IMPLEMENTASI STEGANOGRAFI MENGGUNAKAN METODE SPREAD SPECTRUM DALAM PENGAMANAN DATA TEKS PADA CITRA DIGITAL

SKRIPSI

MERI SINAGA

71153019

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUMATERA UTARA

MEDAN

2020

IMPLEMENTASI STEGANOGRAFI MENGGUNAKAN METODE SPREAD SPECTRUM DALAM PENGAMANAN DATA TEKS PADA CITRA DIGITAL

SKRIPSI

Diajukan Untuk Memenuhi Syarat Mencapai Gelar Sarjana Komputer

MERI SINAGA

71153019

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUMATERA UTARA

MEDAN

2020

PERSETUJUAN SKRIPSI

Hal : Surat Persetujuan Skripsi
Lamp : -

Kepada Yth.,
Dekan Fakultas Sains dan Teknologi
Universitas Islam Negeri Sumatera Utara Medan

Assalamu'alaikum Wr. Wb

Setelah membaca, meneliti, memberikan petunjuk dan mengoreksi serta mengadakan perbaikan, maka kami selaku pembimbing berpendapat bahwa skripsi saudari,

Nama	: Meri Sinaga
Nomor Induk Mahasiswa	: 71153019
Program Studi	: Ilmu Komputer
Judu	: Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital

dapat disetujui untuk segera di *munaqasyahkan.* Atas perhatiannya kami ucapkan terikasih

Medan, 07 Agustus 2020
17 Zulhijah 1441

Komisi Pembimbing,

Pembimbing Skripsi I,	Pembimbing Skripsi II,
Dr. Mhd. Furqan, S.Si, M. Comp. Sc	Yusuf Ramadhan Nasution, M.Kom
NIP. 198008062006041003	NIB. 1100000075

SURAT PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini;

Nama	: Meri Sinaga
Nim	: 71153019
Program Studi	: Ilmu Komputer
Judul	: Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital.

Menyatakan bahwa skripsi ini hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing disebutkan sumbernya. Apabila dikemudian hari ditemukan plagiat dalam skripsi ini maka saya bersedia menerima sanksi pencabutan gelar akademik yang saya proleh dan sanksi lainnya sesuai dengan peraturan yang berlaku.

Medan, 07 Agustus 2020

Meri Sinaga
NIM. 71153019

KEMENTERIAN AGAMA REPUBLIK INDONESIA
UNIVERSITAS ISLAM NEGERI SUMATERA UTARA MEDAN
FAKULTAS SAINS DAN TEKNOLOGI
Jl. IAIN No. 1 Medan 20235
Telp. (061) 6615683-6622925, Fax. (061) 6615683
Url: http://saintek.uinsu.ac.id, E-mail: saintek@uinsu.ac.id

PENGESAHAN SKRIPSI
Nomor: 010/ST/ST.V/PP.01.1/01/2021

Judul	: Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital
Nama	: Meri Sinaga
Nomor Induk Mahasiswa	: 71153019
Program Studi	: Ilmu Komputer
Fakultas	: Sains dan Teknologi

Telah dipertahankan di Hadapan Dewan Penguji Skripsi Program Studi Ilmu Komputer Fakultas Sain dan Teknologi UIN Sumatera Utara Medan dan Dinyatakan **LULUS**.

Pada hari/tanggal	: Jumat, 07 Agustus 2020
Tempat/media	: Via Zoom Meeting

Tim Ujian Munaqasyah
Ketua,

Dr. Mhd. Furqan, S.Si, M.Comp.Sc
NIP. 198008062006041003

Dewan Penguji

Penguji I	Penguji II
Dr. Mhd. Furqan, S.Si, M.Comp.Sc	Yusuf Ramadhan Nasution, M.Kom
NIP. 198008062006041003	NIB. 1100000075
Penguji III	Penguji IV
Rakhmat Kurniawan R, S.T, M.Kom	Armansyah, M.Kom
NIP. 198503162015031003	NIB. 1100000074

Mengesahkan
Dekan Fakultas Sains dan Teknologi
UIN Sumatera Utara Medan

Dr. H. M. Jamil, M,A
NIP. 196609101999031002

ABSTRAK

Berbagai macam teknik digunakan untuk melindungi informasi digital terutama informasi yang dirahasiakan dari orang yang tidak berhak terhadap hak akses informasi tersebut, salah satunya adalah dengan teknik steganografi. Teknik steganografi adalah sebuah teknik yang digunakan untuk mengamankan data dengan cara menyisipkan atau menyembunyikan data kedalam sebuah objek tanpa mengubah bentuk objek tersebut, salah satu objeknya adalah citra digital. Teknik steganografi juga memiliki beberapa metode yaitu metode *Spread Spectrum*. *Spread Spectrum* adalah metode komunikasi dimana sinyal informasi disebar diseluruh frekuensi yang tersedia dengan memilih tempat penyisipan data pada frekuensi yang rendah serta menambahkan *pseudo-noise* (PN). Penelitian ini mengamankan data teks dengan teknik steganografi *Spread Spectrum* pada objek citra digital dengan tahapan mengubah nilai RGB *pixel* citra dan data teks kedalam biner, melakukan *spreading* pada data teks, dilanjutkan dengan pembangkitan kunci serta proses modulasi hasil *spreading* data teks dengan kunci hasil pembangkitan. Sehingga menghasilkan citra stegano dengan nilai RGB *pixel* yang mengalami perubahan nilai 0 hingga 1 nilai yang tidak mempengaruhi reproduksi warna RGB *pixel* citra.

Kata Kunci: *Citra, Teks, Steganografi, Spread Spectrum*

ABSTRACT

Various techniques are used to protect digital information, especially information that is kept secret from people who are not have the right of access to the information, one of which is the technique of steganography. Steganography is a technique used to secure data by inserting or hiding data into an object without change the shape of the object, one of the objects is a digital image. Steganography technique also has several methods, namely the Spread Spectrum method. Spread Spectrum is a method of communication where information signals are spread across all available frequencies by selecting where to insert data at low frequencies and adding pseudo-noise (PN). This study secures text data with the Spread Spectrum steganography technique on digital image objects by the steps of changing the value of the RGB pixel image and text data into binary, spreading the text data, followed by key generation and the modulation process of spreading text data with the generated key results. Then to produce a stegano image with an RGB pixel value that changes the value of 0 to 1 value that does not affect the color reproduction of the RGB pixel image.

***Keywords**: Image, Text, Steganography, Spread Spectrum*

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Segala puji dan syukur penulis panjatkan kehadirat Allah SWT, karena dengan izin-Nya lah penulis dapat menyelesaikan Skripsi ini. Shalawat beserta salam semoga senantiasa tercurah kepada junjungan penulis, yakni Nabi Besar Muhammad SAW.

Adapun penulis akhirnya dapat menyelesaikan skripsi ini berkat bimbingan serta bantuan dari berbagai pihak dengan judul skripsi ***"Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital".***

Untuk itu penulis menyadari dan pada kesempatan ini penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Bapak Prof. Dr. Saidurrahman, M.Ag, selaku Rektor Universitas Islam Negeri Sumatera Utara.
2. Bapak Dr. H. M. Jamil, MA, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sumatera Utara.
3. Bapak Dr. Mhd Furqan, S,Si., M.Comp.Sc selaku Ketua Jurusan Ilmu Komputer dan selaku dosen pembimbing skripsi I serta pembimbing akademik yang telah membantu menyelesaikan skripsi ini.
4. Bapak Yusuf Ramadhan Nasution, M.Kom sebagai dosen pembimbing skripsi II yang telah membantu menyelesaikan skripsi ini.
5. Bapak Rakhmat Kurniawan R, ST, M.Kom selaku Kepala Laboraturium Fakultas Sains Dan Teknologi Universitas Islam Negeri Sumatera Utara Medan.
6. Kedua orang tua penulis Bapak Muhammad Samin dan Ibu Siti Aisah yang rela mengorbankan apapun demi kelancaran penulis dalam penyelesaian pendidikan.
7. Kepada Kakak dan Abang kandung penulis, Riandra, Lina Daniati, Chandra dan Jurmiah terimakasih untuk dukungan, doa dan semangatnya yang selalu

diberikan untuk penulis serta teman-teman seperjuangan yang ikut memberikan semangat terhadap penulis.

Semoga hasil dari penulisan skripsi ini ada manfaatnya bagi pihak yang membutuhkan, untuk itu penulis mengharapkan kritik dan saran dari pembaca yang bersifat membangun. Dan mudah-mudahan skripsi ini dapat diterima dan dapat membawa manfaat yang besar bagi pembacanya. Atas perhatian dan kerja sama yang baik, penulis ucapkan banyak terimakasih.

Medan, Agustus 2020

Hormat Saya,

Meri Sinaga

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

Lampiran **Judul Lampiran**

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan teknologi informasi telah memberikan kemudahan dalam menyelesaikan pekerjaan manusia. Pertukaran data dan informasi menjadi lebih mudah dan cepat. Namun disisi lain juga memiliki kekurangan, yaitu informasi yang dikirim dapat dengan mudah dicuri oleh oknum yang tidak bertanggung jawab. Berbagai macam teknik digunakan untuk melindungi informasi digital terutama informasi yang dirahasiakan dari orang yang tidak berhak terhadap hak akses informasi tersebut, maka diperlukan suatu cara untuk mengamankan data dan informasi. Diantaranya adalah dengan cara menyembunyikan data tersebut kedalam sebuah objek yang tidak dapat dicurigai dengan teknik steganografi.

Didalam Al-qur'an terdapat ayat yang membahas tentang mencuri hak orang lain.

حَكِيمٌ عَزِيزٌ وَاللَّهُ اللَّهِ مِنَ نَكَالًا كَسَبَا بِمَا جَزَاءً أَيْدِيَهُمَا فَاقْطَعُوا وَالسَّارِقَةُ وَالسَّارِقُ

"Laki-laki yang mencuri dan perempuan yang mencuri, potonglah tangan keduanya (sebagai) pembalasan bagi apa yang mereka kerjakan dan sebagai siksaan dari Allah. Dan Allah Maha Perkasa lagi Maha Bijaksana" (QS. Al Maidah: 38) (Tafsirq, 2020).

Berdasarkan pada masalah keamanan data yang dapat merugikan pihak yang memiliki otoritas, solusi yang diberikan adalah dengan memanfaatkan sebuah teknik steganografi. Teknik steganografi adalah sebuah teknik yang digunakan untuk mengamankan data dengan cara menyisipkan atau menyembunyikan data kedalam sebuah objek tanpa mengubah bentuk objek tersebut (Hafiz, 2019). Sehingga data yang disisipkan kedalam objek tidak dapat dicurigai oleh pihak yang tidak bertanggung jawab.

Proses penyisipan teknik steganografi juga membutuhkan sebuah perhitungan menggunakan metode dan harus memiliki jenis objek yang akan dijadikan wadah penyisipan data teks. Salah satunya adalah teknik steganografi dengan metode *Spread Spectrum.* Teknik *spread spectrum* adalah metode di mana sinyal (misalnya, sinyal listrik, elektromagnetik, atau akustik) yang dihasilkan dengan bandwidth tertentu secara sengaja disebarkan dalam domain frekuensi, menghasilkan sinyal dengan bandwidth yang lebih luas, sehingga algoritma *Spread Spectrum* memiliki kelebihan terhadap serangan *jamming* dan *inteferensi* yang akan membuat data penyisipan tetap dapat dipresepsi apabila terjadi kerusakan sinyal. (T Sutoyo, dkk, 2009). Secara ringkas metode *Spread Spectrum* bekerja dengan menyisipkan dan menyebarkan data kedalam objek dengan melakukan perhitungan modulasi terlebih dahulu (Anti, dkk, 2017).

Jenis objek yang diterapkan pada penelitian adalah objek citra digital. Pemilihan objek citra digital bertujuan untuk mengurangi rasa curiga pihak yang tidak memiliki otoritas dalam mengambil informasi data teks. Hal ini dilandasi berdasarkan komunikasi modern saat ini yang salah satunya adalah komunikasi visual menggunakan citra digital. Sehingga penelitian ini akan menyisipkan pesan data teks kedalam sebuah citra digital menggunakan metode *Spread Spectrum*.

Berdasarkan urain latar belakang di atas, maka penulis menetapkan judul penelitian ini adalah ***"Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital".***

1.2 Rumusan Masalah

Berdasarkan uraian dalam latar belakang, maka rumusan masalah dalam penelitian ini adalah :

1. Bagaimana menerapkan metode *Spread Spectrum* untuk menyisipkan data teks kedalam citra digital ?
2. Bagaimana merancang dan membangun sistem aplikasi pengamanan data teks menggunakan metode *Spread Spectrum* ?

1.3 Batasan Masalah

Berdasarkan dari paparan rumusan masalah di atas, maka batasan masalah dalam penelitian ini adalah :

1. Pesan yang akan disisipkan berupa data teks karakter.
2. File citra digital sebagai objek penampung berformat .jpg dan hasil dari proses penyisipan berformat .jpg.
3. *Size* pesan yang akan disisipkan pada contoh manual adalah sebesar 4 *byte.*
4. Ukur data teks harus lebih kecil dari ukuran data citra.
5. Maksimal data teks yang dapat disisipkan adalah 1000 karakter.
6. *Software* yang digunakan untuk membangun aplikasi adalah *Microsoft Visual Studio* 2012 dengan bahasa pemograman *Csharp* (C#).

1.4 Tujuan Penelitian

Berdasarkan paparan dari perumusan dan batasan masalah di atas, adapun tujuan dari penelitian ini antara lain :

1. Menerapkan metode *Spread Spectrum* untuk menyisipkan data teks kedalam citra digital.
2. Merancang dan membangun sistem aplikasi pengamanan data teks menggunakan metode *Spread Spectrum*.

1.5 Manfaat Penelitian

Berdasarkan paparan dari tujuan di atas, adapun manfaat yang diberikan dalam penelitian ini adalah :

1. Mengurangi terjadinya manipulasi pada informasi yang terkandung pada data teks.
2. Mengetahui bagaimana cara pengamanan data teks dengan metode *spread spectrum.*
3. Dapat memberikan kontribusi bagi perkembangan ilmu pengetahuan pada khususnya, maupun masyarakat pada umunya mengenai *steganografi*.

BAB II
TINJAUAN PUSTAKA

2.1 Implementasi

Implementasi yaitu menempatkan sistem sehingga siap untuk bekerja. Langkah ini melibatkan penulisan program kode program jika paket perangkat lunak tidak digunakan. (Jogiyanto, 2005).

Fase implementasi terdiri dari fase-fase berikut:

1. Jalankan rencana implementasi

   Perencanaan implementasi merupakan kegiatan awal dalam tahap implementasi, karena kegiatan implementasi harus memiliki rencana implementasi terlebih dahulu agar dapat berfungsi sesuai dengan keinginan.

2. Kegiatan implementasi

   Kegiatan pelaksanaan dilaksanakan berdasarkan kegiatan yang direncanakan dalam rencana pelaksanaan. Kegiatan yang dapat dilakukan selama tahap implementasi adalah pemrograman dan pengujian program.

3. Implementasi selanjutnya

   Bahkan setelah diimplementasikan, tindak lanjut berikut harus dilakukan. Bahkan setelah program baru diimplementasikan, analis sistem masih perlu menguji penerimaan program.

Konsep implementasi memiliki implikasi bahwa setiap pengembangan harus dibagi beberapa tahap berbeda. Implementasi dimulai dengan tahapan analisa kebutuhan dan perencanaan yang mana harus menghasilkan susunan yang terperinci sebelum memulai tahapan tindak lanjut berikutnya. (Jogiyanto, 2005)

2.2 Keamanan

Masalah keamanan adalah salah satu dari aspek yang terpenting bagi sebuah sistem informasi. Masalah keamanan tidak luput dari kurangnya mendapatkan perhatian dpada saat para perancang dan pengolahan sistem informasi melalukan pekerjaanya serta kadang kala keamanan berada di urutan

setelah tampilan, atau bahkan ditempatkan pada urutan terakhir. Apabila terjadi gangguan peforma sistem, sering kali masalah keamanan tidak begitu dipedulikan bahkan ditiadakan (Dony,2006).

Keamanan data adalah kebebasan dari bahaya atau sebagai kondisi keselamatan.Keamanan secara rinci adalah perlindungan data di dalam sesuatu sistem melawan terhadap otorisasi tidak sah, modifikasi, atau perusakan dan perlindungan sistem komputer terhadap pengguna yang tidak bertanggung jawab (Gunawan, dkk 2018).

2.3 Steganografi

Kata steganografi berasal dari kata Yunani "stegos" yang berarti "penutup" dan "grafia" yang berarti "tulisan" mendefinisikannya sebagai "tulisan tertutup". Steganografi adalah teknik menyembunyikan data rahasia dalam file atau pesan biasa, non-rahasia untuk menghindari deteksi, data rahasia tersebut kemudian diekstrak di tujuannya. Ada banyak cara untuk menyembunyikan informasi menggunakan Steganografi. Metode steganografi bekerja dengan menanamkan informasi ke sebuah objek, contohnya adalah kedalam gambar digital. Dengan penggunaan aplikasi steganografi, peretas mengubah bit yang paling kecil dari file data dan menyematkan kode berbahaya ke dalam gambar.

Setelah pengguna yang ditargetkan mendownload dan membuka file gambar di komputer mereka, malware tersebut diaktifkan. Bergantung pada pemrogramannya, malware sekarang dapat membuka jalan bagi penyerang untuk mendapatkan kendali atas perangkat atau jaringan pengguna. Keungunalan dari teknik Steganografi adalah perbedaan antara gambar asli dan gambar steganografi tidak kentara dan keduanya tidak dapat dibedakan dengan mata telanjang (T Sutoyo, dkk, 2009).

Steganografi mengguanakn dua jenis data yaitu wadah untuk penampung dan data rahasia yang disembunyikan kedalam wadah penampung. Steganografi digital memanfaatkan media digital yang berguna sebagai wadah penampung data, contohnya adalah citra, audio, teks dan video. Data rahasia yang

disembunyikan juga dapat berupa citra, audio, teks atau video (T Sutoyo, dkk, 2009).

Berikut adalah sebuah ilustrasi dari proses setganografi dalam menyisispkan data.

Gambar 2.1 Proses Penyimpanan Data Rahasia

Sumber : T Sutoyo, dkk, 2009

2.3.1 Teknik Steganografi

Steganografi mempunyai 7 macam teknik dasar, yaitu sebagai berikut;

1. Injeksi, adalah teknik menanamkan pesan rahasia secara langsung ke dalam suatu media. Salah satu kendala pada teknik ini adalah ukuran media yang diinjeksi menjadi lebih besar dari ukuran normalnya sehingga mudah untuk dideteksi. Teknik ini sering disebut *embedding*.
2. Substitusi, data normal diganti dengan data rahasia. Biasanya hasil dari teknik ini tidak terlalu mengubah ukuran data asli, tetapi bergantung pada media file dan data yang akan disembunyikan. Teknik substitusi dapat menurunkan kualitas media yang dikendarai.
3. *Transform* domain, teknik ini sangat efektif. Pada dasarnya, domain transformasi menyembunyikan data dan mengubah ruang. Teknik ini akan sangat efektif diterapkan pada file berekstensi JPG.
4. *Spread Spectrum,* suatu teknik transmisi dengan menggunakan kode *pseudo-noise,* yang tidak bergantung pada data informasi sebagai *immodulator* bentuk gelombang untuk menyebarkan energi sinyal dalam jalur komunikasi (*bandwidth*) yang lebih besar dari sinyal jalur komunikasi informasi. Oleh penerima, sinyal dikumpulkan kembali menggunakan replika kode kebisingan semu yang disinkronkan.

5. Metode Statistik, teknik ini disebut juga dengan skema steganografi 1-bit. Skema ini dikatakan menyematkan satu bit informasi pada media perjalanan dan mengubah statistik meskipun hanya 1 bit. Perubahan statistik ditunjukkan dengan indikasi 1 dan jika terjadi perubahan maka indikasi 0. Sistem ini bekerja berdasarkan kemampuan penerima untuk membedakan informasi yang belum dimodifikasi.
6. *Distorsi*, metode ini membuat perubahan pada objek tempat data rahasia berada.
7. *Cover Generation,* metode ini unik dibandingkan metode lain karena objek sampul dipilih untuk menyembunyikan pesan. Contoh dari metode ini adalah *Spam Mimic*. (Anti, dkk, 2015)

2.3.2 Kriteria Steganografi

Menyembunyikan file atau data rahasia menjadi gambar digital akan mengubah kualitas keaslian gambar. Ini tergantung pada file media tempat Anda disimpan dan ukuran file pesan yang disisipkan. Untuk menghasilkan kerahasiaan data yang terjaga dengan baik, berikut beberapa kriteria steganografi yang baik untuk penyembunyian data, yaitu;

1. *Fidelity*

   Kualitas gambar dari wadah data tidak banyak berubah. Setelah menambahkan pesan rahasia, stego-data masih terlihat. Pengamat tidak mengetahui bahwa stego-data mengandung pesan rahasia.
2. *Rebutsness*

   Pesan tersembunyi tersebut harus mampu menahan (robust) berbagai operasi manipulasi yang dilakukan pada stego-data, seperti mengubah kontras, penajaman, *cropping*, *rotating*, memperbesar gambar, *cropping*, enkripsi, dan lain sebagainya. Jika operasi pemrosesan gambar dilakukan pada gambar *container*, pesan tersembunyi tidak boleh rusak (masih *valid* jika ditarik kembali).

3. *Revovery*

   Data rahasia yang telah disembunyikan harus dipulihkan. Karena tujuan steganografi adalah menyembunyikan informasi, setiap saat pesan rahasia di dalam stego-data harus diambil untuk digunakan lebih lanjut. (Rifai,dkk 2017)

2.3.3 Prinsip Kerja Steganografi

Kerangka kerja dari pengiriman data dalam steganography adalah data rahasia disisipkan ke sebuah data lain (*cover*) dengan menggunakan suatu kunci atau yang dikenal dengan stego-key sehingga menghasilkan data baru atau yang dikenal stego-object. Data yang menjadi media kirim harus terlihat berarti sehingga tidak menimbulkan kecurigaan pihak lain. Setelah data diterima, pihak penerima harus mengetahui stegokey agar dapat mengambil data rahasia yang telah disisipkan.

