BAB III

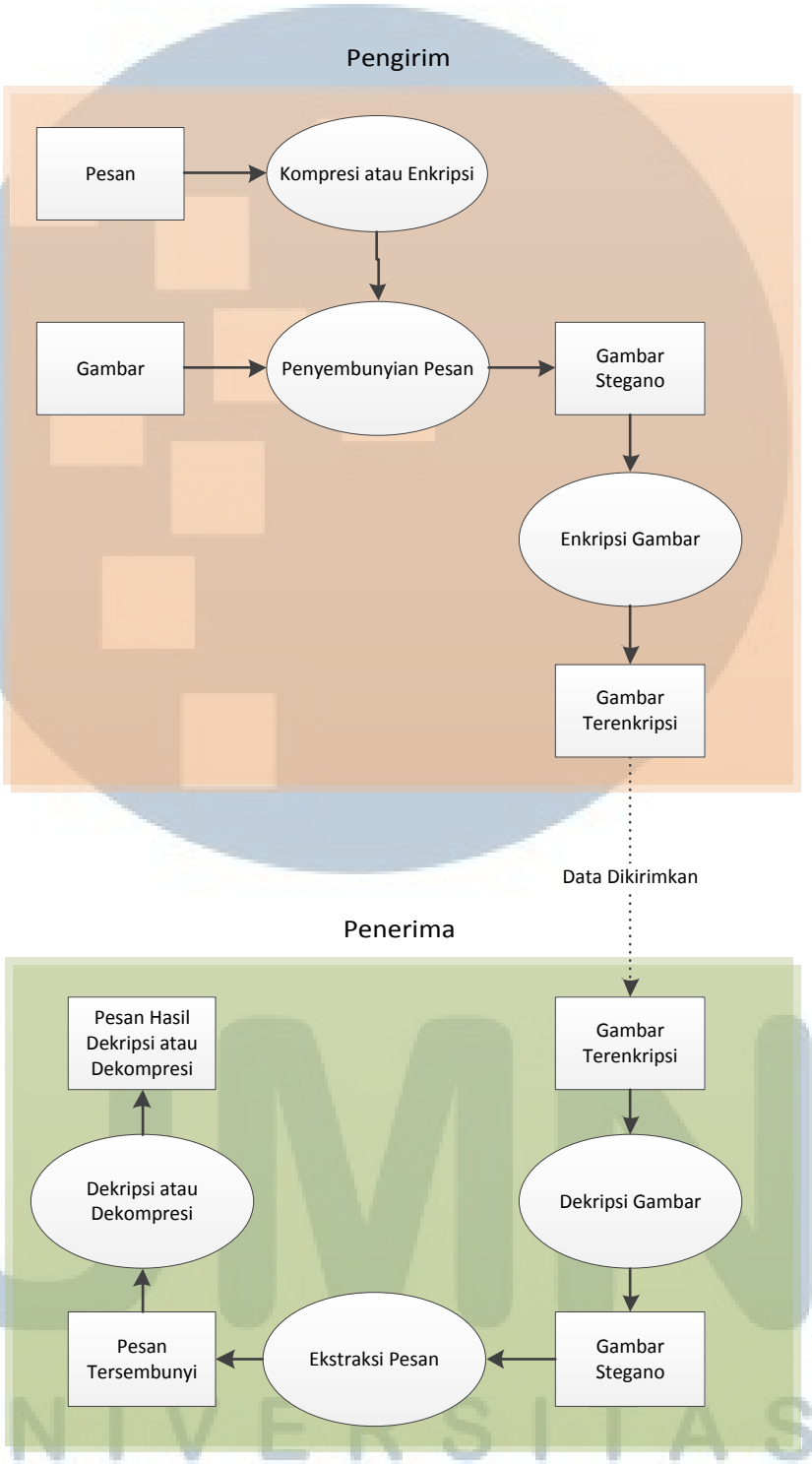
ANALISIS DAN PERANCANGAN SISTEM

3.1 Spesifikasi Sistem

Sistem yang dibuat meliputi diagram sistem, fungsionalitas, tujuan, batasan, serta masukan dan keluaran sistem.

3.1.1 Diagram Sistem

Gambar 3.1 menunjukkan diagram yang menunjukan kerja sistem. Masing-masing proses dalam diagram tersebut merupakan proses yang terpisah dan bersifat *optional*. Data yang dikirimkan dapat berupa teks atau gambar. Teks dapat berupa teks normal (*plaintext*), teks terenkripsi atau pun terkompresi. Gambar dapat berupa gambar yang hanya dienkripsi, gambar yang hanya di steganografi, dan gambar yang melalui proses steganografi dan enkripsi.



Gambar 3.1 Diagram Sistem

3.1.2 Fungsionalitas Sistem

Fungsionalitas yang dimiliki oleh sistem adalah.

1. Melakukan kompresi pada pesan dengan algoritma *deflate*.
2. Melakukan dekompresi pada pesan terkompresi dengan algoritma *deflate*.
3. Melakukan enkripsi pada pesan dengan algoritma AES 128 bit.
4. Melakukan dekripsi pada pesan terenkripsi dengan algoritma AES 128 bit.
5. Menyisipkan pesan (baik terenkripsi atau terkompresi atau pun tidak sama sekali) ke dalam gambar dengan metode steganografi *bit-plane-complexity-segmentation*.
6. Mengambil pesan tersembunyi pada gambar yang telah melalui proses steganografi yang oleh sistem yang dibuat.
7. Melakukan enkripsi terhadap gambar dengan metode *nonlinear-chaotic-algorithm.*
8. Melakukan dekripsi terhadap gambar yang telah terenkripsi dengan metode *nonlinear-chaotic-algorithm*.

3.1.3 Masukkan dan Keluaran Sistem

Masukkan yang dibutuhkan sistem:

1. Untuk modul kompresi pesan.
   a. Pesan (*string*) atau lokasi *file* yang berisi pesan yang akan dikompresi.
   b. Lokasi *file* teks hasil kompresi yang akan didekompresi.
   c. Lokasi dan nama *file* hasil kompresi atau dekompresi.

2. Untuk modul enkripsi pesan.
    a. Pesan (*string*) atau lokasi *file* yang berisi pesan yang akan dienkripsi
    b. Lokasi *file* teks hasil enkripsi yang akan didekripsi.
    c. *Key* dan *initial value* (IV) untuk mendekripsi pesan.
    d. Lokasi dan nama *file* hasil enkripsi atau dekripsi.

3. Untuk modul steganografi.
    a. Lokasi *file* gambar yang akan disteganografi.
    b. Lokasi *file* gambar hasil steganografi untuk diambil kembali pesannya.
    c. Pesan (*string*) atau lokasi *file* yang berisi pesan yang disembunyikan.
    d. Lokasi dan nama *file* gambar hasil penyembunyian pesan.
    e. Lokasi dan nama *file* pesan hasil ekstraksi.

4. Untuk modul enkripsi gambar.
    a. Lokasi *file* gambar yang akan dienkripsi.
    b. *Key* (Alpha, Beta, dan x0) untuk enkripsi atau dekripsi
    c. Lokasi dan nama *file* hasil enkripsi atau dekripsi.

Keluaran dari sistem berupa:

1. Untuk modul kompresi pesan.
    a. *File* hasil kompresi atau dekompresi.
2. Untuk modul enkripsi pesan.
    a. *Key* dan *IV* yang digunakan dalam enkripsi

b. *File* hasil enkripsi atau dekripsi.

3. Untuk modul steganografi.
   a. Gambar dari lokasi yang dimasukkan oleh pengguna.
   b. Gambar hasil steganografi.
   c. *File* gambar hasil penyembunyian pesan.
   d. *File* pesan hasil ekstraksi.

4. Untuk modul enkripsi gambar.
   a. Gambar dari lokasi yang dimasukkan oleh pengguna.
   b. Gambar hasil enkripsi.
   c. *File* gambar hasil enkripsi.

3.2 Desain Sistem

Sistem akan dibuat sebagai sebuah aplikasi *Multiple Document Interface* (MDI). Aplikasi ini akan dapat mengakses modul sebagai berikut.

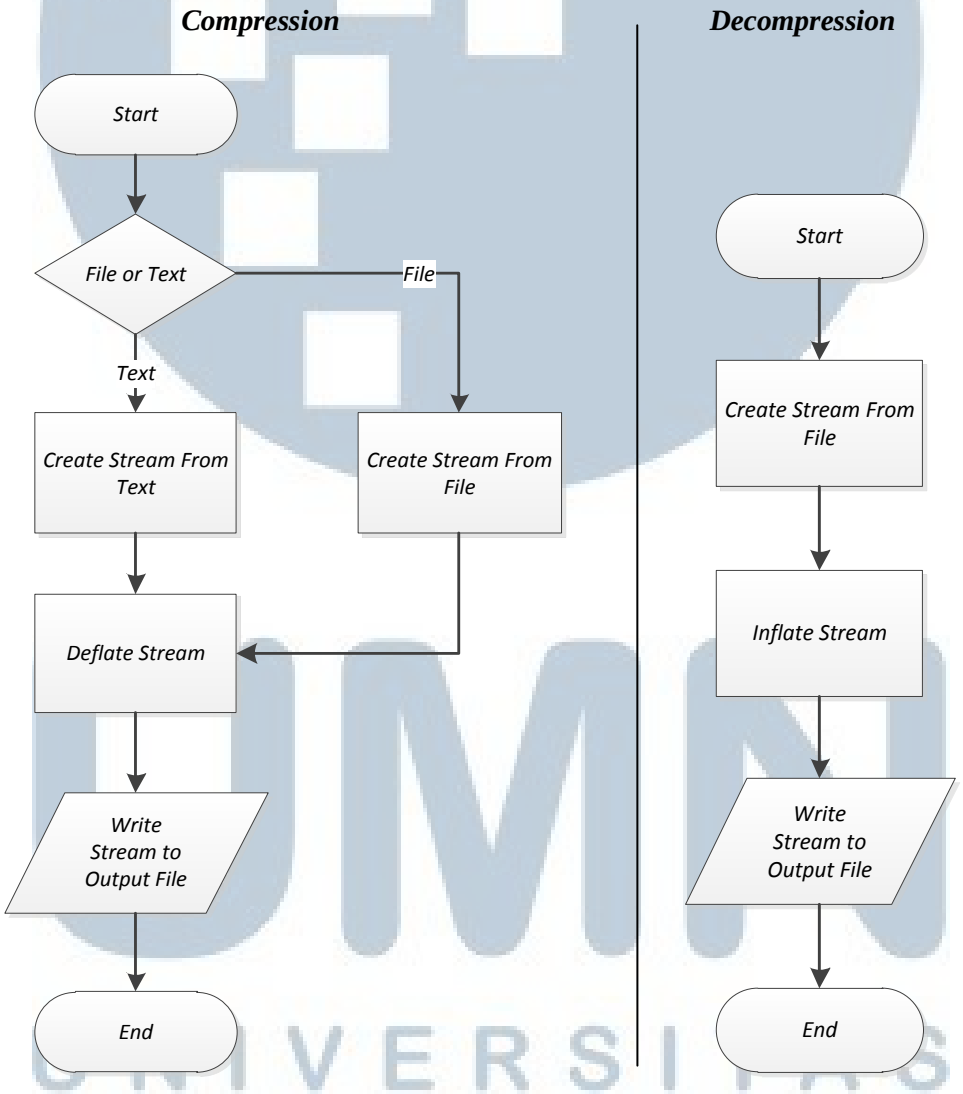
1. Modul kompresi pesan.
2. Modul enkripsi pesan.
3. Modul steganografi.
4. Modul enkripsi gambar.

