketua Departemen Matematika
FMKSD ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003
Surabaya, 6 Agustus 2018

KEAMANAN PESAN RAHASIA PADA STEGANOGRAFI CITRA MENGGUNAKAN KODE HAMMING (15,11)

Nama Mahasiswa : Ridwanul Fata
NRP : 06111240007002
Jurusan : Matematika
Dosen Pembimbing : Dr. Darmaji,S.Si, MT

Abstrak

Abstrak—Steganografi merupakan sebuah disiplin ilmu yang mempelajari tentang sebuah teknik menyembunyikan pesan rahasia pada sebuah media digital sehingga seseorang tidak menyadari keberadaan pesan rahasia tersebut dalam sebuah media digital. Dalam proses ini media digital yang digunakan adalah dari citra ke sebuah citra. Untuk proses menyisipkan citra ke sebuah citra akan digunakan metode LSB (*Least Significant Bit*).

Oleh karena itu akan diimplementasikan sebuah algoritma steganografi citra yang aman berdasarkan konsep Kode Hamming (15,11). Pada pengujian yang dilakukan nilai PSNR untuk citra beresolusi 225x225 dan berukuran 52.4 Kb dengan citra pesan rahasia berukuran 41 Kb diperoleh nilai 52.23 dB dan nilai PSNR akan semakin besar jika resolusi dan ukuran dari sebuah citra medium semakin besar. Selain itu dalam salah satu uji coba steganalisis , software yang dikhususkan untuk menganalisa steganografi yang menggunakan metode LSB tidak dapat membaca sebuah citra medium yang mengandung steganografi didalamnya. Hal itu menunjukkan bahwa menyembunyikan pesan rahasia pada sebuah media digital menggunakan konsep Kode Hamming masih aman**.**

***Kata Kunci*—** Steganografi, Least Significant Bit (LSB), Kode Hamming (15,11), *Encoder Parity Check , Decoder Syndroms*.

“Halaman ini sengaja dikosongkan”

SECRET MESSAGE SECURITY ON IMAGE STEGANOGRAPHY USING (15,11) HAMMING CODE

Name	***: Ridwanul Fata***
NRP	***: 06111240007002***
Department	***: Mathematics***
Supervisor	***: Dr. Darmaji, S.Si, MT***

Abstract

Steganography is a discipline that study a technique to hide a secret message in a digital media so people would not realize the existance of that message in digital media. In this process, the digital media used is from an image to an image. For inserting an image to an image, LSB (Least Significant Bit) method is used.

For that process, a safe image steganography algorithm will be implemented based on Hamming Code (15,11). At the testing process, the PSNR value for image that has resolution of 225x225 and 52.4 kb in size with a 41Kb secret image, resulting 52.23 DB. The bigger resolution and size of an image, the bigger PSNR value would be. Beside that, in one of steganalysis testing, the specialized software used analyze steganography using LSB method could not read a medium image containing steganography. That shows us that hiding a secret message in a digital media using Hamming Code is still safe..

Keywords: *Steganography, Least Significant Bit,Hamming Code (15,11)Parity Check,Syndrom.*

“Halaman ini sengaja dikosongkan”

KATA PENGANTAR

Segala Puji bagi Allah SWT yang telah memberikan karunia, rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: **"Keamanan pesan rahasia pada Steganografi Citra menggunakan Kode Hamming (15,11) "** yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Studi S-1 pada Jurusan Matematika Fakultas Matematika Komputasi dan Sains Data Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan berkat kerjasama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis mengucapkan terima kasih kepada:

1. Dr. Darmaji , S.Si, MT selaku dosen pembimbing yang senantiasa membimbing dengan sabar dan memberikan kritik dan saran dalam penyusunan Tugas Akhir ini.
2. Prof. Dr. Imam Mukhlas, S.Si, M.T selaku Ketua Jurusan Matematika.
3. Ibu Soleha, M. Si selaku Dosen Wali.
4. Dr. Imam Mukhlas S.Si, M.T, Drs. Qomar Baihaqi, M.Si dan Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku dosen penguji Tugas Akhir ini.
5. Drs. Iis Herisman M.Si selaku Koordinator Tugas Akhir.
6. Seluruh jajaran dosen dan staf jurusan Matematika ITS.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Juli 2018

Penulis

special thanks to

Selama proses pembuatan Tugas Akhir ini, banyak pihak yang telah memberikan bantuan dan dukungan untuk penulis. Penulis mengucapkan terima kasih dan apresisasi secara khusus kepada:

1. Orang tua penulis bapak Atqiya Zaini dan ibu Siti Rofiatin yang senantiasa dengan ikhlas memberikan semangat, perhatian, kasih sayang, doa, motivasi dan nasihat yang sangat berarti bagi penulis.
2. Si Mbah dan Paman di Kediri, Jombang dan Surabaya yang menjadi salah satu motivasi bagi penulis untuk segera menyelesaikan Tugas Akhir ini.
3. Pondok Pesantren Salafiyah Sentot Alibasya dan MA Ja-alHaq Bengkulu almamater yang telah mengantarkan saya sampai bisa di kampus perjuangan ini.
4. Tahta Dari Timur, Muhammad Zufar dan Alam Arraad Stone yang telah meluangkan waktu membantu penulis menyelesaikan Tugas Akhir ini. Semoga amal ibadah sampean semua bisa membantu mencapai apa yang sampean cita-citakan dan menjadi amal jariah yang tidak akan pernah terputus pahalanya. Aamiin .
5. Teman-teman D12 CSS MoRA ITS, keluarga pertama di Surabaya insya Allah sampe tua esok .
6. Teman-teman Omah Karahayon selama tiga periode yang tidak bisa saya sebutkan satu persatu, terima kasih telah membersamai dalam kondisi apapun.
7. Keluarga besar CSS MoRA ITS sebagai tempat kembali kemanapun saya pergi.
8. Sainstek HIMATIKA ITS dua periode. Terima kasih kesempatan bergabung berama kalian. Semoga kita sukses selalu. Aamiin.
9. Keluarga besar HIMATIKA ITS yang telah banyak membantu saya survive di kampus perjuangan sampai saat ini.

10. UKM PSHT ITS tempat menimba olahraga fisik selama kurang lebih 2 semester
11. PMII Sepuluh Nopember ITS tempat menimba ilmu pergerakan di kampus perjuangan.
12. Futsal MATEMATIKA ITS dan Futsal CSS MoRA ITS yang telah sedia memberikan kesempatan belajar dan bertanding dalam atmosfer pertandingan yang besar. Insya Allah IFC 2019 Futsal MATEMATIKA ITS JUARA !
13. Keluraga besar warung kopi taman baca, tempat pelarian terbaik selama kuliah, mengerjakan Tugas Akhir dan tempat dimana segala kebaikan dan kebahagiaan terlahir dari sana. Semoga warung kopi taman baca selalu menjadi tempat terbaik untuk diskusi para aktivis-aktivis kampus perjuangan.

Tentu saja masih banyak pihak lain yang turut andil dalam penyelesaian tugas akhir ini yang tidak bisa penulis sebutkan satu persatu. Semoga Allah membalas dengan balasan yang lebih baik bagi semua pihak yang telah membantu penulis. *Amin ya rabbal 'alamin*.

“Halaman ini sengaja dikosongkan”

DAFTAR ISI

DAFTAR GAMBAR

“Halaman ini sengaja dikosongkan”

DAFTAR TABEL

“Halaman ini sengaja dikosongkan”

BAB I
PENDAHULUAN

Latar Belakang

Berdasarkan kemajuan dan perkembangan teknologi tentang teknologi informasi yang sangat pesat di bidang transfer data, kebanyakan orang menjadi takut jika data yang mereka punya diretas oleh seorang *hacker*[2]. Seseorang mungkin tidak bisa menghindar dari retasan *hacker*, namun seseorang dapat menggunakan steganografi untuk menyembunyikan data atau informasi yang bersifat rahasia.

Steganografi adalah sebuah disiplin ilmu yang mempelajari tentang teknik menyembunyikan pesan rahasia hingga seseorang tidak menyadari keberadaan pesan rahasia tersebut pada sebuah media. Steganografi dibagi menjadi dua bagian, yaitu steganografi konvensional dan steganografi modern. Penggunaan steganografi konvensional dan modern lebih berdasarkan kepada masa dimana steganografi tersebut digunakan. Semua teknik steganografi baik modern maupun konvensional, keduanya sama sama berusaha merahasiakan komunikasi dengan cara menyembunyikan atau mengkamuflase sebuah pesan. Maka sesungguhnya prinsip dasar dalam steganografi lebih ditekankan pada bagaimana informasi itu dirahasiakan, jadi bukan terletak pada datanya.

Steganografi konvensional mulai dikenal sejak tahun 440 sebelum masehi. Dalam buku seorang sejarawan Yunani, Herodotus menuliskan bagaimana steganografi sangat berperan dalam sebuah peperangan. Dalam buku tersebut disebutkan bahwa Histiacus menggunakan teknik steganografi untuk mengirimkan sebuah pesan rahasia dengan cara mentattokan kepala seorang budak yang sudah dipangkas habis rambut kepalanya. Tattoo tersebut berisi sebuah pesan rahasia. Untuk mengirimkan sebuah pesan tersebut menunggu rambut kepala budak tersebut tumbuh dulu sehingga tidak ada kecurigaan

terhadap pesan rahasianya. Setelah itu barulah si budak tersebut dikirimkan dan untuk melihat pesan rahasia tersebut maka penerima pesan rahasia harus memangkas rambut budak tersebut.

Sedangkan steganografi modern data rahasia disematkan ke dalam *citra medium* seperti text, gambar, audio atau video. Seiring dengan berkembangnya steganografi, banyak program software tentang analisis steganografi yang bagus dan hal hal yang tidak diinginkan pun terjadi. Analisis steganografi tersebut disalahgunakan oleh orang yang tidak berwenang untuk mendapatkan kembali isi dari informasi rahasia yang telah disematkan kedalam *carrier file*.

Pada [3] menggunakan 9 sampel barisan video yang tidak dikompresi dan sebelum proses penyisipan pesan rahasia, Kode Hamming (7,4) digunakan untuk meng-*encode* pesan rahasia tersebut supaya lebih aman. Skema steganografi yang digunakan masih berupa *citra* dan format citra yang digunakan pada video adalah format *YUV*.

Oleh karena itu, pada proposal penelitian Tugas Akhir kali ini penulis akan mengimplementasikan sebuah algoritma steganografi citra yang aman berdasarkan prinsip Kode Hamming. Sebuah citra dengan ukuran yang lebih besar akan digunakan untuk *cover data* dan sebuah citra biner yang digunakan sebagai pesan rahasia. Kemudian pesan rahasia di-*encode* dengan mengaplikasikan Kode Hamming (15,11) sebelum proses penyisipan, supaya lebih aman. Setelah itu hasil dari pesan yang sudah di-*encode* menggunakan Kode Hamming (15,11), akan diproses menggunakan metode *LSB (Least Significant Bit)*. Setelah itu citra pesan sudah siap disisipkan ke cover citra tersebut. Salah satu cara untuk mengukur kualitas suatu steganografi itu baik atau buruk adalah ketangguhan *(robustness)*, dan untuk menguji ketangguhan sebuah stego citra diperlukan sebuah pengujian untuk membandingkan antara nilai maksimum dari signal citra yang diukur dengan besarnya *noise* yang berpengaruh pada signal pada citra tersebut. Proses membandingkan nilai maksimum tersebut biasa disebut dengan

Peak Signal to Noise Ratio (PSNR). Mengenai kualitas signal, *PSNR* dari stego citra, semakin besar nilai *PSNR* yang dihasilkan semakin baik kualitas dari stego citra yang dihasilkan. *PSNR* sering dinyatakan dalam skala logaritmik dalam desibel (dB). Nilai *PSNR* dibawah 30 dB mengindikasikan bahwa kualitas suatu stego medium tersebut relatif rendah dan sebaliknya jika nilainya diatas 40 dB mengindikasikan bahwa kualitas stego citra tersebut relatif baik karena kualitas gambar stego citra memiliki kemiripan mendekati citra asal.

Rumusan Masalah

Adapun masalah yang diselesaikan dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara menyisipkan citra pesan ke sebuah citra medium menggunakan Kode Hamming (15,11) ?
2. Bagaimana tingkat keamanan stego citra yang dihasilkan ketika diuji coba dengan software steganalisis ?

Batasan Masalah

Batasan Masalah yang akan dibahas dalam penelitian Tugas Akhir ini adalah sebagai berikut :

1. Algoritma pengkodean yang dipakai adalah Kode Hamming (15,11)
2. Encoder yang digunakan adalah *Parity Check*
3. Decoder yang digunakan adalah *Syndrom*
4. Citra yang digunakan adalah citra greyscale
5. Bahasa pemrograman yang digunakan adalah bahasa Python

Tujuan

Tujuan yang diharapkan dari penulisan Tugas Akhir ini adalah :

1. Melakukan pengamanan data melalui penyisipan pesan rahasia ke dalam file citra

2. Melakukan penerapan modifikasi Kode Hamming sebagai encoder pada proses penyisipan pesan rahasia
3. Mengimplementasikan program steganografi citra menggunakan Kode Hamming (15,11)

Manfaat

Manfaat dari hasil penulisan Tugas Akhir ini adalah sebagai berikut :

1. Memberikan manfaat pada bidang *data hiding* terutama pada bidang steganografi yang dapat digunakan untuk proses pengamanan informasi atau data melalui media citra
2. Mengahasilkan program steganografi citra dengan memanfaatkan Kode Hamming (15,11)

Sistematika Penulisan Tugas Akhir

Sistematika penulisan didalam Tugas Akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang pembuatan Tugas Akhir, rumusan dan batasan masalah yang dihadapi dalam penelitian Tugas Akhir, tujuan dan manfaat pembuatan Tugas Akhir serta sistematika penulisan Tugas Akhir.

BAB II STUDI LITERATUR

Bab ini menjelaskan tentang penelitian sebelumnya dan bagaimana penulis mengambil referensi dari berbagai macam jurnal, buku, dan penelitian yang berkaitan dengan Tugas Akhir penulis. Serta akan dipelajari tentang dasar dasar teori yang digunakan dalam Kode Linear, *Parity Check*, Kode Hamming (15,11), pendefinisian Kode Linear dan Kode Hamming serta

metode metode encoding dan decoding (*Encoder Parity Check, Decoder Syndrom*).