Pada hakikatnya tidak semua data dapat dijadikan cover untuk pengiriman data rahasia, diperlukan modifikasi data agar data rahasia tidak terlihat. Data yang dijadikan cover harus lebih besar agar data rahasia tidak terlihat. Data yang digunakan sebagai cover seabiknya digunakan satu kali. Apabila digunakan lebih dari satu kali, maka akan menimbulkan kecurigaan pihak lain. (Saragih, 2006)

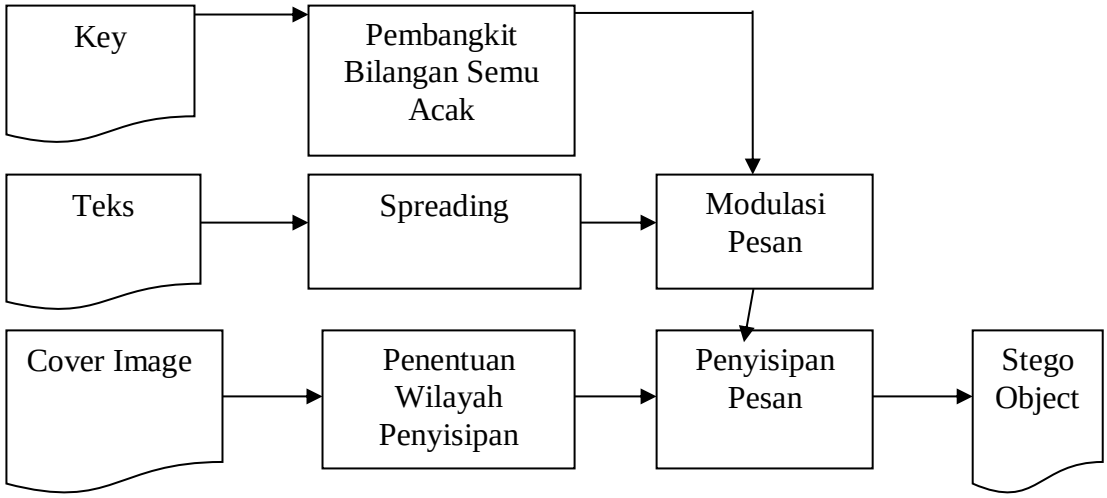
2.3.4 Metode Spread Spectrum

Teknik *spread spectrum* adalah metode di mana sinyal (misalnya, sinyal listrik, elektromagnetik, atau akustik) yang dihasilkan dengan bandwidth tertentu secara sengaja disebarkan dalam domain frekuensi, menghasilkan sinyal dengan bandwidth yang lebih luas. Teknik-teknik ini digunakan untuk berbagai alasan, termasuk membangun komunikasi yang aman, meningkatkan ketahanan terhadap gangguan alam, kebisingan, dan gangguan, untuk mencegah deteksi, untuk membatasi kerapatan fluks daya (misalnya, dalam tautan bawah satelit), dan untuk mengaktifkan banyak -Akses komunikasi.

Spread Spectrum mengacu pada sistem yang awalnya dikembangkan untuk aplikasi militer, untuk menyediakan komunikasi yang aman dengan menyebarkan sinyal melalui pita frekuensi besar. Berdasarkan definisi dapat

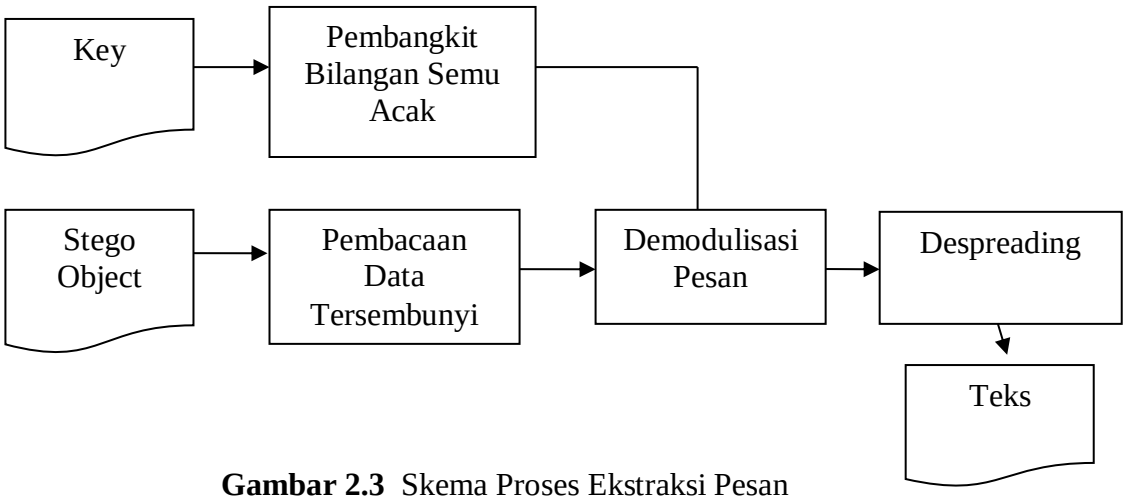
dikatakan bahwa steganografi menggunakan metode *spread spectrum* memperlakukan cover-objek baik sebagai derau (*noise*) ataupun sebagai usaha untuk menambahkan derau semu (*pseudo-noise*) ke dalam cover-objek.

Pada metode spread spectrum, penyisipan pesan atau informasi mengandung kunci yang digunakan untuk mengenkripsi pesan. Kami mendapatkan kunci melalui generator nomor pseudo acak dengan algoritma LCG (Linear Conruential Generator). Adapun skema dari metode *spread spectrum* dapat dilihat pada gambar di bawah ini:



Gambar 2.2 Skema Penyisipan Pesan
Sumber : Noercholis, 2016

Selanjutnya adalah proses ektraksi, adapun skema dari ekstraksi dapat dilihat pada di bawah ini :



Gambar 2.3 Skema Proses Ekstraksi Pesan
Sumber : Noercholis, 2016

2.4 Citra Digital

Citra adalah suatu persepsi visual yang dihasilkan dari pantulan cahaya yang menerangi objek dan sebagian dipantulkan dari berkas cahaya tersebut. Perangkat optik seperti mata manusia, kamera, dan pemindai menangkap pantulan cahaya ini, sehingga citra suatu objek yang disebut citra terekam. Secara sederhana dapat dikatakan sebagai gambar pada bidang dua dimensi.

Citra digital direpresentasikan sebagai matriks yang indeks baris dan kolomnya mengidentifikasi suatu titik pada citra dan nilai elemen matriks yang sesuai adalah tingkat warna pada titik tersebut. Elemen-elemen ini disebut elemen gambar, elemen gambar, piksel, atau pel. Resolusi gambar dalam gambar digital ditentukan oleh piksel. Semakin tinggi resolusinya, semakin kecil ukuran pikselnya, yang berarti gambar yang dihasilkan semakin halus. (Septayuda, dkk, 2014)

2.4.1 Representasi Citra Digital

Gambar adalah gambaran atau kesamaan suatu benda. Gambar analog yang tidak dapat ditampilkan di komputer, sehingga tidak dapat langsung diproses oleh komputer. Tentunya untuk bisa diproses di komputer, gambar analog harus diubah menjadi gambar digital. Citra digital merupakan citra yang dapat diolah oleh komputer. Sedangkan gambar yang dihasilkan dari peralatan digital (gambar digital) dapat langsung diproses oleh komputer. Mengapa? Pasalnya, pada perangkat digital terdapat sistem sampling dan kuantifikasi. Sedangkan peralatan analog tidak dilengkapi kedua sistem tersebut (Furqan, dkk, 2020).

Sistem pengambilan sampel merupakan sistem yang mengubah citra kontinu menjadi citra digital dengan membagi citra analog menjadi garis *M* dan kolom N, sehingga menjadi citra diskrit. Semakin tinggi nilai *M* dan *N*, gambar digital yang dihasilkan akan semakin mulus. Pertemuan antara baris dan kolom disebut piksel. Sistem kuantifikasi merupakan sistem yang mengubah intensitas analog menjadi intensitas diskrit, sehingga dengan proses ini dimungkinkan untuk membuat gradasi warna sesuai kebutuhan. Kedua sistem ini bertanggung jawab untuk memotong citra menjadi *M* baris dan *N* kolom (proses pengambilan sampel)

serta menentukan intensitas pada titik tersebut (proses kuantifikasi), untuk menghasilkan resolusi citra yang diinginkan. (Muljono, 2017). Citra juga dapat disegmentasi dengan proses clustering, kebutuhan segmentasi, dan deteksi tepi (Heri, dkk, 2019).

2.5 Data Teks

Teks data adalah sebaris data yang tersusun dari karakter berupa huruf, angka, dan simbol yang merepresentasikan hasil dari kode ASCII. Data teks dapat diidentifikasi dengan ekstensi yang tercantum dalam nama file, seperti ekstensi DOC dan TXT. Karakter tersebut mewakili semua data yang disimpan dalam format 0 dan 1. Dalam fungsi *Boolean*, 0 didefinisikan sebagai False dan 1 sebagai True. Namun, dalam kode biner, nilai 1 dan 0 disebut sebagai bit karakter.

ASCII (*American Standard Code for Information Interchange*) dan EBCDIC (*Extended Binary Coded Decimal Interchange Code*) adalah pelopor dari himpunan karakter lainnya. ASCII terdiri dari 128 karakter dengan lebar masing-masing 7 bit atau tujuh digit 0 dan 1 dari 0000000 hingga 1111111. Mengapa 7 bit? Karena komputer awalnya memiliki memori yang sangat terbatas, 128 karakter dianggap cukup untuk menampung semua huruf Latin bersela dan beberapa karakter kontrol. ASCII distandarisasi oleh ANSI (*American National Standards Institute*) sebagai standar ANSI X3.4-1986. EBCDIC adalah kumpulan karakter yang dibuat oleh IBM. Salah satu alasan IBM menggunakan set karakter non-ASCII sebagai standar pada komputer IBM adalah karena EBCDIC lebih mudah dikodekan pada kartu berlubang yang masih banyak digunakan pada tahun 1960-an. Penggunaan EBCDIC pada mainframe IBM dilakukan hingga hari ini, meskipun kartu berlubang tidak lagi digunakan. Seperti ASCII, EBCDIC juga terdiri dari 128 karakter dengan ukuran masing-masing 7 bit. Hampir semua karakter di ASCII juga ada di set karakter EBCDIC (Irawan, dkk 215).

2.6 Penyisipan Pesan

Untuk menyisipkan pesan, baik dalam pesan teks, gambar, suara maupun video, diperlukan input berupa file digital yang akan disisipkan ke dalam pesan, pesan yang akan disisipkan (pesan) dan *password*. Menurut Alatas, 2009, ada beberapa contoh cara penyisipan pesan rahasia yang digunakan dalam teknik steganografi, antara lain.

1. Teks

   Pada algoritma steganografi yang menggunakan teks sebagai media penyisipannya, teknik NLP (Natural Language Processing) sering digunakan agar teks yang telah disisipkan dengan pesan rahasia tidak mencurigakan bagi yang melihatnya.

2. Gambar

   Format gambar adalah yang paling sering digunakan, karena format ini adalah salah satu format file yang paling sering dipertukarkan di Internet. Alasan lainnya adalah banyaknya algoritma steganografi yang tersedia untuk wadah media dalam bentuk gambar.

3. Suara

   Format suara sering dipilih karena file dalam format ini umumnya berukuran relatif besar. Sehingga bisa juga menampung pesan rahasia dalam jumlah banyak.

4. Video

   Format video memang merupakan format dengan ukuran file yang relatif sangat besar, akan tetapi jarang digunakan karena ukurannya yang terlalu besar sehingga mengurangi kepraktisannya dan juga kurangnya algoritma yang mendukung format ini. (Rohayah, dkk, 2015)

2.7 Aplikasi Sistem

2.7.1 Microsoft Visual Studio 2012

Microsoft Visual Studio adalah sebuah perangkat lunak (*suite*) yang lengkap dimana aplikasi baik itu aplikasi bisnis, aplikasi pribadi atau komponen aplikasi, dapat dikembangkan dalam bentuk aplikasi konsol, aplikasi *Windows*

atau aplikasi web. *Visual Studio* mencakup kompiler, SDK, lingkungan pengembangan terintegrasi (IDE), dan dokumentasi (umumnya pustaka MSDN). Kompiler yang disertakan dalam paket Visual Studio termasuk *Visual* C ++, *Visual* C #, *Visual* Basic, *Visual Basic*.NET, *Visual InterDev*, *Visual* J ++, *Visual* J #, *Visual FoxPro*, dan *Visual SourceSafe*. *Microsoft Visual Studio* dapat digunakan untuk mengembangkan aplikasi dalam kode asli (dalam bentuk bahasa mesin yang berjalan di *Windows*) atau kode yang dikelola (dalam bentuk *Microsoft Intermediate Language* melalui .NET *Framework*). Selain itu, *Visual Studio* juga dapat digunakan untuk mengembangkan aplikasi *Silverlight* dan aplikasi *Windows Mobile* (yang berjalan di .NET *Compact Framework*) (Afriani, 2019).

2.7.2 Bahasa Pemograman C#

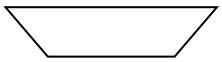
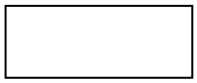
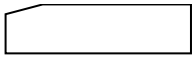
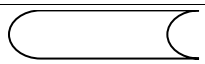
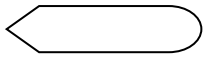
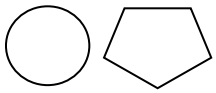
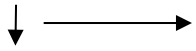
C # (C *Sharp*) adalah bahasa pemrograman berbasis objek yang digunakan oleh *Microsft* .NET *Framework*. Bahasa pemrograman ini merupakan bahasa pemrograman baru dari *Microsoft* yang dikembangkan di bawah arahan Anders Hejlsberg, yang menciptakan berbagai bahasa pemrograman diantaranya *Borland Turbo* C ++ dan *orland Delphi*. Bahasa C # distandarisasi secara internasional oleh ECMA. Seperti bahasa pemrograman lainnya, C # dapat digunakan untuk membuat berbagai jenis aplikasi, misalnya: B. Aplikasi berbasis *Windows* (*desktop*) dan aplikasi berbasis web dan aplikasi web (Purnama, dkk 2018).

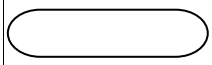
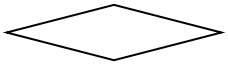
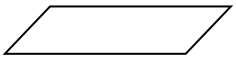
2.8 *FlowChart*

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut (Ratumurun, 2015). Tujuan utama membuat *Flowchart* adalah:

1. Menggambarkan suatu tahapan penyelesaian masalah.
2. Secara sederhana, terurai, rapi dan jelas.
3. Menggunakan simbol-simbol standar.

Tabel 2.1 Simbol-simbol *Flowchart*

No	Simbol	Nama	Keterangan
1		Kegiatan Manual	Menunjukan Pekerjaan Manual.
2		Proses	Menunjukan kegiatan atau proses dan operasi program komputer.
3		Dokumen	Menunjukan dokumen *input* dan *output* baik untuk proses manual, mekanik, komputer.
4		Kartu Plong	Menunjukan *input* atau *output* kartu plong.
5		Pita Magnetik	Menunjukan *input* atau *output* menggunakan pita magnetik.
6		*Harddisk*	Menunjukan *input* atau *output* menggunakan *harddisk*.
7		Disket	Menunjukan *input* atau *output* menggunakan disket.
8		*Display*	Menunjukan *output* yang ditampilkan ke monitor.
9		Penghubung	Menunjukan penjelasan dan penghubung ke halaman yang masih sama atau kehalaman lain.
10		Garis Alir	Menunjukan arus dan proses.

11		Pita Kertas	Menunjukan *input* atau *output* menggunakan pita kertas berlubang.
12		*Start*	Awal/akhir program.
13		Inisialisasi	Menginisialisasikan i*nput* kedalam bentuk variabel untuk diteruskan kedalam proses.
14		Keputusan	Langkah pengambilan keputusan.
15		*Input/Output*	Memasukan data /menunjukan hasil dari suatu proses.

Sumber : Ratumurun, 2015

2.9 Riset Riset Terkait

Dalam penelitian ini penulis mencantumkan beberapa hasil penelitian yang dilakukan oleh beberapa peneliti-peneliti yang telah berhasil membuat sebuah sistem atau aplikasi dalam mengamankan sebuah data rahasia dengan menggunakan metode spread spectrum, diantaranya sebagai berikut.

1. Penelitian yang dilakukan oleh Azkar Kumala, Bambang Pramono dan Rahmat Ramadhan, berdasarkan kasus uji coba yang telah dilakukan, perangkat lunak yang mengimplementasikan steganografi dengan teknik spread spectrum pada berkas audio MP3 berhasil dibangun.
2. Penelitian yang dilakukan oleh Achmad Noercholis dan Yohanes Nugraha, Pengamanan Pesan Teks Menggunakan Teknik Steganografi Spread Spectum Berbasis Android pada kesimpulannya menjelaskan berdasarkan proses pengujian yang dilakukan menggunakan 10 image dengan ukuran berbeda terdapat 1 dari 10 image tersebut mengalami perubahan bentuk image sehingga tingkat keberhasilan sistem mencapai 90%.
3. Penelitian yang dilakukan oleh Winda Winanti pada kesimpulannya telah berhasil dikembangkan perangkat lunak yang dapat melakukan steganografi

pada citra terkomprensi JPEG. Kebutuhan fungsional dari perangkat lunak, serta penggunaan kunci sudah dapat dilakukan dengan benar.

4. Penelitian yang dilakukan oleh Riko Arlando Saragih, Metode Parity Coding Versus Metode Spread Spectrum Pada Audio Steganography, berdasarkan kesimpulannya bahwa pada metode spread spectrum, nilai SNR berubah berdasarkan banyaknya data yang disisipkan. Semakin banyak data yang disisipkan, maka nilai SNR semakin kecil, tetapi keamanan data lebih terjamin karena menggunakan kode penyebar yang tidak diketahui oleh ihak lain.
5. Penelitian selanjutnya dilakukan oleh Irfan Santiko Implementasi Model Steganografi Dalam Mengelola Kerahasiaan Informasi Dengan Metode LSB, berdasarkan kesimpulannya dengan metode LSB keamanan dan kerahasiaan informasi yang disampaikan lebih terjamin karena dilengkapi dengan password untuk membuka file.

BAB III
METODE PENELITIAN

3.1 Tempat dan Waktu Penelitian

3.1.1 Tempat Penelitian

Adapun tempat peneliti melakukan penelitian dalam skripsi ini yaitu Laboratorium UINSU, Jl. IAIN No.1, Gaharu, Kec. Medan Tim., Kota Medan, Sumatera Utara 20235.

3.1.2 Waktu dan Jadwal Pelaksaan Penelitian

Waktu yang digunakan peneliti untuk penelitian ini dilaksanakan sejak tanggal dikeluarkannya ijin penelitian dalam kuran waktu bulan November s/d Januari 2020.

3.2 Bahan dan Alat Penelitian

Pada penelitian ini bahan yang digunakan adalah terbagi menjadi dua bagian yaitu bahan untuk data teks berupa file teks dengan format .txt dan bahan sebuah objek citra digital yang berformat .jpg dan .png. Bahan gambar dalam penelitian ini didapatkan dari *website* UINSU. Sedangkan alat penelitian meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*).

3.2.1 Perangkat Keras

Adapun kebutuhan perangkat keras (*hardware*) yang digunakan penulis untuk mendukung penelitian adalah laptop dengan spesifikasi sebagai berikut :

Tabel 3.1 Kebutuhan *Hardware*

Nama Komponen	Spesifikasi
Procesor	Intel(R) Dual Core 3.0
Memory	2 GB
Harddisk	500 GB
Monitor	14 inchi
Keyboard / Mouse	*Standard*

3.2 Perangkat Lunak

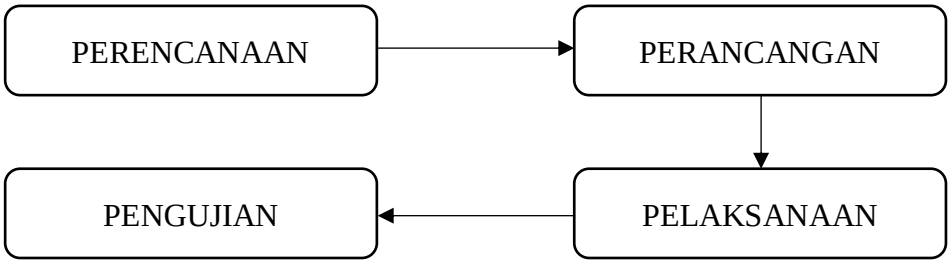
Adapun kebutuhan perangkat lunak (*software*) penulis yang digunakan untuk perancangan Aplikasi Steganografi adalah sebagai berikut :

1. *Microsoft Visual Studio Express* 2012
2. *Microsoft Visio* 2010
3. *Microsoft Office Exel* 2007

3.3. Cara Kerja

Penelitian ini bertujuan untuk mengembangkan produk aplikasi keamanan pesan teks dengan teknik steganografi berupa objek citra digital berbasis dekstop. Penelitian ini termasuk ke dalam penelitian *Research and Development* (R & D). Metode penelitian ini digunakan untuk menghasilkan produk tertentu dan mengkaji keefektifan produk tersebut.

Research and Development (R & D) merupakan suatu proses atau langkah-langkah untuk mengembangkan suatu produk baru atau menyempurnakan produk yang telah ada yang dapat dipertanggung jawabkan. Menurut Sukmadinata (2011), produk tersebut tidak selalu berbentuk benda atau perangkat keras seperti buku, modul alat bantu pembelajaran di kelas atau di laboratorium, tetapi dapat juga berupa perangkat lunak atau program komputer, model pendidikan, pembelajaran, atau pelatihan. Secara garis besar, langkah-langkah dalam penelitian ini meliputi perencanaan (*planning*), Perancangan, pelaksanaan (*acting*), pengujian. Keempat langkah tersebut dapat dilihat dari bagan berikut ini



Gambar 3.1 Bagan Langkah Penelitian

3.3.1 Perencanaan Penelitian

Perencanaan penelitian ini menggunakan aplikasi yang diimplementasikan pada program aplikasi *Microsoft Visual Studio Express* 2012. Pada pengerjaanya, metode penelitian yang digunakan, yaitu dengan tahapan sebagai berikut

1. Tahap Analisis Masalah
2. Tahap Pengumpulan Data
3. Tahap Desain
4. Tahap Pengujian
5. Pembuatan Laporan

Masing-masing tahapan akan dijelaskan sebagai berikut :

1. Tahap Analisis Masalah

   Pada proses analisis masalah pada penelitian ini adalah data teks yang bersifat rahasia harus diamankan mengingat banyak penyerangan atau pemanfaatan data teks yang bersifat pribadi oleh pihak yang tidak bertanggung jawab untuk mendapatkan keuntungan. Oleh sebab itu dibutuhkanya sebuah teknik pengamanan steganografi untuk menyembunyikan data teks kedalam objek berupa citra digital menggunakan algoritma *spread spectrum*. Tahap awal proses penyisipan adalah penyebaran biner dari karakter dengan besaran skalar empat sehingga menghasilkan segmen baru, dilanjutkan dengan pembangkitan *psoudonoise* menggunakan kunci proses penyisipan yang akan menghasilkan *psoudonoise* dalam bentuk biner, kemudian dilakukan proses modulasi terhadap biner karakter yang sudah disebar dengan biner *psoudonoise* untuk menghasilkan biner-biner baru yang akan di sisipkan kedalam objek citra digital.
2. Tahap Desain

   Desain merupakan tahap yang meliputi penentuan unsur-unsur yang perlu dimuatkan dalam *software* yang akan dikembangan sesuai dengan desain pembelajaran. Proses desain pengembangan software pembelajaran meliputi dua aspek desain, yaitu aspek model ID (*Instructional Design* atau desain instruksional) dan aspek isi pengajaran yang akan diberikan.

a. Flowchart

Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* program merupakan keterangan yang lebih rinci tentang bagaimana setiap langkah program sesungguhnya dilaksanakan. Pada *flowchart* digunakan simbol-simbol khusus untuk menggambarkan urutan-urutan prosedur dari suatu program.

b. Rancangan antarmuka (UI)

Antarmuka *(user interface)* merupakan mekanisme komunikasi antara pengguna (user) dengan komputer. Sejalan dengan antarmuka pemakai *(user interface)* pada *mobile* aplikasi, diharapkan dapat menerima informasi dari pengguna (user) dan memberikan informasi kepada pengguna (user) untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan suatu solusi.

