BAB 2
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Kriptografi adalah salah satu teknik yang dapat digunakan untuk mengamankan sebuah data atau informasi. Algoritma *Twofish* merupakan salah satu algoritma kriptografi yang kuat dan merupakan salah satu dari lima kandidat utama *Advanced Encryption Standard*(AES)oleh *National Institute of Standards and Technology* (NIST). Beberapa kasus penelitian enkripsi dengan menggunakan algoritma *Twofish* telah banyak dilakukan. Menurut Ratih (2007), dalam penelitiannya tentang "Studi dan Implementasi Enkripsi Pengiriman Pesan Suara dengan Menggunakan Algortima *Twofish"*, menyebutkan bahwa algoritma *Twofish* merupakan algoritma yang sampai saat ini dinyatakan aman karena masih belum ada serangan kriptanalisis yang benar-benar dapat mematahkan algoritma tersebut. Selain itu algoritma *Twofish* dapat diterapkan untuk melakukan enkripsi pada aliran pesan suara dengan menyesuaikan modenya menjadi mode operasi *counter*. Pada penelitian ini sistem hanya dapat mengenkripsi suara dan digunakan untuk pengiriman pesan suara antara dua buah komputer melalui jaringan. Dari hasil penelitian yang dilakukan menunjukkan bahwa algoritma *Twofish* merupakan algoritma yangdapat diterapkan untuk melakukan enkripsi aliranpesan suara dengan cukup baik setelahmengalami modifikasi pada mode operasinya. Kualitas suara setelah mengalami enkripsi dekripsi memiliki kualitas yang cukup baik dan *delay* yang dihasilkan tidak terlalu menggangu.

Selain itu, Dani (2006), dalam penelitiannya tentang "Algoritma *Twofish* Sebagai Finalis AESdan Metode Kriptanalisisnya", menyebutkan bahwa algoritma *Twofish* merupakan algoritma yang tidak mengandung kunci yang lemah, sangat efisien, serta memiliki desain yang fleksible dan sederhana. Pada penelitian ini dilakukan pengujian proses enkripsi dan dekripsi beserta lama waktu proses enkripsi dan dekripsi algoritma *Twofish* dengan algoritma Serpent, MARS, RC6, Rjindael, dan DES serta dilakukan kriptanalisis terhadap algoritma *Twofish*.

Dari hasil penelitian yang dilakukan menunjukkan bahwa *Twofish* adalah algoritma ideal yang efisien untuk diterapkan baik pada mikropresesor besar, *smart cards,* maupun perangkat keras *dedicated.Twofish* memerlukan waktu yang lebih lama dalam melakukan proses enkripsi dan dekripsi pada berbagai panjang kunci dibandingkan dengan algoritma MARS, RC6, Rjindael dan memerlukan waktu lebih cepat dalam melakukan proses enkripsi dan dekripsi pada berbagai panjang kunci dibandingkan dengan algoritma Serpent. Selain itu tidak terdapat serangan yang lebih efisien lagi terhadap algortima *Twofish* daripada *brute force*.

Selain kriptografi, terdapat juga teknik yang dapat digunakan untuk mengamankan sebuah data atau informasi yaitu steganografi. Steganografi adalah sebagai suatu seni penyembunyian pesan ke dalam pesan lainnya. Berbeda dengan kriptografi, steganografi menyembunyikan pesan rahasia agar orang awam tidak menyadari keberadaan pesan yang disembunyikan. *Least Significant Bit* (LSB) merupakan salah satu teknik steganografi yang dapat diterapkan untuk penyembunyian data atau pesan rahasia pada data digital. Sudah banyak juga penelitian mengenai metode ini. Seperti halnya Henry Setyawan (2009), dalam penelitiannya tentang "Implementasi Steganografi dengan Metode *Least Significant Bit* (LSB)", menyebutkan bahwa data atau informasi yang disembunyikan pada citra digital dengan menggunakan LSB akan sulit dideteksi. Pada penelitian ini, citra yang digunakan sebagai medium berformat PNG, JPG dan BMP yang memiliki ukuran *pixel* masing-masing, serta melakukan analisa hasil steganografi dengan metode subjektif menggunakan MOS (*Mean Opinion Score*). Dari hasil penelitian yang dilakukan menunjukkan bahwa metode LSB sangat tergantung terhadap ukuran citra dan pesan yang akan disisipkan dan berbanding terbalik dengan kualitas citra.

Dalam Tugas Akhir ini, akan dibuat suatu sistem yang mengimplementasikan perpaduan antara algoritma *Twofish* dengan algoritma *Least Significant Bit*untuk menyembunyikan file teks pada file citra digital. Format fileteks adalah .txt dan format citra digitalyang BMP dan PNG yang mempunyai ukuran minimal 200 kb.

2.2 Landasan Teori

2.2.1 Teori Dasar Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa yunani : "*cryptos*" yang artinya "*secret*" (rahasia), dan "*graphein*" yang artinya "*writing*" (tulisan), jadi kriptografi berarti "*secret writing*" (tulisan rahasia). Kriptografi adalah ilmu dan seni untuk menjaga kemanan dan kerahasiaan pesan dengan caramenyandikan ke dalam bentuk yang tidak dapat dimengerti lagi maknanya (Munir, 2006).

Menurut Alfred J. Menezes, Paul C.van Oorschot, dan Scott A. Venstone (1996), ada beberapa tujuan dari kriptografi, antara lain :

- Kerahasiaan (*secrecy*) adalah layanan yang digunakan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berkepentingan. Di dalam kriptografi layanan ini direalisasikan dengan menyadikan pesan menjadi *ciphertext*.
- Integritas data (*data integrity*) adalah layanan yang menjamin bahwa pesan masih asli atau belum pernah di manipulasi selama pengiriman. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak.
- Otentikasi (*authentication*) adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication*) maupun mengidentifikasi kebenaran sumber pesan (*data origin authentication*). Di dalam kriptografi layanan ini direalisasikan dengan menggunakan tanda tangan digital yang menyatakan sumber pesan.
- Nirpenyangkalan (non-*repudiation*) adalah layanan untuk mencegah entitas yang berkomunikasi penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal bahwa telah menerima pesan.

Kriptografi sendiri mempunyai komponen-komponen untuk mencapai tujuan keriptografi. Menurut Ariyus (2009: 19), pada dasarnya kriptografi terdiri dari beberapa komponen seperti :

- **Enkripsi**

  Enkripsi merupakan hal yang sangat penting dalam kriptografi sebagai pengamanan atas data yang dikirimkan agar rahasianya terjaga. Pesan aslinya disebut *plaintext* yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan sebagai *cipher*atau kode. Seperti ketika kita tidak mengerti akan artisebuah kata, kita bisa melihatnya di dalam kamus atau daftar istilah. Berbeda dengan enkripsi, untuk mengubah *plaintext* ke bentuk *ciphertext* digunakan algoritma yang bisa mengkodekan data yang diinginkan.

- **Dekripsi**

  Dekripsi merupakan kebalikan dari enkripsi, pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (*plaintext*), yang disebut dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan yang digunakan untuk enkripsi.

- **Kunci(*key)***

  Kunci yang yang dimaksud di sini adalah kunci yang dipakai untuk melakukan proses enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, yakni kunci pribadi (*privatekey*) dan kunci umum (*publickey*).

- ***Ciphertext***

  *Ciphertext*merupakan suatu pesan yang sudah melalui proses enkripsi. Pesan yang ada pada *ciphertext* tidak bisa dibaca karena berisi karakter-karakter yang tidak memiliki makna (arti).

- ***Plaintext***

  Plaintext sering juga disebut *cleartext*, merupakan suatu pesan bermakna yang ditulis atau diketik dan *plaintext* itulah yang akan diproses menggunakan algoritma kriptografi agar menjadi *ciphertext*.

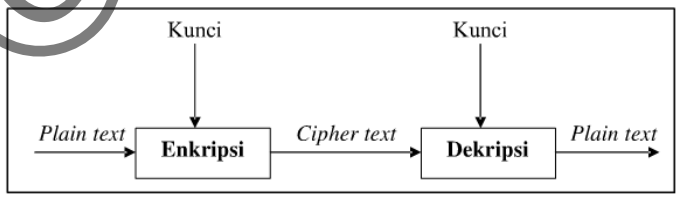
- **Pesan**

  Pesan bisa berupa data atau infomasi yang dikirim (melalui kurir, saluran komunikasi data, dan sebagainya) atau yang disimpan di dalam media perekaman (kertas, *storage*, dan sebagainya).

- ***Cryptanalysis***

  *Cryptanalysis*bisa diartikan sebagai analisis sandi atau suatu ilmu untuk mendapatkan *plaintext* tanpa harus mengetahui kunci yang digunakan dalam proses enkripsi. Jika suatu *ciphertext* berhasil menjadi *plaintext* tanpa menggunakan kunci yang sah, maka proses tersebut dinamakan *breaking code* yang dilakukan oleh para *cryptanalys*. Analisis sandi juga mampu menemukan kelemahan dari suatu algoritma kriptografi dan akhirnya bisa menemukan kunci atau *plaintext* dari *ciphertext* yang dienkripsi menggunakan algoritma tertentu.

Disamping itu kriptografi mempunyai komponen utama yaitu proses enkripsi dan dekripsi. Kriptografi membutuhkan sebuah kunci untuk mengubah *plaintext* menjadi *ciphertext* atau sebaliknya. Aspek kerahasiaan kunci sangatlah penting untuk diperhatikan karena apabila kunci tersebut di ketahui oleh pihak yang tidak bersangkutan maka mereka bisa membongkar pesan yang sudah dilakukan proses enkripsi. Berikut ini adalah skema yang menggambarkan proses enkripsi dan dekripsi pada umumnya.



*Gambar 2.1.*Proses Enkripsi dan Dekripsi

Dikutip dari : Kriptografi, 2006, halaman 6

Secara matematis, proses enkripsi merupakan pengoperasian fungsi E (enkripsi) menggunakan k (kunci) pada M (*plaintext*) sehingga dihasilkan C (*ciphertext*). Notasi dari proses enkripsi seperti dibawah ini:

$$\mathbf{E_k\,(M) = C} \qquad [2.1]$$

Sedangkan untuk proses dekripsi, merupakan pengoperasian fungsi D (dekripsi) menggunakan k (kunci) pada C (*ciphertext*) sehingga dihasilkan M (*plaintext*). Notasi dari proses dekripsi seperti dibawah ini :

$$\mathbf{D_k\,(C) = M} \qquad [2.2]$$

Sehingga dari dua hubungan diatas berlaku :

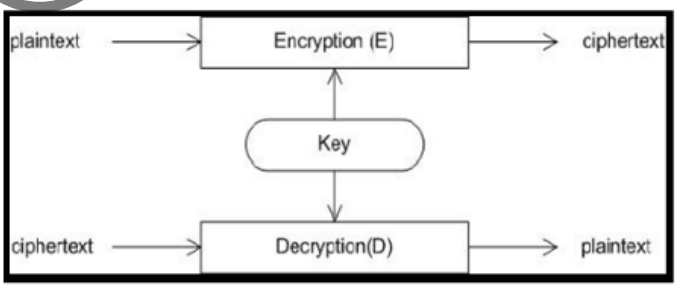
$$\mathbf{D_k\,(E_k\,(M)) = M} \qquad [2.3]$$

2.2.2 Algoritma Kriptografi Berdasarkan Jenis Kunci

Berdasarkan kunci yang dipakai, algoritma kriptografi dibagi menjadi dua, yaitu Algoritma Simetris dan Algoritma Asimetris (Munir, 2006, hlm. 13-14).

2.2.2.1 Algoritma Simetris

Algoritma simetris (*symmetric algorithm*) adalah suatu algoritma dimana kunci enkripsi yang digunakan sama dengan kunci dekripsi, sehingga algoritma ini disebut juga sebagai *single-key algoritm*. Algoritma ini disebut juga algoritma kunci rahasia (*secret-key algorithm*), karena kunci yang digunakan harus terjaga kerahasiaannya dan hanya boleh diketahui oleh pengirim dan penerima pesan saja. Yang termasuk dalam algoritma kunci simetri adalah OTP, DES, RC2, RC4, RC5, RC6, IDEA, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael (AES), Blowfish, GOST, A5, Kasumi dan lain-lain. Gambar 2.2 memenggambarkan skema dari algoritma simetris :



Gambar 2.2. Skema Algoritma Simetris

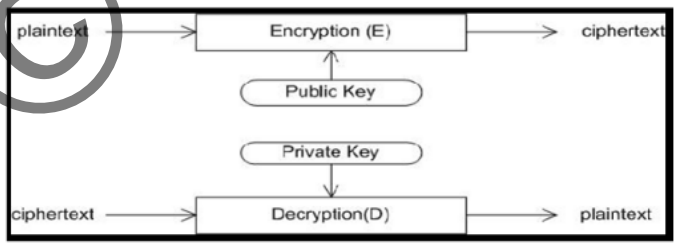
Dikutip dari : Kriptografi, 2006, halaman 6

Algoritma simestris mempunyai kelebihan dan kekurangan antara lain :

- Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetris.
- Dapat digunakan pada sistem *real-time*, karena kecepatan algoritma ini cukup tinggi.
- Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
- Permasalahan dalam pengiriman kunci itu sendiri yang disebut *"key distribution problem"*.

2.2.2.2 Algoritma Asimetris

Algoritma asimetris (*asymmetric algorithm*) adalah suatu algoritma dimana kunci enkripsi yang digunakan tidak sama dengan kunci dekripsi. Pada algoritma ini menggunakan dua kunci yakni kunci publik (*public key*) dan kunci privat (*private key*). Kunci publik disebarkan secara umum dan digunakan untuk proses enkripsi sedangkan kunci privat disimpan secara rahasia oleh si pengguna dan digunakan untuk proses dekripsi.Yang termasuk dalam algoritma kunci asimetri adalah ECC, LUC, RSA, El Gamal dan DH. Gambar 2.3 memenggambarkan skema dari algoritma asimetris.



Gambar 2.3. Skema Algoritma Asimetris

Dikutip dari : Kriptografi, 2006, halaman 14

Algoritma asimestris mempunyai kelebihan dan kekurangan antara lain :

- Masalah keamanan pada distribusi kunci dapat lebih baik daripada algoritma simetris.

- Masalah manajemen kunci yang lebih baik karena jumlah kunci yang lebih sedikit.
- Kecepatan yang lebih rendah bila dibandingkan dengan algoritma simetris.
- Untuk tingkat keamanan sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritma simetris

2.2.3 Algoritma *Block Cipher*

Block cipher merupakan suatu algoritma kriptografi yang beroperasi dalam bentuk blok bit yang mana input dan outputnya berupa satu blok, dan setiap blok terdiri dari beberapa bit (1 blok dapat terdiri dari 32bit, 64 bit atau 128 bit). (Ariyus, 2006: 58). Input *plaintext* biasanya dibagi menjadi beberapa blok, misalnya 32 bit untuk setiap bloknya, jika input kurang dari jumlah tersebut maka akan dilakukan penambahan bit atau yang sering disebut*padding,*sehingga blok tersebut menjadi 32 bit. Proses enkripsi dilakukan dalam blok bit *plaintext* menggunakan kunci yang berukuran sama dengan ukuran blok *plaintext* dan menghasilkan*ciphertext* yang sama dengan blok *plaintext*.

Block Cipher terbagi menjadi empat mode operasi yaitu :

1. *Mode Electronic Code Book* (ECB)

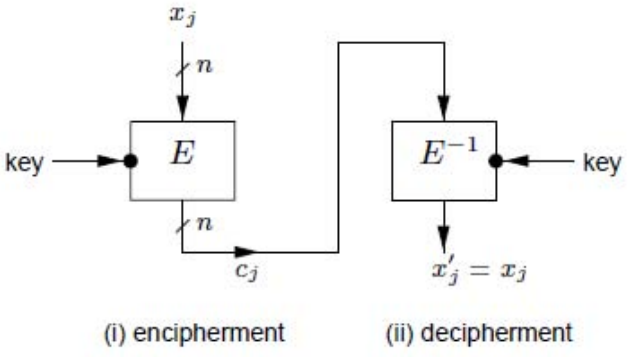
   Ariyus (2006: 58) mengemukakan ECB merupakan suatu *block cipher* yang panjang dibagi dalam bentuk *sequencebinary* menjadi satu blok tanpa mempengaruhi blok-blok yang lain, satu blok terdiri dari 64 bit atau 128 bit, setiap blok merupakan bagian dari pesan yang dienkripsi.

   Secara matematis proses enkripsi dan dekripsi pada mode ECB yaitu:

   Proses enkripsi : $C_i = E_k(P_i)$ [2.4]

   dan proses dekripsi : $P_i = D_k (C_i)$ [2.5]

   Dalam hal ini $P_i$dan$C_i$merupakan blok *plaintext* dan *ciphertext*ke-i. Mode blok ini merupakan mode *block cipher* yang sederhana, dan mempunyai keunggulan yaitu ketika kerusakan terjadi pada satu blok maka tidak akan mempengaruhi blok yang lainnya. Namun kelemahan dari mode ini adalah mudahnya *attacker* dalam membaca pola mode blok ini. Gambar 2.4 merupakan skema dari *Mode Electronic Code Book :*



Gambar 2.4. Skema enkripsi dekripsi pada mode ECB

Dikutip dari : Handbook of Applied Cryptography, 1997, halaman 229

2. *Mode Cipher Block Chaining* (CBC)

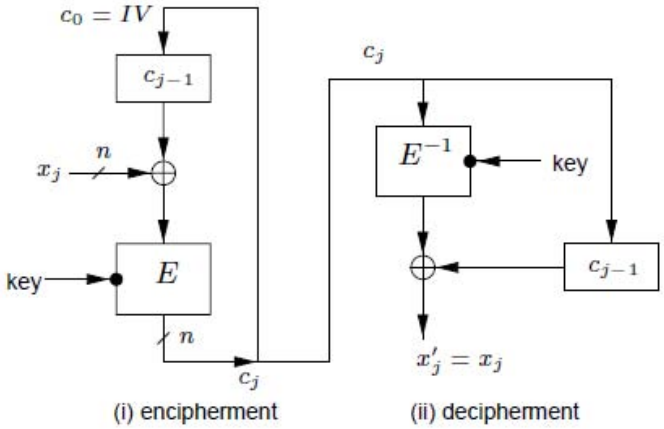
   Ariyus (2006: 59) mengemukakan bahwa sistem dari *Mode Cipher BlockChaining* (CBC) adalah *plaintext*yang samaakan dienkripsi ke dalam bentuk*ciphertext* yang berbeda, disebabkan *block cipher*yang satu tidak berhubungandengan *block cipher*yang lain. Melainkan tergantung pada *ciphertext*yangsebelumnya.