BAB III METODOLOGI PENELITIAN

Bab ini berisi metodologi atau urutan pengerjaan yang dilakukan dalam menyelesaikan Tugas Akhir. Dimana dalam metodologi penelitian akan dijelaskan secara terurut dalam pengerjaan Tugas Akhir ini.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan analisis, perancangan gambaran umum sistem serta menggunakan hasil dari dasar teori pada bab sebelumnya akan dirancang sebuah system Kode Linear dan meng-encode pesan rahasia menggunakan Kode Hamming (15, 11) ke dalam media citra.

BAB V PENGUJIAN DAN PEMBAHASAN HASIL

Bab ini akan menampilkan hasil uji coba program. Hasil dari pesan rahasia yang sudah di-encode menggunakan Kode Hamming akan diuji menggunakan software steganalisis.

BAB VI PENUTUP

Bab ini merupakan penutup, berisi tentang kesimpulan yang dapat diambil berdasarkan data yang ada dan saran apabila Tugas Akhir ini akan dikembangkan.

“Halaman ini sengaja dikosongkan”

BAB II
TINJAUAN PUSTAKA

Pada bab ini menjelaskan tentang kajian teori dari referensi penunjang serta penjelasan masalah yang dibahas dalam Tugas Akhir ini, meliputi Penelitian sebelumnya terkait Tugas Akhir ini.

Penelitian terkait tugas akhir ini dilakukan oleh Ramadhan J Mstafa yang berjudul " A Highly Secure Video Steganography using Hamming Code (7,4), pada penelitian tersebut J. Mstafa menggunakan konsep Kode Hamming dan skema steganografinya menggunakan frame video dengan citra yang sama sebanyak 300 citra. Pada sampel pengujian dengan nilai PSNR diatas 51 dBs dan " attacker " tidak bisa mendeteksi keberadaan pesan dalam sebuah video tersebut.

Selanjutnya penelitian dilakukan oleh Tahta Dari Timur dengan judul " Kajian Kode Hamming dan Simulasinya menggunakan Sage" dalam penelitian tersebut menyimpulkan bahwa penggunaan encoder *Parity Check* tidak mempengaruhi kecepatan komputasi dan dari hasil komputasi decoder *Syndrom* lebih cepat daripada decoder *Nearest-Neighbor*. Encoder Parity Check adalah cara sederhana untuk mendeteksi error hingga satu bit dengan cara memeriksa jumlah digit dalam kata yang ditransmisikan adalah genap. Sedangkan Decoder Syndrom atau biasa disebut error correction adalah sebuah metode ekstraksi pesan yang dapat mengetahui dimana terjadi error pada suatu pengiriman pesan, sehingga pesan rahasia bisa dicek kebenarannya, apakah pesan yang dikirim sama dengan pesan yang diterima[6].

2.1 Pengertian Steganografi

Steganografi adalah seni mengamankan data atau informasi untuk mencegah pendeteksian pada informasi yang telah disembunyikan, dengan kata lain menyembunyikan pesan dengan

suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia[3].

Steganografi berasal dari bahasa yunani yaitu “steganos” yang berarti “tersembunyi atau terselubung”, dan “graphein” yang berarti “menulis”. Dengan kata lain, steganografi adalah ilmu menyembunyikan informasi dengan suatu cara sehingga selain pengirim dan penerima tidak ada yang mengetahui dan menyadari adanya informasi rahasia yang disisipkan pada sebuah media[1].

Steganografi erat kaitannya dengan kriptografi, yang mana kriptografi adalah mengacak sebuah pesan sehingga tidak dapat dimengerti isi dari pesan tersebut. Sedangkan dalam steganografi penyembunyian pesan berada pada *Citra Medium,* sehingga tidak akan terlihat dan tidak ada yang akan menyadari keberadaan pesan rahasia didalamnya. Pada kriptografi jika pesan yang telah dienkripsi dicuri oleh pencuri pesan maka pencuri pesan akan mengetahui bahwa pesan tersebut telah dienkripsi sebelumnya. Namun dengan steganografi, pencuri pesan tidak akan menyadari bahwa ada pesan rahasia pada citra medium tersebut.

2.1.1 Metode *Least Significant Bit* (*LSB)*

Pada sebuah *citra greyscale,* setiap piksel mewakili 8 bits. Bit terakhir dari sebuah piksel tersebut dinamakan *Least Significant Bit* karena nilainya tidak akan mempengaruhi nilai piksel hanya dengan “1” [5]. Metode LSB (*Least Significant Bit*) ini bekerja dengan cara mengganti bit terakhir dari masing-masing piksel dengan pesan yang akan disisipkan. LSB mempunyai kelebihan yakni ukuran gambar tidak akan berubah karena pesan yang disisipkan terbatas. Sedangkan kekurangannya adalah pesan atau data yang akan disisipkan terbatas, sesuai dengan ukuran citra[4]. Misal pada tabel ASCII, huruf “A” direpresentasikan dalam bentuk biner adalah 01000001. Misal 3 piksel pada citra berukuran 24 bit.

00101101	00011100	11011100
10100110	11000100	00001100
11010010	10101101	01100011

Kemudian nilai setelah dimasukkan nilai ‘A’ berubah menjadi

00101100	00011101	11011100
10100110	11000100	00001100
11010010	10101101	01100011

Setalah terjadi transformasi maka nilainya ada pada angka terakhir yang tebal

2.1.2 *Peak Signal to Noise Ratio*(*PSNR*)

PSNR adalah nilai yang didapat dari hasil pengujian perbandingan antara nilai maksimum dari signal dan kekuatan noise. Semakin besar nilai *PSNR* yang dihasilkan semakin baik kualitas dari stego citra yang dihasilkan. Nilai *PSNR* dibawah 30 db menandakan rendahnya kualitas suatu stego citra. Sedangkan nilai PSNR diatas 40 db menandakan baiknya kualitas citra tersebut. Nilai PSNR dapat dihitung melalui persamaan berikut :

$$PSNR = 10 * log_{10} \frac{(2b-1)^2}{MSE} \qquad (2.1.1)$$

Dengan *b* adalah nilai bit citra yang digunakan bisa 8 bit, 16 bit, dan sebagainya. Dan *MSE* adalah singkatan dari *Mean Square Error*. MSE adalah nilai yang didapat dari hasil pengujian nilai error kuadrat rata-rata antara file citra medium dengan file stego citra. Semakin besar nilai *MSE*, semakin tinggi distorsi yang dihasilkan begitu juga sebaliknya. Nilai dari *MSE* dapat dihitung melalui persamaan:

$$MSE = \frac{1}{MN} \sum_{x=1}^{MN} \sum_{y=1} [I(x,y) - I'(x,y)]^2 \qquad (2.1.2)$$

Dengan nilai M adalah panjang citra stego, N adalah lebar citra stego dan $I(x,y)$ adalah nilai piksel dari citra cover serta $I'(x,y)$ adalah nilai piksel dari citra stego.

2.1.3 *Secure Hash Algorithm*-256 (SHA-256)

Hash adalah sebuah metode untuk mengidentifikasi originalitas dari sebuah media. Hash adalah sebuah fungsi pemetaan data digital, dan untuk mendapatkan data yang benar dengan data yang ditransmisikan maka harus dipetakan ulang setelah terjadinya transmisi data digital. Fungsi hash adalah semacam biometric identifikasi digital dari sebuah file seperti teks, citra, audio, dan video.

SHA-256 adalah salah satu fungsi hash penerus untuk SHA-1 atau secara kolektif disebut SHA-2, dan merupakan salah satu fungsi hash terkuat yang ada. SHA-256 tidak jauh lebih kompleks daripada kode SHA-1[9], dan belum dikompromikan dengan cara apa pun. Kunci 256-bit membuatnya menjadi fungsi mitra yang baik untuk AES(Advanced Encryption Standard). Hal ini didefinisikan dalam NIST (National Institute of Standards and Technology) standar 'FIPS 180-4'. NIST juga menyediakan sejumlah vektor uji untuk memverifikasi kebenaran implementasi. Fungsi hash juga bisa diaplikasikan pada steganografi dengan tujuan pesan yang sudah diencode akan menghasilkan pesan yang sama setelah didecode.

2.2 Kode Hamming

Sebelum membahas lebih jauh tentang Kode Hamming, hal pertama yang harus kita bahas adalah konsep tentang kode linear beserta contohnya, *encoder parity check* beserta contohnya dan

decoder syndrom berikut juga contohnya. Karena kode linear merupakan pondasi untuk memahami Kode Hamming.

2.2.1 Kode Linear

Kode adalah sebuah sistem yang mengatur tentang mengubah suatu informasi, misal surat, kata, suara, gambar, atau gerakan ke sebuah bentuk atau representasi. Sebuah himpunan $C \subseteq \mathbb{F}_q{}^n$ disebut *kode* atas lapangan $\mathbb{F}_q$ dengan elemen elemennya disebut *codeword* [7]. Sebelum mendefinisikan Kode Linear, perlu ditinjau contoh berikut.

Diberikan pesan biner x_1, x_2, x_3 dan digit cek x_4, x_5, x_6 dengan:

$$x_4 = x_1 + x_2$$
$$x_5 = x_1 + x_3$$
$$x_6 = x_2 + x_3$$

Atau

$$x_1 + x_2 - x_4 = 0$$
$$x_1 + x_3 - x_5 = 0$$
$$x_2 + x_3 - x_6 = 0$$

Sistem persamaan diatas digunakan untuk memperoleh *codeword.* Misal diberikan pesan 010, maka pesan ini dapat dikodekan menjadi 010101. Semua *codeword* dapat diperoleh dengan cara yang sama dan ada sebanyak 8 *codeword* dalam kode tersebut [6]. Sistem persamaan diatas dapat diformulasikan sebagai matriks *Hx = 0*

$$H = \begin{pmatrix} 1\,1\,0\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,1 \end{pmatrix}$$

Matriks *H* diatas disebut matriks cek berukuran $(n-k) \times n$, dengan k adalah panjang pesan dan $(n-k)$ adalah panjang digit cek. Matriks *H* di atas berbentuk $(A|I_{n-k})$ dengan *A* adalah matriks berukuran$(n-k) * k$ dan I_{n-k} adalah matriks

berukuran $n-k$. Pada contoh matriks H di atas, k=3, n=3 dan $H=(A|I_3)$. Dapat dilihat bahwa rank*(H)* pastilah $(n-k)$, sebab adanya matriks I_{n-k} menjamin baris baris dari H selalu bebas linear. Jika diberikan matriks cek H berukuran $(n-k)\times n$ atas $\mathbb{F}_q$ dengan rank $n-k$. Himpunan

$$C=\{x:\ Hx^T=0, x\in {\mathbb{F}_q}^n\} \qquad (2.2)$$

Jika k adalah panjang pesan, $n-k$ adalah panjang digit cek, maka *C* adalah kode linear.

2.2.2 Encoder dan Decoder

Encode adalah fungsi $E:\mathbb{F}_q^k\rightarrow\mathbb{F}_q^n$. Misal x adalah pesan vektor dimana $x\in\mathbb{F}_2^{11}, x=(0,1,0,1,0,1,0,1,0,1,0)$. Selanjutnya x diencode dengan encoder *parity check* menjadi $y\in\mathbb{F}_2^{15}$, yaitu $y=(0,0,0,0,0,0,1,1,1,1,0,1,1,0,1)$. Pada sebuah transimisi data digital dari suatu *device* ke sebuah *device* lain, rata-rata data tersebut akan dikirim dengan baik. Namun tidak dapat dipastikan apakah data yang diterima akan sama dengan data yang dikirim. Maka untuk menghindari kejadian yang tidak diinginkan mengenai data tersebut, dikenalkanlah metode pendeteksi error yaitu parity check. Parity check adalah sebuah metode deteksi error dimana data yang dikirim oleh *sender*(pengirim pesan) adalah data yang sama yang diterima oleh *receiver*(penerima pesan).

Decode adalah invers dari dari encode, yaitu $D:\mathbb{F}_q^n\rightarrow\mathbb{F}_q^k$. Misal y dikirim, kemudian terjadi error saat pengiriman sehingga berubah menjadi *y'*, $y\neq y'$. Maka untuk menghitung error yang terjadi adalah $y-y'=e$, $y'=y-e$. Karena $Hx^T=0$ untuk semua $x\in C$, maka $Hy'^T=H(y^T-e^T)=He^T$. Selanjutnya, nilai tersebut disebut *syndrome* dari sebuah vektor. Denga kata lain, nilai *syndrome* dari sebuah codeword error y' pasti sama dengan nilai *syndrome* vektor errornya, sehingga error dapat diketahui dan codeword dapat dikoreksi dengan benar.

Sebagai contoh, misal pada kode Hamming (7,4), sebuah pesan vektor $y = (1,0,1,1,0,1,1)$ dikirim. Selanjutnya, vektor $y' = (1,0,1,1,0,0,1)$ diterima. Untuk mendeteksi apakah terjadi error pada y', perlu dihitung nilai *syndrome* dari y', yaitu dengan $Hy'^T = H(y^T + e^T) = He^T$. Setelah diperoleh nilai *syndrome*, dicari pada tabel nilai vektor error (pimpinan Koset) dari *syndrome* yang bersesuaian. Vektor error ini kemudian digunakan untuk koreksi codeword y', yaitu dengan $y = y' - e$. Berikut ini adalah contoh tabel syndrome pada kode Hamming (7,4).

Tabel 2.2.3 : Contoh tabel Syndrom

Pimpinan	Syndrome
000000	000
100000	011
010000	101
001000	110
000100	100
000010	010
000001	001
001001	111

Vektor dengan bobot minimum dalam suatu koset disebut pimpinan koset [6].Dengan demikian telah dibahas metode encoding yaitu *Parity Check* dan metode decoding menggunakan syndrome. Berikut akan dibahas salah satu bentuk khusus dari kode linear yaitu kode Hamming.

