APLIKASI KEAMANAN DATA TEKS MENGGUNAKAN METODE TINY ENCRYPTION ALGORITHM (TEA) DAN LEAST SIGNIFICANT BITS (LSB)

SKRIPSI

Disusun Oleh :

Mochamad Aries Setyawan

12.18.236

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2016

LEMBAR PERSETUJUAN

APLIKASI KEAMANAN DATA TEKS MENGGUNAKAN METODE TINY ENCRYPTION ALGORITHM (TEA) DAN LEAST SIGNIFICANT BITS (LSB)

SKRIPSI

Disusun dan Diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai Gelar Sarjana Komputer Strata Satu (S-1)

Disusun oleh :

Mochamad Aries Setyawan

12.18.236

Diperiksa dan disetujui oleh

Dosen Pembimbing I	Dosen Pembimbing II
Joseph Dedy Irawan, ST,MT.	**Sandy Nataly Mantja, S.Kom**
NIP. 1974041620050110002	**NIP.P. 1030800418**

Mengetahui,

Program Studi Teknik Informatika S-1

Ketua

Joseph Dedy Irawan, ST, MT

NIP. 1974041620050110002

PROGRAM STUDI TEKNIK INFORMATIKA S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2016

LEMBAR KEASLIAN

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama : Mochamad Aries Setyawan
NIM : 12.18.236

Menyatakan dengan sebenar-benarnya bahwa segala pernyataan dalam skripsi yang berjudul "**Aplikasi Keamanan Data Teks Menggunakan Metode Tiny Encryption Algorthm Dan Least Significant Bits**" merupakan gagasan dan hasil karya sendiri dengan arahan komisi pembimbing dan belum pernah diajukan dalam bentuk apapun pada perguruan tinggi manapun. Semua data dan informasi yang digunakan telah dinyatakan secara jelas dan dapat diperiksa kebenarannya.

Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan penulis lain telah disebutkan dalam naskah dan dicantumkan dalam daftar pustaka di bagian akhir skripsi ini.

Malang, 25 Januari 2016

METERAI TEMPEL
8F2C8ADF687825534
6000
ENAM RIBU RUPIAH

Mochamad Aries Setyawan
NIM 12.18.236

APLIKASI KEAMANAN DATA TEKS MENGGUNAKAN METODE TINY ENCRYPTION ALGORTIHM (TEA) DAN LEAST SIGNIFICANT BITS (LSB)

Dosen Pembimbing I : Joseph Dedy Irawan, ST, MT
Dosen Pembimbing II : Sandy Nataly Mantja, S.Kom

ABSTRAK

Masalah keamanan dan kerahasiaan data merupakan hal yang sangat pentung dalam suatu organisasi maupun pribadi. Hal tersebut tentu saja akan menimbulkan risiko bilamana informasi yang sensitif dan berharga tersebut diakases oleh orang-orang yang tidak berhak. Oleh karena itu, untuk menghindari agar hal tersebut tidak terjadi digunakanlah sebuah program khusus proteksi enkripsi data. Ada berbagai algoritma kriptografi dan steganografi yang sekarang telah dan sedang dikembangkan. Salah satunya diantaranya alogoritma kunci simetris ataupun kunci asimetris.

Salah satu metode enkripsi data adalah *Tiny Enctyption Algorithm* (TEA) dan metode penyisipan *Least Significant Bit* (LSB). Kelebihan kriptografi *Tiny Encription Algorithm* dapat memproses waktu secara maksimal dan pemakaian tempat penyimpanan data yang seminimal mungkin. Algoritma kriptografi tersebut akan dikombinasikan dengan teknik penyembunyian data yang disebut dengan steganografi. Teknik steganografi yang akan menjadi pilihan dari penelitian adalah metode *Least Significant Bit (LSB)*. Teknik kriptografi digunakan oleh penulis untuk mengacak pesan yang hendak disispkan pada media dalam arti citra atau gambar sedangkan untuk menyipsikan pesannya penulis menggunakan teknik steganografi agar saat pesan dikirim berupa gambar tidak terlihat mencurigakan karena citra yang tersisipi pesan mirip dengan citra yang tanpa disisipi pesan steganografi.

Algoritma TEA dengan 16 cycles 32 (round) sangat cocok digunakan untuk membangun system keamanan data yang mengandalkan kecepatan dengan proses yang optimal. Proses enkripsi dan dekripsi menggunakan TEA sangat cepat dengan rata-rata 10,6 detik untuk 10.000 kata untuk enkripsi 0,01 detik untuk 10.000 kata untuk dekripsi. Sedangkan citra yang disispi pesan menggunakan steganografi LSB tidak ada perubahan yang mencolok secara visual

Keywords: *TEA, LSB, Keamanan Data, Kriptografi, Steganografi.*

KATA PENGANTAR

Puji syukur terhadap kehadirat Allah SWT atas limpahan rahmat serta hidayah yang diberikan untuk menuntaskan Skripsi dengan lancar.

Skripsi yag berjudul **"Aplikasi Keamanan Data Teks Menggunakan Metode Tiny Encryption Algorithm (TEA) Dan Least Significant Bits (LSB)"** ini dilakukan untuk memenuhi salah satu persyaratan kelulusan di Institut Teknologi Nasional Malang Jurusan Teknologi Industri, Program Studi Teknik Informatika. Namun demikian, sangat disadari bahwa skripsi ini masih jauh dari kesempurnaan yang tak lepas dari kesalahan dan kekurangan, sehingga diharapkan dapat diperbaiki dan disempurnakan dikemudian hari.

Pada kesempatan ini penulis mengucapkan terikasih kepada :

1. Dr. Ir. Lalu Mulyadi, MT, selaku Rektor Institut Teknologi Nasional Malang.
2. Ir. Anang Subardi, MT, selaku Dekan Fakultas Teknologi Insudtri Institut Teknologi Nasional Malang.
3. Joseph Dedy Irawan, ST, MT, selaku Ketua Program Studi Teknik Informatika Institut Teknologi Nasional Malang.
4. Sonny Prasetio, ST, MT, selaku Sekretaris Program Studi Teknik Informatika Intitut Teknologi Nasional Malang.
5. Nurlaily Vendyansyah, ST, selaku Dosen Wali, yang telah memberikan motivasi, bimbingan, masukan, dan saran selama masa perkuliahan.
6. Joseph Dedy Irawan, ST, MT, selaku Pembimbing Utama dan Sandy Nataly Mantja, S.Kom, selaku Pembimbing pendamping, yang dengan sabar telah banyak memberikan bimbingan, motivasi, dan saran dalam proses pembuatan skripsi ini.
7. Dosen Program Studi Teknik Informatika Institut Teknologi Nasional Malang yang telah memberikan pengetahuan selama masa perkuliahan.
8. Seluruh staf dan karyawan Program Studi Teknik Informatika Institut Teknologi Nasional Malang yang telah banyak memberikan bantuan selama perkuliahan.

9. Orang tua yang selalu memberikan dukungan terhadap penulis di Institut Teknologi Nasional Malang, yaitu dukungan moral dan financial selama ini.
10. Teman-teman Teknik Informatika yang telah membantu dalam proses penyelesaian skripsi ini.
11. Serta pihak-pihak lain yang tak dapat disebutkan satu persatu disini yang telah banyak memberikan bantuan demi terselesaikannya skripsi ini.

Akhir kata, semoga skripsi ini banyak memberikan manfaat kepada penulis sendiri khususnya dan pembaca sekalian pada umumnya.

Malang, 25 Januari 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Lalu lintas informasi yang melalui jaringan internet sudah sangat tinggi. Informasi tersebut dapat bersifat umum atau rahasia. Sedangkan komunikasi melalui internet lambat laun kurang begitu aman dikarenakan penggunaan internet yang sangat luas dan dapat diakses siapa saja serta banyak ilmu pengetahuan dalam mempelajari fungsi dari internet baik dalam sisi positif dan negative, oleh karena itu penggunaan jaringan internet untuk pengiriman informasi yang bersifat rahasia kurang begitu aman. Sistem keamanan pengiriman data (komunikasi data yang aman) dipasang untuk mencegah pencurian, kerusakan, dan penyalahgunaaan data yang terkirim melalui jaringan komputer. Dalam praktik , pencurian data berwujud pembacaan oleh pihak yang tidak berwenang biasanya dengan menyadap saluran publik.

Berbagai riset dan survey tentang adanya laporan kejahatan komputer yang terjadi, membuktikan bahwa saat ini tidak ada satupun jaringan komputer yang dapat diasumsikan 100% aman dari serangan *Computer Viruses, Spams, Carding, Email Bomb* atau diterobos langsung *(Penetration)* oleh para *Hackers*. Seorang *Hacker* berpengalaman dapat dengan mudah melakukan kegiatan '*Hacking*' atau memasuki jaringan komputer yang menjadi targetnya. Tidak terhambat kenyataan jaringan komputer tersebut sudah mempunyai sistem pengaman atau belum, ditambah lagi banyak sekali situs-situs dalam internet yang menawarkan informasi tentang cara bagaimana menembus suatu jaringan komputer *(penetrated)* sekaligus mengelabui sistem pengamanannya *(security compromised)*. Informasi "jahat" tersebut tersedia dalam bentuk kumpulan program, dokumentasi atau utiliti. Semakin dominannya akses dan pemanfaatan internet oleh beragam instansi atau institusi di lingkungan pemerintahan, yang mengusik penulis untuk mencoba untuk memberikan gambaran secara umum mengenai aspek pengamanan jaringan komputer serta menumbuhkan '*security awareness*' bagi pengguna jasa Internet.(Hardiayanto.2012). Kejahatan-kejahatan

yang terjadi di dunia maya kian mejadi. Kasus yang beberapa waktu terakhir diungkap jajaran Polda Metro Jaya di antaranya penipuan yang dilakukan warga Nigeria dibantu warga Indonesia terhadap PT AP dan PT BE. Pelaku mencegat percakapan *e-mail* dua perusahaan yang tengah bertransaksi. Pelaku DS, warga Nigeria yang masih buron, memalsukan *e-mail* kedua perusahaan itu. Kedua perusahaan itu merasa tengah berkomunikasi dengan mitranya, padahal dengan tersangka sehingga mereka bersedia saat diminta mentransfer uang senilai miliaran rupiah. Kasus lain adalah ditangkapnya warga Nigeria, AO alias Az, oleh jajaran Resmob Direktorat Reserse Kriminal Umum karena menipu perusahaan di Rusia, Ghips Biruinta, hingga senilai Rp 3,7 miliar. Pelaku mengirim *e-mail* seolah-olah dari Top Glove, perusahaan Malaysia yang bekerja sama dengan perusahaan Ghips, meminta pembayaran atas transaksi sejumlah barang. Pelaku bekerja sama dengan warga Indonesia yang berperan menampung uang hasil penipuan. (Liauw. 2014)

Masalah keamanan dan kerahasiaan data merupakan hal yang sangat penting dalam suatu organisasi maupun pribadi, apalagi jika data tersebut berada dalam suatu jaringan komputer yang terhubung/terkoneksi dengan jaringan lain. Hal tersebut tentu saja akan menimbulkan risiko bilamana informasi yang sensitif dan berharga tersebut diakases oleh orang-orang yang tidak berhak, yang mana jika hal tersebut sampai terjadi, kemungkinan besar akan merugikan bahkan membahayakan orang yang mengirim pesan atau menerima pesan, maupun organisasinya. Informasi yang terkandung didalamnya pun dapat saja berubah sehingga menyebabkan salah penafsiran oleh penerima pesan. Selain itu data yang dibajak tersebut akan memiliki kemungkinan rusak bahkan hilang yang akan menimbulkan kerugian material yang besar

Oleh karena itu, untuk menghindari agar hal tersebut tidak terjadi digunakanlah sebuah program khusus proteksi enkripsi data. Saat ini banyak beredar program khusus proteksi data. Pada umumnya program tersebut tidak hanya menggunakan 1 metode saja, tetapi berbcrapa jenis sehingga kita dapat memilih yang menurut kita paling aman. Ada berbagai algoritma kriptografi dan steganografi yang sekarang telah ada dan sedang dikembangkan. Salah satu

metode enkripsi data adalah *Tiny Enctyption Algorithm* (TEA) dan metode penyisipan *Least Significant Bit* (LSB).

Kelebihan algoritma kriptografi TEA dapat memproses waktu enkripsi dan dekripsi secara maksimal dan pemakaian tempat penyimpanan data yang seminimal mungkin. Algoritma ini merupakan algoritma penyandian *block cipher* dengan menggunakan proses *feistel network* dan fungsi penambahan dan pengurangan sebagai operator pembalik selain XOR. TEA diklaim mempunyai tingkat keamanan yang tinggi jika dibandingkan dengan algoritma kriptografi sejenis. Keunggulan utama dari TEA adalah keringanan prosesnya, operasi-operasi yang digunakan hanya berupa operasi bit biasa, tanpa substitusi, permutasi ataupun operasi matrik, selain itu fungsi dari perhitungan algoritma ini dijalankan di kedua sisi block sehingga data bercampur secara berulang-ulang untuk mencegah penggunaan teknik *exshautive search*. (William. 2009)

Algoritma kriptografi tersebut akan dikombinasikan dengan teknik penyembunyian data yang disebut dengan steganografi. Teknik steganografi yang akan menjadi pilihan dari penelitian adalah metode *LSB*. Teknik kriptografi digunakan oleh penulis untuk mengacak pesan yang hendak disispkan pada media dalam arti citra atau gambar, sedangkan untuk menyipsikan pesannya penulis menggunakan teknik steganografi agar saat pesan dikirim berupa gambar tidak terlihat mencurigakan karena citra yang tersisipi pesan mirip dengan citra yang tanpa disisipi pesan steganografi, yaitu dengan mengganti notasi binari paling kiri bit yang paling tidak berpengaruh pada citra digital dengan bit pesan yang ingin ditanamkan. Dengan teknik ini, pengirim dapat menyembunyikan informasi pada dengan memodifikasi bit yang paling tidak berpengaruh pada citra. Bit yang paling tidak berpengaruh pada citra adalah bit yang bernilai 1 atau 0. Jadi Citra tidak akan berubah meskipun nilai bit citra tersebut telah diubah, tetapi akan mengalami perubahan yang signifikan terhadap ukuran file. (Tezar. 2014)

Berdasarkan latar belakang yang telah diberikan, maka penulis ingin melakukan suatu penelitian yang berjudul "**APLIKASI KEAMANAN DATA TEXT MENGGUNAKAN METODE TEA DAN LSB**".

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah diatas, dapat dirumuskan permasalahan yang akan dibahas adalah :

1. Apa tahap-tahap yang dilakukan untuk merancang aplikasi implementasi TEA dengan LSB ?
2. Bagaimana cara penghitungan metode TEA dan metode LSB ?
3. Bagaimana mengimplementasikan pengenkripsian dan penyembunyian data menggunakan TEA dan LSB?
4. Bagaiman cara kerja dari enkripsi/deskripsi TEA dan steganografi LSB ?
5. Cara penyisipan informasi rahasia agar tidak dapat dibaca oleh pihak yang tidak berkepentingan.

1.3 Tujuan Penelitian

Adapun yang menjadi tujuan penulisan dalam penyusunan proposal skripsi adalah sebagai berikut :

1. Membuat aplikasi penyembunyian informasi text pada citra digital dengan menggunakan metode TEA dan LSB.
2. Mengimplementasikan penggabungan enkripsi dan penyembunyian data menggunakan TEA sebagai enkripsi dan dekripsi pesan sedangkan penyisipan pesan menggunakan LSB.
3. Menganalisa hasil ekripsi pesan yang berupa *plaintext*, dimana *plaintext* keluaran harus sama dengan *plaintext* yang dikirim pada citra digital.
4. Menganalisa hasil dekripsi *plaintext* yang didapat dari citra digital dimana keluaran *plaintext* harus sama dengan pesan yang dikirim.

1.4 Batasan Masalah

Dalam penyusunan proposal skripsi agar menjadi sistematis yang mudah dimengerti, maka akan di terapkan beberapa batasan masalah. Adapun batasan masalah meliputi :

1. Metode enkripsi yang digunakan adalah metode TEA sedangkan metode steganografi yang digunakan adalah metode LSB.
2. Penyembunyian atau penyisipan pesan berupa informasi dilakukan pada citra digital.

3. Panjang *input-bit* teks sebanyak 64-bit tiap blok sedangkan kunci enkripsi dan dekripsi 8 bit sampai 128 bit diketik oleh *user*.
4. Proses enkripsi dan deskripsi dilakukan secara sederhana hanya pada data yang disimpan di dalam *storage*, bukan pada data yang dikirimkan (ditransmisikan) dalam suatu saluran komunikasi
5. Besar informasi yang disembunyikan tergantung dari besar citra digital.
6. Keluaran yang dihasilkan adalah citra dengan format *.bmp / *.png
7. Aplikasi berbasis desktop menggunakan bahasa pemrograman C#
8. Menggunakan *Integrated Development Environments Visual Studio 2013.*

1.5 Metode Penelitian

Dalam penyusunan penelitian skripsi, untuk dapat mencapai keinginan penulis membutuhkan informasi dan metode yang berhubungan dengan tema yang akan di kupas oleh penulis, yaitu aplikasi keamanan data teks menggunakan metode TEA dan LSB. Maka perlu dilakukan dengan langkah-langkah sebagai berikut

1.5.1 Tinjauan Pustaka

Pada tahap tinjauan pustaka, penulis melakukan pembelajaran melalui buku dan jurnal mengenai teknik kriptografi dan teknik steganografi pada citra digital. Pada tahap tinjauan pustaka penulis mencari landasan teori dan metode yang digunakan pada penelitian yaitu metode TEA dan LSB.

1.5.2 Desain Aplikasi

Pada tahapan desain aplikasi, penulis akan mengidentifikasi komponen-komponen sistem yang akan digunakan secara rinci yang meliputi perancangan algoritma matematis sistem, desain *flowchart*, desain blok diagram penentuan *input, process* dan *output*, dan lain-lain yang dapat mendukung perancangan sistem.

1.5.3 Implementasi

Setelah proses desain aplikasi telah dilakukan. Langkah selanjutnya adalah melakukan implementasi pada desain yang telah dibuat. Tahap ini adalah penerapan terahadap rancangan algoritma, *flowchart* dan blok diagram pada

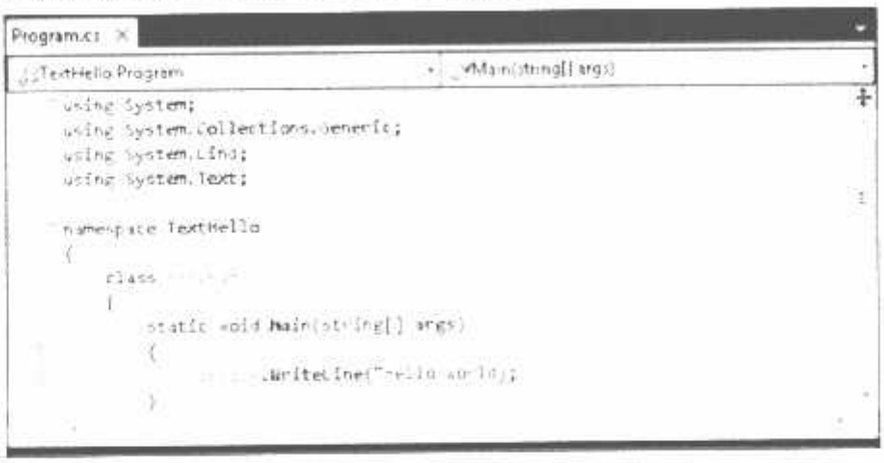
BAB II
LANDASAN TEORI

2.1 Pemrograman C#

Pada tahun 2000 Microsoft meluncurkan bahasa pemrograman baru yang diberi nama *C# Programming Language*. C# dikembangkan oleh Microsoft olch tim yang dipimpin oleh Anders Hejlsberg dan Scott Wiltamuth. C# memiliki kesamaan bahasa dengan C, C++, dan Java, sehingga memudahkan developer yang sudah terbiasa dengan bahasa C untuk menggunakannya, C# mengambil fitur-fitur terbaik dari ketiga bahasa tersebut dan juga menambahkan fitur-fitur baru. C# adalah bahasa pemrograman *Object Oriented* dan memiliki *class library* yang sangat lengkap yang berisi *prebuilt component* sehingga memudahkan programer untuk men-develop program lebih cepat. C# juga distandarkan oleh *Ecma International* pada bulan desember 2002. (Riris.2013)

C# sebagai bahasa pemrograman untuk *framework* .NET memiliki ruang lingkup penggunaan yang sangat luas. Pembuatan program dengan *user interface windows* maupun *console* dapat dilakukan dengan C#, karena *framework* .NET memberikan fasilitas untuk berinteraksi dengan kode yang *unmanaged*. Penggunaan *library* seperti *DirectX8.1* dan *OpenGL* dapat dilakukan. C# juga dapat digunakan untuk pemrograman *website* dan *webservice*. Microsoft sendiri akan terus mengembangkan *framework* .NET dan mengintegrasikan produk-produknya dengan *framework* .NET. Sebagai contoh, *DirectX9* akan memiliki komponen-komponen yang managed untuk menyedot para game developer ke dunia pemrograman .NET. *Windows* dua generasi setelah *Windows* XP yaitu *Windows Blackcomb* diisukan akan menjadi akhir dari era Win32 dengan menjadi *operating system* yang sepenuhnya berpondasikan framework .NET. (Kurniawan,.2014)

Dengan C# dapat dibuat bermacam aplikasi seperti aplikasi *console*, aplikasi *windows form*, aplikasi web, aplikasi web *services*, dan aplikasi untuk *mobile device*. Jadi cukup belajar satu bahasa saja tapi sudah dapat digunakan untuk mengembangkan berbagai macam aplikasi. Dibawah ini merupakan contoh awal dalam pemrograman c#, dapat dilihat pada gambar 2.1



Gambar 2.1 *Hello World pada pemrograman c#*

2.2 ASCII

ASCII adalah singkatan dari *American Standard Code for Information Interchange*. Sesuai dengan namanya, ASCII digunakan untuk pertukaran informasi dan komunikasi data. ASCII merupakan kode angka yang mewakili sebuah karakter. ASCII digunakan karena komputer hanya mengerti angka-angka. ASCII (*American Standard Code For Information Interchange*) merupakan kode *standar* yang digunakan dalam pertukaran informasi pada komputer