   Secara matematis proses enkripsi dan dekripsi pada mode CBC yaitu:

   Proses enkripsi : $C_i = E_k(P_i \oplus C_{i-1})$ [2.6]

   dan proses dekripsi : $P_i = D_k(C_i) \oplus C_{i-1}$ [2.7]

   Tingkat keamanan dari mode CBC ini lebih rumit dari pada EBC karena hasil dari proses enkripsi dari blok sebelumnya mempengaruhi hasil enkripsi blok selanjutnya, jadi tiap blok *ciphertext* tidak hanya bergantung pada blok *plaintext*-nya melainkan juga tergantung pada *ciphertext* sebelumnya. Maka dari itu untuk *plaintext* yang sama belum tentu menghasilkan *ciphertext* yang sama pula. Namun kelemahan dari mode ini adalah ketika terjadi kesalahan 1 bit saja pada salah satu blok maka akan terjadi kesalahan pada blok-blok selanjutnya. Gambar 2.5 merupakan skema dari *Mode Cipher Block Chaining:*



Gambar 2.5. Skema enkripsi dekripsi pada mode CBC

Dikutip dari : Handbook of Applied Cryptography, 1997, halaman 229

3. *Mode Cipher Feed Back* (CFB)

   Mode ini tidak memerlukan tambahan bit (*padding*) karena jumlah panjang blok sama dengan jumlah panjang *plaintexs*-nya, biasanya metode ini digunakan pada *stream cipher*. Dalam mode ini juga melibatkan penggunaan *initializing vector* (IV).

   Secara matematis proses enkripsi dan dekripsi pada mode CFB yaitu :

   Proses enkripsi :

   $C_i = P_i \oplus MSB_m(E_k(X_i))$ [2.8]

   $X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$ [2.9]

   dan proses dekripsi :

   $P_i = C_i \oplus MSB_m(D_k(X_i))$ [2.10]

   $X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$ [2.11]

   Keterangan:

   X_i = isi antrian dengan X1 adalah IV

   E = fungsi enkripsi

   K = kunci

   M = panjang blok enkripsi

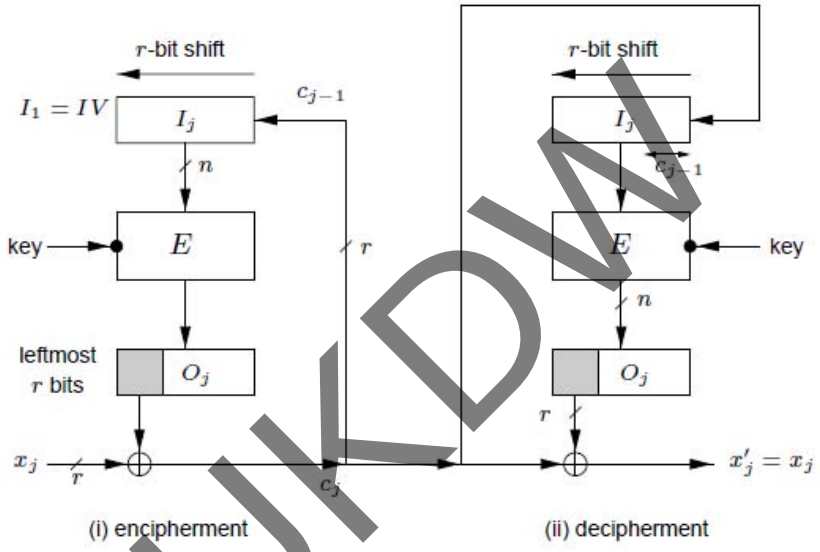
   N = panjang unit enkripsi

   || = operator penyambungan (concatenation)

MSB = *Most Significant Bit*

LSB = *Least Significant Bit*

Kelemahan dari mode ini adalah ketika satu *block cipher* terjadi kesalahan maka kesalahan untuk semua blok yang lain, karena satu blok dan blok yang lain saling berhubungan. Gambar 2.6 merupakan skema dari *Mode Cipher Feed Back :*



Gambar 2.6. Skema enkripsi dekripsi pada mode CFB

Dikutip dari : Handbook of Applied Cryptography, 1997, halaman 229

4. *Mode Output Feed Back* (OFB)

Ariyus (2006: 62) mengemukakan *Mode Output Feed Back* (OFB) tidak mempengaruhi blok yang lain jika terjadi *error*, satu bit yang *error* pada*ciphertext* hanya akan mempengaruhi satu bit *plaintext* pada terjadinya prosesdekripsi. Cara kerja mode OFB mirip dengan mode CFB, kecuali n-bit dari hasil enkripsinya.

Secara matematis proses enkripsi dan dekripsi pada mode CFB yaitu :

Proses enkripsi :

$C_i = P_i \oplus MSB_m(E_k(X_i))$ [2.12]

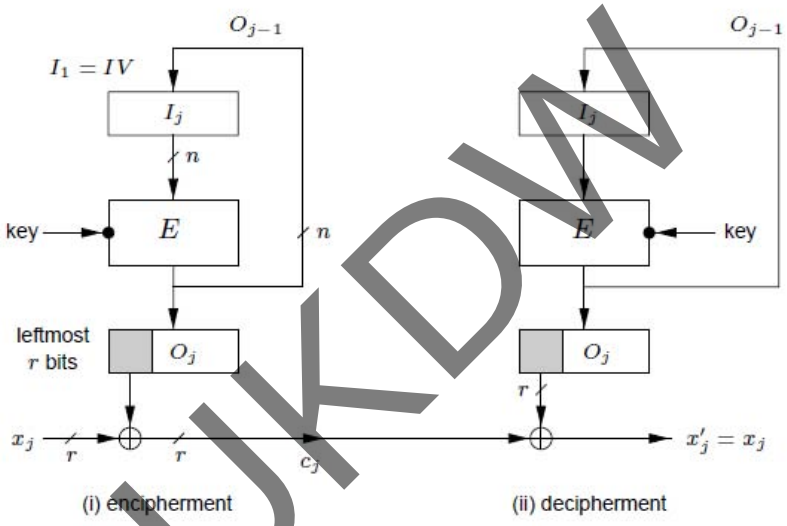
$X_{i+1} = LSB_{m-n}(X_i) \,||\, MSB_m(E_k(X_i))$ [2.13]

dan proses dekripsi :

$P_i = C_i \oplus MSB_m(D_k(X_i))$ [2.14]

$X_{i+1} = LSB_{m-n}(X_i) \,||\, MSB_m(E_k(X_i))$ [2.15]

Mode ini sangat cocok untuk sistem analog seperti suara (*voice*) atau video, karena jika satu bit yang *error* tidak merusak semua blok yang ada. Gambar 2.7 merupakan skema dari *Mode Output Feed Back:*



Gambar 2.7. Skema enkripsi dekripsi pada mode OFB

Dikutip dari : Handbook of Applied Cryptography, 1997, halaman 229

2.2.4 Twofish

2.2.2.1 Deskripsi Algoritma *Twofish*

Dalam ilmu kriptografi modern dikenal dengan algortima *Twofish* yang merupakan sebuah algoritma *block cipher*. Algoritma ini dirancang oleh Bruce Schneier dan "*Counterpane Labs USA*" pada tahun 1997 kemudian dipublikasikan pada tahun 1998. Algoritma ini merupakan salah satu dari lima kandidat utama *Advanced Encryption Standard* (AES) oleh *National Institute of Standards and Technology* (NIST). Algoritma *Twofish* tidak dipatenkan, dan kode sumbernya dapat disalin serta bebas lisensi untuk semua pengguna. Bruce Schneier beserta

timnya di *Counterpane Labs USA* telah mencoba untuk memberikan penamaan yang diawali dengan kata *blow* atau diakhiri dengan kata *fish*. Nama *Twofish* diberikan untuk beberapa alasan, yaitu menjadi tradisi untuk memberikan penamaan yang menggunakan makhluk laut, atau animal pada umumnya dan tim yang tergabung dalam *Counterpane Labs USA* tidak menyetujui untuk memberikan nama *Blowfish* II. Selain itu, fungsi putaran di dalam algoritma *Twofish* diperoleh dari dua fungsi putaran algoritma *Blowfish* secara paralel. (Schneier et al, 1999).

Perancangan algoritma *Twofish* dilakukan dengan memperhatikan kriteria-kriteria yang diajukan *National Institute of Standards and Technology* (NIST) untuk kompetisi *Advanced Encryption Standard* (AES). Tujuan perancangan *Twofish* yang selaras dengan kriteria NIST untuk AES adalah sebagai berikut:

1. Merupakan blok kode dengan kunci simetri 128 bit.
2. Panjang kunci yang digunakan adaah 128 bit, 192 bit, dan 256 bit. Dan tidak mempunyai kunci lemah .
3. Efisiensi algoritma, baik pada Intel Pentium Pro dan perangkat lunak lainnyaserta *platform* perangkat keras.
4. Rancangan yang fleksibel. Dapat menerima panjang kunci tambahan, dapat diterapkan pada *platform*dan aplikasi yang sangat variatif, serta cocok untuk aliran kode, fungsi Hash, dan MAC.
5. Rancangan yang sederhana agar memudahkan proses analisis dan implementasi algoritma

2.2.2.2 Blok Pembangun Algoritma *Twofish*

Algoritma *Twofish* dibangun dari beberapa blok algoritma yaitu :

1. **Jaringan Feistel(*Feistel Network*)**

   Schneier (1998: 4) menjelaskan bahwa jaringan feistel adalahmetode umum untuk mentransformasi fungsi apapun (fungsi *f*) kedalamsebuah permutasi. Jaringan Feistel ditemukan oleh Horst Feistel dalam desainnya tentang Lucifer, dan dipopulerkan oleh DES. FEAL, GOST, *Khufu and Khafre*, LOKI, CAST-128, *Blowfish*, dan RC5 adalah algoritma kriptografi

yang menggunakan jaringan feistel. Bagian terpenting dari jaringan fietsel adalah fungsi *f* yang mana merupakan sebuah pemetaan*string*input menjadi *string*outputberdasarkan kunci yang digunakan. Fungsi *f* selalu tidak linear dan kemungkinan tidak surjektif :

$$F : \{0,1\}^{n/2} \, x \, \{0,1\}^{N} \rightarrow \{0,1\}^{n/2} \qquad [2.16]$$

Dimana n adalah ukuran blok dari jaringan feistel dan F adalah fungsi yang menerima $n/2$ bit dari blok dan *N* bit kunci sebagai input lalu menghasilkan output $n/2$ bit. Setiap putaran, blok sumber dijadikan sebagai input dari fungsi *f*, dan hasilnya di-XOR-kan dengan blok target, kemudian blok sumber dan blok target dipertukarkan sebelum putaran selanjutnya. *Twofish* menggunakan 16 putaranjaringan fietsel dengan fungsi *f* yang bijektif.

2. **Kotak-S (*S-boxes*)**

   Kotak-S adalah operasi *subsitusi*yang berbasis tabel *non-linear* yang digunakan pada kebanyakan *block cipher*. Kotak-S mempunyai ukuran input dan output yang bervariasi. Kotak-S dapat dibuat, dihitung secara acak atau dihitung dengan algoritma. Lucifer adalah algoritma yang pertama kali menggunakan kotak-S, lalu DES dan diikuti algoritma yang lainnya. *Twofish* menggunakan empat buah 8x8 bit kotak-S yang berbeda, bijektif dan tergantung pada sebuah kunci. Kotak-S ini dibuatmenggunakan dua permutasi 8x8 bit dan material kunci. (Schneier et al, 1998: 5).

3. **Matriks MDS**

   KodeMDS(*Maximum Distance Separable*)pada sebuah *field* adalah pemetaan linier dari elemen *field* x ke elemen *field* y, dan menghasilkan elemen vektor komposit x + y dengan ketentuan bahwa jumlah minimum dari elemen bukan nol pada vektor bukan nol paling sedikit y + 1 . Dengan kata lain, jumlahelemen yang berbeda diantara dua vektor yang berbeda yang dihasilkan oleh pemetaan MDSpaling sedikit y + 1. Dapat dilihat bahwa tidak ada pemetaan yang dapat memiliki jarak pisah yang lebih besar diantara dua vektor yang berbeda, maka dari itu disebut *Maximum*

distance separable (MDS). Pemetaan MDS bisa direpresentasikan oleh matriks MDS yang terdiri dari $x * y$ elemen. Algoritma*Twofish* menggunakan matriks MDS tunggal 4 x 4. (Schneier et al, 1998: 5).

4. Transformasi *Pseudo-Hadamard* (PHT)

Transformasi *Pseudo-Hadamard* (PHT) adalah sebuah operasi pencampuran sederhana yangbekerja dengan cepat dalam perangkat lunak. 32 bit PHT yang menggunakan dua input dapat didefinisikan sebagai berikut :

$$A_0 = a + b \bmod 2^{32} \quad [2.17]$$

$$B_0 = a + 2b \bmod 2^{32} \quad [2.18]$$

SAFER manggunakan 8 bitPHT untuk prosesdifusi. Algoritma *Twofish* menggunakan 32 bit PHT untuk mencampur 2 output 32-bit dari fungsi *g*.(Schneier et al, 1998: 5).

5. *Whitening*

Schneier (1998: 5) menjelaskan bahwa *whitening* merupakan sebuah teknik meng-XOR-kan material kunci sebelum putaran pertama dan setelahputaran terakhir. Whitening digunakan oleh Merkle pada algoritma *Khufu and Khafre* dan ditemukan oleh Rivers untuk algoritma DES-X. *Whitening* meningkatkan kesulitan serangan pencarian kunci terhadap *ciphertext*, dengan menyembunyikan input spesifik pada putaran pertama dan putaran terakhir dari fungsi *f*. *Twofish* meng-XOR-kan 128 bit sub kunci sebelum putaran pertama jaringan feistel dan 128 bit sub kunci lainnya setelah putaran terakhir jaringan feistel. Sub kunci yang digunakan dihitung dengan cara yang sama seperti sub kunci lainnya, tetapi tidak digunakan pada *cipher* lain.

6. Penjadwalan Kunci (*Key Schedule*)

Penjadwalan kunciadalah suatu proses mengubah bit-bit kunci menjadi beberapa sub kunci yang digunakan *cipher*pada setiap putaran.*Twofish* memerlukan banyak materialkunci, dan memiliki penjadwalan kunci yang rumit. Untuk menfasilitasi analisis, penjadwalan kunci menggunakan primitif yang sama seperti fungsi putaran. (Schneier et al, 1998:5). Jadi

secara singkat, penjadwalan kunci adalah prosespengacakan kunci untuk melakukan proses enkripsi sehingga tingkatkerumitannya menjadi tinggi.

2.2.2.3 Algoritma *Twofish*

*Twofish*merupakan suatu algoritma kriptografi *block cipher* dengan menggunakan kunci simetris dengan panjang setiap bloknya adalah 128 bit. Kunci yang dapat diterima oleh algoritma ini adalah 128 bit, 192 bit, atau 256 bit. *Twofish* menggunakan 16 putaran jaringanfeistel dengan tambahan *input whitening*dan output *whitening*.*Twofish* memanfaatkan teknik manipulasi bit, kotak permutasi, jaringan feistel, pemutaran ulang dengan pergiliran kunci sebanyak 16 kali putaran, transformasi *Pseudo-Hadamard*, ekspansi dan filter, serta kotak MDS.

Langkah-langkah dari algoritma *Twofish* adalah sebagai berikut :

1. Input satu blok *plaintext*sebesar 128 bit. Blok tersebut dibagi menjadi 4 buah sub blok yang masing-masing memiliki panjang bloknya sebesar 32 bit. Pembagian tersebut menggunakan konversi *little-Endian.*

$$P_i = \sum_{j=0}^{3} p_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 3$$

[2.19]

Dua subblok menjadi bagian kiri (A dan B) dan dua lainnya menjadi sub blok bagian kanan (C dan D).

2. Masing-masing sub blok tersebut melalui proses *whitening* dengan meng-XOR-kan dengan kunci ekspansi k_0,k_1, k_2, dan k_3.

$$R_{0,i} = P_i \oplus k_i \qquad i = 0, \ldots, 3$$

[2.20]

3. Hasil dari proses *input whitening*, 2 sub blok bagian kiri (A dan B) dijadikan menjadi input dari fungsi *f*(salah satu blok (B) tersebut digeser terlebih dahulu sejauh 8 bit (ROL_8)). Fungsi *f* terdiri dari beberapa bagian yaitu fungsi *g*, transformasi *Pseudo-Hadamard* (PHT), dan penambahan dengan sub kunci yang diperluas.

4. Fungsi *g* terdiri dari 4 buah kotak-S dan matriks MDS. Input fungsi *f* (blok A dan B) dimasukkan ke dalam fungsi *g*. Input tersebut diproses dengan kotak-S terlebih dahulu lalu hasilnyadilakukan pencampuran linear menggunakan matriks MDS.
5. Hasil dari fungsi *g* dimasukkan ke dalam fungsi transformasi *Pseudo-Hadamard* (PHT), kemudian hasilnya ditambahkan dengan 2 buah 32 bit sub kunci dari kunci yang diperluas (k_{2r+8} dan k_{2r+9}).
6. Kedua buah hasil dari fungsi *f* tersebut di XOR-kan dengan 2 sub blok bagian kanan (blok C dan D). Salah hasil fungsi *f* (A) di XOR-kan dengan satu blok bagian kanan (C) lalu hasilnya digeser ke kanan sejauh 1 bit (ROR). Dan blok lain (D) pada bagian kanan digeser terlebih dahulu sejauh 1 bit (ROL), kemudian hasilnya di XOR-kan dengan blok hasil lainnya (B) dari fungsi *f*.
7. Lalu 2 buah 32 bit kiri dan kanan saling dipertukarkan (A dan B dipertukarkan dengan C dan D).
8. Ulangi langkah ke 3 sampai ke 7 (Jaringan Feistel) diulangi sampai dengan 16 kali putaran.

$$(F_{r,0},\ F_{r,1}) = \mathrm{F}\ (R_{r,0},\ R_{r,1}, \mathrm{r}) \quad [2.21]$$

$$R_{r+1,0} = \mathrm{ROR}\ (R_{r,2} \oplus F_{r,0}, 1) \quad [2.22]$$

$$R_{r+1,1} = \mathrm{ROL}\ (R_{r,3}, 1) \oplus F_{r,1} \quad [2.23]$$

$$R_{r+1,2} = R_{r,0} \quad [2.24]$$

$$R_{r+1,3} = R_{r,1} \quad [2.25]$$

untuk *r = 0, . . ., 15*.