2. Tahap Pengumpulan Data Referensi

Tahapan pengumpulan data referensi adalah tahapan memilih data-data referensi yang mencakup aspek penelitian. Pengumpulan data yang dapat diambil dari buku, jurnal, *website*.

3. Tahap Pengujian

Pada saat pengembangan akhir aplikasi akan diujicoba pada perangkat komputer atau laptop. Pada tahap uji pertama dilakukan oleh peneliti tanpa ada peran serta pihak lain. Tahap ini ditujukan untuk memastikan apakah hasil produk aplikasi sudah sesuai dengan tujuan yang diharapkan sebelumnya. Ketika produk lulus pada tahap uji pertama, produk akan memasuki pada tahap uji coba lapangan yang bertujuan mengetahui tanggapan para pengguna secara langsung.

4. Tahap Laporan

Tahap akhir dari penelitian ini adalah pembuatan laporan. Laporan disusun sesuai dengan ketentuan yang tercantum dalam pedoman penulisan karya ilmiah Universitas Islam Sumatera Utara.

3.3.2 Analisa Kebutuhan

Analisa kebutuhan sistem meliputi analisis kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional mendeskripsikan fungsi-fungsi yang harus dilakukan oleh sebuah sistem untuk mencapai tujuan. Sedangkan kebutuhan non-fungsional mendeskripsikan fitur lain seperti karakteristik, batasan sistem, performa, dokumentasi dan yang lainnya agar sistem berjalan sukses (Whitten, 2007) .

1. Anlisa Kebutuhan Fungsional Sistem

   Kebutuhan fungsional yang harus dimiliki oleh sistem keamanan data teks dengan teknik steganografi pada objek citra adalah :

   a. Sistem dapat menyisipkan dan mengamankan data teks kedalam citra digital
2. Analisa Kebutuhan Non-fungsional Sistem

   Kebutuhan non-fungsional yang harus dimiliki oleh sistem keamanan data teks dengan teknik steganografi pada objek citra adalah:

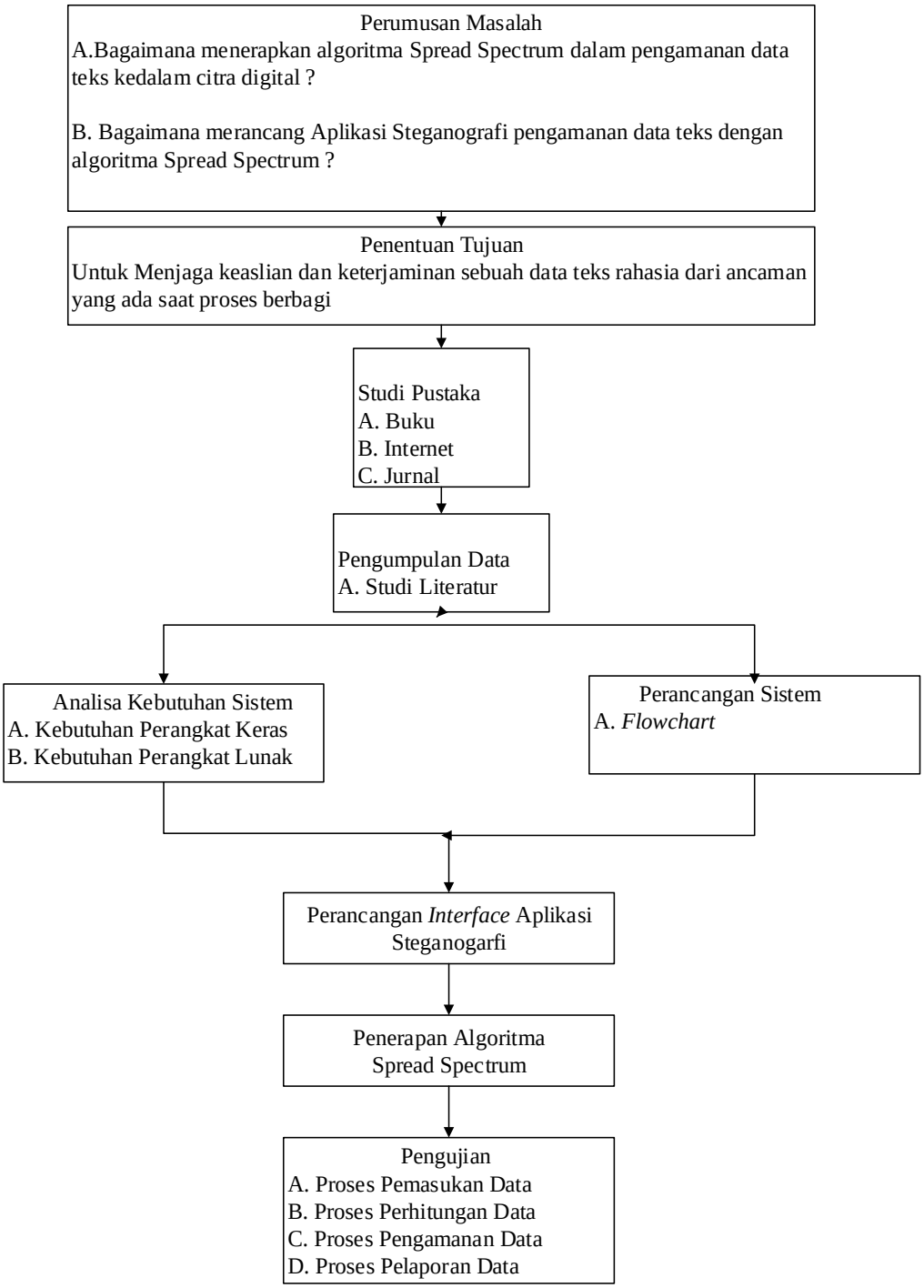
   a. Waktu proses penyisipan data teks kedalam citra cepat, sehingga dapat mengefektifkan waktu pengguna sistem.
   b. Tampilan antarmuka sistem menarik dan dapat dimengerti oleh pengguna sistem

3.3.3 Perancangan Sistem

Perancangan sistem dalam suatu penelitian adalah tahap yang dilakukan peneliti setelah mengumpulkan semua kebutuhan sistem yang akan dirancang. Adapun tahap-tahap yang akan dilakukan meliputi dari perancangan desain penelitian, perancangan *flowchart* metode penelitian dan perancangan *interface* aplikasi.

1. Perancangan Desain Penelitian

   Adapun perancangan desain penelitian yang dilakukan mengikuti alur penelitian yang dapat dilihat pada gambar 3.2 berikut ini :



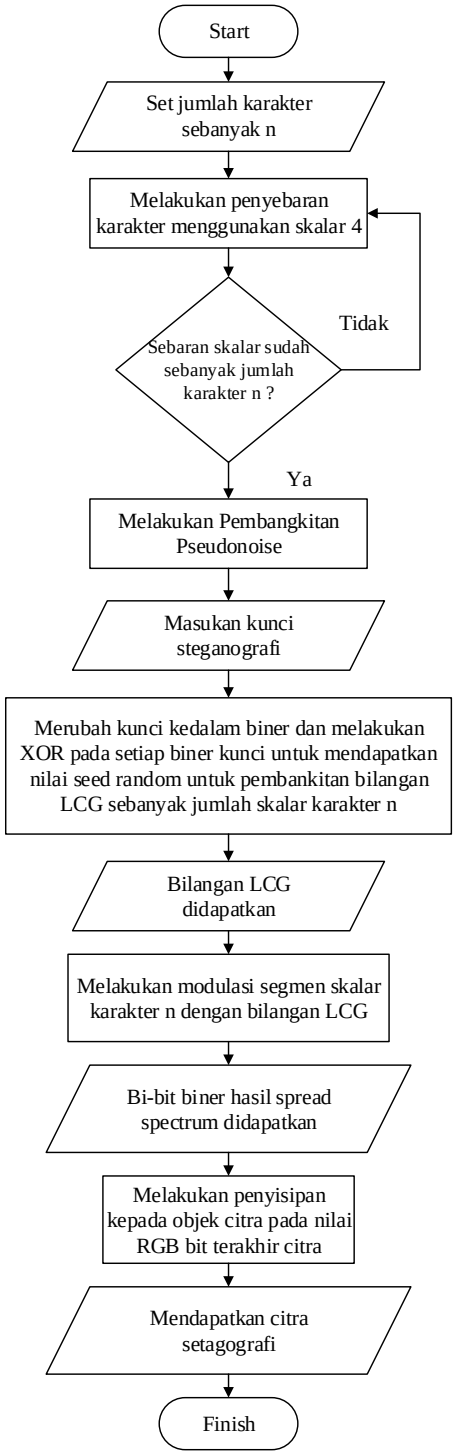
Gambar 3. 2 Langkah-langkah Penelitian

2. Perancangan *Flowchart* Metode Penelitian

   Flwochart metode terdiri dari *flowchart* algoritma *spread spectrum*, *flowchart* proses *embedding* dan *flowchart* proses ekstraksi. Adapun sebagai berikut :

a. *Flowchart Spread Spectrum*

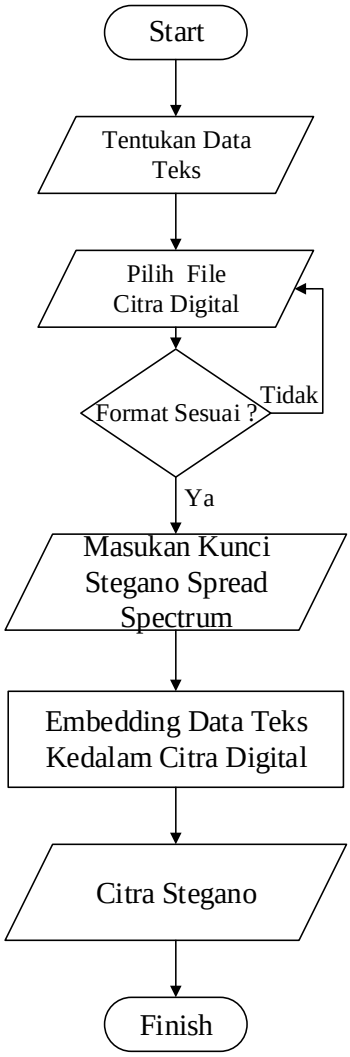
Flowchart spread spectrum adalah gambaran alur proses dari algoritma *spread spectrum* adalam melakukan penyisipan pesan rahasia. Adapun gambar *flowchart spread spectrum* dapat dilihat di bawah ini:



Gambar 3.3 *Flowchart Spread Spectrum*

b. *Flowchart embedding*

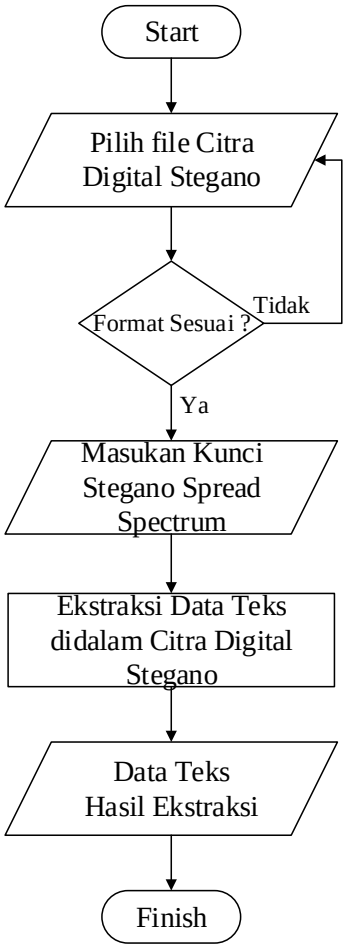
Berikut *flowchart* penyisipan data teks dengan algoritma Spread Spectrum didalam file citra digital yang dapat dilihat pada gambar di bawah ini :



Gambar 3.4 *Flowchart Embedding*

c. *Flowchart* Ekstraksi

Berikut *flowchart* esktraksi data teks didalam file citra digital menggunakan Spread Spectrum yang dapat dilihat pada gambar di bawah ini:



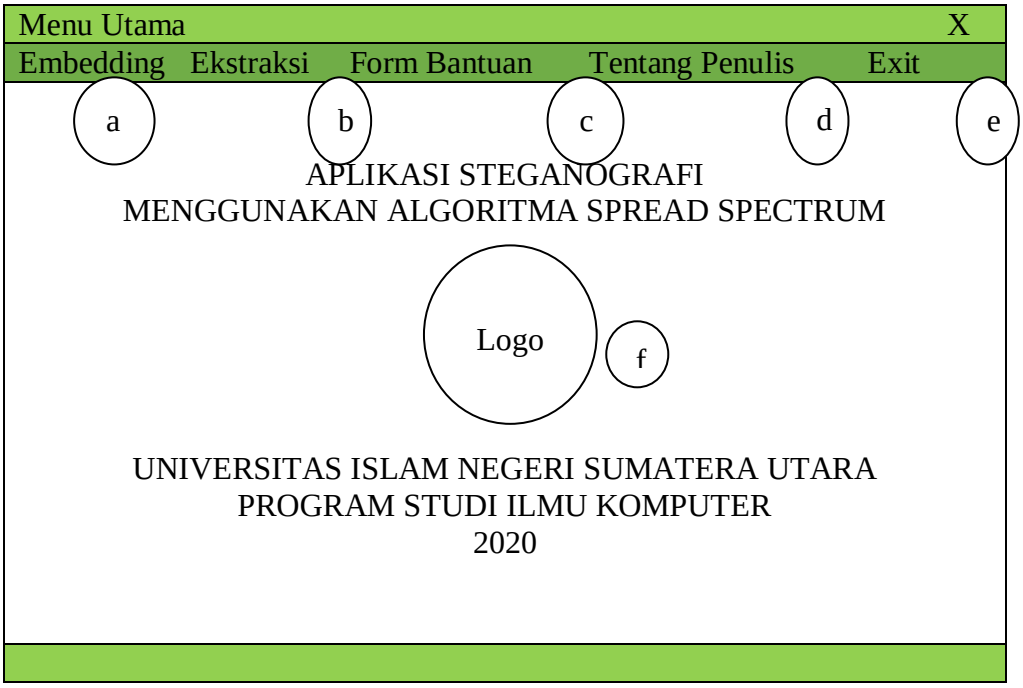
Gambar 3.5 *Flowchart* Ektraksi

2. Perancangan Interface Aplikasi

   Perancangan *interface* bertujuan untuk merancang antar muka aplikasi yang hendak dibangun kedalam sebuah perangat lunak sehingga lebih mudah dalam pembuatan aplikasi dan mudah dimengerti. Berikut adalah bentuk rancangan sistem yang nantinya akan diimplemetasikan kedalam sebuah aplikasi.

   a. Rancangan Menu Utama

      Menu utama adalah *form* yang akan muncul ketika program aplikasi dibuka pertama kali. Adapun tampilan rancangan *form* menu utama adalah sebagai berikut :



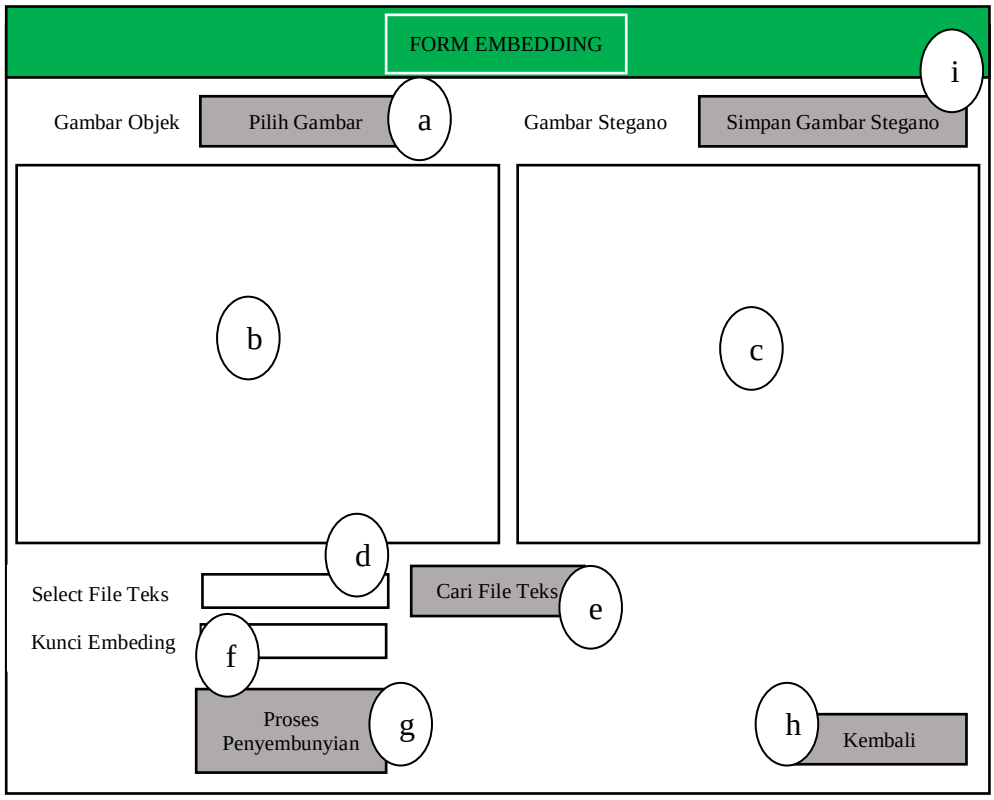
Gambar 3.6 Rancangan Menu Utama

Berdasarkan gambar 3.6 di atas dapat dijelaskan keterangan sebagai berikut:

a) Menu yang menampilkan *form* untuk melakukan proses *embedding*.
b) Menu yang menampilkan *form* untuk melakukan proses estraksi.
c) Menu yang menampilkan *form* bantuan cara pemakaian aplikasi.
d) Menu yang menampilkan *form* tentang penulis dan perancang aplikasi.
e) Menu untuk menutup aplikasi.
f) Logo gambar Universitas Islam Sumatera Utara

b. Rancangan *Form Embedding*

Form embedding berfungsi sebagai *interface* bagi pengguna aplikasi pada saat melakukan kegiatan penyisipan pesan teks kedalam citra digital. Melalui *form embedding* user dapat memasukan data teks dan memilih gambar yang akan dijadikan objek penyisipan, kemudian memasukan kunci steganografi. Adapun rancangan *interface form embedding* dapat dilihat pada gambar berikut :



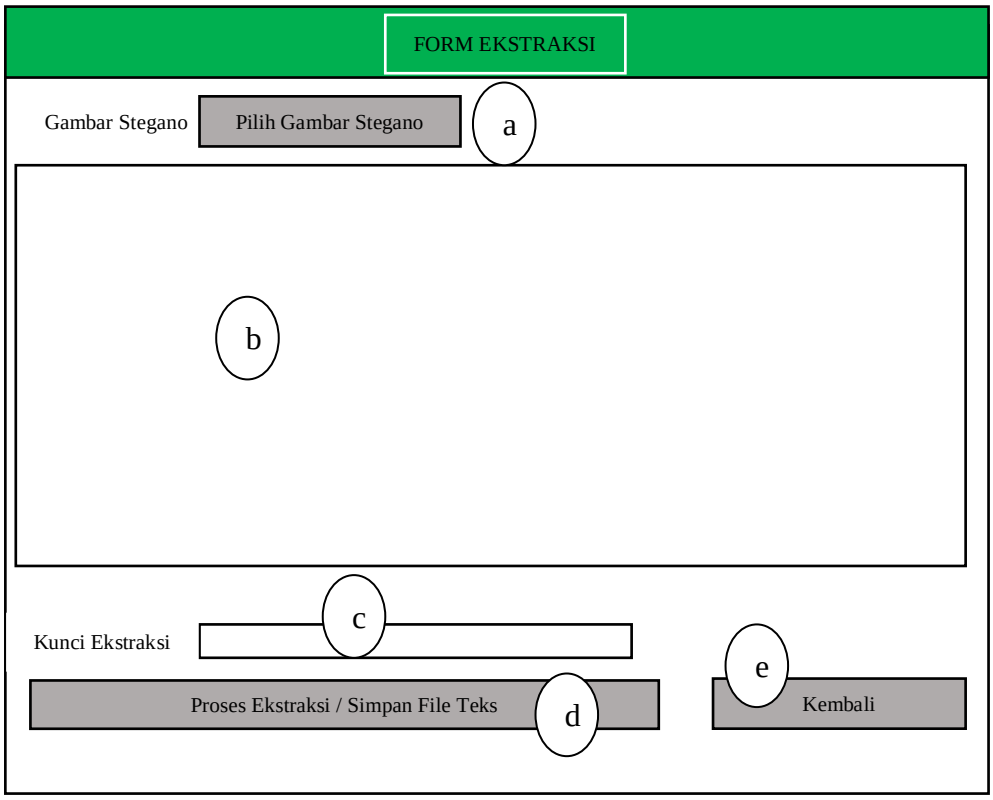
Gambar 3.7 Rancangan *Form Embedding*

Berdasarkan gambar 3.7 di atas dapat dijelaskan keterangan sebagai berikut:

a) *Button* untuk memilih citra digital dari media penyimpanan.
b) *PictureBox* untuk menampilkan citra objek sebelum disisipkan karakter.
c) *PictureBox* untuk menampilkan citra stegano setelah disisipkan objek.
d) *Label* untuk menampilkan nama citra digital objek sebelum disisipkan karakter dan lokasi citra.
e) *Button* untuk mencari file teks didalam direktori.
f) *TextBox* untuk menginput kunci steganografi.
g) *Button* untuk melakukan proses penyisipan pesan teks kedalam citra digital.
h) *Button* untuk kembali ke menu utama aplikasi.
i) *Button* untuk menyimpan citra stegano hasil sisipan.

c. Rancangan *Form* ekstraksi

Form ekstraksi berfungsi sebagai *interface* bagi pengguna aplikasi pada saat melakukan kegiatan ekstraksi data teks didalam citra digital steganografi. Adapun rancangan *interface form* ekstraksi dapat dilihat pada gambar berikut:



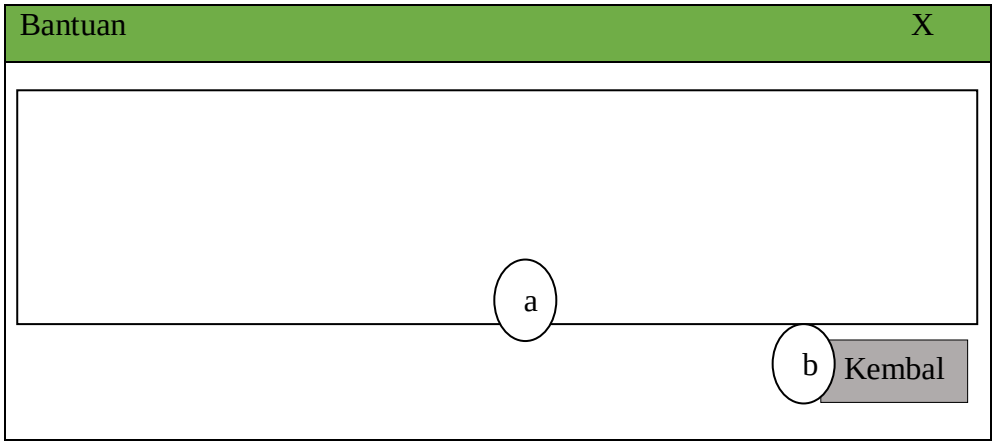
Gambar 3.8 Rancangan *Form* Ekstraksi

Berdasarkan gambar 3.8 di atas dapat dijelaskan keterangan sebagai berikut:

a) *Button* untuk memilih citra stegano dari media penyimpanan.
b) *PictureBox* untuk menampilkan citra stegano.
c) *TextBox* untuk menginput kunci steganografi.
d) *Button* untuk melakukan proses ekstraksi pada citra stegano.
e) *Button* untuk kembali ke menu utama aplikasi.