Jumlah kode ASCII adalah 255 kode. Kode ASCII 0 - 127 merupakan kode ASCII untuk manipulasi teks, sedangkan kode ASCII 128 - 255 merupakan kode ASCII untuk manipulasi grafik. Kode ASCII sendiri dapat dikelompokkan lagi kedalam beberapa bagian:

1. Kode yang tidak terlihat simbolnya seperti kode 10(*Line Feed*), 13(*Carriage Return*), 8(*Tab*), 32(*Space*)
2. Kode yang terlihat simbolnya seperti abjad (A..Z), numerik (0..9), karakter khusus (~!@#$%^&*()_+?:"{})

3. Kode yang tidak ada di keyboard namun dapat ditampilkan. Kode ini umumnya untuk kode-kode grafik.

Kode ASCII terbagai atas 2 bagian yaitu kode ASCII *standart* dan kode ASCII *extended*. Kode ASCII *standart* adalah kode ASCII yang merepresentasikan angka, huruf serta tombol *standar*, *enter*, *escape*, *backspace* dan *space*. Selain itu juga terdapat karakter-karakter yang tidak terdapat pada *keyboard*, yang dapat diaktifkan dengan melakukan penekanan tombol kombinasi "alt" dan angka yang dimaksud, sebagai contoh tombol kombinasi "alt" dan angka "127" akan menghasilkan karakter grafis, sedangkan kode ASCII *extended* adalah kode ASCII yang akan bertindak sebagai kode perluasan (*extended*) dari kode ASCII yang ada, karena tidak semuanya mampu tertampung dalam kode ASCII standard. Kode ASCII jenis ini lebih banyak bertindak sebagai kode-kode tombol khusus, seperti kode untuk tombol F1 s/d F12. Sebagai contoh adalah kode ASCII extended untuk F12 adalah "123". (Febriyani. 2016)

2.3 *Object Oriented Programming* (OOP)

OOP adalah sebuah sistem yang dibangun berdasarkan metoda berorientasi objek adalah sistem yang komponennya di-enkapsulasi menjadi kelompok data dan fungsi, yang dapat mewarisi atribut dan sifat dari komponen lainnya, serta komponen-komponen tersebut saling berinteraksi satu sama lain. Pada proses pembuatan aplikasi ini yang dilakukan pertama kali yaitu menganalisa sejumlah obyek yang berinteraksi dan berperilaku menurut kebutuhan sistem. Tujuan analisis adalah memahami permasalahan dan merencanakan dengan baik model dunia nyata. Dalam analisi dijelaskan tentanga apa yang harus dilakukan, bukan bagaimana melakukan dan sesuatu masalah harus dipahami sebelum dipecahkan, sehingga memudahkan dalam memahami suatu permasalahan maka diperlukan langkah-langkap selama proses analisis (Raji, Yusril Maulidan, el al . 2014). Hasil analisis pemodelan berorientasi object menurut Erward, terdiri atas tahapan-tahapan:

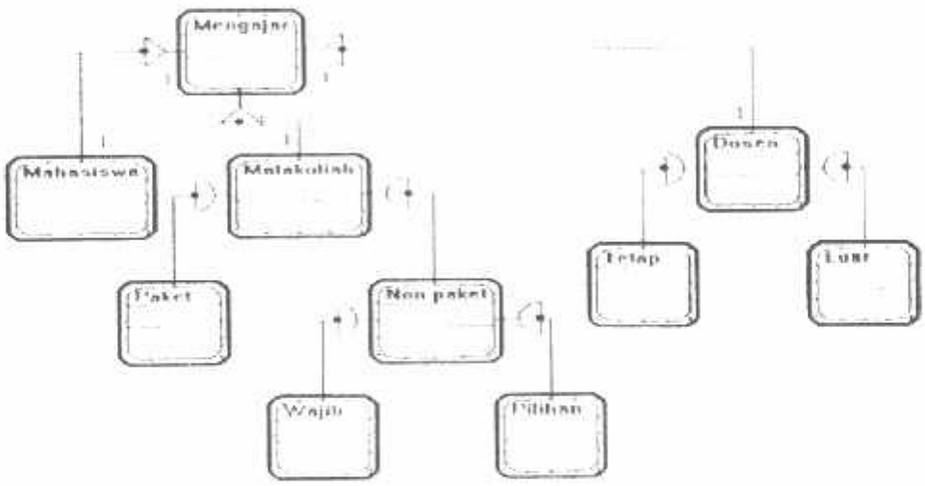
1. Menetukan Obyek dan Kelas

Obyek adalah sebuah abstraksi dari sesuatu yang ada dalam kawasan masalah, menggambarkan kemampuan sistem untuk memuat informasi

tentangnya, berinteraksi dengannya atau keuduanya. Suatu kapsulisasi nilai-nilai atribut serta pelayanan exclusive, sedangkan kelas adalah dekripsi dari grup obyek dengan sifat-sifat (atribut obyek) yang serupa, Kelas mendefinisikan (tema, esensi, dan arti semantic dari suatu kelas) dari struktur dan perilaku obyek-obyek dari bentuk tertentu. (Raji, el al . 2014)

2. Menentukan Struktur

Struktur mengidentifikasikan relasi antar obyek yang membentuk komposisi seperti halnya relasi perwarisan. Terdapat 2 tipe struktur yaitu : *gen-spec* untuk membentuk relasi perwarisan sedangkan struktur *Whole-part* mengindetifikasi relasi komposisi. *Gen-spec* dibangun bila object dipakai oleh semua obyek atau anak. Sedangkan struktur whole-part dibangun bila obyek induk dikompilasi dari berberapa onyek anak (Raji,el al . 2014). Berikut contoh notasi gen-spec dan whole-part dapat dilihat pada gambar 2.2.



Gambar 2.2 *Contoh notasi gambar gen-spec dan whole part*

3. Menentukan Atribut

Atribut dalam pemodelan berorientasi obek terkait dengan atribut obyek dan relasi antar obyek. Atribut dapat memberikan gambaran umum dalam sebuah obyek dan memiliki nilai tertentu. Atribut merupakan data atau bisa juga berupa fungsi-fungsi yang dimiliki oleh kelas tersebut. Atribut diakses melalui notasi bertitik. Atribut-atribut kelas terikat hanya untuk kelas-kelas dimana atribut tersebut didefinisikan. Atribut-atribut data merupakan veriabel variabel yang kita

deklarasikan. Variabel-variabel tersebut dapat digunakan seperti variabel lainnya dan dapat di ubah-ubah nilainya oleh method didalam kelas ataupun di dalam program utama. (Raji , el al . 2014) Contoh atribut dapat dilihat pada gambar 2.3.

Gambar 2.3 *Contoh Atribut pada kelas dosen*

4. Menentukan *Service* atau *Method*

Method adalah suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. *Method* didefinisikan pada *class* akan tetapi dipanggil melalui *object*. (Raji, el al . 2014) Berikut dapat dilihat notasi *method* pada gambar 2.4.

Gambar 2.4 *Notasi Method*

Dalam OOP terdapat berberapa konsep yang sering digunakan dalam membangun aplikasi antara lain :

2.3.1 *Inheritance* (Perwarisan)

Inheritance merupakan pewarisan atribut dan method pada sebuah class yang diperoleh dari class yang telah terdefinisi tersebut. Setiap subclass akan mewarisi state (variabel-variabel) dan behaviour *(method-method)* dari superclass-nya. Subclass kemudian dapat menambahkan state dan behaviour baru yang spesifik dan dapat pula memodifikasi *(override) state* dan *behaviour* yang diturunkan oleh *superclass*-nya.

Subclass atau kelas turunan menyediakan *state/behaviour* yang spesifik yang membedakannya dengan superclass atau kelas induk, hal ini akan memungkinkan programmer C# untuk menggunakan ulang *source code* dari *superclass* yang telah ada. Programmer C# dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut *abstract class*, untuk mendefinisikan

class dengan *behaviour* dan state secara umum. Ada dua *keyword* utama dalam *inheritance*, *super* dan *extends*. *Extends* harus kita tambahkan pada definisi *class* yang menjadi *subclass*. Sedangkan *super* digunakan untuk memanggil konstruktor dari *superclass* atau menjadi variabel yang mengacu pada superclass. (Oktiavia. 2010) Berikut merupakan contoh perwarisan pada pemrograman C# pada gambar 2.5

```
class Quadrilateral
{
    private float width, height;
    protected void CalculateArea(float w, float h)
    {
        width = w;
        height = h;
        Console.WriteLine("area={0}", width * height);
    }
}
class Square : Quadrilateral
{
    public static void Main()
    {
        Square sq = new Square();
        sq.CalculateArea(5, 5);
    }
}
```

Gambar 2.5 *Perwarisan pada C#*

2.3.2 *Polymorphism* (Polimorfisme)

Polimorfisme atau perubahan bentuk merupakan kemampuan dari *reference* untuk mengubah sifat menurut object apa yang dijadikan acuan. Dengan kata lain, kita bisa menggunakan variabel dalam program untuk mengaplikasikan objek untuk memanggil method yang berbeda. Keuntungan dari polimorfisme menyediakan *multiobject* dari *subclasses* yang berbeda untuk diperlakukan sebagai *object* dari *superclass* tunggal, secara otomatis menunjuk *method* yang tepat untuk menggunakannya ke partikular *object* berdasar *subclass* yang termasuk di dalamnya.

Polimorfisme tak lengkap jika tanpa *overloading*. *Overloading* adalah suatu keadaan yakni beberapa method memiliki nama yang sama tetapi memiliki fungsionalitas yang berbeda. Ciri-ciri *overloading* yaitu nama method harus sama, sedangkan parameter harus berbeda. *Overloading* memungkinkan polimorfisme pada kelas super dan sub kelasnya. Konstructor yang terdiri dari dua atau lebih

disebut *overloading constructor* yang termasuk ciri polimorfisme. Berikut contoh polimorfisme pada C# dapat dilihat pada gambar 2.6. (Oktiavia.2010)

Gambar 2.6 *Polimorfisme pada C#*

2.3.3 *Encapsulation* (Enkapsulasi)

Enkapsulasi adalah suatu cara untuk menyembunyikan informasi detail dari suatu class. Dua hal yang mendasar dalam enkapsulasi yakni : *Information hiding* , *Interface to access data.* Anggota *class* dapat diakses baik berupa atribut maupun *method* secara langsung dengan menggunakan objek yang dibuat. Hal ini dikarenakan akses kontrol yang diberikan kepada atribut maupun *method* yang ada di dalam *class* tersebut adalah '*public*'. Informasi dapat disembunyikan dari suatu class sehingga anggota class tersebut tidak dapat diakses dari luar, caranya adalah hanya dengan memberikan akses kontrol '*private*' ketika mendeklarasikan atribut atau method. Proses ini disebut dengan *information hiding.*

Jika telah dilakukan information hiding terhadap suatu atribut pada suatu *class*, lalu bagaimana cara melakukan perubahan terhadap atribut yang disembunyikan tersebut, caranya adalah dengan membuat suatu interface berupa method untuk menginisialisasi atau merubah nilai dari suatu atribut tersebut yang dinamakan dengan *interface to access data.* Pada gambar 2.7 adalah contoh penulisan enkapsulasi pada C#. (Oktiavia.2010)

```
class CheckingAccount : BankAccountProtected
{
    protected override void ApplyPenalties()
    {
        Console.WriteLine("Checking Account Applying Penalties");
    }

    protected override void CalculateFinalInterest()
    {
        Console.WriteLine("Checking Account Calculating Final Interest");
    }

    protected override void DeleteAccountFromDB()
    {
        base.DeleteAccountFromDB();
        Console.WriteLine("Checking Account Deleting Account from DB");
    }
}
```

Gambar 2.7 *Enkapsulasi pada C#*

2.4 IDE Visual Studio Community 2013

Gambar 2.8 *Logo Visual Studio 2013*

Visual Studio 2013 adalah alat untuk para pengembang guna memberi kemudahan dalam merancang aplikasi modern agar dapat terhubung dengan semua *platform microsoft*, mulai dari windows azure dan windows server 2012 hingga windows 8.1 dan office 365.

Visual Studio merupakan salah satu *Integrated Development Environment* (IDE) yang paling banyak digunakan khususnya untuk mengembangkan aplikasi *mobile*, *desktop*, maupun *website*. Terdapat tiga edisi visual studio 2013, yaitu edisi standard yang diberikan secara cuma-cuma, edisi *professional* dengan harga USD 499 (Rp 6 juta) dan edisi *ultimate* dengan harga USD 4.249 (Rp 51 juta). edisi *ultimate* memiliki harga mahal karena biasanya digunakan pada perusahaan-perusahaan besar.

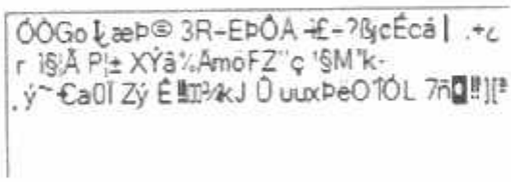
Untuk pengembang individu, visual studio 2013 telah dilengkapi dengan fitur yang membantu produktivitas penggunanya, seperti fitur *peek definition* dan *CodeLens*, dengan berberapa fitur tersebut hal ini dapat menyajikan beragam informasi penting tentang lokasi *source code* yang sedang anda olah dalam editor

visual studio. Fitur lainnya adalah *Browser Link, Office 365 Cloud Business Apps, Performance and Diagnostics Hub, Agile Portfolio Management* dan tentunya *editor XAML* yang telah ditingkatkan.

Microsoft juga mengembangkan visual studio online yang merupakan layanan bagi *developer* dan menjadi bagian dari Microsoft Cloud OS. Alat ini akan memberi pengalaman baru bagi *developer* guna membuat aplikasi yang lebih mudah, cepat, dan tentunya bebas masalah, saat ini microsoft telah memasukkan visual studio online ke dalam bagian MSDN.(Lannueardy. 2013)

2.5 *Cryptography* (Kriptografi)

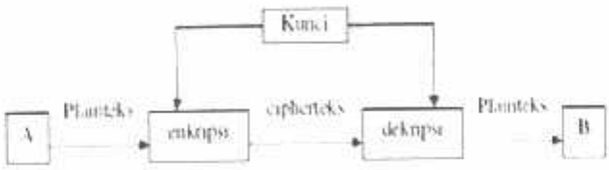
Kriptografi (cryptography) merupakan ilmu dan seni untuk menjaga pesan agar aman. (*Cryptography is the art and science of keeping messages secure*) "Crypto" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan). Para pelaku atau praktisi kriptografi disebut *cryptographers*. Sebuah algoritma kriptografik (*cryptographic algorithm*), disebut *cipher*, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat. Proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut *plaintext*) menjadi pesan yang tersembunyi (disebut *ciphertext*) adalah enkripsi (*encryption*). *Ciphertext* adalah pesan yang sudah tidak dapat dibaca dengan mudah. Menurut ISO 7498-2, terminologi yang lebih tepat digunakan adalah "*encipher*". Proses sebaliknya, untuk mengubah *ciphertext* menjadi *plaintext*, disebut dekripsi (*decryption*). Menurut ISO 7498-2, terminology yang lebih tepat untuk proses ini adalah "*decipher*".(Ariyus. 2013)



Gambar 2.9 *Contoh kriptografi*

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya :

1. Algoritma simetris ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi maupun dekripsi. Algoritma ini sudah ada sejak lebih dari 4000 tahun yang lalu (Sari. 2009). Bila mengirim pesan dengan menggunakan algoritma ini, penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsikan pesan yang terkirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan. TEA termasuk didalam menggunakan algoritma simetris karena membutuhkan kunci untuk mendeskripsikan pesan.



Gambar 2.10 *Konsep dasar algoritma simetris*

2. Algoritma asimetri sering juga disebut dengan algoritma kunci public, dengan arti kata kunci yang digunakan melakukan enkripsi dan dekripsi berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu :

a. Kunci umum (public key), kunci yang boleh semua orang tahu (dipublikasikan).
b. Kunci rahasia (private key), kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci public orang dapat mengenkripsi pesan tetapi tidak bisa mendekripsikannya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsikan pesan tersebut. Algoritma asimetri bisa mengirimkan pesan dengan lebih aman daripada algoritma simetri.

Dalam kriptografi terdapat berberapa istilah yang sering digunakan seperti *plaintext/clear text, Ciphertex, Cipher, Substitution cipher, Transposition cipher, Block Cipher,* Kunci *(key),* Enkripsi (*encryption*), Dekripsi (*decryption*).

1. *Plaintext :* data atau informasi asli bersifat terbuka yang isinya dapat dibaca dan dipahami secara langsung, Menjadi sumber data untuk proses enkripsi

2. *Ciphertext* : data hasil dari proses penekripsian.
3. *Chipper* algoritma untuk mengubah plaintext menjadi ciphertext menggunakan persamaan matematika.
4. *Substitution cipher* adalah algoritma mengubah *plaintext* menjadi *ciphertext* dengan cara mengganti menggunakan persamaan matematika tertentu
5. *Transposition* cipher adalah algoritma mengubah *plaintext* menjadi *ciphertext* dengan cara menggeser menggunakan persamaan matematika tertentu.
6. *Block Cipher* adalah algoritma mengubah *plaintext* menjadi *ciphertext* untuk setiap block data. Jumlah data atau besarnya block adalah tertentu.
7. Kunci (*key*) adalah data atau nilai yang sangat spesifik yang diketahui oleh pengirim dan penerima yang berhak. Digunakan bersama-sama dengan algoritma kriptografi untuk melakukan proses enkripsi dan dekripsi.
8. Enkripsi (*encryption*) adalah proses yang digunakan untuk menyamarkan/menyembunyikan *linetext*. Hasil dari proses enkripsi adalah data sandi (ciphertext).
9. Dekripsi (*decryption*) kebalikan dari proses enkripsi yaitu mengembalikan ciphertext menjadi plaintext.

2.6 TEA

Tiny Encription Algorithm (TEA) merupakan suatu algoritma sandi yang diciptakan oleh David Wheeler dan Roger Needham dari *Computer Loboratory*, Cambridge Univercity, England pada bulan november 1994. Algoritma ini merupakan algoritma penyandian blok *cipher* yang dirancang untuk penggunaan memori yang seminimal mungkin dengan kecepatan proses yang maksimal.

Sistem penyandingan dengan TEA menggunakan proses *feistel network* dengan menambahkan fungsi matematika berupa penambahan dan pengurangan sebagai operator pembalik selain XOR. Proses *feistel network* adalah membagi *plaintext* kedalam beberapa blok dan melakukan pertukaran letak blok dalam setiap *round*, yang akan memberikan efek konsep konfusi dan difusi. Konfusi

adalah mengaburkan hubungan *plaintext* dan *ciphertext* yang menimbulkan kesulitan dalam usaha untuk mencari keteraturan *plaintext* dan ciphertext, sedangkan difusi adalah menyebarkan redudansi *plaintext* dengan menyebarkan masukan ke seluruh *ciphertext*, hal ini dimaksudkan untuk menciptakan pergeseran dua arah (ke kiri dan ke kanan) menyebabkan semua bit kunci dan data bercampur secara berulang-ulang. .

2.6.1 Cara Kerja TEA

1. Pergeseran (*Shift*)

   Blok teks terang pada kedua sisi yang masing masing sebanyak 32-bit akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.

2. Penambahan

   Langkah selanjutnya setelah digeser kekiri dan kekanan, maka Y dan Z yang telah digeser akan ditambahkan dengan kunci k[0]-k[3]. Sedangkan Y dan Z awal akan ditambahkan dengan sum (*delta*).

3. XOR

   Proses selanjutnya setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng-XOR-an dengan rumus untuk satu *round* adalah sebagai berikut:

$$y = y + (((z<<4) + k[0])\hat{}z + sum\hat{}((z>>5) + k[1]))$$

$$z = z + (((y<<4) + k[2])\hat{}y + sum\hat{}((y>>5) + k[3]))$$

   Hasil penyandian dalam satu *cycle* satu blok teks terang 64-bit menjadi 64-bit teks sandi adalah dengan menggabungkan Y dan Z. Untuk penyandian pada *cycle* berikutnya Y dan Z ditukar posisinya, sehingga Y_1 menjadi Z_1 dan Z_1 menjadi Y_1 lalu dilanjutkan proses seperti langkah langkah diatas sampai dengan *16 cycle (32 round)*.

4. Key Schedule

   Algoritma TEA menggunakan *key schedule*-nya sangat sederhana. Yaitu kunci k[0] dan k[1] konstan digunakan untuk *round* ganjil sedangkan kunci k[2] dan k[3] konstan digunakan untuk *round* genap.