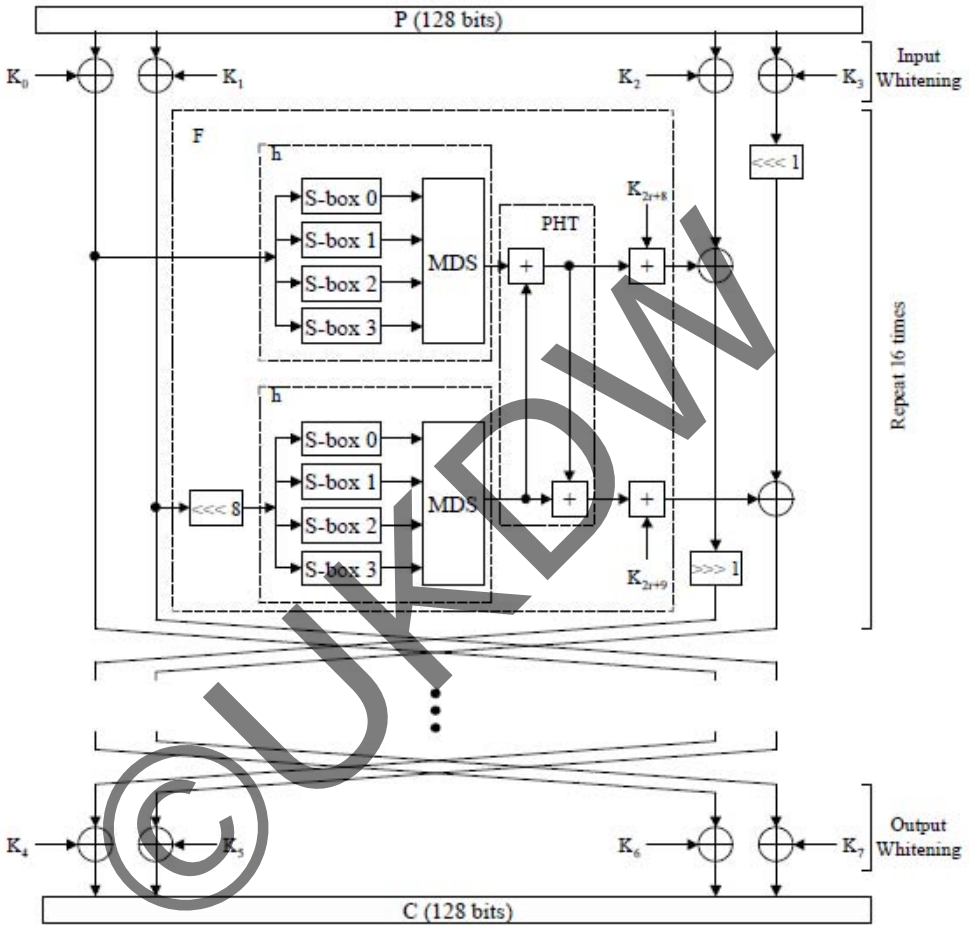
9. Setelah ke empat sub blok tersebut mengalami proses jaringan feistel sebanyak 16 kali putaran, ke empat sub blokdilakukan undo swap yang mana menukarkan kembali blok A dan B dengan C dan D. Setelah itu, hasilnya di lakukan proses output *whitening*.

$$c_i = R_{16,(i+2) mod\ 4} \oplus K_{i+4} \quad i = 0, \ldots, 3 \quad [2.26]$$

10. Empat sub blok dari *ciphertext* ditulis seperti 16 byte $c_0,\ldots,c_{15}$dengan menggunakan konversi *little-Endian.*

$$c_i = \left\lfloor \frac{C_{[i/4]}}{2^{8(i \bmod 4)}} \right\rfloor mod\ 2^8 \qquad i = 0,\ldots,15 \qquad [2.27]$$

Gambar 2.8 berikut ini merupakan diagram dari algoritma *Twofish.*



Gambar 2.8. Struktur Algoritma *Twofish*

Dikutip dari : Chodowiec, P & Gaj, K. (1999*). Implementation of the Twofish Cipher Using FPGA Devices,* halaman 2.

(Schneier et al, 1998: 5).

2.2.2.4 Fungsi f

Fungsi *f* adalah suatu permutasi yang bergantung pada kunci (*key-dependent*) dengan nilai 64 bit. Fungsi ini menerima 3 argumen yaitu, 2 buah input 32 bit R_0 dan R_1 dan nomor putaran yang juga digunakan untuk menentukan subkunci mana yang akan dipakai pada putatan tersebut. R_0dimasukkan ke dalam fungsi *g* dan akan menghasilkan T_0, dan R_1 akan digeser 8 bit ke kanan terlebih dahulu kemudian dimasukkan ke dalam fungsi *g* dan akan menghasilkan T_1. T_0 dan T_1 selanjutkan dikombinasikan ke dalam sebuah fungsi transformasi*Pseudo-Hadamard*(PHT) lalu ditambahkan dengan 2 buah 32 bit sub kunci dari kunci yang diperluas.

$$T_0 = \mathrm{g}(R_0) \quad [2.28]$$

$$T_1 = \mathrm{g}\big(ROL(R_1, 8)\big) \quad [2.29]$$

$$F_0 = (T_0 + T_1 + K_{2r+8}) \bmod 2^{32} \quad [2.30]$$

$$F_1 = (T_0 + T_1 + K_{2r+9}) \bmod 2^{32} \quad [2.31]$$

F_0 dan F_1 merupakan output dari fungsi *f,* yang masing-masing panjangnya 32 bit. Output fungsi *f* akan dipertukarkan dan dimasukkan kembali ke putaran selanjutanya sampai 16 kali putaran (Jaringan Feistel). (Schneier et al, 1998: 7).

2.2.2.5 Fungsi g

Fungsi *g* merupakan jantung dari keseluruan algoritma *Twofish*. 32 bit yang menjadi input dari fungsi *f* dipecah menjadi 4 bagian, masing-masing bagian tersebut memiliki panjang 8 bit. Setiap 8 bit kemudain di proses dengan kotak-S yang bergantung pada vektor S yang di dapat dari kunci.Setiap kotak-S bersifat bijektif, yaitu menerima input sebesar 8 bit dan mengeluarkan output sebesar 8 bit juga. 4 buah 8 bit hasil dari 4 kotak-S kemudiankemudian diinterpretasikan sebagai vektor yang panjangnya 4 di atas *GF* (2^8), dan dikalikan dengan matriks *Most Distance Separable* (MDS) 4x4 (menggunakan bidang *GF* (2^8) untuk perhitungannya). Vektor yang diinterpretasikan menjadi blok 32-bit, yang juga merupakan hasil dari fungsi *g* :

$$x_i = \left\lfloor X/2^{8i} \right\rfloor \bmod 2^8 \qquad i = 0, \ldots, 3 \quad [2.32]$$

$$y_i = s_i[x_i] \qquad i = 0, \ldots, 3 \tag{2.33}$$

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} . & \ldots\ldots\ldots & . \\ & MDS & \\ . & \ldots\ldots\ldots & . \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \tag{2.34}$$

$$Z = \sum_{i=0}^{3} z_i \cdot 2^{8i} \tag{2.35}$$

Di mana $s_i$adalah kotak-S yang bergantung pada vektor S yang di dapat dari kunci dan *Z* adalah output dari fungsi *g*. Matriks MDS yang setiap elemennya ditampilkan sebagai heksadesimal adalah sebagai berikut:

$$\text{MDS} = \begin{pmatrix} 01 & EF\,5B & 5B \\ 5B & EF\,3F & 01 \\ EF & 5B\,01 & EF \\ EF & 01\,EF & 5B \end{pmatrix} \tag{2.36}$$

(Schneier et al, 1998:7)

2.2.2.6 Penjadwalan Kunci (*Key Schedule*)

Pada algoritma *twofish* diperlukan 40 kunci yang diperluas dari kunci yang diinputkan oleh pengguna (k_0, k_1, k_2, …, k_{39}). Penjadwalan kunci pada algoritma *twofish* menghasilkan 40 sub kunci yang diperluas dan 4 buah *S-Boxeskey-dependent* yang mana nantinya digunakan pada fungsi *g*. *Twofish* mendefinisikan kunci dengan panjang *N* = 128, *N* = 192, dan *N* = 256. Untuk ukuran-ukuran kunci yang tidak didefinisikan, kunci akan mengalami penambahan niai nol pada bagian akhir kunci sehingga memenuhi persyaratan kunci yang dibutuhkan (*padding*). Selain itu *twofish* mendefinisikan *k* = *N* / 64 dan juga kunci *M* yang terdiri dari *8k* byte $m_0, \ldots, m_{8k-1}$. Kunci *M* pertama-tama di ubah kedalam *2k* blok bit yang mana berukuran 32 bit :

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 2k-1$$

[2.37]

untuk*i = 0, ..., 2k-1*.

Lalu hasil dari perhitungan diatas dimasukkan ke dalam 2 vektor dari panjang k :

$$M_e = (M_0, M_2, \ldots, M_{2k-2}) \quad [2.38]$$

$$M_o = (M_1, M_3, \ldots, M_{2k-1}) \quad [2.39]$$

Untuk vektor s mempunyai perhitungan :

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \cdot \\ RS & \\ \cdot & \cdot \end{pmatrix} \cdot \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix}$$

[2.40]

$$S_i = \sum_{j=0}^{3} s_{i,j} \cdot 2^{8j}$$

[2.41]

untuk *i = 0, . . ., k-1*, dan

$$S = (S_{k-1}, S_{k-2}, \ldots, S_0) \quad [2.42]$$

Vektor S disusun secara terbalik (*reverse*).

Dalam pemetaan ini, matriks RS di tunjukkan sebagai berikut :

$$\text{MDS} = \begin{pmatrix} 01 & A455 & 875A & 58DB & 9E \\ A4 & 5682 & F31E & C668 & E5 \\ 02 & A1FC & C147 & AE3D & 19 \\ A4 & 5587 & 5A58 & DB9E & 03 \end{pmatrix} \quad [2.43]$$

Vektor *Me, Mo,* dan *S* merupakan factor utama untuk membentuk penjadwalan kunci.Dimana Nilai output *Me* dan *Mo* akan digunakan untuk langkah *ExpandedKey*dan *S* akan digunakan dalam fungsi *g*.

(Schneier et al, 1998:8)

2.2.2.7 Contoh Perhitungan *Twofish*

Sebagai contohnya terdapat sebuah *plainteks* yang mempunyai pesan "aku cinta padamu" dan mempunyai panjang sebesar 16 byte (128 bit). *Plainteks* tersebut mempunyai nilai seperti berikut :

Tabel 2.1.

Tabel *plainteks*.

Char	a	k	u		c	i	n	t
ASCII	97	107	117	32	99	105	110	116
Char	a		p	a	d	a	m	u
ASCII	97	32	112	97	100	97	109	117

Nilai *plainteks* diatas akan dikonversi dengan konversi *little endian* dengan rumus persamaan 2.19. Hasilnya adalah sebagai berikut :

Tabel 2.2.

Tabel hasil konversi *littleendian*.

R_0	R_1	R_2	R_3
544566113	1953392995	1634738273	1970102628

Hasil dari konversi*little endian* akan dilakukan proses *input whitening* dengan meng-xor-kan dengan 4 sub kunci pertama yang diperluas (k_0,k_1,k_2,k_3) sama seperti pada persamaan 2.20.

Tabel 2.3.

Tabel proses *input whitening*.

	R_0	R_1	R_2	R_3
Input	544566113	1953392995	1634738273	1970102628
Sub kunci	115732435	3043150273	1072959771	3517274163
Hasil	647005362	3238830242	1585721722	2764585303

Hasil dari *inputwhiteining* dilakukan proses jaringan feistel sebanyak 16 kali putaran dengan persamaan seperti yang tertera pada sub bab 2.2.4.3 yaitu Persamaan 2.21, 2.22, 2.23, 2.24 dan 2.25. Hasilnya adalah sebagai berikut :

Tabel 2.4.

Tabel proses jaringanfeistel.

Iterasi (r)	R_0	R_1	R_2	R_3	K_{2r+8}	K_{2r+9}
0	647005362	3238830242	1585721722	2764585303	115732435	3043150273
1	3938856177	3324242417	647005362	3238830242	1072959771	3517274163
2	2954775943	3179291326	3938856177	3324242417	2703525541	848325878
3	900600684	3207882727	2954775943	3179291326	3170651288	3793507117
4	2914485192	2823373066	900600684	3207882727	2703525541	848325878
5	1020522609	4102037791	2914485192	2823373066	567519777	2430035447
6	95331966	1498560202	1020522609	4102037791	1664617477	1503341690
7	3662138391	2513085710	95331966	1498560202	2451909503	869933272
8	494841434	1612126245	3662138391	2513085710	2703525541	848325878
9	3139067796	628707199	494841434	1612126245	3497402678	3586494292
10	191349621	514332435	3139067796	628707199	1329380455	1660678226
11	705609236	2908440753	191349621	514332435	2451909503	869933272
12	2721651084	2408414570	705609236	2908440753	567519777	2430035447
13	1317697685	568586814	2721651084	2408414570	1212830135	2116456516
14	1957029023	4292465352	1317697685	568586814	2703525541	848325878
15	1627955235	13497065	1957029023	4292465352	567519777	2430035447
Hasil	2010587585	1787530041	1627955235	13497065		

Hasil dari proses jaringan feistel dilakukan proses *undo-swap* yang mana menukarkan R_0 dan R_1 dengan R_2 dan R_3.

Tabel 2.5.

Tabel proses *undo-swap*.

	R_0	R_1	R_2	R_3
Input	2010587585	1787530041	1627955235	13497065
undo-swap	1627955235	13497065	2010587585	1787530041

Setelah itu akan dilakukan proses *ouput whitening* dengan meng-xor-kan dengan 4 sub kunci pertama yang diperluas (k_4,k_5,k_6,k_7) sama seperti pada persamaan 2.26. Hasilnya adalah sebagai berikut :

Tabel 2.6.

Tabel proses *output whitening*.

	R_0	R_1	R_2	R_3
Input	1627955235	13497065	2010587585	1787530041
Sub kunci	2703525541	848325878	902756799	2272698399
Hasil	3606292324	1478223823	1422285212	2277200630

Langkah terakhir adalah mengkonversi hasil dari output *whitening* dengan konversi *little endian* untuk mendapatkan*chiperteks* yang mempunyai panjang sebesar 16 byte (128 bit) dengan persamaan 2.27. Dan hasilnya adalah sebagai berikut :

Tabel 2.7.

Tabel *chiperteks*.

ASCII	100	167	243	214	207	231	27	88
Char	d	§	ó	Ö	Ï	ç	**ESC**	X
ASCII	156	89	198	84	246	82	187	135
Char	œ	Y	Æ	T	ö	R	»	‡

Plainteks“aku cinta padamu” mengalami proses enkripsi twofish menghasilkan *chiperteks* seperti pada tabel 2.7 yang mana pada tiap bit nya tidak hanya menghasilkan nilai “ASCII *PrintableCharacters”*(32 - 126) namun juga menghasilkan nilai “ASCII *ControlCharacters”*(1 - 31, 127)dan nilai “*The extended ASCII codes*” (128 - 255). Maka dari itu, jika *chiperteks* pada tabel 2.7 di tulis dalam bentuk file maka akan menjadi “d§óÖÏçXœYÆTöR»‡”.

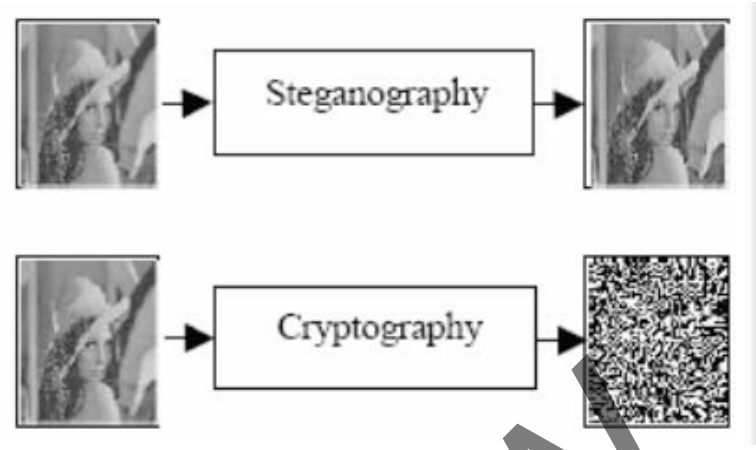
2.2.5 Steganografi

Kata steganografi berasal dari bahasa Yunani, yaitu dari kata “*Steganif*” (tersembunyi) dan “*Graptos*” (tulisan). Steganografi di dunia modern biasanya mengacu pada informasi atau suatu arsip yang telah disembunyikan ke dalam suatu arsip citra digital, audio atau video.

Steganografi adalah suatu teknik untuk menyembunyikan informasi yang bersifat pribadi dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya. Media yang digunakan pada umumnya merupakan suatu media yang berbeda dengan media pembawa informasi rahasia, dimana disinilah fungsi dari teknik steganografi yaitu teknik penyamanaran menggunakan media lain yang berbeda sehingga informasi rahasia dalam media awal tidak terlihat secara jelas [Waheed, 2000].