Berikut merupakan penjelasan masing-masing modul, hirarki menu, serta tampilan antarmuka dari sistem.

3.2.1 Desain Modul

a. Desain Modul Kompresi Pesan

Modul ini berfungsi untuk mengompres dan mendekompresi pesan (teks). Kompresi akan menerima masukan sebuah *file* teks atau pesan secara langsung. Berikut merupakan *flowchart* dari modul ini.



Gambar 3.2 *Flowchart* Kompresi dan Dekompresi Pesan

Modul kompresi ini menggunakan *deflate algorithm* yang telah tersedia pada *.NET Framework*. Ketika melakukan kompresi, pengguna akan diminta untuk memberikan lokasi dan nama *file* hasil pengompresan. Pengompresan hanya akan berjalan apabila pengguna memasukkan lokasi dan nama *file* hasil pengompresan.

b. Desain Modul Enkripsi Pesan

Modul ini berfungsi untuk mengenkripsi dan mendekripsi pesan (teks). Enkripsi akan menerima masukan sebuah *file* teks atau pesan secara langsung. Gambar 3.2 menunjukkan *flowchart* dari modul ini.

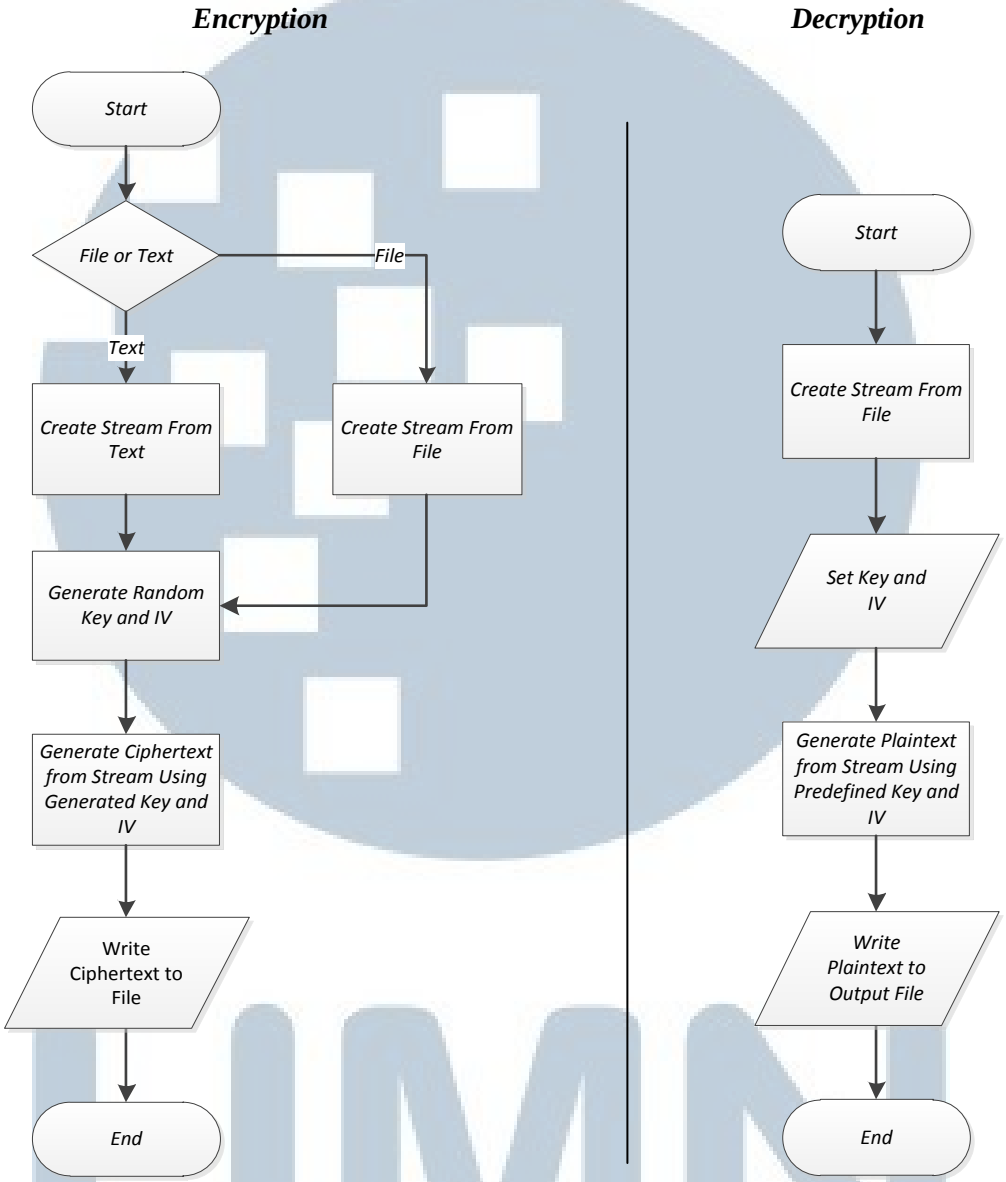
Modul enkripsi ini menggunakan algoritma AES 128 bit yang telah tersedia pada *.NET Framework*. Ketika melakukan enkripsi, pengguna akan diminta untuk memberikan lokasi dan nama *file* hasil enkripsi. Enkripsi hanya akan berjalan apabila pengguna memasukkan lokasi dan nama *file* hasil enkripsi.

Setelah Enkripsi selesai, modul akan memberikan keluaran berupa *Initital Value* (IV) dan *Key* yang merupakan *Random Generated Value*. Kedua nilai ini akan dibutuhkan saat akan melakukan dekripsi.

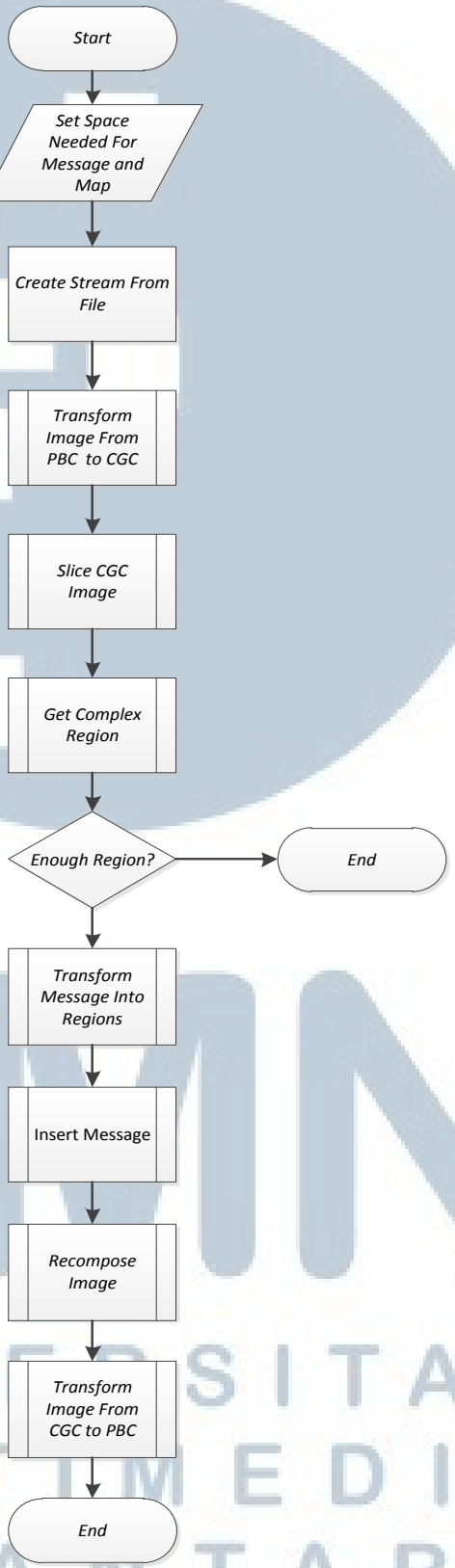
c. Desain Modul Steganografi

Modul steganografi dibuat dengan menggunakan metode *bit-plane-complexity-segmentation*. Metode ini memiliki ciri memanfaatkan kompleksitas gambar untuk menyembunyikan data (pesan). Metode

steganografi *bit-plane-complexity-segmentation* tidak memiliki suatu bentuk pasti. Walaupun berdasarkan pada prinsip yang sama, setiap orang dapat memakai cara yang berbeda dalam pengimplementasian algoritma ini. Hal ini disebutkan sebagai salah satu kelebihan metode karena secara praktik akan hampir mustahil untuk membuat aplikasi yang sama persis dan dapat mengambil pesan tersembunyi yang disimpan dengan metode ini. Gambar 3.3 merupakan *flowchart* yang menggambarkan pengimplementasian metode *bit-plane-complexity-segmentation* pada penelitian ini.