5. Enkripsi dan Dekripsi

Proses dekripsi sama halnya seperti pada proses penyandian yang berbasis *feistel cip*her lainnya. Yaitu pada prinsipnya adalah sama pada saat proses enkripsi. Hal yang berbeda adalah penggunaan teks sandi sebagai *input* dan kunci yang digunakan urutannya dibalik. Proses dekripsi semua *round* ganjil menggunakan k[1] terlebih dahulu kemudian k[0], demikian juga dengan semua *round* genap digunakan k[3] terlebih dahulu kemudian k[2]. Rumus untuk enkripsi dekripsi seperti dibawah ini:

Proses Enkripsi:

$L_0 = L_0 + f(R_0 , k[0], k[1], sum)$ $R_0 = R_0 + f(L_0, k[2], k[3], sum)$

Jadi L_0 merupakan hasil penjumlahan dari L_0 ditambahkan dengan f (R_0 , k[0], k[1], sum).Proses enkripsi untuk satu round digunakan rumus:

y = y + (((z<<4) + k[0])^z + sum^((z>>5) + k[1]))

z = z + (((y<<4) + k[2]^y + sum^((y>>5) + k[3]))

Proses Dekripsi

L0 = L0 + f(R0 , k[1], k[0], sum) R0 = R0 + f(L0, k[3], k[2], sum)

Jadi L_0 merupakan hasil penjumlahan dari L_0 ditambahkan dengan f (R_0 , k[0], k[1], sum).Proses dekripsi untuk satu round digunakan rumus:

y = y + (((z<<4) + k[1])^z + sum^((z>>5) + k[0]))

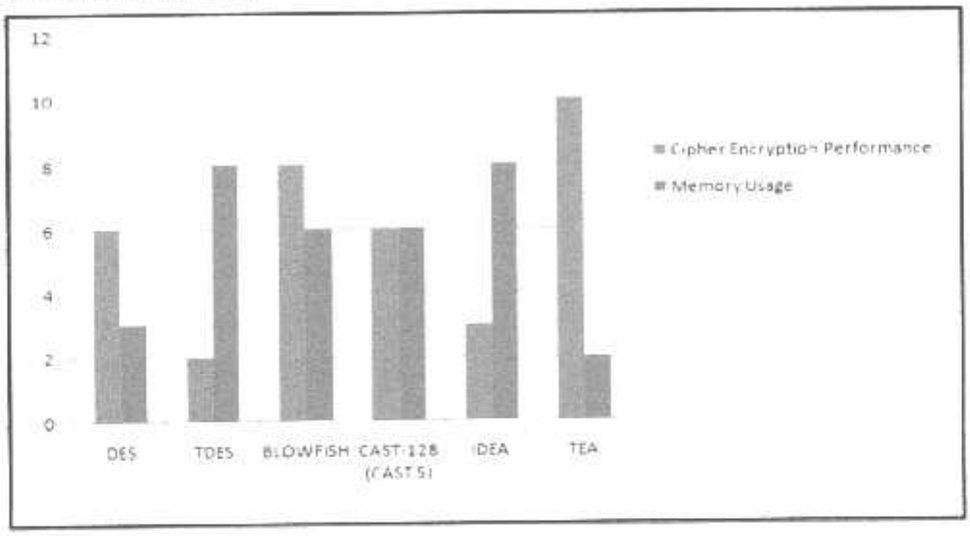
z = z + (((y<<4) + k[3]^y + sum^((y>>5) + k[2]))

Rumus Y diatas menjelaskan bahwa Y merupakan hasil dari Y yang ditambahkan dengan Z yang yang digeser kekiri sebanyak empat kali dengan penambahan kunci k[1], kemudian hasil penjumlahan tadi di XOR kan dengan Z yang dijumlahkan dengan sum(*delta*). Hasil dari peng-XORan dari kedua penjumlahan tadi di XORkan lagi dengan Z yang digeser kekanan sebanyak lima kali dengan penambahan kunci k[0]. Demikian juga dengan rumus Z sama halnya dengan rumus Y, hanya kunci yang digunakan menggunakan kunci k[2] dan k[3].(Sari. 2009)

2.6.2 Skalabilitas Kriptografi TEA

Skalabilitas dari algoritma yang berbeda dianalisis pada dasar penggunaan memori dan kinerja enkripsi (enkripsi dan penjadwalan kunci). Penggunaan memori dapat didefinisikan sebagai jumlah fungsi dilakukan oleh algoritma, lebih

kecil semakin besar penggunaan memori akan efisiensi. Tingkat enkripsi adalah waktu pemrosesan diperlukan oleh algoritma untuk ukuran data tertentu. Tingkat enkripsi tergantung pada kecepatan prosesor, dan kompleksitas algoritma dan lain lain. (Ebrahim. 2013)



Gambar 2.11 *Perbandingan kinerja enkripsi dan penggunaan memori*

Benchmark adalah teknik pengetesan dengan menggunakan suatu nilai standar. Suatu program atau pekerjaan yang melakukan perbandingan kemampuan dari berbagai kerja dari bebcrapa peralatan dengan tujuan untuk meningkatkan kualitas pada produk yang baru.

Berikut adalah tolok ukur kecepatan untuk beberapa yang paling umum digunakan algoritma kriptografi. semua diberi kode di C ++, yang disusun dengan Microsoft Visual C ++ 2005 SP1 dan dengan spesifikasi komputer pada prosesor 2 GHz Intel Core 1.83 di bawah Windows Vista dalam mode 32-bit. x86 / MMX / SSE2.(Dai. 2009)

Tabel 2.1 *Benchmark Algorithm*

Algorithm	**MiB/Second**	**Cycles Per Byte**	**Microseconds to Setup Key and IV**	**Cycles to Setup Key and IV**
DES/CTR	32	54.7	8.372	15320
Blowfish/CTR	58	30.0	62.683	114710

CAST-128/CTR	55	31.9	1.008	1844
IDEA/CTR	35	49.9	0.698	1277
TEA/CTR	27	65.1	0.638	1167

2.7 *Steganography* (Steganografi)

Steganografi adalah seni dan ilmu menulis pesan tersembunyi atau menyembunyikan pesan dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Sebaliknya, kriptografi menyamarkan arti dari suatu pesan, tapi tidak menyembunyikan bahwa ada suatu pesan. Kata "steganografi" berasal dari bahasa Yunani *steganos*, yang artinya "tersembunyi atau terselubung", dan *graphein*, "menulis". Tujuan dari steganografi adalah merahasiakan atau menyembunyikan keberadaan dari sebuah pesan tersembunyi atau sebuah informasi.(Efvy .2013), kebanyakan pesan disembunyikan dengan membuat perubahan tipis terhadap data digital lain yang isinya tidak akan menarik perhatian dari penyerang potensial, sebagai contoh sebuah gambar yang terlihat tidak berbahaya

2.8 LSB

Least significant bit (LSB) adalah bagian dari barisan data biner (basis dua) yang mempunyai nilai paling tidak berarti/paling kecil. Letaknya adalah paling kanan dari barisan bit. Sedangkan most significant bit adalah sebaliknya, yaitu angka yang paling berarti/paling besar dan letaknya disebelah paling kiri. Pada steganografi, metode *Least Significant Bit* adalah metode subtitusi pada posisi bit paling kanan pada notasi binary. Bit tersebut digantikan dengan pecahan informasi yang akan diselipkan. Penyembunyian data dilakukan dengan mengganti bit-bit data yang tidak terlalu berpengaruh di dalam segmen citra dengan bit-bit data rahasia.

Contohnya adalah bilangan biner dari 255 adalah 1111 1111. Bilangan tersebut dapat berarti:

1	1	1	1	1	1	1	1
1×2^7	1×2^6	1×2^5	1×2^4	1×2^3	1×2^2	1×2^1	1×2^0
128	64	32	16	8	4	2	1
MSB							***LSB***

Dari barisan angka 1 di atas, angka 1 paling kanan bernilai 1, dan itu adalah yang paling kecil. Bagian tersebut disebut dengan *least significant bit* (bit yang paling tidak berarti), sedangkan bagian paling kiri bernilai 128 dan disebut dengan *most significant bit* (bit yang paling berarti). Bit yang cocok untuk diganti adalah bit *LSB*, sebab perubahan tersebut hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya.(Fajar. 2013)

2.8.1 Skalabilitas Steganografi LSB

Pada skalabilitas LSB akan dibandingkan dengan MSB dan Hybrid. Perbandingan steganografi menggunakan citra dengan ekstensi JPG dan PNG, parameter untuk membandingkan yaitu: *Messages Texts* , dengan sub parameter *Encoding time (milliseconds), MSE (dB), PSNR (dB).*(Olalekan, Akinola. 2015)

Tabel 2.2 Perbandingan LSB, MSB dan Hybrid

No	Image	Dimesion	File Size	Algorithm	Messages Text (31,160 Bytes) 30,4 KB		
					Encoding (ms)	MSE (dB)	PSNR (dB)
1	Rose.JPG	560 x 448	96.3 KB	LSB	160	0.114	57.54
				MSB	152	1836	15.49
				Hybrid	127	884	18.66

Peak Signal to Noise Ratio (PSNR) merupakan perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR biasanya diukur dalam satuan decibel (dB). PSNR digunakan untuk mengetahui perbandingan kualitas citra cover (asli) sebelum dan sesudah disisipkan pesan. Untuk menentukan PSNR, terlebih dahulu harus ditentukan nilai

MSE (Mean Square Error). MSE adalah nilai error kuadrat rata-rata antara citra asli dengan citra manipulasi (dalam kasus steganografi : MSE adalah nilai error kuadrat rata-rata antara citra asli (cover-image) dengan citra hasil penyisipan (stego-image).

PSNR sering dinyatakan dalam skala logaritmik dalam decibel (dB). Nilai PSNR jatuh dibawah 30 dB mengindikasikan kualitas yang relative rendah, dimana distorsi yang dikarenakan penyisipan terlihat jelas. Akan tetapi kualitas stego-image yang tinggi berada pada nilai 40dB dan diatasnya (Solichin.2015)

2.9 Citra Digital

Citra digital dapat direpresentasikan sebagai matriks gambar dua dimensi yang bisa ditampilkan pada layar komputer sebagai himpunan/ diskrit nilai digital yang disebut pixel/ picture elements. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi.

Citra digital adalah citra f(x,y) dimana dilakukan diskritisasi koordinat sampling/ spasial dan diskritisasi tingkat kwantisasi (kabuan/ kecemerlangannya).Citra digital merupakan fungsi intensitas cahaya f(x,y), dimana harga x dan harga y adalah koordinat spasial. Harga fungsi tersebut di setiap titik (x,y) merupakan tingkat kecemerlangan citra pada titik tersebut.

Citra digital merupakan suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar/ *pixel/ piksel/ pels/ picture element*) menyatakan tingkat keabuan pada titik tersebut.Matrik yang dinyatakan Citra digital yaitu dengan matriks berukuran N (baris/tinggi) x M (kolom/lebar).

N = jumlah baris $0 = y = N - 1$.

M = jumlah kolom $0 = x = M - 1$.

L = maksimal warna intensitas $0 = f(x,y) = L - 1$.

(gray level/ derajat keabuan) (Bertayla. 2005)

Kompresi citra itu adalah aplikasi kompresi data yang dilakukan terhadap citra digital dengan tujuan untuk mengurangi redudansidari data-data yang terdapat dalam citra sehingga dapat disimpan atau ditransmisikan secara efisien. Kompresi algoritma data digital dapat diklasifikasikan dalam 2 kategori.

1. *Loosy* : Algoritma kompresi *lossy* adalah lawan dari algoritma kompresi *loseless*, dimana citra hasil dekompresi tidak identik atau sama dengan citra asli yang dilakukan kompresi.
2. *Loseless* : Algoritma kompresi *loseless* adalah cara dimana sumber identitas citra dapat dibangun kembali seperti awal dari data kompresi citra original. (Pamungkas, 2012)

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Permasalahan

Dengan mengidentifikasi masalah yang ada dalam sebuah sistem, akan membantu dalam menganalisis persyaratan sistem yang akan dikembangkan sehingga tujuan dari sistem tersebut tercapai. Masalah utama yang akan dikembangkan oleh penulis adalah bagaimana mengamankan data atau informasi yang sangat rahasia sehingga tidak dapat diketahui atau diambil oleh pihak yang tidak berhak.

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan proses identifikasi dan evaluasi permasalahan-permasalahan yang ada, sehingga sistem yang dibangun sesuai dengan keluaran yang diharapkan. Metode TEA dan Metode LSB akan diimplementasikan untuk memberikan pengamanan pada pesan dari pengirim ke pada penerima agar pesan yang dikirim aman. Untuk mempermudah menganalisis sebuah sistem dibutuhkan dua jenis kebutuhan. Kebutuhan fungsional adalah kebutuhan yang berisi proses proses apa saja yang akan nantinya dilakukan oleh sistem. Sedangkan kebutuhan non fungsional adalah kebutuhan yang menitik beratkan pada properti perilaku yang dimiliki oleh sistem

3.2.1 Kebutuhan Fungsional

Analisis fungsional dibutuhkan untuk mengetahui hal-hal yang dapat dilakukan oleh sistem. Berikut merupakan persyaratan yang harus dimiliki oleh sistem antara lain :

1. Aplikasi dapat merubah *plaintext* menjadi *chipertext* dengan metode TEA serta dapat disisipkan pada citra 2D menggunakan metode LSB .
2. Aplikasi dapat merubah file document dengan format *.txt menjadi pesan (*plaintext*).
3. Aplikasi dapat membatasi kunci pesan sebesar 8 bit sampai 128 bit atau minimal 1 karakter sedangkan maksimal 16 karakter.

4. Aplikasi harus dapat mengekstrak pesan yang telah tersisip pada citra 2D dan memunculkannya pada aplikasi.
5. Aplikasi harus dapat merubah *chipertext* dari pesan kedalam *plaintext* agar dapat dimengerti oleh penerima dengan memasukkan kunci (*key*) yang sama.
6. Aplikasi dapat menampilkan tiap proses yang terjadi pada enkripsi / dekripsi pesan serta proses penyisipan pesan / pengektrakan pesan.
7. Hasil dari penggabungan proses enkripsi dan penyembunyian data disimpan kedalam format *(*.bmp / *.png)*.
8. Hasil pendekripsian pesan disimpan kedalam format (**.txt*).

3.2.2 Kebutuhan Non Fungsional

Persyaratan fungsional meliputi performa, mudah untuk dipelajari dan digunakan, hemat biaya, dokumentasi, manajemen kualitas, dan kontrol :

1. Performa

   Sistem yang dibangun harus dapat menunjukkan hasil proses enkripsi maupun dekripsi / penyisipan pesan maupun pengekstrakan pesan / hasil proses kombinasi antara TEA dan LSB .
2. Mudah digunakan

   Sistem yang dibangun harus sederhana, terdapat pengertian dari algoritma yang digunakan serta bantuan berupa navigator penggunaan aplikasi agar mudah dioperasikan oleh pengguna (*user*).
3. Hemat Biaya

   Sistem atau perangkat lunak yang dibangun tidak memerlukan perangkat tambahan ataupun perangkat pendukung lainnya yang dapat mengeluarkan biaya.
4. Dokumentasi

   Sistem yang dibangun harus dapat menampilkan hasil pemrosesan yang dilakukan serta menyimpannya kedalam media / *storage* dan juga Sistem atau perangkat lunak yang dibangun dapat menunjukkan waktu pemrosesan data dan dapat melakukan penyimpanan dari hasil proses penggabungan tersebut.

5. Manajemen Kualitas

   Sistem atau perangkat lunak yang dibangun harus memiliki kualitas yang baik yaitu *cover* citra harus mampu menampung besarnya ukuran data yang disisipkan.

6. Kontrol

   Sistem yang dibangun memiliki kontrol berupa *enable* dan *disable*, yaitu pada saat sistem harus terlebih dahulu memasukkan pesan atau *plaintext* serta kunci sehingga jika belum terpenuhi maka tombol enkripsi, tombol steganografi, tombol simpan gambar masih dalam keadaan *disable*. Sedangkan untuk proses dekripsi dan pengekstrakan pesan terlebih dahulu memasukkan citra stego, jika belum terpenuhi maka button ekstrak dan dekripsi dalam kondisi *disable*.

3.3 Blok Diagram

Diagram blok adalah suatu pernyataan gambar yang ringkas, dari gabungan sebab dan akibat antara masukkan dan keluaran dari suatu system.

3.3.1 Blok Diagram Sistem

Gambar 3.1 *Blok diagram sistem*

Pada blok diagram sistem yang akan dibuat. Pada gambar 3.1 adalah gambaran dari sistem yang akan dibuat. Pada sekenario tersebur terdapat 2 pelaku yakni pengirim pesan dan penerima pesan. Pengirim pesan apabila akan mengirimkan pesan rahasia kepada penerima pesan akan menggunakan citra yang telah disisipi oleh pesan yang terenkripsi. Kemudian pesan tersebut dikirim melalui internet atau perantara lainnya. Kemudian penerima pesan akan menerima pesan berupa citra yang telah disisipi pesan. Untuk membaca pesan tersebut maka

diperlukan kunci kriptografi untuk melakukan dekripsi informasi pada pesan yang disisipkan. Oleh karena itu dalam pengiriman kunci rahasia menggunakan cara yang berbeda melalui protokol lain atau media melalui internet, karena apabila kunci tersebut sampai bocor maka pesan tersebut akan dibaca oleh orang yang tidak berkepentingan.

3.3.2 Blok Diagram Enkripsi dan Enkoding

Gambar 3.2 *Blok diagram pengirim (enkripsi dan enkoding)*

Plaintext merupakan istilah untuk pesan yang akan dikirim atau pesan murni, pesan yang akan dikirim sebelumnya akan di enkripsi terlebih dahulu. Enkripsi yang digunakan adalah enkripsi TEA yang kemudian membutuhkan kunci sebesar 8 – 128 bit atau minimal 1 karakter maksimal 16 karakter. Setelah selesai maka akan menghasilkan *chipertext* atau isitilah pesan yang telah diacak.

Chipertext kemudian disisipkan kedalam citra 2D. Teknik dalam penyisipan pesan tersebut menggunakan teknik LSB . Selah pesan disisipkan maka pengirim akan menerima citra stego yang akan dikirimkan kepada penerima.

Berikut adalah penjelasan langkah-langkah dalam gambar blok diagram 3.2 pembuatan aplikasi steganografi dengan kriptografi pada proses enkripsi data :

1. Pesan Yang Akan Dikirim (*Plaintext*)

   Ini adalah sebuah proses input dimana pengguna akan memasukan informasi yang akan enkripsi kemudian disisipkan pada citra. Ini adalah sebuah informasi berbasis teks. Informasi tersebut akan dienkripsi oleh aplikasi.

2. Kunci Enkripsi Pesan

   Kunci enrkipsi juga dimasukkan oleh pengguna untuk dijadikan kunci enkripsi pengguna. Pada aplikasi ini menggunakan kriptografi simetris. Sehingga kunci enkripsi juga dijadikan sebagai kunci untuk mendekripsi informasi yang telah dienkripsi.

3. Enkripsi TEA

   Pada proses ini aplikasi akan melakukan enkripsi atau pengacakan dengan algoritma TEApada informasi yang telah dimasukkan oleh pengguna menggunakan kunci yang juga telah dimasukkan oleh pengguna.

4. Chipertext

   Chipertext adalah sebuah informasi yang telah diacak menggunakan teknik enkripsi. Chipertext tidak akan bisa dibaca oleh pencuri dengar kecuali sang pencuri dengan tahu kunci dari enkripsi tersebut.

5. Cover Image / Citra Steganografi

   Ini adalah citra awal yang akan kita sisipi dengan chipertext. Citra akan modifikasi sedemikian rupa sehingga informasi chipertext tidak dapat dibaca.

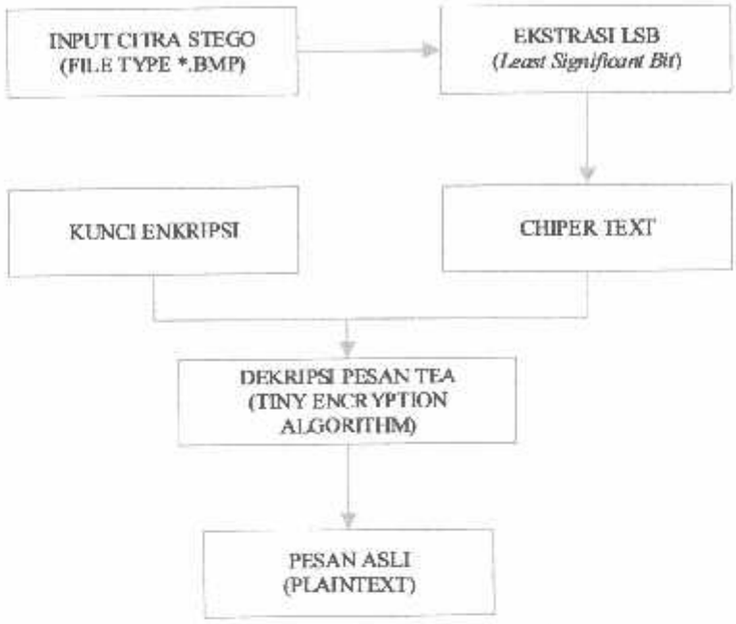
6. *Embbeding* LSB

   Proses penyisipan pesan ke dalam text, pesan pesan tersebut akan diletakkan pada akhir file dari citra yang dipilih.

7. Citra Steganografi

   Ini adalah hasil citra yang telah dilakukan steganografi dengan informasi dari chipertext. Citra dapat dibuka kembali dan dibaca informasi yang ada didalamnya hanya dengan menggunakan kunci steganografi.

3.3.3 Blok Diagram Penerima



Gambar 3.3 *Blok diagram penerima*

Apabila ingin membaca pesan yang tersisip di dalam citra, pertama harus memasukkan kedalam aplikasi berupa citra stego dengan ekstenis **.bmp / *.png*. Citra stego tersebut akan diekstrak menggunakan algoritma *LSB*. Hasil dari ekstrak stego akan mengeluarkan *chipertext*.

Chipertext tersebut akan didekripsi agak dapat dipahami dan dibaca oleh penerima dengan syarat harus memesaukkan kunci enkripsi yang diberikan oleh pengirim dengan benar dan tepat, Setelah selesai terdekripsi maka *chipertext* akan berubah menjadi *plaintext* atau pesan asli. Pada gambar 3.3 blok diagram dekripsi adalah proses untuk mengambil informasi yang telah disimpan pada citra yang telah dilakukan proses steganografi. Berikut adalah penjelasan tiap bloknya :

1. Input Citra Stego

   Pada blok citra steganografi adalah *input* citra yang telah diproses enkripsi dan steganografi. Citra ini akan diambil informasi rahasia yang tersimpan.

2. Ekstrasi LSB

   Proses ini adalah untuk mengambil *LSB* dari citra steganografi. Nilai *LSB* tersebut kemudian disusun per 8 *bit*. *Bit* yang telah disusun itu kemudian diubah menjadi desimal dan disimbolkan dengan karakter.

3. Chipertext

   Chipertext adalah hasil pengambilan informasi steganografi pada citra. *Chipertext* berbentuk susunan karakter yang masih acak yang belum dapat dibaca informasinya.

4. Kunci Enkripsi

   Kunci enkripsi ini digunakan untuk membuka dan mengekstrak informasi pada *chipertext*. Kunci enkripsi adalah kunci yang digunakan pada proses enkripsi sebelumnya.

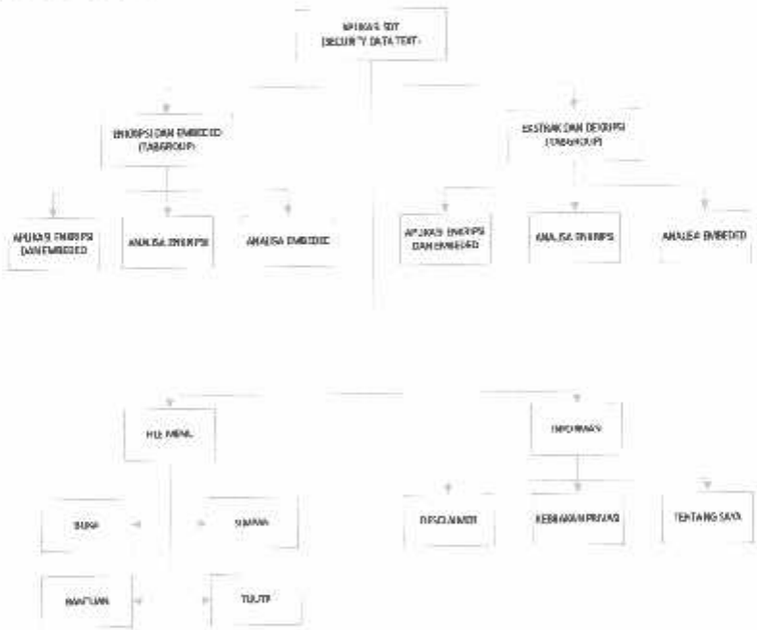
5. Dekripsi TEA

   Dekripsi adalah proses untuk mengubah *chipertext* ke sebuah informasi yang dapat dibaca menggunakan kunci enkripsi tersebut. Proses ini membalikan proses enkripsi untuk melakukan proses dekripsi.