Steganografi berbeda dengan kriptografiyang dengan mudah dideteksi dengan panca indera manusia. Perbedaan yang mendasar antara kedua teknik

tersebut adalah informasi yang di lindungi dengan teknik steganografi tidak nampak pada indera penglihatan manusia. Gambar 2.9 yang menunjukkan bagaimana perbedaan antara steganografi dan kriptografi.



Gambar 2.9. Perbedaan Steganografi dengan Kriptografi

Dikutip dari : Aditya, Y, et al.(2010*). Studi Pustaka untuk Steganografi dengan Beberapa Metode,* hlm G-33

Teknik Steganografi yang digunakan dalam dunia modern seperti dewasa ini sudah amat beragam, mulai dari algoritma yang digunakan sampai pada media yang digunaknnya. Beberapa contoh media penyisipan pesan rahasia yang digunakan dalam teknik Steganografi antara lain :

1. **Teks**

   Dalam algoritma steganografi yang menggunakan teks sebagai media penyisipannya biasanya digunakan teknik NLP sehingga teks yang telah disisipi pesan rahasia tidak akan mencurigakan untuk orang lain yang melihatnya.

2. **Audio**

   Format ini pun sering dipilih karena biasanya berkas dengan format ini berukuran relatif besar. Sehingga dapat menampung pesan rahasia dalam jumlah yang besar pula.

3. **Citra**

   Format pun paling sering digunakan, karena format ini merupakan salah satu format file yang sering dipertukarkan dalam dunia Internet. Alasan

lainnya adalah banyaknya tersedia algoritma steganografi untuk media penampung yang berupa citra.

4. **Video**

   Format ini memang merupakan format dengan ukuran file yang relatif sangat besar namun jarang digunakan karena ukurannya yang terlalu besar sehingga mengurangi kepraktisannya dan juga kurangnya algoritma yang mendukung format ini.

Penyembunyian pesan rahasia ke dalam media penampung pasti mengubah kualitas media tersebut. Kriteria yang harus diperhatikan dalam penyembunyian pesan adalah (Munir, 2006, hal: 307):

1. ***Imperceptibility***

   Keberadaan pesan rahasia tidak dapat dipersepsi oleh panca indera manusia. Misalnya, jika *covertext* berupa citra, maka penyisipan pesan membuat citra *stegotext*sulit dibedakan oleh mata dengan citra *covertext*-nya. Jika *covertext* berupa audio (misalnya berkas mp3, wav, midi, dan sebagainya), maka indera telinga tidak dapat mendeteksi perubahan pada audio *stegotext*-nya.

2. ***Fidelity***

   Mutu media penampung tidak berubah banyak akibat penyisipan. Perubahan tersebut tidak dapat dipersepsi oleh panca indera manusia. Misalnya, jika *covertext* berupa citra, maka penyisipan pesan membuat citra *stegotext* sukar dibedakan oleh mata dengan citra *covertext*-nya. Jika *covertext* berupa audio (misalnya berkas mp3, wav, midi, dan sebagainya), maka audio *stegotext* tidak rusak dan indera telinga tidak dapat mendeteksi perubahan tersebut.

3. ***Recovery***

   Pesan rahasia yang telah disisipkan ke dalam media penampung harus Pesan yang disembunyikan harus dapat diungkapkan kembali (*reveal*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu pesan rahasia di dalam *stegotext* harus dapat diambil kembali untuk digunakan lebih lanjut.

2.2.6 *Least Significant Bit* (LSB)

Salah satu metode steganografi yang banyakdigunakan adalah metode modifikasi LSB (*Least Significant Bit*). Metode LSB tergolong metode yang menggunakan tekniksubstitusi dan berbasis media (*media-based steganography*).

Metode LSB menyembunyikan data atau pesan rahasia dalam *pixel-pixel* yang tidak signifikan dari sebuah citra digital. Pengubahan nilai *pixel-pixel* yang tidak signifikan pada dasarnya memberikan pengaruh pada berkas citra digital, tetapi karena perubahan yang terjadi sangat kecil, sehingga indra penglihatan manusia tidak dapat menyadari perubahan yang terjadi pada citra digital tersebut. Sebagai contohnya terdapat bit pada citra digital dengan ukuran 3 *pixel*sebagai berikut :

0011111	11101001	11001000
0011111	11001000	11101001
1100000	00100111	11101001

Pesan yang akan disisipkan adalah sebuah karakter ‘A’, karakter tersebut memiliki nilai biner 01000001, maka *pixel-pixel* pada citra digital tersebut akan dirubah menjadi seperti berikut:

0011110	11101001	11001000
0011110	11001000	11101000
1100000	00100111	11101001

Perubahan yang tidak signifikan seperti contoh diatas tidak akan tertangkap oleh indera penglihatan manusia.

Pada contoh diatas penggantian *pixel* yang tidak signifikan dilakukan secara terurut akan tetapi penggantian *pixel* yang tidak signifikan juga dapat dilakukan secara tak terutut. Hal tersebut dapat meningkatan kemanan data atau pesan rahasia yang disembunyikan (*imperceptability*). Penggantian *pixel* dapat dilakukan bisa mulai dari tengah atau dari titik lain pada citra digital yang memungkinkan untuk menyimpan seluruh informasi rahasia tanpa menimbulkan permasalahan saat pengungkapan data.

Least Significant Bit mudah diterapkan, tetapi berkas stego hasil metode ini mudah rusak (dirusak). Perubahan seditkit pada berkas stego memungkinkan akan merusak informasi rahasia yang tersimpan di dalamnya.

Disamping kemungkinan kerusakan informasi yang tersimpan dalam berkas stego akibat perubahan pada berkas stego, steganografi dengan metode *Least Significant Bit* juga hanya mampu menyimpan informasi dengan ukuran yang sangat terbatas. Misalnya kita akan menyembunyikan sebuah data yang mempunyai ukuran 50 bit pada sebuah citra berwarna (24-bit, R=8-bit, G=8-bit, B=8-bit). Setiap *pixel* dalam citra tersebut dapat menyimbat 3 bit informasi dari data yang akan disembunyikan. Sehingga jika dikalkulasikan dibutuhkan setidaknya citra berukuran 17 bit *pixel* atau setara 17 x 3 x 8 = 408 bit (sekitar 8 kali lipat dari ukuran bit yang akan disembunyikan). Jadi bisa disimpulkan bahwa suatu citra 24-bit jika digunakan menjadi media untuk menyembunyikan informasi rahasia hanya dapat menampung informasi maximal seperdelapan (1/8) dari ukuran citra penampung.

BAB 3
PERANCANGAN SISTEM

3.1 Alat Penelitian

3.1.1 Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pembangunan sistem dan penilitian adalah sebagai berikut :

- Sistem Operasi Windows 7 Ultimate 64 bit
- Microsoft Visual Studio 2010 VB .NET
- Visual Binary Diff 3.0_beta4 (By Christopher J. Madsen)

3.1.2 Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem adalah sebuah notebook dengan spesifikasi sebagai berikut :

- Processor Intel Pentium P6200 2.13 GHz
- Memory4096MB
- Harddisk 320 GB

3.2 Perancangan Proses

Dalam penelitian ini, penulis merancang sebuah sistem yang dapat mengamankan sebuah pesan dengan mengombinasikan algoritma kriptografi dan steganografi yaitu *Twofish* dan *Least Significant Bit* (LSB).

3.2.1 Algoritma Enkripsi dan Penyisipan Pesan

- Input :
  1. Fileteksyang memiliki format .txt.
  2. Citra digital yang memiliki format BMP atau PNG.

- Proses :
  1. Pilih fileteksyang akan dilakukan proses enkripsi dan disisipkan.
  2. Pilih filecitra digital yang digunakan sebagai media penyisipan pesan.Setelah itu dilakukan validasi terhadap filecitra digital tersebut, apakah filecitra tersebut valid atau tidak? Jika valid maka lanjut ke langkah berikutnya, tetapi jika tidak valid program meminta untuk memilih filecitra lagi.
  3. Masukkan kunci untuk proses enkripsi sebanyak 2 kali. Kunci tersebut akan dilakukan proses validasi, apakah kunci sesuai dengan ketentuan atau tidak dan ke dua kunci yang dimasukkan juga sama atau tidak? Jika valid maka lanjut ke langkah berikutnya, tetapi jika tidak valid program meminta untuk memasukkan kunci lagi sampai memenuhi persyaratan kunci yang dapat digunakan.
  4. Program melakukan proses enkripsi (*twofish*) dan penyisipan pesan (LSB).
  5. Jika proses enkripsi dan penyisipan pesan berhasil maka akan tercipta file citra baru yang terlah disisipi file teks yang terenkripsi.
- Output : Filecitra digital yang berektensi sesuai dengan citra digital yang dipilih sebgai media penyisipan pesan.

Flowchart enkripsi dan penyisipan pesan yang dilakukan sistem secara umum dapat dilihat pada Gambar 3.1

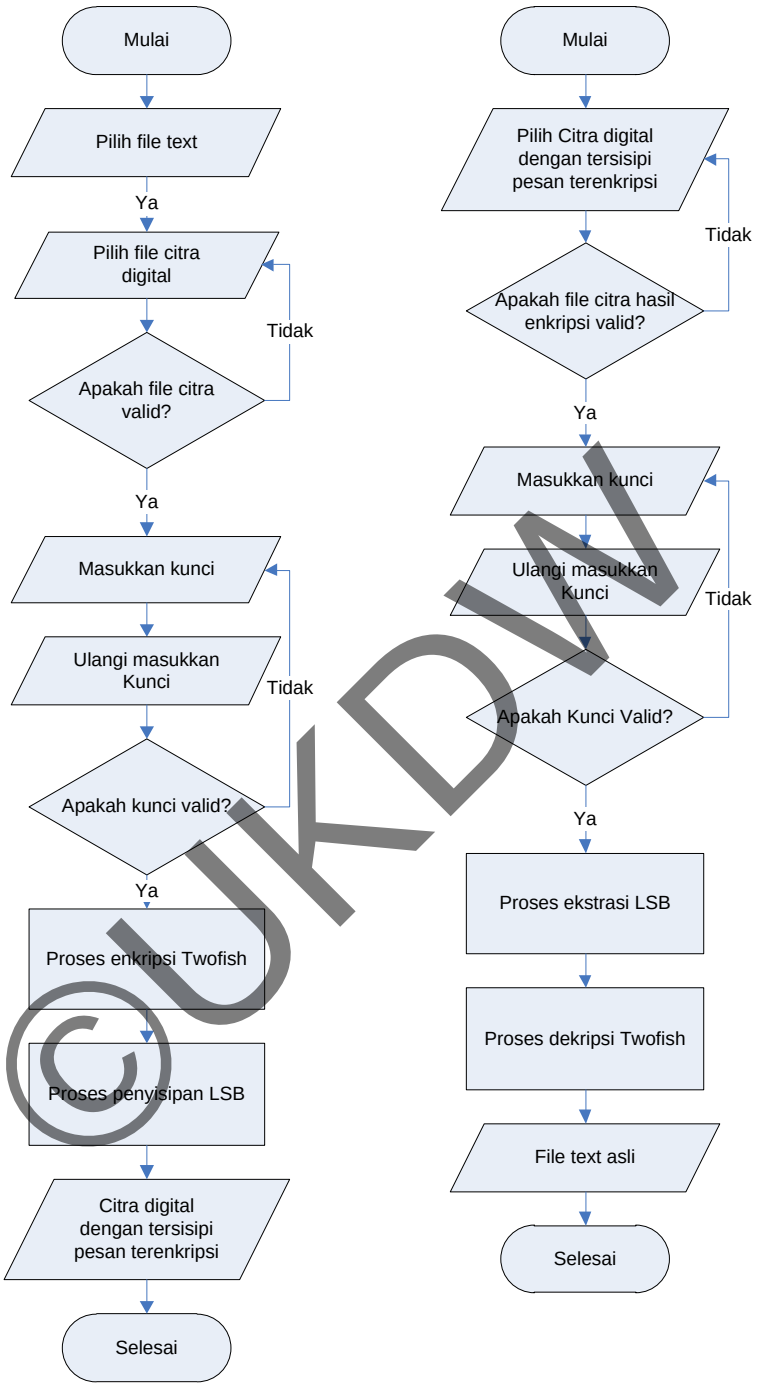
3.2.2 Algoritma Ektrasi dan Dekripsi Pesan

- Input :
  1. Citra digital yang sudah disisipi pesan yang terenkripsi
- Proses :
  1. Pilih filecitra digital yang telah disisipi pesan yang sudah dienkripsi.Setelah itu dilakukan validasi terhadap filecitra digital tersebut, apakah filecitratersebut valid atau tidak? Jika valid maka lanjut ke langkah berikutnya, tetapi jika tidak valid program meminta untuk memilih filecitra digital lagi.

2. Masukkan kunci yang digunakan pada proses enkripsi sebelumnya sebanyak 2 kali. Kunci tersebut akan dilakukan proses validasi, apakah kunci sesuai dengan ketentuan atau tidak dan ke dua kunci yang dimasukkan juga sama atau tidak? Jika valid maka lanjut ke langkah berikutnya, tetapi jika tidak valid program meminta untuk memasukkan kunci lagi sampai memenuhi persyaratan kunci yang dapat digunakan.
3. Program melakukan proses ekstrasi (LSB) dan dekripsi pesan (*twofish*).
4. Jika proses ekstrasi dan dekripsi pesan berhasil maka akan muncul *file text* asli yang disembunyikan sebelumnya pada lokasi penyimpanan *file* citra digital yang dipilih.

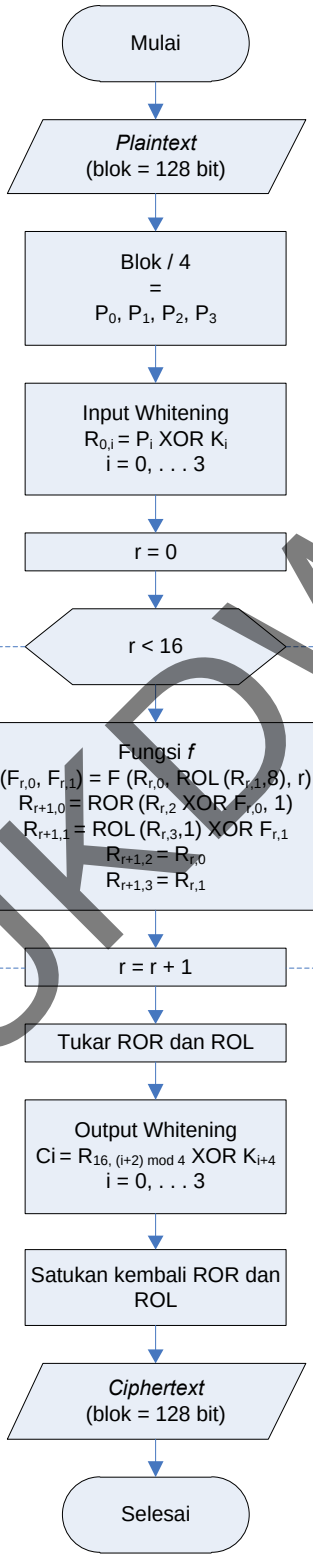
- Output : Fileteksyang isinyasama dengan fileaslinya sebelum dilakukan proses enkripsi dan penyisipan.

Flowchart ekstrasi dan dekripsi pesan yang dilakukan sistem secara umum dapat dilihat pada Gambar 3.1



Gambar 3.1. Flowchart Penyembunyian Pesan dan Pengambilan Pesan

3.2.3 Flowchart Enkripsi Algoritma *Twofish*

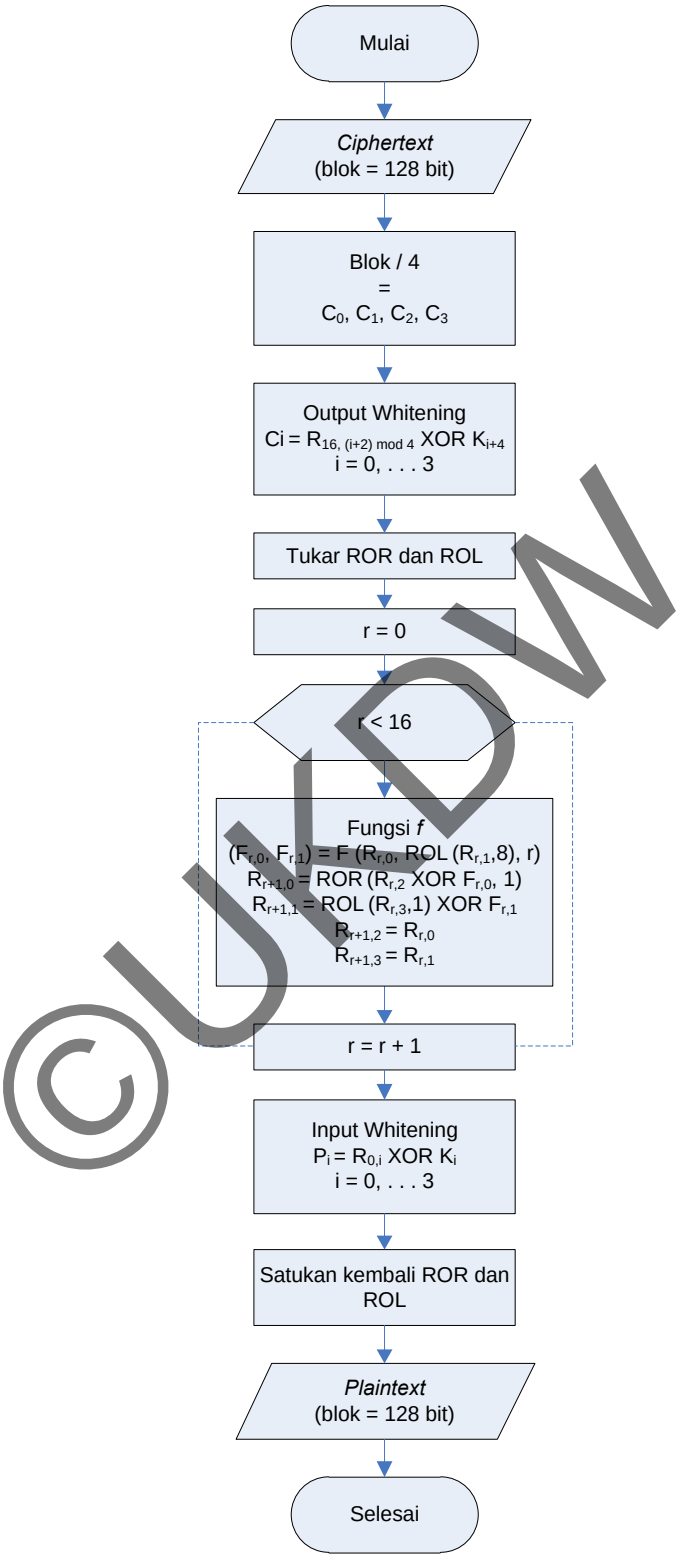


Gambar 3.2. Flowchart Enkripsi *Twofish*

Dari Gambar 3.2 *Flowchart* enrkipsi algoritma *twofish* dapat dijelaskan sebagai berikut :

1. Proses enkripsi dimulai dengan input *plaintext*dengan ukuran blok 128 bit.
2. *Plaintext* tersebut dibagi menjadi 4 blok 32 bit yaitu P_0, P_1, P_2dan P_3.
3. Ke empat blok tersebut di-XOR-kan dengan 4 sub kunci dari kunci yang diperluas (*input whitening*).
4. Hasil dari *input whitening* akan melalui proses fungsi *f*sebanyak 16 kali putaran. Pada setiap putaran kedua blok pertama dijadikan sebgai input dari fungsi *f* (blok ke dua dirotasikan ke kiri sejauh 8 bit terlebih dahulu), dan blok ketiga dilakukan operasi XOR dengan hasil fungsi *f* yang pertama lalu dirotasikan ke kanan sejauh 1 bit, dan blok ke empat dirotasikan ke kiri sejauh 1 bit terlebih dahulu selanjutnya di XOR kan dengan hasil ke dua dari fungsi *f*. Blok pertama dan ke tiga saling dipertukarkan untuk digunakan pada putaran selanjutnya begitu juga blok ke dua dan ke empat.
5. Hasil dari pertukaran terakhir dipertukarkan lagi (ROR dipertukarkan dengan ROL)
6. Ke empat blok hasil dari pertukaran pada langkah 5 di XOR kan dengan 4 sub kunci dari kunci yang diperluas (*output whitening*).
7. Hasil ROR dan ROL dari proses *output whitening* disatukan kembali menjadi 1 blok bagian 128 bit.
8. Dan menghasilkan *ciphertext*128 bit hasil dari proses enkripsi algoritma *twofish*.
9. Selesai.

