BAB 2

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Steganografi (*steganography*) adalah teknik menyembunyikan suatu informasi yang rahasia atau sensitif pada suatu media perantara agar tidak terlihat seperti semestinya (*Mangarae, Aelphaeis, Steganography FAQ, Zone-H.Org,* 2006).

Steganografi adalah ilmu dan seni untuk menyembunyikan informasi dengan menyisipkan pesan ke dalam pesan lainnya. Kata *steganography* berasal dari bahasa Yunani, yaitu dari kata *Steganos* yang artinya tersembunyi dan *Graphein* yang artinya tulisan (Sinta Dewi, Agus Urip Ari Wibowo, Heni Rachmawati, Jurnal Teknik Informatika, Vol 1 September 2012)

Bit-plane Complexity Segmentation (*BPCS*) adalah salah satu teknik steganografi yang diperkenalkan oleh Eiji Kawaguchi dan R. O. Eason pada tahun 1997. *BPCS* sendiri memanfaatkan karakteristik penglihatan manusia. Dalam teknik *BPCS*, data tersembunyi dalam *MSB* (*Most Significant Bit*) maupun *LSB (Least Significant Bit)* sehingga penyembunyian pesan menjadi lebih kompleks. Pada saat Eiji Kawaguchi dan R. O. Eason memperkenalkan teknik *BPCS* ini diterapkan pada dokumen citra berwarna yang tidak terkompresi dengan format *bitmap (bmp)*. Dokumen citra tersebut dibagi menjadi beberapa segmen dengan ukuran 8x8 piksel setiap segmennya (Kawaguchi et.al, 1998).

Dari pemaparan toeri diatas penulis ingin mengimplementasikan algoritma *Bit-plane Complexity Segmentation* (*BPCS*) ke dalam sebuah sistem steganografi yang akan dirancang dan dibuat.

2.2 Landasan Teori

2.2.1 *Bit-plane Complexity Segmentation* (*BPCS*)

Bit-plane Complexity Segmentation (*BPCS*) adalah salah satu teknik steganografi yang diperkenalkan oleh Eiji Kawaguchi dan R. O. Eason pada tahun 1997. Teknik *Bit-plane Complexity Segmentation* (*BPCS*) ini merupakan teknik steganografi yang memiliki kapasitas besar, karena dapat menampung data rahasia dengan kapasitas yang relatif besar jika dibandingkan dengan metode steganografi lain seperti *LSB (Least Significant Bit)*. Teknik *BPCS* ini adalah teknik steganografi yang tidak berdasarkan teknik pemrograman, tetapi teknik yang menggunakan sifat penglihatan manusia. Sifat penglihatan manusia yang dimanfaatkan yaitu kelemahan manusia dalam menginterpretasi pola biner yang sangat rumit.

Pada *BPCS* dokumen citra dibagi menjadi segmen-segmen dengan ukuran 8x8 piksel setiap segmennya. Pada dokumen citra 8-*bit*, setiap satu segmen akan memiliki 8 buah *bit-plane* yang merepresentasikan piksel-piksel dari setiap *bit* tersebut. Proses penyisipan data dilakukan pada segmen yang memiliki kompleksitas yang tinggi. Segmen yang memiliki kompleksitas tinggi ini disebut *noise-like regions*. Pada segmen-segmen ini penyisipan dilakukan tidak hanya pada *LSB (Least Significant Bit)*, tapi pada seluruh *bit-plane*.

2.2.2 *PBC* dan *CGC*

Perbedaan antara *CGC (Canonical Gray Code)* dan *PBC (Pure Binary Code)* dapat dilihat pada tabel berikut:

Tabel 2.1 Tabel perbedaan *PBC* dan *CGC*

Desimal	*PBC*	*CGC*
0	0000 0000	0000 0000
1	0000 0001	0000 0001
2	0000 0010	0000 0011
3	0000 0011	0000 0010
4	0000 0100	0000 0110
5	0000 0101	0000 0111
6	0000 0110	0000 0101
7	0000 0111	0000 0100
8	0000 1000	0000 1100
9	0000 1001	0000 1101
10	0000 1010	0000 1111

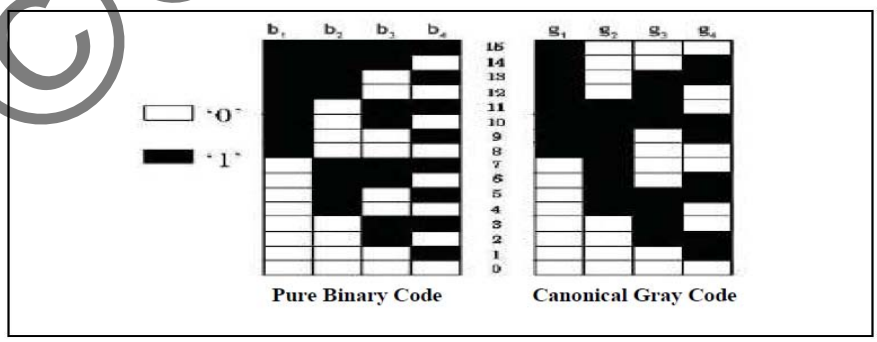
Pada *BPCS*, proses penyisipan dilakukan pada *bit-plane* dengan sistem *CGC (Canonical Gray Code)* karena proses *bit slicing* pada *CGC* cenderung lebih baik dibandingkan pada *PBC* (Kawaguchi et.al, 1998). Sehingga pada proses penyisipan, *bit-plane* dengan representasi *PBC* diubah menjadi *bit-plane* dengan representasi *CGC*.

Tujuan utama dari *BPCS* adalah untuk memanfaatkan sebesar-besarnya ruang pada *image* pembawa tanpa banyak mengurangi kualitas tampilan visual dari gambar aslinya. *Pure Binary Coding* (*PBC*) pada *bit-plane* menyediakan wilayah yang lebih besar untuk *embedding*. Tapi *PBC* mengalami masalah "*Hamming Cliff*", dimana perubahan kecil dalam warna mempengaruhi banyak *bit* dari nilai warna.

Misalkan dalam gambar 8–*bit*, ada dua piksel berturut-turut memiliki nilai intensitas 127 dan 128 masing-masing. Pada *PBC*, 127 direpresentasikan sebagai 01111111 dan 128 direpresentasikan sebagai 10000000. Kedua piksel tampaknya identik dengan mata manusia tetapi sangat berbeda dalam representasi *bit*. Ini disebut konsep "*Hamming Cliff*" (Khaire, Shrikan et. al., 2010). Jika data

rahasia tertanam, maka ada kemungkinan bahwa bisa menjadi 11111111 01111111 dan bisa menjadi 00000000 10000000.

Sebelumnya, ada perbedaan satu tingkat keabuan yang diabaikan oleh mata manusia. Sekarang, setelah *embedding*, perbedaan tingkat abu-abu adalah bahwa dari 255 yaitu satu *pixel* muncul hitam gelap sementara piksel lainnya tampak putih murni. Perubahan ini mudah terlihat oleh mata manusia. Kelemahan ini dihindari oleh *Cannonical Gray Coding* (*CGC*), pada *CGC* teknik pengkodean abu-abu digunakan. Dengan demikian, 127 yang direpresentasikan dalam bentuk biner sebagai 01111111 sekarang direpresentasikan sebagai 01000000 di *CGC*. Demikian pula, 128 direpresentasikan dalam *CGC* sebagai 11000000. Sekarang, dua piksel terlihat sama, tetapi berbeda hanya dengan satu *bit*. Hal ini persis berlawanan dengan *PBC*. Jadi, *CGC* tidak mengalami masalah dari "*Hamming Cliff*". Setelah *embedding*, 01000000 bisa menjadi 11000000 dan 11000000 bisa menjadi 01000000. Jadi, perubahan piksel terjadi dalam tingkat intensitas yang tidak terlalu mencolok. Oleh karena itu, *CGC* lebih baik digunakan dari pada *PBC* di sistem *BPCS* (*Bit-plane Complexity Segmentation*).



Gambar 2.1 Gambar Biner dengan Sistem *PBC* dan *CGC*

Berikut adalah rumus persamaan antara gambar biner *PBC* dan *CGC*.

(dengan $\oplus$ adalah *Exclusive* OR) :

$$g1 = b1 \quad [2.1]$$

$$gi = bi\text{-}1 \oplus bi\ ,\ i > 1 \quad [2.2]$$

$$b1 = g1 \quad [2.3]$$

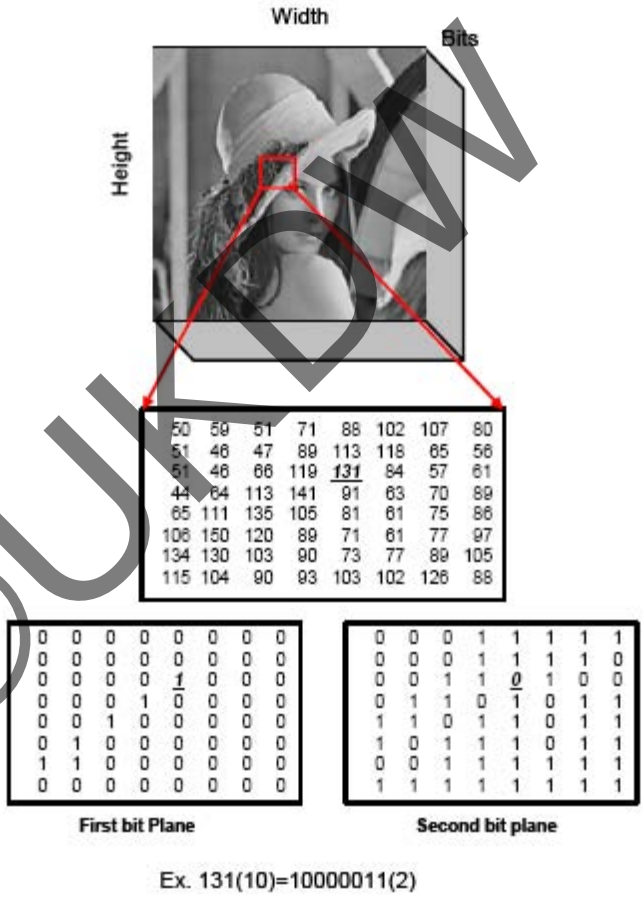
$$bi = gi \oplus bi\text{-}1,\ i > 1 \quad [2.4]$$

dengan,

gi : nilai *bit* ke-i pada sistem *CGC*

bi : nilai *bit* ke-i pada sistem *PBC*

2.2.3 *Bit-plane Slicing*



Gambar 2.2 Konsep pemisahan *bit-plane* dengan nilai piksel 131

Piksel adalah nomor digital yang terdiri dari *bit*. Dalam gambar 8-*bit*, intensitas setiap piksel diwakili oleh 8-*bit*. Gambar 8-*bit* terdiri dari delapan daerah 1-*bit plane*, dari *bit plane* '0 '(*LSB*)

ke *bit-plane* '7' (*MSB*). *Plane* '0' berisi semua *bit* urutan terendah (*LSB*) dari semua piksel dalam gambar sementara *plane* '7' berisi semua *bit* orde tinggi (*MSB*). Pemisahan *bit plane* berguna untuk kompresi gambar. Kompleksitas dari setiap pola *bit-plane* meningkat secara monotical dari *MSB* ke *LSB*.

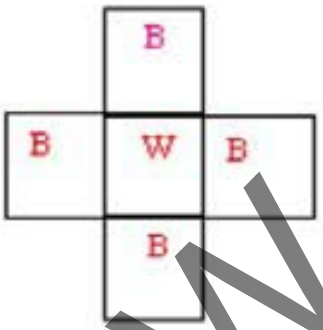
Perhatikan gambar dibawah ini, pada gambar akan di jelaskan bagaimana proses perubahan citra menjadi segmen-segmen *bit-plane*.

Gambar 2.3 Proses perubahan citra menjadi segmen-segmen *bit-plane*.

2.2.4 Kompleksitas Gambar Biner

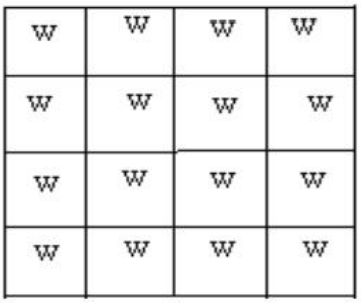
Hal terpenting dalam *BPCS* adalah untuk menemukan daerah "kompleks" di gambar pembawa pesan (*vessel image*) sehingga data dari gambar rahasia dapat disembunyikan tanpa kecurigaan apapun. Tidak ada definisi standar dalam penghitungan nilai kompleksitas. Pada dasarnya ada tiga metode ukuran kompleksitas. Namun, tulisan ini berfokus pada ukuran

kompleksitas berdasarkan panjang perbatasan hitam dan putih dalam gambar biner (black-white border *image* complexity) (Kawaguchi et.al,1998). Total panjang perbatasan hitam dan putih adalah sama dengan penjumlahan dari jumlah perubahan warna di sepanjang baris dan kolom dalam gambar.

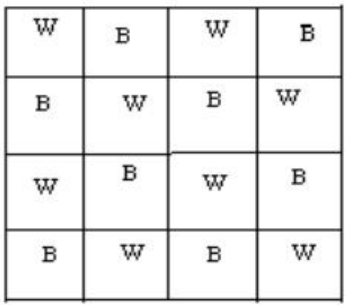


Gambar 2.4 Piksel Putih (W) yang Dikelilingi 4 Piksel Hitam (B)

Pada gambar, sebuah piksel putih tunggal dikelilingi oleh 4 piksel hitam. Jadi ada total 4 perubahan warna dengan panjang perbatasan adalah 4. Perubahan warna yang dihitung adalah berdasarkan posisi atas, kanan, bawah, dan kiri. Secara horizontal dan vertikal, tidak secara diagonal.



Gambar 2.5 Piksel Putih



Gambar 2.6 Hitam Putih

Dalam gambar 2.5 piksel putih. Akibatnya tidak ada perubahan warna di sepanjang baris dan kolom. Oleh karena itu, panjang total perbatasan nol. Jadi panjang perbatasan minimum nol. Pada gambar 2.6 ada alternatif piksel putih dan hitam yaitu gambar papan *checker*. Secara visual, perubahan warna total di sepanjang baris dan kolom adalah 24.

Hal ini dapat juga dihitung dengan bantuan rumus-rumus. Persamaan untuk panjang maksimum perbatasan untuk (2 ^ 2 ^ m x m) citra biner diberikan oleh 2 * 2 ^ m * (2 ^ m-1). Jadi, dalam kasus gambar 2.5 maupun 2.6, adalah gambar biner 4 x 4, sehingga nilai m = 2. Dengan menggunakan rumus, kita mendapatkan panjang maksimum untuk gambar 2.5 dan 2.6 sebagai 24. Kompleksitas gambar dilambangkan dengan 'α' dan diberikan oleh persamaan.

$$\alpha = \frac{k}{2 \times 2^m \times (2^m - 1)} \quad [2.5]$$

Dimana ‘k’ adalah total panjang dari batas pada gambar dan α adalah jangkauan antara 0 dan 1. Dari ke dua gambar di atas tadi, diperoleh nilai α untuk gambar 2.13 adalah α = 0/24 = 0. Dan gambar 2.14 α = 24/24 = 1.

2.2.5 *Informative* dan *Noise-like region*

Informatif *image* berarti gambar yang simpel, sementara *noise-like region* berarti gambar yang kompleks. Hal ini hanya berlaku pada kasus dimana sebuah gambar biner merupakan bagian dari sebuah gambar yang natural. Kompleksitas sebuah area *bit-plane* adalah parameter yang digunakan dalam menentukan sebuah *bit-plane* merupakan *informative* atau *noise-like region*.

Parameter kompleksitas ini dibatasi oleh nilai *threshold* ($\alpha 0$). Sebuah *bit-plane* tergolong sebagai *informative region* apabila memiliki nilai kompleksitas yang lebih kecil dibandingkan *threshold* ($\alpha \leq \alpha 0$) dan sebaliknya akan dianggap sebagai *noise-like region*.