6. Pesan Asli

   Informasi *plaintext* adalah informasi yang dihasilkan dari proses dan *input*. Informasi ini adalah tujuan dari proses dekripsi pesan.

3.4 Struktur Menu



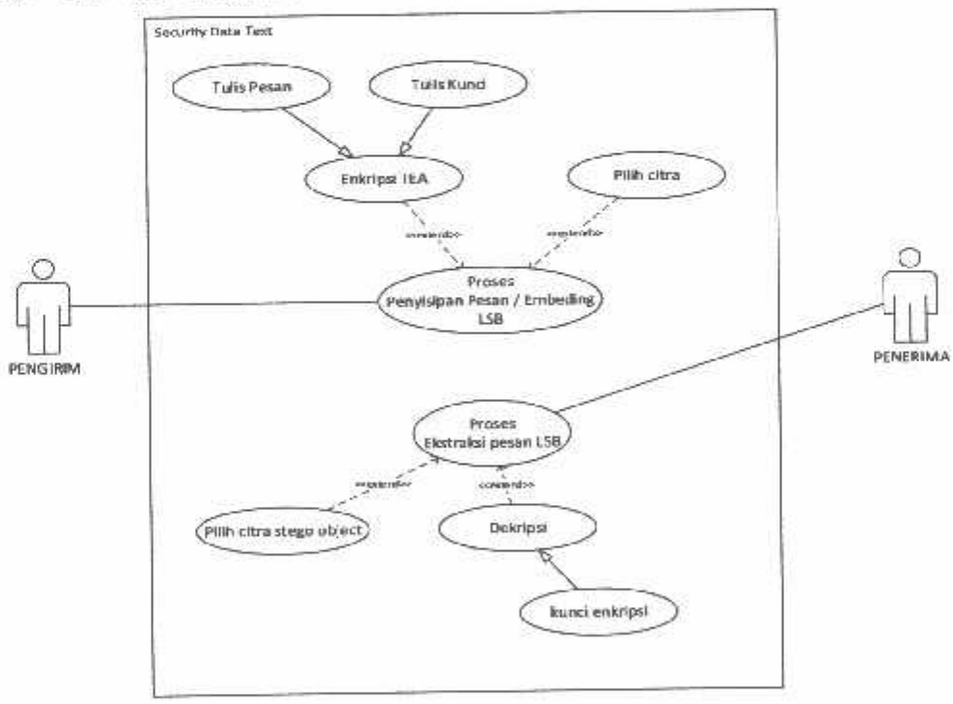
Gambar 3.4 *Struktur menu aplikasi*

Aplikasi yang hendak dibagun memiliki berberapa fasilitas menu yang akan memudahkan user berinteraksi dengan sistem, Pertama untuk pengirim pesan user

akan dihadapkan pilihan menu enkripsi dan penyisipan. kemudian pada tampilan menu enkripsi dan penyisipan pesan terdapat sub menu yakni aplikasi enkripsi dan enkoding. Tampilan enkripsi dan enkoding memiliki fungsi user dapat mengekripsi pesan dan kemudian menyisipkan hasil enkripsian kedalam citra, selanjutnya terdapat menu analisa enkripsi dan analisa enkoding, fungsi dari analisa ini adalah untuk melihat secara detail proses yang sedang terjadi sehingga user dapat mempelajari algoritma yang dibuat oleh penulis

Kedua untuk penerima pesan maka akan dihadapkan oleh berberapa pilihan menu ekstrak dan dekripsi. kemudian pada tampilan menu enkripsi dan penyisipan pesan terdapat sub menu yakni aplikasi denkripsi dan ekraksi pesan. Tampilan ekstrak dan dekripsi memiliki fungsi user dapat mengekstrak pesan yang diberikan pengirim sehingga penerima dapat membaca text yang dikirim oleh pengirim. Selanjutnya tedapat menu analisa dekripsi dan analisa dekoding pesan, fungsi dari analisa ini adalah untuk melihat secara detail proses yang baik proses dekripsi chipertext maupun dekoding pesan dari citra.

3.5 Use Case Diagram



Gambar 3.5 *Proses Uses Case Diagram*

User menginputkan *plaintext*, *plaintext* yang di enkripsi dengan menggunakan algoritma TEA schingga menghasilkan *ciphertext*, kemudian user memilih citra serta menggunakan LSB dalam penyisipan pesan, sehingga hasil citra terdapat *ciphertext* yang dinamakan *citra-stego*. Kemudian pengirim pesan memberitahukan kunci untuk mendekripsi pesan tersebut.

Dalam pengungkapan *citra-stego*, dilakukan pengekstrakan menggunakan LSB sehingga dihasilkan kembali *ciphertext* dan citra gambar yang asli.*chipertext* tersebut akan di dekripsi oleh penerima dengan kunci yang telah disepakati. Untuk mendekripsikan kembali *ciphertext* digunakan algoritma TEA.

3.5.1 Use Case Pengirim

Proses *uses case* pengirim dapat dilhat secara detail pada Tabel 3.1

Tabel 3.1 Use Case Diagram Form "Pengirim"

Nama	**Pengirim**
Actor	*User* Pengirim
Trigger	*User* memilih menu strip enkripsi dan enkoding pesan
Precondition	*User* melakukan aplikasi pengaman data
Post Condition	*User* dapat melihat hasil penyisipan data yang berupa gambar *stego-object*
Success Scenario	1. *User* membuka aplikasi keamanan data. 2. *User* memilih menu strip "Pengirim". 3. Sistem menampilkan *Form* Pengirim. 4. *User* terlebih dahulu menginputkan *Key* kemudian *plaintext*. 5. *User* menekan *button "enkripsi"* untuk mengenkripsikan dan menampilkan *ciphertext*. 6. *User* menekan *button* "cari" untuk menginputkan citra asli sebagai *cover* dalam penyisipan pesan. 7. *User* mengeksekusi dengan menekan *button* "Sisip" untuk melakukan penyisipan pesan.

	8. Sistem menampilkan file *stego-object.* 9. *User* menekan *button* "simpan" untuk menyimpah file *stego-object.*

3.5.2 Use Case Diagram Penerima

Proses *uses case* pengirim dapat dilhat secara detail pada Tabel 3.2

Tabel 3.2 Use Case Diagram Form "Penerima"

Nama	**Pengirim**
Actor	*User* Penerima
Trigger	*User* memilih menu strip denkripsi dan dekoding pesan
Precondition	*User* melakukan aplikasi pengaman data
Post Condition	*User* dapat melihat hasil *stego-object* yang berupa citra asli dan *ciphertext* dan mendekripsikan *ciphertext* menjadi *plaintext.*
Success Scenario	*1. User* membuka aplikasi keamanan data. *2. User* memilih menu strip "Penerima". 3. Sistem menampilkan *Form* Penerima. *4. User* menekan *button* "cari" untuk mengambil *file stegoobject.* *5. User* mengeksekusi dengan menekan *button* "ekstrak" untuk mengeluarkan data yang telah disisipkan. 6. Sistem menampilkan citra asli dan *ciphertext* dari *stegoobject.* *7. User* terlebih dahulu menginputkan *Key* sebelum mendekripsikan . *8. User* mengeksekusi dengan menekan *button* "dekripsi" untuk mendekripsikan *ciphertext* menjadi *plaintext.*

3.6 Flowchart

Flowchart adalah representasi grafik dari langkah-langkah yang harus diikuti dalam menyelesaikan suatu permasalahan yang terdiri atas sekumpulan

simbol, dimana masing-masing simbol merepresantasikan suatu kegiatan tertentu. *Flowchart* diawali dengan penerimaan input, pemrosesan input, dan diakhiri dengan penampilan output. *Flowchart* melukiskan suatu aliran kegiatan dari awal hingga akhir mengenai suatu langkah-langkah dalam penyelesaian suatu masalah. Masalah tersebut bisa bermacam-macam, mulai dari masalah yang sederhana sampai masalah yang kompleks. Masalah yang dihadapai tentunya masalah pemrograman.

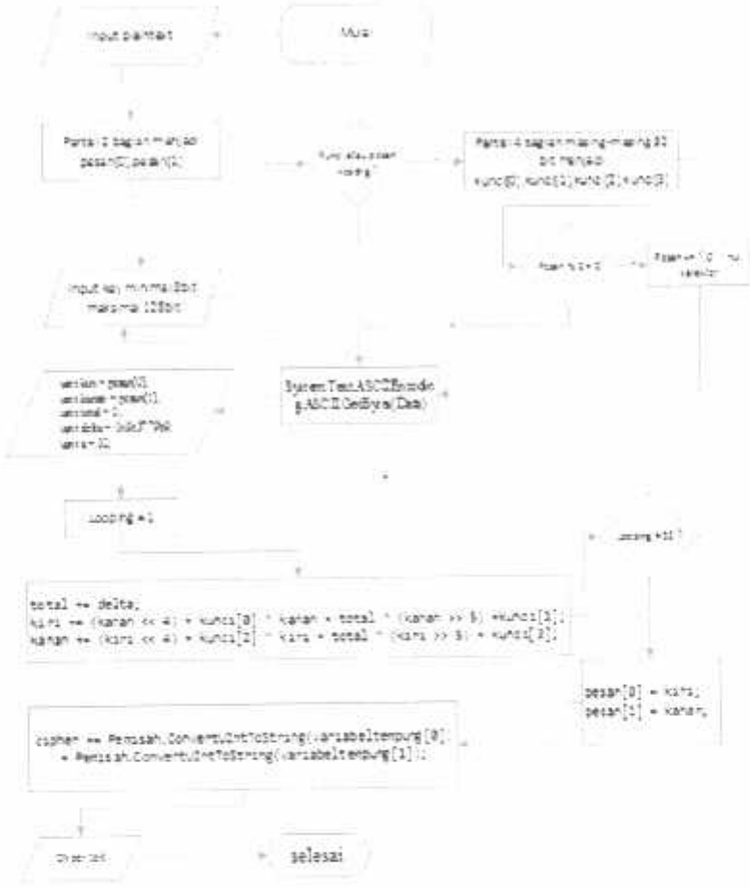
Sistem ini terdiri dari beberapa proses, yaitu proses enkripsi dan dekripsi dengan menggunakan algoritma TEA untuk mengacak pesan, dan proses enkoding dan decoding dalam penyembunyian data enkripsi menggunakan LSB .

Gambar 3.6 *Diagram Alir Aplikasi*

Gambar 3.6 adalah diagram alir yang menjelaskan proses berjalannya program, berikut adalah penjelasan dari diagram alir :

1. Pengguna menjalankan program
2. Pengguna memasukkan menu.
3. Jika pengguna memilih menu enkripsi maka,
   a. Pengguna masukkan informasi yang akan disimpan pada citra
   b. Pengguna masukkan kunci enkripsi
   c. Proses enkripsi TEA informasi dengan kunci enkripsi
   d. Keluaran *chipertext*
   e. Memilih citra cover untuk disisipi pesan rahasia
   f. Proses penyisipan *chipertext* pada citra menggunakan *LSB*
   g. Keluaran citra steganografi
   h. Simpan citra steganografi dengan format **.bmp / *.png*
   i. Keluar dari program
4. Jika tidak, jika pengguna memilih menu dekripsi maka,
   a. Pengguna masukkan citra steganografi
   b. Pengguna masukkan kunci dekripsi
   c. Dekoding *LSB* pada citra steganografi
   d. Keluaran *Chipertext* dari proses dekoding.
   e. Dekripsi TEA *Chipertext* dengan kunci dekripsi
   f. Keluaran informasi yang disembunyikan
   g. Keluar dari program
5. Jika tidak, jika pengguna memilih menu bantuan maka
   a. Menu bantuan enkripsi dan encoding.
   b. Menu bantuan dekripsi dan decoding.
6. Keluar dari program

3.6.1 Diagram Alir Enkripsi



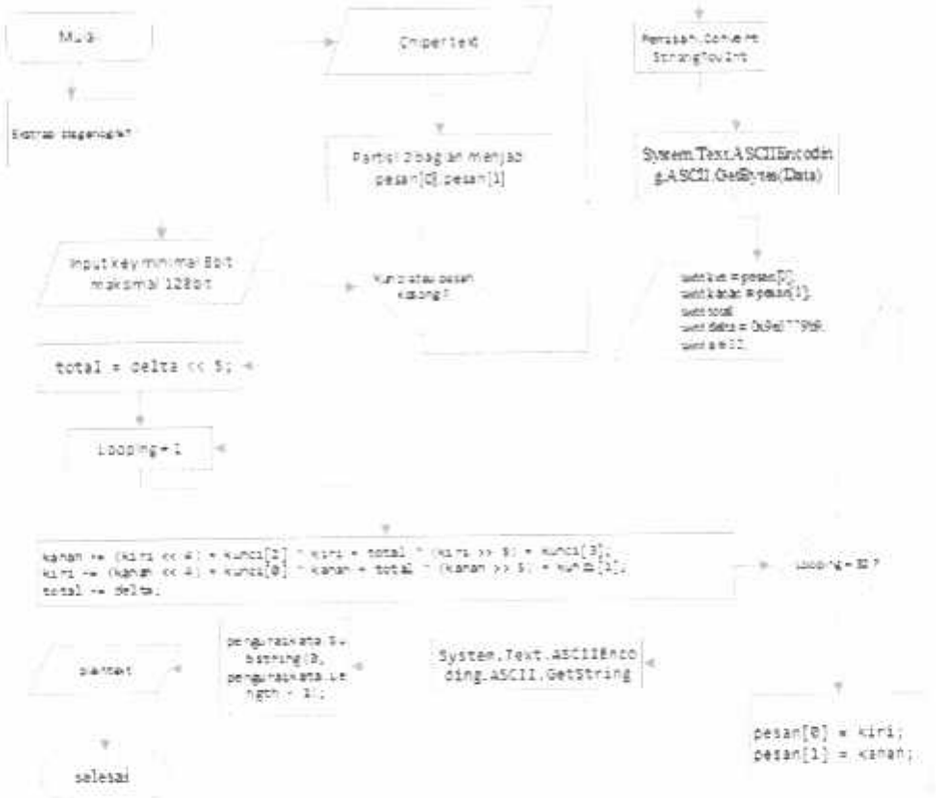
Gambar 3.7 *Diagaram alir enkripsi*

Gambar 3.7 adalah diagram alir yang menjelaskan proses berjalannya program, berikut adalah penjelasan dari diagram alir :

1. Pengguna memasukkan informasi berupa text dan kunci yang akan didekripsi
2. Informasi pesan akan dibagi menjadi 2 blok bagian sepanjang 64 bit. 32 bit kiri dan 32 bit kanan. Dan panjang kunci 1 sampai 16 karakter
3. Blok teks terang pada kedua sisi yang masing masing sebanyak 32-bit akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.
4. Langkah selanjutnya setelah digeser kekiri dan kekanan, maka Y dan Z yang telah digeser akan ditambahkan dengan kunci k[0]-k[3]. Sedangkan Y dan Z awal akan ditambahkan dengan sum (delta).

5. Proses selanjutnya setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng-XOR-an dengan
6. Keluaran berupa *chipertext.*

3.6.2 Diagram Alir Dekripsi



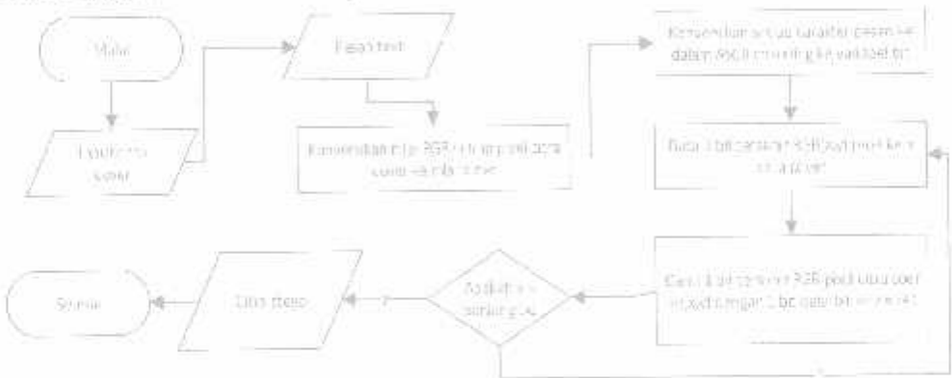
Gambar 3.8 *Diagram alir dekripsi*

Gambar 3.8 adalah gambar dari diagram alir proses dekripsi data menggunakan metode TEA . Berikut adalah penjelasan dari diagram alir :

1. Pengguna memasukkan *chipertext* yang terekstrak dari citra stego.
2. Pengguna memasukkan kunci kriptografi.
2. Informasi pesan akan dibagi menjadi 2 blok bagian sepanjang 64 bit. 32 bit kiri dan 32 bit kanan. Dan panjang kunci 1 sampai 16 karakter
3. Blok pesan teks pada kedua sisi yang masing masing sebanyak 32-bit akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.

4. Langkah selanjutnya setelah digeser kekiri dan kekanan, maka Y dan Z yang telah digeser akan ditambahkan dengan kunci k[0]-k[3]. Sedangkan Y dan Z awal akan ditambahkan dengan sum (delta).
5. Proses selanjutnya setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng-XOR-an dengan
6. Hal yang berbeda dari enkripsi adalah penggunaan dari kata teks sandi sebagai *input* dan kunci yang digunakan urutannya dibalik. Proses dekripsi semua *round* ganjil menggunakan k[1] terlebih dahulu kemudian k[0], demikian juga dengan semua *round* genap digunakan k[3] terlebih dahulu kemudian k[2].
7. Muncul keluaran pesan asli

3.6.3 Diagram Alir Enkoding



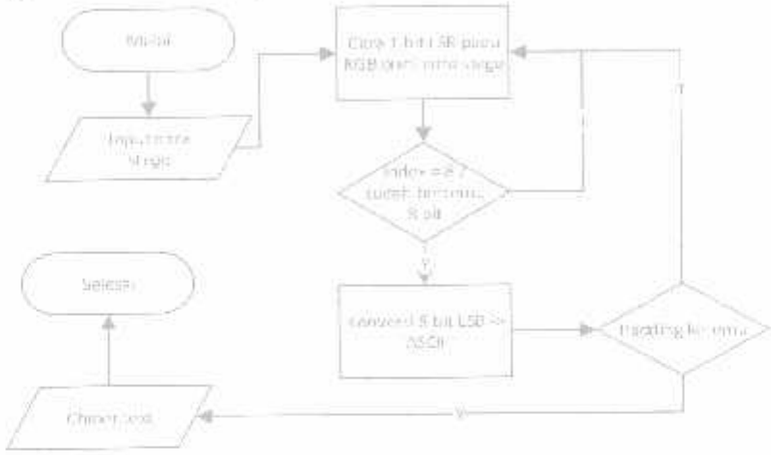
Gambar 3.9 *Diagram alir proses enkoding*

Gambar 3.9 adalah gambar dari diagram alir proses enkoding pesan ke citra digital menggunakan metode LSB . Berikut adalah penjelasan dari diagram alir :

1. Pengguna memasukkan citra digital yang akan disisipi dengan pesan.
2. Pengguna akan memasukkan pesan yang akan disisipkan pada citra digital.
3. Proses enkoding pesan yang dimasukkan pengguna menggunakan standar karakter pengguna. yaitu proses merepresentasikan pesan yang berupa teks atau deretan karakter menjadi deretan angka.
4. Konversi hasil representasi angka enkoding pesan menjadi bilangan biner.

5. Susun semua pesan yang telah dikonversi menjadi biner menjadi satu paket *bit.*
6. Sisipkan tiap *bit* dari paket *bit* yang telah dibuat pada citra digital, hingga deret *bit* terakhir
7. Simpan citra baru yang telah disisipi dengan pesan tiap intensitasnya.
8. Citra steganografi yang telah disisipi pesan.

3.6.4 Diagram Alir Dekoding

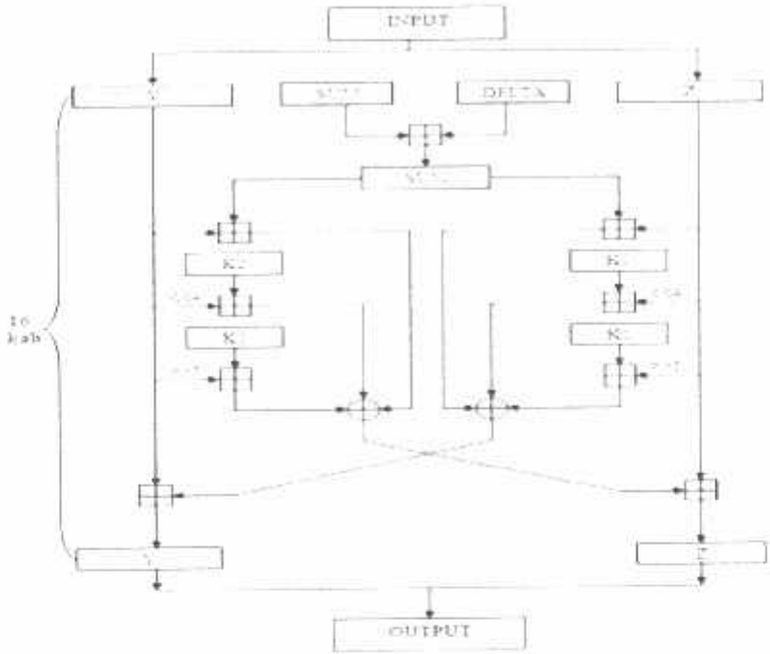


Gambar 3.10 *Diagram alir proses dekoding*

Gambar 3.10 adalah gambar dari diagram alir proses dekoding pesan yang ada dalam citra digital untuk melihat pesan pada citra digital. Berikut adalah penjelasan dari diagram alir :

1. Pengguna memasukkan citra steganografi yang telah dibuat sebelumnya.
2. Proses baca tiap intensitas *RGB* pada tiap piksel citra digital.
3. Proses modulus tiap intensitas *RGB* untuk menghasilkan nilai *LSB* pada tiap intensitas.
4. Menyusun nilai *LSB* menjadi per n bit. N adalah jumlah bit yang digunakan untuk menyimpan tiap karakter. Nilai n tergantung pada total bit terbanyak dari jumlah standar karakter pengguna.
5. Padding ketemu.
6. Konversikan tiap n bit menjadi bilangan desimal.
7. Dekoding hasil konversi yang berupa nilai untuk mendapatkan susunan karakter pesan.
8. Hasil berupa pesan yang telah disisipkan pada citra digital.