3.2.4 Flowchart Dekripsi Algoritma *Twofish*

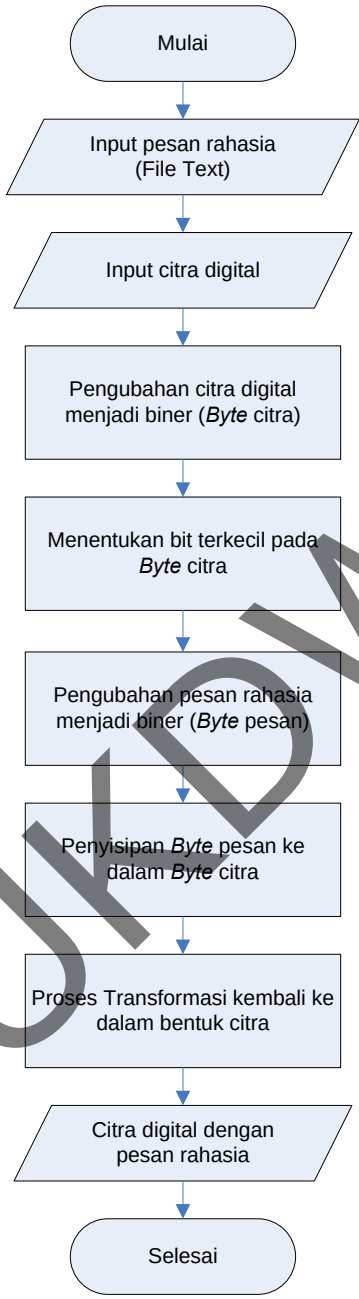


Gambar 3.3. Flowchart Dekripsi *Twofish*

Dari Gambar 3.3*Flowchart* dekripsi algoritma *twofish* dapat dijelaskan sebagai berikut :

1. Proses dekripsi dimulai dengan input *ciphertext* dengan ukuran blok 128 bit.
2. *Ciphertext* tersebut dibagi menjadi 4 blok 32 bit yaitu C_0, C_1, C_2dan C_3
3. Ke empat blok tersebut di XOR kan dengan 4 sub kunci dari kunci yang diperluas (*output whitening*).
4. Ke empat blok hasil dari *output whitening* saling dipertukaran terlebih dahulu, blok pertama ditukar dengan blok ke tiga dan blok ke dua ditukar dengan blok ke 4 (ROR ditukar dengan ROL).
5. Hasil dari pertukaran pada langkah ke 4akan melalui proses fungsi *f*sebanyak 16 kali putaran. Pada setiap putaran kedua blok pertama dijadikan sebgai input dari fungsi *f* (blok ke dua dirotasikan ke kiri sejauh 8 bit terlebih dahulu), dan blok ketiga dilakukan operasi XOR dengan hasil fungsi *f* yang pertama lalu dirotasikan ke kanan sejauh 1 bit, dan blok ke empat dirotasikan ke kiri sejauh 1 bit terlebih dahulu selanjutnya di XOR kan dengan hasil ke dua dari fungsi *f*. Blok pertama dan ke tiga saling dipertukarkan untuk digunakan pada putaran selanjutnya begitu juga blok ke dua dan ke empat.
6. Hasil dari 16 putaran fungsi f akan melalui proses *input whitening* dengan meng-XOR-kan ke empat blok tersebut dengan 4 buah sub kunci yang diperluas.
7. Hasil dari *input whitening* disatukan kembali menjadi 1 blok bagian 128 bit.
8. Dan menghasilkan *plaintext* 128 bit hasil dari proses dekripsi algoritma *twofish.*
9. Selesai.

3.2.5 Flowchart Penyisipan Algoritma *Least Significant Bit*



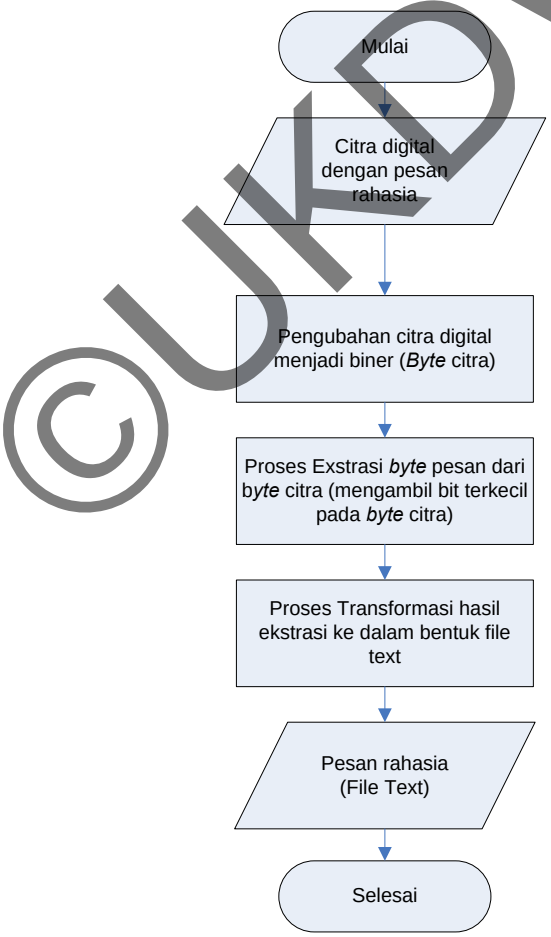
Gambar 3.4. Flowchart Penyisipan *Least Significant Bit*

Dari Gambar 3.4*Flowchart* penyisipan pesan rahasia pada algoritma *Least Significant Bit* dapat dijelaskan sebagai berikut :

1. Proses penyisipan dimulai dengan input sebuah pesan rahasia (fileteks)

2. Dilanjutkan dengan input sebuah citra digital yang menjadi wadah untuk proses steganografi.
3. Citra digital diubah menjadi biner (*byte* citra), dan di baca nilai setiap *pixel*dari citra tersebut.
4. Lalu menentukan bit terkecil pada citra digital tersbut
5. Pesan rahasia diubah menjadi biner (*byte* pesan).
6. *byte* pesan tersebut kemudian disisipkan ke dalam *byte*citra.
7. Setelah *byte* citra tersisipi *byte* pesan, dilakukan transformasi ulang ke dalam bentuk citra yang baru.
8. Dan diperoleh file citra digital yang telah disisipi pesan rahasia.
9. Selesai.

3.2.6 Flowchart Ekstrasi Pesan Algoritma *Least Significant Bit*



Gambar 3.5. Flowchart Ekstrasi *Least Significant Bit*

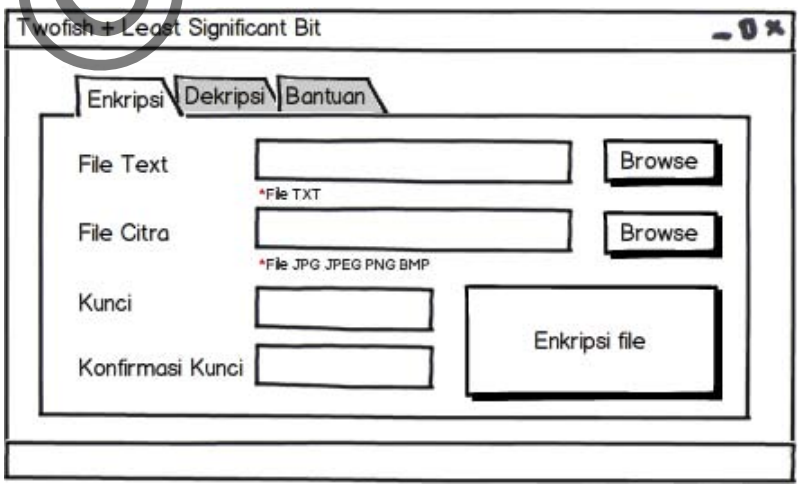
Dari Gambar 3.5*Flowchart* ekstrasi pesan rahasia dari citra digital pada algoritma *Least Significant Bit* dapat dijelaskan sebagai berikut :

1. Proses ekstrasi dimulai dengan input sebuah citra digital yang telah disisipi pesan rahasia
2. Citra tersebut diubah menjadi biner (*byte*citra), dan di baca nilai tiap *pixel* dari citra tersebut.
3. Lalu dilakukan proses ekstrasi pada *byte* citra tersebut dengan cara mengambil *byte* pesan yang disisipkan pada bit terakhir di setiap *pixel* citra.
4. Kemudai *byte* pesan yang diperoleh di transformasikan ulang menjadi sebuah file teks yang berisi pesan rahasia.
5. Dan diperoleh pesan rahasia (file teks) asli yang sebelumnya di sisipkan pada citra digital yang ditentukan.
6. Selesai.

3.3 Perancangan Antarmuka

Antarmuka sistem yang dibangun oleh penulis terdiri dari 1 *form* terdiri dari 3 tab yaitu tab Enkripsi, tab Dekripsi, dan tab Bantuan.

Tab Enkripsi adalah tab halaman yang digunakan untuk melakukan penyembunyian file text pada citra digital. Gambar 3.6 adalah rancangan dari tab enkripsi.

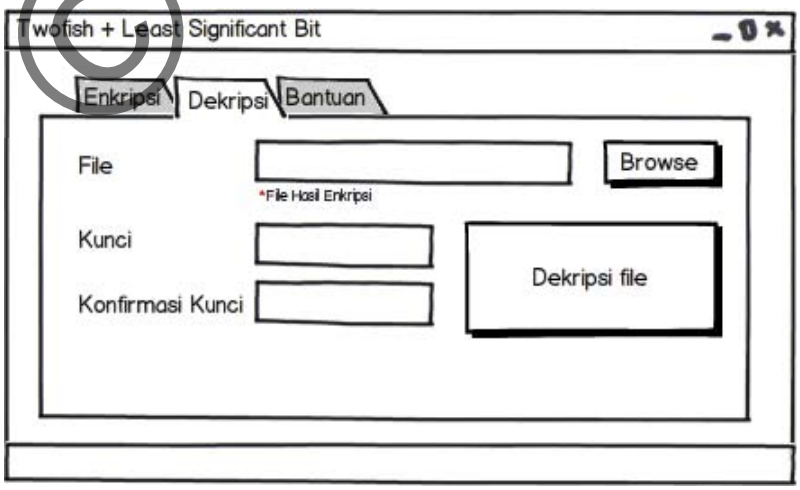


Gambar 3.6. Rancangan tab Enkripsi

Penjelasan dari tab enkripsi adalah sebagai berikut :

- *Field* File Text adalah sebuah inputan file teks yang akan disembunyikan pada citra digital. Format file teksyang dapat digunakan pada sistem adalah txt.
- *Field* File Citra adalah sebuah inputan citra yang akan digunakan sebagai media penampung pesan rahasia pada algoritma LSB. Pada sistem ini, media penampung yang digunakan haruslah file citra yang berformat PNG dan BMP.
- *Field* Kunci adalah sebuah inputan kunci yang digunakan untuk proses enkripsi pada algoritma *twofish* yang diterapkan pada sistem ini.
- *Field* Konfirmasi kunci sama dengan *Field*Kunci. *Field*ini digunakan untuk verifikasi kunci yang digunakan pada sistem ini.
- *Button* Enkripsi File adalah sebuah tombol yang digunakan untuk melakukan proses enkripsi dan penyisipan file teks pada citra digital yang dijalankan pada sistem ini.

Tab Dekripsi adalah tab halaman yang digunakan untuk melakukan ekstrasi dan dekripsi file teks dari citra digital yang telah disisipi pesan rahasia yang terenkripsi. Gambar 3.7 adalah rancangan dari tab dekripsi.

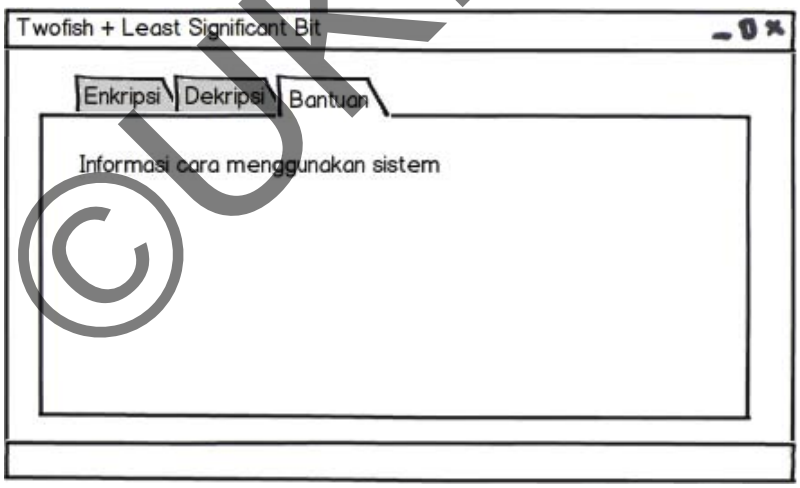


Gambar 3.7. Rancangan tab Dekripsi

Penjelasan dari tab dekripsi adalah sebagai berikut :

- *Field* File adalah sebuah inputan filecitra digital yang disisipi fileteks yang terenkripsi yang dilakukan melalui sistem ini.
- *Field* Kunci adalah sebuah inputan kunci yang digunakan untuk proses dekripsi pada algoritma *twofish* yang diterapkan pada sistem ini.
- *Field* Konfirmasi kunci sama dengan Kunci. *Field*ini digunakan untuk verifikasi kunci yang digunakan pada sistem ini.
- *Button* Dekripsi File adalah sebuah tombol yang digunakan untuk melakukan proses ekstraksidan dekripsi file teks dari citra digital yang dijalankan pada sistem ini. Jika proses tersebut berhasil, maka file pesan rahasia akan secara otomatis terunduh ke dalam folder dimana file yang menjadi inputan itu berada.

Tab Bantuan adalah tab halaman yang berisi informasi cara menggunakan sistem dan penjelasan instilah-istilah yang ada pada sistem ini. Gambar 3.8 adalah rancangan tampilan tab bantuan.



Gambar 3.8. Rancangan tab Bantuan

BAB 4
IMPLEMENTASI DAN ANALISIS SISTEM

4.1 Implementasi Sistem

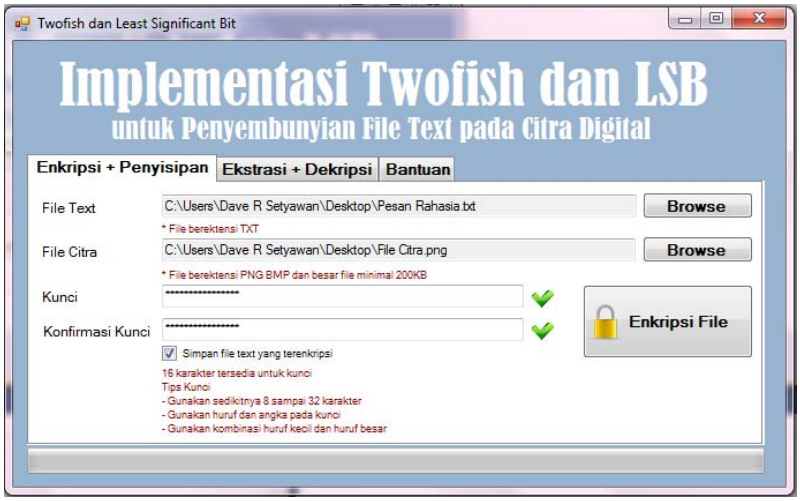
Pada bagian ini akan dibahas mengenai implementasi *Twofishdan Least Significant Bit* (LSB) pada sistem yang dibangun oleh penulis.