d. Rancangan *Form* Bantuan

Form bantuan ini berfungsi untuk menampilkan rincian bantuan cara pemakaian dari aplikasi. Adapun rancangan *interface* dari *form* bantuan dapat dilihat pada gambar berikut ini :



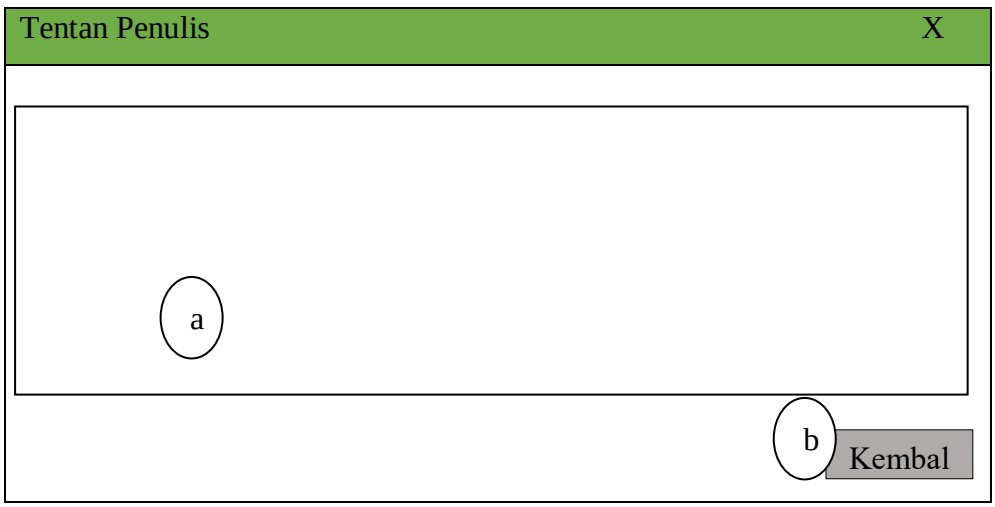
Gambar 3.9 Rancangan *Form* Bantuan

Berdasarkan gambar 3.9 di atas dapat dijelaskan keterangan sebagai berikut:

a) *RichTextBox* untuk menampilkan informasi dari cara pemakaian aplikasi.
b) *Button* untuk kembali pada menu utama aplikasi.

e. Rancangan *Form* Tentang Penulis

Form tentang penulis berfungsi untuk menampilkan informasi tentang penulis dan perancangan aplikasi. Adapun rancangan *interface* pada *form* tentang penulis dapat dilihat pada gambar berikut :



Gambar 3.10 Rancangan *Form* Tentang Penulis

Berdasarkan gambar 3.10 di atas dapat dijelaskan keterangan sebagai berikut:

a) *RichTextBox* untuk menampilkan informasi data diri perancang aplikasi.
b) *Button* untuk kembali pada menu utama aplikasi.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Penerapan Penyisipan Algoritma Spread Spectrum

Berdasarkan pembahasan bab ini, prosesnya adalah melakukan penyembunyian data teks yang berisi karakter kedalam objek citra digital menggunakan algoritma *Spread Spectrum*. Tahap awal proses penyisipan adalah penyebaran biner dari karakter dengan besaran skalar empat sehingga menghasilkan segmen baru, dilanjutkan dengan pembangkitan *psoudonoise* menggunakan kunci proses penyisipan yang akan menghasilkan *psoudonoise* dalam bentuk biner, kemudian dilakukan proses modulasi terhadap biner karakter yang sudah disebar dengan biner *psoudonoise* untuk menghasilkan biner-biner baru yang akan di sisipkan kedalam objek citra digital. Adapun sampel citra digital berformat .jpg dengan resolusi 100 x 200 *pixel* dan dengan *size* yaitu 74,5 byte.

Sampel citra digital tersebut diambil nilai RGB dari setiap *pixel* desimalnya, kemudian dirubah kedalam bentuk nilai-nilai biner yang nantinya akan digunakanan sebagai wadah proses penyisipan. Adapun sampel citra digital yang menjadi objek penyembunyian data teks adalah sebagai berikut:

Gambar 4.1 Sampel Objek Citra Digital

Berdasarkan pada gambar sampel citra digital diambil beberapa sampel *pixelnya* untuk keperluan hitungan manual, adapun sampel *pixel* yang diambil berdasarkan jumlah karakter yang disisipkan nanti. Hal ini dikarenakan jumlah karakter sangat berpengaruh dengan berapa banyak jumlah *pixel* yang akan menjadi objek sisipan. Objek sisipan harus lebih besar dari data teks yang akan disisipkan.

Adapun dalam keperluan hitungan manual penelitian ini, resolusi dari sampel citra digital yang akan meenjadi objek sisipkan adalah 8 x 6 *pixel* atau sebanyak 48 *pixel*. Kemudian setiap *pixel* diekstraksi nilai RGBnya. Berikut nilai RGB dari keseluruhan resolusi citra sampel untuk keperluan hitungan manual:

Tabel 4.1 Nilai Biner Citra Sampel 8 x 6 *Pixel*

Pixel	Warna	Desimal	Biner
l	R	73	01001001
	G	108	01101100
	B	109	01101101
2	R	117	01110101
	G	109	01101101
	B	112	01110000
3	R	117	01110101
	G	116	01110100
	B	101	01100101
4	R	114	01110010
	G	80	01010000
	B	177	10110001
5	R	169	10101001
	G	180	10110100
	B	133	10000101
6	R	178	10110010
	G	109	01101101
	B	169	10101001
7	R	108	01101100
	G	120	01111000
	B	153	10011001
8	R	161	10100001
	G	123	01111011
	B	167	10100111
9	R	152	10011000
	G	177	10110001
	B	157	10011101
10	R	149	10010101
	G	137	10001001
	B	190	10111110

Lanjutan Tabel 4.1 Nilai Biner Citra Sampel 8 x 6 *Pixel*

11	R	128	10000000
	G	169	10101001
	B	179	10110011
12	R	190	10111110
	G	164	10100100
	B	169	10101001
13	R	177	10110001
	G	157	10011101
	B	121	01111001
14	R	180	10110100
	G	192	11000000
	B	168	10101000
15	R	177	10110001
	G	102	01100110
	B	112	01110000
16	R	150	10010110
	G	120	01111000
	B	189	10111101
17	R	177	10110001
	G	109	01101101
	B	150	10010110
18	R	120	01111000
	G	157	10011101
	B	137	10001001
19	R	120	01111000
	G	180	10110100
	B	177	10110001
20	R	167	10100111
	G	177	10110001
	B	180	10110100
.....	.....	.....	……
45	R	99	01100011
	G	129	10000001
	B	184	10111000

Lanjutan Tabel 4.1 Nilai Biner Citra Sampel 8 x 6 *Pixel*

48	R	178	10110010
	G	191	10111111
	B	182	10110110

Bredasarkan pada tabel 4.1, telah diketahui nilai RGB dari setiap sampel citra resolusi 8 x 6 *pixel* atau 48 *pixel*. Selanjutnya adalah menentukan karakter data teks yang akan disisipkan kedalam objek sampel citra digital. Berikut adalah karakter data teks yang akan disisipkan kedalam citra sampel:

Karakter : MERI

Karakter di atas, dikonversikan kedalam bentuk biner seperti pada tabel berikut:

Tabel 4.2 Nilai Desimal dan Biner Sampel Data Teks

No	Karakter	Nilai Desimal	Biner
1	M	77	01001101
2	E	69	01000101
3	R	82	01010010
4	I	73	01001001

Berdasarkan pada tabel di atas, nilai desimal dan biner dari karakter MERI didapatkan dari melihat tabel ASCII. Selanjutnya adalah tahapan penyisipan dengan algoritma *Spread Spectrum*.

1. Penyeberan Biner Data Teks

   Berdasarkan pada tabel di atas, nilai biner karakter data teks sudah didapatkan. Selanjutnya nilai biner karakter data teks disebar menggunakan besaran skalar empat sehingga menghasilkan segmen baru seperti berikut yang berjumlah 128 bit. Adapun prosesnya adalah membuat setiap bilangan biner menjadi 4 digit bilangan biner yang sama. Contoh bilangan biner karakter M= 01001101 jika disebar dengan skalar empat menjadi 00001111 00000000 11111111 00001111. Yang berwarna merah adalah nilai biner sebenarnya. Sehingga hasil dari keseluruhan penyebaran biner skalar empat didapat 128 bit biner baru yaitu:

   00001111 00000000 11111111 00001111
   00001111 00000000 00001111 00001111
   00001111 00001111 00000000 11110000
   00001111 00000000 11110000 00001111

2. Pembangkitan *Pseudonoise*

Pada proses pembangkitan bilangan *pseudonoise* memerlukan kata kunci. Kata kunci yang dipakai dalam proses hitungan manual ini adalah *string* "kami". Proses *Pseudonoise* menggunakan operasi logika XOR untuk mendapatkan bilangan yang digunakan untuk pembangkit sebuah bilangan acak. Adpaun Proses dalam mencari nilai *pseudonoise* sebagai berikut :

Tabel 4.3 Nilai Desimal dan Biner Kunci

Karakter Kunci	Nilai Desimal	Nilai Biner
k	107	01101011
a	97	01100001
m	109	01101101
i	105	01101001

Berdasarkan pada tabel 4.3 di atas, selanjutnya adalah melakukan XOR pada setiap bilangan biner karakter kunci. Adapun tahapanya sebagai berikut ini:

Lakukan XOR untuk setiap nilai biner kunci seperti berikut :

Biner karakter kunci "k" di XOR dengan biner karakter kunci "a", sehingga menghasilkan nilai sebagai berikut :

01101011 XOR 01100001 = 00001010

Selanjutnya hasil XOR dari karakter "k" dan "a" di XOR kembali dengan karakter kunci "m" menghasilkan nilai sebagai berikut:

00001010 XOR 01101101 = 01100111

Selanjutnya hasil XOR dari karakter "m" XOR kembali dengan karakter kunci "i" menghasilkan nilai sebagai berikut:

01100111 XOR 01101001 = **00001110 (7)** dalam desimal

Berdasarkan hasil XOR didapat nilai 7 yang akan digunakan sebagai seed random untuk pembangkitan bilangan acak menggunakan metode *Linear Congruential Generator* (LCG). Adapun untuk membangkitan bilangan LCG ditentukan nilai sebagai berikut :

a = 17

c = 8

m = 84

Adapun proses dari metode *Linear Congruential Generator* (LCG) dalam membentuk bilangan acak sebagai berikut :

Z1 = (17 * 7 + 8) modulus 84 = 43

Z2 = (17 * 43 + 8) modulus 84 = 67

Z3 = (17 * 67 + 8) modulus 84 = 55

Z4 = (17 * 55 + 8) modulus 84 = 19

Z5 = (17 * 19 + 8) modulus 84 = 79

Untuk proses seterusnya dilakukan dengan cara yang salam hingga Z6, Z7, Z8, Z9,….. Zn. Untuk nilai biner yang akan disebar berjumlah 128 bit, sheingga nilai LCG yang diambil harus berjumlah sama, yaitu 128 bit atau 16 angka. Adapun nilai yang telah didapat adalah sebagai berikut:

43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 =01001111 , 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 = 01001111, 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011

3. Modulasi

Selanjutnya adalah melakukan teknik modulasi yang meng-XOR nilai data teks denga nilai segmen *psoudonoise.* Adapun prosesnya adalah sebagai berikut ini:

Segmen data teks 1 = 00001111000000001111111100001111

Segmen *psoudonoise* 1 = 00101011010000110011011100010011

_____XOR

00100100010000111100100000011100

Segmen data teks 2 = 00001111000000000000111100001111

Segmen *psoudonoise* 2 = 01001111000001110010101101000011

_____XOR

01000000000001110010010001001100

Segmen data teks 3 = 00001111000011110000000011110000

Segmen *psoudonoise* 3 = 00110111000100110100111100000111

_____XOR

00111000000111000100111111110111

Segmen data teks 4 = 00001111000000001111000000001111

Segmen *psoudonoise* 4 = 00101011010000110011011100010011

_____XOR

00100100010000111100011100011100

Hasil dari modulasi antara biner karakter data teks dengan biner *pseudonoise* akan disisipkan ke dalam objek sampel citra digital dengan resolusi 8 x 6 *pixel*. Adapun hasil modulasi adalah sebagai berikut :

00100100010000111100100000011100
01000000000001110010010001001100
00111000000111000100111111110111
00100100010000111100011100011100

4. Proses Penyisipan

Selanjutnya adalah melakukan proses penyisipan. Pada proses penyisipan, stegografi membutuhkan sebuah penanda yang berfungsi sebagai akhir untuk pembatas dalam pengambilan bit biner pada citra pada saat proses ekstraksi. Adapun penanda yang digunakan adalah karakter "#" yang dirubah kedalam bentuk biner **00100011**. Proses penyisipkan data teks diaplikasikan pada bit data ke-8 dari nilai biner citra digital, serta pada proses akhir menyisipkan bit penanda akhir berupa nilai biner "#". Jumlah keseluruhan nilai yang bit yang akan disisipkan adalah sebanyak 128 bit sampel karakter data teks hasil proses *spread spectrum* dan 8 bit penanda akhir sehingga total keseluruhan bit yang akan disisipkan pada sampel citra adalah 136 bit. Adapun proses penyisipan atau perpindahan nilai biner sampel karakter data tesk dapat dilihat pada tabel di bawah ini :

Tabel 4.4 Proses Penyisipan Bit Data Teks *Spread Spectrum*

Sampel Citra Objek				Bit Data Teks Spread Spectrum	Sampel Citra Stegano			
Pixel	Warna	Desimal	Biner		Pixel	Warna	Desimal	Biner
1	R	73	01001001	0	1	R	72	01001000
	G	108	01101100	0		G	108	01101100
	B	109	01101101	1		B	109	01101101
2	R	117	01110101	0	2	R	116	01110100
	G	109	01101101	0		G	108	01101100
	B	112	01110000	1		B	113	01110001
3	R	117	01110101	0	3	R	116	01110100
	G	116	01110100	0		G	116	01110100
	B	101	01100101	0		B	100	01100100
4	R	114	01110010	1	4	R	115	01110011
	G	80	01010000	0		G	80	01010000
	B	177	10110001	0		B	176	10110000
5	R	169	10101001	0	5	R	168	10101000
	G	180	10110100	0		G	180	10110100
	B	133	10000101	1		B	133	10000101
6	R	178	10110010	1	6	R	179	10110011
	G	109	01101101	1		G	109	01101101
	B	169	10101001	1		B	169	10101001
7	R	108	01101100	0	7	R	108	01101100
	G	120	01111000	0		G	120	01111000
	B	153	10011001	1		B	153	10011001
8	R	161	10100001	0	8	R	160	10100000
	G	123	01111011	0		G	122	01111010
	B	167	10100111	0		B	166	10100110
9	R	152	10011000	0	9	R	152	10011000
	G	177	10110001	0		G	176	10110000
	B	157	10011101	0		B	156	10011100
10	R	149	10010101	1	10	R	149	10010101
	G	137	10001001	1		G	137	10001001
	B	190	10111110	1		B	191	10111111
11	R	128	10000000	0	11	R	128	10000000
	G	169	10101001	0		G	168	10101000
	B	179	10110011	0		B	178	10110010

Lanjutan Tabel 4.4 Proses Penyisipan Bit Data Teks *Spread Spectrum*

12	R	190	10111110	1	12	R	191	10111111
	G	164	10100100	0		G	164	10100100
	B	169	10101001	0		B	168	10101000
13	R	177	10110001	0	13	R	176	10110000
	G	157	10011101	0		G	156	10011100
	B	121	01111001	0		B	120	01111000
14	R	180	10110100	0	14	R	180	10110100
	G	192	11000000	0		G	192	11000000
	B	168	10101000	0		B	168	10101000
15	R	177	10110001	0	15	R	176	10110000
	G	102	01100110	0		G	102	01100110
	B	112	01110000	0		B	112	01110000
16	R	150	10010110	1	16	R	151	10010111
	G	120	01111000	1		G	121	01111001
	B	189	10111101	1		B	189	10111101
17	R	177	10110001	0	17	R	176	10110000
	G	109	01101101	0		G	108	01101100
	B	150	10010110	1		B	151	10010111
18	R	120	01111000	0	18	R	120	01111000
	G	157	10011101	0		G	156	10011100
	B	137	10001001	1		B	137	10001001
19	R	120	01111000	0	19	R	120	01111000
	G	180	10110100	0		G	180	10110100
	B	177	10110001	0		B	176	10110000
20	R	167	10100111	1	20	R	167	10100111
	G	177	10110001	0		G	176	10110000
	B	180	10110100	0		B	180	10110100
…	…	…	…	…	…	…	…	…
44	R	99	01100011	0	44	R	98	01100010
	G	129	10000001	1		G	129	10000001
	B	184	10111000	0		B	184	10111000
45	R	178	10110010	0	45	R	178	10110010
	G	191	10111111	0		G	190	10111110
	B	182	10110110	1		B	183	10110111
46	R	190	10111110	1	46	R	191	10111111
	G	89	01011001			G	89	01011001

Berdasarkan pada proses penyisipan biner sampel karakter data teks, nilai desimal RGB dari setiap *pixel* sampel mengalami perubahan dari pengurangan dan penambahan nilai sebanyak 1 nilai, adapun sebagian memiliki nilai yang tetap. Hal ini terjadi dikarenakan perpindahan dilakukan pada bit akhir (ke 8) nilai *pixel* sampel sehingga citra objek tidak mengalami perbuahan bentuk dan warna yang signifikan. Adapun nilai RGB keseluruhan *pixel* citra sampel yang telah disisipkan dengan biner sampel karakter data teks menggunakan algoritma *Spread Spectrum* dapat dilihat pada tabel di bawah ini :

Tabel 4.5 Nilai RGB *Pixel* Sampel Objek Citra Stegano

Sampel Citra Stegano			
Pixel	Warna	Desimal	Biner
l	R	72	1001000
	G	108	1101100
	B	109	1101101
2	R	116	1110100
	G	108	1101100
	B	113	1110001
3	R	116	1110100
	G	116	1110100
	B	100	1100100
4	R	115	1110011
	G	80	1010000
	B	176	10110000
5	R	168	10101000
	G	180	10110100
	B	133	10000101
6	R	179	10110011
	G	109	1101101
	B	169	10101001
7	R	108	1101100
	G	120	1111000
	B	153	10011001

Lanjutan Tabel 4.5 Nilai RGB Pixel Sampel Objek Citra Stegano

8	R	160	10100000
	G	122	1111010
	B	166	10100110
9	R	152	10011000
	G	176	10110000
	B	156	10011100
10	R	149	10010101
	G	137	10001001
	B	191	10111111
11	R	128	10000000
	G	168	10101000
	B	178	10110010
12	R	191	10111111
	G	164	10100100
	B	168	10101000
13	R	176	10110000
	G	156	10011100
	B	120	1111000
14	R	180	10110100
	G	192	11000000
	B	168	10101000
15	R	176	10110000
	G	102	1100110
	B	112	1110000
16	R	151	10010111
	G	121	1111001
	B	189	10111101
17	R	176	10110000
	G	108	1101100
	B	151	10010111
18	R	120	1111000
	G	156	10011100
	B	137	10001001

Lanjutan Tabel 4.5 Nilai RGB *Pixel* Sampel Objek Citra Stegano

19	R	120	1111000
	G	180	10110100
	B	176	10110000
20	R	167	10100111
	G	176	10110000
	B	180	10110100
...	...	…	…
44	R	98	1100010
	G	129	10000001
	B	184	10111000
45	R	178	10110010
	G	190	10111110
	B	183	10110111
46	R	191	10111111
	G	89	1011001
	B	145	10010001

4.2 Penerapan Ekstraksi Algoritma Spread Spectrum

Proses ekstraksi dilakukan untuk mengembalikan sampel karakter data teks yang sudah di sembunyikan ke dalam sampel citra digital. Adapun tahaoan dari proses ekstraksi sama dengan proses embedding, dimana kunci yang akan digunakan adalah kunci yang sama saat proses penyisipan. Adapun sampel citra *stgano* yang akan di ekstraksi dapat dilihat pada tabel 4.5 di atas.