2.2.6 Konjugasi Pada Gambar Biner

Konjugasi dari suatu gambar biner P adalah sebuah gambar biner lainnya yang memiliki nilai kompleksitas sebesar satu dikurangi nilai kompleksitas P. Dari Gambar 1 dapat dilihat bahwa P adalah gambar yang memiliki piksel *background* dengan pola W dan piksel *foreground* dengan pola B. P* yang merupakan konjugasi dari P memiliki spesifikasi sebagai berikut:

1) Memiliki bentuk area *foreground* sama dengan P.
2) Memiliki pola area *foreground* sama dengan pola Bc.
3) Memiliki pola area *background* sama dengan pola Wc.

Untuk membangun sebuah konjugasi P* dari sebuah gambar P, dapat dilakukan dengan rumus berikut, dimana "$\oplus$" menandakan operasi *Exclusive* OR.

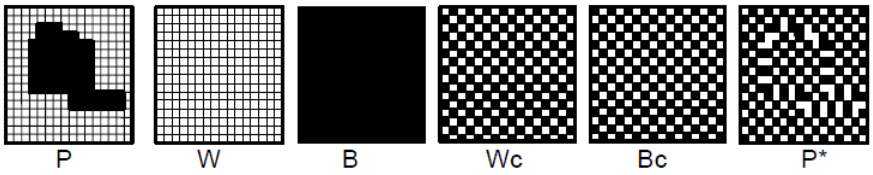
$$P^* = P \oplus Wc \qquad [2.6]$$

$$(P^*)^* = P \qquad [2.7]$$

$$P^* \neq P \qquad [2.8]$$

Jika α(P) adalah kompleksitas dari P, maka:

$\alpha(P^*) = 1 - \alpha(P)$ [2.9]



Gambar 2.7 Konjugasi pada Gambar Biner

2.2.7 Algoritma *Bit-plane Complexity Segmentation (BPCS)*

Bit-plane Complexity Segmentation Steganografi adalah teknik steganografi terbaru yang dapat menyembunyikan informasi dengan kapasitas yang besar. Seperti ditunjukkan pada bagian sebelumnya, penggantian daerah kompleks pada setiap warna gambar *bit-plane* dengan pola biner acak yang tidak terlihat oleh mata manusia.

Dengan strategi *embedding* kita dapat menyembunyikan informasi. Dalam metode ini, kami memanggil gambar *carrier* "*vessel*" atau "*dummy*". Ini adalah gambar berwarna, yang menyembunyikan (*embeds*) informasi rahasia (*file* dalam format apapun). Kami membagi segmen setiap *file* rahasia untuk dimasukkan ke dalam serangkaian blok yang masing-masing memiliki 8 *byte* data .

Blok ini dianggap sebagai pola gambar 8 × 8, biasanya blok ini disebut juga blok rahasia. berikut ini langkah-langkah menanamkan blok rahasia ke dalam gambar "*vessel*". Adapun langkah-langkah penyisipan data pada algoritma *Bit-plane Complexity Segmentation* (*BPCS*) sebagai berikut:

1. Mengubah *cover image* dari sistem *PBC* menjadi sistem *CGC*. Sebelumnya, gambar tersebut di*slice* terlebih dahulu menjadi *bit-plane*. Setiap *bitplane* mewakili *bit* dari setiap piksel.

2. Segmentasi setiap *bit-plane* pada *cover image* menjadi *informative* dan *noise-like region* dengan menggunakan nilai batas/*threshold* (α0).
3. Bagi setiap *byte* pada data rahasia menjadi blok-blok.
4. Jika blok (P) tidak lebih kompleks dibandingkan dengan nilai batas, maka lakukan konjugasi terhadap P untuk mendapatkan P* yang lebih kompleks.
5. Sisipkan setiap blok data rahasia ke *bit-plane* yang merupakan *noise-like region*. Kemudian simpan data konjugasi pada "*conjugation map*".
6. Sisipkan juga pemetaan konjugasi (*conjugation map*) yang telah dibuat.
7. Ubah stego-*image* dari sistem *CGC* menjadi sistem *PBC*.

Proses ekstraksi data rahasia dapat dilakukan dengan menerapkan langkah-langkah penyisipan secara terbalik.

2.2.8 Citra Digital

Citra adalah gambar pada bidang dua dimensi. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Ketika sumber cahaya menerangi objek, objek memantulkan kembali sebagian cahaya tersebut. Pantulan ini ditangkap oleh alat-alat pengindera optik, misalnya mata manusia, kamera, *scanner* dan sebagainya. Bayangan objek tersebut akan terekam sesuai intensitas pantulan cahaya. Ketika alat optik yang merekam pantulan cahaya itu merupakan mesin digital, misalnya kamera digital, maka citra yang dihasilkan merupakan citra digital. Pada citra digital, kontinuitas intensitas cahaya dikuantisasi sesuai resolusi alat perekam.

Menurut Susanto (2011) citra digital merupakan fungsi intensitas cahaya f(x,y), dimana harga x dan y merupakan

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

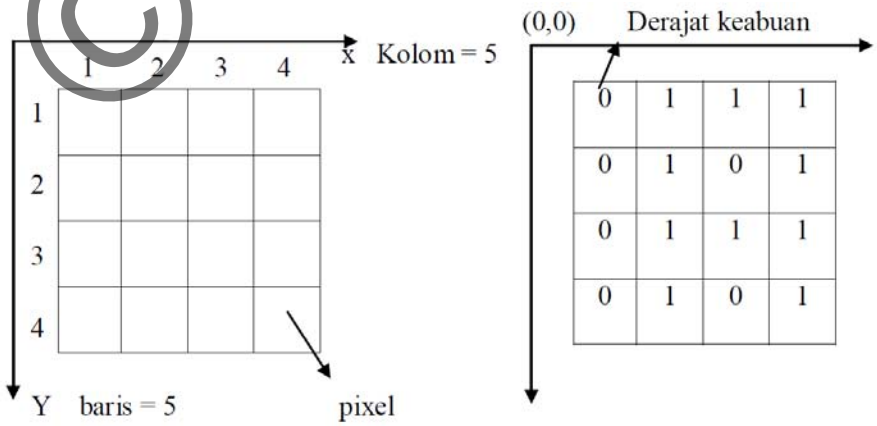
Citra digital merupakan suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar / piksel / *pixel* / *picture element* / pels) menyatakan tingkat keabuan pada titik tersebut. Citra berukuran N x M dinyatakan dengan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

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.8 Matriks Citra Digital M × N

N = Jumlah Baris $0 < y \leq N-1$

M = Jumlah Kolom $0 < x \leq M-1$



Gambar 2.9 Ilustrasi Sistem Koordinasi Piksel

Contoh : $f(2,2) = 1$. Hasil ini bedasarkan hasil koordinat piksel di layar.

Di dalam komputer, citra digital disimpan sebagai suatu *file* dengan format tertentu. Format citra tersebut menunjukan cara sebuah citra digital disimpan, misalnya apakah dengan suatu kompresi atau tidak. Contoh format citra digital adalah *.bmp, .jpg, .png, .tif* dan sebagainya. Ukuran citra digital dinyatakan dalam *pixel* (*picture element*). Umumnya, nilai setiap *pixel* merupakan kuantisasi harga intensitas cahaya. Dengan demikian, suatu citra digital dapat dipandang sebagai sebuah matriks yang elemen-elemennya menunjukkan intensitas cahaya terkuantisasi.

2.2.9 Citra Digital *Bitmap*

Format *file* ini merupakan format grafis yang fleksibel untuk *platform* Windows sehingga dapat dibaca oleh program grafis manapun. Format ini mampu menyimpan informasi dengan kualitas tingkat 1 *bit* samapi 24 *bit*. Kelemahan format *file* ini adalah tidak mampu menyimpan *alpha channel* serta ada kendala dalam pertukaran *platform*. Untuk membuat sebuah objek sebagai *desktop wallpaper*, simpanlah dokumen anda dengan format *file* ini. Akan tetapi format ini memiliki ukuran *file* yang besar dibandingkan dengan format lain. Anda dapat mengkompres format *file* ini dengan kompresi *RLE*. Format *file* ini mampu menyimpan gambar dalam *mode* warna *RGB*, *Grayscale, Indexed Color, dan Bitmap*.

2.2.10 Audio OGG Vorbis (OGG)

Ogg adalah format pemuat berkas video dan audio standar terbuka bebas yang dipelihara oleh *Xiph.Org Foundation.* Para pencipta format *Ogg* mengaku bahwa format ini tidak dibatasi oleh paten perangkat lunak dan dirancang untuk menyediakan *streaming* dan manipulasi yang efisien terhadap multimedia digital bermutu

tinggi. *Ogg* Vorbis adalah format kompresi *audio* baru. Ini kira-kira sebanding dengan format lain yang digunakan untuk menyimpan dan memutar musik digital, seperti MP3, VQF, AAC, dan format audio digital. Hal ini berbeda dengan format lainnya yang mana format ini gratis, terbuka, dan *unpatented*. Nama "*Ogg*" merujuk kepada format berkas yang dapat memultiplekskan sejumlah *codec open source* yang saling mandiri dan terpisah untuk *audio, video, teks* (seperti terjemahan film), dan *metadata*.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Dalam bab ini akan diuraikan mengenai penerapan teori-teori yang ada ke dalam sebuah sistem. Dari sistem yang akan dibuat diharapkan mampu memberikan pemahaman yang lebih baik terkait dengan proses steganografi dengan menggunakan metode *BPCS* (*Bit-Plane Complexity Segmentation*).

3.1 Bahan dan Alat

Hardware dan *software* yang digumakan penulis dalam membangun sistem adalah :

1. *Hardware*
   a. *Prosessor* Intel (R) *Core* (TM) i5-2400, 3.10 GHz, 6 MB *cache*, LGA 1155
   b. Patriot *memory* 4 GB DDR 3
   c. MSI P67A-GD55 *Motherboard*
   d. *Hardisk Western* Digital 1 TB *Green*
   e. Monitor *LED* Philips 19 Inch (resolusi 1366 x 768)

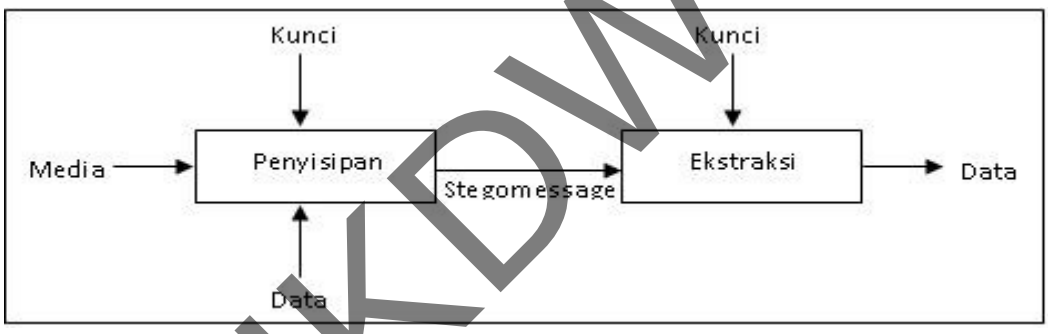
2. *Software*
   a. Sistem Operasi : Windows 7 Professional 64-*bit*
   b. Bahasa Pemrograman : Borland Delphi 7

3.2 Prinsip Kerja Sistem

Secara umum gambaran kerja sistem ialah dengan mengambil data objek *bitmap* (*vessel*), dimana objek *bitmap* ini akan digunakan sebagai

wadah bagi *file* yang akan di sembunyikan. Selanjutnya dengan mengambil *file* data rahasia berupa suara berformat *ogg* (*secret*). Kemudian dilanjutkan dengan proses penyisipan *file secret* tadi ke dalam *file vessel* dengan menggunakan metode *BPCS* (*Bit-Plane Complexity Segmentation*). Setelah melalui proses penyisipan nantinya akan menghasilkan sebuah citra yang sekilas nampak persis mirip dengan citra *vessel*. Namun sesungguhnya *vessel* ini telah memuat sebuah pesan rahasia. Untuk melihat pesan rahasia yang terdapat pada *vessel* harus di *encode* terlebih dahulu.

Dibawah ini merupakan gambaran umum proses steganografy



Gambar 3.1 Proses steganografy secara umum

Dari gambaran umum diatas, dibawah ini akan di jelaskan lebih rinci lagi mengenai proses steganografy dengan metode *BPCS* (*Bit-Plane Complexity Segmentation*).

1. Proses *Encode*
   a. Ambil *file bitmap* (*vessel*).
   b. Konversikan *file bitmap* (*vessel*) kedalam bentuk *layer RGB*.
   c. Konversikan nilai-nilai pada tiap-tiap *layer RGB* menjadi format biner 8 *bit CGC*.
   d. Konversikan nilai-nilai biner 8 *bit CGC* kedalam bentuk CGC (*Canonical Gray Code*).

e. Konversikan bentuk CGC ke bentuk *bit*-plane, urutankan dari *bit-plane* tertinggi ke rendah (*bit-plane* 7 sampai *bit-plane* 0).

f. Hitung nilai kompleksitas tiap-tiap *bit-plane* dengan menggunakan persamaan

$$\alpha = \frac{k}{2 \times 2^m \times (2^m - 1)} \quad [3.1]$$

g. Ambil *file* suara *ogg* (*secret*) yang ingin di sisipkan.

h. Konversikan *file* suara *ogg* menjadi *ASCII*.

i. Konversikan nilai *ASCII file* suara *ogg* menjadi bentuk CGC (*Canonical Gray Code*).

j. Kelompokkan pesan rahasia menjadi rangkaian blok rahasia 8 x 8.

k. Hitung nilai kompleksitas tiap-tiap blok dengan menggunakan persamaan [3.1] pada langkah f diatas

l. Lakukan proses konjugasi pada blok *secret* yang belum kompleks, sehingga dihasilkan blok *secret* yang lebih kompleks. Jangan lupa catatkan tanda pada map konjugasi.

m. Sisipkan rangkaian blok-blok rahasia yang sudah kompleks ke dalam *bit-plane vessel*.

n. Konversikan *bit-plane* hasil penyisipan ke bentuk CGC (*Canonical Gray Code*).

o. Dari bentuk CGC (*Canonical Gray Code*) konversikan menjadi bentuk *CGC*.

p. Gabungkan kembali 8 *bit-plane* untuk tiap-tiap *layer* sehingga didapatkan kembali ketiga buah *layer RGB*

q. Gabungkan ketiga buah *layer RGB* menjadi sebuah gambar (data *bitmap*).