4.1.1 Implementasi Input

Input pada sistem yang dibangun oleh penulis terdiri dari dua jenis proses yaitu proses enkripsi pesan lalu penyisipan pesan (*Embedding*) pada file citra dan proses ekstrasi pesan (*Extracting*) dari file citra lalu dekripsi pesan. Dalam setiap proses tersebut mempunyai input masing-masing yang hampir sama.

4.1.2.1 Implementasi Input pada Proses Enkripsi dan Penyisipan

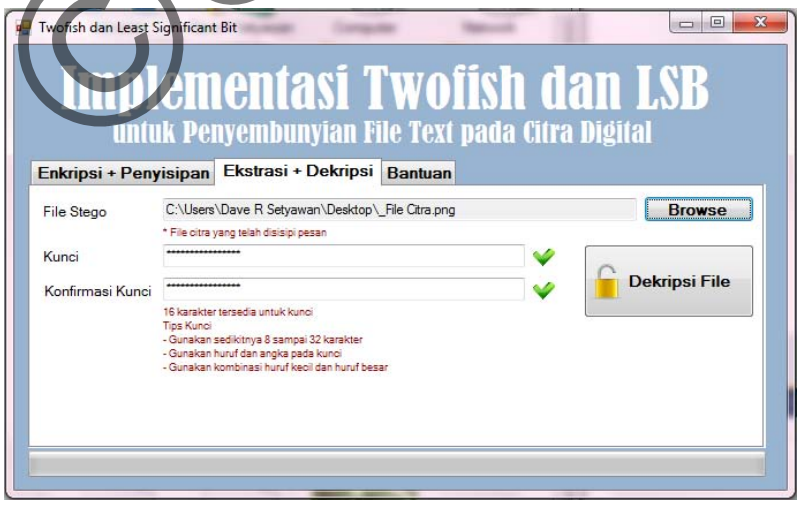
Input yang diperlukan dalam proses enkripsi dan penyisipan ada 3 input, diantaranya yaitu file teks, file citra, dan kunci. Untuk file teks, file harus mempunyai format txt sedangkan untuk file citra, file harus berformat PNG atau BMP. Untuk kunci, panjang kunci yang diperbolehkan adalah 8 sampai 32 karakter dan harus mempunyai kombinasi huruf kecil, huruf besar dan angka. Sistem mengimplementasikan fungsi *validasi*format citra untuk memastikan bahwa inputan file citra yang dipilih pengguna dengan melakukan pengecekan apakah file citra tersebut adalah file citra asli atau file lain yang diganti ekstensinya seperti file citra. Selain itu file citra juga dilakukan *validasi* terhadap ukuran file citra yang dipilih. Ukuran minimal file citra yang digunakan adalah 200 kb. Sistem juga menyediakan pilihan ketika pengguna ingin menyimpan file teks hasil enkripsi dalam bentuk file. Gambar 4.1 menunjukkan tampilan antarmuka dari sistem untuk proses enkripsi dan penyisipan :



Gambar 4.1. Tampilan tab Enkripsi dan Penyisipan

4.1.2.2 Implementasi Input pada Proses Ekstrasi dan Dekripsi

Input yang diperlukan dalam proses ekstrasi dan dekripsi ada 2 input, diantaranya yaitu file stego dan kunci. File stego haruslah memiliki format PNG atau BMP. Sama dengan proses enkripsi dan penyisipan, sistem juga akan melakukan validasi terhadap file stego yang diinputkan oleh pengguna, file tersebut haruslah file citra asli dan bukan file lain yang diganti ekstensi nya. Dan juga kunci pada tab ini sama dengan kunci yang berada pada tab enkripsi dan penyisipan. Gambar 4.2 menunjukkan tampilan antarmuka dari sistem untuk proses ekstrasi dan dekripsi :



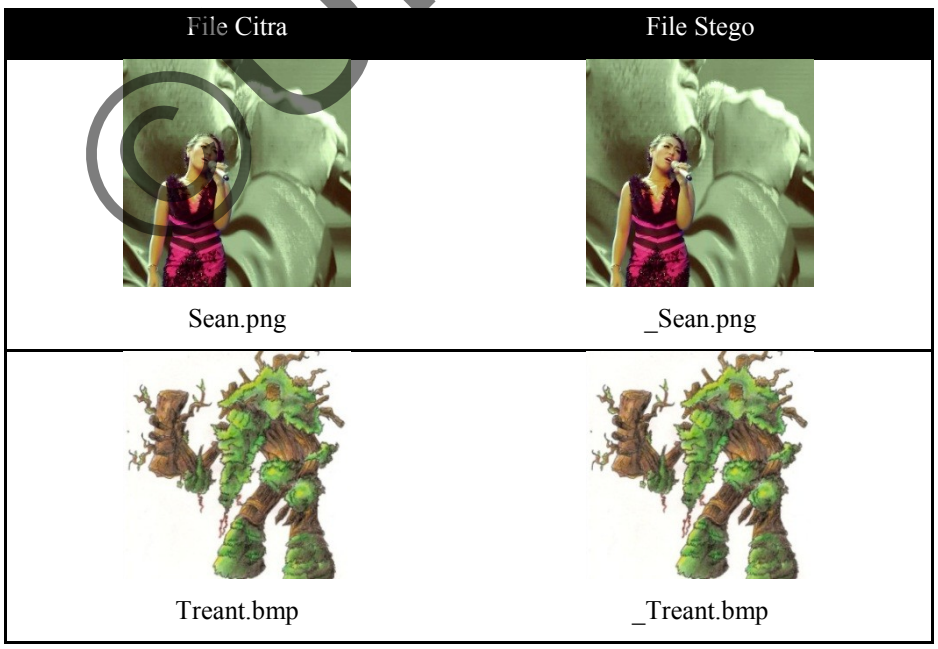
Gambar 4.2. Tampilan tab Ekstrasi dan Dekripsi

4.1.2 Implementasi output

Output pada sistem yang dibangun penulis terdiri dari dua jenis output, diantaranya yaitu output untuk proses enkripsi dan penyisipan pesan (*Embedding*) yang berupa file citra dan file teks hasil enkripsi dengan ekstensi .twf (*optional*) dan output untuk proses ekstraksi pesan (*Extracting*) dan dekripsi yang berupa file teks.

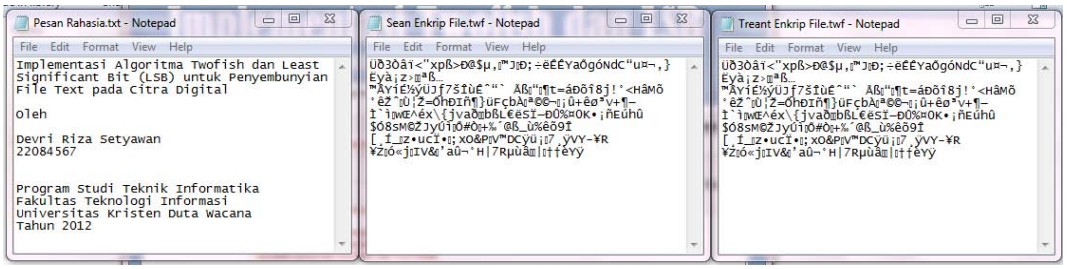
4.1.2.1 Implementasi Output pada Proses Enkripsi dan Penyisipan

Output yang dihasilkan dari proses enkripsi dan penyisipan adalah sebuah citra (file stego) yang telah disisipkan pesan rahasia (file teks) yang sudah terenkripsi terlebih dahulu dengan algoritma *twofish*. Output sistem berupa file stego akan secara otomatis mempunyai nama file yang hampir sama dengan input file citra yang digunakan. Sebagai contoh, file citra yang digunakan adalah "sean.png", maka file stegonya akan bernama "_sean.png". Dan ketika pengguna ingin menyimpan file teks hasil enkripsi, maka sistem akan mengeluarkan output file dengan nama "nama_file_citra (spasi) Enkrip File" dengan ekstensinya .twf (Contoh : Sean Enkrip File.twf). Gambar 4.3 menunjukkan contoh file citra asli dan file citra yang sudah disisipi pesan terenkripsi (file stego) :



Gambar 4.3. Contoh citra penampung sebelum dan setelah penyisipan pesan terenkripsi

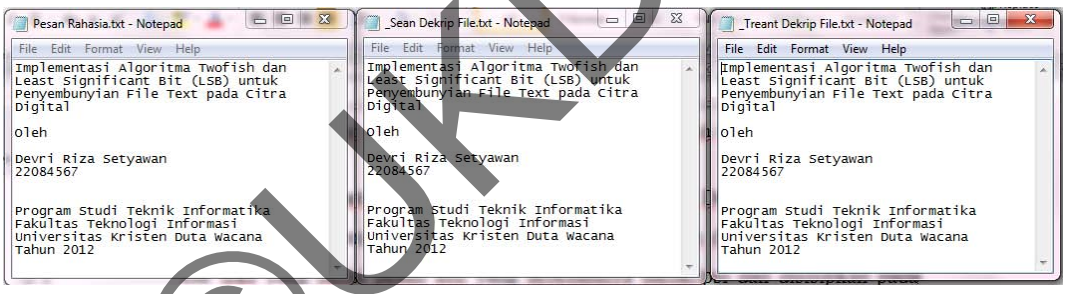
Gambar 4.4 menunjukkan contoh file teks asli dan file teks yang sudah terenkripsi oleh sistem :



Gambar 4.4. Contoh file teks asli sebelum dan setelah terenkripsi.

4.1.2.2 Implementasi Output pada Proses Ekstrasi dan Dekripsi

Output yang dihasilkan dari proses ekstrasi dan dekripsi adalah sebuah file teks yang berisi pesan asli yang sebelumnya dienkripsi dan disisipkan pada file stego. File output tersebut mempunyai nama file yaitu "nama_file_stego (spasi) Dekrip File". Gambar 4.5 menunjukkan contoh pesan rahasia sebelum di enkripsi dan disisipkan dan setelah diekstrasi dan didekripsi :



*Gambar 4.5.*Contoh file teks sebelum di enkripsi dan disisipkan dan setelah diekstrasi dan didekripsi

4.1.3 Implementasi Proses

4.1.3.1 Implementasi Penjadwalan Kunci Algoritma *Twofish*

Sebelum melakukan proses enkripsi ataupun dekripsi pada algoritma *twofish* sangat diperlukan adanya penjadwalan kunci. Sebelum melakukan penjadwalan kunci, kunci yang mana sebagai input pengguna dilakukan komputasi sehingga membentuk vektor *Me, Mo* dan *S*. Vektor *Me* dan *Mo* digunakan sebagai inputan pada penjadwalan kunci, sedangakan vektor *S* digunakan pada kotak S (*S-Boxes*) yang berada pada fungsi *g*. Vektor *Me* dan

*Mo*dilakukan perhitungan fungsi *h* untuk mendapatkan kunci yang diperluas sebanyak 40 buah. Potongan program dari penjadwalan kunci dari algoritma *twofish* dapat dilihat pada lampiran dari laporan ini.

Pada proses penjadwalan kunci algoritma *twofish* membaca inputan berupa kunci yang mempunyai panjang kunci sebanyak 16 karakter (128 bit), 24 karakter (192 bit) atau 32 karakter (256 bit). Jika pengguna menginputkan tidak sesuai dengan panjang kunci seperti yang tertera diatas maka akan dilakukan penambahan karakter 0 pada kunci yang dimasukkan (*padding)* terlebih dahulu sehingga memenuhi kriteria kunci yang ditentukan. Sebagai contoh : user memasukkan kunci sepanjang 10 karakter ("Rahasia123"), maka sistem akan melakukan *padding*terhadap input tersebut sehingga memenuhi persyaratan kunci yang diijinkan oleh sistem, kunci yang diterima sistem menjadi "Rahasia123000000" dan panjangnya menjadi 16 karakter (128 bit). Setelah itu kunci tersebut di lakukan perhitungan dengan bantuan fungsi *h* akan membentuk 40 sub kunci yang diperluas.

4.1.3.2 Implementasi Proses Enkripsi Algoritma *Twofish*

Proses enkripsi pada algoritma *twofish* yang di implementasikan dalam sistem yang dibangun melalui beberapa tahap. Pertama-tama *twofish* menerima masukan *plainteks* sebesar 128 bit lalu dipecah menjadi 4 bagian yang masing-masing sebesar 32 bit dengan menggunakan konversi *little endian*. 4 bagian tersebut di lakukan proses *input whitening* dengan meng-XOR-kan dengan kunci yang sudah diperluas, kunci yang digunakan pada *input whitening* adalah k_0, k_1, k_2 dan k_3. Hasil dari *inputwhitening* (R_0 dan R_1) akan dilakukan perhitungan melalui fungsi *F* dan akan mempunyai output F_0 dan F_1dan masing-masing di-XOR-kan dengan R_2 dan R_3. F_0di-XOR-kan dengan R_2 lalu hasilnya dirotasikan ke kanan sejauh 1 bit, sedangankan F_1 dirotasikan ke kiri sejauh 1 bit terlebih dahulu lalu di-XOR-kan dengan R_3. Proses perhitungan Fungsi *f* dilakukan sebanyak 16 kali iterasi. Setelah itu dilakukan *swap* blok akhir yaitu dengan meng-*undo swap* blok paling akhir. Terakhir dilakukan proses *output whitening* dengan meng-XOR-an dengan kunci yang sudah diperluas, kunci yang digunakan pada *output*

*whitening*adalah k_4, k_5, k_6 dan k_7. Hasil dari *output whitening* dikonversi dengan fungsi *konversi little endian* untuk mendapatkan *chipertext*. Potongan program dari proses enkripsi algoritma *twofish*dapat dilihat pada lampiran dari laporan ini.

4.1.3.3 Implementasi Proses Dekripsi Algoritma *Twofish*

Proses yang dilalui pada saat proses dekripsi yang di implementasikan pada sistem yang dibangun sama saja dengan pada saat proses enkripsi, tetapi hanya arahnya saja yang berlawan. Proses yang dilalui secara berurutan yaitu : konversi *little endian*, *output whitening*, *swap blok* terakhir, 16 iterasi fungsi *f* dekripsi, *input whitening* dan konversi*little endian* untuk mendapatkan *plainteks*. Inputnya berupa *chiperteks* dan kunci yang sama pada saat digunakan pada proses enkripsi. Potongan program dari proses dekripsi algoritma *twofish*dapat dilihat pada lampiran dari laporan ini.

4.1.3.4 Implementasi Proses Penyisipan Algoritma LSB

Proses penyisipan dengan algoritma LSB yang di implementasikan pada sistem yang dibangun melalui beberapa tahap. Pertama-tama pesan yang terenkripsi dihitung panjangnya terlebih dahulu. Kemudian sediakan variable sepanjang satu *byte* untuk memberikan identitas pada file citra yang akan disisipi pesan terenkripsi. Nilai dari identitas yang digunakan adalah 220. Setelah itu sediakan variable sepanjang 2 *byte* untuk menampung panjang pesan terenkripsi yang akandisisipkan. Lalu sisipkan satu per satu ke dalam citra yang digunakan sebagai wadah dimulai dari 1 *byte* identitas, 2 *byte* panjang pesan dan pesan terenkripsi yang disisipkan. Penyisipan pesan tersebut dilakukan dengan cara mengakses *pixel* citra yang digunakan satu persatu dan mengganti bit terakhir dari nilai RGB *pixel* dengan nilai bit yang di konversi dari *array of byte* pesan terenkripsi yang akan disisipkan. Potongan program dari proses penyisipan algoritma *least significant bit* (LSB) dapat dilihat pada lampiran dari laporan ini.

4.1.3.5 Implementasi Proses Ekstrasi Algoritma LSB

Proses ekstrasi dengan algoritma LSB yang di implementasikan pada sistem yang dibangun juga melalui beberapa tahap. Pertama dilakukan proses esktrasi 24 bit pertama yang berisi nilai identitas sebesar 8 bit (1 *byte*) dan panjang pesan sebesar 16 bit (2 *byte*) dari semua pesan yang disisipkan. Setelah itu identitas di cek terlebih dahulu apakah bernilai 220 atau tidak, jika tidak akan di munculkan pesan bahwa citra tidak tersisipi pesan namun jika bernilai 220 akan dilakukan proses selanjutnya. Proses tersebut adalah menghitung panjang pesan untuk untuk menentukan berapa kali proses perulangan yang perlu dilakukan untuk mengekstrasi pesan terenkripsi yang disisipkan pada file citra tersebut. Proses ekstrasi tersebut mengekstrasi semua *byte* yang tersisipkan yaitu identitas, panjang pesan dan pesan terenkripsi. Setelah proses ekstrasi selesai, sistem akan melakukan pemisahan antara identitas dan panjang pesan dengan pesan terenkripsi. Hanya pesan terenkripsi yang akan dikeluarkan dari proses ekstrasi pada sistem ini yang mana hasilnya akan dilakukan proses dekripsi. Potongan program dari proses ekstrasi algoritma *least significant bit* (LSB) dapat dilihat pada lampiran dari laporan ini.

4.2 Analisa Sistem

4.2.1 Tujuan Analisis

Dalam peneletian ini, terdapat 2 hal yang merupakan tujuan dari analisis dari sistem yang mengimplementasikan algoritma *Twofish* dan *Least Significant Bit* (LSB). Hal-hal tersebut diantaranya adalah :

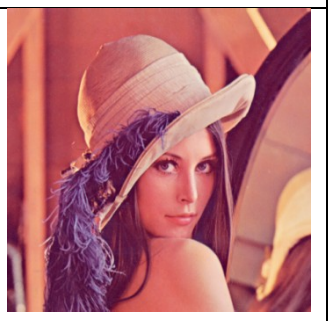
1. Menganalisa perubahan ukuran file teks dalam mengalami proses enkripsi berdasarkan panjang kunci yang digunakan dan bagaimana pengaruhnya terhadap kapasitas penyisipan.
2. Menganalisa perubahan yang terjadi dalam file citra yang mengalami proses penyisipan pesan yang terenkripsi.
3. Menganalisa ketahanan file stego, apakah dapat menghasilkan pesan rahasia yang utuh jika dilakukan beberapa operasi manipulasi seperti penambahan efek berupa penambahan ketajaman citra(*Sharpen*),

penambahan nilai kontras dan proses transformasi berupa rotasi 90^o, 180^o, dan 270^o.