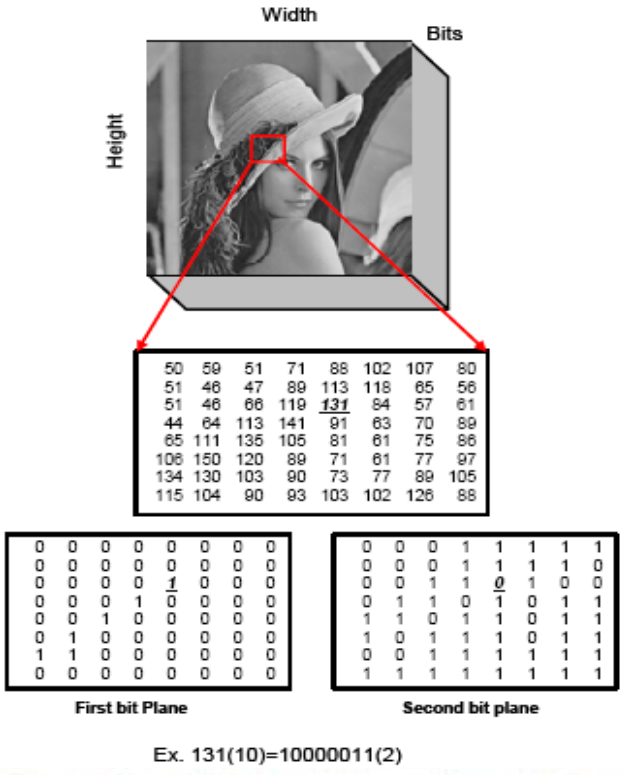
Gambar 3.3 *Flowchart* Enkripsi dan Dekripsi Pesan



Gambar 3.4 *Flowchart* Steganografi *Bit-Plane-Complexity-Segmentation*

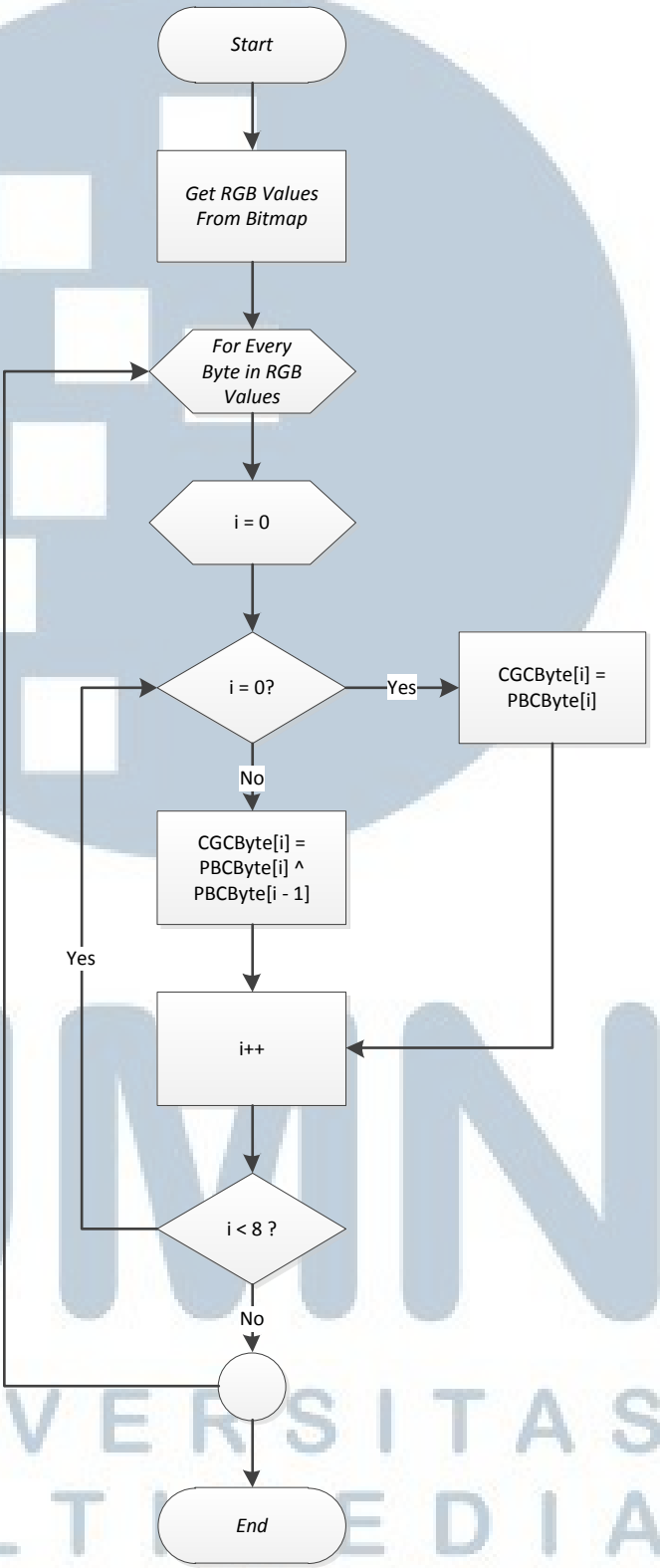
Flowchart pada Gambar 3.3 menunjukkan alur dari metode steganografi *bit-plane-complexity-segmentation*. Sebelum digunakan gambar berektensi bmp yang memiliki sistem PBC diubah menjadi bersistem CGC. Hal ini dilakukan untuk menghindari efek "*Hamming Cliffs*" seperti yang telah dijelaskan pada bab sebelumnya. Gambar 3.5 menunjukkan cara merubah gambar bersistem PBC menjadi bersistem CGC pada penelitian ini.

Gambar yang telah diubah menjadi bersistem CGC kemudian dipecah menjadi 24 gambar biner (*bit plane*) yang merupakan hasil pemotongan dari 8 bit representasi warna merah, 8 bit representasi warna hijau, dan 8 bit representasi warna biru. Gambar 3.5 menunjukkan contoh proses pemecahan gambar.

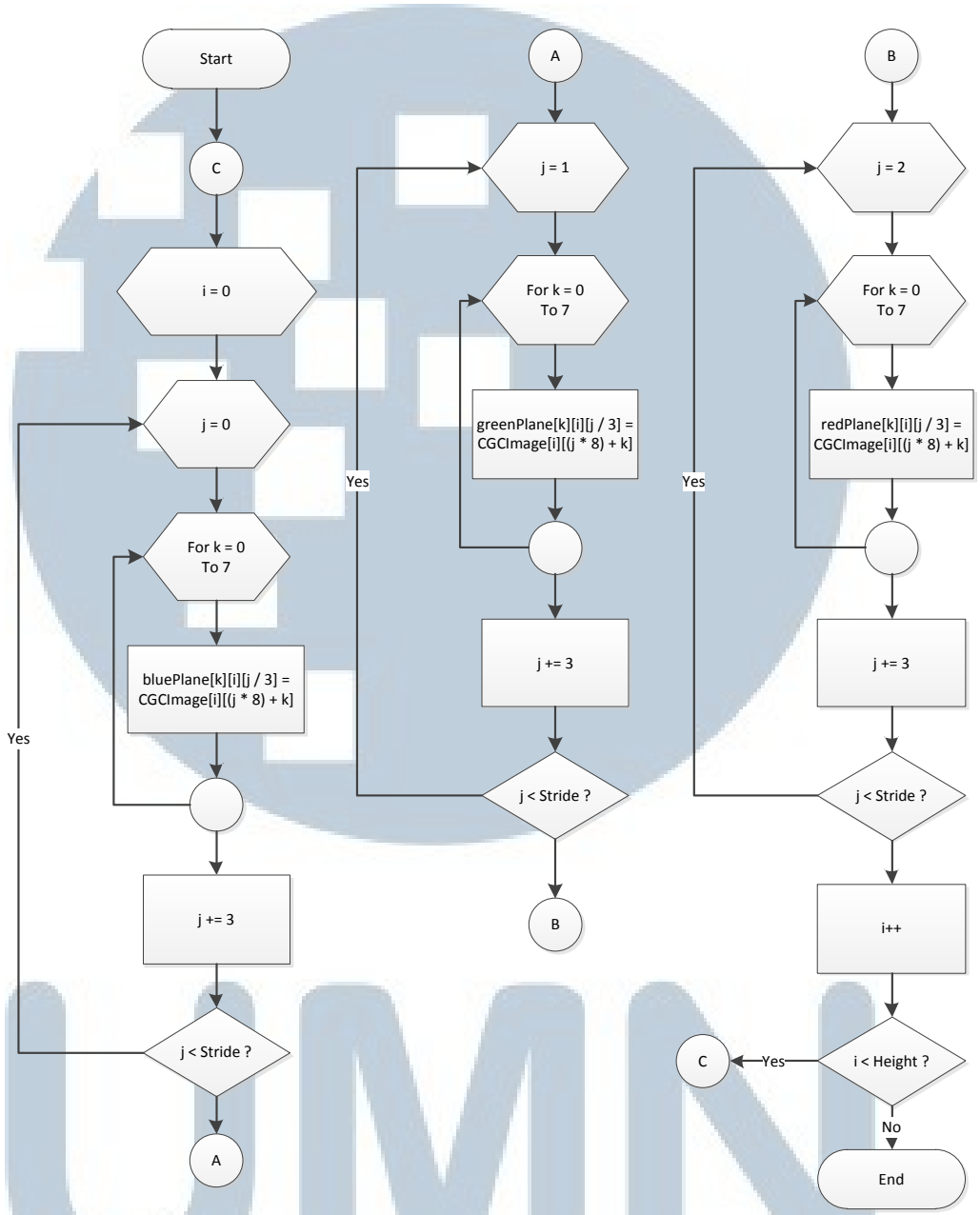


Gambar 3.5 *Bit Slicing*
Sumber: Shrikant S. Khaire & Sanjay L. Nalbalwar, 2010

Setelah didapatkan 24 gambar biner, tiap gambar biner dipecah menjadi *region* yang berukuran 8 x 8 bit. Tiap *region* ini dikelompokkan ke dalam 2 kategori, kompleks dan simpel. Hanya *region* yang termasuk dalam kategori kompleks yang akan disubtitusi dengan pesan. Selain itu juga dihitung banyaknya *region* keseluruhan untuk mengukur ruang yang tersedia untuk menyimpan pesan cukup atau tidak.