2. Proses *Decode*

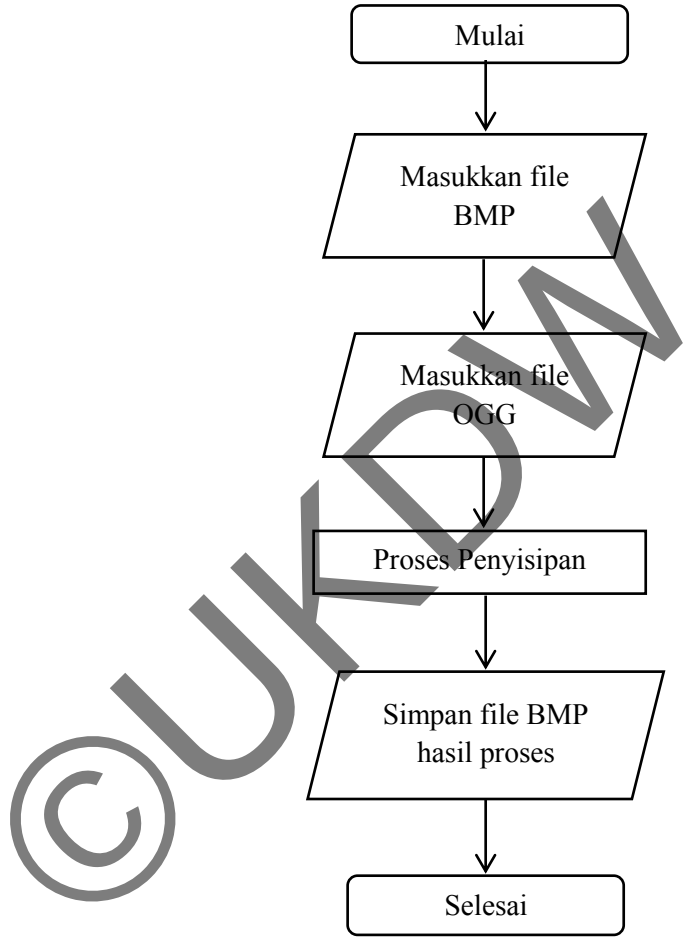
a. Ambil data *bitmap* yang berisi pesan rahasia

b. Konversikan *file* data *bitmap* yang berisi pesan rahasia ke dalam bentuk *layer RGB*
c. Konversikan nilai-nilai pada tiap-tiap *layer RGB* menjadi format biner 8 *bit CGC*.
d. Konversikan nilai-nilai biner 8 *bit CGC* kedalam bentuk CGC (*Canonical Gray Code*).
e. Konversikan bentuk CGC ke bentuk *bit*-plane, urutankan dari *bit-plane* tertinggi ke rendah (*bit-plane* 7 sampai *bit-plane* 0).
f. Lakukan penghitungan nilai kompleksitas tiap-tiap *bit-plane*, jika $\alpha > \alpha$ threshold maka *bit-plane* merupakan pesan rahasia.
g. Lakukan pengecekan proses konjugasi pada *conjugation map* (*bit-plane* ke 0), jika ada lakukan proses konjugasi untuk mendapatkan pesan rahasia.
h. Susun kembali *bit-bit* pada *bit-plane* dalam bentuk blok-blok pesan, penyusunan pesan diurutkan dari *bit-plane* terendah (*bit-plane* 1 - *bit-plane* 7) *bit-plane* 0 tidak digunakan karena merupakan *bit-plane* untuk *conjugation map*.
i. Konversikan kembali *bit-bit* pesan rahasia menjadi suara berformat *ogg*

3.3 *Flowchart* Sistem

3.3.1 *Flowchart* Utama Penyisipan Pesan (*Encode*)

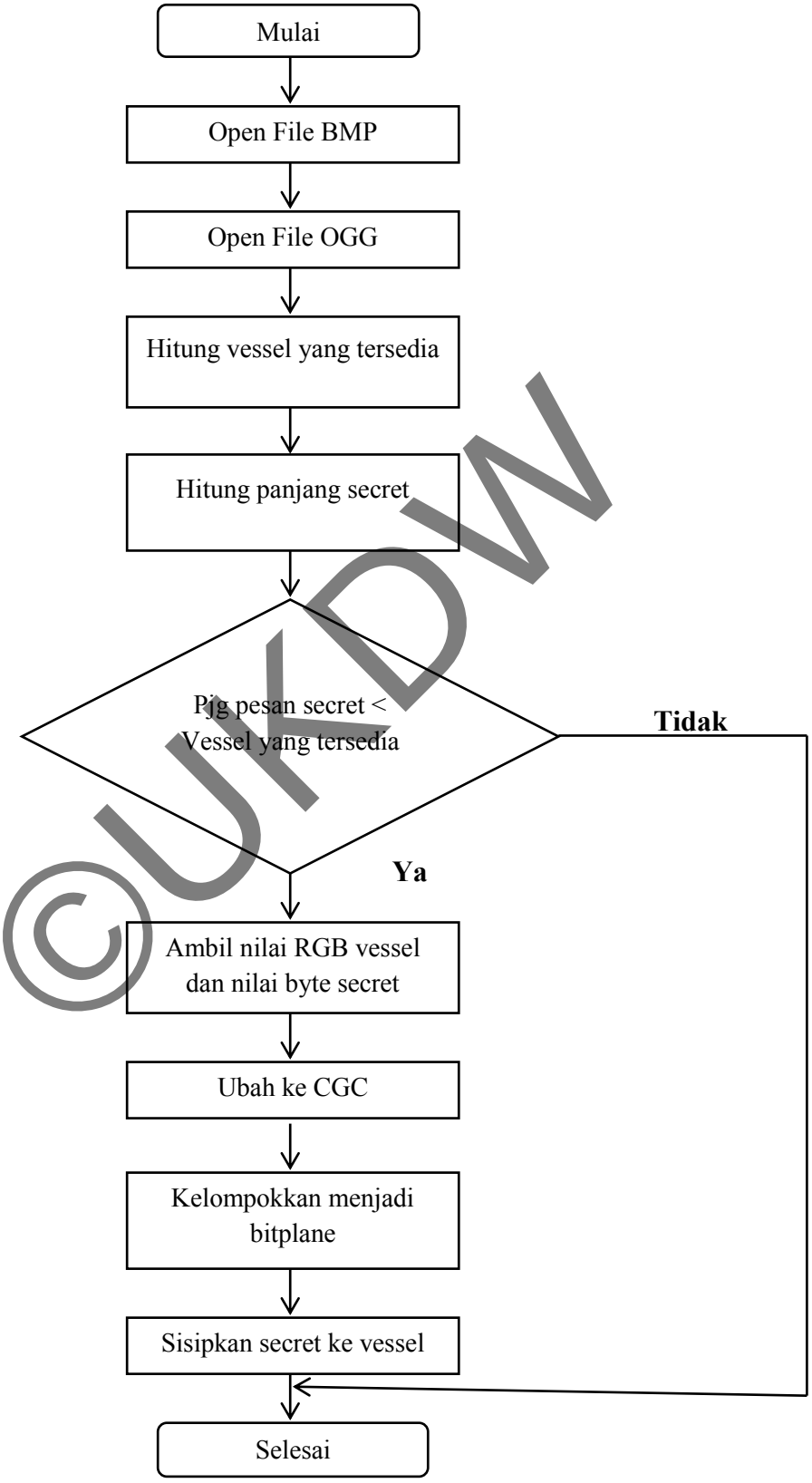
Berikut dibawah merupakan *flowchart* utama penyisipan pesan.



Gambar 3.2 *Flowchart* utama penyisipan pesan

3.3.2 *Flowchart* Proses Penyisipan Pesan (*Encode*)

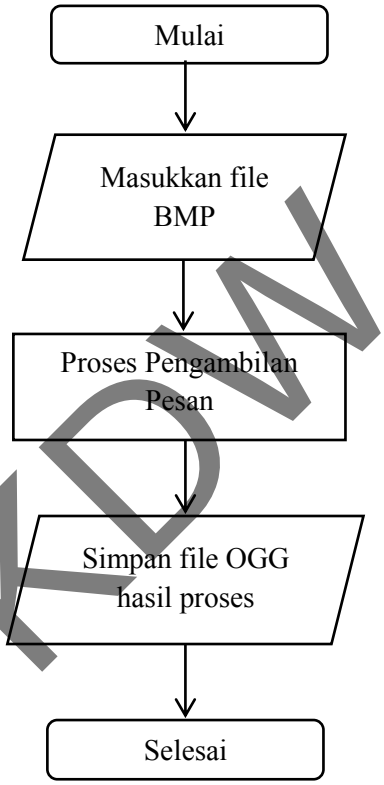
Berikut dibawah merupakan potongan dari proses penyisipan pesan pada *Flowchart* utama diatas.


Mulai
Open File BMP
Open File OGG
Hitung vessel yang tersedia
Hitung panjang secret
Pjg pesan secret < Vessel yang tersedia
Tidak
Ya
Ambil nilai RGB vessel dan nilai byte secret
Ubah ke CGC
Kelompokkan menjadi bitplane
Sisipkan secret ke vessel
Selesai

Gambar 3.3 *Flowchart* proses penyisipan pesan

3.3.3 *Flowchart* Utama Pengambilan Pesan (*Decode*)

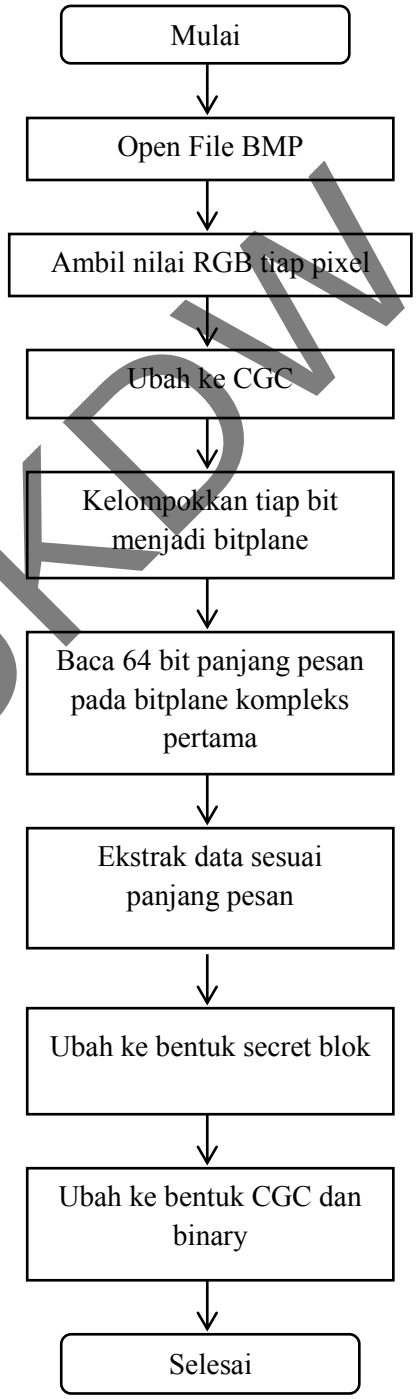
Berikut dibawah merupakan *flowchart* utama pengambilan pesan.



Gambar 3.4 *Flowchart* utama pengambilan pesan

3.3.4 *Flowchart* Proses Pengambilan Pesan (*Decode*)

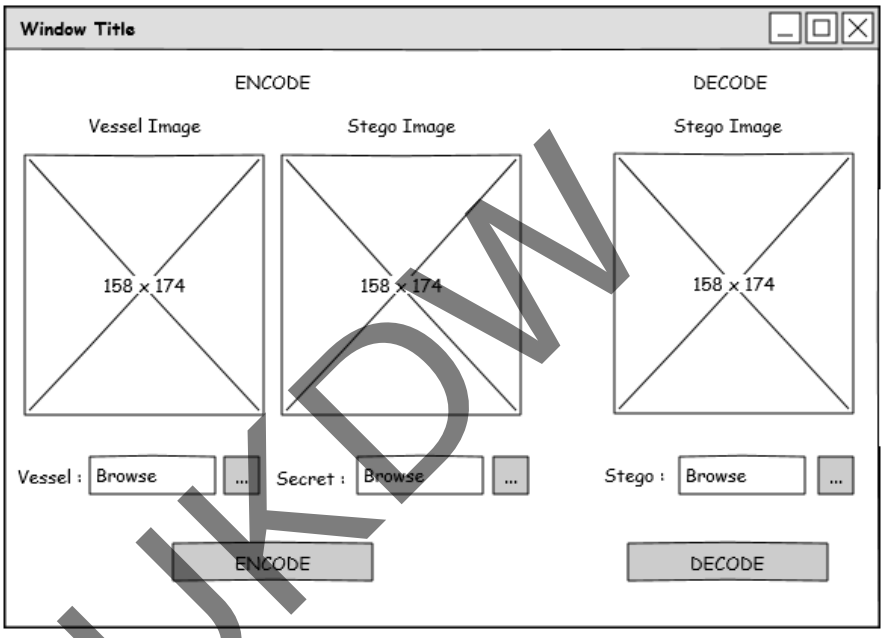
Berikut dibawah merupakan potongan dari proses pengambilan pesan pada *flowchart* utama diatas.



Gambar 3.5 *Flowchart* proses pengambilan pesan

3.4 Perancangan Tampilan Sistem

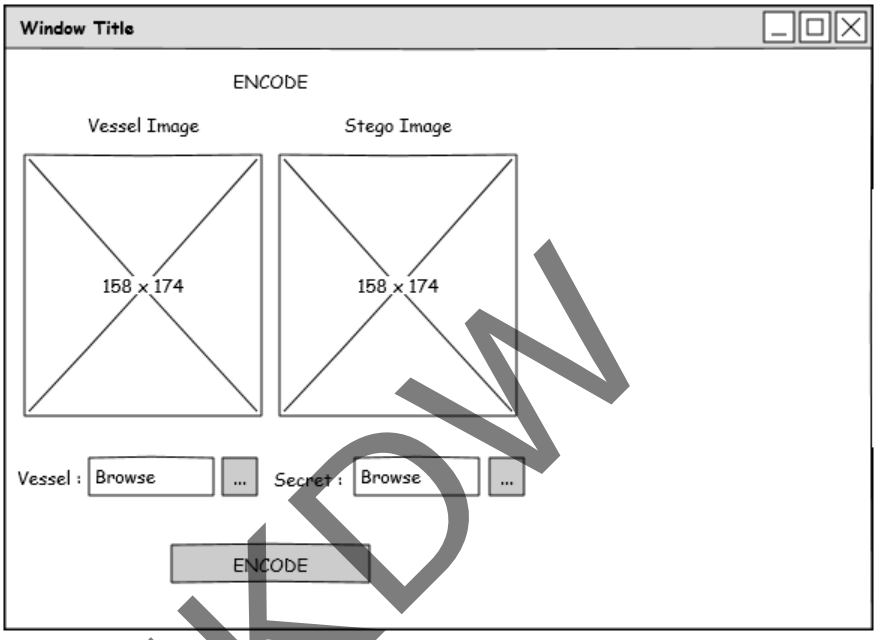
3.4.1 Menu Utama



Gambar 3.6 Tampilan rancangan menu utama

Pada menu utama terdapat dua sisi, yaitu di sebelah kiri dan kanan. Pada sebelah kiri merupakan menu *encode* dan pada bagian kanan merupakan menu *decode*. Menu *encode* berfungsi untuk menyisipkan pesan rahasia suara berformat *ogg* ke dalam *vessel* citra berformat *bitmap*. Sedangkan menu decodo berfungsi untuk mengambil kembali *file* suara berformat *ogg* dari dalam *stego* citra *bitmap*.

3.4.2 Menu *Encode*



Gambar 3.7 Tampilan rancangan menu *encode*

Berikut dibawah ini penjelasan masing-masing fungsi bagian-bagian komponen yang terdapat menu *encode* diatas:

1. Label *encode*

   Berfungsi untuk menandai sisi bagian kiri yang berisi kumpulan komponen yang digunakan untuk melakukan proses *encode*
2. Kotak *preview vessel image*

   Berfungsi sebagai kotak yang akan menampilkan *preview* gambar yang akan kita gunakan sebagai *vessel*.
3. Kotak *preview stego image*

   Berfungsi sebagai kotak yang akan menampilkan *preview* gambar hasil proses *encode* (penyisipan).
4. Kolom *browse vessel*

Berfungsi untuk mencari *file* yang akan digunakan sebagai *vessel* atau wadah penyembunyian pesan. *File* yang dipilih nantinya akan di tampilkan di kotak *preview vessel image*

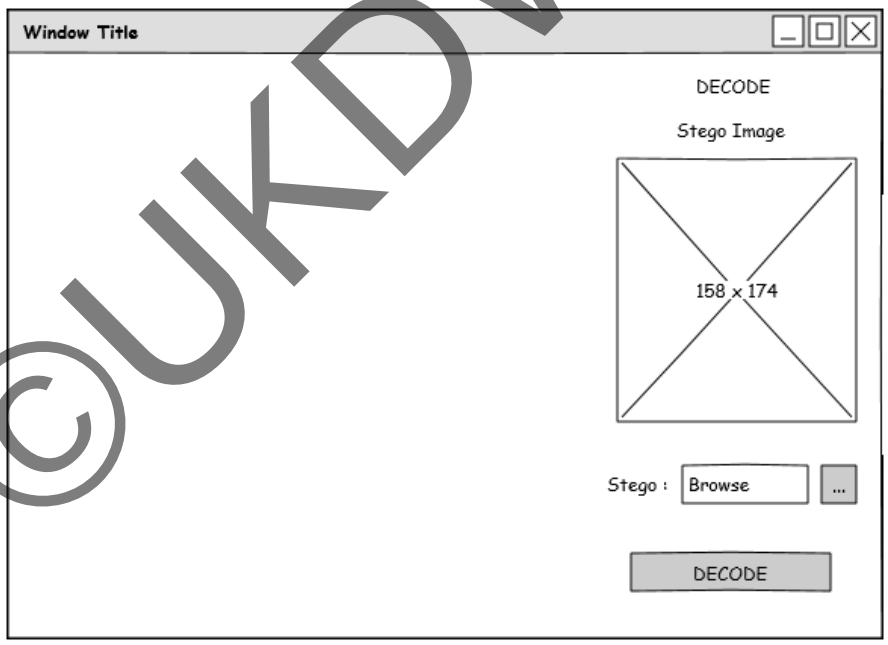
5. Kolom *browse secret*

   Digunakan untuk mencari *file* yang akan digunakan sebagai *secret* atau *file* yang ingin di sembunyikan

6. Tombol *encode*

   Berfungsi untuk melakukan proses penyisipan data *file secret* ke dalam *file vessel*

3.4.3 Menu *Decode*



Gambar 3.8 Tampilan rancangan menu *decode*

1. Label *decode*

   Berfungsi untuk menandai sisi bagian kanan yang berisi kumpulan komponen yang digunakan untuk melakukan proses *decode*.