4.2.2 Data Analisis

Subbab data analisis berisi data file citra dan file teks yang digunakan sebagai pengujian sistem. Tabel 1 menunjukkan citra yang akan digunakan sebagai penampung pesan rahasia yang terenkripsi dalam analisis sistem yang berupa citra PNG dan BMP.

Tabel 4.1.
Data File Citra penampung pesan terenkripsi.

No	Nama File	Ukuran File	Dimensi	Citra
1	Penguin.bmp	480,054 *bytes*	400 x 400 pixels 24 bit	
2	lena.png	428,593 *bytes*	400 x 400 pixels 24 bit	

Untuk data pesan rahasia atau file teks yang akan digunakan dalam pengujian sistem ini terdapat 3 buah pesan yang mempunyai panjang pesan yang berbeda. Tabel 2 menunjukkan data pesan rahasia / file teks yang akan dienkripsi dan disisipkan ke dalam citra dalam analisis sistem.

Tabel 4.2.

Data file teks yang akan dienkripsi dan disisipkan.

No	Nama File	Ukuran File	Kunci yang digunakan
1	Pesan Pendek.txt	2,279 *bytes*	• Panjang Kunci : 16 Karakter (128 bit) DaveRSetyawan123 • Panjang Kunci : 24 Karakter (192 bit) DevriRizaSetyawan1234567 • Panjang Kunci : 32 Karakter (256 bit) ImplementasiTwofishLSB2208456712
2	Pesan Sedang.txt	7,992 *bytes*	• Panjang Kunci : 16 Karakter (128 bit) DaveRSetyawan123 • Panjang Kunci : 24 Karakter (192 bit) DevriRizaSetyawan1234567 • Panjang Kunci : 32 Karakter (256 bit) ImplementasiTwofishLSB2208456712
3	Pesan Panjang.txt	21,293 *bytes*	• Panjang Kunci : 16 Karakter (128 bit) DaveRSetyawan123 • Panjang Kunci : 24 Karakter (192 bit) DevriRizaSetyawan1234567 • Panjang Kunci : 32 Karakter (256 bit) ImplementasiTwofishLSB2208456712

4.2.3 Kasus Analisis

4.2.3.1 Kasus Pengaruh Penggunaan Kunci pada Proses Enkripsi terhadap Perubahan Ukuran File yang disisipkan.

Pada kasus ini, file citra yang akan digunakan sebagai wadah untuk disisipi pesan terenkripsi adalah *lena.png* dan *Penguin.bmp*yang mempunyai dimensi 400 x 400 pixel dan mempunyai ukuran 428 KB dan 480 KB. Untuk file teks yang akan dienrkipsi dan disisipkan, pengujian dibagi menjadi 3 bagian pengujian berdasarkan panjang kunci yang digunakan yaitu 16 karakter (128 bit), 24 karakter (192 bit) dan 32 karakter (256 bit).

Tabel 4.3.

Hasil pengujian enkripsi terhadap file teks dengan menggunakan panjang kunci 128 bit.

No	File Citra	File Teks	Sebelum Enkripsi	Sesudah Enkripsi	Disisipkan& Diekstrasi
1	Penguin.bmp	Pesan Pendek.txt	2,279 bytes	2,888 bytes	V
2	Penguin.bmp	Pesan Sedang.txt	7,922 bytes	7,936 bytes	V
3	Penguin.bmp	Pesan Panjang.txt	21,293 bytes	21,296 bytes	V
4	lena.png	Pesan Pendek.txt	2,279 bytes	2,888 bytes	V
5	lena.png	Pesan Sedang.txt	7,922 bytes	7,936 bytes	V
6	lena.png	Pesan Panjang.txt	21,293 bytes	21,296 bytes	V

Tabel 4.4.

Hasil pengujian enkripsi terhadap file teks dengan menggunakan panjang kunci 192 bit.

No	**File Citra**	**File Teks**	**Sebelum Enkripsi**	**Sesudah Enkripsi**	**Disisipkan & Diekstrasi**
1	Penguin.bmp	Pesan Pendek.txt	2,279 bytes	2,888 bytes	V
2	Penguin.bmp	Pesan Sedang.txt	7,922 bytes	7,936 bytes	V
3	Penguin.bmp	Pesan Panjang.txt	21,293 bytes	21,296 bytes	V
4	lena.png	Pesan Pendek.txt	2,279 bytes	2,888 bytes	V
5	lena.png	Pesan Sedang.txt	7,922 bytes	7,936 bytes	V
6	lena.png	Pesan Panjang.txt	21,293 bytes	21,296 bytes	V

Tabel 4.5.

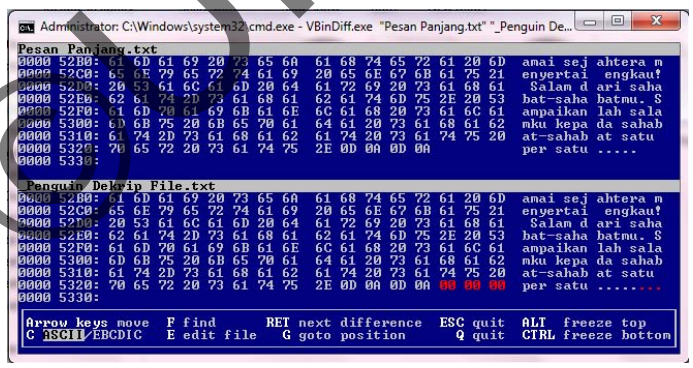
Hasil pengujian enkripsi terhadap file teks dengan menggunakan panjang kunci 256 bit.

No	**File Citra**	**File Teks**	**Sebelum Enkripsi**	**Sesudah Enkripsi**	**Disisipkan & Diekstrasi**
1	Penguin.bmp	Pesan Pendek.txt	2,279 bytes	2,888 bytes	V
2	Penguin.bmp	Pesan Sedang.txt	7,922 bytes	7,936 bytes	V
3	Penguin.bmp	Pesan Panjang.txt	21,293 bytes	21,296 bytes	V
4	lena.png	Pesan Pendek.txt	2,279 bytes	2,888 bytes	V
5	lena.png	Pesan Sedang.txt	7,922 bytes	7,936 bytes	V
6	lena.png	Pesan Panjang.txt	21,293 bytes	21,296 bytes	V

Catatan : V=Berhasil, X= Gagal

Dari hasil pengujian diatas, dapat disimpulkan bahwa proses enkripsi dengan algoritma *twofish* hampir tidak mengubah ukuran file teks. Dari data diatas perubahan yang terjadi hanya sangat kecil sekali sehingga tidak terlalu mempengaruhi kapasitas yang akan dibutuhkan pada file citra yang menjadi wadah steganografi. Hal ini disebabkan karena algoritma *twofish* berjalan pada 128 blok bit, input *plainteks* sebesar 128 bit dan output *chiperteks*-nya sebesar 128 bit juga. Penggunaan panjang kunci tidak mempengaruhi besar file output yang dihasilkan dari proses enkripsi dan dekripsi algoritma ini. Kunci yang digunakan hanya berpengaruh dalam proses penjadwalan kunci, kotak S (*S-boxes*) dan PHT (*Pseudo-Hadamard Transforms*) pada algoritma *twofish*.

Perubahan kecil yang terjadi pada file teks yang menjadi input proses enkripsi disebabkan oleh penambahan karakter atau yang sering disebut dengan istilah *padding* yang dilakukan supaya memenuhi persyaratan dari algoritma *twofish*. Sebagai contoh, terdapat inputan sebesar 80 bit sedangkan algoritma *twofish*berjalan pada 128 blok bit, maka inputan tersebut akan dilakukan proses penambahan karakter atau nilai "0" setelah akhir dari file inputan yang digunakan sampai memenuhi panjangnya menjadi 128 bit. Gambar 4.6 akan menunjukkan perubahan yang terjadi pada file inputan algoritma *twofish* :



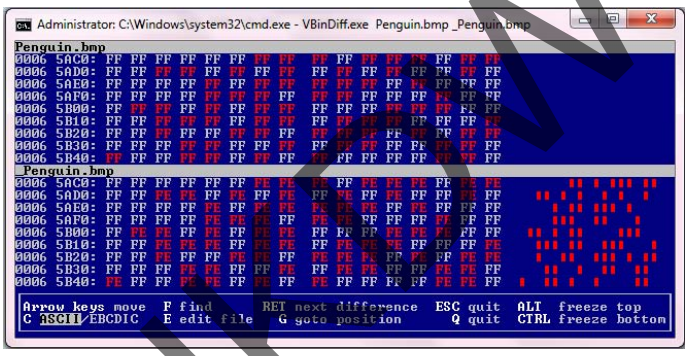
*Gambar 4.6.*Salah satu contoh perubahan yang terjadi pada file teks yang digunakan pada proses enkripsi dan dekripsi algoritma *twofish*.

Dari hasil perbandingan file sebelum enkripsi dan file setelah mengalami proses enkripsi dan dekripsi seperti pada gambar 4.6, dapat disimpulkan bahwa algoritma *twofish*berjalan pada 128 blok bit dan menambahkan karakter atau nilai

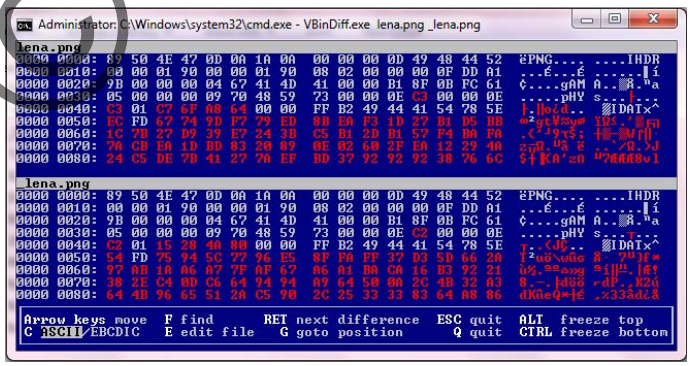
0 pada akhir inputan jika tidak memenuhi syarat 128 bit. Dan perubahan yang terjadi hanya terjadi karena *padding* yang dilakukan.

4.2.3.2 Kasus Perubahan yang Terjadi pada File Citra yang disisipi

Untuk kasus ini, file citra, file teks dan kunci yang digunakan sama seperti pada kasus sebelumnya. Perubahan file citra yang terjadi sebelum dan sesudah disisipi file teks yang terenkripsi bisa dilihat dengan bantuan perangkat lunak bernama VBinDiff-3.0_beta4buatan Christopher J. Madsen, yang mana perangkat lunak ini membantu penelitian untuk membandingkan 2 file dalam bentuk *heksadesimal*. Gambar 4.7 dan gambar 4.8 menunjukkan perbandingan file BMP dan PNG yang belum disisipi pesan (file teks terenkripsi) dan sudah disisipi:



Gambar 4.7. Perbandingan File Citra BMP yang belum disisipi dan sudah disisipi pesan terkenkripsi (Pesan Sedang.txt dengan kunci 256 bit)



Gambar 4.8. Perbandingan File Citra PNG yang belum disisipi dan sudah disisipi pesan terkenkripsi (Pesan Sedang.txt dengan kunci 256 bit)

Pada kasus ini, file citra sebelum disisipi dan sesudah disisipi dibandingkan dengan perangkat lunak yang bernama VBinDiff dan menghasilkan sebuah kesimpulan bahwa citra yang berektensi BMP mengalami penyisipan dibagian akhir dari file citra yang dijadikan wadah sedangkan file citra yang berektensi PNG mengalami penyisipan dari awal file citra yang digunakan. Hal itu bisa dilihat pada gambar 4.12 dan gambar 4.13 yang mana pada citra BMP penyisipan yang terjadi dimulai dari 0060 5AC0, sedangkan pada citra PNG penyisipan yang terjadi dimulai dari 0000 0030.

Tabel 4.6.
Hasil pengujian penyisipan pesan terenkripsi pada file citra.

No	File Citra	File Teks	Kunci	Sebelum Disisipi	Sesudah Disisipi
1	Penguin.bmp	Pesan Pendek.txt	128 bit	480,054 bytes	480,054 bytes
2	Penguin.bmp	Pesan Pendek.txt	192 bit	480,054 bytes	480,054 bytes
3	Penguin.bmp	Pesan Pendek.txt	256 bit	480,054 bytes	480,054 bytes
4	Penguin.bmp	Pesan Sedang.txt	128 bit	480,054 bytes	480,054 bytes
5	Penguin.bmp	Pesan Sedang.txt	192 bit	480,054 bytes	480,054 bytes
6	Penguin.bmp	Pesan Sedang.txt	256 bit	480,054 bytes	480,054 bytes
7	Penguin.bmp	Pesan Panjang.txt	128 bit	480,054 bytes	480,054 bytes
8	Penguin.bmp	Pesan Panjang.txt	192 bit	480,054 bytes	480,054 bytes
9	Penguin.bmp	Pesan Panjang.txt	256 bit	480,054 bytes	480,054 bytes
10	lena.png	Pesan Pendek.txt	128 bit	428,593 bytes	430,702 bytes
11	lena.png	Pesan Pendek.txt	192 bit	428,593 bytes	430,674 bytes
12	lena.png	Pesan Pendek.txt	256 bit	428,593 bytes	430,695 bytes
13	lena.png	Pesan Sedang.txt	128 bit	428,593 bytes	430,923 bytes
14	lena.png	Pesan Sedang.txt	192 bit	428,593 bytes	430,967 bytes
15	lena.png	Pesan Sedang.txt	256 bit	428,593 bytes	430,967 bytes
16	lena.png	Pesan Panjang.txt	128 bit	428,593 bytes	431,319 bytes
17	lena.png	Pesan Panjang.txt	192 bit	428,593 bytes	431,180 bytes
18	lena.png	Pesan Panjang.txt	256 bit	428,593 bytes	431,246 bytes

Dari tabel4.6 hasil pengujian penyisipan pada kedua file yang berbeda, dapat dilihat bahwa pada file citra BMP tidak mengalami perubahan ukuran file sesudah disisipi pesan terenkripsi sedangkan pada file citra PNG terjadi perubahan

ukuran file sekitar ± 2 bytes. Hal tersebut bisa terjadi dikarenakan file citra PNG bersifat *lossless compression*yang mana berarti tidak ada satupun informasi citra yang dihilangkan ketika citra tersebut mengalami manipulasi isi. Selain itu, dapat dilihat bahwa file citra BMP mempunyai daya tampung lebih besar daripada file citra PNG. Hal itu dapat dilihat pada file citra BMP dan PNG yang sama-sama mempunyai resolusi 400 x 400 *pixel,* file citra BMP mempunyai ukuran file lebih besar daripada file citra PNG.

4.2.3.3 Kasus Ketahanan Output Sistem (File Stego) terhadap Beberapa Operasi Manipulasi

Pada kasus ini, pengujian terhadap ketahanan file stego (file citra yang disisipi file teks yang terenkripsi) dengan melakukan penambahan efek ketajaman (*Sharpen*), penambahan kontras, dan proses transformasi berupa rotasi 90^o, 180^o, dan 270^o.

Tabel 4.7.
Hasil pengujian ketahanan citra terhadap beberapa operasi manipulasi.

No	File Citra	Kunci	Operasi Manipulasi	Berhasil Diekstrak	Berhasil Didekripsi
1	Penguin.bmp	192 bit	Penambahan efek *Sharpen*	V	X
2	lena.png	192 bit	Penambahan efek *Sharpen*	X	X
3	Penguin.bmp	192 bit	Penambahan kontras	X	X
4	lena.png	192 bit	Penambahan kontras	X	X
5	Penguin.bmp	192 bit	Rotasi 90^o	X	X
6	lena.png	192 bit	Rotasi 90^o	X	X
7	Penguin.bmp	192 bit	Rotasi 180^o	X	X
8	lena.png	192 bit	Rotasi 180^o	X	X
9	Penguin.bmp	192 bit	Rotasi 270^o	X	X
10	lena.png	192 bit	Rotasi 270^o	X	X

Catatan :
V= Berhasil, X= Gagal

Dari hasil pengujian dengan sistem yang telah dibangun, dapat dilihat bahwa file stego (citra yang sudah disisipi pesan terenkripsi) tidak tahan terhadap beberapa operasi manipulasi. File stego yang telah dilakukan operasi manipulasi tidak dapat menghasilkan output ekstrasi yang sama dengan input yang disisipkan bahkan identitas yang disematkan pada proses penyisipan pun nilianya berubah sehingga tidak dapat mengekstrasi pesan yang disisipkan pada file stego. Ketidak tahanan file stego terhadap beberapa operasi manipulasi adalah karena adanya perubahan nilai bit pada tiap *pixel* citra karena operasi manipulasi yang dilakukan pada citra tersebut, sehingga data yang telah disisipkan pun telah berubah termasuk identitas yang disisipkan untuk menandai file stego.