Gambar 3.6 ***Flowchart Subroutine Transform Image from PBC to CGC***



Gambar 3.7 *Flowchart Subroutine Slicing CGC Image*

Pengelompokan *region* dalam kategori kompleks didasarkan pada nilai α. Pada penelitian ini, digunakan nilai α = 0,3. Nilai ini merupakan nilai standar yang juga digunakan Eiji Kawaguchi dan Richard O. Eason. Karena *region* berukuran 8 x 8 maka nilai perubahan hitam dan putih maksimal dalam *region* adalah 112. Karena nilai α yang digunakan adalah 0,3 maka,

region akan dikategorikan kompleks jika memiliki perubahan hitam dan putih sebanyak 0,3 x 112 = 33,6 (34). Perlu diketahui bahwa semakin kecil α, maka akan semakin banyak pesan yang bisa ditampung dalam gambar. Akan tetapi, semakin banyak pesan yang disembunyikan, maka kualitas gambar akan semakin menurun. Gambar 3.7 menunjukkan proses pengelompokan *region.*

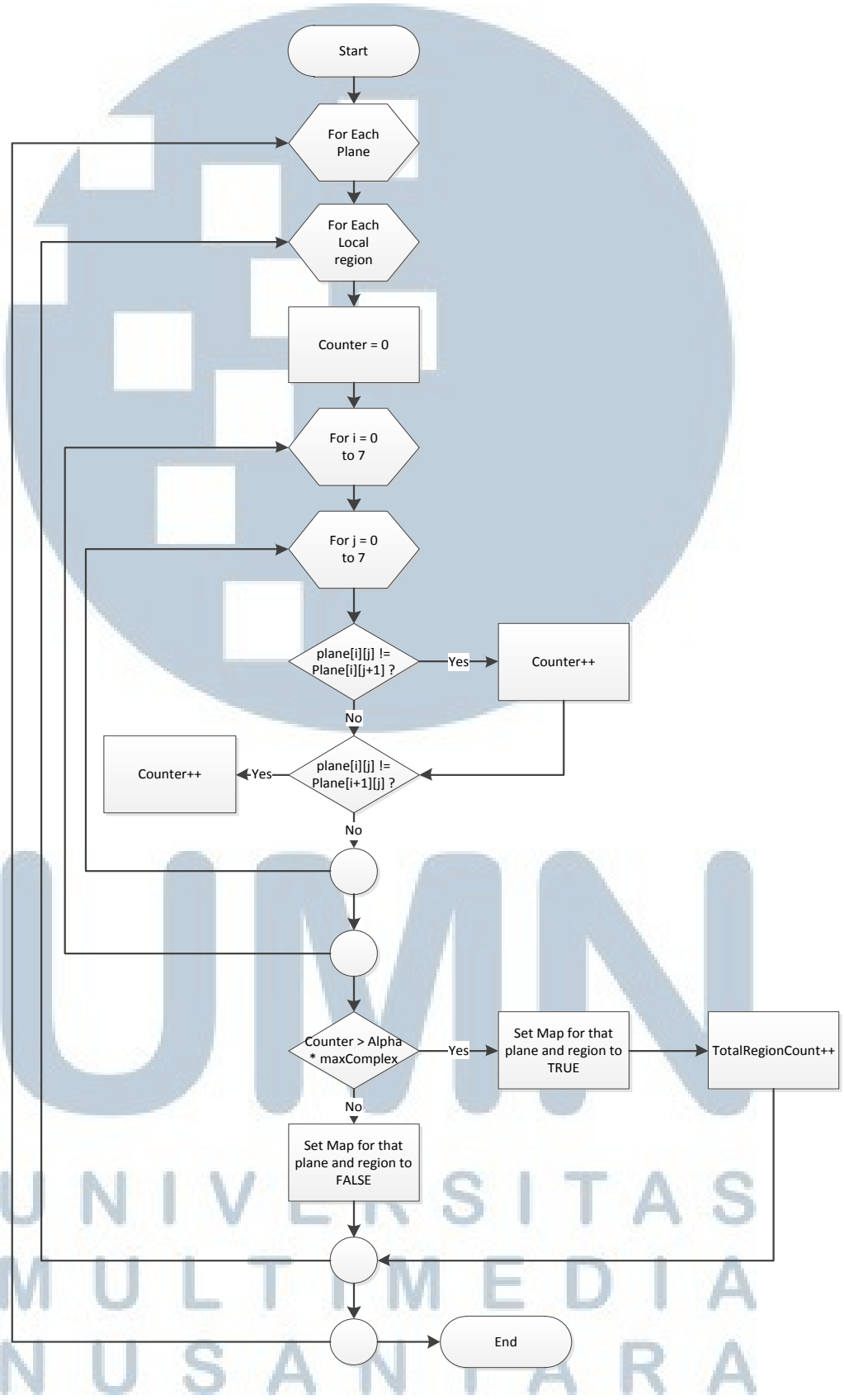
Apabila gambar memiliki ruang yang cukup untuk menyembunyikan pesan, maka pesan yang akan disembunyikan kemudian diubah menjadi *region-region* berukuran 8 x 8 bit (8 byte) untuk disubtitusi dengan *region* pada gambar yang terkategorikan kompleks. Setiap 8 byte pesan yang dirubah menjadi *region* 8 x 8 bit kemudian juga harus diperiksa kompleksitasnya. Metode steganografi ini memanfaatkan kompleksitas gambar sehingga perubahan yang dilakukan tidak terlihat. Namun, tentu saja pesan yang dimasukkan juga harus bersifat kompleks. Jika tidak, maka akan terlihat jelas perubahannya pada gambar. *Region* pesan yang tidak cukup kompleks akan melalui proses konjugasi seperti yang telah dijelaskan pada bab sebelumnya. Setiap konjugasi yang dilakukan dicatat pada suatu *conjugation map* sehingga kita dapat mengetahui pesan mana yang harus dikonjugasi ulang ketika kita mengambil kembali pesan tersembunyi tersebut.

Pada penelitian ini, *Conjugation map* dibuat dengan merepresentasikan setiap *region* pesan dengan 1 bit nilai. Setiap 8 byte pesan, apabila *region* yang terbentuk perlu dikonjugasi, maka akan

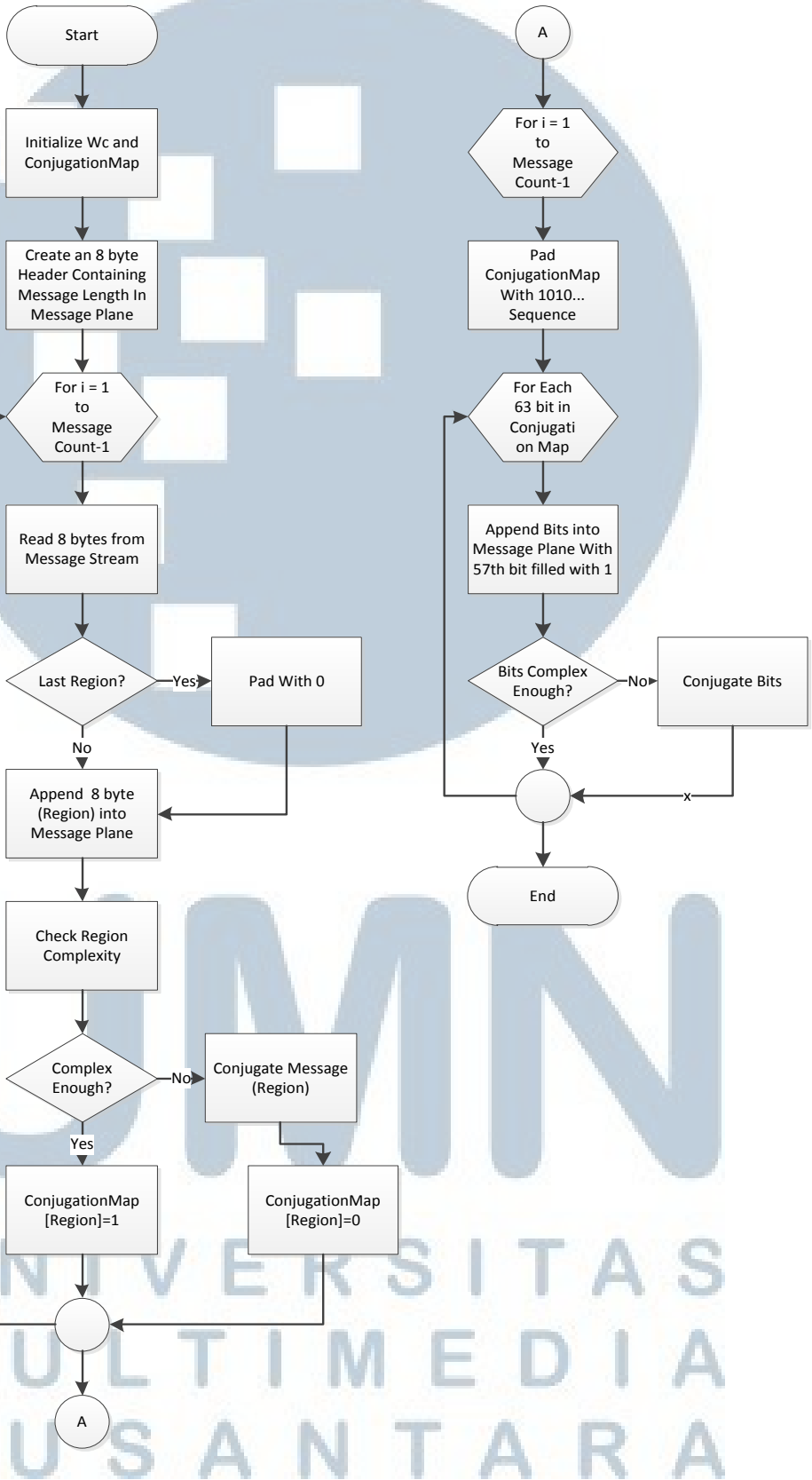
ditambahkan nilai 0 pada *conjugation map.* Sebaliknya jika *region* tidak perlu dikonjugasi maka ditambakan nilai 1 pada *conjugation map.* Karena *conjugation map* itu sendiri harus disembunyikan juga dalam pesan. Maka *conjugation map* tersebut kemudian juga dibentuk menjadi *region-region* 8 x 8 bit dan dikonjugasikan apabila tidak cukup kompleks.