2. Kotak *preview stego image*

   Berfungsi sebagai kotak yang akan menampilkan *preview* gambar *stego*.

3. Kolom *browse stego*

   Befungsi untuk mencari *file* citra *stego* berformat *bitmap* yang akan kita ekstrak *file ogg* dari dalamnya

4. Tombol *decode*

   Berfungsi untuk mengambil (ekstrak) *file* rahasia yang terdapat didalam *file stego*.

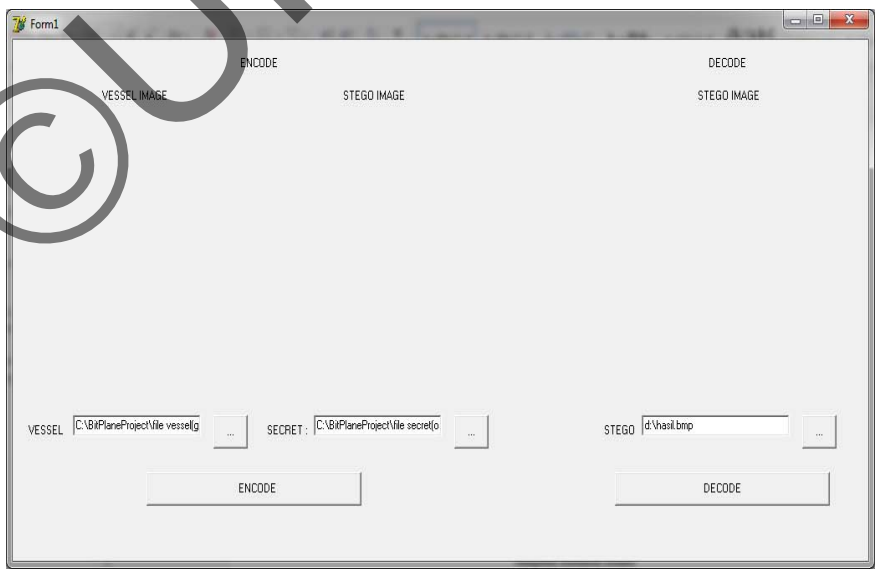
BAB 4

IMPLEMENTASI DAN ANALISIS SISTEM

4.1 Implementasi Sistem

4.1.1 Halaman Utama

Gambar 4.1 merupakan tampilan dari halaman utama sistem steganografi penyembunyian data suara. Pada halaman ini terdapat 2 buah menu, yang pertama yaitu menu *encode* yang berada di sisi kiri. Menu *encode* ini digunakan untuk menyisipkan pesan suara ke dalam gambar. Selanjutnya yang kedua yaitu menu *decode* yang berada di sisi kanan. Menu *decode* ini berfungsi untuk mengekstrak atau mengambil kembali suara yang berada dalam gambar.



Gambar 4.1 Tampilan halaman utama program

4.1.2 Menu *Encode*

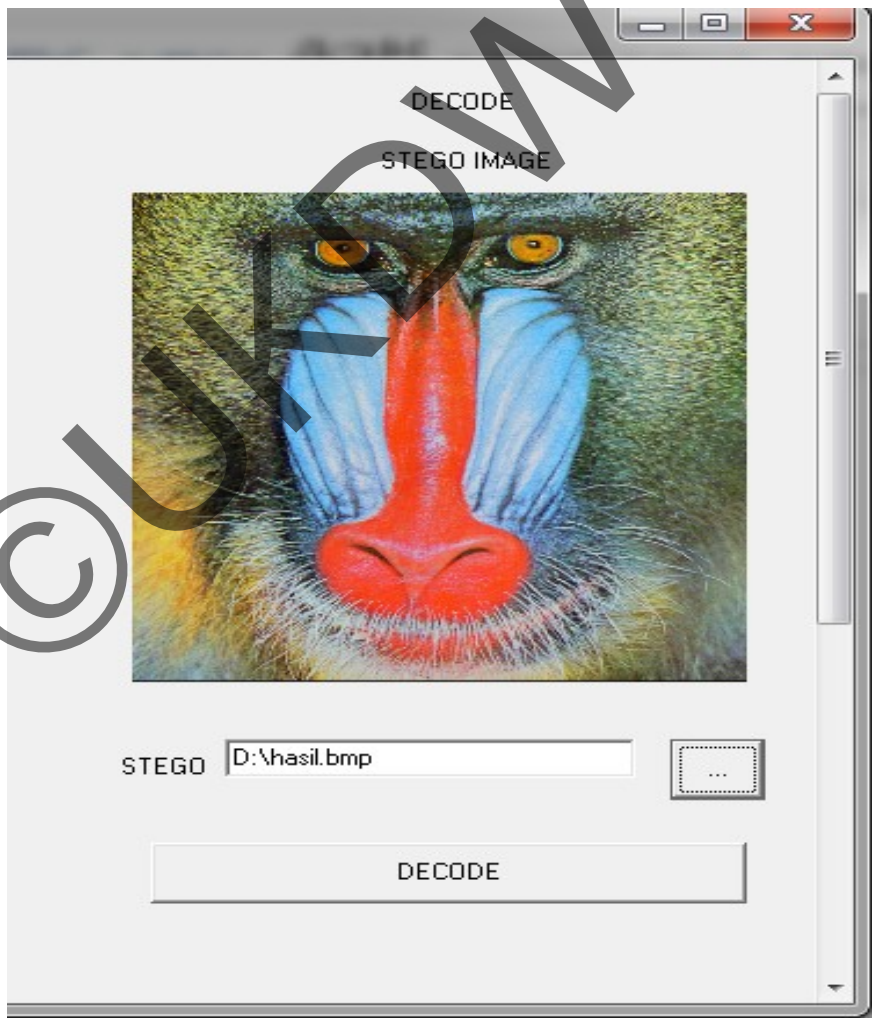
Gambar 4.2 merupakan tampilan menu *encode* yang berada di sisi kiri pada menu utama. Menu *encode* ini merupakan menu yang digunakan untuk meng*encode* atau menyisipkan data suara ke dalam gambar. Pada menu ini terdapat kolom *vessel* yang berfungsi untuk mencari *file vessel* yang akan digunakan sebagai wadah penyembunyian pesan, kemudian di tampilkan di kolom *vessel image*. Selanjutnya kolom *secret* yang berfungsi untuk mencari *file* yang akan digunakan sebagai *secret* atau *file* yang ingin di sembunyikan.

Setelah memilih *file vessel* dan *secret*nya selanjutanya kita dapat menggunakan tombol *encode* yang berada di bagian bawah untuk memulai melakukan penyisipan *file* suara ke dalam *file* gambar. Setelah itu hasilnya yang berupa gambar *stego* di tampilkan di kolom *stego image*.

Gambar 4.2 Tampilan menu *encode* program

4.1.3 Menu *Decode*

Gambar 4.3 merupakan tampilan menu *decode* yang berada di sisi kanan pada menu utama. Menu *decode* ini merupakan menu yang digunakan untuk men*decode* atau mengekstrak data suara yang berada dalam gambar. Pada menu ini terdapat kolom *stego* yang berfungsi untuk mencari *file* yang berisi pesan rahasia yang ingin kita ekstrak pesannya. Setelah memilih *stego image* yang ingin di ekstrak kemudian akan ditampilkan di kolom *stego image*s. Selanjutnya untuk memulai proses ekstraksi dengan mengklik tombol *decode* yang tersedia dibawah.



Gambar 4.3 Tampilan menu *decode* program

4.2 Analisis Sistem

4.2.1 Pengujian pengaruh variasi *vessel* terhadap keberhasilan steganografi dengan metode *BPCS*

Tabel 4.1 Tabel pengujian tabel variasi *vessel* terhadap keberhasilan steganografi dengan metode *BPCS*

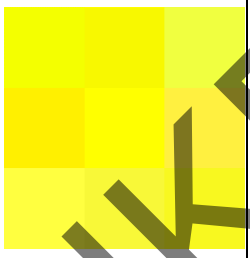
No	Vessel	Secret	Hasil pengujian					
			Stego			Unstego		
			Error	Noise	Ok	Error	Noise	Ok
1	Yellow	Rayman1 sec	✓			✓		
2	Diamond	Rayman1 sec			✓			✓
3	Flower	Rayman1 sec			✓			✓
4	Foto	Rayman1 sec			✓			✓
5	Lamp	Rayman1 sec			✓			✓
6	Line	Rayman1 sec			✓			✓
7	Rain	Rayman1 sec			✓			✓
8	Babon	Rayman1 sec			✓			✓

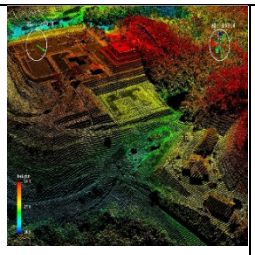
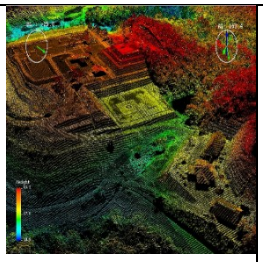
Tabel 4.1 menampilkan hasil pengujian pengaruh variasi vessel terhadap keberhasilan steganografi *BPCS*. Dapat kita lihat pada sampel no 2 sampai no 8 sudah berhasil, baik pada proses penyisipan *secret* kedalam *vessel* (*Stego*) maupun pada saat pengambilan kembali *secret* dari *image stego* (*Unstego*). Sedangkan pada gambar no 1 *error* karena gambar tersebut tidak bisa disisipi data pesan rahasia, hal ini disebabkan oleh rendahnya gradasi perubahan warna pada gambar no 1, sehingga otomatis proses *unstego (decode)* pun tidak bisa dilakukan (*error*).

Faktor suksesnya proses *BPCS* ini terletak pada variasi perbedaan gradasi warna antar piksel yang satu dengan piksel yang

lain disekitarnya. Jadi semakin tinggi perubahan warna yang terdapat pada *vessel* maka semakin besar pula kemungkinan berhasilnya. Untuk lebih jelasnya kita dapat melihat gambar yang terdapat pada tabel 4.2, pada tabel tersebut gambar no 2 sampai no 8 memiliki gradasi warna yang lebih banyak di bandingkan dengan gambar no 1.

Tabel 4.2 Tabel perbandingan gambar *vessel* dengan gambar hasil *stego* dengan mengacu pada tabel 4.1

No	*Vessel*	*Secret*	Hasil *Stego*	Jumlah perbedaan piksel
1	Yellow 	Rayman1sec	Yellow *Stego* ERROR	Tidak Tersedia
2	Diamond 	Rayman1sec	Diamond *Stego* 	22154
3	Flower 	Rayman1sec	Flower *Stego* 	22527
4	Foto	Rayman1sec	Foto *Stego*	

				22302
5	Lamp 	Rayman1sec	Lamp *Stego* 	22251
6	Line 	Rayman1sec	Line *Stego* 	22208
7	Rain 	Rayman1sec	Rain *Stego* 	22225
8	Babon 	Rayman1sec	Babon *Stego* 	22296

Tabel 4.2 menampilkan perbedaan gambar antara gambar asli (*vessel*) dengan gambar yang telah di sisipi (*stego*). Dari

gambar diatas dapat kita lihat secara kasat mata tampak sama persis, sehingga orang lain tidak akan mencurigai gambar tersebut telah berisi pesan rahasia. Hal ini menjadi indikator bahwa proses penyisipan berhasil untuk semua gambar.

Adapun perbedaan jumlah piksel antara gambar vessel dengan gambar *stego* yang cukup banyak namun tampak kedua gambar tersebut sama persis, hal ini mengindikasikan bahwa walaupun terjadi perubahan pada piksel-piksel tersebut tetapi tidak berpengaruh besar terhadap penglihatan manusia. Sehingga orang tidak akan sadar akan adanya pesan rahasia dalam gambar tersebut.

4.2.2 Pengujian pengaruh ukuran *vessel* terhadap keberhasilan steganografi dengan metode *BPCS*

Tabel 4.3 Tabel pengujian pengaruh ukuran *vessel* terhadap keberhasilan steganografi dengan metode *BPCS*

No	Vessel	Ukuran	Secret	Hasil pengujian		
				Stego		
				Error	Noise	Ok
1	Babon	300x300	Rayman1 sec	✓		
2	Babon	400x400	Rayman1 sec		✓	
3	Babon	500x500	Rayman1 sec		✓	
4	Babon	600x600	Rayman1 sec			✓
5	Babon	700x700	Rayman1 sec			✓
6	Babon	800x800	Rayman1 sec			✓
7	Babon	900x900	Rayman1 sec			✓
8	Babon	1000x1000	Rayman1 sec			✓

Tabel 4.3 menampilkan hasil pengujian pengaruh ukuran vessel terhadap keberhasilan steganografi *BPCS*. Dapat kita lihat pada tiap sampel yang mana memiliki ukuran piksel yang berbeda antara satu dengan yang lain. Dengan sampel *vessel* yang sama tetapi memiliki ukuran yang berbeda kita mencoba menyisipkan *file* sampel suara berformat *ogg* yang sama. Hasilnya pada percobaan ke-1 gagal untuk meyisipkan *file* suara berformat *ogg* dengan durasi 1 detik. Hal ini disebabkan karena *vessel* untuk penyisipannya tidak cukup untuk *file* suara *ogg* berdurasi 1 detik ini. Sedikit berbeda dengan percobaan ke-2 dan ke-3 yang masih terdapat *noise* pada hasilnya. Sedangkan pada percobaan ke-4 sampai ke-8 berhasil.

Dari percobaan yang dilakukan dan dengan melihat hasil yang diperoleh maka ukuran *vessel* mempengaruhi keberhasilan *stego*nya. Semakin besar ukuran *vessel,* bisa mempertinggi tingkat keberhasilan *stego*.

4.2.3 Pengujian pengaruh isi pesan (*secret* data) yang disisipkan terhadap keberhasilan steganografi dengan metode *BPCS*

Tabel 4.4 Tabel pengujian pengaruh isi pesan terhadap keberhasilan steganografi dengan metode *BPCS*

No	Vessel	Ukuran	Secret	Hasil pengujian		
				Stego		
				Error	Noise	Ok
1	Babon	800x800	Rayman0.25sec			✓
2	Babon	800x800	Rayman0.5sec			✓
3	Babon	800x800	Rayman1sec			✓
4	Babon	800x800	Rayman1.5sec		✓	
5	Babon	800x800	Rayman2sec		✓	
6	Babon	800x800	Rayman2.5sec		✓	
7	Babon	800x800	Rayman3sec		✓	
8	Babon	800x800	Rayman3.5sec		✓	

Tabel 4.4 menampilkan hasil pengujian pengaruh isi pesan terhadap keberhasilan steganografi *BPCS*. Dapat kita lihat pada percobaan pertama sampai ke tiga berhasil, sedangkan pada percobaan ke empat sampai ke delapan terdapat *noise*. Hal ini menunjukkan bahwa ukuran pesan mempengaruhi keberhasilan *stego*.