2.3 Pengertian Kode Hamming

Kode Hamming adalah sebuah kode biner C_m sepanjang $n = 2^m - 1, m \geq 2$ dengan matriks cek H berukuran $m\ x\ (2^m - 1)$ dan semua kolom-kolomnya tak nol[7]. Untuk kode Hamming biner, dapat digunakan metode khusus

dalam penentuan error dan koreksi. Misal kolom-kolom H diurutkan sedemikian hingga kolom ke-i adalah representasi biner dari i. Jika y diterima dan $S(y) = Hy^T = He^T$ adalah kolom ke-i dari H, maka error terjadi pada kolom ke-i pada y.

Misal : Diberikan C_3 kode Hamming (7, 4) dengan matriks cek adalah

$$H = \begin{pmatrix} 0\,0\,0\,0\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix}$$

Dapat dilihat bahwa kolom ke-i dari H adalah representasi biner dari i. Jika $S(y) = 101$, maka error terjadi pada elemen ke-5, sebab 101 adalah kolom ke-5 dari H.

Kode Hamming (15,11) adalah Hamming orde ke empat dari $[2^m - 1,\ 2^m - m - 1, 3]$, dimana orde ke satu adalah nol, orde kedua adalah kode Hamming (7,4,3) dan yang ketiga dalah (15,11,3). Kode Hamming bisa menyediakan implementasi jarak minimal decoding, dan membuktikan bahwa itu memang mewujudkan kemungkinan maksimum decoding, yaitu bentuk terbaik dari decoding[10].

Berikut adalah tingkatan orde untuk Kode Hamming

Kode Hamming orde ke 3 = $(2^m - 1,\ 2^m - m - 1,3)$ [7,4,3]
Kode Hamming orde ke 4 = $(2^m - 1,\ 2^m - m - 1,3)$ [15,11,3]

BAB III
METODOLOGI

3.1 Tahap Penelitian

Dalam tahap pengerjaan tugas akhir ini terdapat beberapa tahapan, yaitu:

1. Studi literatur

   Pada tahap ini akan dipelajari tentang dasar dasar teori yang digunakan dalam Kode Linear, Parity Check , Kode Hamming (15,11), pendefinisian Kode Linear, metode metode encoding dan decoding (Encoder Parity Check, Decoder Syndrom).

2. Perancangan Program

   Menggunakan hasil dari dasar teori pada bab sebelumnya akan dirancang sebuah system Kode Linear dan encoding pesan rahasia menggunakan Kode Hamming (15,11) ke dalam media citra.

3. Implementasi Program

   Pada tahap ini akan diimplementasikan program encoding pesan rahasia menggunakan Kode Hamming (15,11). Program dibuat dengan fitur IDE dari python yaitu *pycharm*. Pengguna dapat memberikan input data digital dan pesan rahasia berupa citra.

   Berikut adalah proses penyisipan citra pesan pada citra medium menggunakan Kode Hamming (15,11).

   - Citra pesan dan citra medium dimuat dalam bentuk matriks integer
   - Citra pesan dan citra medium diubah nilai pikselnya menjadi bentuk biner

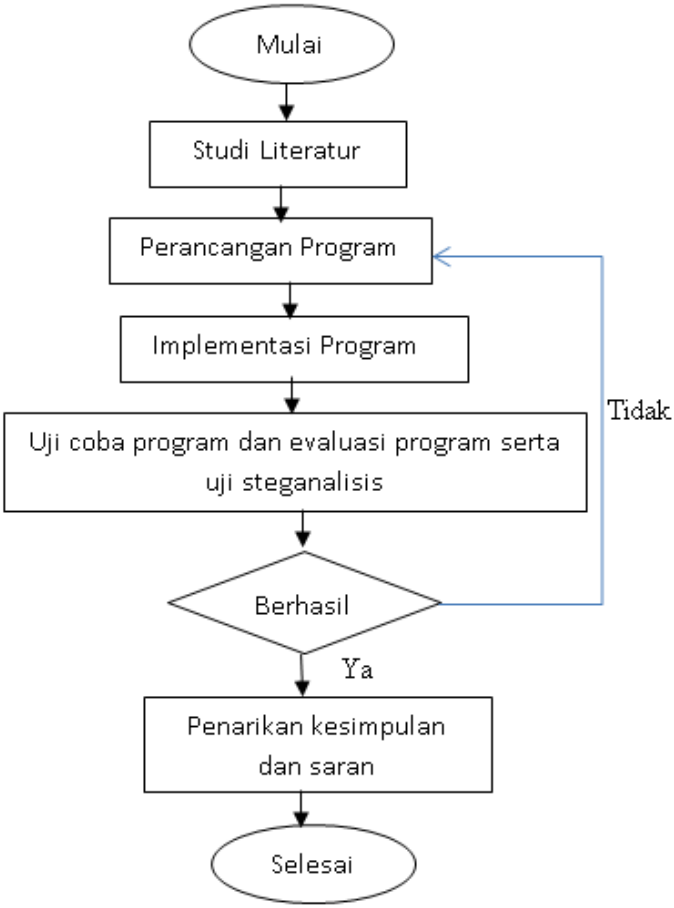
- Sebelum di encode, citra pesan diberi Hash (SHA-256)
- Citra pesan diencode per piksel menggunakan kode Hamming (15,11)
- Untuk menambah keamanan hasil encode, pesan diberi noise secara acak sebanyak 1 bit, karena radius koreksi kode Hamming adalah 1 bit
- Citra pesan disisipkan kedalam citra medium menggunakan metode LSB (*Least Significant Bit)*
- Citra medium yang telah disisipi citra pesan menjadi citra stego

Berikut adalah proses pengambilan citra pesan pada stego citra :

- Citra stego dimuat kedalam bentuk biner,
- Setiap digit terakhir diambil secara berurutan dan dijadikan vektor berukuran 15,
- Setiap vektor tersebut didecode menjadi vektor sepanjang 11 bit menggunakan decoder *syndrome*,
- Hasil dari decoding dikembalikan ke citra pesan yang diinginkan

4. Uji coba program dan uji steganalisis
5. Penarikan kesimpulan dan saran
6. Penulisan tugas akhir

3.2 Diagram Alur Penelitian



Gambar 3.2 Diagram alur metode penelitian

“Halaman ini sengaja dikosongkan”

BAB IV
PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini akan dibahas mengenai analisa dan perancangan perangkat lunak yakni tentang deskripsi metode steganografi, analisa perangkat lunak dan perancangan perangkat lunak.

4.1 Analisis Kebutuhan

Dalam sub bab ini akan dibahas mengenai analisa kebutuhan didalam sebuah sistem dimana terdapat analisa kebutuhan user, analisa kebutuhan system, deskripsi metode encoding dan metode decoding

4.1.1 Analisis Kebutuhan *User (pengguna)*

Perangkat lunak yang akan dibangun dapat membantu *user (pengguna)* dalam menyelesaikan permasalahan tentang pesan rahasia terutama tentang tingkat keamanan pesan. Oleh karena itu perangkat lunak yang dirancang harus memenuhi kebutuhan :

1. Perangkat lunak yang dirancang harus sesuai dengan kebutuhan user
2. Perangkat lunak yang dirancang bisa menyelesaikan masalah yang dihadapi *user* (pengguna)
3. Perangkat lunak yang dirancang *user friendly* (mudah dalam pengoperasian)
4. Perangkat lunak yang dirancang mempunyai tampilan yang menarik dan familiar bagi pengguna

4.1.2 Analisis Kebutuhan Sistem

Pada uji coba steganografi citra dengan kode Hamming (15,11) akan digunakan perangkat keras dan perangkat lunak. Untuk perangkat lunak, tool yang digunakan adalah Pycharm dan Net Beans 8.2, lebih jelasnya ditunjukkan pada tabel 4.1

Tabel 4.1. Tabel kebutuhan minimal sistem

Perangkat Keras	Prosesor	: Intel® Core ™ i5-Resolusi 1366x768 CPU @2.5 Ghz
	RAM Memory : 2 GB	
Perangkat Lunak	Sistem Operasi : Linux Mint	
	Tools	: Pycharm Community 2018.1.3 : Net Beans 8.2

4.2 Deskripsi Metode

4.2.1 Metode Encoding

Dalam steganografi, istilah yang sangat sering di jumpai adalah encode. Istilah simpel untuk encode adalah merubah atau mengganti sebuah data dari bentuk normal ke bentuk tertentu atau dalam artian menyamarkan sebuah pesan biasa menjadi sebuah pesan tertentu untuk keperluan karahasiaan pesan. Dalam sub bab ini akan dibahas mengenai penyisipan pesan rahasia dalam bentuk citra ke sebuah citra medium sebagai media untuk steganografi.

Sebelum melakukan encoding terhadap sebuah pesan rahasia, dalam hal ini metode yang digunakan untuk proses penyisipan pesan pada sebuah media citra adalah metode LSB (*Least Significant Bit*). Metode LSB menjadi metode yang sering dipakai untuk proses penyisipan pesan rahasia pada sebuah file citra, audio, maupun video dikarenakan bit yang berubah adalah bit terakhir dari sebuah piksel sehingga perubahan warna pada bit terakhir tidak akan terlalu merubah warna dari sebuah piksel dan perubahan bit terakhir jika dilihat secara kasat mata perubahannya tidak

terlalu signifikan dan cenderung tidak berubah dari bentuk awal.

Berbeda dengan metode MSB (*Most Significant Bit*), bit yang berubah adalah bit awal dari sebuah bentuk piksel biner sehingga ketika yang berubah adalah awal dari sebuah bit maka perubahan yang sangat significant akan terjadi dan akan terlihat jelas oleh kedua mata manusia.

4.2.2 Proses Penyisipan Citra

Dalam sebuah proses penyisipan pesan ke dalam sebuah citra, baik pesan itu sebuah plaint teks atau sebuah citra. Maka yang harus dulu kita pahami adalah unsur unsur yang ada dalam sebuah citra terlebih dahulu. Untuk memperjelas proses penyisipan citra ke sebuah media berupa citra, maka akan dicontohkan sebagai berikut :

Misal sebuah citra pesan rahasia yang akan disisipkan adalah citra “test.bmp”. Format citra yang digunakan adalah BMP dengan dimensi 113x20 dan berukuran 382 bytes.

Gambar 4.1 Citra pesan rahasia

Misal pada citra voldemort di atas memiliki matriks citra berukuran 1x2 , dengan nilai sebagai berikut :

$$[0 \quad 255]$$

Kemudian bentuk matriks tersebut diubah menjadi bentuk biner sebagai berikut :

$$[00000000 \quad 11111111]$$

Selanjutnya bentuk biner dari citra pesan rahasia akan disisipkan ke dalam sebuah media citra yang ukuran dan dimensi nya lebih besar dari citra pesan rahasia. Misal media citra yang menjadi wadah adalah citra "cameraman.bmp", menggunakan format yang sama yaitu bmp dan berdimensi 256x256 dengan ukuran 32KB.

Gambar 4.2 Citra medium

Ukuran media untuk penyisipan pesan harus lebih besar supaya pesan rahasia yang telah disisipkan kedalam media citra yang lebih besar bisa tercover dan lebih mudah tersamarkan keberadaan pesan rahasia tersebut.

Misal diasumsikan untuk citra medium memiliki matriks berukuran 4x4 sebagai berikut :

$$\begin{bmatrix} 34 & 45 & 77 & 69 & 45 \\ 35 & 75 & 70 & 95 & 75 \\ 39 & 72 & 78 & 99 & 72 \\ 61 & 40 & 71 & 92 & 40 \\ 35 & 75 & 70 & 95 & 75 \\ 61 & 40 & 71 & 92 & 40 \end{bmatrix}$$

Gambar 4.3 Matriks citra medium

Kemudian matriks diatas diubah bentuk menjadi bentuk biner. Karena dalam proses penyisipan, metode yang dipakai adalah metode LSB (*Least Significant Bit*). Metode LSB adalah sebuah metode penyisipan pesan kedalam bit terakhir dalam sebuah piksel dari sebuah bentuk biner. Bentuk biner dari matriks 4x4 diatas adalah :

$$\begin{bmatrix} 00100010 & 00101101 & 01001101 & 01000101 & 00101101 \\ 00100011 & 01001011 & 01000110 & 01011111 & 01001011 \\ 00100111 & 01001000 & 01001110 & 01100011 & 01001000 \\ 00111101 & 00101000 & 01000111 & 01011100 & 00101000 \\ 00100011 & 01001011 & 01000110 & 01011111 & 01001011 \\ 00111101 & 00101000 & 01000111 & 01011100 & 00101000 \end{bmatrix}$$

Gambar 4.4 Matriks biner citra medium

Langkah selanjutnya adalah Meng-encode citra pesan rahasia dengan Kode Hamming (15,11). Karena citra pesan rahasia masih berbentuk citra dengan ukuran 8 bit, maka citra 8 bit tersebut diubah menjadi citra berukuran 11 bit dengan menambahkan 3 bit citra dengan nilai 0 didepan. Selanjutnya citra 11 bit tersebut di-encode dengan Kode Hamming supaya menjadi 15 bit.

Tabel 4.2 Konversi citra dari 8 bit ke 15 bit

Desimal	Citra 8 bit	Citra 11 bit	Citra 15 bit
0	00000000	00000000000	000000000000000
255	11111111	11111111111	111111111111111

Dari Tabel 4.2 diatas dapat disimpulkan bahwa citra pesan rahasia setelah di-encode menggunakan kode Hamming (15,11) terdapat perubahan ukuran bit pada citra. Jadi dapat ditarik kesimpulan bahwa untuk citra berukuran 8 bit, maka ukuran minimal untuk penyisipan citra pesan rahasia adalah 8x2 piksel. Namun setelah citra pesan rahasia tersebut di-encode dengan Kode Hamming (15,11) maka ukuran piksel nya harus lebih dari 15x2 piksel.