Lalu, bagaimana kita mengetahui *region conjugation map* mana yang dikonjugasi dan yang mana yang tidak? Menjawab hal ini, pada setiap *conjugation map* disisipkan sebuah bit bernilai 1. Apabila region dikonjugasikan maka bit tersebut akan berubah menjadi 0. Oleh karena itu, dapat disimpulkan bahwa pada *region* 64 bit yang merupakan representasi dari *conjugation map*, 63 bit merupakan penanda untuk *region* pesan dan 1 bit berfungsi sebagai penanda untuk *region* itu sendiri. Gambar 3.8 menunjukkan proses perubahan pesan dan *conjugation map.*

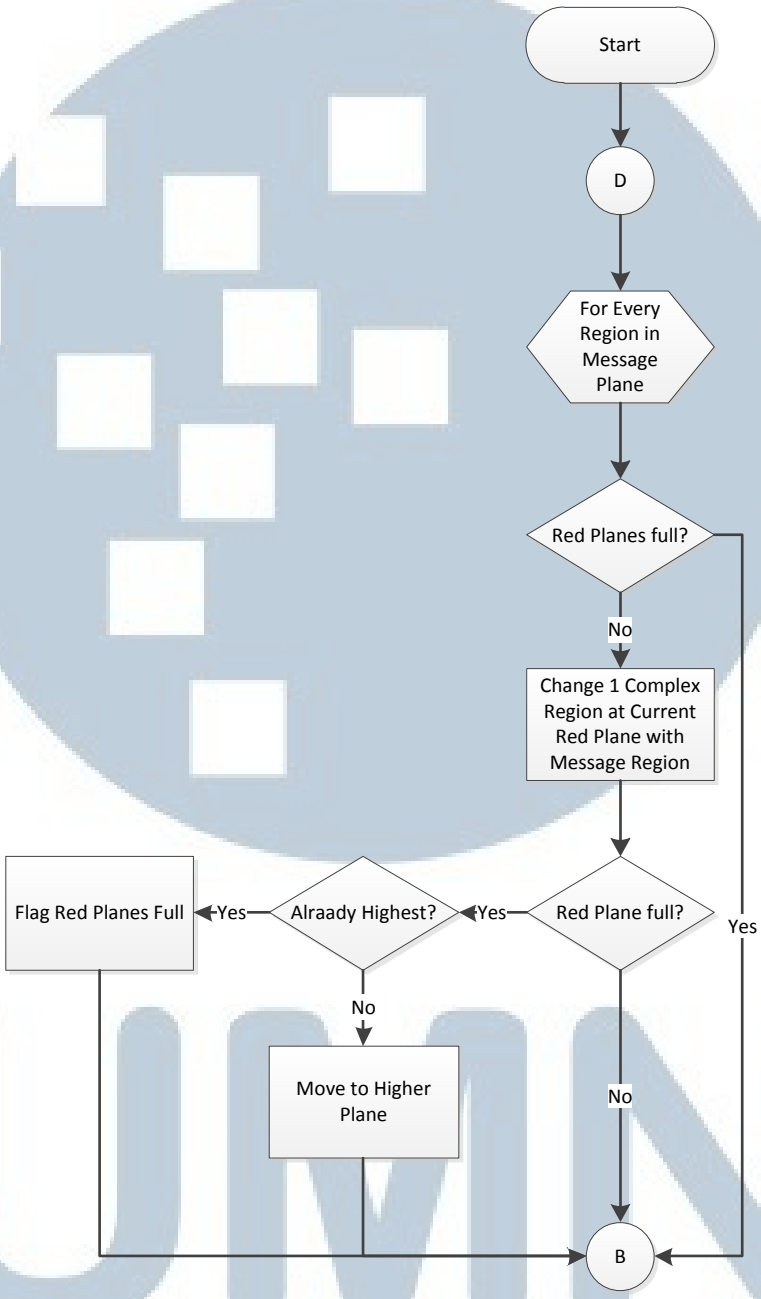
Setelah *region* pesan dan *conjugation map* siap, selanjutnya tinggal mensubtitusi *region* kompleks pada gambar dengan *region* yang telah kita siapkan. Subtitusi *region* pada gambar tidak memiliki pola yang baku. Penelitian ini mensubtitusi pesan secara bergantian mulai dari *plane* merah, hijau, dan biru dan disimpan mulai dari *plane* terendah ke yang tertinggi. Subtitusi dimulai dari *plane* terendah karena pada *plane* tersebut perubahan lebih tidak terlihat dibandingkan *plane* di atasnya. Semakin tinggi tingkat *plane* maka subtitusi akan semakin terlihat.



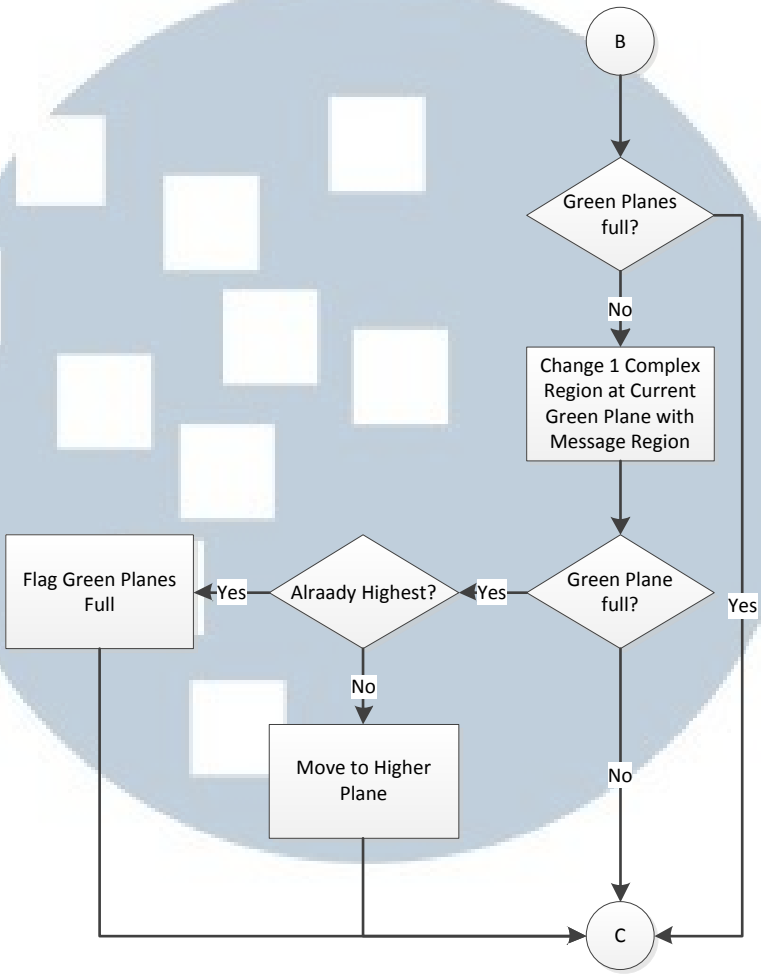
Gambar 3.8 ***Flowchart Subroutine Get Complex Region***



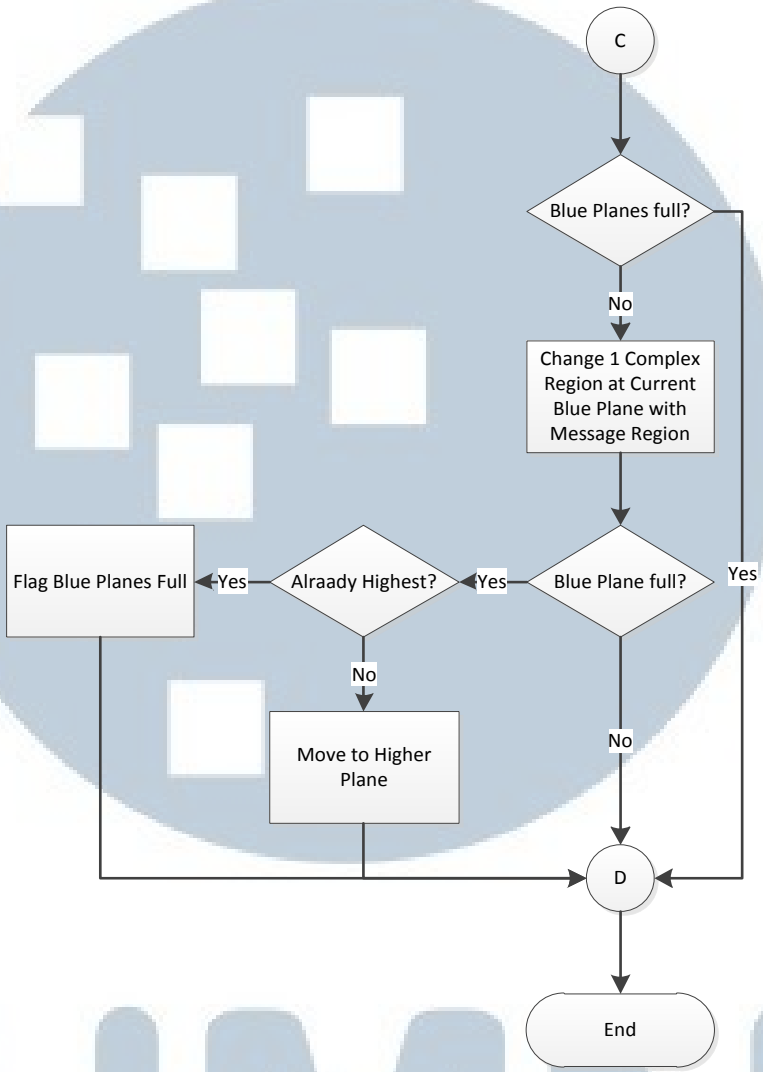
Gambar 3.9 ***Flowchart Subroutine Transform Message to Regions***