3.7 Analisa Kriptografi



Gambar 3.11 *Diagram Feistel*

TEA merupakan Algoritma yang menggunakan kunci simetris, dimana plaintext yang berjumlah 64 bit dibagi menjadi 2 blok, yaitu blok kiri dan blok kanan, setiap blok berjumlah 32 bit. Kemudian memiliki proses pembentukan kunci algoritma TEA yaitu kunci yang berjumlah 128 bit dibagi menjadi 4 blok, dimana setiap blok berjumlah 32 bit. TEA berbasiskan dengan jaringan fiestel dan memiliki 32 kali putaran (1 kali putaran memiliki 2 round).. Langkah-langkah penyandian dengan algoritma TEA dalam satu cycle (dua round) :

1. Pergeseran *(shift)*

   Blok teks terang pada kedua sisi yang masing masing sebanyak 32-bit akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.

2. Penambahan

   Langkah selanjutnya setelah digeser kekiri dan kekanan, maka Y dan Z yang telah digeser akan ditambahkan dengan kunci k[0]-k[3]. Sedangkan Y dan Z awal akan ditambahkan dengan sum (delta).

3. Peng-XOR-an

   Proses selanjutnya setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng-XOR-an.

4. Hasil penyandian dalam satu *cycle* satu blok teks terang 64-bit menjadi 64-bit teks sandi adalah dengan menggabungkan Y dan Z. Untuk penyandian pada *cycle* berikutnya Y dan Z ditukar posisinya, sehingga Y1 menjadi Z1 dan Z1 menjadi Y1 lalu dilanjutkan proses seperti langkah langkah diatas sampai dengan *16 cycle (32 round).*

5. Proses dekripsi sama halnya seperti pada proses penyandian yang berbasis *feistel cip*her lainnya. Yaitu pada prinsipnya adalah sama pada saat proses enkripsi. Hal yang berbeda adalah penggunaan teks sandi sebagai *input* dan kunci yang digunakan urutannya dibalik. Proses dekripsi semua *round* ganjil menggunakan k[1] terlebih dahulu kemudian k[0], demikian juga dengan semua *round* genap digunakan k[3] terlebih dahulu kemudian k[2]. Rumus untuk enkripsi dan dekripsi seperti dibawah ini :

Rumus Enkripsi:

SUM = n * Delta

Ln = L(n-1)+((R(n-1)Shl 4)+K[0]) XOR (R(n-1)+SUM) XOR ((R(n-1) Shr 5)+K[1]))

Rn = R(n-1) + ((Ln Shl 4)+K[2]) XOR (Ln+SUM) XOR ((Ln Shr 5)+K[3])

Rumus Dekripsi:

Rn = R(n-1) – ((L(n-1) Shl 4)+k[2]) XOR (L(n-1)+SUM) XOR ((L(n-1) Shr 5)+k[3])

Ln = L(n-1) – ((Rn Shl 4)+k[0]) XOR (Rn+SUM) XOR ((Rn Shr 5)+k[1]))

Di mana : Delta = (√5 -1)231 = (2654435770)D= ($9e3779b9)H

n = *Round* ke..

3.7.1 Proses Enkripsi

Berikut adalah informasi yang akan dienkripsi dan kunci enkripsi yang berupa teks.

Teks : ITNMLG
Kunci :1218236

Plaintext dibagi menjadi 2 blok kedalam blok A dan blok B :

A = ITNM

B = LG__

Kemudian, Key = 128 bit, dibagi menjadi 4 blok masing-masing 32 bit :

Kunci[0] = 1218

Kunci[1] = 236_

Kunci[2] = ____

Kunci[3] = ____

Plaintext diubah menjadi kode ASCII, kemudian diubah ke biner ITNMLG__

I = 073 = 01001001

T = 084 = 01010100

N = 078 = 01001110

M = 077 = 01001101

L = 076 = 01001100

G = 071 = 01000111

= null = 00000000

= null = 00000000

Key diubah menjadi kode ASCII, kemudian diubah ke biner

1 = 049 = 00110001

2 = 050 = 00110010

1 = 049 = 00110001

8 = 056 = 00111000

2 = 050 = 00110010

3 = 051 = 00110011

6 = 054 = 00110110

= null = 00000000

= null = 00000000

= null = 00000000

= null = 00000000

= null = 00000000

= null = 00000000

= null = 00000000

Biner plaintext digabungkan dan akan terbentuk seperti berikut :

A(r) = 01001001 01010100 01001110 01001101

B(l) = 01001100 01000111 00000000 00000000

Binary Key digabungkan dan akan terbentuk seperti berikut

Kunci[0] = 00110001 00110010 00110001 00111000

Kunci[1] = 00110010 00110011 00110110 00000000

Kunci[2] = 00000000 00000000 00000000 00000000

Kunci[3] = 00000000 00000000 00000000 00000000

Plaintext mengalami pergeseran bit ke kiri sebanyak 4 bit kemudian 5 bit pergeseran ke kanan:

A(r) = 01001001 01010100 01001110 01001101

A(kiri) =11010101011101100011011001100000

A(kanan) = 00000100100101010100010011100010

A(kiri) ditambah dengan kunci[0]

A(kiri) =11010101011101100011011001100000

Kunci[0] = 00110001001100100011000100111100

AL(Kunci[0]) = 01110000111000010110000101010111

A(kanan) ditambah dengan kunci[1]

A(kanan) = 00000100100101010100010011100010

Kunci[1] = 00110010001100110011011000000000

AL(Kunci[1]) = 01101000111110111011000011100010

Plaintext awal A(r) ditambah dengan bilangan delta. Secara konstan nilai delta, diubah ke nilai Hexadesimal = 9E3779B9 dan di ubah ke biner dengan membagi kedalam 4 bagian:

9E = 10011110

37 = 00110111

79 = 01111001

B9 = 10111001

A(r) = 01001001010101000100111001001101

Delta = 10011110001101110111100110111001

Ar(Delta) = 01110011110001011110010000000110

Kemudian Ar(Delta) di XOR kan dengan *plaintext* AT.(kunci[0]) = 11010001001010101011010101111

Kemudian hasil tersebut di XOR kan dengan *plaintext* AR(kunci[1]) = 10111001110100010001111011011101

Untuk Plaintext B(l) mengalami pergeseran bit ke kiri sebanyak 4 bit kemudian 5 bit pergeseran ke kanan.

B(r) = 01001001010101000100111001001101

B(kiri) =11101110010000000000000000000000

B(kanan) = 10011000100011100000000000

B(kiri) ditambah dengan Kunci[2] :

B(kiri) = 11101110010000000000000000000000

Kunci[2] = 00000000000000000000000000000000

BL(Kunci[2]) = 01110111001000000000000000000000

B(kanan) ditambah dengan Kunci[3] :

B(kanan) = 10011000100011100000000000

Kunci[3] = 00000000000000000000000000000000

BR(Kunci[3]) = 01001100010001110000000000

Plaintext awal B(l) ditambah dengan bilangan delta.

B(l) = 01001100010001110000000000000000

Delta = 10011110001101110111100110111001

B(i)Delta = 01110101001111110011110011011001

Kemudian B(i)Delta di XOR kan dengan BL(Kunci[2]) = 10111000010001100001100100011l

Kemudian hasil tersebut di XOR kan dengan BR(kunci[3]) = 10110001110011101111001000111

Hasil Akhir = Ar(Kunci[1]) + B(i) =

10111001110100010001110110111101 + 01001100010001110000000000000000 = 01010100100101111100011011011101

Hasil Akhir = Br(Kunci[3]) + A(r) =

01001100010001110000000000 + 01001001010101000100111001001101= 01001011101101101000011001001101

Telah didapat proses hasil enkripsi dari TEAdengan 2 round(1 cycle) dengan hasil A1(11) digabung denga B1(R1), namun proses enkripsi akan berakhir sampai 32 round (16 cycle), dimana setiap penyelesainnya setiap 2 round digunakan hasil cipher sebelumnya untuk melanjukan 2 round berikutnya. Hasil ciphertext dari studi kasus tersebut adalah :

¡e8Â®¡¶7D5¤nö#+æQCQ½«QCQ½«1-âm3

3.7.2 Proses Dekripsi

Konsep pendekripsian dari TEAmemiliki proses yang sama dengan enkripsi, perbedaannya pada penjumlahan Kunci yang terjadi setelah pergeseran bit. Pada saat dekripsi A(r) dilakukan pergeseran bit ke kiri sebanyak 4 bit maka ditambah Kunci[1]. Kemudian setelah pergeseran bit ke kanan sebanyak 5 bit ditambah Kunci [0]. Untuk dekripsi A(l) dilakukan pergeseran bit ke kiri sebanyak 4 bit maka ditambah Kunci[3]. Kemudian setelah pergeseran bit ke kanan sebanyak 5 bit ditambah Kunci [2].

3.8 Analisa Steganografi

Metode steganografi yang digunakan adalah metode LSB . Yaitu dengan mengubah bit paling kanan atau bit yang paling tidak berpengaruh. Pada citra digital, kita akan mengubah *LSB* dari tiap intensitas *RGB* dari tiap piksel. Pesan yang disisipkan pada proses *LSB* harus dilakukan proses dekoding untuk mengubah pesan informasi yang berupa teks menjadi deretan biner. Yang selanjutnya pesan berupa deretan biner tersebut akan disisipkan tiap biatnya pada *LSB* intensitas citra digital. Untuk melakukan proses dekoding dan enkoding karakter, akan digunakan karakter standar pengguna.

3.8.1 Analisa Enkoding

Sebuah citra digital memiliki piksel dan intensitas. Piksel merupakan lokasi intensitas berada yang dapat digambarkan seperti matriks. Sedangkan intensitas adalah warna penyusun dari citra digital tersebut. Representasi citra digital dengan tipe warna *RGB 8 bit* dapat digambarkan sebagai berikut.

Gambar 3.12 *Citra Digital Map Indonesia*

R=255	R=23	R=52	R=66	R=10	R=212
G=128	G=211	G=100	G=245	G=255	G=125
B=33	B=34	B=67	B=101	B=100	B=56
R=55	R=23	R=123	R=90	R=47	R=100
G=66	G=111	G=41	G=255	G=89	G=23
B=200	B=55	B=250	B=100	B=245	B=56
R=100	R=100	R=211	R=199	R=25	R=56
G=45	G=51	G=255	G=20	G=251	G=99
B=56	B=23	B=65	B=50	B=222	B=255

Untuk langkah awal proses enkoding steganografi dengan menggunakan metode *LSB* adalah dengan melakukan enkoding pesan menjadi angka desimal berdasarkan standar karakter pengguna.

Pesan : ITN

Enkoding pesan tersebut menjadi berupa angka berdasarkan standar karakter pengguna yang telah didefinisikan. Angka desimal tersebut kemudian dikonversikan menjadi biner berbasis 8 *bit* sesuai perhitungan standar *bit* tiap

karakter berdasarkan jumlah standar karakter pengguna. Kemudian biner pesan tersebut dijadikan satu deretan biner utama.

Pesan : ITN

I	T	N
73	84	78
01001001	01010100	01001110

Deret Pesan Biner : 010010010101010001001110

Setelah informasi telah diubah menjadi deret pesan biner. Langkah selanjutnya adalah menyiapkan citra digital yang akan disisipi dengan deret pesan biner tersebut.

R=66	R=10	R=212
G=245	G=255	G=125
B=101	B=100	B=56
R=90	R=47	R=100
G=255	G=89	G=23
B=100	B=245	B=56
R=199	R=25	R=56
G=20	G=251	G=99
B=50	B=222	B=255

Langkah selanjutnya adalah dengan mengkonversikan nilai desimal intensitas menjadi nilai biner berbasis 8 bit.

R=01000010	R=00001010	R=11010100
G=11110101	G=11111111	G=01111101
B=01100101	B=01100100	B=00111000
R=01011010	R=00101111	R=01100100
G=11111111	G=01011001	G=00010111
B=01100100	B=11110101	B=00111000
R=11000111	R=00011001	R=00111000
G=00010100	G=11111011	G=01100011

B=00110010	B=11011110	B=11111111

Proses selanjutnya adalah dengan menyisipkan deret pesan biner yang dibuat sebelumnya ke dalam tiap tiap *LSB* intensitas citra digital. Nilai *LSB* pada citra digital terletak pada posisi paling kanan bit.

Deret Pesan Biner: 010 010 010 101 010 001 001 110

Biner Citra Awal:

R=01000010	R=00001010	R=11010100
G=11110101	G=11111111	G=01111101
B=01100101	B=01100100	B=00111000
R=01011010	R=00101111	R=01100100
G=11111111	G=01011001	G=00010111
B=01100100	B=11110101	B=00111000
R=11000111	R=00011001	R=00111000
G=00010100	G=11111011	G=01100011
B=00110010	B=11011110	B=11111111

Biner Citra Setelah Disisipi :

R=01000010	R=00001011	R=11010100
G=11110100	G=11111110	G=01111101
B=01100100	B=01100100	B=00111000
R=01011010	R=00101110	R=01100100
G=11111111	G=01011001	G=00010111
B=01100101	B=11110100	B=00111000
R=11000111	R=00011001	R=00111000
G=00010100	G=11111010	G=01100011
B=00110010	B=11011111	B=11111110

Setelah menyisipkan deretan biner pesan ke tiap-tiap intensitas *LSB*. Langkah selajutnya adalah dengan melakukan konversi biner citra digital yang

BAB IV
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan menjelaskan fungsionalitas dan pengujian steganografi dan kripotografi pada aplikasi yang telah dibuat.

4.1 Perangkat Yang Digunakan

Perangkat keras yang digunakan adalah lingkungan aplikasi saat pembuatan dan pengujian. Hal ini dibutuhkan sebagai parameter pengujian aplikasi. Perangkat yang digunakan adalah sebuah *notebook* atau laptop. Berikut adalah spesifikasi perangkat yang digunakan :

1. Perangkat Keras :
   a. AMD-A8-4500M APU Raderon HD Graphic (4CPUs) ~ 1.9Hz
   b. 4096MB Random Access Memory
   c. Built in monitor 14"
   d. Built in keyboard
   e. Mouse
2. Perangkat Lunak :
   a. Sistem Operasi Windows 7 Ultimate 64-bit Service Pack 1
   b. Microsoft Visual Studio 2013 Comunity

4.2 Implementasi

Implementasi adalah tahap penjelasan tentang fungsionalitas dan alur dari aplikasi yang telah dibuat.

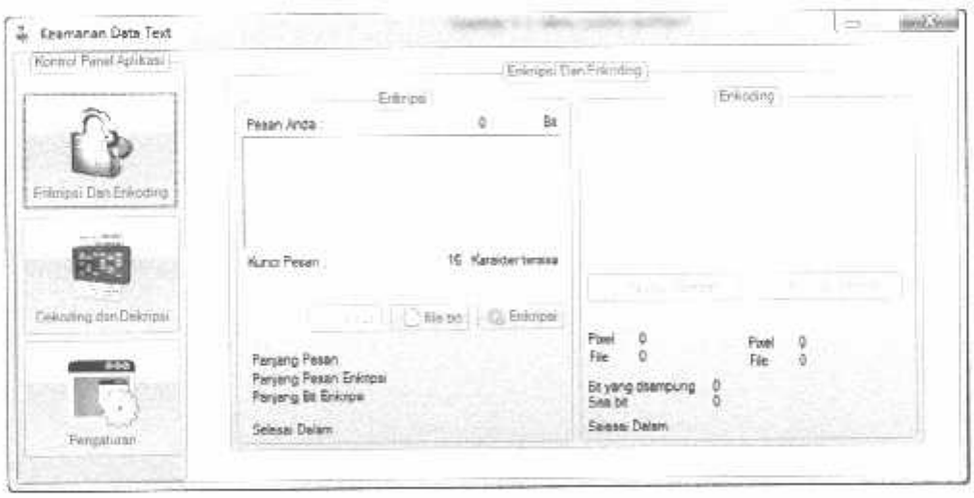
4.2.1 Menu Utama

Gambar 4.1 merupakan tampilan pertama yang dihadapkan kepada pengguna. Pengguna pertama kali akan dihadapkan dengan icon serta nama dari aplikasi KDT-Indonesia kemudian pada panel kiri terdapat fasilitas menu yang dapat digunakan oleh pengguna dalam pengoperasian aplikasi KDT-Indonesia yang terdiri dari enkripsi dan enkoding, dekoding dan dekripsi, dan pengaturan. Untuk lebih jelas perhatikan gambar 4.1

Gambar 4.1 *Menu utama Aplikasi*

4.2.2 Form Enkripsi dan Enkoding

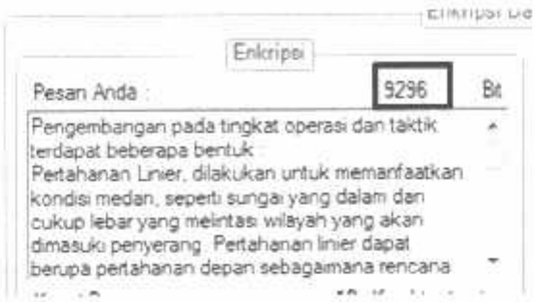
Setelah pengguna dihadapkan dengan menu utama, maka selanjutnya pengguna dapat memanfaatkan fasilitas enkripsi dan enkoding guna mengamankan pesan menggunakan metode TEA yang akan dikirim yang kemudian pesan tersebut disisipkan kedalam sebuah citra digital 2D dengan menggunakan metode LSB . Untuk lebih jelasnya silahkan perhatikan gambar 4.2



Gambar 4.2 *Fitur enkripsi dan enkoding*

Pada gambar 4.2 terdapat berberapa fitur yang dapat digunakan oleh pengguna dalam mengamankan pesan dan menyisipkan pesan tersebut. Fitur tersebut adalah menuliskan pesan. Pengguna juga dapat mengupload pesan kedalam aplikasi dengan format **.txt*. Untuk batasan pesan tidak ada, jadi

pengguna dapat mengirim pesan sepanjang yang dimau.Untuk lebih jelasnya dapat dilihat seperti gambar 4.3

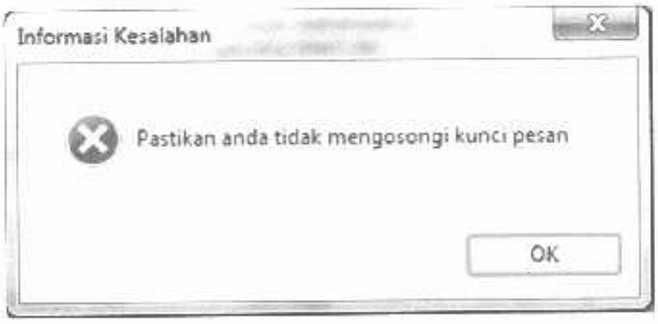


Gambar 4.3 *Penulisan pesan yang hendak dikirim*

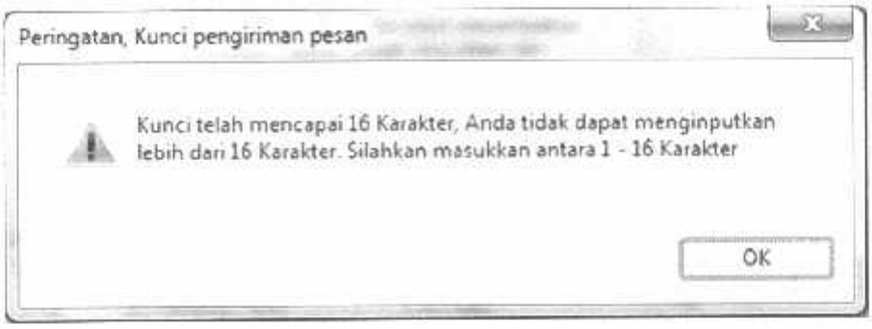
Pada gambar 4.3 terdapat kota hitam yang menunjukkan besar semua bit dari pesan yang hendak dikirim, dalam aplikasi ini menggunakan standart ASCII jadi tiap karakter terdiri dari 8 bit 0-127. Untuk kode ASCII silahkan lihat pada lampiran. Setelah pengguna menuliskan pesan, maka pengguna harus menuliskan kunci enkripsi. Kunci enkripsi ini terdiri dari 8 – 128 bit, jadi minimal 1 karakter dan maksimal 16 karakter. Untuk lebih jelas silahkan perhatikan 4.4

Gambar 4.4 *Penulisan kunci enkripsi*

Pada aplikasi ini apabila pengguna memasukkan kunci lebih dari 16 dan tidak memasukkan kunci makan aplikasi akan memunculkan peringatan. Untuk lebih jelasnya pada gambar 4.5 adalah peringatan saat tidak memasukkan kunci dan 4.6 adalah peringatan saat kunci melebihi 16 karakter

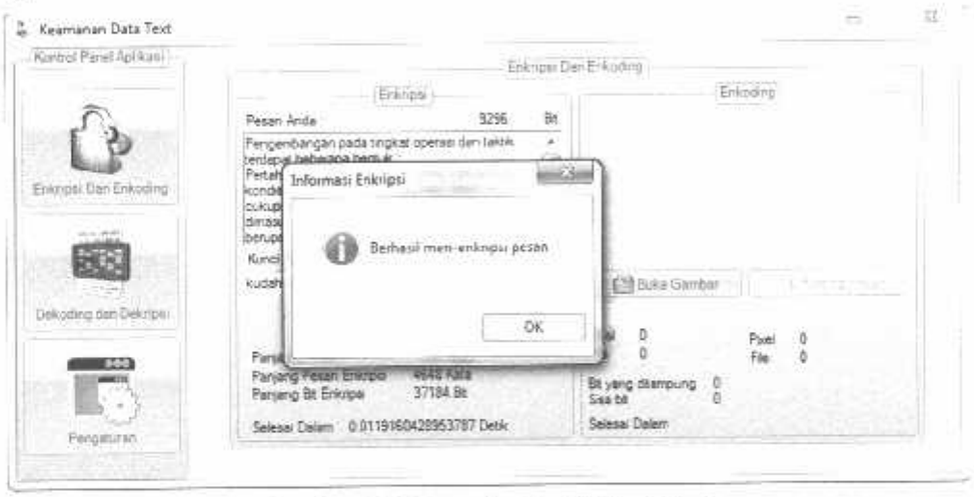


Gambar 4.5 *Pesan kesalahan saat kunci kosong*



Gambar 4.6 *Pesan peringatan saat karakter kunci lebih dari 16*

Setelah pengguna menuliskan pesan yang akan dikirim serta kuncinya maka pengguna dapat langsung enkripsi pesan tersebut menggunakan metode TEA dengan cara menekan tombol enkripsi. Maka secara otomatis pesan akan dienkripsi. Kemudian aplikasi akan memberikan balasan sebuah *Messeges Box*. Berupa "Berhasil meng-enkripsi pesan". Untuk lebih jelasnya perhatikan gambar 4.7