Berikutnya adalah menyisipkan citra pesan rahasia yang sudah di-encode dengan Kode Hamming (15,11) ke dalam satu bit terakhir dari citra medium menggunakan metode LSB (Least Significant Bit). Untuk lebih jelas bisa dilihat pada Tabel 4.3 dan Gambar 4.5 proses penyisipan citra pesan rahasia ke dalam citra medium

Tabel 4.3 Bentuk biner dari citra pesan rahasia

Desimal	Citra Pesan Rahasia
0	**000000000000000**
255	**111111111111111**

$$\begin{bmatrix} 00100001\mathbf{0} & 00101100 & 01001100 & 01000100 & 00101100 \\ 00100001\mathbf{0} & 01001010 & 01000110 & 01011110 & 01001010 \\ 00100110 & 01001000 & 01001110 & 01100010 & 01001000 \\ 00111101 & 00101001 & 01000111 & 01011101 & 00101001 \\ 00100011 & 01001011 & 01000111 & 01011111 & 01001011 \\ 00111101 & 00101001 & 01000111 & 01011101 & 00101001 \end{bmatrix}$$

Gambar 4.5 Bentuk biner dari citra stego

Citra medium yang sudah disisipi pesan rahasia bisa terlihat pada bit akhir yang tercetak tebal. Metode LSB pada aplikasinya bisa merubah satu atau dua bit terakhir dari sebuah citra medium.

Selanjutnya setelah proses penyisipan berhasil maka bentuk biner dari citra medium tersebut diubah kembali menjadi bentuk desimal seperti pada gambar 4.6 dibawah ini.

$$\begin{bmatrix} 34 & 45 & 77 & 69 & 45 \\ 35 & 75 & 70 & 95 & 75 \\ 39 & 72 & 78 & 99 & 72 \\ 61 & 40 & 71 & 92 & 40 \\ 35 & 75 & 70 & 95 & 75 \\ 61 & 40 & 71 & 92 & 40 \end{bmatrix}$$

Gambar 4.6 Bentuk desimal dari stego citra

. Proses terakhir adalah merubah bentuk desimal dari Gambar 4.6 di atas menjadi stego citra seperti Gambar 4.7 di bawah ini.

Gambar 4.7 stego-citra

4.2.3 Metode Decoding

Untuk men-decode sebuah pesan rahasia dari sebuah citra maka citra yang menjadi masukan adalah citra yang telah disisipkan sebuah pesan rahasia biasa dikenal dengan stego-citra.

Gambar 4.8 Stego-citra

Matriks desimal dari stego-citra diatas adalah :

$$\begin{bmatrix} 34 & 45 & 77 & 69 & 45 \\ 35 & 75 & 70 & 95 & 75 \\ 39 & 72 & 78 & 99 & 72 \\ 61 & 40 & 71 & 92 & 40 \\ 35 & 75 & 70 & 95 & 75 \\ 61 & 40 & 71 & 92 & 40 \end{bmatrix}$$

Gambar 4.9 Bentuk desimal dari stego citra

Kemudian matriks diatas diubah bentuk menjadi bentuk biner untuk men-decode pesan rahasia yang ada pada

stego-citra. Maka matriks diatas ketika diubah menjadi bentuk biner akan menjadi :

$$\begin{bmatrix} 00100001 & 00101100 & 01001100 & 01000100 & 00101100 \\ 00100001 & 01001011 & 01000110 & 01011110 & 01001011 \\ 00100110 & 01001000 & 01001110 & 01100010 & 01001000 \\ 00111101 & 00101001 & 01000111 & 01011101 & 00101001 \\ 00100011 & 01001011 & 01000111 & 01011111 & 01001011 \\ 00111101 & 00101001 & 01000111 & 01011101 & 00101001 \end{bmatrix}$$

Gambar 4.10 Matriks biner stego-citra

Setelah itu sistem decode akan mengambil bit-bit terakhir dari setiap piksel yang ada pada stego-citra tersebut sehingga bit-bit terakhir tadi ketika dikumpulkan akan membentuk sebuah bilangan biner.

0	255
000000000000000	**111111111111111**

Gambar 4.11 Matriks biner pesan rahasia

Selanjutnya adalah mendecode pesan yang sudah di encode dengan Kode Hamming (15,11) dengan cara mengganti 15 bit pesan rahasia menjadi 8 bit pada dictionary.

Tabel 4.4 Konversi citra dari 15 bit ke 8 bit

Desimal	Citra 8 bit	Citra 11 bit	Citra 15 bit
0	00000000	00000000000	000000000000000
255	11111111	11111111111	111111111111111

Dari bilangan biner tersebut didapat untuk citra 15 bit yang sudah didecode maka citra 8 bitnya adalah sesuai pada Tabel 4.4

0	255

Untuk proses terakhir yaitu bentuk desimal diatas yang akan menjadi pesan rahasia dan sistem akan menampilkan citra pesan rahasia tersebut dalam perangkat lunak seperti ditunjukkan pada Gambar 4.12

Gambar 4.12 Pesan rahasia yang telah di decode

4.3 Perancangan Perangkat Lunak

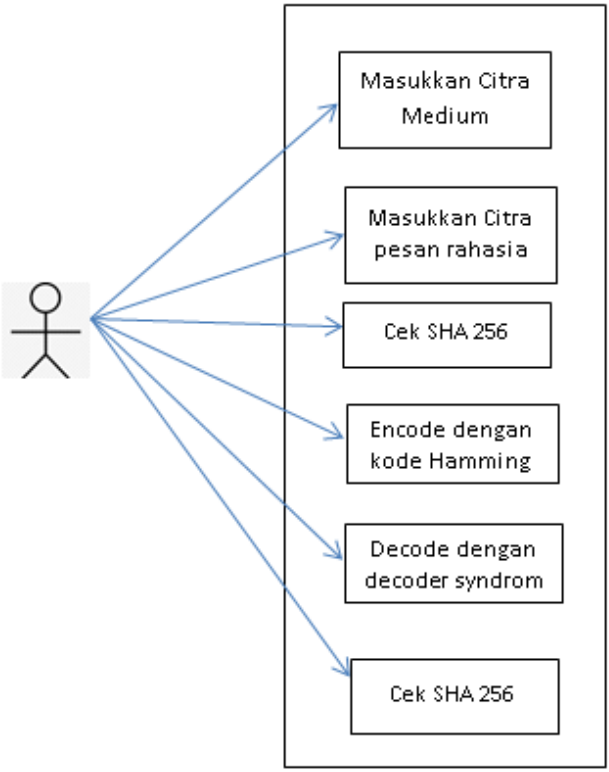
Dalam sub bab ini akan dijelaskan mengenai proses perancangan perangkat lunak yang akan disajikan dalam bentuk *Use Case Diagram* dan *Activity Diagram* serta design antar muka system.

4.3.1 Gambaran Umum Sistem

Pada sub bab ini menjelaskan gambaran awal dari suatu perangkat lunak tersebut meliputi hal-hal apa saja yang dapat dilakukan oleh user yang akan disajikan dalam bentuk *Use Case diagram.* Dalam *Use Case diagram*, user dapat melakukan hal-hal berikut :

1. Memasukkan citra medium. Sebagai citra wadah untuk penyisipan pesan rahasia maka citra medium harus memiliki ukuran yang lebih besar dari citra pesan rahasia
2. Cek SHA-256. Tujuan pemberian SHA-256 kedalam sebuah pesan rahasia adalah untuk memastikan citra yang dikirim melalui transformasi digital adalah citra yang sama dengan citra yang diterima.
3. Memasukkan pesan rahasia. Pesan rahasia yang diinputkan berupa citra yang berukuran lebih kecil dari citra medium.
4. Encode pesan rahasia dengan Kode Hamming. Sebelum dilakukan proses LSB, citra pesan rahasia di-encode menggunakan kode Hamming (15,11)
5. Decode stego-citra untuk mendapatkan pesan rahasia
6. Melakukan cek SHA-256 kembali untuk memastikan signature yang ada ada pesan rahasia sama dengan pesan awal.

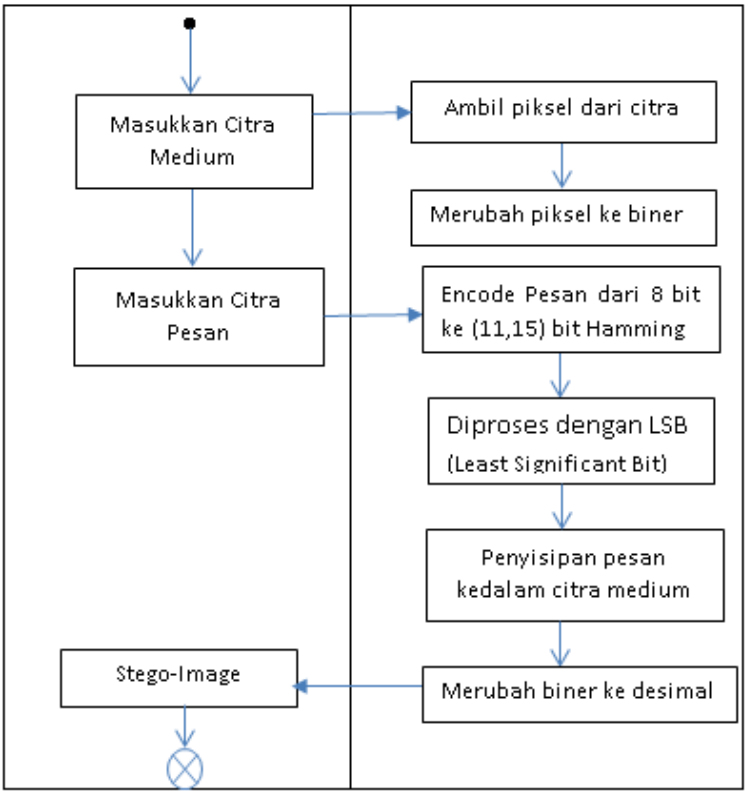
4.3.2 Use Case Diagram



Gambar 4.13 Use Case diagram pada steganografi citra

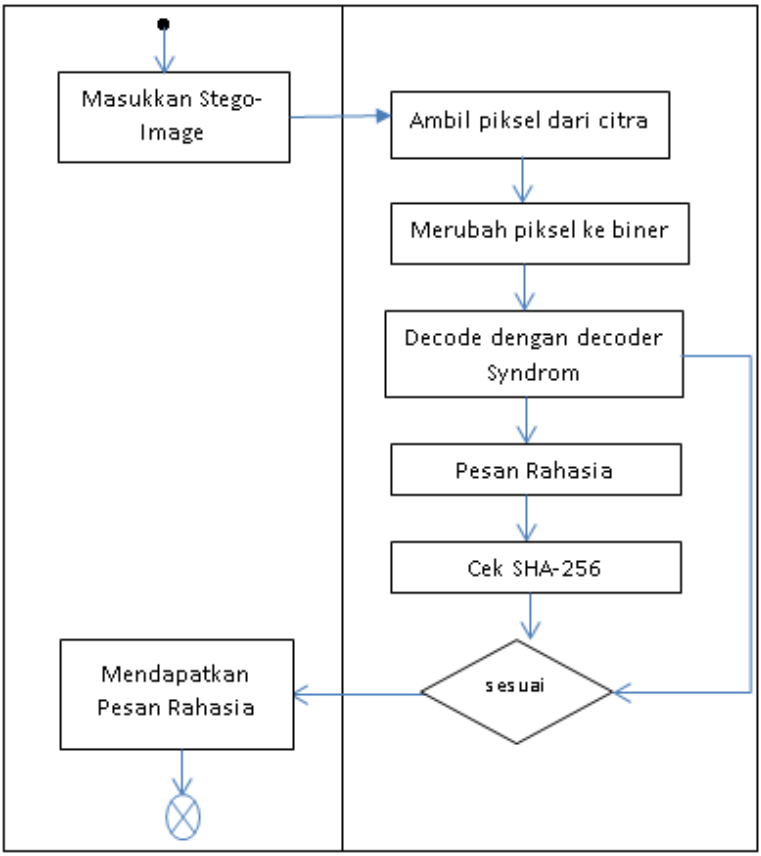
Pada use case diagram di atas, user atau pengguna dapat melakukan 6 step untuk melakukan sebuah pengamanan data pada steganografi citra menggunakan konsep Kode Hamming. Namun untuk keamanan data dilakukan cek ulang SHA untuk menjamin data yang terkirim alamat SHA nya sama.

4.3.3 Activity Diagram



Gambar 4.14 Diagram Activity proses encoding

Dari Gambar 4.14 di atas menjelaskan proses encoding pesan rahasia dengan Kode Hamming (15,11). Jadi sebelum pesan rahasia disisipkan ke dalam citra medium, terlebih dahulu melakukan pengamanan data dengan menggunakan Kode Hamming (15,11).



Gambar 4.15 Diagram Activity pada decoding

Dari Gambar 4.15 dapat diambil kesimpulan bahwa untuk proses decoding, decoder yang digunakan adalah decoder syndrome dan untuk memastikan pesan rahasia yang didecode sudah benar, maka perlu di check SHA-256 yang ada pada pesan rahasia tersebut. Ketika SHA-256 tidak sesuai dengan SHA-256 pada waktu proses encoding maka perlu decode ulang, sampai SHA-256 sesuai.