Gambar 3.10 *Flowchart Subroutine Insert Message* bagian I

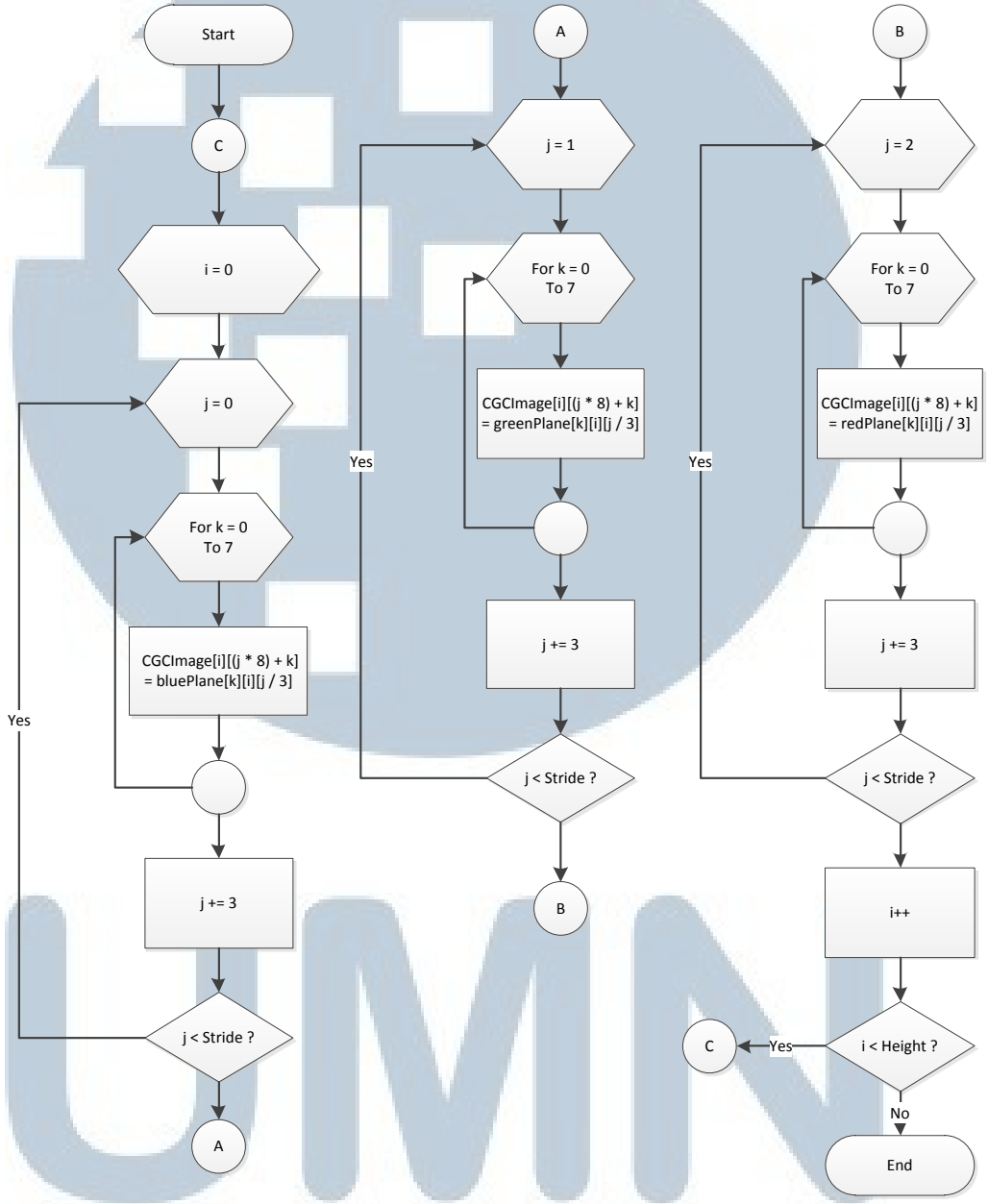


Gambar 3.11 *Flowchart Subroutine Insert Message* bagian II

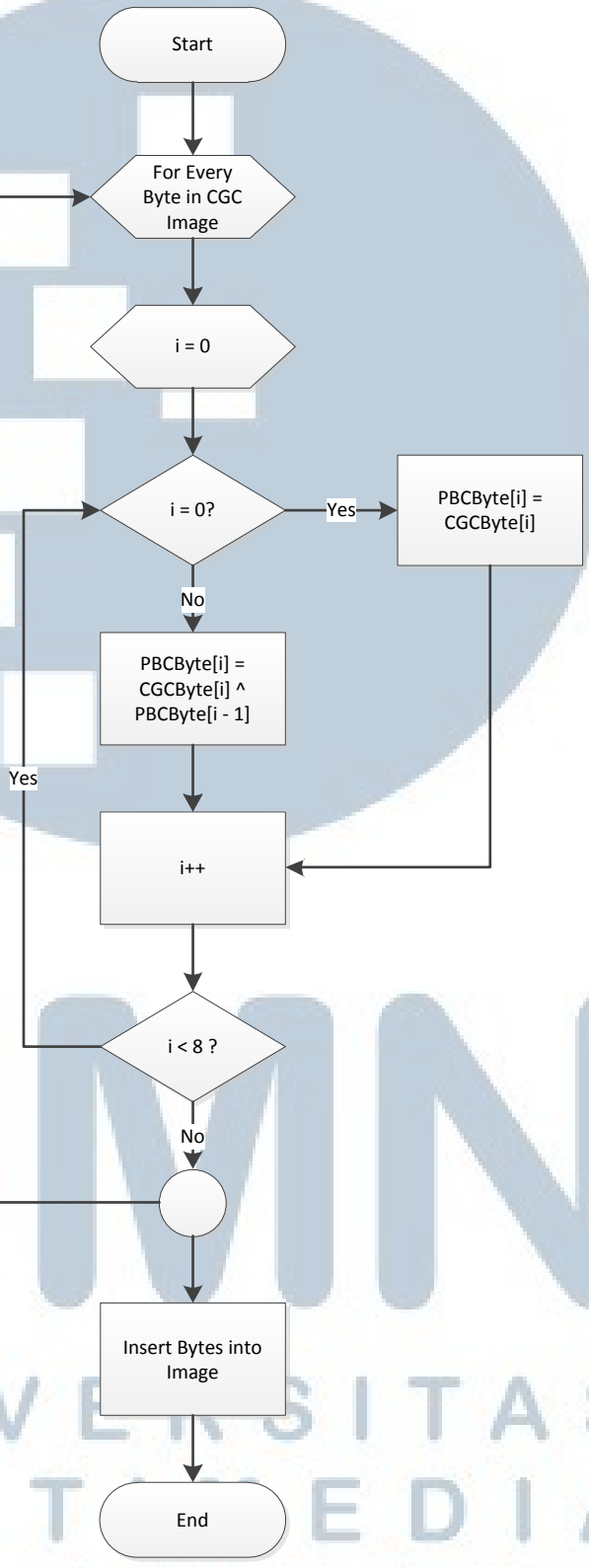


Gambar 3.12 *Flowchart Subroutine Insert Message* bagian III

Gambar 3.9, 3.10 dan 3.11 menunjukkan proses subtitusi *region* pada gambar. Selesai menyubtitusikan *region* pesan dan *conjugation map*, ke 24 *plane* disatukan kembali menjadi gambar dengan sistem CGC dengan proses seperti yang digambarkan pada gambar 3.12. Gambar yang telah disatukan kemudian diubah kembali menjadi sistem PBC dan ditulis menjadi *file* gambar baru. Proses pengembalian gambar menjadi sistem PBC dapat dilihat pada gambar 3.13.



Gambar 3.13 ***Flowchart Subroutine Recompose Image***



Gambar 3.14 ***Flowchart Subroutine Transform Image from CGC to PBC***

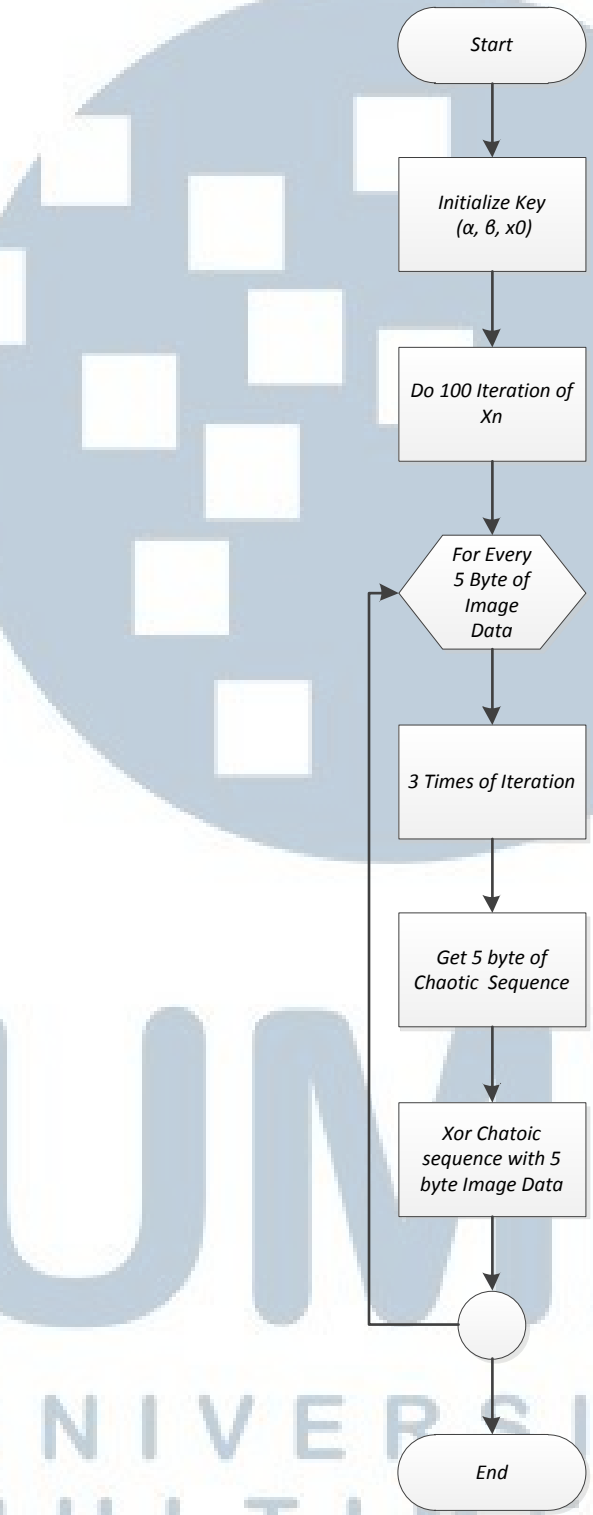
d. Desain Modul Enkripsi Gambar

Modul enkripsi gambar menerima masukan berupa gambar bmp. Modul ini di menggunakan *nonlinear-chaotic-algorithm* sebagai metode enkripsinya. Algoritma ini menggunakan *chaotic sequence* yang dihasilkan dengan menggunakan rumus persamaan *nonlinear*. Gambar 3.14 menjelaskan proses enkripsi yang terjadi pada gambar.