LAMPIRAN A

KODE PROGRAM

Deklarasi Variable

```
unit bpcsproject;


interface


uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, ExtDlgs, StdCtrls, ComCtrls, ExtCtrls,math;
type myrecord = record
    desimal :byte;
    biner : string;
    cgc : string;
end;


type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    imagevessel: TImage;
    imagestego: TImage;
    imgTampilVessel: TImage;
    imgtampilstego: TImage;
    Label5: TLabel;
    Label6: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Edit2: TEdit;
    Button2: TButton;
    Button3: TButton;
```

```
    memo1: TRichEdit;
    memo2: TRichEdit;
    memo3: TRichEdit;
    Button5: TButton;
    Edit3: TEdit;
    Button6: TButton;
    ENCODE: TStaticText;
    DECODE: TStaticText;
    OpenDialog1: TOpenDialog;
    OpenDialog2: TOpenDialog;
    OpenPictureDialog1: TOpenPictureDialog;
    imagestegodecode: TImage;
    imgtampilstegodecode: TImage;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);


  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  tabelcode : array[0..255]of myrecord;
  isipesan : array[1..10000000]of byte;
  statusbpcvessel: array[0..7]of boolean;
```

```
  pjgdata : longint;
  pjgpesan :longint;
  BITPLANEvessel: array[1..1000000,0..7,0..7]of string ;
  PERFECTLYCOMPLEXBLOCK:array[0..7]of string[8];
  hasilXORblock:array[0..7]of string[8];
  DATAsecret: array[1..1000000,0..3,0..7]of string ;
  jmliterasi:longint;
  isidatavessel : array[1..3,0..1600,0..1600]of byte;
   currentmatrik:array[1..1600,1..1600]of string[8];
  currentkBitPlaneVessel:array[0..7]of byte;
   currentLBitPlaneVessel:array[0..7]of real;
   currentNoiseRegionVessel:array[1..1000000]of longint;
   currentLBitPlaneSecret:array[0..3]of real;
   currentKBitPlaneSecret:array[0..3]of byte;
   currentSecret:array[0..3]of string[8];
   currentKonjugasiSecret:array[0..7]of byte;
   iiterasi :longint;
   iiterasipesan :longint;
  n:integer; // disini karena ukuran 8x8 maka n selalu 112
  L0:real; // alpha0 merupakan threshold apakah region termasuk
noise atau informatif region
  jmlRegionNoise:byte;
  jmldata , cursordata:longint;
  jmlregionnoisetotal:longint;
```

Fungsi Set Perfectly Complex Block

```
implementation


procedure set_perfectlycomplexblock;
begin
PERFECTLYCOMPLEXBLOCK[0]:='10101010';
PERFECTLYCOMPLEXBLOCK[1]:='01010101';
PERFECTLYCOMPLEXBLOCK[2]:='10101010';
PERFECTLYCOMPLEXBLOCK[3]:='01010101';
PERFECTLYCOMPLEXBLOCK[4]:='10101010';
PERFECTLYCOMPLEXBLOCK[5]:='01010101';
PERFECTLYCOMPLEXBLOCK[6]:='10101010';
PERFECTLYCOMPLEXBLOCK[7]:='01010101';
end;
```

Fungsi Konversi CGC ke desimal

```
function cgctodesimal(nilai:string):integer;
var i:integer;
    hasil:integer;
begin
for i:= 0 to 255 do begin
if tabelcode[i].cgc = nilai then begin
hasil:= tabelcode[i].desimal ;
break;
end;
end;
cgctodesimal := hasil;
end;
```