Adapun kunci yang digunakan pada proses ekstraksi adalah *string* “kami”. Berdasarkan tabel 4.5, tahap-tahap proses ekstraksi adalah sebagai berikut :

1. Mengembalikan bit-bit yang telah disisipkan

   Pada tahap pengembalian bit-bit yang telah disisipkan, bit-bit yang akan digunakan adalah bit-bit akhir dari sampel citra stegano. Proses pengambilan bit akan berhenti ketika mendaptkan bit dari pendanda akhir yaitu # dalam biner **00100011**. Adapun proses pengembalian bit-bit akhir yang telah disisipkan dapat dilihat pada tabel dibawah ini :

Tabel 4.6 Proses Pengembalian Bit Data Teks

Sampel Citra Stegano				Bit Data Teks Spread Spectrum	Keterangan
Pixel	Warna	Desimal	Biner		
1	R	72	01001000	0	**00100100** Biner tidak sesuai dengan biner # penanda akhir, maka proses pengambilan bit dilanjutkan
	G	108	01101100	0	
	B	109	01101101	1	
2	R	116	01110100	0	
	G	108	01101100	0	
	B	113	01110001	1	
3	R	116	01110100	0	
	G	116	01110100	0	
	B	100	01100100	0	**01000011** Biner tidak sesuai dengan biner # penanda akhir, maka proses pengambilan bit dilanjutkan
4	R	115	01110011	1	
	G	80	01010000	0	
	B	176	10110000	0	
5	R	168	10101000	0	
	G	180	10110100	0	
	B	133	10000101	1	
6	R	179	10110011	1	
	G	109	01101101	1	**11001000** Biner tidak sesuai dengan biner # penanda akhir, maka proses pengambilan bit dilanjutkan
	B	169	10101001	1	
7	R	108	01101100	0	
	G	120	01111000	0	
	B	153	10011001	1	
8	R	160	10100000	0	
	G	122	01111010	0	
	B	166	10100110	0	
9	R	152	10011000	0	**00011100** Biner tidak sesuai dengan biner # penanda akhir, maka proses pengambilan bit dilanjutkan
	G	176	10110000	0	
	B	156	10011100	0	
10	R	149	10010101	1	
	G	137	10001001	1	
	B	191	10111111	1	
11	R	128	10000000	0	
	G	168	10101000	0	
	B	178	10110010	0	

12	R	191	10111111	1	**01000000** Biner tidak sesuai dengan biner # penanda akhir, maka proses pengambilan bit dilanjutkan
	G	164	10100100	0	
	B	168	10101000	0	
13	R	176	10110000	0	
	G	156	10011100	0	
	B	120	01111000	0	
14	R	180	10110100	0	
	G	192	11000000	0	**00000111** Biner tidak sesuai dengan biner # penanda akhir, maka proses pengambilan bit dilanjutkan
	B	168	10101000	0	
15	R	176	10110000	0	
	G	102	01100110	0	
	B	112	01110000	0	
16	R	151	10010111	1	
	G	121	01111001	1	
	B	189	10111101	1	

Proses yang sama dilanjutkan hingga ditemukan nilai biner # yaitu **00100011,** adapun lanjutan hasilnya sebagai berikut:

Lanjutan Tabel **4.6** Proses Pengembalian Bit Data Teks

44	B	134	10000110	0	**00100011** Biner ini sesuai dengan biner # penanda akhir, maka proses pengambilan bit dihentikan
45	R	98	01100010	0	
	G	129	10000001	1	
	B	184	10111000	0	
47	R	178	10110010	0	
	G	190	10111110	0	
	B	183	10110111	1	
48	R	191	10111111	1	
	G	89	01011001		
	B	145	10010001		

Berdasarkan tabel di atas, biner dari penanda akhir # tidak digunakan pada tahap selanjutnya. Maka hasil dari keseluruhan proses pengembalian bit-bit yang telah disisipkan adalah sebagai berikut :

00100100010000111100100000011100

01000000000001110010010001001100

00111000000111000100111111110111

00100100010000111100011100011100

2. Pembangkitan *Pseudonoise*

Pada proses pembangkitan bilangan *pseudonoise* memerlukan kata kunci. Kata kunci yang dipakai dalam proses hitungan manual ini adalah *string* "kami". Proses *Pseudonoise* menggunakan operasi logika XOR untuk mendapatkan bilangan yang digunakan untuk pembangkit sebuah bilangan acak. Adpaun Proses dalam mencari nilai *pseudonoise* sebagai berikut :

Tabel 4.7 Nilai Desimal dan Biner Kunci

Karakter Kunci	Nilai Desimal	Nilai Biner
k	107	01101011
a	97	01100001
m	109	01101101
i	105	01101001

Berdasarkan pada tabel 4.3 di atas, selanjutnya adalah melakukan XOR pada setiap bilangan biner karakter kunci. Adapun tahapanya sebagai berikut ini:

Lakukan XOR untuk setiap nilai biner kunci seperti berikut :

Biner karakter kunci "k" di XOR dengan biner karakter kunci "a", sehingga menghasilkan nilai sebagai berikut :

01101011 XOR 01100001 = 00001010

Selanjutnya hasil XOR dari karakter "k" dan "a" di XOR kembali dengan karakter kunci "m" menghasilkan nilai sebagai berikut:

00001010 XOR 01101101 = 01100111

Selanjutnya hasil XOR dari karakter "m" XOR kembali dengan karakter kunci "i" menghasilkan nilai sebagai berikut:

01100111 XOR 01101001 = **00001110 (7)** dalam desimal

Berdasarkan hasil XOR didapat nilai 7 yang akan digunakan sebagai seed random untuk pembangkitan bilangan acak menggunakan metode *Linear*

Congruential Generator (LCG). Adapun untuk membangkitan bilangan LCG ditentukan nilai sebagai berikut :

a = 17

c = 8

m = 84

Adapun proses dari metode *Linear Congruential Generator* (LCG) dalam membentuk bilangan acak sebagai berikut :

Z1 = (17 * 7 + 8) modulus 84 = 43

Z2 = (17 * 43 + 8) modulus 84 = 67

Z3 = (17 * 67 + 8) modulus 84 = 55

Z4 = (17 * 55 + 8) modulus 84 = 19

Z5 = (17 * 19 + 8) modulus 84 = 79

Untuk proses seterusnya dilakukan dengan cara yang salam hingga Z6, Z7, Z8, Z9,….. Zn. Untuk nilai biner yang akan disebar berjumlah 128 bit, sheingga nilai LCG yang diambil harus berjumlah sama, yaitu 128 bit atau 16 angka. Adapun nilai yang telah didapat adalah sebagai berikut:

43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 =01001111 , 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 = 01001111, 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011

3. Demodulasi

Selanjutnya adalah melakukan teknik demodulasi yang meng-XOR nilai data teks dengan nilai segmen *psoudonoise.* Adapun prosesnya adalah sebagai berikut ini:

Segmen data teks 1 = 00001111000000001111111100001111

Segmen *psoudonoise* 1 = 00101011010000110011011100010011

_____XOR

00100100010000111100100000011100

Segmen data teks 2 = 01000000000001110010010001001100

Segmen *psoudonoise* 2 = 01001111000001110010101101000011

_____XOR

00001111000000000000111100001111

Segmen data teks 3 = 00111000000111000100111111110111

Segmen *psoudonoise* 3 = 00110111000100110100111100000111

_____XOR

00001111000011110000000011110000

Segmen data teks 4 = 00100100010000111100011100011100

Segmen *psoudonoise* 4 = 00101011010000110011011100010011

_____XOR

00001111000000001111000000001111

Hasil dari demodulasi antara biner karakter data teks *spread spectrum* dengan biner *pseudonoise* akan disisipkan ke dalam objek sampel citra digital dengan resolusi 8 x 6 *pixel*. Adapun hasil demodulasi adalah sebagai berikut :

00001111 00000000 11111111 00001111
00001111 00000000 00001111 00001111
00001111 00001111 00000000 11110000
00001111 00000000 11110000 00001111

4. Tahap Mempekecil Biner Hasil Demodulasi

Pada tahap ini akan membagi lima hasil demodulasi yang berguna untuk menyusutkan hasil demodulasi menjadi biner sampel karakter data teks yang sebenarnya. Proses penyusutan (*de-spreading*) segmen tersebut kebalikan dari proses pembentukan besaran skalar 4, yaitu dengan menghapus 3 bit nilai biner yang sama dari setiap bit biner, sehingga hasilnya menjadi:

01001101 01000101 01010010 01001001

Berdasarkan biner-biner hasil penyusutan, maka biner sampel karakter data teks dirubah kembali kedalam bentuk desimal dan menjadi bentuk karakter. Adapun hasilnya dapat dilihat pada tabel dibawah ini :

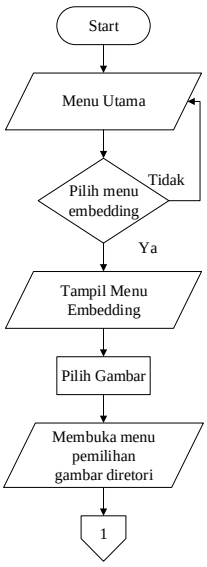
Tabel 4.8 Nilai Desimal dan Biner Sampel Data Teks Kembalii

No	Karakter	Nilai Desimal	Biner
1	M	77	01001101
2	E	69	01000101
3	R	82	01010010
4	I	73	01001001

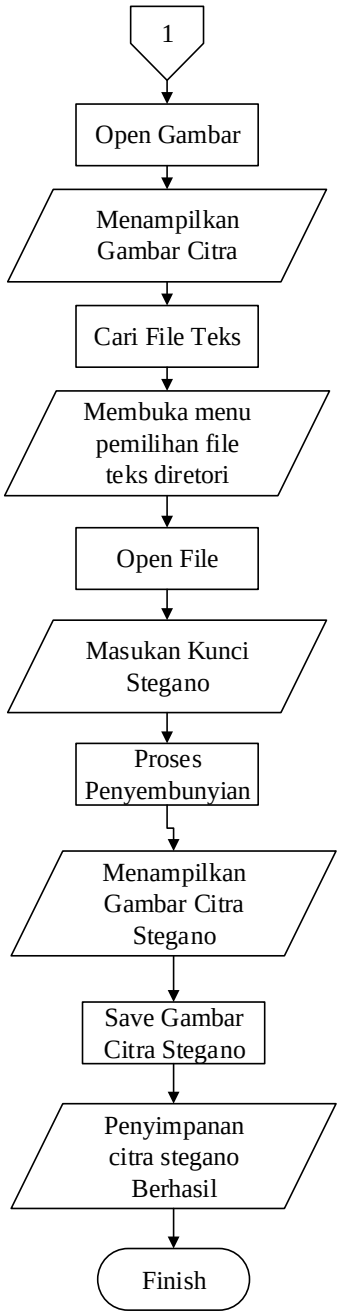
Berdasarkan pada tabel 4.8 karakter awal sebelum disisipkan kembali yaitu karakter “MERI”.

4.3 Pengujian Steganografi Pada Aplikasi

Berdasarkan hasil perancangan aplikasi dan proses hitungan manual penyisipan data teks pada citra digital menggunakan algoritma *spread spectrum*, maka diimplementasikanlah kedalam sebuah aplikasi. Adapun proses dari penerapan steganografi pada aplikasi ini terdiri dari proses *embedding* yaitu menyisipkan pesan kedalam objek citra dan proses ekstraksi yaitu mengeluarkan pesan dari objek citra digital. Sebelum melakukan proses *embedding*, adapun *flowchart* dari proses penyisipan pesan pada menu *embedding* aplikasi sebagai berikut :



Gambar 4.2 Flowchart Menu *Embedding*



Gambar 4.2 Flowchart Menu *Embedding* **(Lanjutan)**

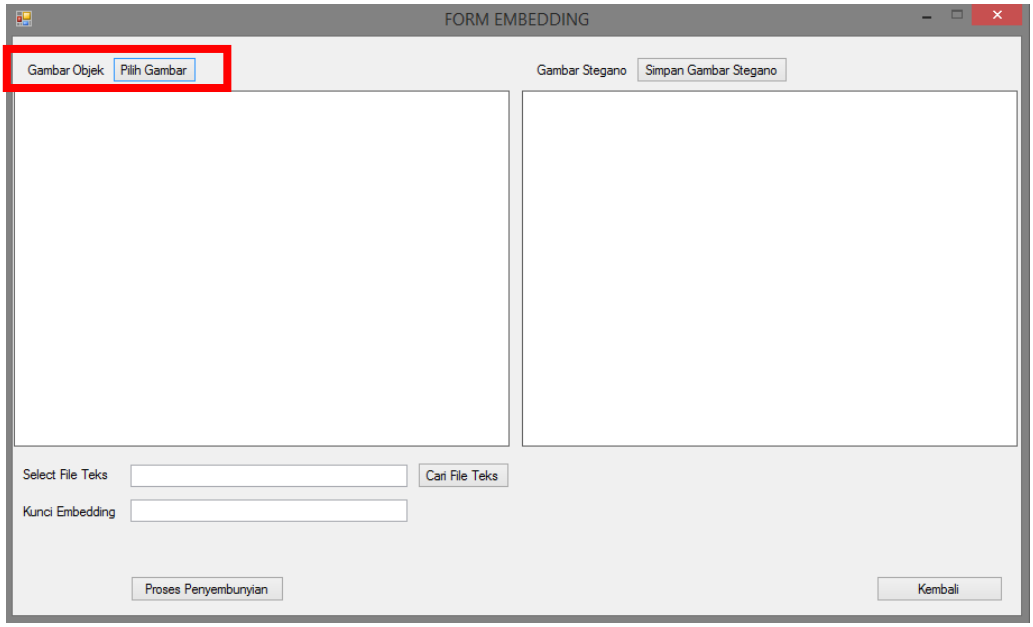
Adapun prosesnya didalam aplikasi adalah sebagai berikut ini:

1. Proses *Embedding* Data Teks

   Sebelum melakukan proses *embedding*, ketika program aplikasi pertama kali dibuka maka akan menampilkan menu utama seperti pada gambar di bawah ini:

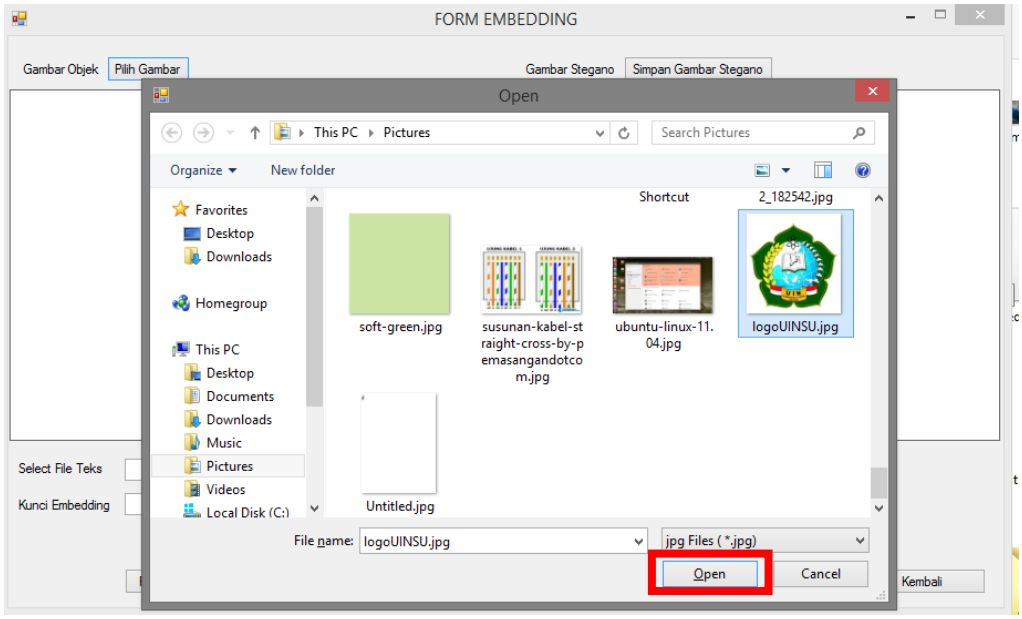
Gambar 4.3 Tampilan Menu Utama

Berdasarkan pada gambar 4.3 menu utama, terdapat beberapa menu yang memiliki fungsi masing-masing. Menu *Embedding* digunakan untuk melakukan proses pemyembunyian dengan teknik steganografi pada data teks kedalam citra digital, menu Ekstraksi digunakan untuk melakukan pengembalian data teks didalam citra digital steganografi, menu Bantuan berupa informasi aplikasi, menu Tentang Penulis berupa informasi tentang penulis dan menu *Exit* digunakan untuk keluar dari aplikasi utama. Untuk melakukan proses *embedding* makan *user* memilih menu *embedding* sehingga menampilkan gambar seperti di bawah ini:



Gambar 4.4 Menu *Embedding*

Berdasarkan pada gambar di atas, proses pertama adalah memilih gambar citra digital yang menjadi objek penampung data teks dengan menekan *button* "Pilih Gambar", sehingga muncul *pop up* pemilihan gambar seperti di bawah ini:

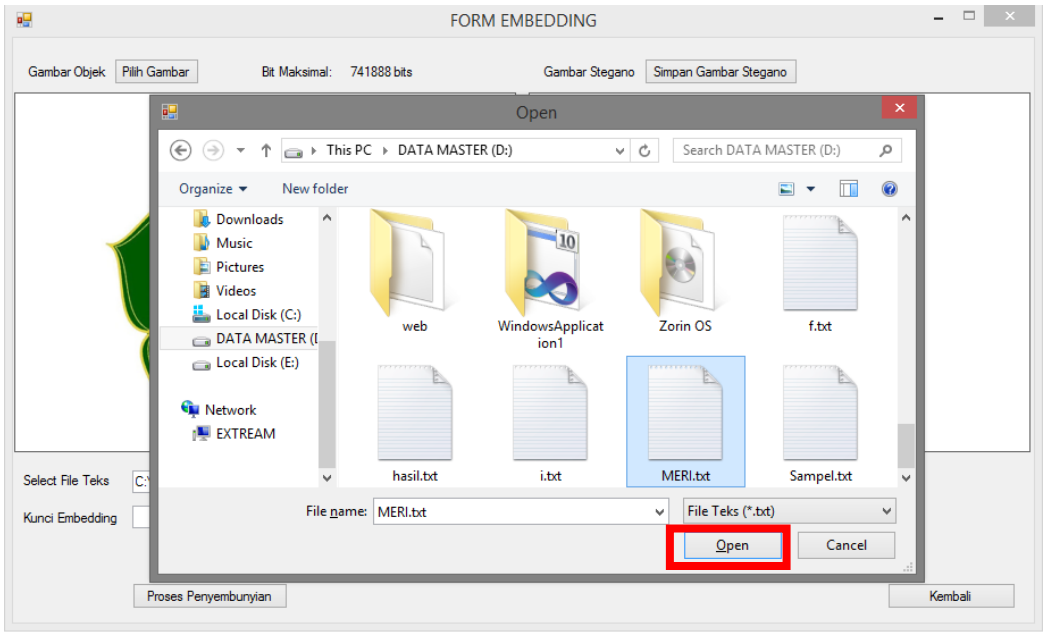


Gambar 4.5 *Pop Up* Pilih Gambar Pada Menu *Embedding*

Berdasarkan pada gambar di atas, kemudian memilih gambar dan menekan *button* "*Open*". Dalam kasus ini gambar yang dipilih dengan nama logoUINSU.jpg. Setelah dipilih, kemudian tampil seperti gambar di bawah ini:

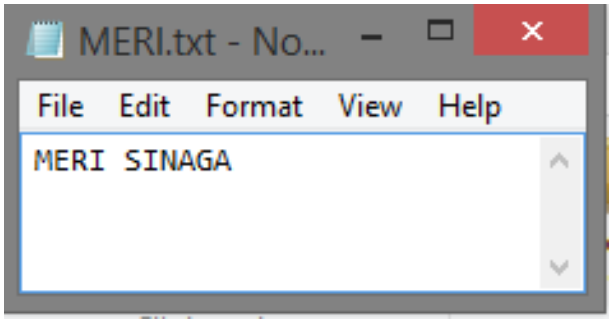
Gambar 4.6 Objek Tampil Pada Menu *Embedding*

Berdasarkan pada gambar di atas, pemilihan objek gambar berhasil, kemudian dilanjutkan dengan pemilihan file data teks yang akan disembunyikan kedalam file gambar citra digital dengan menekan *button* "Cari File Teks" sehingga muncul *pop menu* cari file teks seperti gambar di bawah ini:



Gambar 4.7 Objek Tampil Pada Menu *Embedding*

Berdasarkan pada gambar di atas, file data teks yang dipilih bernama MERI.txt dengan isian karakter "MERI SINAGA" seperti gambar bawah ini:

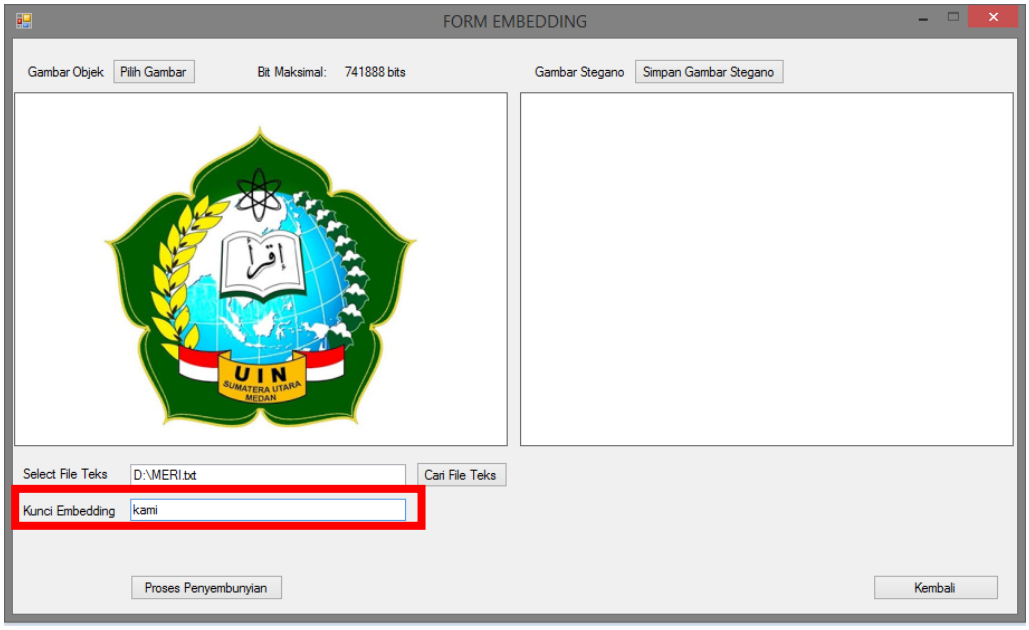


Gambar 4.8 Isi Karakter File Teks Meri

Kemudian *user* menekan *button* "*open*" sehingga tampil seperti gambar di bawah ini:

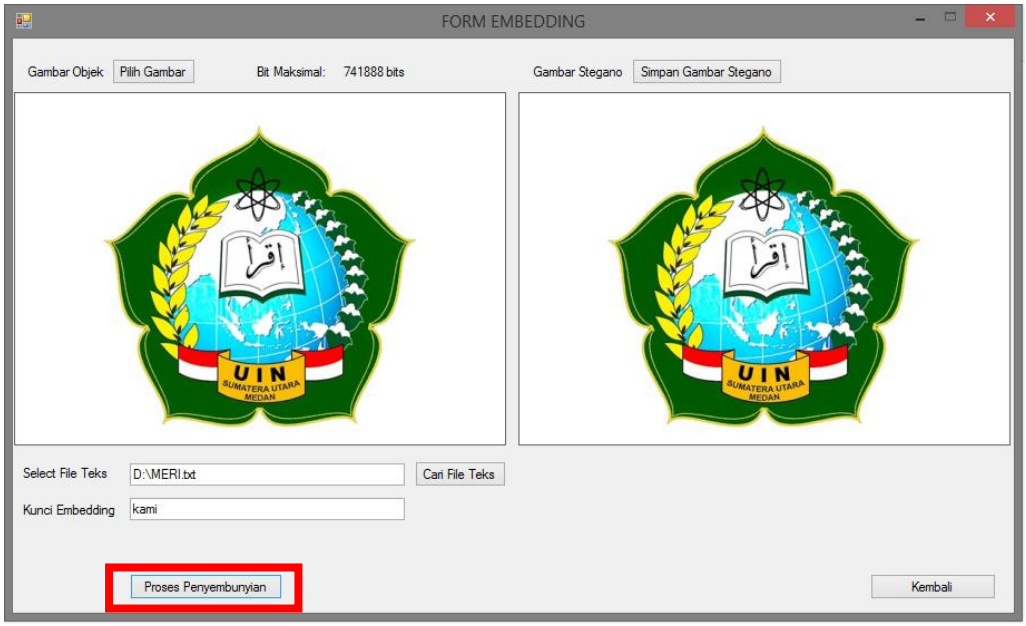
Gambar 4.9 Hasil Pemilihan File Teks Pada Menu *Embedding*

Berdasarkan pada gambar 4.9, kemudian untuk memulai proses penyisipan pesan terlebih dahulu memasukan kunci. Kunci yang digunakan pada contoh penerapan ini adalah kata "kami" seperti gambar di bawah ini:



Gambar 4.10 Kunci Pada Menu *Embedding*

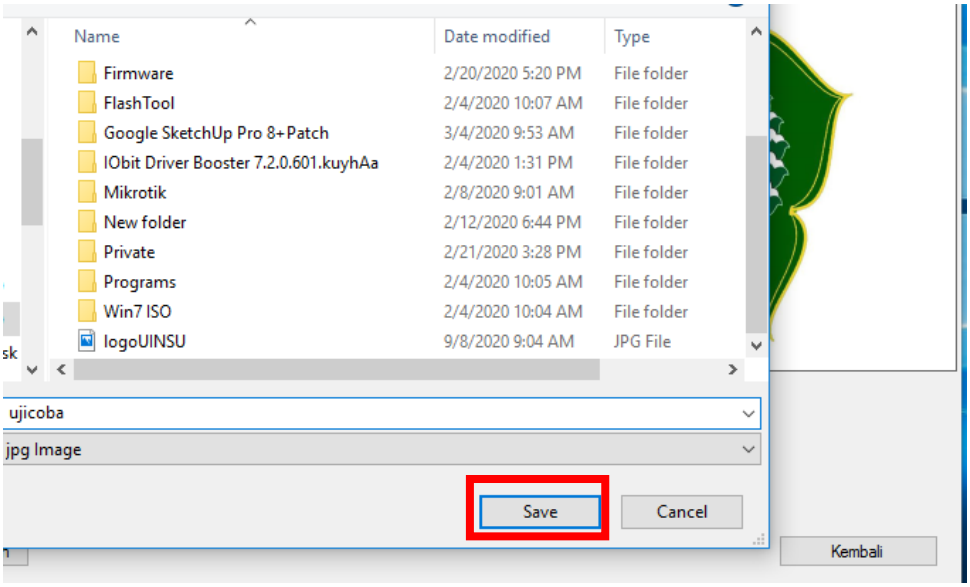
Berdasarkan pada gambar di atas, untuk melakukan proses *embedding* (penyisipan) pesan *user* menekan *button* "Proses Penyembunyian" sehingga menampilkan hasil seperti gambar di bawah ini:



Gambar 4.11 Citra Stegano Pada Menu *Embedding*

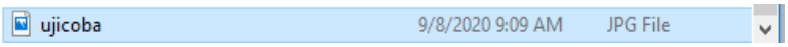
Berdasarkan pada gambar 4.11, didapati hasil penyisipan data teks kedalam citra digital berupa citra digital baru yang dinamakan citra stegano. Dapat diperhatikan tidak terdapat perbeedaan bentuk, warna dan resolusi, hal ini

dikerenakan penyisipan dilakukan pada bid terakhir citra. Kemudian hasil disimpan dengan menekan *button* "Simpan Gambar Stegano" sehingga muncul *pop menu* simpan seperti gambar di bawah ini:



Gambar 4.12 Simpan Citra Stegano Pada Menu *Embedding*

Berdasarkan pada gambar di atas, hasil dari citra stegano disimpan kedalam format .jpg dengan nama file citra "ujicoba.jpg", kemudian tekan *button* "*Save*" sehingga file citra stegano tersimpan kedalam direktori penyimpanan dengan hasil sepeti gambar di bawah ini:



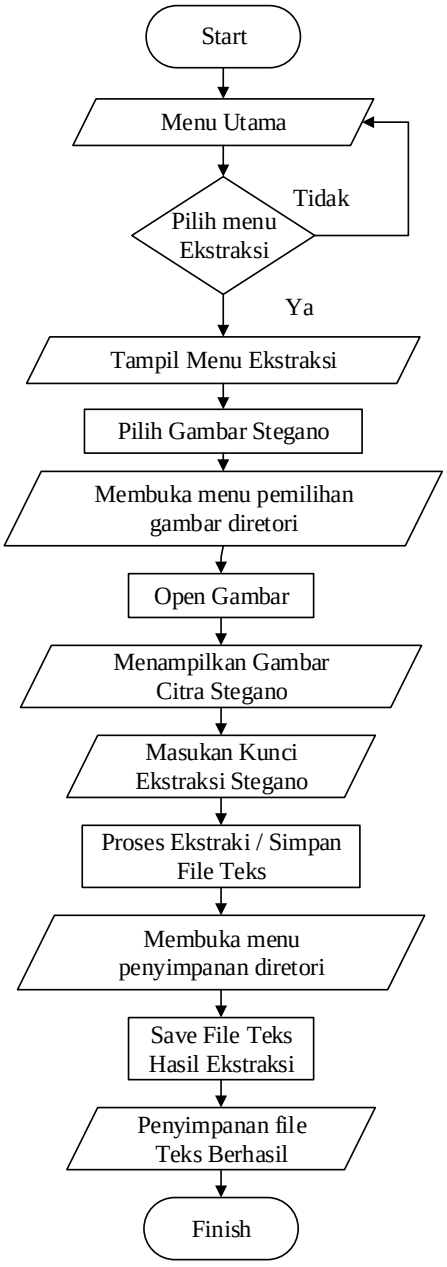
Gambar 4.13 Hasil Penyimpan Citra Stegano Pada Menu *Embedding*

a b

Gambar 4.14 Citra Asli a, Citra Stegano b

Berdasarkan pada gambar di atas, sekilas tidak dapat perbedaan citra asli dan citra steganografi.

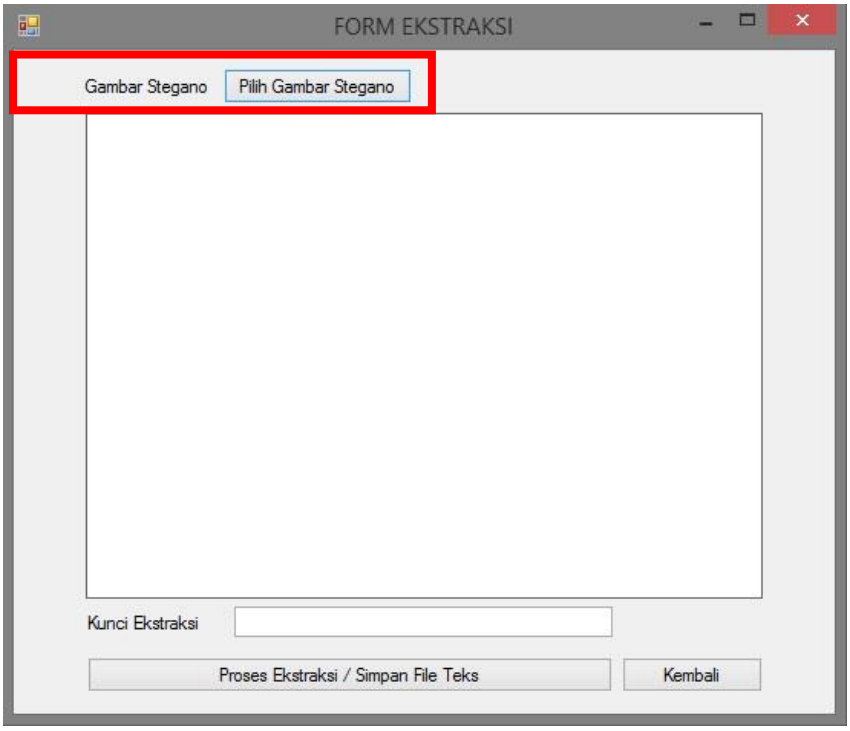
Sebelum melakukan proses ekstraksi didalam aplikasi, adapun *flowchart* sistemnya dapat dilihat pada gambar berikut ini:



Gambar 4.15 Flowchart Menu Ekstraksi

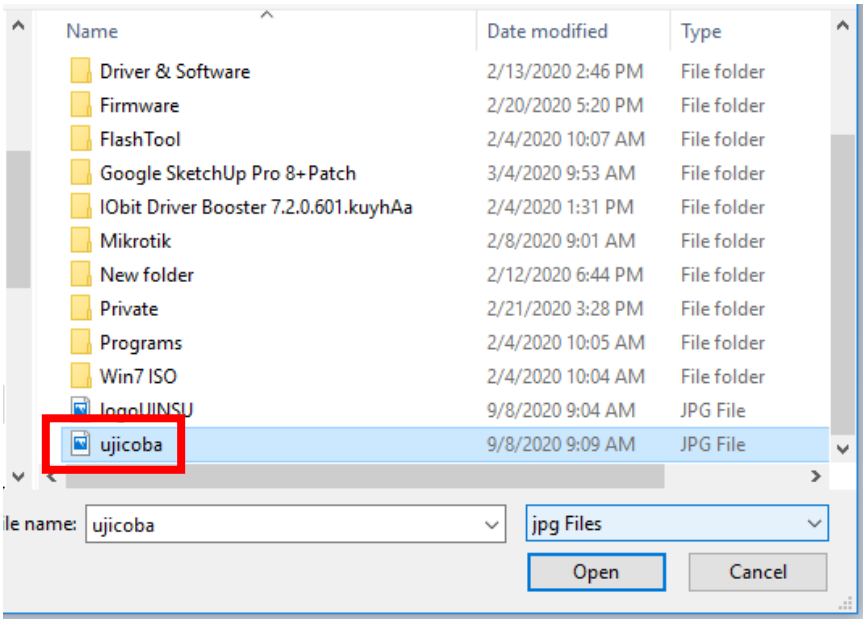
2. Proses Ekstraksi Data Teks

   Setelah dilakukan proses *embedding* data teks kedalam citra digital, kemudian untuk mengembalikan file teks dari dalam citra digital maka dilakukan proses ekstraksi dengan menekan menu “Ekstraki” pada menu utama sehingga muncul menu ekstraksi seperti gambar di bawah ini:



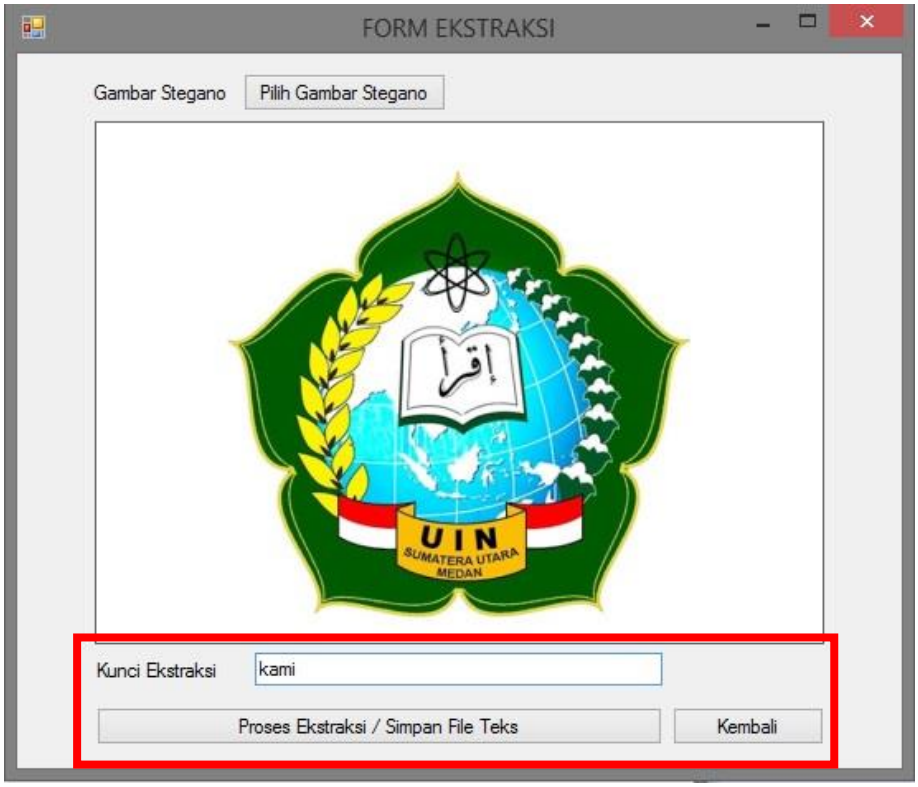
Gambar 4.16 Tampilan Menu Ekstraksi

Berdasarkan menu ekstraksi data teks didalam citra digital, untuk memulai terlebih dahulu memilih file citra / gambar stegano hasil *embedding* dengan format .jpg dengan cara menekan *button* "Pilih Gambar Stegano" dan menampilkan *pop up* menu pilih citra stegano seperti gambar di bawah ini:



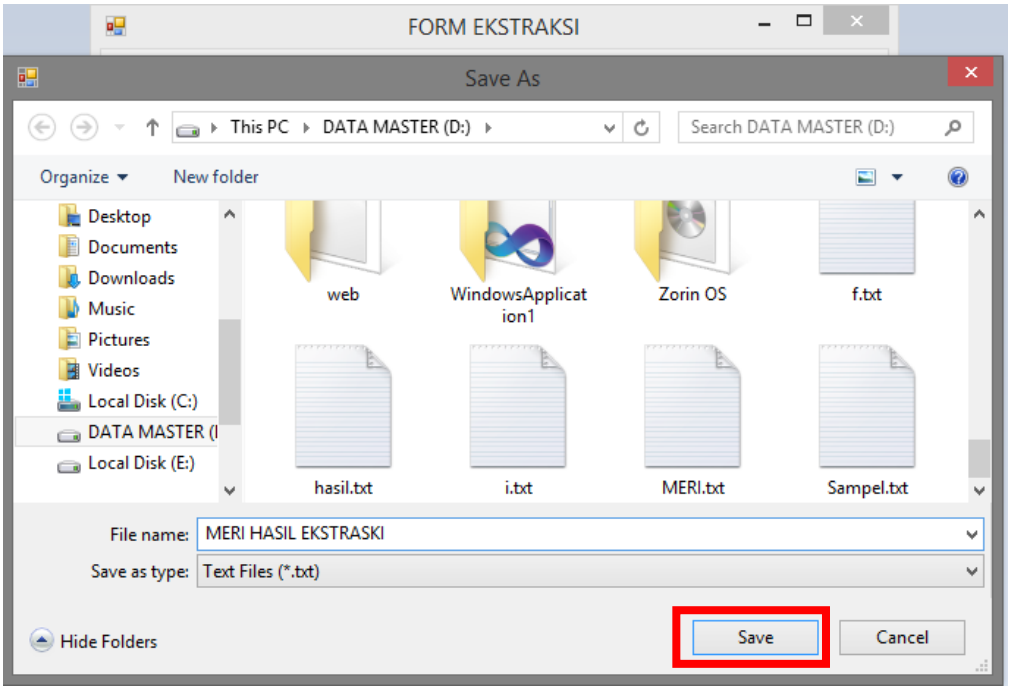
Gambar 4.17 *Pop Up* Citra Stegano Pada Menu Ekstraksi

Berdasarkan pada gambar di atas, dipilih gambar stegano hasil proses *embedding* sebelumnya dengan nama file "ujicoba.jpg" kemudian menekan *button* "open" dan menampilkan hasil seperti gambar di bawah ini:



Gambar 4.18 Citra Stegano Pada Menu Ekstraksi

Berdasarkan pada gambar 4.18, kemudian memasukan kunci yang sama saat proses *embedding*, sehingga kunci saat ekstraksi adalah karakter "kami". Proses selanjuttnya adalah melakukan ekstraksi dan menyimpan file data teks kedalam direktori penyimpanan dengan cara menekan *button* "Proses Ekstraksi / Simpan File Teks" seperti gambar di bawah ini:

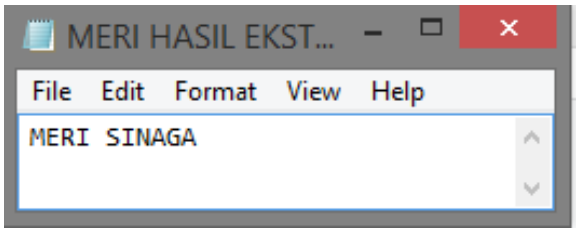


Gambar 4.19 *Pop Up* Simpan File Data Teks Pada Menu Ekstraksi

Berdasarkan pda gambar di atas, *user* menyimpan file teks hasil ekstraksi dengan nama file "MERI HASIL EKSTRAKSI" kemudian untuk memproses ekstraksi dan penyimpanan secara langsung dengan menekan *button* "*Save*". Adapun file teks yang disimpan seperti gambar di bawah ini:

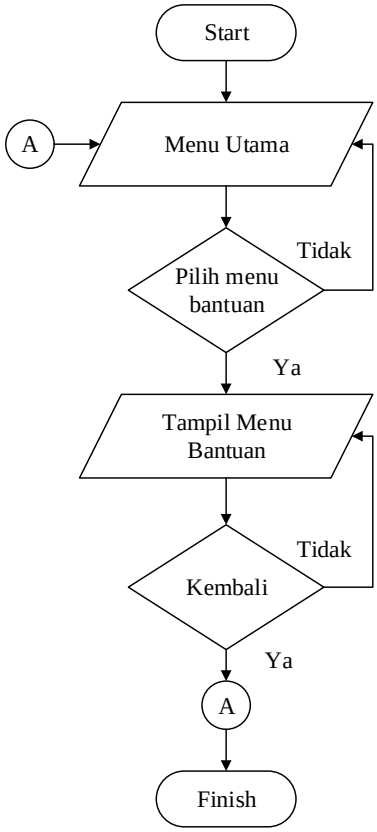
Gambar 4.20 File Data Teks Ekstraksi Pada Menu Ekstraksi

Adapun isi karakter dari file hasil ekstraksi akan kembali seperti semula yang dapat dilihat pada gambar di bawah ini:



Gambar 4.21 Isi File Data Teks Hasil Ekstraksi Pada Menu Ekstraksi

Sebelum membuka menu bantuan di aplikasi, adapun *flowchart* dari menu bantuan adalah sebagai berikut:

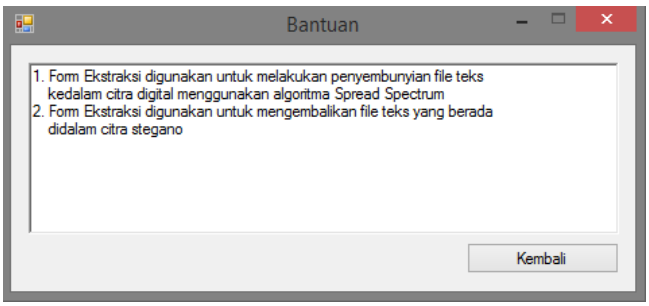


Gambar 4.22 *Flowchart* Menu Bantuan

Berdasarkan pda gambar di atas, adapun menu bantuan dapat dilihat sebagai berikut:

3. Tampilan Menu Bantuan

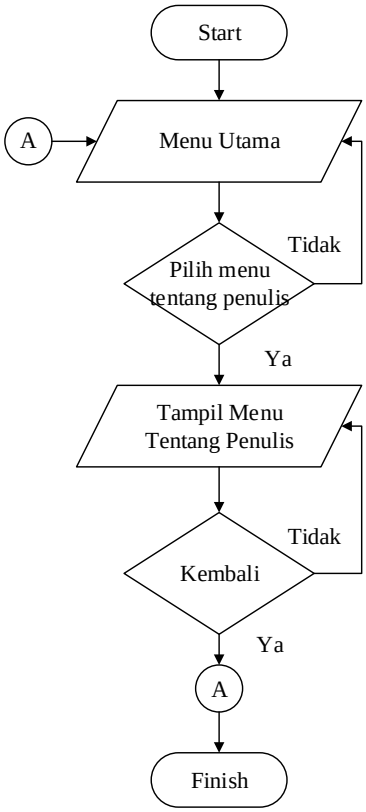
   Adapun tampilan menu bantuan dapat dilihat pada gambar di bawah ini:



Gambar 4.23 Tampilan Menu Bantuan

Berdasarkan pada gambar di atas, menu bantuan berisikan informasi tentang fungsi dari *form* proses pada aplikasi. Adapun *button* "Kembali" berfungsi untuk kembali pada menu utama.

Sebelum membuka menu tentang penulis di aplikasi, adapun *flowchart* dari menu tentang penulis adalah sebagai berikut:

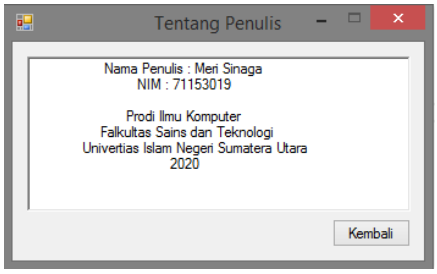


Gambar 4.24 *Flowchart* Menu Tentang Penulis

Berdasarkan pda gambar di atas, adapun menu tentang penulis dapat dilihat sebagai berikut:

4. Tampilan Menu Tentang Penulis

Adapun tampilan menu tentang penulis dapat dilihat pada gambar di bawah ini:



Gambar 4.25 Tampilan Menu Tentang Penulis

Berdasarkan pada gambar di atas, menu tentang penulis berisikan tentang penulis laporan pada aplikasi. Adapun *button* "Kembali" berfungsi untuk kembali pada menu utama.

4.4 Pembahasan Hasil Pengujian

Berdasarkan dari pengujian steganografi pada aplikasi didapatkan hasil bahwa citra digital yang menjadi objek penyisipan file data teks tidak mengalami perubahan bentuk dan warna. Hal ini tentu membuat data file teks aman didalam citra digital dikarenakan secara visual tidak ada bentuk yang mencurigakan dari gambar hasil setganografi. Sehingga kecil kemungkinan terjadi kebocoran akan kerahasian dari file teks yang akan dikirimkan kepada pihak penerima dengan objek citra digital.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penelitian yang dilakukan terhadap pengamanan pesan teks yang disembunyikan ke dalam file citra digital menggunakan algoritma *spread sprectum*, maka terdapat beberapa kesimpulan berdasarkan uraian yang telah tercantum pada bab-bab sebelumnya. Adapun kesimpulan dari hasil penelitian ini adalah sebagai berikut:

1. Pesan teks dengan format .txt dapat diamankan dengan teknik steganografi yang disembunyikan ke dalam sebuah media berupa citra digital gambar dengan format .jpg.
2. Proses penerapan algoritma *spread sprectum* dilakukan dengan data masukan yang disebar menjadi skala empat sehingga menghasilkan nilai *psoudonoise*.
3. Data citra digital hasil penyisipan tidak mengalami perubahan, hal ini dikarenakan proses penyisipan dilakukan pada bit akhir nilai biner citra digital yang hanya menambah satu nilai atau mengurangi satu nilai *pixel* citra.
4. Aplikasi yang dirancang menggunakan *Visual Studio* 2012 dengan algoritma *spread sprectum* dapat mempermudah proses pengamanan dan penyembunyian pesan teks ke dalam file citra digital.

5.2 Saran

Adapun saran-saran yang usulkan adalah sebagai berikut:

1. Dengan adanya rancangan aplikasi ini, maka diharapkan dapat mempermudah pengamanan pesan teks yang bersifat rahasiadan meminimalisir tingkat kerusakan atau kehilangan pesan teks rahasia.
2. Diharapkan adanya pengembangan lebih lanjut dari aplikasi yang dirancang dalam pengoperasian *double* pengamanan pesan teks.
3. Dapat dikembangkan menjadi aplikasi yang berbasis *web* atau *android*.

DAFTAR PUSTAKA

Achmad Noercholis & Yohanes Nugraha, 2016, *Pengamanan Data Tek Menggunakan Teknik Steganografi Spread Spectrum Berbasis Android.* Jurnal Antivirus, Vol.10, No.1, 1 Mei 2016 p-ISSN:1978-5232 e-ISSN:2527337X, 33-34

Aliy Hafiz, 2019, *Steganografi Berbasis Citra Digital Untuk Menyembunyikan Data Menggunakan Metode Least Significant Bit (LSB).* Jurnal Cendikia, Vol. XVII Cendikia 2019 P-ISSN:0216-9436 Bandar Lampung, April 2019 E-ISSN:2622-682, 194-195.

Ari Septayuda, Dr., Ir. Bambang Hidayat, DEA & Hilal Hudan Nuha, MT, 2014, *Analisis Steganografi Citra Digital Menggunakan Metode Spread Spectrum Berbasis Android.* E-Proceding of Engineering : Vol.1, No.1 Desember 2014 ISSN:2365-9365, 3.

Azkar Kumala, Bambang Pramono & Rahmat Ramadhan. 2017. Implementasi Metode Spread Spectrum Dalam Steganografi Pada File MP3 Berbasis Android. SemanTIK. Vol.3 No.2, Jul-Des 2017. Pp.127-132 ISSN:2502-8928. 128-129.

Defra Afriani, 2019, Perancangan Knowledge Managemen Sistem Dengan Seci Model Pada Layanan Perbaikan AC Mobil di Bengkel Agung Motor Cinere Menggunakan VB.Net, Jurnal Informatika SIMANTIK, Vol.4, No.1, ISSN:2541-3244, 30.