Gambar 4.7 *Pesan berhasil di enkripsi*

Setelah aplikasi memberikan balasa, maka pesan telah berhasil di acak menggunakan metode TEA. Secara umum prosesnya bahwa pesan akan bagi menjadi blok 64 bit, tiap sub blok 32 bit. Kiri dan kanan. Kemudian input kunci enkripsi 128bit. Partisi blok kunci menjadi 4 sub blok kunci kunci[0] kunci[1] kunci[2] kunci[3] masing-masih 32 bit. Geser sesuai protokol tea dan looping sebanya 32 kali. Selama proscs looping sistem akan menggabugkan text sehingga menjadi deretan *chipertext*

Setelah itu apabila pengguna ingin melihat hasil dari enkripsi tersebut maka tekan tombol lihat. Untuk lebih jelasnya perhatikan gambar 4.8

Gambar 4.8 *Hasil dari pesan yang terenkripsi*

Aplikasi dapat juga memberikan informasi semua proses enkripsi mulai dari panjang pesan yang dikirim, berapa panjang pesan yang telah terenkripsi dan juga seberapa cepat aplikasi dapat mengenkripsi pesan dalam satuan detik. Untuk lebih jelas perhatikan gambar 4.9

Gambar 4.9 *Informasi dari hasil enkripsi*

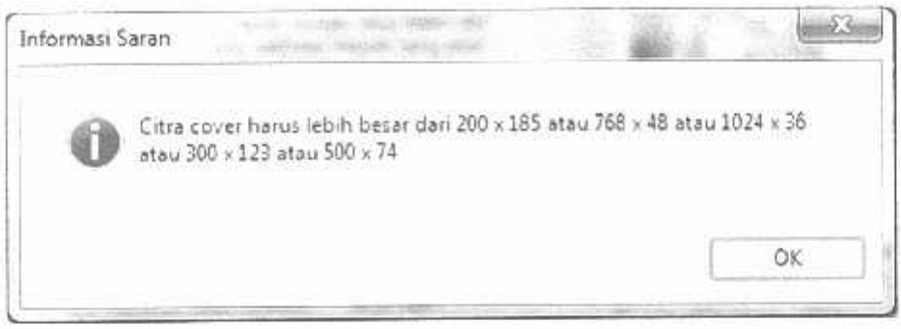
Apabila pengguna telah selesai mengenkripsi pesan, maka selanjutnya pengguna akan dihadapkan dengan fitur enkoding, fitur ini dapat menyisipkan pesan kedalam citra digital dengan menggunakan metode LSB. Selanjutnya pengguna memilih citra cover sebagai media dalam penyisipan pesan. Dalam aplikasi KDT-Indonesia format citra yang dapat digunakan adalah **.jpg, *jpeg, *bmp, *.tif, *png*. Untuk lebih jelasnya dapat dilihat pada gambar 4.10

Gambar 4.10 *Proses pemilihan citra cover*

Setelah pengguna selesai citra yang hendak dijadikan cover pesan dalam penyisipan pesan maka aplikasi akan menampilkan informasi mengenai citra tersebut seperti berpapa panjang dan lebar dari citra tersebut, besar ukuran file citra tersebut, serta berapa banyak bit pesan yang dapat ditampung dan juga berapa bit yang tersisa. Untuk lebih lengkapnya perhatikan gambar 4.11

Gambar 4.11 *Informasi dari citra cover*

Aplikasi KDT-Indonesia dapat juga menyeleksi apakah citra cover mampu dalam menampung pesan tersebut. Apabila mampu maka akan seperti gambar 4.11, tetapi apabila citra cover tersebut tidak mampu menampung maka aplikasi akan memberikan informasi saran berupa *Messeges Box*.. Untuk lebih jelasnya perhatikan gambar 4.12

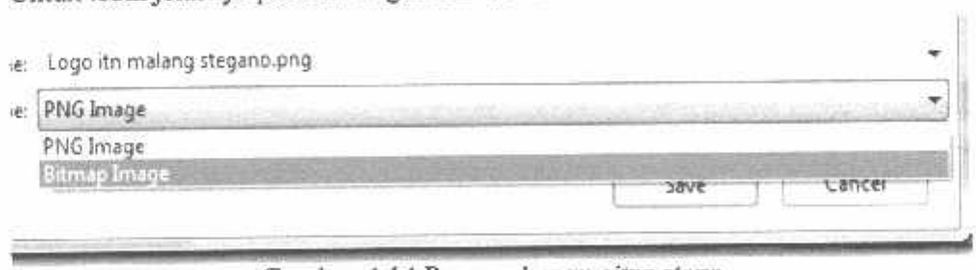


Gambar 4.12 *Pesan peringatan citra cover lebih kecil dari pesan enkripsi*

Apabila sudah selesai menentukan citranya sekarang pesan tersbut disisipkan kedalam citra tersebut dengan cara menekan tombol enkoding pesan kemudian tunggu hingga proses selesai. Untuk lebih jelasnya perhatikan gambar 4.13 untuk pemberitahuan berhasil disisipkan

Gambar 4.13 *Proses enkoding berhasil dieksekusi*

Setelah berhasil dieksekusi proses enkoding, maka secara otomatis aplikasi akan mengeluarkan dialog untuk melakukan proses penyimpanan. Tujuan dari proses ini agar pengguna tidak akan lupa untuk menyimpan citra stego tersebut. Untuk lebih jelasnya perhatikan gambar 4.14.



Gambar *4.14 Proses simpan citra stego*

Pada gambar 4.14 dapat diperhatikan bahwa ekstensi citra yang tersedia dalam menyimpan hanya terdapat 2 ekstensi yakni ekstensi citra *.*png* dan *.*bmp*. Apabila telah selesai menyimpan citra stego, maka aplikasi akan menampilkan informasi mengenai hasil enkoding, ukuran panjang dan lebar , ukuran besar file citra stego serta seberapa cepat proses enkoding terhadap citra yang dipilih. Untuk lebih jelasnya perhatikan gambar 4.15.

Gambar 4.15 *Informasi hasil dari enkoding*

LSB tersedia adalah jumlah total bit LSB yang dapat disisipi pada citra digital tersebut, untuk menghitung jumlah LSB tersedia menggunakan persamaan $\boldsymbol{LSB\ Tersedia} = \frac{(\boldsymbol{Widhth} \times \boldsymbol{Height} \times 3)}{8}$ proses perkalian dengan 3 dikarenakan pada citra digital terdapat 3 intensitas warna yaitu R, G dan B yang tiap LSB dari intesitas tersebut akan digunakan untuk menyisipkan informasi. LSB Tersisa adalah sisa LSB jika disisipi informasi persamaan untuk menghitung jumlah LSB Tersisa adalah LSB Tersisa = LSB Tersedia – Jumlah Karakter Informasi.

4.2.3 Form Dekoding dan Dekripsi

Pada form dekoding dan dekripsi pengguna dapat menggunakan fasilitas mengekstrak pesan yang terdapat pada citra stego dan juga dapat mengdekripsi pesan acak yang diberikan pengirim kepada penerima dengan menggunakan kunci dekripsi. Untuk lebih jelasnya pehatikan gambar 4.16

Gambar 4.16 *Fitur decoding dan dekripsi*

Selanjutnya, apabila pengguna ingin mengetahui pesan apa yang terdapat pada citra stego langsung menekan tombol buka citra. Ekstensi yang diizinkan adalah **.bmp, *.png*. Dalam aplikasi KDT-Indonesia secara otomatis akan menscleksi semua format citra yang digunakan. Apabila citra tersebut memiliki selain **.bmp, *jpg*. Maka secara otomatis akan tidak ditampilkan. Untuk lebih jelasnya perhatikan gambar 4.17

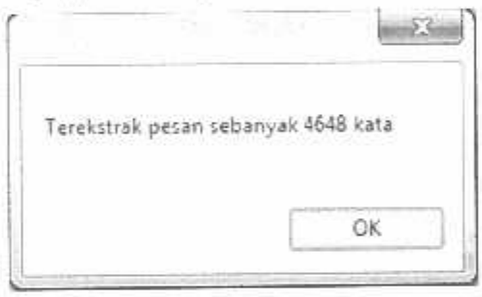
Gambar 4.17 *Proses pemilihan citra stego*

Setelah memilih citra yang hendak dilakukan pengekstrakan pesan, maka aplikasi secara otomatis akan menampilakn informasi dari citra sebut seperti

ukuran pixel citra tersebut, ukuran besar file citra tesebut, serta lokasi dari citra tersebut. Untuk lebih jelasnya perhatikan gambar4.18

Gambar 4.18 *Informasi dari citra stego*

Pada proses selanjutnya adalah mengekstrak pesan yang terdapat pada citra stego dengan cara menekan tombol ekstrak pesan, setelah selesai maka aplikasi akan memberikan informasi berupa berapa panjang pesan yang dapat dibaca oleh LSB. untuk lebih jelasnya perhatikan gambar 4.19



Gambar 4.19 *Informasi panjang pesan yang dapat dibaca LSB*

Jadi pesan yang terbaca dari citra tersebut adalah 4648 kata, jumlah ini harus sama ketika malukan enkripsi. Karena jika berbeda 1 angka saja maka proses enkripsi tidak dapat dilakukan. Sebagai perbandingan jumlah pesan yang terbaca dengan pesan yang dikirim, untuk lebih jelasnya perhatikan gambar 4.20

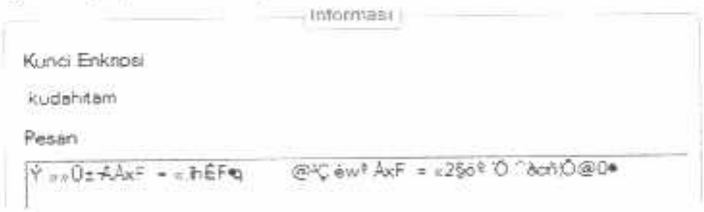
Gambar 4.20 *Pencocokan jumlah karakter yang dikirim dan diterima*

Seperti yang ditampilkan pada gambar 4.20, informasi yang dikirim serta informasi yang diterima memiliki jumlah informasi yang sama. Dengan begitu proses pendekripsian berhasil serta proses pembacaan pesan tidak ada yang terlewat. Jadi dapat dipastikan informasi penerima akan sama dengan informasi pengirim.

Setelah berhasil maka pesan hasil ekstraksi akan ditampilkan pada form pesan. Dalam hal ini pesan yang ditampilkan adalah pesan yang masih terenkripsi. Untuk lebih jelasnya perhatikan gambar 4.21

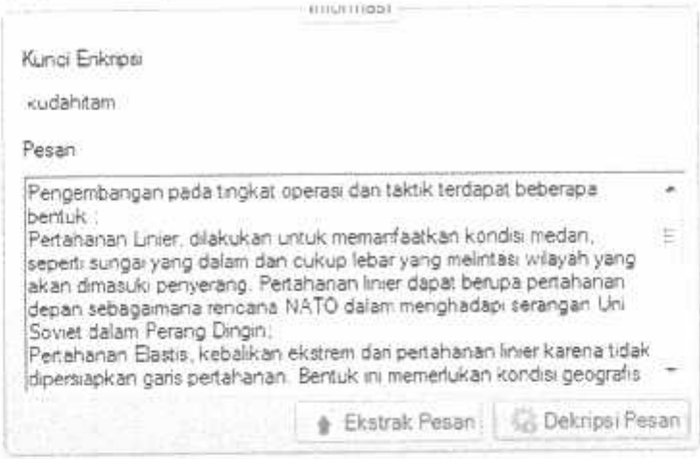
Gambar 4.21 *Informasi pesan yang masih terenkripsi*

Setelah melakukan dekoding pesan citra stego, maka aplikasi akan menampilkan informasi berupa seberapa cepat aplikasi dalam mendekoding pesan citra stego dalam satuan detik. Setelah mengetahui ternyata ada pesan rahasia langkah selanjutnya adalah memasukkan kunci enkripsi. Kunci enkripsi ini adalah kunci dimana kata yang digunakan dalam proses pendekripsian pesan. Kunci ini harus sama , karena algoritma TEA merupakan algoritma dengan kunci simetris. Apabila kunci tersebut berbeda 1 huruf maka hasil yang didekripsi akan berbeda. Untuk lebih jelasnya perhatikan gambar 4.22



Gambar 4.22 Memasukkan kunci enkripsi

Setelah memasukkan kunci maka tekan tombol dekripsi pesan. Maka secara otomatis akan menerjemahkan maksud dari pesan yang tidak dapat dibaca. Untuk lebih jelasnya pehatikan gambar 4.23



Gambar 4.23 *Hasil dekripsi dari pesan rahasia*

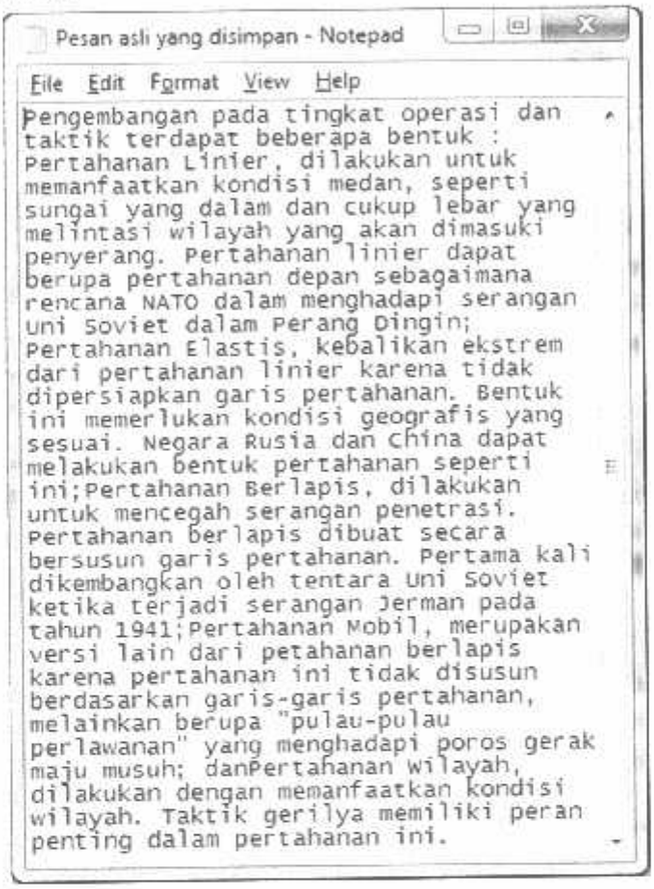
Setelah melakukan proses dekripsi maka aplikasi akan memberikan informasi seberapa cepat aplikasi dalam mengdekripsi pesan. Untuk lebih jelasnya perhatikan gambar 4.24

Gambar 4.24 *Informasi kecepatan dekripsi*

Dalam hal ini pesan sudah dapat dimengerti oleh penerima, tetapi apabila penerima ingin menyimpan pesan tersebut, maka pengguna hanya perlu menekan tombol simpan pesan. Tombol simpan pesan akan muncul apabila terjadi proses dekripsi pesan rahasia. Untuk lebih jelasnya perhatikan gambar 4.25

Gambar 4.25 *Proses penyimpanan pesan*

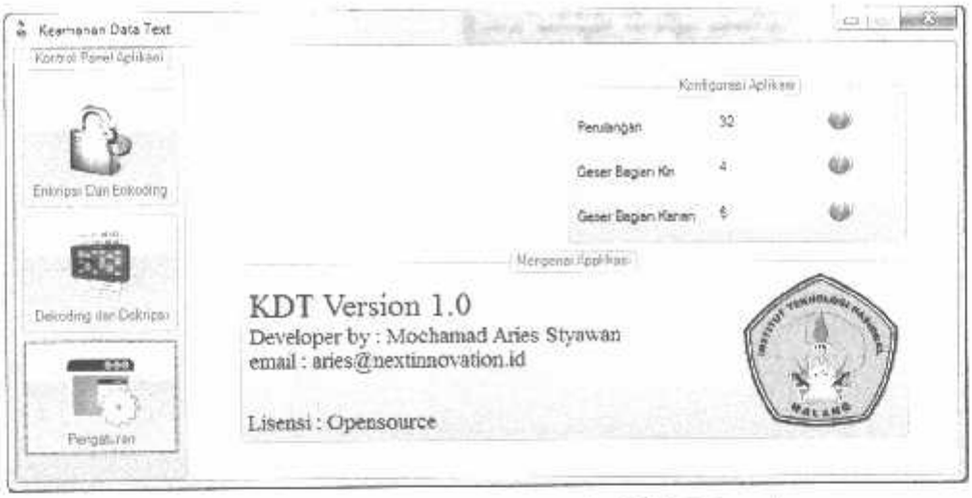
Kemudian untuk melihat pesan tersebut maka pengguna hanya tinggal klik pada nama file yang disimpan tadi. Untuk pembuktian apakah pesan tersimpan semua perhatikan gambar 4.26



Gambar 4.26 *Informasi pesan yang dibaca oleh penerima*

4.2.4 Form Pengatuan

Pada form pengaturan, terdapat nama dari pembuat aplikasi email serta lisensi dari aplikasi KDT-Indonesua dan juga konfigurasi untuk algoritma TEA. Konfigurasi ini mengenai berapa banyak looping, serta pergeseran nilai bit pada pesan rahasia. Untuk lebih jelasnya perhatikan gambar 4.27



Gambar 4.27 *Form pengaturan pada KDT-Indonesia*

Fungsi dari form ini adalah untuk riset seperti TEA apabila looping dirubah dan juga apabila pergeseran bit dirubah , serta dapat juga digunakan untuk tingkat keamanan data, karena dengan adanya fitur ini. Maka orang yang hendak merusak pesan rahasia harus berfikir berpakah looping yang digunakan serta berapakah pergeseran bit yang digunakan.

4.3 Pengujian

Pengujian dilakukan untuk menguji aplikasi yang telah dibuat dan menguji keluaran citra digital yang telah diproses steganografi dengan kriptografi.

4.3.1 Pengujian Kriptografi Enkripsi dan Dekripsi

Proses pengujian kriptografi digunakan untuk menguji hasil pengacakan pesan yang hendak disisipkan ke dalam citra 2D dan dikirimkan kepada penerima. Informasi tersebut di enkripsi menggunakan algoritma TEA

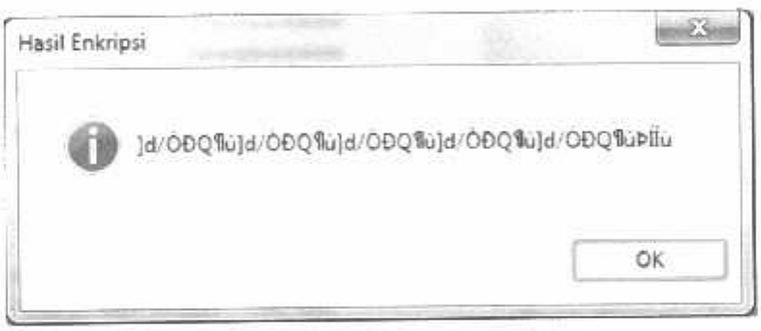
4.3.1.1 Perubahan Pergeseran Bit

Pada pengujian kriptografi enkripsi, pesan yang diacak . dalam pengujian ini akan membedakan dari bentuk *chipertext* berdasarkan bit yang digeser untuk pertama penulis akan memasukkan rumus dengan geser 4 bit serta geser 5 bit pada enkripsi maupun dekripsi. Untuk lebih jelasnya perhatikan gambar 2.28

Gambar 2.28 *Pengaturan pergeseran bit pesan*

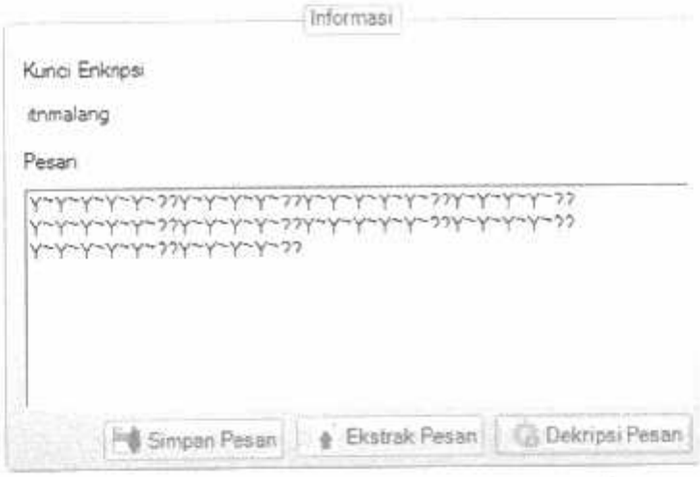
Informasi yang akan dienkripsi adalah huruf "a" yang ditulis 90 kali dan dipishkan tiap spasi tiap 10 karakternya serta menggunakan kunci kriptografi "itn malang".

```
Kunci :
itn malang
Informasi :
aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa
aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa
```

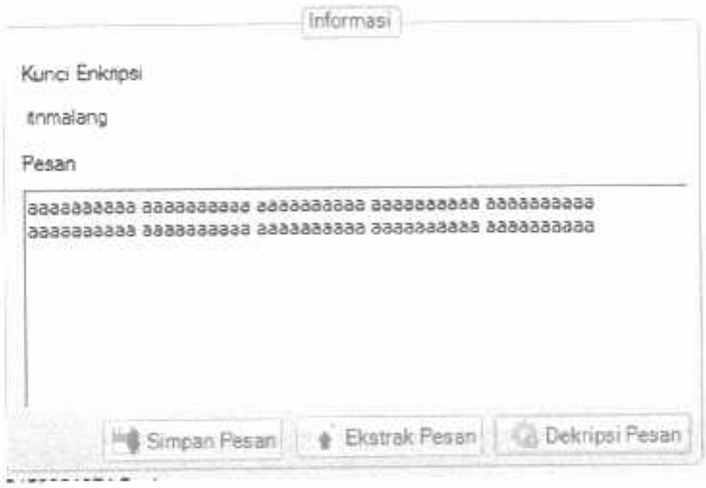


Gambar 2.29 *Merupakan hasil enkripsi dengan pergeseran 4 kiri dan 5 kanan*

Pada gambar 2.29 merupakan hasil enkripsi dengan pergeseran 4 kiri dan 5 kanan. Dalam algoritma TEA pergerseran bit harus sama dengan pergeseran bit enkripsi. Apabila tidak sama maka hasil dekripsi akan berbeda dengan pesan yang dikirim. Dalam percobaan penulis akan merubah geser bit 4 kiri menjadi geser 5 bit kiri. Untuk lebih jelasnya perhatikan gambar 2.30



Gambar 2.30 *Hasil dari dekripsi dengan bit geser 5 kiri dan bit geser 5 kanan*



Gambar 2.31 *Hasil dari dekripsi dengan bit geser 4 kiri dan bit geser 5 kanan*

Dari perbedaan gambar 2.30 dan 2.31 dapat disimpulkan bahwa enkripsi dan dekripsi TEA sangat berpengaruh pada bergesarn bit pada tiap tiap bloknya.