Fungsi Set CGC_BINARY_CODE

```
PROCEDURE set_CGC_BINARY_CODE;
begin
  // set isi tabel code dengan nilai desimal, biner,dan cgc
  tabelcode[0].desimal := 0;
  tabelcode[0].biner := '00000000';
  tabelcode[0].cgc := '00000000';

  tabelcode[1].desimal := 1;
  tabelcode[1].biner := '00000001';
  tabelcode[1].cgc := '00000001';

  tabelcode[2].desimal := 2;
  tabelcode[2].biner := '00000010';
  tabelcode[2].cgc := '00000011';

  tabelcode[3].desimal := 3;
  tabelcode[3].biner := '00000011';
  tabelcode[3].cgc := '00000010';

  tabelcode[4].desimal := 4;
  tabelcode[4].biner := '00000100';
  tabelcode[4].cgc := '00000110';

  tabelcode[5].desimal := 5;
  tabelcode[5].biner := '00000101';
  tabelcode[5].cgc := '00000111';

  tabelcode[6].desimal := 6;
  tabelcode[6].biner := '00000110';
  tabelcode[6].cgc := '00000101';
```

```
tabelcode[7].desimal := 7;
tabelcode[7].biner := '00000111';
tabelcode[7].cgc := '00000100';

tabelcode[8].desimal := 8;
tabelcode[8].biner := '00001000';
tabelcode[8].cgc := '00001100';

tabelcode[9].desimal := 9;
tabelcode[9].biner := '00001001';
tabelcode[9].cgc := '00001101';

tabelcode[10].desimal := 10;
tabelcode[10].biner := '00001010';
tabelcode[10].cgc := '00001111';

tabelcode[11].desimal := 11;
tabelcode[11].biner := '00001011';
tabelcode[11].cgc := '00001110';

tabelcode[12].desimal := 12;
tabelcode[12].biner := '00001100';
tabelcode[12].cgc := '00001010';

tabelcode[13].desimal := 13;
tabelcode[13].biner := '00001101';
tabelcode[13].cgc := '00001011';

tabelcode[14].desimal := 14;
tabelcode[14].biner := '00001110';
tabelcode[14].cgc := '00001001';
```

```
tabelcode[15].desimal := 15;
tabelcode[15].biner := '00001111';
tabelcode[15].cgc := '00001000';

tabelcode[16].desimal := 16;
tabelcode[16].biner := '00010000';
tabelcode[16].cgc := '00011000';

tabelcode[17].desimal := 17;
tabelcode[17].biner := '00010001';
tabelcode[17].cgc := '00011001';

tabelcode[18].desimal := 18;
tabelcode[18].biner := '00010010';
tabelcode[18].cgc := '00011011';

tabelcode[19].desimal := 19;
tabelcode[19].biner := '00010011';
tabelcode[19].cgc := '00011010';

tabelcode[20].desimal := 20;
tabelcode[20].biner := '00010100';
tabelcode[20].cgc := '00011110';

tabelcode[21].desimal := 21;
tabelcode[21].biner := '00010101';
tabelcode[21].cgc := '00011111';

tabelcode[22].desimal := 22;
tabelcode[22].biner := '00010110';
tabelcode[22].cgc := '00011101';
```

```
tabelcode[23].desimal := 23;
tabelcode[23].biner := '00010111';
tabelcode[23].cgc := '00011100';

tabelcode[24].desimal := 24;
tabelcode[24].biner := '00011000';
tabelcode[24].cgc := '00010100';

tabelcode[25].desimal := 25;
tabelcode[25].biner := '00011001';
tabelcode[25].cgc := '00010101';

tabelcode[26].desimal := 26;
tabelcode[26].biner := '00011010';
tabelcode[26].cgc := '00010111';

tabelcode[27].desimal := 27;
tabelcode[27].biner := '00011011';
tabelcode[27].cgc := '00010110';

tabelcode[28].desimal := 28;
tabelcode[28].biner := '00011100';
tabelcode[28].cgc := '00010010';

tabelcode[29].desimal := 29;
tabelcode[29].biner := '00011101';
tabelcode[29].cgc := '00010011';

tabelcode[30].desimal := 30;
tabelcode[30].biner := '00011110';
tabelcode[30].cgc := '00010001';
```

```
tabelcode[31].desimal := 31;
tabelcode[31].biner := '00011111';
tabelcode[31].cgc := '00010000';

tabelcode[32].desimal := 32;
tabelcode[32].biner := '00100000';
tabelcode[32].cgc := '00110000';

tabelcode[33].desimal := 33;
tabelcode[33].biner := '00100001';
tabelcode[33].cgc := '00110001';

tabelcode[34].desimal := 34;
tabelcode[34].biner := '00100010';
tabelcode[34].cgc := '00110011';

tabelcode[35].desimal := 35;
tabelcode[35].biner := '00100011';
tabelcode[35].cgc := '00110010';

tabelcode[36].desimal := 36;
tabelcode[36].biner := '00100100';
tabelcode[36].cgc := '00110110';

tabelcode[37].desimal := 37;
tabelcode[37].biner := '00100101';
tabelcode[37].cgc := '00110111';

tabelcode[38].desimal := 38;
tabelcode[38].biner := '00100110';
tabelcode[38].cgc := '00110101';
```

```
tabelcode[39].desimal := 39;
tabelcode[39].biner := '00100111';
tabelcode[39].cgc := '00110100';

tabelcode[40].desimal := 40;
tabelcode[40].biner := '00101000';
tabelcode[40].cgc := '00111100';

tabelcode[41].desimal := 41;
tabelcode[41].biner := '00101001';
tabelcode[41].cgc := '00111101';

tabelcode[42].desimal := 42;
tabelcode[42].biner := '00101010';
tabelcode[42].cgc := '00111111';

tabelcode[43].desimal := 43;
tabelcode[43].biner := '00101011';
tabelcode[43].cgc := '00111110';

tabelcode[44].desimal := 44;
tabelcode[44].biner := '00101100';
tabelcode[44].cgc := '00111010';

tabelcode[45].desimal := 45;
tabelcode[45].biner := '00101101';
tabelcode[45].cgc := '00111011';

tabelcode[46].desimal := 46;
tabelcode[46].biner := '00101110';
tabelcode[46].cgc := '00111001';
```

```
tabelcode[47].desimal := 47;
tabelcode[47].biner := '00101111';
tabelcode[47].cgc := '00111000';

tabelcode[48].desimal := 48;
tabelcode[48].biner := '00110000';
tabelcode[48].cgc := '00101000';

tabelcode[49].desimal := 49;
tabelcode[49].biner := '00110001';
tabelcode[49].cgc := '00101001';

tabelcode[50].desimal := 50;
tabelcode[50].biner := '00110010';
tabelcode[50].cgc := '00101011';

tabelcode[51].desimal := 51;
tabelcode[51].biner := '00110011';
tabelcode[51].cgc := '00101010';

tabelcode[52].desimal := 52;
tabelcode[52].biner := '00110100';
tabelcode[52].cgc := '00101110';

tabelcode[53].desimal := 53;
tabelcode[53].biner := '00110101';
tabelcode[53].cgc := '00101111';

tabelcode[54].desimal := 54;
tabelcode[54].biner := '00110110';
tabelcode[54].cgc := '00101101';
```

```
tabelcode[55].desimal := 55;
tabelcode[55].biner := '00110111';
tabelcode[55].cgc := '00101100';

tabelcode[56].desimal := 56;
tabelcode[56].biner := '00111000';
tabelcode[56].cgc := '00100100';

tabelcode[57].desimal := 57;
tabelcode[57].biner := '00111001';
tabelcode[57].cgc := '00100101';

tabelcode[58].desimal := 58;
tabelcode[58].biner := '00111010';
tabelcode[58].cgc := '00100111';

tabelcode[59].desimal := 59;
tabelcode[59].biner := '00111011';
tabelcode[59].cgc := '00100110';

tabelcode[60].desimal :=60;
tabelcode[60].biner := '00111100';
tabelcode[60].cgc := '00100010';

tabelcode[61].desimal :=61;
tabelcode[61].biner := '00111101';
tabelcode[61].cgc := '00100011';

tabelcode[62].desimal :=62;
tabelcode[62].biner := '00111110';
tabelcode[62].cgc := '00100001';
```

```
tabelcode[63].desimal :=63;
tabelcode[63].biner := '00111111';
tabelcode[63].cgc := '00100000';

tabelcode[64].desimal :=64;
tabelcode[64].biner := '01000000';
tabelcode[64].cgc := '01100000';

tabelcode[65].desimal :=65;
tabelcode[65].biner := '01000001';
tabelcode[65].cgc := '01100001';

tabelcode[66].desimal :=66;
tabelcode[66].biner := '01000010';
tabelcode[66].cgc := '01100011';

tabelcode[67].desimal :=67;
tabelcode[67].biner := '01000011';
tabelcode[67].cgc := '01100010';

tabelcode[68].desimal :=68;
tabelcode[68].biner := '01000100';
tabelcode[68].cgc := '01100110';

tabelcode[69].desimal :=69;
tabelcode[69].biner := '01000101';
tabelcode[69].cgc := '01100111';

tabelcode[70].desimal :=70;
tabelcode[70].biner := '01000110';
tabelcode[70].cgc := '01100101';
```

```
tabelcode[71].desimal :=71;
tabelcode[71].biner := '01000111';
tabelcode[71].cgc := '01100100';

tabelcode[72].desimal :=72;
tabelcode[72].biner := '01001000';
tabelcode[72].cgc := '01101100';

tabelcode[73].desimal :=73;
tabelcode[73].biner := '01001001';
tabelcode[73].cgc := '01101101';

tabelcode[74].desimal :=74;
tabelcode[74].biner := '01001010';
tabelcode[74].cgc := '01101111';

tabelcode[75].desimal :=75;
tabelcode[75].biner := '01001011';
tabelcode[75].cgc := '01101110';

tabelcode[76].desimal :=76;
tabelcode[76].biner := '01001100';
tabelcode[76].cgc := '01101010';

tabelcode[77].desimal :=77;
tabelcode[77].biner := '01001101';
tabelcode[77].cgc := '01101011';

tabelcode[78].desimal :=78;
tabelcode[78].biner := '01001110';
tabelcode[78].cgc := '01101001';
```

```
tabelcode[79].desimal :=79;
tabelcode[79].biner := '01001111';
tabelcode[79].cgc := '01101000';

tabelcode[80].desimal :=80;
tabelcode[80].biner := '01010000';
tabelcode[80].cgc := '01111000';

tabelcode[81].desimal :=81;
tabelcode[81].biner := '01010001';
tabelcode[81].cgc := '01111001';

tabelcode[82].desimal :=82;
tabelcode[82].biner := '01010010';
tabelcode[82].cgc := '01111011';

tabelcode[83].desimal :=83;
tabelcode[83].biner := '01010011';
tabelcode[83].cgc := '01111010';

tabelcode[84].desimal :=84;
tabelcode[84].biner := '01010100';
tabelcode[84].cgc := '01111110';

tabelcode[85].desimal :=85;
tabelcode[85].biner := '01010101';
tabelcode[85].cgc := '01111111';

tabelcode[86].desimal :=86;
tabelcode[86].biner := '01010110';
tabelcode[86].cgc := '01111101';
```

```
tabelcode[87].desimal :=87;
tabelcode[87].biner := '01010111';
tabelcode[87].cgc := '01111100';

tabelcode[88].desimal :=88;
tabelcode[88].biner := '01011000';
tabelcode[88].cgc := '01110100';

tabelcode[89].desimal :=89;
tabelcode[89].biner := '01011001';
tabelcode[89].cgc := '01110101';

tabelcode[90].desimal :=90;
tabelcode[90].biner := '01011010';
tabelcode[90].cgc := '01110111';
// sampe sini

tabelcode[91].desimal :=91;
tabelcode[91].biner := '01011011';
tabelcode[91].cgc := '01110110';

tabelcode[92].desimal :=92;
tabelcode[92].biner := '01011100';
tabelcode[92].cgc := '01110010';

tabelcode[93].desimal :=93;
tabelcode[93].biner := '01011101';
tabelcode[93].cgc := '01110011';

tabelcode[94].desimal :=94;
tabelcode[94].biner := '01011110';
```

```
tabelcode[94].cgc := '01110001';

tabelcode[95].desimal :=95;
tabelcode[95].biner := '01011111';
tabelcode[95].cgc := '01110000';

tabelcode[96].desimal :=96;
tabelcode[96].biner := '01100000';
tabelcode[96].cgc := '01010000';

tabelcode[97].desimal :=97;
tabelcode[97].biner := '01100001';
tabelcode[97].cgc := '01010001';

tabelcode[98].desimal :=98;
tabelcode[98].biner := '01100010';
tabelcode[98].cgc := '01010011';

tabelcode[99].desimal :=99;
tabelcode[99].biner := '01100011';
tabelcode[99].cgc := '01010010';

tabelcode[100].desimal :=100;
tabelcode[100].biner := '01100100';
tabelcode[100].cgc := '01010110';

tabelcode[101].desimal :=101;
tabelcode[101].biner := '01100101';
tabelcode[101].cgc := '01010111';

tabelcode[102].desimal :=102;
tabelcode[102].biner := '01100110';
```

```
tabelcode[102].cgc := '01010101';

tabelcode[103].desimal :=103;
tabelcode[103].biner := '01100111';
tabelcode[103].cgc := '01010100';

tabelcode[104].desimal :=104;
tabelcode[104].biner := '01101000';
tabelcode[104].cgc := '01011100';

tabelcode[105].desimal :=105;
tabelcode[105].biner := '01101001';
tabelcode[105].cgc := '01011101';

tabelcode[106].desimal :=106;
tabelcode[106].biner := '01101010';
tabelcode[106].cgc := '01011111';

tabelcode[107].desimal :=107;
tabelcode[107].biner := '01101011';
tabelcode[107].cgc := '01011110';

tabelcode[108].desimal :=108;
tabelcode[108].biner := '01101100';
tabelcode[108].cgc := '01011010';

tabelcode[109].desimal :=109;
tabelcode[109].biner := '01101101';
tabelcode[109].cgc := '01011011';

tabelcode[110].desimal :=110;
tabelcode[110].biner := '01101110';
```

```
tabelcode[110].cgc := '01011001';

tabelcode[111].desimal :=111;
tabelcode[111].biner := '01101111';
tabelcode[111].cgc := '01011000';

tabelcode[112].desimal :=112;
tabelcode[112].biner := '01110000';
tabelcode[112].cgc := '01001000';

tabelcode[113].desimal :=113;
tabelcode[113].biner := '01110001';
tabelcode[113].cgc := '01001001';

tabelcode[114].desimal :=114;
tabelcode[114].biner := '01110010';
tabelcode[114].cgc := '01001011';

tabelcode[115].desimal :=115;
tabelcode[115].biner := '01110011';
tabelcode[115].cgc := '01001010';

tabelcode[116].desimal :=116;
tabelcode[116].biner := '01110100';
tabelcode[116].cgc := '01001110';

tabelcode[117].desimal :=117;
tabelcode[117].biner := '01110101';
tabelcode[117].cgc := '01001111';

tabelcode[118].desimal :=118;
tabelcode[118].biner := '01110110';
```

```
tabelcode[118].cgc := '01001101';

tabelcode[119].desimal :=119;
tabelcode[119].biner := '01110111';
tabelcode[119].cgc := '01001100';

tabelcode[120].desimal :=120;
tabelcode[120].biner := '01111000';
tabelcode[120].cgc := '01000100';

tabelcode[121].desimal :=121;
tabelcode[121].biner := '01111001';
tabelcode[121].cgc := '01000101';

tabelcode[122].desimal :=122;
tabelcode[122].biner := '01111010';
tabelcode[122].cgc := '01000111';

tabelcode[123].desimal :=123;
tabelcode[123].biner := '01111011';
tabelcode[123].cgc := '01000110';

tabelcode[124].desimal :=124;
tabelcode[124].biner := '01111100';
tabelcode[124].cgc := '01000010';

tabelcode[125].desimal :=125;
tabelcode[125].biner := '01111101';
tabelcode[125].cgc := '01000011';

tabelcode[126].desimal :=126;
tabelcode[126].biner := '01111110';
```

```
tabelcode[126].cgc := '01000001';

tabelcode[127].desimal :=127;
tabelcode[127].biner := '01111111';
tabelcode[127].cgc := '01000000';

tabelcode[128].desimal := 128;
tabelcode[128].biner := '10000000';
tabelcode[128].cgc := '11000000';

tabelcode[129].desimal := 129;
tabelcode[129].biner := '10000001';
tabelcode[129].cgc := '11000001';

tabelcode[130].desimal := 130;
tabelcode[130].biner := '10000010';
tabelcode[130].cgc := '11000011';

tabelcode[131].desimal := 131;
tabelcode[131].biner := '10000011';
tabelcode[131].cgc := '11000010';

tabelcode[132].desimal := 132;
tabelcode[132].biner := '10000100';
tabelcode[132].cgc := '11000110';

tabelcode[133].desimal := 133;
tabelcode[133].biner := '10000101';
tabelcode[133].cgc := '11000111';

tabelcode[134].desimal := 134;
tabelcode[134].biner := '10000110';
```

```
tabelcode[134].cgc := '11000101';

tabelcode[135].desimal := 135;
tabelcode[135].biner := '10000111';
tabelcode[135].cgc := '11000100';

tabelcode[136].desimal := 136;
tabelcode[136].biner := '10001000';
tabelcode[136].cgc := '11001100';

tabelcode[137].desimal := 137;
tabelcode[137].biner := '10001001';
tabelcode[137].cgc := '11001101';

tabelcode[138].desimal := 138;
tabelcode[138].biner := '10001010';
tabelcode[138].cgc := '11001111';

tabelcode[139].desimal := 139;
tabelcode[139].biner := '10001011';
tabelcode[139].cgc := '11001110';

tabelcode[140].desimal := 140;
tabelcode[140].biner := '10001100';
tabelcode[140].cgc := '11001010';

tabelcode[141].desimal := 141;
tabelcode[141].biner := '10001101';
tabelcode[141].cgc := '11001011';

tabelcode[142].desimal := 142;
tabelcode[142].biner := '10001110';
```

```
tabelcode[142].cgc := '11001001';

tabelcode[143].desimal := 143;
tabelcode[143].biner := '10001111';
tabelcode[143].cgc := '11001000';

tabelcode[144].desimal := 144;
tabelcode[144].biner := '10010000';
tabelcode[144].cgc := '11011000';

tabelcode[145].desimal := 145;
tabelcode[145].biner := '10010001';
tabelcode[145].cgc := '11011001';

tabelcode[146].desimal := 146;
tabelcode[146].biner := '10010010';
tabelcode[146].cgc := '11011011';

tabelcode[147].desimal := 147;
tabelcode[147].biner := '10010011';
tabelcode[147].cgc := '11011010';

tabelcode[148].desimal := 148;
tabelcode[148].biner := '10010100';
tabelcode[148].cgc := '11011110';

tabelcode[149].desimal := 149;
tabelcode[149].biner := '10010101';
tabelcode[149].cgc := '11011111';

tabelcode[150].desimal := 150;
tabelcode[150].biner := '10010110';
```

```
tabelcode[150].cgc := '11011101';

tabelcode[151].desimal := 151;
tabelcode[151].biner := '10010111';
tabelcode[151].cgc := '11011100';

tabelcode[152].desimal := 152;
tabelcode[152].biner := '10011000';
tabelcode[152].cgc := '11010100';

tabelcode[153].desimal := 153;
tabelcode[153].biner := '10011001';
tabelcode[153].cgc := '11010101';

tabelcode[154].desimal := 154;
tabelcode[154].biner := '10011010';
tabelcode[154].cgc := '11010111';

tabelcode[155].desimal := 155;
tabelcode[155].biner := '10011011';
tabelcode[155].cgc := '11010110';

tabelcode[156].desimal := 156;
tabelcode[156].biner := '10011100';
tabelcode[156].cgc := '11010010';

tabelcode[157].desimal := 157;
tabelcode[157].biner := '10011101';
tabelcode[157].cgc := '11010011';

tabelcode[158].desimal := 158;
tabelcode[158].biner := '10011110';
```

```
tabelcode[158].cgc := '11010001';

tabelcode[159].desimal := 159;
tabelcode[159].biner := '10011111';
tabelcode[159].cgc := '11010000';

tabelcode[160].desimal := 160;
tabelcode[160].biner := '10100000';
tabelcode[160].cgc := '11110000';

tabelcode[161].desimal := 161;
tabelcode[161].biner := '10100001';
tabelcode[161].cgc := '11110001';

tabelcode[162].desimal := 162;
tabelcode[162].biner := '10100010';
tabelcode[162].cgc := '11110011';

tabelcode[163].desimal := 163;
tabelcode[163].biner := '10100011';
tabelcode[163].cgc := '11110010';

tabelcode[164].desimal := 164;
tabelcode[164].biner := '10100100';
tabelcode[164].cgc := '11110110';

tabelcode[165].desimal := 165;
tabelcode[165].biner := '10100101';
tabelcode[165].cgc := '11110111';

tabelcode[166].desimal := 166;
tabelcode[166].biner := '10100110';
```

```
tabelcode[166].cgc := '11110101';

tabelcode[167].desimal := 167;
tabelcode[167].biner := '10100111';
tabelcode[167].cgc := '11110100';

tabelcode[168].desimal := 168;
tabelcode[168].biner := '10101000';
tabelcode[168].cgc := '11111100';

tabelcode[169].desimal := 169;
tabelcode[169].biner := '10101001';
tabelcode[169].cgc := '11111101';

tabelcode[170].desimal := 170;
tabelcode[170].biner := '10101010';
tabelcode[170].cgc := '11111111';

tabelcode[171].desimal := 171;
tabelcode[171].biner := '10101011';
tabelcode[171].cgc := '11111110';

tabelcode[172].desimal := 172;
tabelcode[172].biner := '10101100';
tabelcode[172].cgc := '11111010';

tabelcode[173].desimal := 173;
tabelcode[173].biner := '10101101';
tabelcode[173].cgc := '11111011';

tabelcode[174].desimal := 174;
tabelcode[174].biner := '10101110';
```

```
tabelcode[174].cgc := '11111001';

tabelcode[175].desimal := 175;
tabelcode[175].biner := '10101111';
tabelcode[175].cgc := '11111000';

tabelcode[176].desimal := 176;
tabelcode[176].biner := '10110000';
tabelcode[176].cgc := '11101000';

tabelcode[177].desimal := 177;
tabelcode[177].biner := '10110001';
tabelcode[177].cgc := '11101001';

tabelcode[178].desimal := 178;
tabelcode[178].biner := '10110010';
tabelcode[178].cgc := '11101011';

tabelcode[179].desimal := 179;
tabelcode[179].biner := '10110011';
tabelcode[179].cgc := '11101010';

tabelcode[180].desimal := 180;
tabelcode[180].biner := '10110100';
tabelcode[180].cgc := '11101110';

tabelcode[181].desimal := 181;
tabelcode[181].biner := '10110101';
tabelcode[181].cgc := '11101111';

tabelcode[182].desimal := 182;
tabelcode[182].biner := '10110110';
```

```
tabelcode[182].cgc := '11101101';

tabelcode[183].desimal := 183;
tabelcode[183].biner := '10110111';
tabelcode[183].cgc := '11101100';

tabelcode[184].desimal := 184;
tabelcode[184].biner := '10111000';
tabelcode[184].cgc := '11100100';

tabelcode[185].desimal := 185;
tabelcode[185].biner := '10111001';
tabelcode[185].cgc := '11100101';

tabelcode[186].desimal := 186;
tabelcode[186].biner := '10111010';
tabelcode[186].cgc := '11100111';

tabelcode[187].desimal := 187;
tabelcode[187].biner := '10111011';
tabelcode[187].cgc := '11100110';

tabelcode[188].desimal :=188;
tabelcode[188].biner := '10111100';
tabelcode[188].cgc := '11100010';

tabelcode[189].desimal :=189;
tabelcode[189].biner := '10111101';
tabelcode[189].cgc := '11100011';

tabelcode[190].desimal :=190;
tabelcode[190].biner := '10111110';
```

```
tabelcode[190].cgc := '11100001';

tabelcode[191].desimal :=191;
tabelcode[191].biner := '10111111';
tabelcode[191].cgc := '11100000';

tabelcode[192].desimal :=192;
tabelcode[192].biner := '11000000';
tabelcode[192].cgc := '10100000';

tabelcode[193].desimal :=193;
tabelcode[193].biner := '11000001';
tabelcode[193].cgc := '10100001';

tabelcode[194].desimal :=194;
tabelcode[194].biner := '11000010';
tabelcode[194].cgc := '10100011';

tabelcode[195].desimal :=195;
tabelcode[195].biner := '11000011';
tabelcode[195].cgc := '10100010';

tabelcode[196].desimal :=196;
tabelcode[196].biner := '11000100';
tabelcode[196].cgc := '10100110';

tabelcode[197].desimal :=197;
tabelcode[197].biner := '11000101';
tabelcode[197].cgc := '10100111';

tabelcode[198].desimal :=198;
tabelcode[198].biner := '11000110';
```

```
tabelcode[198].cgc := '10100101';

tabelcode[199].desimal :=199;
tabelcode[199].biner := '11000111';
tabelcode[199].cgc := '10100100';

tabelcode[200].desimal :=200;
tabelcode[200].biner := '11001000';
tabelcode[200].cgc := '10101100';

tabelcode[201].desimal :=201;
tabelcode[201].biner := '11001001';
tabelcode[201].cgc := '10101101';

tabelcode[202].desimal :=202;
tabelcode[202].biner := '11001010';
tabelcode[202].cgc := '10101111';

tabelcode[203].desimal :=203;
tabelcode[203].biner := '11001011';
tabelcode[203].cgc := '10101110';

tabelcode[204].desimal :=204;
tabelcode[204].biner := '11001100';
tabelcode[204].cgc := '10101010';

tabelcode[205].desimal :=205;
tabelcode[205].biner := '11001101';
tabelcode[205].cgc := '10101011';

tabelcode[206].desimal :=206;
tabelcode[206].biner := '11001110';
```

```
tabelcode[206].cgc := '10101001';

tabelcode[207].desimal :=207;
tabelcode[207].biner := '11001111';
tabelcode[207].cgc := '10101000';

tabelcode[208].desimal :=208;
tabelcode[208].biner := '11010000';
tabelcode[208].cgc := '10111000';

tabelcode[209].desimal :=209;
tabelcode[209].biner := '11010001';
tabelcode[209].cgc := '10111001';

tabelcode[210].desimal :=210;
tabelcode[210].biner := '11010010';
tabelcode[210].cgc := '10111011';

tabelcode[211].desimal :=211;
tabelcode[211].biner := '11010011';
tabelcode[211].cgc := '10111010';

tabelcode[212].desimal :=212;
tabelcode[212].biner := '11010100';
tabelcode[212].cgc := '10111110';

tabelcode[213].desimal :=213;
tabelcode[213].biner := '11010101';
tabelcode[213].cgc := '10111111';

tabelcode[214].desimal :=214;
tabelcode[214].biner := '11010110';
```

```
tabelcode[214].cgc := '10111101';

tabelcode[215].desimal :=215;
tabelcode[215].biner := '11010111';
tabelcode[215].cgc := '10111100';

tabelcode[216].desimal :=216;
tabelcode[216].biner := '11011000';
tabelcode[216].cgc := '10110100';

tabelcode[217].desimal :=217;
tabelcode[217].biner := '11011001';
tabelcode[217].cgc := '10110101';

tabelcode[218].desimal :=218;
tabelcode[218].biner := '11011010';
tabelcode[218].cgc := '10110111';

tabelcode[219].desimal :=219;
tabelcode[219].biner := '11011011';
tabelcode[219].cgc := '10110110';

tabelcode[220].desimal :=220;
tabelcode[220].biner := '11011100';
tabelcode[220].cgc := '10110010';

tabelcode[221].desimal :=221;
tabelcode[221].biner := '11011101';
tabelcode[221].cgc := '10110011';

tabelcode[222].desimal :=222;
tabelcode[222].biner := '11011110';
```

```
tabelcode[222].cgc := '10110001';

tabelcode[223].desimal :=223;
tabelcode[223].biner := '11011111';
tabelcode[223].cgc := '10110000';

tabelcode[224].desimal :=224;
tabelcode[224].biner := '11100000';
tabelcode[224].cgc := '10010000';

tabelcode[225].desimal :=225;
tabelcode[225].biner := '11100001';
tabelcode[225].cgc := '10010001';

tabelcode[226].desimal :=226;
tabelcode[226].biner := '11100010';
tabelcode[226].cgc := '10010011';

tabelcode[227].desimal :=227;
tabelcode[227].biner := '11100011';
tabelcode[227].cgc := '10010010';

tabelcode[228].desimal :=228;
tabelcode[228].biner := '11100100';
tabelcode[228].cgc := '10010110';

tabelcode[229].desimal :=229;
tabelcode[229].biner := '11100101';
tabelcode[229].cgc := '10010111';

tabelcode[230].desimal :=230;
tabelcode[230].biner := '11100110';
```

```
tabelcode[230].cgc := '10010101';

tabelcode[231].desimal :=231;
tabelcode[231].biner := '11100111';
tabelcode[231].cgc := '10010100';

tabelcode[232].desimal :=232;
tabelcode[232].biner := '11101000';
tabelcode[232].cgc := '10011100';

tabelcode[233].desimal :=233;
tabelcode[233].biner := '11101001';
tabelcode[233].cgc := '10011101';

tabelcode[234].desimal :=234;
tabelcode[234].biner := '11101010';
tabelcode[234].cgc := '10011111';

tabelcode[235].desimal :=235;
tabelcode[235].biner := '11101011';
tabelcode[235].cgc := '10011110';

tabelcode[236].desimal :=236;
tabelcode[236].biner := '11101100';
tabelcode[236].cgc := '10011010';

tabelcode[237].desimal :=237;
tabelcode[237].biner := '11101101';
tabelcode[237].cgc := '10011011';

tabelcode[238].desimal :=238;
tabelcode[238].biner := '11101110';
```

```
tabelcode[238].cgc := '10011001';

tabelcode[239].desimal :=239;
tabelcode[239].biner := '11101111';
tabelcode[239].cgc := '10011000';

tabelcode[240].desimal :=240;
tabelcode[240].biner := '11110000';
tabelcode[240].cgc := '10001000';

tabelcode[241].desimal :=241;
tabelcode[241].biner := '11110001';
tabelcode[241].cgc := '10001001';

tabelcode[242].desimal :=242;
tabelcode[242].biner := '11110010';
tabelcode[242].cgc := '10001011';

tabelcode[243].desimal :=243;
tabelcode[243].biner := '11110011';
tabelcode[243].cgc := '10001010';

tabelcode[244].desimal :=244;
tabelcode[244].biner := '11110100';
tabelcode[244].cgc := '10001110';

tabelcode[245].desimal :=245;
tabelcode[245].biner := '11110101';
tabelcode[245].cgc := '10001111';

tabelcode[246].desimal :=246;
tabelcode[246].biner := '11110110';
```

```
tabelcode[246].cgc := '10001101';

tabelcode[247].desimal :=247;
tabelcode[247].biner := '11110111';
tabelcode[247].cgc := '10001100';

tabelcode[248].desimal :=248;
tabelcode[248].biner := '11111000';
tabelcode[248].cgc := '10000100';

tabelcode[249].desimal :=249;
tabelcode[249].biner := '11111001';
tabelcode[249].cgc := '10000101';

tabelcode[250].desimal :=250;
tabelcode[250].biner := '11111010';
tabelcode[250].cgc := '10000111';

tabelcode[251].desimal :=251;
tabelcode[251].biner := '11111011';
tabelcode[251].cgc := '10000110';

tabelcode[252].desimal :=252;
tabelcode[252].biner := '11111100';
tabelcode[252].cgc := '10000010';

tabelcode[253].desimal :=253;
tabelcode[253].biner := '11111101';
tabelcode[253].cgc := '10000011';

tabelcode[254].desimal :=254;
tabelcode[254].biner := '11111110';
```

```
  tabelcode[254].cgc := '10000001';


  tabelcode[255].desimal :=255;
  tabelcode[255].biner := '11111111';
  tabelcode[255].cgc := '10000000';
end;
```