4.4 Implementasi Program

4.4.1 Implementasi Encoding

Untuk melakukan proses encoding pesan rahasia menggunakan Kode Hamming (15,11), dapat dilakukan di Pycharm dengan :

```
import cv2
import numpy
import LSB
from enc import enc
import hashlib

gambar_rahasia = 'test.bmp'
gambar_carrier = 'carrier.bmp'
gambar_hasil = 'new_image.bmp'

# data
img = cv2.imread(gambar_rahasia, cv2.IMREAD_GRAYSCALE)
h = img.shape[0]
w = img.shape[1]

sha_signature = hashlib.sha256(img).hexdigest()
print('SHA256: ', sha_signature)

# ubah piksel dari desimal ke 11 digit binary (string)
img_bin = []
for i in range(0, h):
    img_bin2 = []
    for j in range(0, w):
        img_bin2.append('{0:011b}'.format(img[i, j]))
    img_bin.append(img_bin2)
# encode menggunakan dictionary enc (code hamming)
menjadi 15 digit binary (string)
encoded_img = []
for i in range(0, h):
    encoded_img2 = []
    for j in range(0, w)
```

Source Code 4.4.1 Proses Encoding

4.4.2 Implementasi Decoding

Untuk melakukan proses decoding citra yang sudah di-encode dengan Kode Hamming (15,11), supaya hasil lebih presisi dan SHA-256 yang tepat maka untuk panjang dan lebar pesan rahasia harus tepat :

```
import cv2
import numpy
import LSB
from dec import dec
import hashlib

gambar_hasil = 'new_image.bmp'

# data
steg = cv2.imread(gambar_hasil)
h = steg.shape[0]
w = steg.shape[1]

# LSB steganography decode
img, carrier = LSB.decode(steg, 1)

# ubah ke 1 channel
img, _, _ = cv2.split(img)

# ubah piksel dari desimal ke 15 digit binary (string)
img_bin = []
for i in range(0, h):
    img_bin2 = []
    for j in range(0, w):
        img_bin2.append('{0:015b}'.format(img[i, j]))
    img_bin.append(img_bin2)

# decode menggunakan dictionary dec (code hamming)
menjadi 11 digit binary (string)
h = 20
w = 113
encoded_img = []
for i in range(0, h):
    encoded_img2 = []
```

Source Code 4.4.2 Proses Decoding

4.4.3 Metode LSB

Untuk menyisipkan pesan ke citra medium akan digunakan metode Least Significant Bit sebagai berikut :

```
import cv2, copy, time

def int_to_bin(number):
    return "{0:b}".format(number).zfill(8)

def change_bits(secret_px_bin_val, cover_px_bin_val,
bit_count):
    return cover_px_bin_val[:-bit_count] +
secret_px_bin_val[:bit_count]

def extract_bits(steg_px_bin_val, bit_count):
    return steg_px_bin_val[-bit_count:]

def extract_cover_bits(cover_px_bin_val, bit_count):
    return cover_px_bin_val[:-bit_count]

def normalize_secret_bits(extract_steg_px_bin_val):
    bit_val = extract_steg_px_bin_val.ljust(8,'0')
    extract_steg_px_int_to_bin_val =
bin_to_int(bit_val)
    return extract_steg_px_int_to_bin_val

def normalize_cover_bits(extract_cover_px_bin_val):
    bit_val = extract_cover_px_bin_val.zfill(8)
    extract_cover_px_val = bin_to_int(bit_val)
    return extract_cover_px_val

def bin_to_int(bin_value):
    return int(bin_value, 2)

#this function change the cover pixel value using
secret pixel value
def change_lsb(secret_px, cover_px, bit_count):
    #RGB binary values of secret image pixel
    secret_px_b_val = int_to_bin(secret_px[0])
```