4.3.2 Pengujian Steganografi Enkoding dan Dekoding

Proses pengujian steganografi digunakan untuk menguji hasil citra digital yang telah disisipi dengan informasi yang terenkripsi dengan menggunakan proses steganografi *LSB*.

4.3.2.1 Ukuran Berkas Citra

Proses pengujian dilakukan dengan membandingkan ukuran berkas citra awal dengan ukuran berkas citra sesudah dilakukan proses steganografi. Proses ini dilakukan dengan membandingkan tiap format citra digital sebelum dan sesudah

disisipi dengan informasi dengan menggunakan metode LSB. Berikut adalah daftar citra yang digunakan untuk pengujian.

Format Citra : JPEG
Nama Berkas : gunungfuji.jpg
Ukuran Berkas : 403 Kb
Ukuran Citra : 1600 x 900

Format Citra : PNG
Nama Berkas : alam.png
Ukuran Berkas : 836 Kb
Ukuran Citra : 758 x 421

Format Citra : GIF
Nama Berkas : heroindonesia.gif
Ukuran : 129 Kb
Ukuran Citra : 720 x 480

Format Citra : BMP
Nama Berkas : ranukumbolo.bmp
Ukuran Berkas : 943 KB
Ukuran Citra : 691 x 461

Informasi yang akan dimasukkan dan dienkripsi adalah huruf "a" yang ditulis 90 kali dan dipishkan tiap spasi tiap 10 karakternya. Dan dengan menggunakan kunci kriptografi "itn malang".

```
Kunci :
itn malang
Informasi :
aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa
aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa aaaaaaaaaa
```

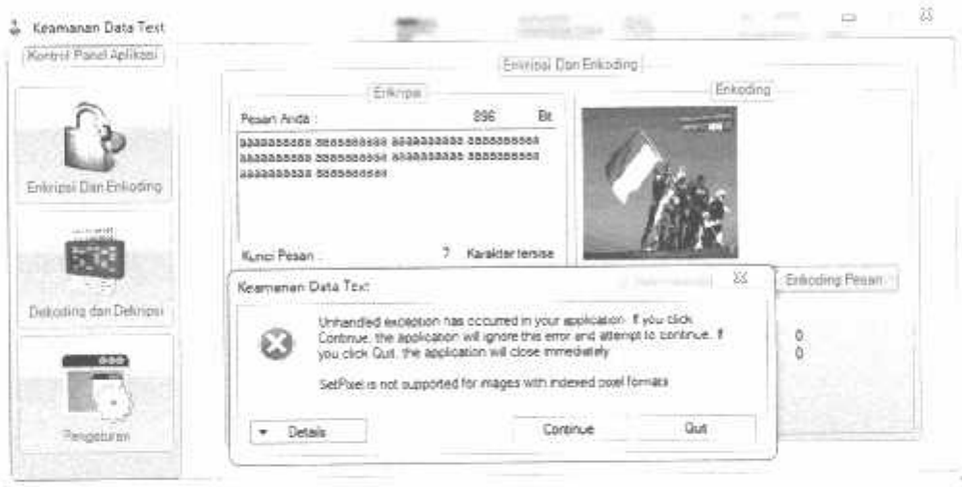
Tiap citra yang di uji akan menggunakan kunci kriptografi dan informasi yang sama pada saat penyisipan citra.

Gambar 4.32 adalah salah satu contoh pengujian citra digital yang diuji dengan parameter kunci dan informasi yang telah ditentukan. Proses tersebut diulangi sebanyak citra yang akan diuji. Yaitu pada citra berformat jpg, gig, bmp dan png. Setelah tiap citra digital disimpan, akan dilihat ukuran berkas sebelum dan sesudah disimpan.

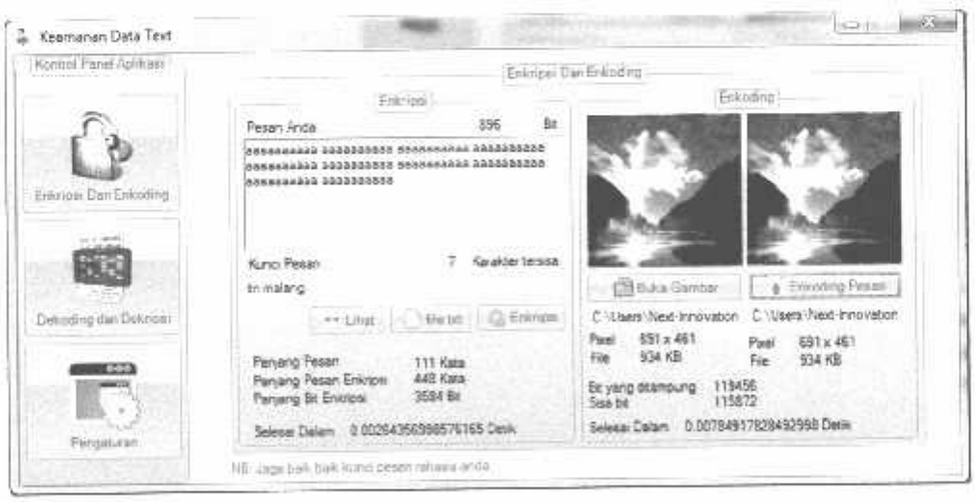
Gambar 4.32 *ujicoba stego citra jpg*

Gambar 4.33 *ujicoba stego citra png*

Gambar 4.34 *Ujicoba citra stego GIF*



Gambar 4.35 *Ujicoba citra stego error GIF*



Gambar 4.36 *Ujicoba sitra stego BMP*

Gambar 4.32 – 4.36 adalah contoh pengujian terhadap citra digital yang berformat tertentu yang disisipi informasi. Seperti yang dilihat bahwa terdapat kesalahan apabila citra yang memiliki format GIF karena perbedaan itensitas warna, karena GIF memiliki layer layer sehingga tidak dapat dilakukan penyisipan pesan pada citra tersebut. Untuk format selain GIF, PNG, JPG,BMP dapat disisipi pesan.

Gambar 4.37 *Ujicoba sitra stego decoding dan dekripsi JPG*

Gambar 4.38 *Ujicoba citra stego decoding dan dekripsi PNG*

Gambar 4.39 *Ujicoba sitra stego decoding dan dekripsi BMP*

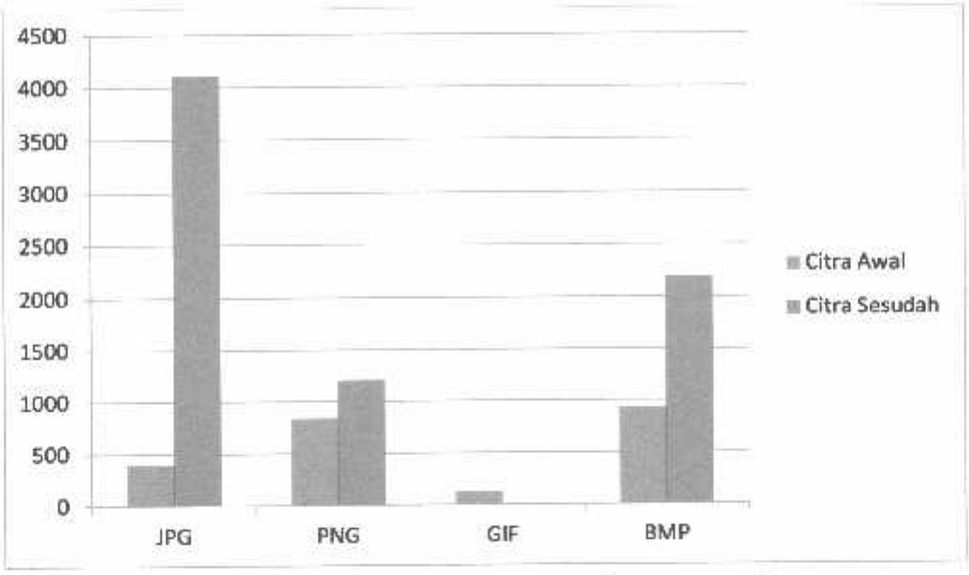
Gambar 4.37 – 4.39 adalah contoh pengujian decoding dan dekripsi pesan yang tersimpan pada citra tersebut. Untuk semua format yang diuji berhasil memunculkan pesan awal dan benar. Untuk format GIF tidak diuji karena saat penyisipan pesan mengalami kesalahan.

Tabel dibawah ini adalah tabel perbandingan hasil ukuran citra sebelum dan sesudah disisipi dengan informasi steganografi.

Tabel 4.1 Perbandingan ukuran citra stego

Nama Citra	Format Citra Sebelum	Format Citra Sesudah	Ukuran Sebelum	Ukuran Sesudah	Informasi Terbaca
gunungfuji.jpg	JPG	PNG	403 Kb	4.11 MB	Ya
alam.png	PNG	PNG	836 Kb	1.21 MB	Ya
heroindonesia.gif	GIF	PNG	128 Kb	error	Tidak
rakum.bmp	BMP	PNG	934 KB	934 KB	Ya

Dari hasil pengujian tersebut, dapat diketahui bahwa beberapa citra tersebut tidak dapat dibaca kembali pesan yang disisipkan. Hal ini dikarenakan perbedaan format citra. Format cita yang masih dapat dibaca informasinya adalah JPG, BMP dan PNG sedangkan format citra dan GIF tidak dapat disisipi pesan. Grafik perubahan ukuran citra digital dapat dilihat pada gambar 4.41.



Gambar 4.40 *Grafik perbandingan ukuran antar citra*

Gambar 4.40 adalah gambar dari diagram dari ukuran berkas citra sebelum dilakukan proses stegano dan sesudah proses stegano. Pada diagram tersebut dapat dilihat bahwa citra jpg memiliki ukuran yang berubah cukup banyak karna pada citra jpg menggunakan *loosy compression* sedangkan pada citra bmp menggunakan *loseless compression.* Schingga ukuran citra tersebut menjadi lebih besar. Sedangkan pada citra gif mengalami pengurangan ukuran berkas dikarenakan citra gif yang digunakan adalah citra gif yang memiliki banyak citra

4.3.2.2 Pengujuan Visual Citra

Pengujian tampilan citra digunakan untuk menguji dan membadingankan citra sebelum dan sesudah dilakukan proses steganografi. Proses ini untuk melihat perbedaan secara fisik tampilan citra digital. Citra yang akan diuji menggunakan citra yang memiliki satu warna yaitu citra hitam dan citra putih. Berikut adalah citra yang akan digunakan untuk dilakukan proses pengujian tampilan citra.

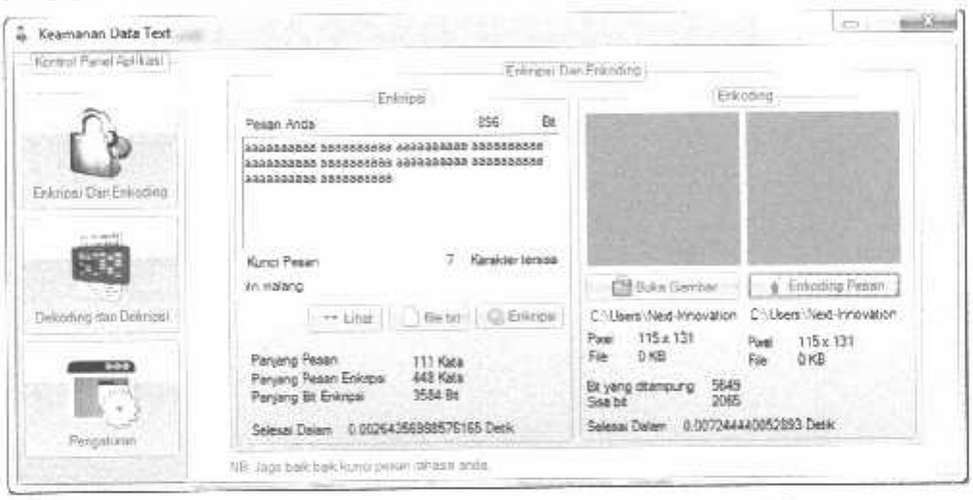
	
Format Citra : PNG	Format Citra : TIF

Nama Berkas :biru.png	Nama Berkas : abu.tif
Ukuran Berkas : 532 byte	Ukuran Berkas :1.41 KB
Ukuran Citra : 151 x 131	Ukuran Citra :115 x 131

Sedangkan informasi yang akan disisipkan adalah sebagai berikut

```
Kunci :
itn malang
Informasi :
aaaaaaaaaa bbbbbbbbbb cccccccccc dddddddddd eeeeeeeeee
ffffffffff gggggggggg hhhhhhhhhh iiiiiiiiii jjjjjjjjjj
```

Penggunaan informasi dan kunci enkripsi digunakan pada kedua citra pengujian untuk mendapatkan hasil yang diinginkan.



Gambar 4.41 *Ujicoba sitra stego tampilan citra PNG*

Gambar 4.42 *Ujicoba sitra stego tampilan citra tif*

Sebelum Proses	Sebelum Proses
 Format Citra : PNG Nama Berkas :biru.png Ukuran Berkas : 532 byte	 Format Citra : PNG Ukuran Berkas : 44.5 Kb Steganografi : Terbaca
Format Citra : TIF Nama Berkas :abuabu.tif Ukuran Berkas : 12,7 Kb	Format Citra : PNG Ukuran Berkas : 44.5 Kb Steganografi : Terbaca

Dari hasil pengujian diatas dapat dilihat bahwa citra digital biru dan abu abu sebelum dan sesudah proses steganografi tidak megalami perbedaan yang terlihat. Hal ini dikarenakan citra proses steganografi dilakukan pada LSB tiap intensitas citra RGB. LSB pada citra digital yang diubah tidak mengubah signifikan tampilan citra.

4.3.2.3 Modifikasi Citra

Proses pengujian steganografi dengan memodifikasi citra digunakan untuk melihat apakah informasi yang disisipkan pada citra digital tetap ada walaupun citra steganografi telah dilakukan proses modifikasi pada citra steganografi tersebut.

Gambar 4.43 *Gambar yang akan diuji*

Gambar 4.43 adalah gambar dari citra yang akan digunakan sebagai uji modifikasi citra digital. Pertama-tama citra akan disisipi dengan informasi dan kunci kriptografi berikut ini :

```
Kunci :
itn malang
Informasi :
aaaaaaaaaa bbbbbbbbbb cccccccccc dddddddddd eeeeeeeeee
ffffffffff gggggggggg hhhhhhhhhh iiiiiiiiii jjjjjjjjjj
```

Setelah citra digital yang telah ditentukan telah disisipi dengan informasi tersebut, langkah selanjutnya adalah dengan melakukan modifikasi citra pada citra steganografi. Proses modifikasi citra tersebut antara lain adalah proses resizing, cropping, filtering, dan kompresi. Berikut adalah gambar yang telah dimodifikasi menggunakan piranti lunak pengeditan citra yaitu *adobe photoshop*.

Citra	**Informasi**
	Format Citra : BMP Proses : crop kanan Ukuran Berkas : 1,952 Kb Ukuran Citra : 740 x 900
	Format Citra : BMP Proses : crop atas Ukuran Berkas : 1,369 KB

	Ukuran Citra : 1600 x 202
	Format Citra : BMP Proses : crop bawah Ukuran Berkas : 2.238 KB Ukuran Citra : 1600 x 264
	Format Citra : BMP Proses : crop kiri Ukuran Berkas : 1,361 KB Ukuran Citra : 516 x 900
	Format Citra : BMP Proses : filter gaussian Ukuran Berkas : 4,219 KB Ukuran Citra :1600 x 900
	Format Citra : JPG Proses : Kompress JPG Ukuran Berkas : 238 KB Ukuran Citra :1600 x 900
	Format Citra : BMP Proses : Resize citra Ukuran Berkas : 1,055 KB Ukuran Citra :800 x 450

Tiap gambar hasil modifikasi tersebut akan diproses pada proses dekripsi pada aplikasi yang telah dibuat.Pada gambar 4.44 merupakan pengujian dekripsi dan dekoding pesan yang disisipkan. Pengujian tersebut adalah citra stego yang telah dirusak dengan cara di filter Gaussian.

Gambar 4.44 *Contoh dekripsi dengan proses crop atas*

Berikut adalah hasil pengujian pada tiap citra yang telah dimodifikasi.

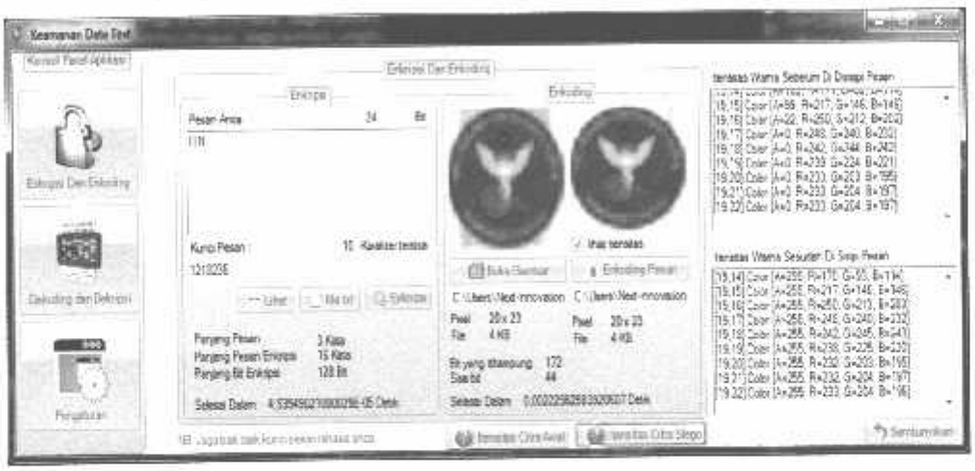
Tabel 4.2 Perbandingan hasil dari modifikasi citra stego

Proses Modifikasi	Pesan didapat	Kesamaan Informasi
Crop Kanan	aaaaaaaaaa bbbbbbbbbb cccccccccc dddddddddd eeeeeeeeee ffffffffff gggggggggg hhhhhhhhhh iiiiiiiii jjjjjjjjj	Ya
Crop atas	Ap ????????5%5%?s?z2?9?_?z????,?-DuE?]??_????To\????_?-?Y ?1?B2??_?J?%???_?_?_'__???~>_n?_??9Q? _(_o???3??F_ W????y,_???f ?_???????bk???G?S?(l??R?T?Oy??~c_????8!'?=2?WZ ?V ?_ q+??F%?Cl??_???_ Xv? F?_@b_???S????????j?_7#Q{???_;_??_M98e?_5	Tidak
Crop kiri	?_? ?_?ghW_??_??~]?U???_???N???_???d??9??_?A? ?_??C_???^?tPk????	Tidak
Crop bawah	aaaaaa Z?_?:c_,?k?k?k?H]??_??_>_?/????????5?KH????&?? ?#_2?9Jb?}?~A??_4A_??x?????pb?nR?$?__b??I?n?_?E? 0p???_0??#?_U_?A?^d?_O???)_??@????n_M]IWY????m?V ???u?]??{?_? ??<1i??{???l?G-?}?^?????0?~_>_??@??????&??& ????w??Nj??q?:G??|r_5w?>??.????g??BLd_ ?? ????(_??mS_?2??_N?'?Oh???!u?NR6?93 ?-??L??um6CW?<P_???LZ?5?+no?(?? ??a??_(??x???_r??_|?{? _? ?2?H3??q???!???????__???(???_Y]#A??p????|?????O?8?_??CR?^??_6@??_*_???aL_?_????A_???V_?AO8m?b<<?g??x_?_?E?????_N??@?U_=??z???|5_e7 ???m???Fp?_?|???\)p???????_?_ ????PD????}?8?????_?m$e	Tidak
Filter gaussian	????H??0?_Ho_ __?_??_???^?P???s_8???A?C3_?{_Oo_J?Y??N_#4D$?_?Z?? _?^???(Kg???-???^??e?j?_?YG??lc?_\??? 6;_?????)????D ?????KM?h?d_?????8?]?<?H?rE_s?_O?u_??_??v_??_9?VW O?_????(tF4?h _????4f????E?A_?a_???-??_? _??5L'?_?s`?{??????B???[7[??\???y?=MF_?????t_?:]???g???? ?-?_??Fq?_?_GY??TgZ_X?yh??b?=Z	Tidak

Kompresi JPEG	)??[??_?_?_??D ?0>_??_^N_??V_?? b _?-e??j V? ??E??`???? /_{ ????cpRsI????^z?_?@?f?s????`4????_???aV+_%!???? ??_??__8H????G9???]mu?i?%a72_m???{?h?q?q??:??_ ?R?? _IGL_?_?_O'_????U?Qog?Y<? K? ??_&	Tidak
Resize citra	L_r???6<????????(?d?g??y?\??_vJ?_???0?u??????V- _??? ?:?&?_?3&?z_????$- xZz}4?_?? _H? ?U?__I??%_???&?F?????_?_?,??_??_?V3?????g? ?_??a? z?/?0??%b ????m????_?DfR?_VM_?=_EN?:P9?_I?:_??U???_g??=???| g|_??N??B:???D	Tidak

4.3.2.4 Perbedaan Intensitas Citra

Pada pengujian intensitas warna citra dapat dilihat pada gambar 4.45.