Fungsi Konversi Biner ke Desimal

```
function bintodec(data : string):longint;
var i:integer;
    hasil:longint;
begin
hasil:=0;
for i := 1 to length(data) do begin
hasil    :=    hasil    +    round(power(2,length(data)-i)    *
strtoint(data[i]));
end;
bintodec:=hasil;
end;
```


Fungsi Konversi Desimal Ke Biner

```
function dectobin(data : longint;pjg : integer):string;
var hasil:string;
    i : integer;
    hasildiv:longint;
    hasilmod:longint;
    thasil:string;
begin
thasil:='';
```

```
hasildiv := data;
while hasildiv<>0 do begin
hasilmod := hasildiv mod 2;
hasildiv:= hasildiv div 2;
thasil := thasil + inttostr(hasilmod);
end;

// balik
hasil:='';
for i:= 1 to length(thasil) do begin
hasil:= hasil + copy(thasil,length(thasil)-i+1,1);
end;

// genapkan sesuai pjg bit
if pjg<>0 then begin
if length(hasil)mod pjg <>0 then begin
for i:= (length(hasil) mod pjg )+1 to pjg do begin
hasil:= '0' + hasil;
end;
end;
end;
if data = 0 then begin
hasil := '0';
if pjg<>0 then begin
for i := (length(hasil) mod pjg)+1 to pjg do begin
hasil := '0'+ hasil;
end;
end;
end;
dectobin:=hasil;
end;
{$R *.dfm}
```

Fungsi Open File Image

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if openpicturedialog1.Execute then begin
edit1.Text := openpicturedialog1.FileName;
imagevessel.Picture.LoadFromFile(edit1.Text);
imgtampilvessel.Picture.LoadFromFile(edit1.Text);
end;
end;
```

Fungsi Open File Audio

```
procedure TForm1.Button2Click(Sender: TObject);
begin
if opendialog1.Execute then begin
edit2.Text := opendialog1.FileName;
end;
end;
```


Fungsi Proses Penyisipan (ENCODE)

```
procedure TForm1.Button3Click(Sender: TObject);
var

inMs : TMemoryStream;
 readCount: longint;
 oneByte : byte;
 i,j:longint;
 stampil:string;
```

```
 idx:longint;
 warna :longint;
 jmliterasix,jmliterasiy:longint;
 ix,iy,ik:longint;
 cursorpesan:longint;
 selesai:boolean;
 ivessel:longint;
 tmpstr:string;
 tmpbyte:byte;
 statusregionnoisegagal:boolean;


begin


// set n
// n slalu 112 karena perubahan bit biner maksimal utk ukuran 4x8 adalah 112
n := 112;
// set L0(threshold region)
L0 := 0.3 ;


// load image
imagevessel.Picture.LoadFromFile(edit1.Text);
imgtampilvessel.Picture.LoadFromFile(edit1.Text);


// baca file vessel dalam format rgb yang diambil dari canvas imagevessel
idx:=0;
for i:= 0 to imagevessel.height-1 do begin
for j:= 0 to imagevessel.Width - 1 do begin
idx:=idx+1;
warna := imagevessel.Canvas.Pixels[j,i] ;
isidatavessel[1,j,i]:= (warna shr 16) and 255;
isidatavessel[2,j,i]:= (warna shr 8) and 255;
```

```
isidatavessel[3,j,i]:= warna and 255;
end;
end;

pjgdata := idx ;

//baca file wav,masukkan variabel isipesan
inMs := TMemoryStream.Create;
  try

  inMs.LoadFromFile(edit2.text) ;

  readCount := inMs.Size;

  //set a position to begin reading
  inMs.Position := 0;

  // isi pesan 1-8 digunakan untuk digit ukuran pesan
  tmpstr := dectobin(inms.Size,64);

for i := 1 to 8 do begin
  tmpbyte := bintodec(copy(tmpstr,(i-1)*8+1,8));
  isipesan[i] := tmpbyte ;
end;

// data dari 8
    i:=8;
    while inMs.Position < inMs.Size do
    begin
      //read a single byte
      inMs.Read(oneByte,1);
```

```
    i:=i+1;
    isipesan[i]:= onebyte;

    end;
  finally
    inMs.free;
  end;
  pjgpesan := readcount+8;

  cursorpesan:=0;
iiterasipesan:=0;

// tampilkan var isi pesan  dalam CGC
memo1.Clear;
stampil:='';
for i := 1 to pjgpesan do begin

if i<pjgpesan then begin
stampil := stampil + tabelcode[isipesan[i]].cgc + ' | '   ;
end else begin
stampil := stampil + tabelcode[isipesan[i]].cgc   ;
end;

end;
memo1.text := stampil;



//hitung jmliterasi yang dibutuhkan utk embbeding pesan
selesai:= false;

if pjgpesan mod 32 = 0 then
jmliterasi := pjgpesan div 32
```

```
else
jmliterasi := (pjgpesan div 32 ) + 1;


if jmliterasi*64> pjgdata then begin
  showmessage('vessel tidak cukup untuk proses embedding pesan sebesar ' + inttostr(pjgpesan) );
exit;


end else begin
  showmessage('Mulai Proses Embbeding Pesan');
  jmlregionnoisetotal:=0;


// utk setiap embbedding, butuh matriks 8x8, utk 4x8 pjg pesan
if imagevessel.Width mod 8 <> 0 then begin
jmliterasix := (imagevessel.Width div 8) + 1;
end else begin
jmliterasix := imagevessel.Width div 8;
end;
if imagevessel.Height  mod 8 <> 0 then begin
jmliterasiy := (imagevessel.Height  div 8) + 1;
end else begin
jmliterasiy := imagevessel.Height  div 8;
end;


// load dulu imagestego dengan image vessel
imagestego.Picture.LoadFromFile(edit1.Text);


iiterasi:=0;
statusregionnoisegagal:=false;
for i:= 1 to jmliterasiy do begin
for j:= 1 to jmliterasix do begin
iiterasi := iiterasi + 1;
```

```
// ambil warna b taruh ke currentmatrix
for ix:= 0 to 7 do begin
for iy:= 0 to 7 do begin
currentmatrik[ix,iy] := tabelcode[isidatavessel[1,(j-1)*8+ ix,(i-1)*8 + iy]].cgc ;
end;
end;

// tampilkan currentmatrik ke memo2
memo2.Clear;
stampil:='';
for iy:= 0 to 7 do begin
 for ix:= 0 to 7 do begin

if (ix<7) or (iy<7)  then begin
stampil := stampil + currentmatrik[ix,iy] + ' | '   ;
end else begin
stampil := stampil + currentmatrik[ix,iy]   ;
end;
end;
stampil := stampil + chr(13);

end;
memo2.text := stampil;

// ubah ke format bit plane
for ik:= 0 to 7 do begin
for iy:= 0 to 7 do begin
BITPLANEvessel[iiterasi,ik,iy]:='';
for ix := 0 to 7 do begin
BITPLANEvessel[iiterasi,ik,iy]:= BITPLANEvessel[iiterasi,ik,iy] + currentmatrik[ix,iy][ik+1];
```

```
end;
end;
end;

// tampilkan bitplane ke memo3
memo3.Clear;
stampil:='';
for iy:= 0 to 7 do begin
 for ik:= 0 to 7 do begin

if (ik<7) or (iy<7)  then begin
stampil := stampil + BITPLANEvessel[iiterasi,ik,iy] + ' | '   ;
end else begin
stampil := stampil + BITPLANEvessel[iiterasi,ik,iy]   ;
end;
end;
stampil := stampil + chr(13);

end;
memo3.text := stampil;

// mencari k utk current matrik sekarang
for ik:= 0  to 7 do begin // bit plane ke-
currentkBitPlaneVessel[ik]:=0;
for iy:= 0 to 7 do begin
   for ix:= 1 to 8 do begin
       if  Bitplanevessel[iiterasi,ik,iy][ix]='1'  then  begin  //
karena 1 mk diproses
           // dicari perubahan atasnya
           if iy>0 then begin
           if bitplanevessel[iiterasi,ik,iy-1][ix]='0' then begin
            inc(currentkbitplanevessel[ik],1);
```

```
            end;
            end;
            // dicari perubahan bawahnya
            if iy<7 then begin
            if bitplanevessel[iiterasi,ik,iy+1][ix]='0' then begin
             inc(currentkbitplanevessel[ik],i);
            end;
            end;
            // dicari perubahan kanannya
            if ix<8 then begin
            if bitplanevessel[iiterasi,ik,iy][ix+1]='0' then begin
             inc(currentkbitplanevessel[ik],i);
            end;
            end;
            // dicari perubahan kirinya
            if ix>1 then begin
            if bitplanevessel[iiterasi,ik,iy][ix-1]='0' then begin
             inc(currentkbitplanevessel[ik],i);
            end;
            end;

        end;
   end;
end;

// di cari alpa nya
currentLbitPlanevessel[ik] := currentkbitplanevessel[ik]/n;
end;

// dicek ada >4 kah yang nilai komplekistasnya di atas threshold
noise(region noise)
jmlRegionNoise:=0;
```

```
for ik:= 0 to 7 do begin
if currentLbitplanevessel[ik]>L0 then begin
jmlRegionNoise:=jmlRegionNoise+1;
end;
end;



if jmlRegionNoise >= 4 then begin


iiterasipesan  := iiterasipesan + 1 ;
jmlregionnoisetotal:=jmlregionnoisetotal+1;


for ik:= 0 to 3 do begin
for iy:= 0 to 7 do begin
cursorpesan:= cursorpesan + 1;
if  cursorpesan>pjgpesan then begin
DATASecret[iiterasipesan,ik,iy] := tabelcode[32].cgc ;
end else begin
DATASecret[iiterasipesan,ik,iy]                                    :=
tabelcode[isipesan[cursorpesan]].cgc ;
end;
if cursorpesan > pjgpesan-10 then begin
end;


end;
end;


// hitung nilai kompleksitas secret data sekarang
for ik:= 0 to 3 do begin
currentkBitPlaneSecret[ik]:=0;


for iy:= 0 to 7 do begin
```

```
   for ix:= 1 to 8 do begin
       if  DATAsecret[iiterasipesan,ik,iy][ix]='1'  then  begin  //
karena 1 mk diproses
           // dicari perubahan atasnya
           if iy>0 then begin
           if    DATAsecret[iiterasipesan,ik,iy-1][ix]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
           end;
           end;
           // dicari perubahan bawahnya
           if iy<7 then begin
           if    DATAsecret[iiterasipesan,ik,iy+1][ix]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
           end;
           end;
           // dicari perubahan kanannya
           if ix<8 then begin
           if    DATAsecret[iiterasipesan,ik,iy][ix+1]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
           end;
           end;
           // dicari perubahan kirinya
           if ix>1 then begin
           if    DATAsecret[iiterasipesan,ik,iy][ix-1]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
           end;
           end;

       end;
   end;
```

```
end;


// di cari alpa nya
currentLbitPlanesecret[ik] := currentkbitplanesecret[ik]/n;
end;


// mencari lokasi vessel yang tepat untuk disisipkan secret sekarang
for ik:= 0 to 7 do begin
currentKonjugasiSecret[ik]:=0;
end;


// reset status sisipvessel nya
for ivessel:= 6 downto 0 do begin
statusbpcvessel[ivessel]:=true;
end;


for ik:= 0 to 3 do begin
for ivessel:= 6 downto 0 do begin
if statusbpcvessel[ivessel]=true then begin
statusbpcvessel[ivessel]:=false;
if currentLbitPlanevessel[ivessel]>L0 then begin
if currentLbitplanesecret[ik]<L0 then begin
    for iy:= 0 to 7 do begin
    hasilxorblock[iy]:='';
       for ix:= 1 to 8 do begin
       if
DATAsecret[iiterasipesan,ik,iy][ix]=PERFECTLYCOMPLEXBLOCK[iy][ix]
then begin
       hasilxorblock[iy]:=hasilxorblock[iy]+'0';
       end else begin
       hasilxorblock[iy]:=hasilxorblock[iy]+'1';
       end;
```

```
       end;
    end;

    currentKonjugasiSecret[ivessel]:=1;

// pindah ke vessel sekarang
for iy:=0 to 7 do begin
  DATAsecret[iiterasipesan,ik,iy]:=hasilxorblock[iy];
  bitplanevessel[iiterasi,ivessel,iy]:=hasilxorblock[iy];
end;

// hitung alpa sekarang
currentkBitPlaneSecret[ik]:=0;
for iy:= 0 to 7 do begin
   for ix:= 1 to 8 do begin
       if DATAsecret[iiterasipesan,ik,iy][ix]='1' then begin
           // dicari perubahan atasnya
           if iy>0 then begin
           if    DATAsecret[iiterasipesan,ik,iy-1][ix]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
           end;
           end;
           // dicari perubahan bawahnya
           if iy<7 then begin
           if    DATAsecret[iiterasipesan,ik,iy+1][ix]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
           end;
           end;
           // dicari perubahan kanannya
           if ix<8 then begin
```