Proses dimulai dengan menginisialisasi *key* yang terdiri dari 3 nilai (α, β, x0). Ketiga nilai tersebut akan berpengaruh sangat signifikan terhadap *sequence* yang akan dihasilkan pada proses. Perubahan yang sangat kecil pada kunci tersebut dapat menghasilkan *sequence* yang sangat jauh berbeda. Ketiga nilai tersebut merupakan *floating-point* sehingga dapat digunakan nilai dengan *digit* yang banyak untuk menambah keamanan.

Secara umum, algoritma ini bekerja dengan menghasilkan *chaos sequence* yang kemudian. Dari *sequence,* diambil 15 angka penting yang dibagi menjadi 5 *integer* yang masing-masing memiliki 3 *digit.* Tiap *integer* kemudian di modulus 256 yang akhirnya menghasilkan 5 byte data. 5 byte data ini lah yang kemudian digunakan pada operasi Xor dengan gambar.

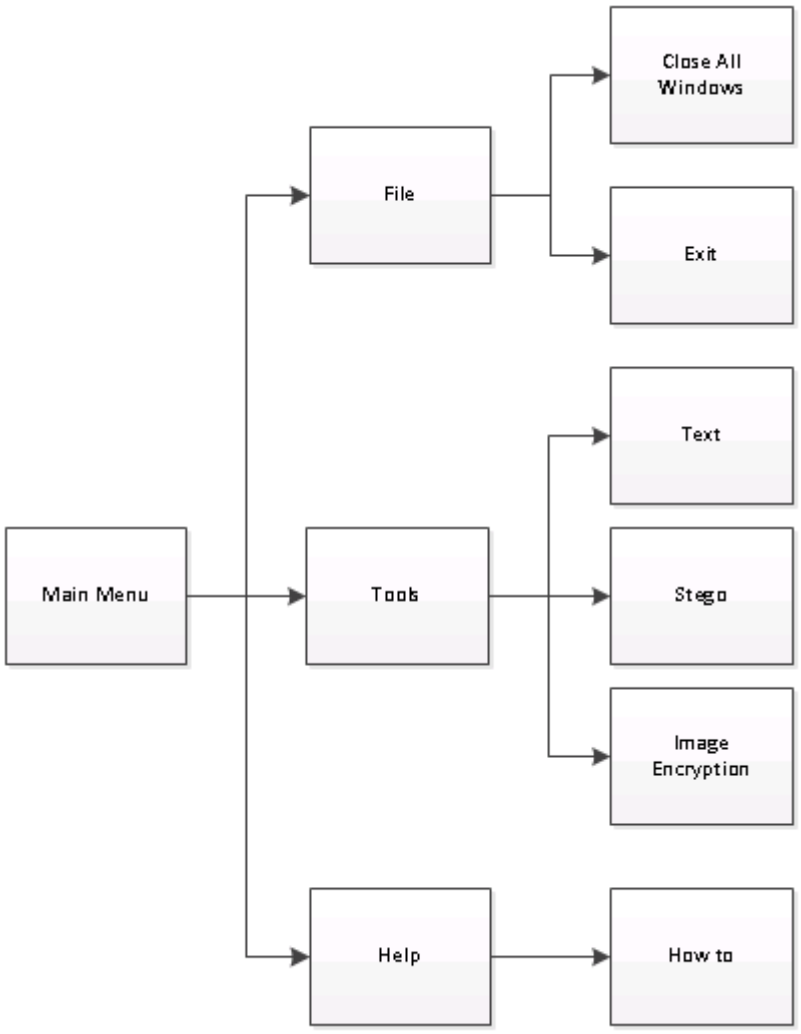
Sequence yang digunakan pada enkripsi ini memiliki selang, hal ini dilakukan untuk membuat relasi antar data menjadi kompleks sehingga hasil enkripsi tidak mudah diserang.



Gambar 3.15 ***Flowchart Nonlinear-Chaotic-Algorithm***

3.2.2 Hirarki Menu

Berikut merupakan hirarki menu dari sistem.



Gambar 3.16 Hirarki Menu

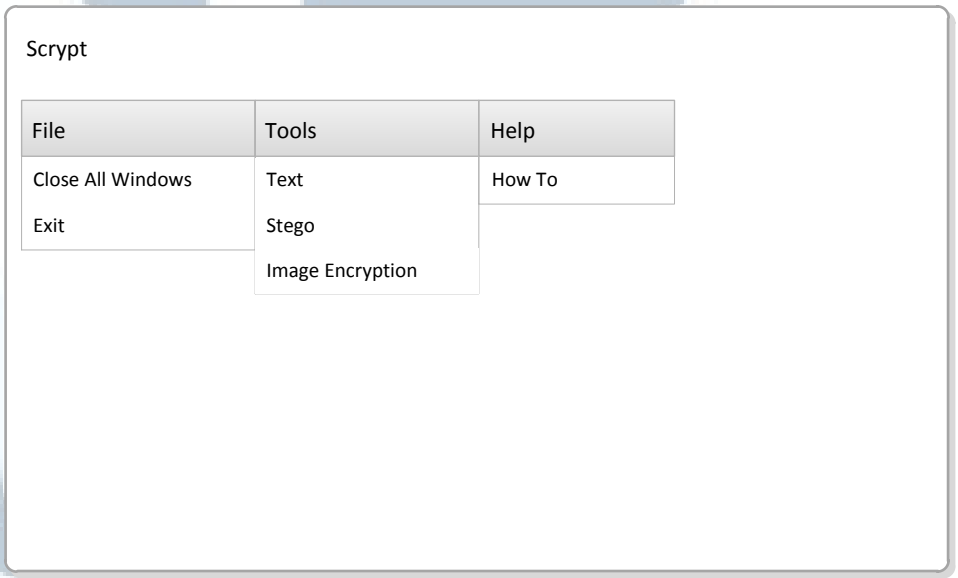
Menu *Text* akan membuka halaman untuk mengkompresi, dekompresi, enkripsi, dan dekripsi pesan. Menu *Stego* akan membuka halaman untuk melakukan steganografi. Menu *Image Encryption* akan membuka halaman untuk melakukan enkripsi dan dekripsi pada gambar. Penggunaan sistem dan keterangan

dapat dilihat pada menu *Help* → *How To*. Untuk keluar dari aplikasi dapat menggunakan menu *Exit* dan untuk menutup halaman-halaman yang aktif dapat menggunakan menu *Close All Windows.*

3.2.3 Tampilan Antarmuka

a. Menu Utama

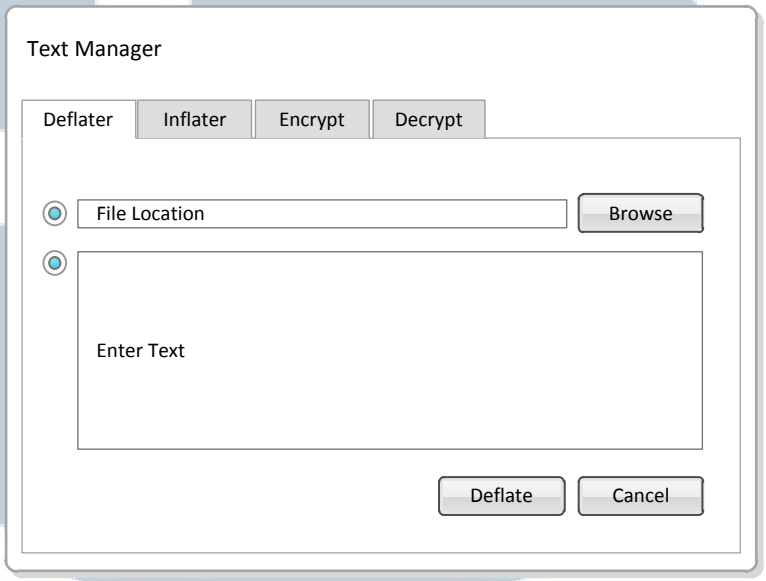
Halaman menu utama merupakan *MDI container*, halaman ini berfungsi untuk menampung halaman modul lainnya. Berikut merupakan rancangan tampilan antarmuka halaman menu utama.



Gambar 3.17 Rancangan Menu Utama

b. Halaman *Text*

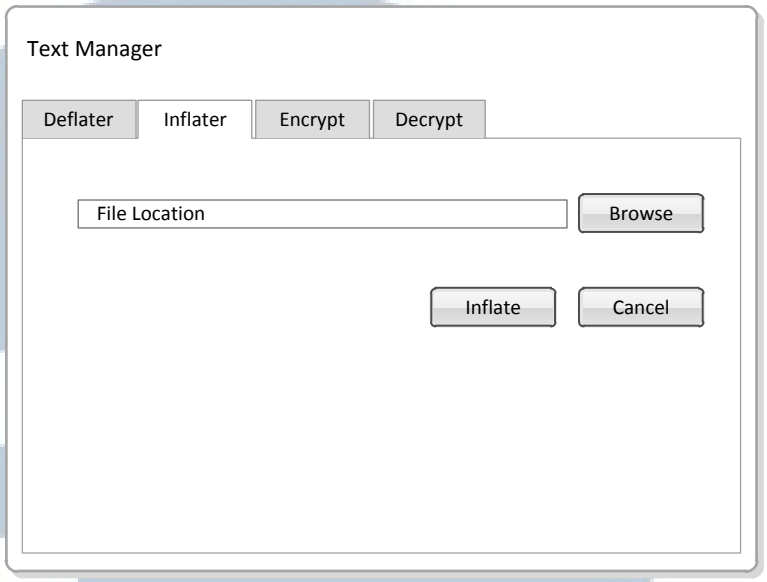
Halaman *Text* berisi modul-modul yang berfungsi untuk mengompresi pesan, dekompresi pesan, enkripsi pesan, dan dekripsi pesan.