Gambar 4.45 *Pengujian intensitas warna citra*

Pada aplikasi KDT-Indonesia, lokasi penyisipan citra berada pada akhir citra, dengan value citra asal pada array ke [19,22] Color [A=0, R=233, G=204, B=197] menjadi [19,22] Color [A=255, R=233, G=204, B=196]. Pesan akan disisipkan dari bawah keatas sepanjang pesan yang dikirim. Penyisipan diletakkan pada intensitas warna merah , hijau, biru atau lebih dikenal dengan *Red Green Blue Color* (RGB)

BAB V
PENUTUP

5.1 Kesimpulan

Setelah melakukan pembahasan pada setiap bab dan sub bab dalam laporan skripsi, maka dapat ditarik berberapa kesimpulan yaitu

1. Aplikasi dapat mengenkripsi pesan dari file *.txt yang ditampilkan ke dalam *richtextbox* aplikasi
2. Proses enkripsi menggunakan TEA relative sangat cepat dengan rata-rata 11 detik untuk 10.000 karakter
3. Panjang *plaintext* dibagi setiap 64 bit yang miliki blok kiri dan blok kanan dengan masing masing menjadi 32 bit
4. Panjang *plaintext* terbatas dari jumlah penampung citra cover sedangkan untuk kunci hanya sampai 128 bit yang dibagi 4 blok, dimana setiap blok 32 bit
5. Apabila citra cover tidak mampu untuk menampung pesan maka pesan tidak dapat disisipkan karena apabila dipaksakan maka informasi yang dikirim tidak dapat di dekripsi dengan benar
6. Citra yang disisipi steganografi tidak ada perubahan mencolok secara visual, tetapi ada perubahan yang signifikan pada ukuran file.
7. Pesan rahasia akan disisipkan pada akhir *pixel* citra.
8. Pesan yang disisipkan pada citra akan rusak apabila citra dicrop pada awal pesan disisipi, *resize*, *filter* dikarenakan nilai itensitas tiap *pixel* berubah, sehingga pada saat didekripsi, hasil dekoding LSB tidak sama dengan enkoding LSB
9. LSB dapat disisipkan disetiap format citra dengan catatan hasil output berupa *.bmp / *.png (kecuali GIF)
10. Citra GIF tidak dapat disisipi pesan dikarenakan citra GIF memiliki susunan layer yang tidak sama dengan citra BMP, JPG,TIF,PNG.
11. Pergeseran bit sangat mempengaruhi dari hasil enkripsi dan dekripsi. Apabila pergeseran bit dekripsi tidak sama dengan enkripsi maka pesan asli tidak dapat ditampiilkan

5.2 Saran

1. Dalam melakukan proses ekripsi, sebaiknya usahakan untuk tidak menggunakan kunci yang sama untuk mengenkripsi *file* yang berbeda.
2. Memanfaatkan steganografi yang robustness karena apabila citra stego dimodifikasi informasi tidak hilang
3. Agar kinerja TEA lebih optimal maka digabungkan dengan algoritma kriptografi asimetris dimana algoritma TEA hanya digunakan untuk proses enkripsi dan dekripsi pesan, sementara untuk pembentukan kunci digunakan algoritma asimetris agar kunci yang didistrubusikan tetap aman

DAFTAR PUSTAKA

Ariyus, Dony. 2013. *Pengantar Ilmu Kriptografi*. Jakarta: Andi Publisher.

Bertayla. 2005. *Representasi Citra*. url (http://bertalya.staff.gunadarma.ac.id/Downloads/files/13447/Representasi+Citra.pdf) diunduh tanggal 27 Oktober 2015

Dai, Wei. 2009. *Crypto++ Benchmarks*. url (https://www.cryptopp.com/benchmarks.html) diakses tanggal 23 Oktober 2015

Efvy, Zam. 2013. *Anti Privacy Melacak, Membajak dan Membobol Data Rahasia*. Jakarta: Media Kita.

Fajar, Astuti Hermawati. 2013. *Pengolahan Citra Digital : Konsep dan Teori* . Jakarta: Andi Publisher.

Febriyani, Dea. 2016. *ASCII (American Standard Code for Information Interchange)*. url (http://blog.umy.ac.id/deafebryani/teknologi/ascii-american-standard-code-for-information-interchange/) diakses tanggal 15 Oktober 2015

Kurniawan, Reski, Nur Asriyanti Bagenda, Raodah Jabir, Muh Jayadi Kara. 2014. *Laporan Metode Komputasi*. Universitas Hasanuddin. url (https://www.academia.edu/9762069/Metode_Komputasi_Menggunakan_C_untuk_perhitungan_Simple_Pendulum_Equation) diundunh tanggal 20 Oktober 2015

Lannueardy, Eko. 2013, *Visual Studio 2013, Alat Kerja Terbaru Untuk Developer*. url (http://www.chip.co.id/news/web_internet-software_os-microsoft/9038/visual_studio_2013_alat_kerja_terbaru_untuk_developer) diakses tanggal 22 Oktober 2015

Liauw, Hindra. 2014. *Kejahatan di Dunia Maya Kian Berbahaya*. url (http://megapolitan.kompas.com/read/2014/10/24/17561041/print.kompas.com) diakses tanggal 13 Oktober 2015

Oktiavia, Elvi. 2010. *Konsep Object Oriented Programming (OOP) Dalam Pemrograman Visual*. Skripsi Universitas Sriwijaya : tidak diterbitkan

LAMPIRAN

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : Mochamad Aries Setyawan
NIM : 1218236
JURUSAN : Teknik Informatika S-1
JUDUL : Aplikasi Keamanan Data Teks Menggunakan Metode Tiny Encryption Algorithm (TEA) Dan Least Significant Bits (LSB)

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :
Hari : Jum'at
Tanggal : 15 Januari 2016
Nilai : 85,92 (A)

Panitia Ujian Skripsi :

Ketua Majelis Penguji

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Anggota Penguji :

Dosen Penguji I

Febriana Santi Wahyuni, S.Kom, M.Kom.
NIP. P. 1031000425

Dosen Penguji II

Agung Panji Sasmito, S.Pd, M.Pd.
NIP. P 1031500499

Lampiran 1 : Lembar Dosen Pembimbing 1

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NAGARI MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting) Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 23 Oktober 2015

Nomor : ITN-593/I.INF/TA/2015
Lampiran : —
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Joseph Dedy Irawan, ST, MT
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : MOCHAMAD ARIES SETYAWAN
Nim : 1218236
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

23 Oktober 2015 S/D 23 Maret 2016

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
Ketua,

Joseph Dedy Irawan,ST., MT.
NIP : 197404.62005021002

Form S-4a

Lampiran 1 : Lembar Dosen Pembimbing 2

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BINA PERSADA MALANG
UNIVERSITAS BRAWIJAYA MALANG
Kampus I Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 23 Oktober 2015

Nomor : ITN-593/I.LN/-TA/2015
Lampiran : —
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Sandy Nataly Mantja, S.Kom
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : MOCHAMAD ARIES SETYAWAN
Nim : 1218236
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

23 Oktober 2015 S/D 23 Maret 2016

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
Ketua,

Joseph Dedy Irawan,ST., MT.
NIP : 197404162005021002

Form S-4a

Lampiran 2 : Lembar asistensi dosen pembimbing 1

LEMBAR BIMBINGAN SKRIPSI

Nama : Mochamad Aries Setyawan
NIM : 1218236
Masa Bimbingan : 23 Oktober 2015 s/d 23 Maret 2016
Judul Skripsi : Aplikasi Keamanan Data Teks Menggunakan Metode Tiny Encryption Algorithm (TEA) Dan Least Significant Bits (LSB)

No	Tanggal	Uraian	Paraf
1	4 November 2015	ACC Design Aplikasi. Bab I. Revisi : Gunakan kalimat yang efektif, bahasa selain bahasa Indonesia dicetak miring, kata hubung tidak diawal kalimat.Berikan batasan masalah besaran pesan tegantung besaran citra.	
2	7 November 2015	Revisi : Perubahan metode dari End Of File menjadi Least Significant Bits	
3	9 November 2015	Bab I : OK. Bab II. Revisi : Mengapa Tiny Encryption Algorithm (TEA) dan Least Significant Bit (LSB).jabarkan kedalam bentuk table dan grafik. Jabarkan juga perbedaan kunci simetris dan kunci asimetris	
4	14 November 2015	Bab II : OK. Bab III. Revisi : Pada blog diagram hanya pelaku saja yang dimasukkan, pada UML dan jabarkan secara singkat dan jelas, perhitungan manual TEA 1 blok saja	
5	16 November 2015	Bab III : OK, Bab IV. Revisi :untuk pengujian TEA di tampilkan pesan sebelum di enkripsi dan sesudah dienkripsi, diberikan informasi pesan contoh panjang pesan, berapa bit yang dipakai.	
6	17November 2015	Bab IV : OK. Pada setiap proses baik kriptografi maupun steganografi diberikan informasi pada akhir dari prosesnya.Padding pada aplikasi disembunyikan, Untuk pesannya coba sisipkan pada akhir pixel citra.	
7	24 November 2015	Demo Aplikasi ACC + ACC Seminar Progress	

No	Tanggal	Uraian	Paraf
8	5 Desember 2015	Demo Aplikasi. Revisi : Mendinamiskan bitwise TEA, Memberikan fasilitas auto save ketika menyisipkan pesan. Memberikan suggest besaran citra agar muat disisipi pesan.	
9	8 Desember 2015	Bab V : Kesimpulan diambil dari apa yang sudah dikerjakan dan tidak melenceng terlalu jauh dari rumusan masalah.	
10	10 Desember 2015	Bab V : OK + Revisi abstrak,terdiri dari pendahuluan, apa yang dibuat, dan kesimpulan.	
11	11 Desember 2015	ACC Seminar Hasil + Power Point OK	
12	5 Januari 2016	Demo Aplikasi : OK, cek laporan, periksa dan gunakan kata baku	
13	10 Januari 2016	Perhitungan manual TEA dan LSB : OK. Aplikasi : OK, ACC Sidang Kompre	

Dosen Pembimbing I

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Lampiran 2 : Lembar asistensi dosen pembimbing 1

LEMBAR BIMBINGAN SKRIPSI

Nama : Mochamad Aries Setyawan
NIM : 1218236
Masa Bimbingan : 23 Oktober 2015 s/d 23 Maret 2016
Judul Skripsi : Aplikasi Keamanan Data Teks Menggunakan Metode Tiny Encryption Algorithm (TEA) Dan Least Significant Bits (LSB)

No	Tanggal	Uraian	Paraf
1	4 November 2015	ACC Design Aplikasi. Bab I. Revisi : Gunakan kalimat yang efektif, bahasa selain bahasa Indonesia dicetak miring, kata hubung tidak diawal kalimat.Berikan batasan masalah besaran pesan tegantung besaran citra.	
2	7 November 2015	Revisi : Perubahan metode dari End Of File menjadi Least Significant Bits	
3	9 November 2015	Bab I : OK. Bab II. Revisi : Mengapa Tiny Encryption Algorithm (TEA) dan Least Significant Bit (LSB).jabarkan kedalam bentuk table dan grafik. Jabarkan juga perbedaan kunci simetris dan kunci asimetris	
4	14 November 2015	Bab II : OK. Bab III. Revisi : Pada blog diagram hanya pelaku saja yang dimasukkan, pada UML dan jabarkan secara singkat dan jelas, perhitungan manual TEA 1 blok saja	
5	16 November 2015	Bab III : OK, Bab IV. Revisi :untuk pengujian TEA di tampilkan pesan sebelum di enkripsi dan sesudah dienkripsi, diberikan informasi pesan contoh panjang pesan, berapa bit yang dipakai.	
6	17November 2015	Bab IV : OK. Pada setiap proses baik kriptografi maupun steganografi diberikan informasi pada akhir dari prosesnya.Padding pada aplikasi disembunyikan, Untuk pesannya coba sisipkan pada akhir pixel citra.	
7	24 November 2015	Demo Aplikasi ACC + ACC Seminar Progress	

No	Tanggal	Uraian	Paraf
8	1 Desember 2015	Bab IV , Revisi : Saat implementasi berikan penjelasan pada tiap proses minimal 3 kalimat. Saat pengujuan cepat itu seberapa cepat jabarkan dalam bentuk angka yang valid. Tombol pada aplikasi jangan di aktifkan semua, kondisikan aktif saat proses sebelumnya dijalankan	
9	2 Desember 2015	Aplikasi : OK, Bab IV. Refisi : Pengujian ditambahkan dan cari referensi cara menguji TEA dan LSB.	
10	10 Desember 2015	Bab IV : OK, Bab V. Revisi : Untuk kesimpulan ditambahkan minimal 5 kesimpulan, minimal harus ada 1 kesimpulan yang dikaitkan pada abstrak.	
11	12 Desember 2015	BAB V: OK, Daftar pustaka : OK. Pelajari lagi manual dari TEA dan LSB, buat PPT untuk ujian kompre. Aplikasi di deploy menjadi package installan	

Dosen Pembimbing II

Sandy Nataly Mantja, S.Kom

NIP.P. 1030800418

Lampiran 3 Listing Program : Class AlgoritmaTEA

```
class AlgoritmaTEA
  {
    /*public uint loopingnyabro;
    public int gantiempat;
    public int gantilima; */
    public AlgoritmaTEA() { }
    public uint[] formatkuncipesan(string Kunci)
    {
      Kunci = Kunci.PadRight(16, ' ').Substring(0, 16);
      uint[] formatkunci = new uint[4];
      int j = 0;
      for (int i = 0; i < Kunci.Length; i += 4)
        formatkunci[j++] = Konversi.RubahStringToUInt(Kunci.Substring(i,
4));
      return formatkunci;
    }
    public void encode(uint[] pesan, uint[] kunci, uint loopingnyabro, int
gantiempat, int gantilima)
    {
      uint kiri = pesan[0];
      uint kanan = pesan[1];
      uint total = 0;
      uint delta = 0x9e3779b9;
      uint a = loopingnyabro;
      while (a-- > 0)
      {
        total += delta;
        kiri += (kanan << gantiempat) + kunci[0] ^ kanan + total ^ (kanan >>
gantilima) + kunci[1];
```

```
            kanan += (kiri << gantiempat) + kunci[2] ^ kiri + total ^ (kiri >>
gantilima) + kunci[3];
        }
        pesan[0] = kiri;
        pesan[1] = kanan;
    }
    public string enkripsipesan(string Data, string Key, uint
enkripsiloopingnya, int enkripsigantiempat, int enkripsigantilima)
    {
        uint[] formatkunci = formatkuncipesan(Key);

        if (Data.Length % 2 != 0) Data += '\0';
        byte[] bitkarakter =
System.Text.ASCIIEncoding.ASCII.GetBytes(Data);

        string cipher = string.Empty;
        uint[] variabeltempung = new uint[2];
        for (int i = 0; i < bitkarakter.Length; i += 2)
        {

            variabeltempung[0] = bitkarakter[i];
            variabeltempung[1] = bitkarakter[i + 1];
            encode(variabeltempung, formatkunci, enkripsiloopingnya,
enkripsigantiempat, enkripsigantilima);
            cipher += Konversi.RubahUIntToString(variabeltempung[0]) +
Konversi.RubahUIntToString(variabeltempung[1]);
        }
        return cipher;
    }
    public void decode(uint[] pesan, uint[] kunci, uint loopingnyabro, int
gantiempat, int gantilima)
    {
```

```
        uint n = loopingnyabro;
        uint total;
        uint kiri = pesan[0];
        uint kanan = pesan[1];
        uint delta = 0x9e3779b9;

        total = delta << 5;

        while (n-- > 0)
        {
          kanan -= (kiri << gantiempat) + kunci[2] ^ kiri + total ^ (kiri >>
gantilima) + kunci[3];
          kiri -= (kanan << gantiempat) + kunci[0] ^ kanan + total ^ (kanan >>
gantilima) + kunci[1];
          total -= delta;
        }

        pesan[0] = kiri;
        pesan[1] = kanan;
    }
    public string Decrypt(string Data, string Key, uint enkripsiloopingnya, int
enkripsigantiempat, int enkripsigantilima)
    {
        uint[] formatkunci = formatkuncipesan(Key);

        int x = 0;
        uint[] variabeltempung = new uint[2];
        byte[] bitkarakter = new byte[Data.Length / 8 * 2];
        for (int i = 0; i < Data.Length - 9; i += 8)
        {
          variabeltempung[0] = Konversi.RubahStringToUInt(Data.Substring(i,
4));
```

```
            variabeltempung[1] = Konversi.RubahStringToUInt(Data.Substring(i
+ 4, 4));
            decode(variabeltempung, formatkunci, enkripsiloopingnya,
enkripsigantiempat, enkripsigantilima);
            bitkarakter[x++] = (byte)variabeltempung[0];
            bitkarakter[x++] = (byte)variabeltempung[1];
        }

        string penguraikata =
System.Text.ASCIIEncoding.ASCII.GetString(bitkarakter, 0,
bitkarakter.Length);
        if (penguraikata[penguraikata.Length - 1] == '\0') // Strip the null char if
it was added.
            penguraikata = penguraikata.Substring(0, penguraikata.Length - 1);
        return penguraikata;
    }
  }
```

Lampiran 3 Listing Program : Class LSB

```
class LSB
  {
    public enum Status
    {
      Hiding,
      Filling_With_Zeros
    }
    public static Bitmap embedText(string text, Bitmap bmp)
    {
      //Awalnya, kita membuat status karakter dalam gambar
      Status Status = Status.Hiding;
      //Menampung indeks dari karakter yang sedang disembunyikan
      int charIndex = 0;
      //Menampung nilai karakter dikonversi ke integer
      int charValue = 0;
      //Menampung indeks dari elemen warna (R atau G atau B) yang saat ini
sedang diproses
      long pixelElementIndex = 0;
      //Menampung jumlah trailing "kapan dia berhenti 0" nol yang telah
ditambahkan ketika menyelesaikan proses
      int zeros = 0;
      //Menampung Pixel RGB
      int R = 0, G = 0, B = 0;
      int baris = bmp.Height - 1; int kolom = bmp.Width - 1;
      //Mencari akhir dari panjang dan lebar citra
      for (int i = kolom; i >= 0; i--)
      {
        for (int j = baris; j >= 0; j--)
        {
          Color pixel = bmp.GetPixel(i, j);
```

```
            //Console.WriteLine(pixel);
            //Mendapatkan nilai itensitas RGB
            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++)
            {
              // check if new 8 bits has been processed
              if (pixelElementIndex % 8 == 0)
              {
                // check if the whole process has finished
                // we can say that it's finished when 8 zeros are added
                if (Status == Status.Filling_With_Zeros && zeros == 8)
                {
                  // apply the last pixel on the image
                  // even if only a part of its elements have been affected
                  if ((pixelElementIndex - 1) % 3 < 2)
                  {
                    bmp.SetPixel(i, j, Color.FromArgb(R, G, B));
                  }

                  // return the bitmap with the text hidden in
                  return bmp;
                }

                // check if all characters has been hidden
                if (charIndex >= text.Length)
                {
                  // start adding zeros to mark the end of the text
                  Status = Status.Filling_With_Zeros;
                }
                else
                {
```

```
                // move to the next character and process again
                charValue = text[charIndex++];
            }
        }

        // check which pixel element has the turn to hide a bit in its LSB
        switch (pixelElementIndex % 3)
        {
            case 0:
                {
                    if (Status == Status.Hiding)
                    {
                        // the rightmost bit in the character will be (charValue
% 2)
                        // to put this value instead of the LSB of the pixel
element
                        // just add it to it
                        // recall that the LSB of the pixel element had been
cleared
                        // before this operation
                        R += charValue % 2;

                        // removes the added rightmost bit of the character
                        // such that next time we can reach the next one
                        charValue /= 2;
                    }
                } break;
            case 1:
                {
                    if (Status == Status.Hiding)
                    {
                        G += charValue % 2;
```

```
                    charValue /= 2;
                }
            } break;
        case 2:
            {
                if (Status == Status.Hiding)
                {
                    B += charValue % 2;
                    charValue /= 2;
                }
                bmp.SetPixel(i, j, Color.FromArgb(R, G, B));
            } break;
    }
    pixelElementIndex++;
    if (Status == Status.Filling_With_Zeros)
    {
        // increment the value of zeros until it is 8
        zeros++;
    }
  }
 }
}
return bmp;
}

public static string extractText(Bitmap bmp)
{
  int colorUnitIndex = 0;
  int charValue = 0;
  int a = 0;

  // holds the text that will be extracted from the image
```

```
string extractedText = String.Empty;

int baris = bmp.Height - 1; int kolom = bmp.Width - 1;
for (int i = kolom; i >= 0; i--)
{
   for (int j = baris; j >= 0; j--)
   {
      Color pixel = bmp.GetPixel(i, j);
      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);
            //Console.WriteLine("Now value " + a);
            if (charValue == 120)
            { a = a + 1; } else { a = 0; }
            if (a == 10)
            {
               return extractedText;
            }
            else {
```

```
                    char c = (char)charValue;
                    extractedText += c.ToString();
                }
            }
        }

    }
}

return extractedText;
}

public static int reverseBits(int n)
{
    int result = 0;

    for (int i = 0; i < 8; i++)
    {
        result = result * 2 + n % 2;

        n /= 2;
    }

    return result;
}
}
```

```
class Konversi
  {

    private long start;
    private long stop;
    private long frequency;

    [DllImport("KERNEL32")]
    private static extern bool QueryPerformanceCounter(
    out long lpPerformanceCount);

    [DllImport("Kernel32.dll")]
    private static extern bool QueryPerformanceFrequency(out long
lpFrequency);

    public Konversi()
    {
      if (QueryPerformanceFrequency(out frequency) == false)
      { throw new Win32Exception(); }
    }

    public void Start()
    { QueryPerformanceCounter(out start); }

    public void Stop()
    { QueryPerformanceCounter(out stop); }

    public double Duration()
    { return (((double)(stop - start) / (double)frequency)); }

    public static uint RubahStringToUInt(string Input)
```

```
    {
        uint output;
        output = ((uint)Input[0]);
        output += ((uint)Input[1] << 8);
        output += ((uint)Input[2] << 16);
        output += ((uint)Input[3] << 24);
        return output;
    }
    public static string RubahUIntToString(uint Input)
    {
        System.Text.StringBuilder output = new System.Text.StringBuilder();
        output.Append((char)((Input & 0xFF)));
        output.Append((char)((Input >> 8) & 0xFF));
        output.Append((char)((Input >> 16) & 0xFF));
        output.Append((char)((Input >> 24) & 0xFF));
        return output.ToString();
    }

}
```