```
            if    DATAsecret[iiterasipesan,ik,iy][ix+1]='0'    then
begin
             inc(currentkbitplanesecret[ik],1);
            end;
            end;
            // dicari perubahan kirinya
            if ix>1 then begin
            if    DATAsecret[iiterasipesan,ik,iy][ix-1]='0'    then
begin
             inc(currentkbitplanesecret[ik],1);
            end;
            end;


       end;
   end;
end;
currentLbitPlanesecret[ik]:=currentkbitPlanesecret[ik]/n;


end else begin


// pindah ke vessel sekarang
for iy:=0 to 7 do begin

bitplanevessel[iiterasi,ivessel,iy]:=DATAsecret[iiterasipesan,ik,
iy];
end;
end;
break;
end;
end;
end;
end;


bitplanevessel[iiterasi,7,7]:='';
```

```
for ix:= 0 to 7 do begin
if currentKonjugasiSecret[ix]=0 then
bitplanevessel[iiterasi,7,7]  :=  bitplanevessel[iiterasi,7,7]  + '0'
else
bitplanevessel[iiterasi,7,7]  :=  bitplanevessel[iiterasi,7,7]  + '1';

end;

// tampilkan bitplane ke memo2
 memo2.Clear;
stampil:='';
for iy:= 0 to 7 do begin
 for ik:= 0 to 7 do begin

if (ik<7) or (iy<7)  then begin
stampil := stampil + BITPLANEvessel[iiterasi,ik,iy] + ' | '   ;
end else begin
stampil := stampil + BITPLANEvessel[iiterasi,ik,iy]   ;
end;
end;
stampil := stampil + chr(13);

end;
memo2.text := stampil;

// ubah format bpc ke cgc lagi
 for ix:= 0 to 7 do begin
 for iy:= 0 to 7 do begin
 currentmatrik[ix,iy]:='';
 for ik:= 0 to 7 do begin
```

```
currentmatrik[ix,iy]:=currentmatrik[ix,iy]+bitplanevessel[iiteras
i,ik,iy][ix+1];

end;

end;

end;


// tampilkan currentmatrik ke memo1

memo1.Clear;

stampil:='';

for iy:= 0 to 7 do begin

 for ix:= 0 to 7 do begin


if (ix<7) or (iy<7)  then begin

stampil := stampil + currentmatrik[ix,iy] + ' | '   ;

end else begin

stampil := stampil + currentmatrik[ix,iy]   ;

end;

end;

stampil := stampil + chr(13);


end;

memo1.text := stampil;


// taruh hasil perhitungan ke var isidatavessel kembali

for ix:= 0 to 7 do begin

for iy:= 0 to 7 do begin

isidatavessel[1,(j-1)*8+   ix,(i-1)*8   +   iy]:=   cgctodesimal(
currentmatrik[ix,iy]);

end;

end;


//  tulis  hasil  perhitungan  sekarang  dari  isidatavessel  ke
imagestego canvas
```

```
for ix:= 0 to 7 do begin

for iy:= 0 to 7 do begin

imagestego.Canvas.Pixels[(j-1)*8+ix,(i-1)*8+iy]:=
rgb(isidatavessel[3,(j-1)*8+ix,(i-1)*8+iy],isidatavessel[2,(j-
1)*8+ix,(i-1)*8+iy],isidatavessel[1,(j-1)*8+ix,(i-1)*8+iy]);

end;

end;


// cek jika cursorpesan >= pjg pesan maka selesai

if cursorpesan>=pjgpesan then begin

selesai:=true;

end;


end;

if selesai=true then

break;


end;

if selesai= true then

break;

end;


end;


if (cursorpesan < pjgpesan) then begin
  if (cursorpesan = 0) then begin
      ShowMessage('Data vessel kurang kompleks');
      exit;
  end;
end;


// tulis ke file
 imagestego.Picture.Bitmap.PixelFormat := pf24bit;
```

```
 imagestego.Picture.SaveToFile('C:\PRESENTASI\Output\hasil.bmp');

imgtampilstego.Picture.LoadFromFile('C:\PRESENTASI\Output\hasil.b
mp');


end;


procedure TForm1.FormShow(Sender: TObject);
begin
set_CGC_BINARY_CODE;
set_perfectlycomplexblock;
end;
```

Fungsi Open File Image Stego

```
procedure TForm1.Button6Click(Sender: TObject);
begin
if openpicturedialog1.Execute then begin
edit3.Text := openpicturedialog1.FileName;
imgtampilstegodecode.Picture.LoadFromFile(edit3.Text);
imagestegodecode.Picture.LoadFromFile(edit3.Text);
end;
end;
```

Fungsi Proses Ekstraksi (DECODE)

```
procedure TForm1.Button5Click(Sender: TObject);
var


inMs : TMemoryStream;
 readCount: integer;
```

```
 oneByte : byte;
 i,j:integer;
 stampil:string;
 idx:longint;
 warna :longint;
 jmliterasix,jmliterasiy:integer;
 ix,iy,ik:integer;
 cursorpesan:integer;
 selesai:boolean;
 ivessel:integer;
 pertama:boolean;
 ictr:integer;
 tmpstr:string;
 tmpbyte:byte;
 tmpbiner:string;
 statusregionnoisegagal:boolean;


begin


// set n
// n slalu 112 karena perubahan bit biner maksimal utk ukuran 4x8
adalah 112
n := 112;


// set L0 (threshold region)
L0 := 0.3 ;


// load hasil encode ke image vessel
imagevessel.Picture.LoadFromFile(edit3.Text);
imgtampilvessel.Picture.LoadFromFile(edit3.Text);


// baca file vessel dalam format rgb yang diambil dari canvas
imagevessel
```

```
idx:=0;
for i:= 0 to imagevessel.height-1 do begin
for j:= 0 to imagevessel.Width - 1 do begin
idx:=idx+1;
warna := imagevessel.Canvas.Pixels[j,i] ;
isidatavessel[1,j,i]:= (warna shr 16) and 255;
isidatavessel[2,j,i]:= (warna shr 8) and 255;
isidatavessel[3,j,i]:= warna and 255;
end;
end;

pjgdata := idx ;

// baca file ogg
  inMs := TMemoryStream.Create;
  try

  inMs.LoadFromFile(edit2.text) ;

  readCount := inMs.Size;

//set a position to begin reading
    inMs.Position := 0;

    // data dari 8
    i:=1;
    while inMs.Position < inMs.Size do
    begin
    //read a single byte
      inMs.Read(oneByte,1);

    i:=i+1;
```

```
    isipesan[i]:= onebyte;

    end;
  finally

  end;

pjgpesan := readcount+1;
cursorpesan:=0;
iiterasipesan:=0;

selesai:= false;
if pjgdata mod 64 = 0 then
jmliterasi := pjgdata div 64
else
jmliterasi := (pjgdata div 64 ) + 1;

showmessage('Mulai Proses Decoding Pesan');

if imagevessel.Width mod 8 <> 0 then begin
jmliterasix := (imagevessel.Width div 8) + 1;
end else begin
jmliterasix := imagevessel.Width div 8;
end;
if imagevessel.Height  mod 8 <> 0 then begin
jmliterasiy := (imagevessel.Height  div 8) + 1;
end else begin
jmliterasiy := imagevessel.Height  div 8;
end;
statusregionnoisegagal:=false;
pertama:=true;
iiterasi:=0;
```

```
for i:= 1 to jmliterasiy do begin
for j:= 1 to jmliterasix do begin
iiterasi := iiterasi + 1;

// ambil warna b taruh ke currentmatrix
for ix:= 0 to 7 do begin
 for iy:= 0 to 7 do begin
currentmatrik[ix,iy] := tabelcode[isidatavessel[1,(j-1)*8+ ix,(i-1)*8 + iy]].cgc ;
end;
end;

// tampilkan currentmatrik ke memo2
memo2.Clear;
stampil:='';
for iy:= 0 to 7 do begin
 for ix:= 0 to 7 do begin

if (ix<7) or (iy<7)  then begin
stampil := stampil + currentmatrik[ix,iy] + ' | '   ;
end else begin
stampil := stampil + currentmatrik[ix,iy]   ;
end;
end;

stampil := stampil + chr(13);

end;
memo2.text := stampil;

// ubah ke format bit plane
for ik:= 0 to 7 do begin
```

```
for iy:= 0 to 7 do begin
BITPLANEvessel[iiterasi,ik,iy]:='';
for ix := 0 to 7 do begin
BITPLANEvessel[iiterasi,ik,iy]:= BITPLANEvessel[iiterasi,ik,iy] +
currentmatrik[ix,iy][ik+1];
end;

end;
end;

// tampilkan bitplane ke memo3
memo3.Clear;
stampil:='';
for iy:= 0 to 7 do begin
 for ik:= 0 to 7 do begin

if (ik<7) or (iy<7)  then begin
stampil := stampil + BITPLANEvessel[iiterasi,ik,iy] + ' | '   ;
end else begin
stampil := stampil + BITPLANEvessel[iiterasi,ik,iy]   ;
end;
end;

stampil := stampil + chr(13);

end;
memo3.text := stampil;

// mencari k utk current matrik sekarang
for ik:= 0  to 7 do begin
currentkBitPlaneVessel[ik]:=0;
```

```
for iy:= 0 to 7 do begin
   for ix:= 1 to 8 do begin
       if Bitplanevessel[iiterasi,ik,iy][ix]='1' then begin
           // dicari perubahan atasnya
           if iy>0 then begin
           if bitplanevessel[iiterasi,ik,iy-1][ix]='0' then begin
            inc(currentkbitplanevessel[ik],1);
           end;
           end;
           // dicari perubahan bawahnya
           if iy<7 then begin
           if bitplanevessel[iiterasi,ik,iy+1][ix]='0' then begin
            inc(currentkbitplanevessel[ik],i);
           end;
           end;
           // dicari perubahan kanannya
           if ix<8 then begin
           if bitplanevessel[iiterasi,ik,iy][ix+1]='0' then begin
            inc(currentkbitplanevessel[ik],i);
           end;
           end;
           // dicari perubahan kirinya
           if ix>1 then begin
           if bitplanevessel[iiterasi,ik,iy][ix-1]='0' then begin
            inc(currentkbitplanevessel[ik],i);
           end;
           end;

       end;
   end;
end;
```

```
// di cari alpa nya
currentLbitPlanevessel[ik] := currentkbitplanevessel[ik]/n;
end;


// dicek ada >4 kah yang nilai komplekistasnya di atas threshold noise(region noise)
jmlRegionNoise:=0;
for ik:= 0 to 7 do begin
if currentLbitplanevessel[ik]>L0 then begin
jmlRegionNoise:=jmlRegionNoise+1;
end;
end;



if jmlRegionNoise >= 4 then begin // jika jmlregionnoise >= 4 maka ada data di sini
iiterasipesan:=iiterasipesan + 1;


for ix:= 1 to 8 do begin
if bitplanevessel[iiterasi,7,7][ix] = '0' then
currentKonjugasiSecret[ix-1]:=0
else
if bitplanevessel[iiterasi,7,7][ix] = '1' then
currentKonjugasiSecret[ix-1]:=1;
end;


for ivessel:= 6 downto 0 do begin
statusbpcvessel[ivessel]:=true;
end;


for ik:= 0 to 3 do begin
for ivessel:= 6 downto 0 do begin
```

```
if statusbpcvessel[ivessel]=true then begin
statusbpcvessel[ivessel]:=false;
if currentLbitPlanevessel[ivessel]>L0 then begin

if currentKonjugasiSecret[ivessel]=1 then begin
    for iy:= 0 to 7 do begin
    hasilxorblock[iy]:='';
       for ix:= 1 to 8 do begin
       if
bitplanevessel[iiterasi,ivessel,iy][ix]=PERFECTLYCOMPLEXBLOCK[iy]
[ix] then begin // karena 1 mk diproses
       hasilxorblock[iy]:=hasilxorblock[iy]+'0';
       end else begin
       hasilxorblock[iy]:=hasilxorblock[iy]+'1';
       end;
       end;
    end;

// pindah ke vessel sekarang
for iy:=0 to 7 do begin
  DATAsecret[iiterasipesan,ik,iy]:=hasilxorblock[iy];
end;

// hitung alpa sekarang
currentkBitPlaneSecret[ik]:=0;
for iy:= 0 to 7 do begin
   for ix:= 1 to 8 do begin
       if DATAsecret[iiterasipesan,ik,iy][ix]='1' then begin
           // dicari perubahan atasnya
           if iy>0 then begin
           if    DATAsecret[iiterasipesan,ik,iy-1][ix]='0'    then
begin
            inc(currentkbitplanesecret[ik],1);
```

```
            end;
            end;
            // dicari perubahan bawahnya
            if iy<7 then begin
            if    DATAsecret[iiterasipesan,ik,iy+1][ix]='0'    then
begin
             inc(currentkbitplanesecret[ik],1);
            end;
            end;
            // dicari perubahan kanannya
            if ix<8 then begin
            if    DATAsecret[iiterasipesan,ik,iy][ix+1]='0'    then
begin
             inc(currentkbitplanesecret[ik],1);
            end;
            end;
            // dicari perubahan kirinya
            if ix>1 then begin
            if    DATAsecret[iiterasipesan,ik,iy][ix-1]='0'    then
begin
             inc(currentkbitplanesecret[ik],1);
            end;
            end;

       end;
   end;
end;
currentLbitPlanesecret[ik]:=currentkbitPlanesecret[ik]/n;
end else begin

for iy:=0 to 7 do begin
 DATAsecret[iiterasipesan,ik,iy]                                    :=
bitplanevessel[iiterasi,ivessel,iy];
end;
```

```
end;
break;
end;
end;
end;
end;

// ubah format bpc ke cgc lagi
for ik:= 0 to 3 do begin
 for ix:= 0 to 7 do begin
 currentmatrik[ix,ik]:='';
currentmatrik[ix,ik]:=currentmatrik[ix,ik]+DATAsecret[iiterasipes
an,ik,ix];

end;
end;

// tampilkan currentmatrik ke memo1
memo1.Clear;
stampil:='';
for iy:= 0 to 3 do begin
 for ix:= 0 to 7 do begin
if (ix<7) or (iy<3)  then begin
stampil := stampil + currentmatrik[ix,iy] + ' | '   ;
end else begin
stampil := stampil + currentmatrik[ix,iy]   ;
end;
end;

stampil := stampil + chr(13);
```

```
end;
memo1.text := stampil;


// jika ini adalah data pertama, maka diambil jmldatanya dan set cursor pesan = 31
if pertama then begin
pertama:=false;


// taruh hasil perhitungan ke var isipesan
ictr:=0;
for iy:= 0 to 3 do begin
 for ix:= 0 to 7 do begin
ictr := ictr + 1;
isipesan[(iiterasipesan-1)*32+ictr]:=
cgctodesimal(currentmatrik[ix,iy]);
if (ix>=0) and (ix<=7) and( iy=0) and (iiterasipesan= 1) then begin
if (ix=0) then begin // awal pjg pesan
tmpbyte := cgctodesimal(currentmatrik[ix,iy]);
tmpbiner := dectobin(tmpbyte,8);
tmpstr:= tmpbiner;
end else
if (ix = 7) then begin // akhir pjg pesan
tmpbyte := cgctodesimal(currentmatrik[ix,iy]);
tmpbiner := dectobin(tmpbyte,8)  ;
tmpstr:= tmpstr + tmpbiner;


// ubah ke desimal
pjgpesan := bintodec(tmpstr);
end else begin // tengah2
tmpbyte := cgctodesimal(currentmatrik[ix,iy]);
tmpbiner := dectobin(tmpbyte,8)   ;
tmpstr:= tmpstr + tmpbiner;
```

```
end;


end;
end;
end;


end else
begin


// taruh hasil perhitungan ke var isipesan
ictr:=0;
for iy:= 0 to 3 do begin
 for ix:= 0 to 7 do begin


ictr := ictr + 1;
isipesan[(iiterasipesan-1)*32+ictr]:=
cgctodesimal(currentmatrik[ix,iy]);
end;
end;


end;
if iiterasipesan*32 > pjgpesan then begin
break;
end;


end;


end;
if iiterasipesan*32 > pjgpesan then begin
break;
end;
```

```
end ; // end proses decode

// tuliskan isipesan ke variabel inms
inms.position := 0;
for i:= 8+1 to pjgpesan + 8 do begin
  oneByte := isipesan[i];
  inMs.write(oneByte,1);
end;

inms.Size := pjgpesan;
inms.SaveToFile('C:\PRESENTASI\Output\hasil.ogg');
inms.Free;

end;

end.
```