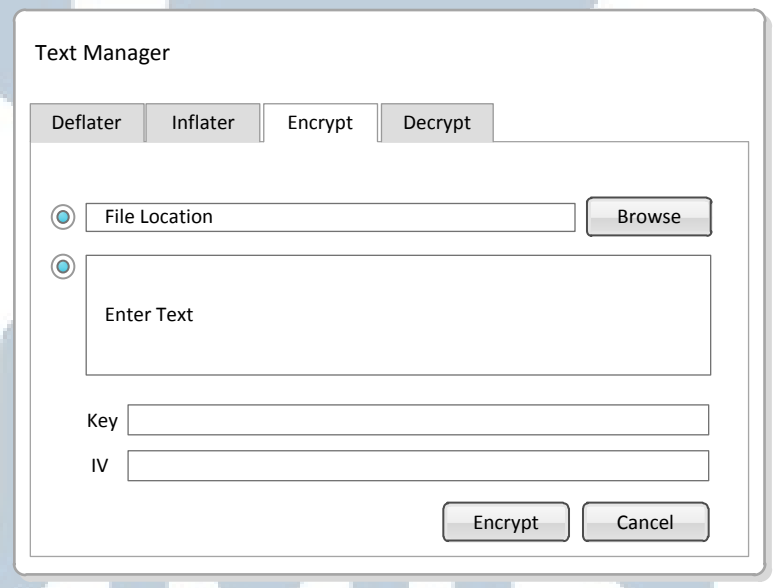
Gambar 3.18 Rancangan Halaman *Text* Bagian *Deflater*

Bagian *deflater* (gambar 3.18) berfungsi untuk mengompres pesan. Pesan dapat berupa *file* teks atau pun *string* yang langsung dimasukkan pada *textbox* yang tersedia. Pengguna hanya bisa memilih salah satu sebagai sumber pesan yang akan dikompresi dan itu dilakukan dengan memilih *radio button* yang tersedia.

Bagian *inflater* (gambar 3.19) berfungsi untuk mendekompresi pesan hasil kompresi bagian *deflater*. Sumber pesan yang akan didekompresi hanya dapat berasal dari *file* teks.

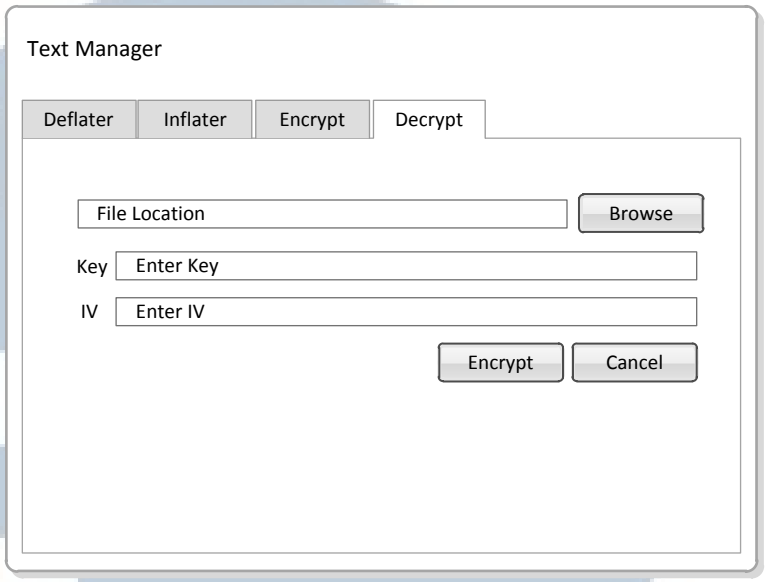


Gambar 3.19 Rancangan Halaman *Text* Bagian In*flater*



Gambar 3.20 Rancangan Halaman *Text* Bagian *Encrypt*

Bagian Encrypt berfungsi untuk melakukan enkripsi pesan. Pengguna dapat memilih sumber dari *file* teks atau memasukkan pesan secara langsung pada *textbox* yang telah disediakan. Setelah enkripsi dilakukan, akan muncul *key* dan IV yang diperlukan untuk melakukan dekripsi.



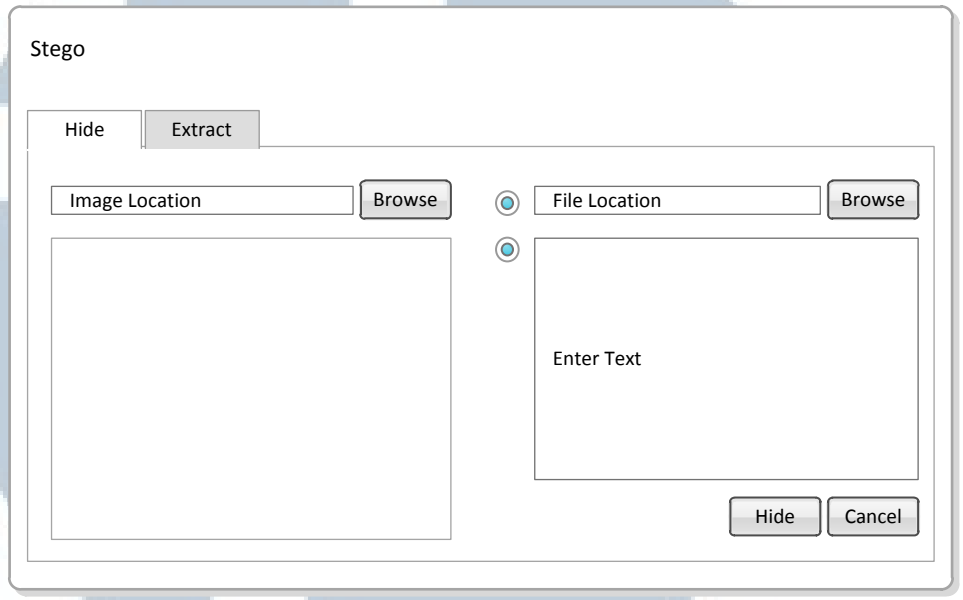
Gambar 3.21 Rancangan Halaman *Text* Bagian *Decrypt*

Bagian *decrypt* berfungsi untuk mendekripsi pesan yang telah dienkripsi pada bagian enkripsi. Untuk melakukan dekripsi pengguna perlu memasukkan lokasi *file* yang akan didekripsi serta *key* dan IV yang didapatkan ketika mengenkripsi *file* sumber.

c. Halaman *Stego*

Halaman *stego* terdiri dari dua bagian, *hide* dan *extract.* Bagian *hide* berfungsi untuk menyembunyikan pesan pada gambar sedangkan bagian *extract* berfungsi untuk mengambil pesan tersembunyi pada gambar. Untuk melakukan penyembunyian pesan, pengguna perlu memilih gambar yang akan dipakai sebagai media penyembunyian serta lokasi *file* atau pesan yang akan disembunyikan. Untuk mengambil kembali pesan yang telah

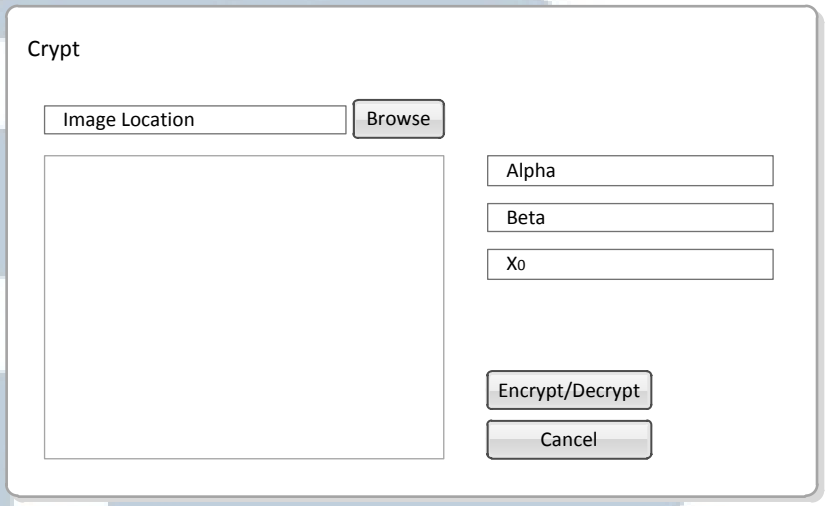
disembunyikan, pengguna cukup memberikan lokasi *file* gambar yang memiliki pesan tersembunyi.



Gambar 3.22 Rancangan Halaman *Stego* Bagian *Hide*

Gambar 3.23 Rancangan Halaman *Stego* Bagian *Extract*

d. Halaman *Image Encryption*

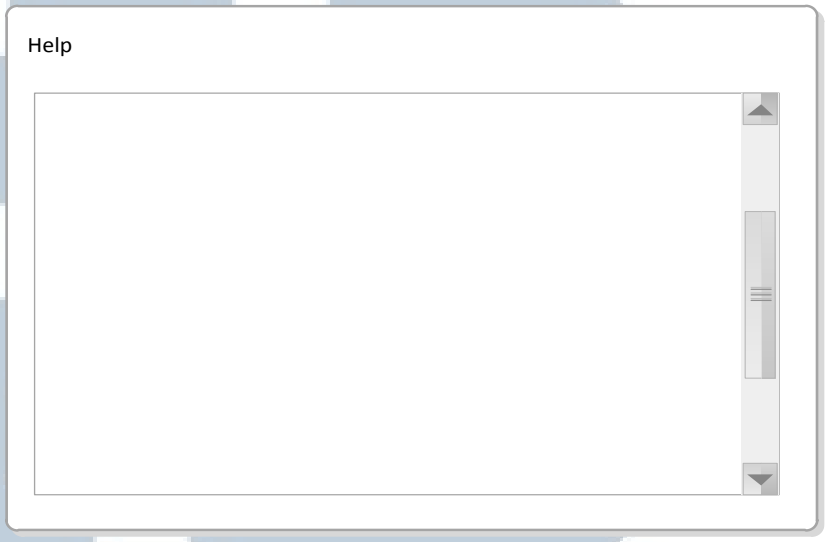


Gambar 3.24 Rancangan Halaman *Image Encryption*

Halaman *image encryption* berfungsi untuk mengenkripsi dan mendekripsi gambar. Pengguna perlu memilih gambar yang akan dienkripsi atau didekripsi dan memasukkan nilai α, β, dan x_0. Ketiga nilai yang dimasukkan dapat berupa *floating-point*.

e. Halaman *How To*

Halaman *how to* berisi cara penggunaan dari aplikasi yang akan dibuat.



Gambar 3.25 Rancangan Halaman *How To*