Furqan, dkk. 2020. Klasifikasi Daun Bugenvil Menggunakan Gray Level Co-Occurrence Matrix dan K-Nearest Neighbor. Jurnal Ilmiah. Vol.6 No.1, pp. 2-29, 2020.

Heri Santoso, Abdul Halim Hasugian & Yusuf Ramadhan Nasution, Aplikasi Deteksi Perubahan Wilayah dengan Menggunakan Post-Classification. Jurnal Arma Informatika. Vol 3, No.1 Juli 2019 ISSN : 2615-6891, 1-14.

Indra Gunawan & Sumarmo. 2018. Penggunaan Algoritma Kriftografi Steganografi Least Significant Bit Untuk Pengamanan Pesan Teks dan

Data Video. Jurnal Sains Komputer & Informatika. Vol.2 No.1 Maret 2018, ISSN:2548-9771/EISSN:2549-7200, 58-60.

Jogiyanto, 2005, Sistem Informasi : Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis, Yogyakarta, Andi.

Pratiwi, Dwi Atmodjo WP, 2016, Peningkatan Keamanan Data Dengan Metode Cropping Selection Pseudorandom, Jurnal Ticom, Vol.4, No.3, 132

Rahmat Adi Purnama, Adi Tri Laksana Putra, 2018, Aplikasi Web Server Berbasis Bahasa CSharp, Jurnal Teknik Komputer, Vol.4, No.1 : 22.

Riko Arlando Saragih, 2006, *Metode Parity Coding Versus Metode Spread Spectrum Pada Audio Steganography*. SNATI, ISSN:1907-5022, 72.

Rio Irawan, Ilhamsyah, Yurio Brinorman, 2015, Aplikasi Enkripsi Dan Dekripsi Pesan Singkat menggunakan Algoritma Knapseck Berbasis Android, Jurnal Coding Sistem Komputer, Vol 03, No.3, ISSNL2338-493X, 58.

Samuel Ratumurung, 2015, Sistem Informasi Akutansi Perminataan Barang Dari Gudang Pada PT Mawasa Sejahtera Ambon, Vol.ix, No.1, ISSN 1978-3612, 60.

Siti Rohayah, Ginanjar Wiro Sasmito & Oman Somantri, 2015, Aplikasi Steganografi Untuk Penyisipan Pesan. JURNAL INFORMATIKA Vol. 9, No. 1, 976-977.

Sutoyo, T., Mulyanto, E Suhartono, V., & Nurhayati, O, D., 2009, *Teori Pengolahan Citra Digital.* Yogyakarta dan Semarang: C.V ANDI OFFSET dan ANDINUS Semarang.

Tafsirq, Akses Tanggal 2 September 2020. Surat Al-‘a;idah Ayat 38 [online]. Available: https://tafsirq.com/5-Al-Ma'idah/ayat-38

Ulan Ari Anti, Awang Harsa Kridalaksana & Dyna Marisa Khairana, 2017, Steganografi Pada Video Menggunakan Metode Least Significant Bit (LSB) dan End of File (EOF), Jurnal Informatika Mulawarna, Vol.12, No.2.

Zaenal Rifai & Solichul Huda, 2013, *Aplikasi Pengamanan Data Email Dengan Teknik Steganografi*. Techno.COM, Vol. 12 No. 2, Mei 2013:73-8173, 75.

LAMPIRAN
PROSES HITUNGAN MANUAL

Karakter : UINS

Karakter di atas, dikonversikan kedalam bentuk biner seperti pada tabel berikut:

Tabel 1. Nilai Desimal dan Biner Sampel Data Teks

No	Karakter	Nilai Desimal	Biner
1	U	85	01010101
2	I	73	01001001
3	N	78	01001110
4	S	82	01010010

Berdasarkan pada tabel di atas, nilai desimal dan biner dari karakter UINS didapatkan dari melihat tabel ASCII. Selanjutnya adalah tahapan penyisipan dengan algoritma *Spread Spectrum*.

1. Penyeberan Biner Data Teks

   Berdasarkan pada tabel di atas, nilai biner karakter data teks sudah didapatkan. Selanjutnya nilai biner karakter data teks disebar menggunakan besaran skalar empat sehingga menghasilkan segmen baru seperti berikut yang berjumlah 128 bit. Adapun prosesnya adalah membuat setiap bilangan biner menjadi 4 digit bilangan biner yang sama. Contoh bilangan biner karakter U= 01010101 jika disebar dengan skalar empat menjadi 00001111 00001111 00001111 00001111. Yang berwarna merah adalah nilai biner sebenarnya. Sehingga hasil dari keseluruhan penyebaran biner skalar empat didapat 128 bit biner baru yaitu:

   00001111 00001111 00001111 00001111
   00001111 00000000 11110000 00001111
   00001111 00000000 11111111 11110000
   00001111 00001111 00000000 11110000

2. Pembangkitan *Pseudonoise*

   Pada proses pembangkitan bilangan *pseudonoise* memerlukan kata kunci. Kata kunci yang dipakai dalam proses hitungan manual ini adalah *string* “kami”. Proses *Pseudonoise* menggunakan operasi logika XOR untuk mendapatkan

bilangan yang digunakan untuk pembangkit sebuah bilangan acak. Adpaun Proses dalam mencari nilai *pseudonoise* sebagai berikut :

Tabel 2. Nilai Desimal dan Biner Kunci

Karakter Kunci	Nilai Desimal	Nilai Biner
k	107	01101011
a	97	01100001
m	109	01101101
i	105	01101001

Berdasarkan pada tabel 4.3 di atas, selanjutnya adalah melakukan XOR pada setiap bilangan biner karakter kunci. Adapun tahapanya sebagai berikut ini:

Lakukan XOR untuk setiap nilai biner kunci seperti berikut :

Biner karakter kunci “k” di XOR dengan biner karakter kunci “a”, sehingga menghasilkan nilai sebagai berikut :

01101011 XOR 01100001 = 00001010

Selanjutnya hasil XOR dari karakter “k” dan “a” di XOR kembali dengan karakter kunci “m” menghasilkan nilai sebagai berikut:

00001010 XOR 01101101 = 01100111

Selanjutnya hasil XOR dari karakter “m” XOR kembali dengan karakter kunci “i” menghasilkan nilai sebagai berikut:

01100111 XOR 01101001 = **00001110 (7)** dalam desimal

Berdasarkan hasil XOR didapat nilai 7 yang akan digunakan sebagai seed random untuk pembangkitan bilangan acak menggunakan metode *Linear Congruential Generator* (LCG). Adapun untuk membangkitan bilangan LCG ditentukan nilai sebagai berikut :

a = 17

c = 8

m = 84

Adapun proses dari metode *Linear Congruential Generator* (LCG) dalam membentuk bilangan acak sebagai berikut :

Z1 = (17 * 7 + 8) modulus 84 = 43

Z2 = (17 * 43 + 8) modulus 84 = 67

Z3 = (17 * 67 + 8) modulus 84 = 55

Z4 = (17 * 55 + 8) modulus 84 = 19

Z5 = (17 * 19 + 8) modulus 84 = 79

Untuk proses seterusnya dilakukan dengan cara yang salam hingga Z6, Z7, Z8, Z9,….. Zn. Untuk nilai biner yang akan disebar berjumlah 128 bit, sheingga nilai LCG yang diambil harus berjumlah sama, yaitu 128 bit atau 16 angka. Adapun nilai yang telah didapat adalah sebagai berikut:

43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 =01001111 , 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 = 01001111, 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011

3. Modulasi

Selanjutnya adalah melakukan teknik modulasi yang meng-XOR nilai data teks denga nilai segmen *psoudonoise.* Adapun prosesnya adalah sebagai berikut ini:

Segmen data teks 1 = 00001111000011110000111100001111

Segmen *psoudonoise* 1 = 00101011010000110011011100010011

_____XOR

00100100010011000011100000011100

Segmen data teks 2 = 00001111000000001111000000001111

Segmen *psoudonoise* 2 = 01001111000001110010101101000011

_____XOR

01000000000001111101101101001100

Segmen data teks 3 = 00001111000000001111111111110000

Segmen *psoudonoise* 3 = 00110111000100110100111100000111

_____XOR

00111000000100111011000011110111

Segmen data teks 4 = 00001111000011110000000011110000

Segmen *psoudonoise* 4 = 00101011010000110011011100010011

_____XOR

00100100010011000011011111100011

Hasil dari modulasi antara biner karakter data teks dengan biner *pseudonoise* akan disisipkan ke dalam objek sampel citra digital dengan resolusi 8 x 6 *pixel*. Adapun hasil modulasi adalah sebagai berikut :

00100100010011000011100000011100

01000000000001111101101101001100

00111000000100111011000011110111

00100100010011000011011111100011

4. Proses Penyisipan

Selanjutnya adalah melakukan proses penyisipan. Pada proses penyisipan, stegografi membutuhkan sebuah penanda yang berfungsi sebagai akhir untuk pembatas dalam pengambilan bit biner pada citra pada saat proses ekstraksi. Adapun penanda yang digunakan adalah karakter "#" yang dirubah kedalam bentuk biner **00100011**. Proses penyisipkan data teks diaplikasikan pada bit data ke-8 dari nilai biner citra digital, serta pada proses akhir menyisipkan bit penanda akhir berupa nilai biner "#". Jumlah keseluruhan nilai yang bit yang akan disisipkan adalah sebanyak 128 bit sampel karakter data teks hasil proses *spread spectrum* dan 8 bit penanda akhir sehingga total keseluruhan bit yang akan disisipkan pada sampel citra adalah 136 bit. Adapun proses penyisipan atau perpindahan nilai biner sampel karakter data tesk dapat dilihat pada tabel di bawah ini :

Tabel 3. Proses Penyisipan Bit Data Teks *Spread Spectrum*

Sampel Citra Objek				Bit Data Teks Spread Spectrum	Sampel Citra Stegano			
Pixel	Warna	Desimal	Biner		Pixel	Warna	Desimal	Biner
l	R	73	01001001	0	l	R	72	01001000
	G	108	01101100	0		G	108	01101100
	B	109	01101101	1		B	109	01101101
2	R	117	01110101	0	2	R	116	01110100
	G	109	01101101	0		G	108	01101100
	B	112	01110000	1		B	113	01110001
3	R	117	01110101	0	3	R	116	01110100
	G	116	01110100	0		G	116	01110100
	B	101	01100101	0		B	100	01100100
4	R	114	01110010	1	4	R	115	01110011
	G	80	01010000	0		G	80	01010000
	B	177	10110001	0		B	176	10110000
5	R	169	10101001	1	5	R	169	10101001
	G	180	10110100	1		G	181	10110101
	B	133	10000101	1		B	133	10000101
6	R	178	10110010	0	6	R	178	10110010
	G	109	01101101	0		G	108	01101100
	B	169	10101001	0		B	168	10101000
7	R	108	01101100	0	7	R	108	01101100
	G	120	01111000	1		G	121	01111001
	B	153	10011001	1		B	153	10011001
8	R	161	10100001	1	8	R	161	10100001
	G	123	01111011	0		G	122	01111010
	B	167	10100111	0		B	166	10100110
9	R	152	10011000	0	9	R	152	10011000
	G	177	10110001	0		G	176	10110000
	B	157	10011101	0		B	156	10011100
10	R	149	10010101	0	10	R	148	10010100
	G	137	10001001	1		G	137	10001001
	B	190	10111110	1		B	191	10111111
11	R	128	10000000	1	11	R	129	10000001
	G	169	10101001	0		G	168	10101000
	B	179	10110011	0		B	178	10110010

Lanjutan Tabel 3. Proses Penyisipan Bit Data Teks *Spread Spectrum*

12	R	190	10111110	0	12	R	190	10111110
	G	164	10100100	1		G	165	10100101
	B	169	10101001	0		B	168	10101000
13	R	177	10110001	0	13	R	176	10110000
	G	157	10011101	0		G	156	10011100
	B	121	01111001	0		B	120	01111000
14	R	180	10110100	0	14	R	180	10110100
	G	192	11000000	0		G	192	11000000
	B	168	10101000	0		B	168	10101000
15	R	177	10110001	0	15	R	176	10110000
	G	102	01100110	0		G	102	01100110
	B	112	01110000	0		B	112	01110000
16	R	150	10010110	0	16	R	150	10010110
	G	120	01111000	1		G	121	01111001
	B	189	10111101	1		B	189	10111101
17	R	177	10110001	1	17	R	177	10110001
	G	109	01101101	1		G	109	01101101
	B	150	10010110	1		B	151	10010111
18	R	120	01111000	0	18	R	120	01111000
	G	157	10011101	1		G	157	10011101
	B	137	10001001	1		B	137	10001001
19	R	120	01111000	0	19	R	120	01111000
	G	180	10110100	1		G	181	10110101
	B	177	10110001	1		B	177	10110001
20	R	167	10100111	0	20	R	166	10100110
	G	177	10110001	1		G	177	10110001
	B	180	10110100	0		B	180	10110100
…	…	…	…	…	…	…	…	…
44	R	99	01100011	0	44	R	98	01100010
	G	129	10000001	1		G	129	10000001
	B	184	10111000	0		B	184	10111000
45	R	178	10110010	0	45	R	178	10110010
	G	191	10111111	0		G	190	10111110
	B	182	10110110	1		B	183	10110111
46	R	190	10111110	1	46	R	191	10111111
	G	89	01011001			G	89	01011001

Berdasarkan pada proses penyisipan biner sampel karakter data teks, nilai desimal RGB dari setiap *pixel* sampel mengalami perubahan dari pengurangan dan penambahan nilai sebanyak 1 nilai, adapun sebagian memiliki nilai yang tetap. Hal ini terjadi dikarenakan perpindahan dilakukan pada bit akhir (ke 8) nilai *pixel* sampel sehingga citra objek tidak mengalami perbuahan bentuk dan warna yang signifikan. Adapun nilai RGB keseluruhan *pixel* citra sampel yang telah disisipkan dengan biner sampel karakter data teks menggunakan algoritma *Spread Spectrum* dapat dilihat pada tabel di bawah ini :

Tabel 4. Nilai RGB *Pixel* Sampel Objek Citra Stegano

Sampel Citra Stegano			
Pixel	Warna	Desimal	Biner
1	R	72	01001000
	G	108	01101100
	B	109	01101101
2	R	116	01110100
	G	108	01101100
	B	113	01110001
3	R	116	01110100
	G	116	01110100
	B	100	01100100
4	R	115	01110011
	G	80	01010000
	B	176	10110000
5	R	169	10101001
	G	181	10110101
	B	133	10000101
6	R	178	10110010
	G	108	01101100
	B	168	10101000
7	R	108	01101100
	G	121	01111001
	B	153	10011001

Lanjutan Tabel 4. Nilai RGB Pixel Sampel Objek Citra Stegano

8	R	161	10100001
	G	122	01111010
	B	166	10100110
9	R	152	10011000
	G	176	10110000
	B	156	10011100
10	R	148	10010100
	G	137	10001001
	B	191	10111111
11	R	129	10000001
	G	168	10101000
	B	178	10110010
12	R	190	10111110
	G	165	10100101
	B	168	10101000
13	R	176	10110000
	G	156	10011100
	B	120	01111000
14	R	180	10110100
	G	192	11000000
	B	168	10101000
15	R	176	10110000
	G	102	01100110
	B	112	01110000
16	R	150	10010110
	G	121	01111001
	B	189	10111101
17	R	177	10110001
	G	109	01101101
	B	151	10010111
18	R	120	01111000
	G	157	10011101
	B	137	10001001

Lanjutan Tabel 4. Nilai RGB *Pixel* Sampel Objek Citra Stegano

19	R	120	01111000
	G	181	10110101
	B	177	10110001
20	R	166	10100110
	G	177	10110001
	B	180	10110100
...	...	…	…
44	R	98	1100010
	G	129	10000001
	B	184	10111000
45	R	178	10110010
	G	190	10111110
	B	183	10110111
46	R	191	10111111
	G	89	1011001
	B	145	10010001

LAMPIRAN
PROSES HITUNGAN MANUAL

Karakter : NAGA

Karakter di atas, dikonversikan kedalam bentuk biner seperti pada tabel berikut:

Tabel 1. Nilai Desimal dan Biner Sampel Data Teks

No	Karakter	Nilai Desimal	Biner
1	N	78	01001110
2	A	65	01000001
3	G	71	01000111
4	A	65	01000001

Berdasarkan pada tabel di atas, nilai desimal dan biner dari karakter NAGA didapatkan dari melihat tabel ASCII. Selanjutnya adalah tahapan penyisipan dengan algoritma *Spread Spectrum*.

5. Penyeberan Biner Data Teks

   Berdasarkan pada tabel di atas, nilai biner karakter data teks sudah didapatkan. Selanjutnya nilai biner karakter data teks disebar menggunakan besaran skalar empat sehingga menghasilkan segmen baru seperti berikut yang berjumlah 128 bit. Adapun prosesnya adalah membuat setiap bilangan biner menjadi 4 digit bilangan biner yang sama. Contoh bilangan biner karakter N= 01001110 jika disebar dengan skalar empat menjadi 00001111 0000000 11111111 11110000. Yang berwarna merah adalah nilai biner sebenarnya. Sehingga hasil dari keseluruhan penyebaran biner skalar empat didapat 128 bit biner baru yaitu:

   00001111 00000000 11111111 11110000
   00001111 00000000 00000000 00001111
   00001111 00000000 00001111 11111111
   00001111 00000000 00000000 00001111

6. Pembangkitan *Pseudonoise*

   Pada proses pembangkitan bilangan *pseudonoise* memerlukan kata kunci. Kata kunci yang dipakai dalam proses hitungan manual ini adalah *string* “kami”. Proses *Pseudonoise* menggunakan operasi logika XOR untuk mendapatkan

bilangan yang digunakan untuk pembangkit sebuah bilangan acak. Adpaun Proses dalam mencari nilai *pseudonoise* sebagai berikut :

Tabel 2. Nilai Desimal dan Biner Kunci

Karakter Kunci	Nilai Desimal	Nilai Biner
k	107	01101011
a	97	01100001
m	109	01101101
i	105	01101001

Berdasarkan pada tabel 4.3 di atas, selanjutnya adalah melakukan XOR pada setiap bilangan biner karakter kunci. Adapun tahapanya sebagai berikut ini:

Lakukan XOR untuk setiap nilai biner kunci seperti berikut :

Biner karakter kunci “k” di XOR dengan biner karakter kunci “a”, sehingga menghasilkan nilai sebagai berikut :

01101011 XOR 01100001 = 00001010

Selanjutnya hasil XOR dari karakter “k” dan “a” di XOR kembali dengan karakter kunci “m” menghasilkan nilai sebagai berikut:

00001010 XOR 01101101 = 01100111

Selanjutnya hasil XOR dari karakter “m” XOR kembali dengan karakter kunci “i” menghasilkan nilai sebagai berikut:

01100111 XOR 01101001 = **00001110 (7)** dalam desimal

Berdasarkan hasil XOR didapat nilai 7 yang akan digunakan sebagai seed random untuk pembangkitan bilangan acak menggunakan metode *Linear Congruential Generator* (LCG). Adapun untuk membangkitan bilangan LCG ditentukan nilai sebagai berikut :

a = 17

c = 8

m = 84

Adapun proses dari metode *Linear Congruential Generator* (LCG) dalam membentuk bilangan acak sebagai berikut :

Z1 = (17 * 7 + 8) modulus 84 = 43

Z2 = (17 * 43 + 8) modulus 84 = 67

Z3 = (17 * 67 + 8) modulus 84 = 55

Z4 = (17 * 55 + 8) modulus 84 = 19

Z5 = (17 * 19 + 8) modulus 84 = 79

Untuk proses seterusnya dilakukan dengan cara yang salam hingga Z6, Z7, Z8, Z9,….. Zn. Untuk nilai biner yang akan disebar berjumlah 128 bit, sheingga nilai LCG yang diambil harus berjumlah sama, yaitu 128 bit atau 16 angka. Adapun nilai yang telah didapat adalah sebagai berikut:

43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 =01001111 , 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011, 79 = 01001111, 7 = 00000111, 43 = 00101011, 67 = 01000011, 55 = 00110111, 19 = 00010011

7. Modulasi

Selanjutnya adalah melakukan teknik modulasi yang meng-XOR nilai data teks denga nilai segmen *psoudonoise.* Adapun prosesnya adalah sebagai berikut ini:

Segmen data teks 1 = 00001111000000001111111111110000

Segmen *psoudonoise* 1 = 00101011010000110011011100010011

_____XOR

00100100010000111100100011100011

Segmen data teks 2 = 00001111000000000000000000001111

Segmen *psoudonoise* 2 = 01001111000001110010101101000011

_____XOR

01000000000001110010101101001100

Segmen data teks 3 = 00001111000000000000111111111111

Segmen *psoudonoise* 3 = 00110111000100110100111100000111

_____XOR

00111000000100110100000011111000

Segmen data teks 4 = 00001111000000000000000000001111

Segmen *psoudonoise* 4 = 00101011010000110011011100010011

_____XOR

00100100010000110011011100011100

Hasil dari modulasi antara biner karakter data teks dengan biner *pseudonoise* akan disisipkan ke dalam objek sampel citra digital dengan resolusi 8 x 6 *pixel*. Adapun hasil modulasi adalah sebagai berikut :

00100100010000111100100011100011

01000000000001110010101101001100

00111000000100110100000011111000

00100100010000110011011100011100

8. Proses Penyisipan

Selanjutnya adalah melakukan proses penyisipan. Pada proses penyisipan, stegografi membutuhkan sebuah penanda yang berfungsi sebagai akhir untuk pembatas dalam pengambilan bit biner pada citra pada saat proses ekstraksi. Adapun penanda yang digunakan adalah karakter "#" yang dirubah kedalam bentuk biner **00100011**. Proses penyisipkan data teks diaplikasikan pada bit data ke-8 dari nilai biner citra digital, serta pada proses akhir menyisipkan bit penanda akhir berupa nilai biner "#". Jumlah keseluruhan nilai yang bit yang akan disisipkan adalah sebanyak 128 bit sampel karakter data teks hasil proses *spread spectrum* dan 8 bit penanda akhir sehingga total keseluruhan bit yang akan disisipkan pada sampel citra adalah 136 bit. Adapun proses penyisipan atau perpindahan nilai biner sampel karakter data tesk dapat dilihat pada tabel di bawah ini :

Tabel 3. Proses Penyisipan Bit Data Teks *Spread Spectrum*

Sampel Citra Objek				Bit Data Teks Spread Spectrum	Sampel Citra Stegano			
Pixel	Warna	Desimal	Biner		Pixel	Warna	Desimal	Biner
l	R	73	01001001	0	l	R	72	01001000
	G	108	01101100	0		G	108	01101100
	B	109	01101101	1		B	109	01101101
2	R	117	01110101	0	2	R	116	01110100
	G	109	01101101	0		G	108	01101100
	B	112	01110000	1		B	113	01110001
3	R	117	01110101	0	3	R	116	01110100
	G	116	01110100	0		G	116	01110100
	B	101	01100101	0		B	100	01100100
4	R	114	01110010	1	4	R	115	01110011
	G	80	01010000	0		G	80	01010000
	B	177	10110001	0		B	176	10110000
5	R	169	10101001	0	5	R	168	10101000
	G	180	10110100	0		G	180	10110100
	B	133	10000101	1		B	133	10000101
6	R	178	10110010	1	6	R	179	10110011
	G	109	01101101	1		G	109	01101101
	B	169	10101001	1		B	169	10101001
7	R	108	01101100	0	7	R	108	01101100
	G	120	01111000	0		G	120	01111000
	B	153	10011001	1		B	153	10011001
8	R	161	10100001	0	8	R	160	10100000
	G	123	01111011	0		G	122	01111010
	B	167	10100111	0		B	166	10100110
9	R	152	10011000	1	9	R	153	10011001
	G	177	10110001	1		G	177	10110001
	B	157	10011101	1		B	157	10011101
10	R	149	10010101	0	10	R	148	10010100
	G	137	10001001	0		G	136	10001000
	B	190	10111110	0		B	190	10111110
11	R	128	10000000	1	11	R	129	10000001
	G	169	10101001	1		G	169	10101001
	B	179	10110011	0		B	178	10110010