Source Code 4.4.3 Metode LSB

4.4.4 PSNR

Berikut ini adalah source code untuk menghitung nilai PSNR pada sebuah citra. Citra yang digunakan untuk menghitung nilai PSNR adalah citra stego dan citra medium. Ketika nilai PSNR pada sebuah stego citra diatas 40 dB maka citra tersebut bernilai baik. Berikut ini adalah perhitungan PSNR pada pycharm

```
import cv2
import numpy

def getPSNR(image1, image2):
    channel = 1
    sse = cv2.absdiff(image1, image2)
    sse = sse.sum()
    if sse <= 1e-10:
        return 0
    else:
        total = image1.shape[0] * image1.shape[1]
        mse = sse / (channel * total)
        psnr = 10.0 * numpy.log10((255 * 255) / mse)
        return psnr

image1 = cv2.imread('cameraman.bmp',
cv2.IMREAD_GRAYSCALE)
image2 = cv2.imread('new_image.bmp',
cv2.IMREAD_GRAYSCALE)

psnr = getPSNR(image1, image2)
print(psnr)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

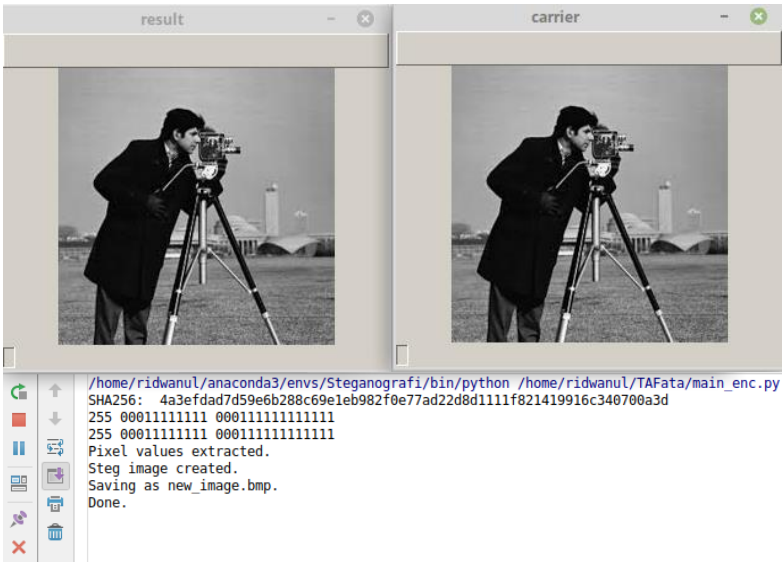
4.4.4 Source Code PSNR

4.5 Hasil Simulasi

Pada sub bab ini akan dilakukan simulasi program. Dari hasil simulasi yang dijalankan dapat ditentukan nilai PSNR dan ketepatan SHA-256 untuk masing masing citra yang sudah di-encode dengan Kode Hamming (15,11). Selain itu juga system akan menampilkan Citra Stego dan Citra Medium sehingga bisa dibandingkan secara langsung.

4.5.1 Simulasi Encoding

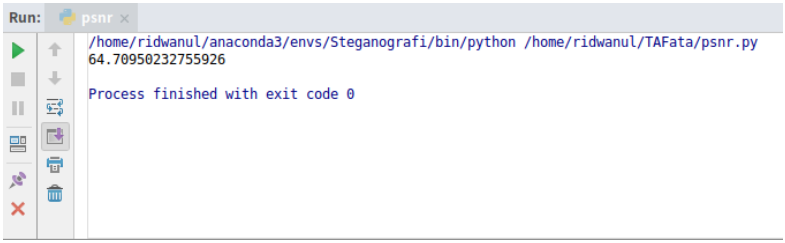
Berikut ini adalah implementasi encoding pada citra cameraman.bmp yang telah disisipi pesan.



Gambar 4.5.1 Perbandingan citra stego dan citra medium

Dari gambar diatas, citra carrier (citra medium) dan citra result (citra stego) dapat dilihat bahwa secara kasat mata hanya ada sedikit perubahan warna antara stego citra

dan citra medium. System juga menampilkan SHA-256 yang ada pada citra pesan rahasia.



Gambar 4.5.3 Nilai PSNR pada stego citra

Proses terakhir dari sebuah pengamanan citra stego adalah membandingkan citra medium dengan citra stego dengan sebuah perhitungan PSNR . Jika nilai PSNR dari citra stego lebih besar dari 40 dB maka citra stego tersebut memiliki nilai PSNR yang bagus, jika nilai PSNR semakin mendekati angka 100 maka citra medium dan citra stego bisa dikatakan tidak memiliki perbedaan sama sekali.

BAB V
PENGUJIAN DAN PEMBAHASAN

Pada bab ini akan dibahas tentang hasil uji coba, perbandingan uji coba kualitatif, perbandingan ukuran file dan Uji Coba steganalisis terhadap Stego citra.

5.1 Data dan Uji Coba

Dalam proses uji coba pada Tugas Akhir ini akan menggunakan lima citra medium yang berbeda dengan format citra BMP.

Tabel 5.1 Sampel citra medium yang diuji coba

No	Citra	Nama	Resolusi	Ukuran
1		Cameraman.bmp	225x225	52.4 KB
2		Imagelena.bmp	512x512	263.2 KB
3		Lion.bmp	970x756	735.9 KB

4		Tiger.bmp	1920x1440	2.8 MB
5		ITS.bmp	1.182x665	788.4 KB

Berikut adalah data uji coba untuk pesan rahasia bisa di lihat pada table 5.2 dibawah ini

Tabel 5.2 Sampel citra pesan rahasia yang diuji coba

No	Citra	Nama	Resolusi	Ukuran
1		math.bmp	500x140	71 KB
2		css.bmp	500x140	71.1 KB
3		Logo.bmp	200x200	41.1 KB

5.1 Uji Coba Kuantitatif

Pada uji coba kuantitatif, tiga citra pesan rahasia akan diuji coba dengan lima citra medium yang telah disebutkan pada table 5.1 dan 5.2 diatas.

Tabel 5.3 Uji coba dengan pesan rahasia math.bmp

No	Citra pesan	Citra medium	Encode	Decode
1	Math.bmp	Cameraman.bmp	Tidak berhasil	Tidak berhasil
2	Math.bmp	Imagelena.bmp	Berhasil	Berhasil
3	Math.bmp	Lion.bmp	Berhasil	Berhasil
4	Math.bmp	Its.bmp	Berhasil	Berhasil
5	Math.bmp	Tiger.bmp	Berhasil	Berhasil

Dari table 5.3 di atas menunjukkan bahwa pesan rahasia tidak berhasil meng-encode dan men-decode dikarenakan citra pesan rahasia lebih besar daripada citra medium sehingga tidak memungkinkan ada tempat yang cukup untuk citra pesan rahasia tersebut.

Tabel 5.4 Uji coba dengan pesan rahasia css.bmp

No	Citra pesan	Citra medium	Encode	Decode
1	css.bmp	Cameraman.bmp	Tidak berhasil	Tidak berhasil
2	css.bmp	Imagelena.bmp	Berhasil	Berhasil
3	css.bmp	Lion.bmp	Berhasil	Berhasil
4	css.bmp	Its.bmp	Berhasil	Berhasil
5	css.bmp	Tiger.bmp	Berhasil	Berhasil

Dari table 5.4 di atas menunjukkan bahwa pesan rahasia tidak berhasil meng-encode dan men-decode dikarenakan citra pesan rahasia lebih besar daripada citra medium sehingga tidak memungkinkan ada tempat yang cukup untuk citra pesan rahasia tersebut.

Tabel 5.5 Uji coba dengan pesan rahasia logo.bmp

No	Citra pesan	Citra medium	Encode	Decode
1	logo.bmp	Cameraman.bmp	Berhasil	Berhasil
2	logo.bmp	Imagelena.bmp	Berhasil	Berhasil
3	logo.bmp	Lion.bmp	Berhasil	Berhasil
4	logo.bmp	Its.bmp	Berhasil	Berhasil
5	logo.bmp	Tiger.bmp	Berhasil	Berhasil

Dari table 5.5 di atas menunjukkan bahwa pesan rahasia berhasil meng-encode dan men-decode kembali dikarenakan ukuran pesan rahasia cukup untuk menempati citra medium yang sudah diujicoba.

5.1 Perbandingan Ukuran File

Pada sub bab ini akan dilakukan perbandingan ukuran file dan dimensi antara citra asal dan stego citra. Untuk masukan citra pesan rahasia mengikuti pada aturan pengujian sebelumnya. Hasil perbandingan ukuran file dapat dilihat pada table 5.5 di bawah ini.

Tabel 5.6 Perbandingan ukuran file

No	Citra Asal	Citra Stego
1	Lena.bmp 512x512 resolusi 8 bit-colour 263.2 KB	Stegolena.bmp 512x512 resolusi 8 bit- colour 263.2 KB
2	Cameraman.bmp 225x225 resolusi 8 bit- colour 52.4 KB	Stegocameraman.bmp 225x225 resolusi 8 bit- colour 52.4 KB
3	Lion.bmp 970x756 resolusi 8 bit- colour 735.9 KB	Stegolion.bmp 970x756 resolusi 8 bit- colour 735.9 KB
4	Tiger.bmp 1920x1440 resolusi 8 bit- colour 2.8 MB	Stegotiger.bmp 1920x1440 resolusi 8 bit- colour 2.8 MB
5	ITS.bmp 1.182x665 8 bit colour 788.4 KB	StegoITS.bmp 1.182x665 8 bit colour 788.4 KB

Dari table 5.5 di atas dapat diambil kesimpulan bahwa citra asli dengan citra yang sudah disisipi pesan (*stego citra*) tidak merubah ukuran dari citra asli. Hal tersebut merubakan kelebihan dari metode LSB (*Least Significant Bit*), citra asli dan citra stego mempunyai ukuran yang sama sehingga tidak akan menimbulkan kecurigaan adanya keberadaan sebuah file dalam sebuah citra stego.

5.2 Uji Coba PSNR (*Peak Signal to Noise Ratio*)

Dalam uji coba kali ini sampel citra yang digunakan adalah lima citra yang telah diujicoba di atas serta dengan citra pesan rahasia yang sama yang telah disisipkan kedalam citra medium dapat dilihat pada table 5.6 di bawah ini.

Tabel 5.7 Hasil Uji Coba PSNR

No	Citra pesan	Citra medium	Nilai PSNR
1	Math.bmp	Cameraman.bmp	Tidak ada
2	Math.bmp	Imagelena.bmp	56.87
3	Math.bmp	Lion.bmp	61.33
4	Math.bmp	Its.bmp	61.58
5	Math.bmp	Tiger.bmp	67.17
6	css.bmp	Cameraman.bmp	Tidak ada
7	css.bmp	Imagelena.bmp	56.90
8	css.bmp	Lion.bmp	61.32
9	css.bmp	Its.bmp	61.65
10	css.bmp	Tiger.bmp	67.14
11	logo.bmp	Cameraman.bmp	52.23
12	logo.bmp	Imagelena.bmp	59.29
13	logo.bmp	Lion.bmp	63.79
14	logo.bmp	Its.bmp	63.16
15	logo.bmp	Tiger.bmp	69.51

Berdasarkan uji coba PSNR di atas didapatkan bahwa nilai PSNR rata rata diatas 60 db, dapat disimpulkan bahwa citra stego yang dihasilkan berkualitas baik. Dari data di atas dapat ditarik kesimpulan juga bahwa jika semakin besar file dan resolusi dari sebuah citra maka nilai PSNR yang di hasilkan akan semakin besar. Semakin besar nilai PSNR pada sebuah stego citra semakin menyerupai citra asli dan akan sangat sulit dibedakan oleh penglihatan manusia. Sehingga kembali ke tujuan awal

bahwa orang tidak akan menyadari adanya file tersembunyi jika nilai PSNR mendekati nilai 100 pada sebuah stego citra.

5.3 Uji Coba Steganalisis

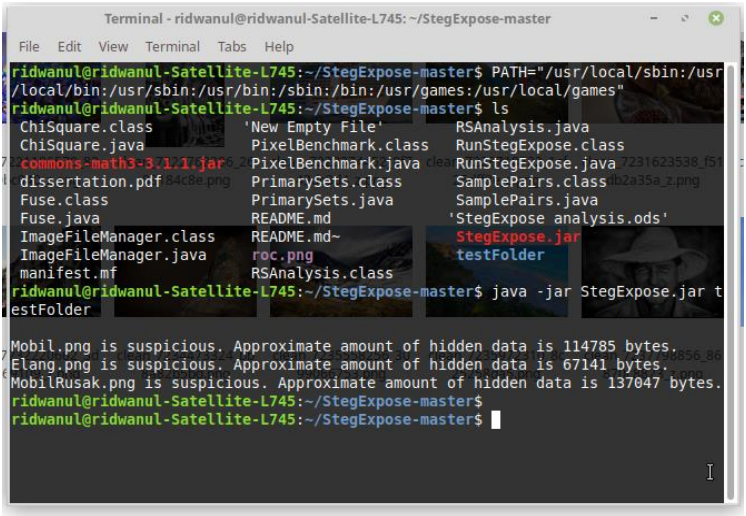
Pada sub bab akan dilakukan uji coba steganalisis terhadap stego citra yang telah disisipkan sebuah file citra menggunakan konsep Kode Hamming. Dalam uji coba kali ini software yang digunakan adalah Netbeans IDE 8.2.

5.5.1 Steganalisis StegExpose

Pada uji coba steganalisis ini, akan dilakukan uji coba dua kali terhadap stego citra. Yang pertama akan menggunakan steganalisis yang bernama StegExpose. StegExpose adalah sebuah steganalisis yang dikhususkan untuk mendeteksi LSB steganografi. User Interface yang digunakan StegExpose adalah ‘Command Line’dan metode yang dihunakan adalah perpaduan antara banyak metode seperti RS analysis, Chi Square Attack, Primari Sets dan Sample Pairs yang mana dari semua metode tersebut untuk mendeteksi keberadaan steganografi dalam sebuah citra[11].

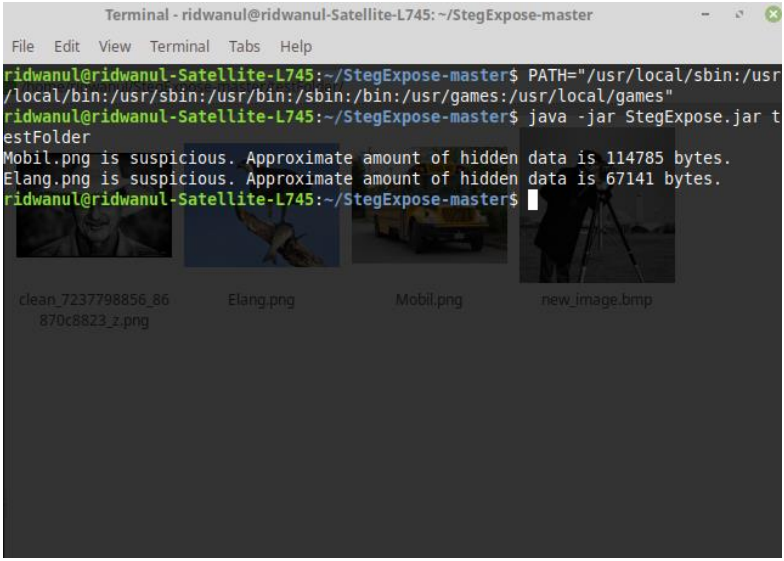
StegExpose juga merupakan sebuah tools yang bisa mendeteksi nilai kuantitatif steganalisis (bisa menentukan ukuran dari data yang tersembunyi dari sebuah steganografi). Aplikasi ini merupakan bagian dari tesis pasca sarjana dari sebuah school of Computing Universitas Kent, di Canterbury, UK[11].

Berikut ini adalah uji coba software steganalisis yang pertama menggunakan aplikasi StegExpose



Gambar 5.5.1 StegExpose mendeteksi keberadaan steganografi

Dari gambar di atas dapat kita lihat bahwa dalam suatu folder tersebut ada tiga citra yang terdeteksi menggunakan steganografi berikut juga ukuran file tersembunyi pada sebuah citra. Ketiga citra yang terdeteksi keberadaan steganografinya adalah Mobil.png, Elang.png dan MobilRusak.png. Berikut nya adalah memasukkan citra stego yang telah disisipkan pesan rahasia menggunakan konsep Kode Hamming pada sebuah folder tersebut. Dan citra yang sudah terdeteksi sebelumnya sebagian dipindahkan ke folder lain. Selanjutnya adalah melakukan running program kembali seperti pada gambar 5.5.1 di bawah ini

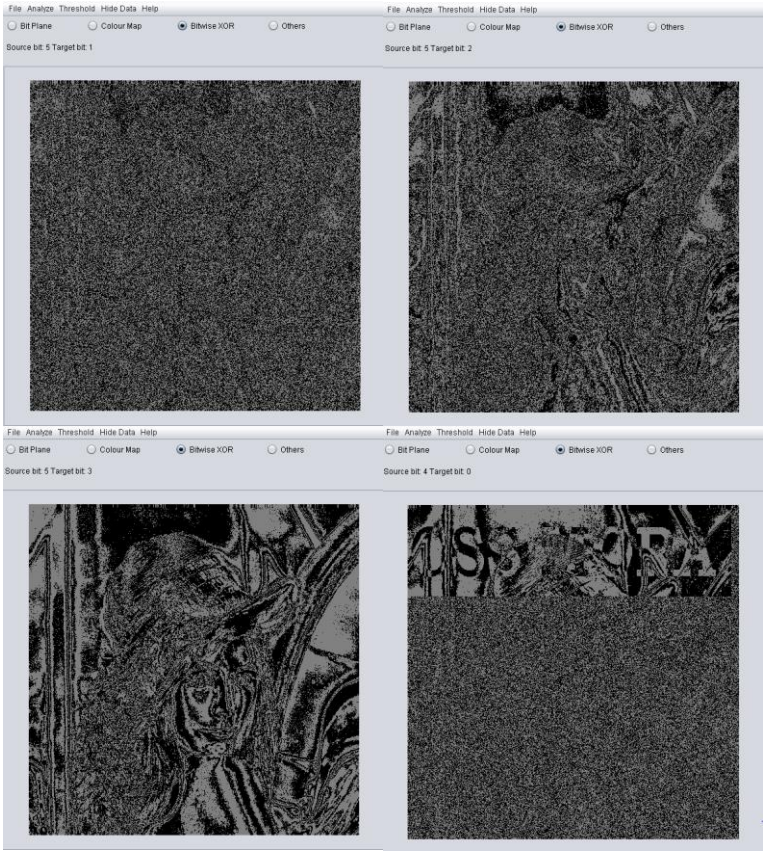


Gambar 5.5.2 StegExpose tidak bisa mendeteksi citra steganografi pada Steg_cam.png dan Steg_ITS.bmp

Dari gambar 5.5.1 di atas bisa kita simpulkan bahwa new_image.bmp tidak bisa dideteksi oleh software StegExpose bahwa kedua gambar tersebut merupakan citra stego yang membawa sebuah data. Dan StegExpose juga tidak dapat mendeteksi berapa kapasitas data yang ada dalam kedua citra tersebut.

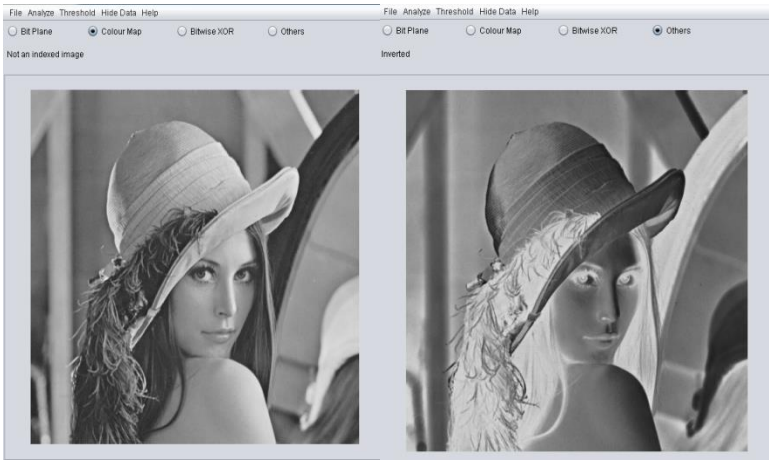
5.5.2 Steganalisis ImageStegano Master

Pada Uji steganalisis yang kedua Aplikasi yang digunakan adalah Image Stegano master, metode yang digunakan dalam aplikasi ini adalah metode Bit Plane, Colour Map dan Bitwise XOR[12]. Berikut adalah pengujian kepada stego citra menggunakan konsep Kode Hamming dengan steganalisis tool ImageStegano :



Gambar 5.5.2 Steganalisis pada metode Bitwise XOR

Dari uji coba di atas pesan rahasia sedikit terbaca dengan citra menggunakan metode bitwise XOR. Namun metode bitwise XOR tidak bisa menentukan berapa ukuran file dan SHA pada citra css.bmp. Berikut ini uji coba menggunakan steganalisis menggunakan metode colour map :

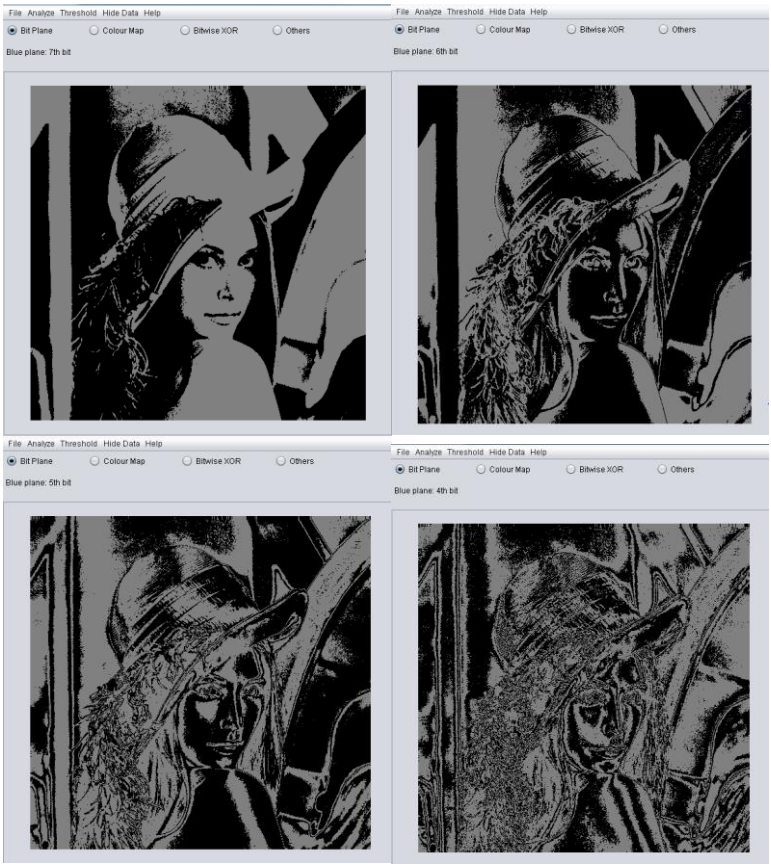


Gambar 5.5.3 Metode Colour Map

Dari gambar 5.5.3 diatas, Metode Colour Map tidak dapat mendeteksi adanya steganografi dalam sebuah stego citra. Dapat dilihat dari gambar diatas bahwa citra stego tersebut bukan termasuk indeks dalam Colour Map.

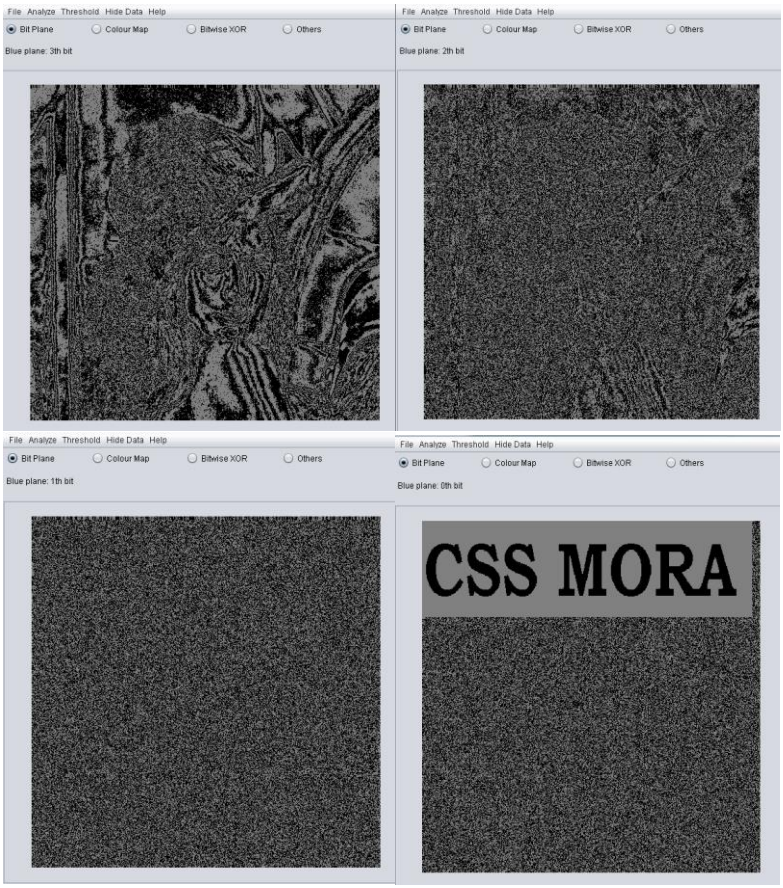
Berikut adalah analisis steganografi menggunakan metode bit plane. Bit plane adalah metode untuk melihat slicing dari setiap slice sebuah citra. Untuk citra berukuran 8 bit, 0 di kodekan dengan 00000000 dan 255 di kodekan dengan 11111111. Bit di sisi paling kiri disebut MSB (*Most Significant Bit*) dan bit yang paling jauh disebut dengan LSB (*Least Significant Bit*).

Pada analisis steganografi menggunakan metode Bit Plane terdapat delapan kali analisis terhadap blue plane, delapan kali terhadap red plane dan delapan kali analisis terhadap green plane. Berikut adalah analisis steganografi menggunakan blue plane dapat dilihat pada gambar 5.5.4



Gambar 5.5.4 Analisis steganografi dengan Bit Plane

Dari gambar 5.5.4 di atas merupakan analisis citra steganografi pada blue plane bit ke tujuh sampai dengan blue plane bit ke 4 menggunakan metode bit plane. Dari gambar diatas citra pesan rahasia belum terdeteksi sampai bit ke empat.



Gambar 5.5.5 Bit Plane bisa mendeteksi ada citra lain

Dari uji coba steganalisis di atas dapat disimpulkan bahwa metode Bit Plane bisa menganalisis masing masing bit dari citra stego sehingga pada bit ke 0 bisa membaca keberadaan citra lain dalam file citra stego. Namun metode bit plane tidak bisa memastikan keorisinilan citra asal yang ada dalam citra stego tersebut. Orisinilitas citra yang dimaksud adalah berapa ukuran

dan dimensi citra tersebut serta alamat SHA yang ada pada citra tersebut. Karena dalam sebuah teks, citra, audio dan video terdapat SHA (Secure Hash Algorithm), yang mana setiap bit pada sebuah teks, citra, audio, maupun video mempunyai identitas sendiri sendiri.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan. Di samping itu, pada bab ini juga dimasukkan beberapa saran yang dapat digunakan jika penelitian ini ingin dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian yang telah dilakukan terhadap "Keamanan pesan rahasia pada steganografi citra mengunakan Kode Hamming (15,11)", maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Sistem berhasil melakukan implementasi encoding dengan konsep Kode Hamming (15,11).
2. Sistem berhasil melakukan penyisipan citra pesan rahasia ke sebuah citra medium dengan menggunakan metode LSB (*Least Significant Bit*) tanpa merubah ukuran file citra medium .
3. Hasil perhitungan PSNR menunjukkan bahwa citra Stego memberikan kualitas yang baik. Semakin besar dimensi dan resolusi dari sebuah citra medium maka PSNR yang dihasilkan pun semakin besar. Dengan citra pesan rahasia yang sama dengan ukuran 382 byte, nilai PSNR untuk citra beresolusi 225x225, 8 bit colour dan berukuran 51.15 KB adalah 64.70955 sedangkan nilai PSNR untuk citra beresolusi 1920x1440, 8 bit colour dan berukuran 2.64 Mb adalah 81.9669.
4. Hasil dari beberapa analisis steganografi menunjukkan bahwa sebagian metode tidak dapat mendeteksi keberadaan steganografi pada sebuah citra. Namun steganalisis dengan metode Bit Plane dan Bitwise XOR dapat melihat adanya file

tersembunyi pada bit tertentu. Namun steganalisis tidak dapat menentukan ukuran dan SHA citra pesan rahasia yang ada pada citra medium tersebut.

6.2 Saran

Berdasarkan hasil yang dicapai pada penelitian ini, ada beberapa hal yang penulis sarankan untuk pengembangan selanjutnya yaitu:

1. Program steganalisis khusus untuk LSB berhasil membaca pesan karena metode LSB adalah metode yang umum digunakan dalam steganografi sehingga untuk penelitian selanjutnya bisa menggunakan metode lain untuk steganografi.
2. Banyak analisis steganografi yang kusus untuk mendeteksi citra steganografi yang menggunakan metode LSB, diharapkan untuk penelitian selanjutnya penyisipan citra bisa dilakukan ke media audio atau video .
3. Pada penelitian ini program masih menggunakan citra pesan rahasia biner sehingga untuk penelitian selanjutnya bisa menggunakan gresyscale maupun RGB.
4. Program belum memiliki UI (*User Interface*)karena keterbatasan penulis diharapkan untuk penelitian selanjutnya bisa menggunakan UI yang bagus dan menarik.

DAFTAR PUSTAKA

[1] N. F. Johnson dan S. Jajodia, *Exploring steganography: Seeing the unseen* Computer, vol.31, no. 2, hal. 26–34, Feb 1998

[2] Mstafa, R. (2014). A Highly Secure Video steganography using hamming code (7.4). *In Proceeding of systems, Applications and Technology Conference (LISAT)* (p. 6845191). New York: IEEE.

[3] Fajri, G. (2017). Reversible Audio Steganography menggunakan metode RDE dan SVM. *Tugas Akhir Jurusan Teknik Informatika ITS*.

[4] Chobitkar, A. (2016). An Approach to Video Steganography Using LSB. *Satellite Conference ICSTSD 2016 International Conference on Science and Technology for Sustainable Development, Kuala Lumpur* (p. 75). Malaysia: Satellite Conference ICSTSD.

[5] Saputra, J. (2017). Steganography Embedding plain text to picture with end of file. *Tugas Akhir Jurusan Matematika ITS*

[6] Timur, T. (2017). Kajian Kode Hamming dan Simulasinya Menggunakan Sage.*Tugas Akhir, Jurusan Matematika ITS*

[7] Lidl, R & Pilz, G. (1997). *Applied Abstract Algebra.* USA: Library of Congress Cataloging-in-Publication Data.

[8] B.Vucetic. (2000). *TURBO CODES : Principles and Applications.* Sydney , Australia: Springer Science+Business Media, LLC.

[9] Chris Veness. SHA 256. Github – chrisveness/crypto

[10] Affeldt, R. & Garrigue, J.(2015). *Formalization of error correcting codes: from Hamming to modern coding theory*,Nanjing, China.
[11] Boehm, B.(2014). StegExpose- A Tool for Detecting LSB Steganography : github.com/b3dk7/StegExpose
[12] github.com/varunon9/ImageStegano/tree/master/dist

Lampiran A
Kode Program untuk proses Encoding