Lanjutan Tabel 3. Proses Penyisipan Bit Data Teks *Spread Spectrum*

12	R	190	10111110	1	12	R	191	10111111
	G	164	10100100	0		G	165	10100101
	B	169	10101001	0		B	168	10101000
13	R	177	10110001	0	13	R	176	10110000
	G	157	10011101	0		G	156	10011100
	B	121	01111001	0		B	120	01111000
14	R	180	10110100	0	14	R	180	10110100
	G	192	11000000	0		G	192	11000000
	B	168	10101000	0		B	168	10101000
15	R	177	10110001	0	15	R	176	10110000
	G	102	01100110	0		G	102	01100110
	B	112	01110000	0		B	112	01110000
16	R	150	10010110	1	16	R	151	10010111
	G	120	01111000	1		G	121	01111001
	B	189	10111101	1		B	189	10111101
17	R	177	10110001	0	17	R	176	10110000
	G	109	01101101	0		G	108	01101100
	B	150	10010110	1		B	151	10010111
18	R	120	01111000	0	18	R	120	01111000
	G	157	10011101	1		G	157	10011101
	B	137	10001001	0		B	136	10001000
19	R	120	01111000	1	19	R	121	01111001
	G	180	10110100	1		G	181	10110101
	B	177	10110001	0		B	176	10110000
20	R	167	10100111	1	20	R	167	10100111
	G	177	10110001	0		G	176	10110000
	B	180	10110100	0		B	180	10110100
…	…	…	…	…	…	…	…	…
44	R	99	01100011	0	44	R	98	01100010
	G	129	10000001	1		G	129	10000001
	B	184	10111000	0		B	184	10111000
45	R	178	10110010	0	45	R	178	10110010
	G	191	10111111	0		G	190	10111110
	B	182	10110110	1		B	183	10110111
46	R	190	10111110	1	46	R	191	10111111
	G	89	01011001			G	89	01011001

Berdasarkan pada proses penyisipan biner sampel karakter data teks, nilai desimal RGB dari setiap *pixel* sampel mengalami perubahan dari pengurangan dan penambahan nilai sebanyak 1 nilai, adapun sebagian memiliki nilai yang tetap. Hal ini terjadi dikarenakan perpindahan dilakukan pada bit akhir (ke 8) nilai *pixel* sampel sehingga citra objek tidak mengalami perbuahan bentuk dan warna yang signifikan. Adapun nilai RGB keseluruhan *pixel* citra sampel yang telah disisipkan dengan biner sampel karakter data teks menggunakan algoritma *Spread Spectrum* dapat dilihat pada tabel di bawah ini :

Tabel 4. Nilai RGB *Pixel* Sampel Objek Citra Stegano

Sampel Citra Stegano			
Pixel	Warna	Desimal	Biner
1	R	72	01001000
	G	108	01101100
	B	109	01101101
2	R	116	01110100
	G	108	01101100
	B	113	01110001
3	R	116	01110100
	G	116	01110100
	B	100	01100100
4	R	115	01110011
	G	80	01010000
	B	176	10110000
5	R	168	10101000
	G	180	10110100
	B	133	10000101
6	R	179	10110011
	G	109	01101101
	B	169	10101001
7	R	108	01101100
	G	120	01111000
	B	153	10011001

Lanjutan Tabel 4. Nilai RGB Pixel Sampel Objek Citra Stegano

8	R	160	10100000
	G	122	01111010
	B	166	10100110
9	R	153	10011001
	G	177	10110001
	B	157	10011101
10	R	148	10010100
	G	136	10001000
	B	190	10111110
11	R	129	10000001
	G	169	10101001
	B	178	10110010
12	R	191	10111111
	G	165	10100101
	B	168	10101000
13	R	176	10110000
	G	156	10011100
	B	120	01111000
14	R	180	10110100
	G	192	11000000
	B	168	10101000
15	R	176	10110000
	G	102	01100110
	B	112	01110000
16	R	151	10010111
	G	121	01111001
	B	189	10111101
17	R	176	10110000
	G	108	01101100
	B	151	10010111
18	R	120	01111000
	G	157	10011101
	B	136	10001000

Lanjutan Tabel 4. Nilai RGB *Pixel* Sampel Objek Citra Stegano

19	R	121	01111001
	G	181	10110101
	B	176	10110000
20	R	167	10100111
	G	176	10110000
	B	180	10110100
...	...	…	…
44	R	98	1100010
	G	129	10000001
	B	184	10111000
45	R	178	10110010
	G	190	10111110
	B	183	10110111
46	R	191	10111111
	G	89	1011001
	B	145	10010001

LISTING PROGRAM

Lampiran 2

1. Proses Embedding

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TubesStegano
{
    public partial class SteganoForm : Form
    {
        private Bitmap bmp1 = null;
        private Bitmap bmp2 = null;
        private string dataText = string.Empty;
        private string fileImage = string.Empty;
        private string extractedText = string.Empty;
        private int sizeEmbedded, sizeFileTxt;
        private Steganography SteganoStandar = new Steganography();
        private SteganoNinePixelDiff SteganoNine = new SteganoNinePixelDiff();
        private SteganoFourPixelDiff SteganoFour = new SteganoFourPixelDiff();

        public SteganoForm()
        {
            InitializeComponent();
        }
        private void openImgButton_Click(object sender, EventArgs e)
        {
            OpenFileDialog open_dialog = new OpenFileDialog();
            open_dialog.Filter = "jpg Files| *.jpg";

            if (open_dialog.ShowDialog() == DialogResult.OK)
            {
                initialImage.Image = Image.FromFile(open_dialog.FileName);
                bmp1 = (Bitmap)initialImage.Image;

                if (ModeBox.SelectedIndex == 0)
                {
```

```
            fileImage = open_dialog.FileName.ToString();
            SteganoStandar.setfileName(fileImage);
            SteganoStandar.setImage(bmp1);
            sizeEmbedded = SteganoStandar.getMaxMsgSize();
            maxBitsLabel.Visible = true;
            countLabel.Text = sizeEmbedded.ToString() + " bits";
            countLabel.Visible = true;
        }
        else if (ModeBox.SelectedIndex == 1)
        {
            fileImage = open_dialog.FileName.ToString();

        }
        else if (ModeBox.SelectedIndex == 2)
        {
            fileImage = open_dialog.FileName.ToString();
            SteganoNine.setImage(bmp1);
            sizeEmbedded = SteganoNine.getMaxMsgSize();
            maxBitsLabel.Visible = true;
            countLabel.Text = sizeEmbedded.ToString() + " bits";
            countLabel.Visible = true;
        }
    }
}

private void browseButton_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "File Teks|*.txt";
    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        fileNameBox.Text = open_dialog.FileName.ToString();
        FileInfo fi = new FileInfo(open_dialog.FileName);
        sizeFileTxt = (int)fi.Length;
        dataText = File.ReadAllText(open_dialog.FileName).ToString();
    }
}

private void fileNameBox_Click(object sender, EventArgs e)
{
    browseButton_Click(sender, e);
}

private void EmbeddingButton_Click(object sender, EventArgs e)
{
    bmp1 = (Bitmap)initialImage.Image;

    if (bmp1 == null)
    {
        MessageBox.Show("You have to load the image first");
        return;
    }

    if (dataText.Equals(""))
    {
```

```
                MessageBox.Show("The text you want to hide can't be empty",
"Warning");
                return;
            }
            else
            {
                if (ModeBox.SelectedIndex == 0)
                {
                    sizeEmbedded = SteganoStandar.getMaxMsgSize();
                    if ((sizeFileTxt - 8) > sizeEmbedded)
                    {
                        MessageBox.Show("The file is too large to hide");
                        return;
                    }
                }
                else if (ModeBox.SelectedIndex == 1)
                {
                    SteganoFour.setCoverObject((Bitmap)initialImage.Image);
                    sizeEmbedded = SteganoFour.getMaxMsgSize();
                    if ((sizeFileTxt - 8) > sizeEmbedded)
                    {
                        MessageBox.Show("The file is too large to hide");
                        return;
                    }
                    SteganoFour.setMessage(dataText);
                }
                else if (ModeBox.SelectedIndex == 2)
                {
                    sizeEmbedded = SteganoNine.getMaxMsgSize();
                    if ((sizeFileTxt - 8) > sizeEmbedded)
                    {
                        MessageBox.Show("The file is too large to hide");
                        return;
                    }
                    SteganoNine.setMessage(dataText);
                }
            }

            if (keyBox.Text == "")
            {
                MessageBox.Show("Key must be provide");
                return;
            }
            else if (keyBox.Text.Length < 4)
            {
                MessageBox.Show("Password at least 4 character");
                return;
            }

        private void saveSteganoImgButton_Click(object sender, EventArgs e)
        {
            bmp2 = (Bitmap)steganoImg.Image;

            if (bmp2 != null)
            {
```

```
                SaveFileDialog save_dialog = new SaveFileDialog();
                save_dialog.Filter = "jpg Image|*.jpg";

                if (save_dialog.ShowDialog() == DialogResult.OK)
                {
                    bmp2.Save(save_dialog.FileName, ImageFormat.Bmp);
                }
            }
        }

        private void openSteganoImgButton_Click(object sender, EventArgs e)
        {
            OpenFileDialog open_dialog = new OpenFileDialog();
            open_dialog.Filter = "jpg Files|*.jpg";

            if (open_dialog.ShowDialog() == DialogResult.OK)
            {
                steganoImg.Image = Image.FromFile(open_dialog.FileName);
                bmp1 = (Bitmap)steganoImg.Image;
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Form2 f2 = new Form2();
            f2.Show();
            this.Close();
        }

    }
}
```

2. Proses Ekstraksi

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TubesStegano
{
    public partial class Form3 : Form
    {
        private Bitmap bmp1 = null;
        private Bitmap bmp2 = null;
        private string dataText = string.Empty;
        private string fileImage = string.Empty;
        private string extractedText = string.Empty;
```

```
private int sizeEmbedded, sizeFileTxt;
private Steganography SteganoStandar = new Steganography();
private SteganoNinePixelDiff SteganoNine = new SteganoNinePixelDiff();
private SteganoFourPixelDiff SteganoFour = new SteganoFourPixelDiff();
public Form3()
{
    InitializeComponent();
}

private void openSteganoImgButton_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "jpg Files|*.jpg";

    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        steganoImg.Image = Image.FromFile(open_dialog.FileName);
        bmp1 = (Bitmap)steganoImg.Image;
    }
}



private void Ekstraksi_Click(object sender, EventArgs e)
{
    bmp2 = (Bitmap)steganoImg.Image;

    if (keyBox.Text == "")
    {
        MessageBox.Show("Key must be provide");
        return;
    }
    else if (keyBox.Text.Length < 4)
    {
        MessageBox.Show("Password at least 4 character");
        return;
    }

    if (bmp2 != null)
    {
        if (ModeBox.SelectedIndex == 0)
        {
            SteganoStandar.clear();
            SteganoStandar.setKey(keyBox.Text);
            extractedText = SteganoStandar.extractText(bmp2);
        }
        else if (ModeBox.SelectedIndex == 1)
        {

        }

        else if (ModeBox.SelectedIndex == 2)
        {
            SteganoNine.setImage(bmp2);
            extractedText = SteganoNine.extractText(bmp2);
```

```
                }

                extractedText = (extractedText, keyBox.Text);

                SaveFileDialog save_dialog = new SaveFileDialog();
                save_dialog.Filter = "Text Files|*.txt";

                if (save_dialog.ShowDialog() == DialogResult.OK)
                {
                    File.WriteAllText(save_dialog.FileName, extractedText);
                }

            }
        }

    }
}
```

3. Algoritma Steganograpy

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.IO;

namespace TubesStegano
{
    class Steganography
    {
        public enum State
        {
            Hiding,
            Filling_With_Zeros
        };

        private string fileName;
        private string message;
        private string key;
        private Bitmap bmp;
        private List<Point> koorSeed = new List<Point>();
        private int counter = 0;

        public void setfileName(string name)
        {
            fileName = name;
        }

        public void setMessage(string text)
```

```
        {
            message = text;
        }

        public void setImage(Bitmap b)
        {
            bmp = b;
        }

        public void setKey(string s)
        {
            key = s;
        }

        public Bitmap embedText()
        {
            // pertama kita akan melakukan penyisipan, state nya hiding
            State state = State.Hiding;

            int charIndex = 0;
            int charValue = 0;
            long pixelElementIndex = 0;
            int zeros = 0;
            int R = 0, G = 0, B = 0;
            Point koordinat = new Point();
            koordinat.setPoint(0,0);

            Bitmap cover = bmp;

            if (isGrayScale())
            {
                // Pemrosesan grayscale image

                while (counter < getMaxMsgSize())
                {
                    koordinat = getSeed(koordinat);

                    Color pixel = cover.GetPixel(koordinat.getX(),
koordinat.getY());

                    // kosongkan bit terakhir
                    R = pixel.R - pixel.R % 2;

                    // cek apakah seluruh 8 bit sudah diproses atau belum
                    if (pixelElementIndex % 8 == 0)
                    {
                        if (state == State.Filling_With_Zeros && zeros == 8)
                        {
                            if ((pixelElementIndex - 1) % 3 < 2)
                            {
                                cover.SetPixel(koordinat.getX(),
koordinat.getY(), Color.FromArgb(R));
                            }

                            counter = 0;
```

```
                        return cover;
                    }

                    if (charIndex >= message.Length)
                    {
                        state = State.Filling_With_Zeros;
                    }
                    else
                    {
                        charValue = message[charIndex++];
                    }
                }

                if (state == State.Hiding)
                {
                    R += charValue % 2;

                    charValue /= 2;
                }

                cover.SetPixel(koordinat.getX(), koordinat.getY(),
Color.FromArgb(R,R,R));

                pixelElementIndex++;

                if (state == State.Filling_With_Zeros)
                {
                    zeros++;
                }
                counter++;
            }
        }
        else
        {
            // pemrosesan RGB image

            while (counter < getMaxMsgSize())
            {
                koordinat = getSeed(koordinat);

                Color pixel = cover.GetPixel(koordinat.getX(),
koordinat.getY());

                // kosongkan bit terakhir
                R = pixel.R - pixel.R % 2;
                G = pixel.G - pixel.G % 2;
                B = pixel.B - pixel.B % 2;

                for (int n = 0; n < 3; n++)
                {
                    // cek apakah seluruh 8 bit sudah diproses atau belum
                    if (pixelElementIndex % 8 == 0)
                    {
                        if (state == State.Filling_With_Zeros && zeros == 8)
                        {
```

```
                                if ((pixelElementIndex - 1) % 3 < 2)
                                {
                                    cover.SetPixel(koordinat.getX(),
koordinat.getY(), Color.FromArgb(R, G, B));
                                }

                                counter = 0;
                                return cover;
                            }

                            if (charIndex >= message.Length)
                            {
                                state = State.Filling_With_Zeros;
                            }
                            else
                            {
                                charValue = message[charIndex++];
                            }
                        }

                        switch (pixelElementIndex % 3)
                        {
                            case 0:
                                {
                                    if (state == State.Hiding)
                                    {
                                        R += charValue % 2;
                                        charValue /= 2;
                                    }
                                } break;
                            case 1:
                                {
                                    if (state == State.Hiding)
                                    {
                                        G += charValue % 2;

                                        charValue /= 2;
                                    }
                                } break;
                            case 2:
                                {
                                    if (state == State.Hiding)
                                    {
                                        B += charValue % 2;

                                        charValue /= 2;
                                    }

                                    cover.SetPixel(koordinat.getX(),
koordinat.getY(), Color.FromArgb(R, G, B));
                                } break;
                        }

                        pixelElementIndex++;
```

```
                    if (state == State.Filling_With_Zeros)
                    {
                        zeros++;
                    }
                }
                counter++;
            }
        }

        return cover;
    }

    public String extractText(Bitmap cover)
    {
        int colorUnitIndex = 0;
        int charValue = 0;
        Point koordinat = new Point();
        koordinat.setPoint(0, 0);
        string extractedText = String.Empty;

        if (isGrayScale())
        {
            // pemrosesan grayscale

            while (true)
            {
                koordinat = getSeed(koordinat);

                Color pixel = cover.GetPixel(koordinat.getX(),
koordinat.getY());

                charValue = charValue * 2 + pixel.R % 2;

                colorUnitIndex++;

                if (colorUnitIndex % 8 == 0)
                {
                    charValue = reverseBits(charValue);

                    if (charValue == 0)
                    {
                        counter = 0;
                        return extractedText;
                    }

                    char c = (char)charValue;

                    extractedText += c.ToString();
                }

                counter++;
            }
        }
        else
        {
```

```
                // pemrosesan RGB
                while (true)
                {
                    koordinat = getSeed(koordinat);

                    Color pixel = cover.GetPixel(koordinat.getX(),
koordinat.getY());

                    for (int n = 0; n < 3; n++)
                    {
                        switch (colorUnitIndex % 3)
                        {
                            case 0:
                                {
                                    charValue = charValue * 2 + pixel.R % 2;
                                } break;
                            case 1:
                                {
                                    charValue = charValue * 2 + pixel.G % 2;
                                } break;
                            case 2:
                                {
                                    charValue = charValue * 2 + pixel.B % 2;
                                } break;
                        }

                        colorUnitIndex++;

                        if (colorUnitIndex % 8 == 0)
                        {
                            charValue = reverseBits(charValue);

                            if (charValue == 0)
                            {
                                counter = 0;
                                return extractedText;
                            }

                            char c = (char)charValue;

                            extractedText += c.ToString();
                        }
                    }
                    counter++;
                }
            }

            return extractedText;
        }

        private int reverseBits(int n)
        {
            int result = 0;

            for (int i = 0; i < 8; i++)
```

```
            {
                result = result * 2 + n % 2;

                n /= 2;
            }

            return result;
        }

        // get random seed spread spectrum
        private Point getSeed(Point koor)
        {
            Point A = new Point();
            bool cek = false;
            int x, y, a ,b;
            x = key[0] + key[2];
            y = key[1] + key[3];
            a = koor.getX() + x;
            b = koor.getY() + y;
            do
            {
                a += 1;
                b += 1;
                if (a >= bmp.Width)
                {
                    a = a - bmp.Width;
                }
                if (b >= bmp.Height)
                {
                    b = b - bmp.Height;
                }
                A.setPoint(a, b);
                if (!sudahAdaPoint(A))
                {
                    cek = true;
                    koorSeed.Add(A);
                }
            } while (!cek);
            return A;
        }

        public void clear()
        {
            koorSeed.Clear();
            counter = 0;
        }

        // cek apakah point sudah pernah dimunculkan atau belum
        private Boolean sudahAdaPoint(Point X)
        {
            bool cek = false;
            for (int i = 0; i < counter; i++)
            {
                if ((X.getX() == koorSeed[i].getX()) && (X.getY() ==
koorSeed[i].getY()))
```

```
                {
                    cek = true;
                }
            }
            return cek;
        }

        // mendapatkan maksimum ukuran pesan pada gambar bitmaps
        public int getMaxMsgSize()
        {
            int size;
            if (isGrayScale())
            {
                // maksimum pesan di grayscale
                size = bmp.Height * bmp.Width;
            }
            else
            {
                // maksimum pesan di RGB
                size = bmp.Height * bmp.Width * 3;
            }

            return size;
        }

        // cek apakah bitmap merupakan grayscale atau RGB
        // mengembalikan true jika grayscale
        // mengembalikan false jika RGB
        public Boolean isGrayScale()
        {
            Boolean cek = true;

            // Buka file gambar
            FileStream inStream = new FileStream(fileName, FileMode.Open,
FileAccess.Read);

            byte[] buffer = new byte[2];
            inStream.Seek(28, 0);
            inStream.Read(buffer, 0, 2);
            Int16 nBit = BitConverter.ToInt16(buffer, 0);

            if (nBit == 8) { /*true grayscale, do nothing*/ }
            else cek = false;

            return cek;
        }

    }
}
```

DAFTAR RIWAYAT HIDUP

(CURRICULUM VITAE)

Nama	: Meri Sinaga
Nim	: 71153019
Tempat, Tanggal Lahir	: Semadam Baru, 03 Agustus 1997
Jenis Kelamin	: Perempuan
Alamat	: Sepakat Segenep
Kel/Desa	: Sepakat Segenep
Kecamatan	: Semadam
Kabupaten	: Aceh Tenggara
Agama	: Islam
Status Nikah	: Belum Menikah
No HP	: 082360028661
Nama Orangtua	
Ayah	: Muhammad Samin
Ibu	: Siti Aisah

PENDIDIKAN FORMAL

2003-2009 : SD Negeri 3 Semadam

2009-2012 : SMP Negeri 5 Lawe Sigala-gala

2012-2015 : SMA Negeri Semadam

2015-2020 : Universitas Islam Negeri Sumatera Utara Medan

KARTU BIMBINGAN SKRIPSI

Semester Gasal/Genap Tahun Akademik ...../.....

Nama	: MERI SINAGA	Pembimbing I	: Dr. Mhd Furqan S.Si, M.Comp.Sc
NIM	: 71153019	Pembimbing II	: Yusuf Ramadhan Nasution, M.Kom
Prog. Studi	: ILMU KOMPUTER	SK Pembimbing	:

Judul Skripsi : Implementasi Steganografi Menggunakan Metode Spread Spectrum Dalam Pengamanan Data Teks Pada Citra Digital

PERT	PEMBIMBING I			PEMBIMBING II		
	Tgl.	Materi Bimbingan	Tanda Tangan	Tgl.	Materi Bimbingan	Tanda Tangan
I	30/1/2020	Perbaiki Latar Belakang		15/1/2020	Perbaiki Bab I	
II	31/1/2020	Perbaiki Bab II		20/1/2020	Lanjut Bab III	
III	05/2/2020	Revisi Tinjauan Pustaka		20/1/2020	Perbaiki Bab VI + Tambahkan Referensi Database	
IV	06/2/2020	Perbaikan Bab III		23/1/2020	Lanjut Bab III	
V	07/2/2020	Perbaiki Flowchart		28/1/2020	Perbaiki Bab III	

VI	10/2/2020	ACC Seminar Proposal		4/2/2020	Acc. Seminar Proposal	
VII		Abstrak Bab IV			Acc. Siang Manajemen	
VIII		Bab V, Daftar pustaka				
IX		Kesimpulan Perian jurnal.				
X		acc sidang manajemen.		1		

Medan, 07 Agustus 2020
An. Dekan
Ketua Jurusan/Program Studi
Ilmu komputer

Dr. Muhd. Furqan, S.Si, M.Comp.Sc
NIP. 19800806 200604 1003

Catatan: *Pada saat bimbingan, kartu ini harus diisi dan ditandatangani oleh pembimbing*