```
import cv2
import numpy
import LSB
from enc import enc
import hashlib

gambar_rahasia = 'test.bmp'
gambar_carrier = 'carrier.bmp'
gambar_hasil = 'new_image.bmp'

# data
img = cv2.imread(gambar_rahasia, cv2.IMREAD_GRAYSCALE)
h = img.shape[0]
w = img.shape[1]

sha_signature = hashlib.sha256(img).hexdigest()
print('SHA256: ', sha_signature)

# ubah piksel dari desimal ke 11 digit binary (string)
img_bin = []
for i in range(0, h):
    img_bin2 = []
    for j in range(0, w):
        img_bin2.append('{0:011b}'.format(img[i, j]))
    img_bin.append(img_bin2)

# encode menggunakan dictionary enc (code hamming)
menjadi 15 digit binary (string)
encoded_img = []
for i in range(0, h):
    encoded_img2 = []
    for j in range(0, w):
        encoded_img2.append(enc[img_bin[i][j]])
    encoded_img.append(encoded_img2)

# test (untuk debug)
#     piksel asli       piksel 11 bit       piksel 15
bit
print(img[0][0],        img_bin[0][0],
encoded_img[0][0])
```

```
print(img[h-1][w-1],    img_bin[h-1][w-1],
encoded_img[h-1][w-1])

# ubah dari 15 digit binary (string) ke uint16
img_bin16U = img.astype(numpy.uint16)
for i in range(0, h):
    for j in range(0, w):
        img_bin16U[i][j] = int(encoded_img[i][j], 2)

# ubah ke 3 channel
img_bin16U = cv2.merge((img_bin16U, img_bin16U,
img_bin16U))

# LSB steganography encode
carrier = cv2.imread(gambar_carrier)
LSB.encode(img_bin16U, carrier, 1, gambar_hasil)

# tampilkan gambar (untuk debug)
result = cv2.imread(gambar_hasil)
cv2.imshow('image', img)
cv2.imshow('carrier', carrier)
cv2.imshow('result', result)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Lampiran B
Kode Program Decoding

```
import cv2
import numpy
import LSB
from dec import dec
import hashlib

gambar_hasil = 'new_image.bmp'

# data
steg = cv2.imread(gambar_hasil)
h = steg.shape[0]
w = steg.shape[1]

# LSB steganography decode
img, carrier = LSB.decode(steg, 1)

# ubah ke 1 channel
img, _, _ = cv2.split(img)

# ubah piksel dari desimal ke 15 digit binary (string)
img_bin = []
for i in range(0, h):
    img_bin2 = []
    for j in range(0, w):
        img_bin2.append('{0:015b}'.format(img[i, j]))
    img_bin.append(img_bin2)

# decode menggunakan dictionary dec (code hamming)
menjadi 11 digit binary (string)
h = 20
w = 113
encoded_img = []
for i in range(0, h):
    encoded_img2 = []
    for j in range(0, w):
        if img_bin[i][j] in dec:
            encoded_img2.append(dec[img_bin[i][j]])
        else:
            encoded_img2.append('11111111')
    encoded_img.append(encoded_img2)
```

```
h = len(encoded_img)
w = len(encoded_img[0])
gambar_rahasia = numpy.zeros((h, w), numpy.uint8)
for i in range(0, h):
    for j in range(0, w):
        gambar_rahasia[i][j] = int(encoded_img[i][j],
2)

sha_signature =
hashlib.sha256(gambar_rahasia).hexdigest()
print('SHA256: ', sha_signature)

cv2.imshow('gambar_rahasia', gambar_rahasia)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Lampiran C
Kode Program LSB (*Least Significant Bit*)

```
import cv2, copy, time

def int_to_bin(number):
    return "{0:b}".format(number).zfill(8)

def change_bits(secret_px_bin_val, cover_px_bin_val,
bit_count):
    return cover_px_bin_val[:-bit_count] +
secret_px_bin_val[:bit_count]

def extract_bits(steg_px_bin_val, bit_count):
    return steg_px_bin_val[-bit_count:]

def extract_cover_bits(cover_px_bin_val, bit_count):
    return cover_px_bin_val[:-bit_count]

def normalize_secret_bits(extract_steg_px_bin_val):
    bit_val = extract_steg_px_bin_val.ljust(8,'0')
    extract_steg_px_int_to_bin_val =
bin_to_int(bit_val)
    return extract_steg_px_int_to_bin_val

def normalize_cover_bits(extract_cover_px_bin_val):
    bit_val = extract_cover_px_bin_val.zfill(8)
    extract_cover_px_val = bin_to_int(bit_val)
    return extract_cover_px_val

def bin_to_int(bin_value):
    return int(bin_value, 2)

#this function change the cover pixel value using
secret pixel value
def change_lsb(secret_px, cover_px, bit_count):

    #RGB binary values of secret image pixel
    secret_px_b_val = int_to_bin(secret_px[0])
    secret_px_g_val = int_to_bin(secret_px[1])
    secret_px_r_val = int_to_bin(secret_px[2])

    #RGB binary values of cover image pixel
```

```
    cover_px_b_val = int_to_bin(cover_px[0])
    cover_px_g_val = int_to_bin(cover_px[1])
    cover_px_r_val = int_to_bin(cover_px[2])

    #change lsb using bit_count
    cover_px_b_bin_val = change_bits(secret_px_b_val,
cover_px_b_val, bit_count)
    cover_px_g_bin_val = change_bits(secret_px_g_val,
cover_px_g_val, bit_count)
    cover_px_r_bin_val = change_bits(secret_px_r_val,
cover_px_r_val, bit_count)

    #apply calculated RGB value to the pixel
    cover_px[0] = bin_to_int(cover_px_b_bin_val)
    cover_px[1] = bin_to_int(cover_px_g_bin_val)
    cover_px[2] = bin_to_int(cover_px_r_bin_val)

    return cover_px

def extract_secret(steg_px, bit_count):

    #RGB binary values of steg image pixel
    steg_px_b_val = int_to_bin(steg_px[0])
    steg_px_g_val = int_to_bin(steg_px[1])
    steg_px_r_val = int_to_bin(steg_px[2])

    #extract lower bits from steg image
    extract_steg_px_b_val =
extract_bits(steg_px_b_val, bit_count)
    extract_steg_px_g_val =
extract_bits(steg_px_g_val, bit_count)
    extract_steg_px_r_val =
extract_bits(steg_px_r_val, bit_count)

    #normalize steg pixel bits for secret image
    norm_extract_steg_px_b_val =
normalize_secret_bits(extract_steg_px_b_val)
    norm_extract_steg_px_g_val =
normalize_secret_bits(extract_steg_px_g_val)
    norm_extract_steg_px_r_val =
normalize_secret_bits(extract_steg_px_r_val)

    #apply calculated RGB value to the pixel
    steg_px[0] = norm_extract_steg_px_b_val
```

```
    steg_px[1] = norm_extract_steg_px_g_val
    steg_px[2] = norm_extract_steg_px_r_val

    return steg_px

def extract_cover(cover_px, bit_count):

    #RGB binary values of steg image pixel
    cover_px_b_val = int_to_bin(cover_px[0])
    cover_px_g_val = int_to_bin(cover_px[1])
    cover_px_r_val = int_to_bin(cover_px[2])

    #extract upper bits from steg image
    extract_cover_px_b_val =
extract_cover_bits(cover_px_b_val, bit_count)
    extract_cover_px_g_val =
extract_cover_bits(cover_px_g_val, bit_count)
    extract_cover_px_r_val =
extract_cover_bits(cover_px_r_val, bit_count)

    #normalize steg pixel bits for cover image
    norm_extract_cover_px_b_val =
normalize_secret_bits(extract_cover_px_b_val)
    norm_extract_cover_px_g_val =
normalize_secret_bits(extract_cover_px_g_val)
    norm_extract_cover_px_r_val =
normalize_secret_bits(extract_cover_px_r_val)

    #apply calculated RGB value to the pixel
    cover_px[0] = norm_extract_cover_px_b_val
    cover_px[1] = norm_extract_cover_px_g_val
    cover_px[2] = norm_extract_cover_px_r_val

    return cover_px

#this function embed secret image array into cover
image array using LSB
def embed_image(secret_px_array, cover_px_array,
bit_count):
    cover_px_array_new = copy.deepcopy(cover_px_array)
    secret_px_array_new =
copy.deepcopy(secret_px_array)
    for i in range(len(secret_px_array_new)):
        for j in range(len(secret_px_array_new[0])):
```

```
            cover_px_array_new[i][j] =
change_lsb(secret_px_array_new[i][j],
cover_px_array_new[i][j], bit_count)
    return cover_px_array_new

def decode_images(steg_image_px_array, bit_count):
    secret_image_px_array_new =
copy.deepcopy(steg_image_px_array)
    cover_image_px_array_new =
copy.deepcopy(steg_image_px_array)
    for i in range(len(secret_image_px_array_new)):
        for j in
range(len(secret_image_px_array_new[0])):
            secret_image_px_array_new[i][j] =
extract_secret(secret_image_px_array_new[i][j],
bit_count)
            cover_image_px_array_new[i][j] =
extract_cover(cover_image_px_array_new[i][j],
bit_count)
    return secret_image_px_array_new,
cover_image_px_array_new

#steg_image = embed_image(secret_px_array,
cover_px_array, bit_count)
#extracted_secret_image, extracted_cover_image =
decode_images(steg_image_px_array, bit_count)

def encode(secret, cover, bit_count, filename):
    s_rows, s_cols, s_channels = secret.shape
    c_rows, c_cols, c_channels = cover.shape

    #get RGB values of cover & secret
    secret_px_array = secret[0:s_rows, 0:s_cols]
    cover_px_array = cover[0:c_rows, 0:c_cols]
    print('Pixel values extracted.')

    steg_image = embed_image(secret_px_array,
cover_px_array, bit_count)

    print('Steg image created.')

    print('Saving as '+filename+'.')

    cv2.imwrite(filename, steg_image)
```

```
    print('Done.')

def decode(steg, bit_count):
    s_rows, s_cols, s_channels = steg.shape

    #get RGB values of steg
    steg_image_px_array = steg[0:s_rows, 0:s_cols]
    print('Pixel values extracted.')

    extracted_secret_image, extracted_cover_image =
decode_images(steg_image_px_array, bit_count)

    print('Secret image & cover image created.')

    print('Done.')

    return extracted_secret_image,
extracted_cover_image
```

Lampiran D
Tabel ASCII (*American Standard Code for Information Interchange*)

Symbol	HEX	DEC	Description
NUL	0	0	Null char
SOH	1	1	Start of Heading
STX	2	2	Start of Text
ETX	3	3	End of Text
EOT	4	4	End of Transmission
ENQ	5	5	Enquiry
ACK	6	6	Acknowledgment
BEL	7	7	Bell
BS	8	8	Back Space
HT	9	9	Horizontal Tab
LF	0A	10	Line Feed
VT	0B	11	Vertical Tab
FF	0C	12	Form Feed
CR	0D	13	Carriage Return
SO	0E	14	Shift Out / X-On
SI	0F	15	Shift In / X-Off
DLE	10	16	Data Line Escape
DC1	11	17	Device Control 1 (oft. XON)
DC2	12	18	Device Control 2
DC3	13	19	Device Control 3 (oft. XOFF)
DC4	14	20	Device Control 4
NAK	15	21	Negative Acknowledgement
SYN	16	22	Synchronous Idle
ETB	17	23	End of Transmit Block

Symbol	HEX	DEC	Description
CAN	18	24	Cancel
EM	19	25	End of Medium
SUB	1A	26	Substitute
ESC	1B	27	Escape
FS	1C	28	File Separator
GS	1D	29	Group Separator
RS	1E	30	Record Separator
US	1F	31	Unit Separator
	20	32	Space
!	21	33	Exclamation mark
"	22	34	Double quotes (or speech marks)
#	23	35	Number
$	24	36	Dollar
%	25	37	Procenttecken
&	26	38	Ampersand
'	27	39	Single quote
(	28	40	Open parenthesis (or open bracket)
)	29	41	Close parenthesis (or close bracket)
*	2A	42	Asterisk
+	2B	43	Plus
,	2C	44	Comma
-	2D	45	Hyphen
.	2E	46	Period, dot or full stop
/	2F	47	Slash or divide
0	30	48	Zero
1	31	49	One
2	32	50	Two
3	33	51	Three

Symbol	HEX	DEC	Description
4	34	52	Four
5	35	53	Five
6	36	54	Six
7	37	55	Seven
8	38	56	Eight
9	39	57	Nine
:	3A	58	Colon
;	3B	59	Semicolon
<	3C	60	Less than (or open angled bracket)
=	3D	61	Equals
>	3E	62	Greater than (or close angled bracket)
?	3F	63	Question mark
@	40	64	At symbol
A	41	65	Uppercase A
B	42	66	Uppercase B
C	43	67	Uppercase C
D	44	68	Uppercase D
E	45	69	Uppercase E
F	46	70	Uppercase F
G	47	71	Uppercase G
H	48	72	Uppercase H
I	49	73	Uppercase I
J	4A	74	Uppercase J
K	4B	75	Uppercase K
L	4C	76	Uppercase L
M	4D	77	Uppercase M
N	4E	78	Uppercase N
O	4F	79	Uppercase O

Symbol	HEX	DEC	Description
P	50	80	Uppercase P
Q	51	81	Uppercase Q
R	52	82	Uppercase R
S	53	83	Uppercase S
T	54	84	Uppercase T
U	55	85	Uppercase U
V	56	86	Uppercase V
W	57	87	Uppercase W
X	58	88	Uppercase X
Y	59	89	Uppercase Y
Z	5A	90	Uppercase Z
[	5B	91	Opening bracket
\	5C	92	Backslash
]	5D	93	Closing bracket
^	5E	94	Caret - circumflex
_	5F	95	Underscore
`	60	96	Grave accent
a	61	97	Lowercase a
b	62	98	Lowercase b
c	63	99	Lowercase c
d	64	100	Lowercase d
e	65	101	Lowercase e
f	66	102	Lowercase f
g	67	103	Lowercase g
h	68	104	Lowercase h
i	69	105	Lowercase i
j	6A	106	Lowercase j
k	6B	107	Lowercase k

Symbol	HEX	DEC	Description
l	6C	108	Lowercase l
m	6D	109	Lowercase m
n	6E	110	Lowercase n
o	6F	111	Lowercase o
p	70	112	Lowercase p
q	71	113	Lowercase q
r	72	114	Lowercase r
s	73	115	Lowercase s
t	74	116	Lowercase t
u	75	117	Lowercase u
v	76	118	Lowercase v
w	77	119	Lowercase w
x	78	120	Lowercase x
y	79	121	Lowercase y
z	7A	122	Lowercase z
{	7B	123	Opening brace
|	7C	124	Vertical bar
}	7D	125	Closing brace
~	7E	126	Equivalency sign - tilde
	7F	127	Delete
€	80	128	Euro sign
	81	129	
‚	82	130	Single low-9 quotation mark
ƒ	83	131	Latin small letter f with hook
„	84	132	Double low-9 quotation mark
…	85	133	Horizontal ellipsis
†	86	134	Dagger

Symbol	HEX	DEC	Description
‡	87	135	Double dagger
ˆ	88	136	Modifier letter circumflex accent
‰	89	137	Per mille sign
Š	8A	138	Latin capital letter S with caron
‹	8B	139	Single left-pointing angle quotation
Œ	8C	140	Latin capital ligature OE
	8D	141	
Ž	8E	142	Latin captial letter Z with caron
	8F	143	
	90	144	
‘	91	145	Left single quotation mark
’	92	146	Right single quotation mark
“	93	147	Left double quotation mark
”	94	148	Right double quotation mark
•	95	149	Bullet
–	96	150	En dash
—	97	151	Em dash
˜	98	152	Small tilde
™	99	153	Trade mark sign
š	9A	154	Latin small letter S with caron
›	9B	155	Single right-pointing angle quotation mark
œ	9C	156	Latin small ligature oe
	9D	157	
ž	9E	158	Latin small letter z with caron
Ÿ	9F	159	Latin capital letter Y with diaeresis
	A0	160	Non-breaking space

Symbol	HEX	DEC	Description
¡	A1	161	Inverted exclamation mark
¢	A2	162	Cent sign
£	A3	163	Pound sign
¤	A4	164	Currency sign
¥	A5	165	Yen sign
¦	A6	166	Pipe, Broken vertical bar
§	A7	167	Section sign
¨	A8	168	Spacing diaeresis - umlaut
©	A9	169	Copyright sign
ª	AA	170	Feminine ordinal indicator
«	AB	171	Left double angle quotes
¬	AC	172	Not sign
	AD	173	Soft hyphen
®	AE	174	Registered trade mark sign
¯	AF	175	Spacing macron - overline
°	B0	176	Degree sign
±	B1	177	Plus-or-minus sign
²	B2	178	Superscript two - squared
³	B3	179	Superscript three - cubed
´	B4	180	Acute accent - spacing acute
µ	B5	181	Micro sign
¶	B6	182	Pilcrow sign - paragraph sign
·	B7	183	Middle dot - Georgian comma
¸	B8	184	Spacing cedilla
¹	B9	185	Superscript one
º	BA	186	Masculine ordinal indicator

Symbol	HEX	DEC	Description
»	BB	187	Right double angle quotes
¼	BC	188	Fraction one quarter
½	BD	189	Fraction one half
¾	BE	190	Fraction three quarters
¿	BF	191	Inverted question mark
À	C0	192	Latin capital letter A with grave
Á	C1	193	Latin capital letter A with acute
Â	C2	194	Latin capital letter A with circumflex
Ã	C3	195	Latin capital letter A with tilde
Ä	C4	196	Latin capital letter A with diaeresis
Å	C5	197	Latin capital letter A with ring above
Æ	C6	198	Latin capital letter AE
Ç	C7	199	Latin capital letter C with cedilla
È	C8	200	Latin capital letter E with grave
É	C9	201	Latin capital letter E with acute
Ê	CA	202	Latin capital letter E with circumflex
Ë	CB	203	Latin capital letter E with diaeresis
Ì	CC	204	Latin capital letter I with grave
Í	CD	205	Latin capital letter I with acute
Î	CE	206	Latin capital letter I with circumflex
Ï	CF	207	Latin capital letter I with diaeresis
Ð	D0	208	Latin capital letter ETH
Ñ	D1	209	Latin capital letter N with tilde
Ò	D2	210	Latin capital letter O with grave
Ó	D3	211	Latin capital letter O with acute
Ô	D4	212	Latin capital letter O with circumflex

Symbol	HEX	DEC	Description
Õ	D5	213	Latin capital letter O with tilde
Ö	D6	214	Latin capital letter O with diaeresis
×	D7	215	Multiplication sign
Ø	D8	216	Latin capital letter O with slash
Ù	D9	217	Latin capital letter U with grave
Ú	DA	218	Latin capital letter U with acute
Û	DB	219	Latin capital letter U with circumflex
Ü	DC	220	Latin capital letter U with diaeresis
Ý	DD	221	Latin capital letter Y with acute
Þ	DE	222	Latin capital letter THORN
ß	DF	223	Latin small letter sharp s - ess-zed
à	E0	224	Latin small letter a with grave
á	E1	225	Latin small letter a with acute
â	E2	226	Latin small letter a with circumflex
ã	E3	227	Latin small letter a with tilde
ä	E4	228	Latin small letter a with diaeresis
å	E5	229	Latin small letter a with ring above
æ	E6	230	Latin small letter ae
ç	E7	231	Latin small letter c with cedilla
è	E8	232	Latin small letter e with grave
é	E9	233	Latin small letter e with acute
ê	EA	234	Latin small letter e with circumflex
ë	EB	235	Latin small letter e with diaeresis
ì	EC	236	Latin small letter i with grave
í	ED	237	Latin small letter i with acute
î	EE	238	Latin small letter i with circumflex

Symbol	HEX	DEC	Description
ï	EF	239	Latin small letter i with diaeresis
ð	F0	240	Latin small letter eth
ñ	F1	241	Latin small letter n with tilde
ò	F2	242	Latin small letter o with grave
ó	F3	243	Latin small letter o with acute
ô	F4	244	Latin small letter o with circumflex
õ	F5	245	Latin small letter o with tilde
ö	F6	246	Latin small letter o with diaeresis
÷	F7	247	Division sign
ø	F8	248	Latin small letter o with slash
ù	F9	249	Latin small letter u with grave
ú	FA	250	Latin small letter u with acute
û	FB	251	Latin small letter u with circumflex
ü	FC	252	Latin small letter u with diaeresis
ý	FD	253	Latin small letter y with acute
þ	FE	254	Latin small letter thorn
ÿ	FF	255	Latin small letter y with diaeresis

“Halaman ini sengaja dikosongkan”

BIODATA PENULIS

Penulis memiliki nama lengkap Ridwanul Fata, lahir di Lampung Utara pada tanggal 4 Juli 1994. Penulis Merupakan anak pertama dari 4 bersaudara dari Atqiya Zaini dan Siti Rofiatin. Penulis berasal dari Bengkulu, bertempat tinggal di Desa Sukasari RT.13/RW.03 Kecamatan Air Periukan, Kabupaten Seluma, Bengkulu. Pendidikan formal yang pernah ditempuh yaitu SDN 25 Air Periukan,MTs Ja-alHaq Bengkulu, dan MA Ja-alHaq Bengkulu. Kemudian, penulis melanjutkan studi di jurusan Matematika ITS melalui jalur Program Beasiswa Santri Berprestasi dengan NRP 1212100702/06111240007002 dengan bidang minat ilmu komputer. Dalam bidang minat ini penulis mulai mengenal bahasa pemrograman diantaranya adalah C++, Java, PHP-MySQL, MATLAB dan Python. Semasa menempuh jenjang pendidikan S-1, penulis juga aktif dalam kegiatan non-akademis diantaranya aktif di organisasi kemahasiswaan Matematika ITS sebagai Staff di departemen Sains dan Teknologi dan CSS MoRA ITS sebagai staff DAGRI serta mengikuti kepanitiaan acara besar yang ada di ITS diantaranya: Olimpiade Matematika ITS(OMITS). Selama penulisan Tugas Akhir ini Penulis tidak lepas dari kekurangan, untuk itu penulis mengharapkan kritik, saran, dan pertanyaan mengenai Tugas Akhir ini yang dapat dikirimkan melalui *e-mail* ke ridwanulfata@gmail.com.