LAMPIRAN

LAMPIRAN A
LISTING PROGRAM

A. Form Utama

```
Imports System.IO
Imports System.Text
Imports System.Drawing
Imports System.Drawing.Imaging

Imports Twofish___LSB.FungsiPendukung
Imports Twofish___LSB.Conveter
Imports Twofish___LSB.lsb
Imports Twofish___LSB.twofish

PublicClassTA

PrivateSub TA_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) HandlesMyBase.Load
        Handling_awal()
EndSub

#Region"Handling"

#Region"Kondisi Awal"
Sub Handling_awal()
        btnEnkripsi.Enabled = False
        btnDekripsi.Enabled = False
        pbEnkripsi1.Image = Nothing
        pbEnkripsi2.Image = Nothing
        pbDekripsi1.Image = Nothing
        pbDekripsi2.Image = Nothing
        txtFileCitra.ReadOnly = True
        txtFileText.ReadOnly = True
        txtFileCitraYangDisisipi.ReadOnly = True
EndSub
#EndRegion

#Region"Enkripsi + Penyisipan"
PrivateSub btnOpenFileText_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnOpenFileText.Click
Dim OpenFileText AsNewOpenFileDialog
        OpenFileText.Title = "Pilih File Text"
        OpenFileText.Filter = "Text File (*.txt)|*.txt"
        OpenFileText.InitialDirectory =
System.Environment.GetFolderPath(Environment.SpecialFolder.MyComputer)

If OpenFileText.ShowDialog() = Windows.Forms.DialogResult.OK Then
            txtFileText.Text = OpenFileText.FileName
EndIf
        OpenFileText.Dispose()
EndSub

PrivateSub btnOpenFileCitra_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnOpenFileCitra.Click
Dim OpenFileCitra AsNewOpenFileDialog
```

```
Dim fileCitra AsString = ""
Dim flag AsBoolean = True

        OpenFileCitra.Title = "Pilih File Citra"
        OpenFileCitra.Filter = "File Citra
(*.PNG;*.BMP)|*.PNG;*.BMP"'(*.JPG;*.JPEG;*.PNG;*.BMP)|*.JPG;*.JPEG;*.PNG;
*.BMP"
        OpenFileCitra.InitialDirectory =
System.Environment.GetFolderPath(Environment.SpecialFolder.MyComputer)

If OpenFileCitra.ShowDialog() = Windows.Forms.DialogResult.OK Then
            fileCitra = OpenFileCitra.FileName
Else
            flag = False
EndIf
        OpenFileCitra.Dispose()

If flag Then
If IsValidImage(fileCitra) Then
'cek ukuran file
Dim ukuran AsDouble = New IO.FileInfo(fileCitra).Length / 1024
If ukuran >= 200 Then
                    txtFileCitra.Text = fileCitra
Else
MessageBox.Show("Ukuran file citra minimal 200KB!")
EndIf
Else
                txtFileCitra.Text = ""
MessageBox.Show("Error!!! File yang dipilih bukan File Citra!")
EndIf
EndIf
EndSub
#EndRegion

#Region"Dekripsi + Ekstrasi"
PrivateSub btnOpenFileCitraYangDisisipi_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnOpenFileCitraYangDisisipi.Click
Dim OpenFileCitraHasilEnkripsi AsNewOpenFileDialog
Dim fileCitraEnkripsi AsString = ""
Dim flag AsBoolean = True

        OpenFileCitraHasilEnkripsi.Title = "Open File Stego"
        OpenFileCitraHasilEnkripsi.Filter = "File Stego
(*.PNG;*.BMP)|*.PNG;*.BMP"
        OpenFileCitraHasilEnkripsi.InitialDirectory =
System.Environment.GetFolderPath(Environment.SpecialFolder.MyComputer)

If OpenFileCitraHasilEnkripsi.ShowDialog() =
Windows.Forms.DialogResult.OK Then
            fileCitraEnkripsi = OpenFileCitraHasilEnkripsi.FileName
Else
            flag = False
EndIf
        OpenFileCitraHasilEnkripsi.Dispose()

If flag Then
If IsValidImage(fileCitraEnkripsi) Then
                txtFileCitraYangDisisipi.Text = fileCitraEnkripsi
Else
                txtFileCitraYangDisisipi.Text = ""
MessageBox.Show("Error!!! File inputan bukan berupa File Citra!")
EndIf
```

```
EndIf
EndSub
#EndRegion

#Region"Error Handling Enkripsi"
PrivateSub txtKunciEnrkipsi_TextChanged(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtKunciEnrkipsi.TextChanged
If regexpass(txtKunciEnrkipsi.Text) Then
            pbEnkripsi1.Image = My.Resources.Resources.yes
Else
            pbEnkripsi1.Image = My.Resources.Resources.no
EndIf
If regexpass(txtKunciEnkripsi2.Text) _
And matchpass(txtKunciEnrkipsi.Text, txtKunciEnkripsi2.Text) Then
            pbEnkripsi2.Image = My.Resources.Resources.yes
Else
            pbEnkripsi2.Image = My.Resources.Resources.no
EndIf
        enableEnkrip()
        lblKunciEnkripsi.Text = 32 - txtKunciEnrkipsi.TextLength &"
karakter tersedia untuk kunci"
EndSub

PrivateSub txtKunciEnkripsi2_TextChanged(ByVal sender AsObject, ByVal e
As System.EventArgs) Handles txtKunciEnkripsi2.TextChanged
If regexpass(txtKunciEnkripsi2.Text) _
And matchpass(txtKunciEnrkipsi.Text, txtKunciEnkripsi2.Text) Then
            pbEnkripsi2.Image = My.Resources.Resources.yes
Else
            pbEnkripsi2.Image = My.Resources.Resources.no
EndIf
        enableEnkrip()
EndSub

PrivateSub txtKunciEnrkipsi_GotFocus(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtKunciEnrkipsi.GotFocus
        ToolTip.SetToolTip(txtKunciEnrkipsi, "Gunakan Kunci 8 sampai 32
karater"& Chr(13) &"dan terdiri dari huruf kecil,"& Chr(13) &"huruf besar
dan angka.")
        pbEnkripsi1.Image = My.Resources.Resources.no
If regexpass(txtKunciEnrkipsi.Text) Then
            pbEnkripsi1.Image = My.Resources.Resources.yes
Else
            pbEnkripsi1.Image = My.Resources.Resources.no
EndIf
        enableEnkrip()
EndSub

PrivateSub txtKunciEnkripsi2_GotFocus(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtKunciEnkripsi2.GotFocus
        pbEnkripsi2.Image = My.Resources.Resources.no
If regexpass(txtKunciEnkripsi2.Text) _
And matchpass(txtKunciEnrkipsi.Text, txtKunciEnkripsi2.Text) Then
            pbEnkripsi2.Image = My.Resources.Resources.yes
Else
            pbEnkripsi2.Image = My.Resources.Resources.no
EndIf
        enableEnkrip()
EndSub

PrivateSub txtFileText_TextChanged(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtFileText.TextChanged
        enableEnkrip()
```

```
EndSub

PrivateSub txtFileCitra_TextChanged(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtFileCitra.TextChanged
        enableEnkrip()
EndSub

Sub enableEnkrip()
If txtFileCitra.Text <>"" _
And txtFileText.Text <>"" _
And regexpass(txtKunciEnrkipsi.Text) _
And matchpass(txtKunciEnrkipsi.Text, txtKunciEnkripsi2.Text) Then
            btnEnkripsi.Enabled = True
Else
            btnEnkripsi.Enabled = False
EndIf
EndSub
#EndRegion

#Region"Error Handling Dekripsi"
PrivateSub txtKunciDekripsi_TextChanged(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtKunciDekripsi.TextChanged
If regexpass(txtKunciDekripsi.Text) Then
            pbDekripsi1.Image = My.Resources.Resources.yes
Else
            pbDekripsi1.Image = My.Resources.Resources.no
EndIf
If regexpass(txtKunciDekripsi2.Text) _
And matchpass(txtKunciDekripsi.Text, txtKunciDekripsi2.Text) Then
            pbDekripsi2.Image = My.Resources.Resources.yes
Else
            pbDekripsi2.Image = My.Resources.Resources.no
EndIf
        enableDekrip()
        lblKunciDekripsi.Text = 32 - txtKunciDekripsi.TextLength &"
karakter tersedia untuk kunci"
EndSub

PrivateSub txtKunciDekripsi_GotFocus(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtKunciDekripsi.GotFocus
        ToolTip.SetToolTip(txtKunciDekripsi, "Gunakan Kunci 8 sampai 32
karater"& Chr(13) &"dan terdiri dari huruf kecil,"& Chr(13) &"huruf besar
dan angka.")
        pbDekripsi1.Image = My.Resources.Resources.no
If regexpass(txtKunciDekripsi.Text) Then
            pbDekripsi1.Image = My.Resources.Resources.yes
Else
            pbDekripsi1.Image = My.Resources.Resources.no
EndIf
        enableDekrip()
EndSub

PrivateSub txtKunciDekripsi2_TextChanged(ByVal sender AsObject, ByVal e
As System.EventArgs) Handles txtKunciDekripsi2.TextChanged
If regexpass(txtKunciDekripsi2.Text) _
And matchpass(txtKunciDekripsi.Text, txtKunciDekripsi2.Text) Then
            pbDekripsi2.Image = My.Resources.Resources.yes
Else
            pbDekripsi2.Image = My.Resources.Resources.no
EndIf
        enableDekrip()
EndSub
```

```
PrivateSub txtKunciDekripsi2_GotFocus(ByVal sender AsObject, ByVal e As
System.EventArgs) Handles txtKunciDekripsi2.GotFocus
        pbDekripsi2.Image = My.Resources.Resources.no
If regexpass(txtKunciDekripsi2.Text) _
And matchpass(txtKunciDekripsi.Text, txtKunciDekripsi2.Text) Then
            pbDekripsi2.Image = My.Resources.Resources.yes
Else
            pbDekripsi2.Image = My.Resources.Resources.no
EndIf
        enableDekrip()
EndSub

PrivateSub txtFileCitraYangDisisipi_TextChanged(ByVal sender AsObject,
ByVal e As System.EventArgs) Handles txtFileCitraYangDisisipi.TextChanged
        enableDekrip()
EndSub

Sub enableDekrip()
If txtFileCitraYangDisisipi.Text <>"" _
And regexpass(txtKunciDekripsi.Text) _
And matchpass(txtKunciDekripsi.Text, txtKunciDekripsi2.Text) Then
            btnDekripsi.Enabled = True
Else
            btnDekripsi.Enabled = False
EndIf
EndSub
#EndRegion

#EndRegion

#Region"Core Program"
PrivateSub btnEnkripsi_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEnkripsi.Click

        progbarExe.Value = 0

'================================================================================
==========================================================='
'Implementasi Enkripsi Algoritma Twofish

Dim filetext AsByte() = File.ReadAllBytes(txtFileText.Text)
Dim filetext_128bit AsByte(,) = split_byte(filetext)
Dim filetext_hasil_enkrip(filetext_128bit.Length - 1) AsByte
Dim counter_filetext_hasil_enkrip AsInteger = -1

        progbarExe.Maximum = ((filetext_128bit.Length / 16) * 16) +
((filetext_128bit.Length / 16) * 16)
        progbarExe.Step = 1


Try
For i AsInteger = 0 To filetext_128bit.Length / 16 - 1 Step 1
Dim sebelum_enkrip(15) AsByte
Dim setelah_enkrip(15) AsByte

For j AsInteger = 0 To 15 Step 1
                    sebelum_enkrip(j) = filetext_128bit(i, j)
                    progbarExe.PerformStep()
Next

                setelah_enkrip = enkrip_twofish(sebelum_enkrip,
txtKunciEnrkipsi.Text)
```

```
For x AsInteger = 0 To 15 Step 1
                    counter_filetext_hasil_enkrip += 1
                    filetext_hasil_enkrip(counter_filetext_hasil_enkrip)
= setelah_enkrip(x)
                    progbarExe.PerformStep()
Next

Next
Catch ex AsException
MessageBox.Show("Enkripsi Gagal!!!")
EndTry

'================================================================================
============================================================'
'Ketika user ingin menyimpan file text hasil enkripsi dalam bentuk teks!

Try
If cbFile.Checked = TrueThen
Dim lokasi AsString = Path.GetDirectoryName(txtFileCitra.Text)
Dim nama_file AsString =
Path.GetFileNameWithoutExtension(txtFileCitra.Text)

Dim oFileStream As System.IO.FileStream
                oFileStream = New System.IO.FileStream(lokasi &"\"&
nama_file &" Enkrip File.twf", System.IO.FileMode.Create)
                oFileStream.Write(filetext_hasil_enkrip, 0,
filetext_hasil_enkrip.Length)
                oFileStream.Close()

EndIf
Catch ex AsException
MessageBox.Show("Simpan File Gagal!")
EndTry

'================================================================================
============================================================'
'Implementasi Penyisipan Pesan Algoritma LSB

Try
If lsb_insert(txtFileCitra.Text, filetext_hasil_enkrip) Then
                progbarExe.PerformStep()
                progbarExe.Value = progbarExe.Maximum

MessageBox.Show("Selamat!"& Chr(13) _
&"Enkripsi dan Penyembunyian Pesan dengan TWOFISH dan LSB BERHASIL!")
                txtFileCitra.Text = ""
                txtFileText.Text = ""
                txtKunciEnrkipsi.Text = ""
                txtKunciEnkripsi2.Text = ""
                cbFile.Checked = False
EndIf
Catch ex AsException
MessageBox.Show("Sisip File Gagal!")
EndTry

'================================================================================
============================================================'

EndSub

PrivateSub btnDekripsi_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDekripsi.Click
        progbarExe.Value = 0
```

```
'=============================================================================
============================================================'

'Implementasi Ekstrasi Pesan Algoritma LSB

Dim pesan_hasil_ekstrasi AsByte() =
lsb_ekstrasi(txtFileCitraYangDisisipi.Text)
If pesan_hasil_ekstrasi.Length = 0 Then
MessageBox.Show("Ekstrasi dan Dekripsi Pesan dengan TWOFISH dan LSB
GAGAL!")
Else
Try
Dim filetext_128bit AsByte(,) = split_byte(pesan_hasil_ekstrasi)
Dim lokasi AsString =
Path.GetDirectoryName(txtFileCitraYangDisisipi.Text)
Dim nama_file AsString =
Path.GetFileNameWithoutExtension(txtFileCitraYangDisisipi.Text)


                progbarExe.Maximum = (filetext_128bit.Length / 16 * 16) +
(filetext_128bit.Length / 16 * 16)
                progbarExe.Step = 1
'=============================================================================
============================================================'
'Implementasi Dekripsi Algoritma Twofish

Dim filetext_hasil_dekrip(filetext_128bit.Length - 1) AsByte
Dim counter_filetext_hasil_dekrip AsInteger = -1

For i AsInteger = 0 To filetext_128bit.Length / 16 - 1 Step 1
Dim sebelum_dekrip(15) AsByte
Dim setelah_dekrip(15) AsByte

For j AsInteger = 0 To 15 Step 1
                        sebelum_dekrip(j) = filetext_128bit(i, j)
                        progbarExe.PerformStep()
Next

                    setelah_dekrip = dekrip_twofish(sebelum_dekrip,
txtKunciDekripsi.Text)

For x AsInteger = 0 To 15 Step 1
                        counter_filetext_hasil_dekrip += 1

filetext_hasil_dekrip(counter_filetext_hasil_dekrip) = setelah_dekrip(x)
                        progbarExe.PerformStep()
Next

Next

'=============================================================================
============================================================'
'Menyimpan Hasil Dekripsi dalam bentuk File Txt

Dim oFileStream As System.IO.FileStream
                oFileStream = New System.IO.FileStream(lokasi &"\"&
nama_file &" Dekrip File.txt", System.IO.FileMode.Create)
                oFileStream.Write(filetext_hasil_dekrip, 0,
filetext_hasil_dekrip.Length)
                oFileStream.Close()
                progbarExe.PerformStep()
                progbarExe.Value = progbarExe.Maximum
```

```
MessageBox.Show("Selamat!"& Chr(13) &"Ekstrasi dan Dekripsi Pesan dengan
TWOFISH dan LSB BERHASIL!")
                txtFileCitraYangDisisipi.Text = ""
                txtKunciDekripsi.Text = ""
                txtKunciDekripsi2.Text = ""

'================================================================
============================================='
Catch ex AsException
MessageBox.Show("Ekstrasi dan Dekripsi Pesan dengan TWOFISH dan LSB
GAGAL!")
MessageBox.Show(ex.ToString)
EndTry

EndIf

'================================================================
============================================='
EndSub
#EndRegion

EndClass
```

B. Module *Twofish*

```
Moduletwofish

PrivateFunction ROR(ByVal input AsString, ByVal kali AsInteger)
AsString
For i AsInteger = 0 To kali - 1 Step 1
            input = input.Substring(input.Length - 1, 1) &
input.Substring(0, input.Length - 1)
Next
Return input
EndFunction

PrivateFunction ROL(ByVal input AsString, ByVal kali AsInteger)
AsString
For i AsInteger = 0 To kali - 1 Step 1
            input = input.Substring(1, input.Length - 1) &
input.Substring(0, 1)
Next
Return input
EndFunction

PrivateFunction xorstring(ByRef a AsString, ByVal b AsString)
AsString
Dim c AsString = ""
For i AsInteger = 0 To a.Length - 1 Step 1
            c &= CInt(a.Substring(i, 1)) XorCInt(b.Substring(i,
1))
Next
Return c
EndFunction

PrivateFunction xor_8bit(ByVal in1 AsLong, ByVal in2 AsLong)
AsInteger
```

```
Dim result AsLong = (in1 Xor in2) Mod (2 ^ 8)
ReturnCLng(result)
EndFunction

PrivateFunction xor_4bit(ByVal in1 AsInteger, ByVal in2 AsInteger)
AsInteger
Dim result AsInteger = (in1 Xor in2) Mod (2 ^ 4)
ReturnCInt(result)
EndFunction

Function permutasi_q0(ByVal input AsLong) AsInteger

'For the permutation q0 the 4-bit S-boxes are given by
't0 = [ 8 1 7 D  6 F  3 2 0 B  5 9 E  C  A  4 ] 'hex
't0 = [ 8 1 7 13 6 15 3 2 0 11 5 9 14 12 10 4 ] 'dec
Dim t0 AsInteger() = {8, 1, 7, 13, 6, 15, 3, 2, 0, 11, 5, 9, 14,
12, 10, 4}

't1 = [ E  C  B  8 1 2 3 5 F  4 A  6 7 0 9 D  ]
't1 = [ 14 12 11 8 1 2 3 5 15 4 10 6 7 0 9 13 ]
Dim t1 AsInteger() = {14, 12, 11, 8, 1, 2, 3, 5, 15, 4, 10, 6, 7,
0, 9, 13}

't2 = [ B  A  5 E  6 D  9 0 C  8 F  3 2 4 7 1 ]
't2 = [ 11 10 5 14 6 13 9 0 12 8 15 3 2 4 7 1 ]
Dim t2 AsInteger() = {11, 10, 5, 14, 6, 13, 9, 0, 12, 8, 15, 3, 2,
4, 7, 1}

't3 = [ D  7 F  4 1 2 6 E  9 B  3 0 8 5 C  A  ]
't3 = [ 13 7 15 4 1 2 6 14 9 11 3 0 8 5 12 10 ]
Dim t3 AsInteger() = {13, 7, 15, 4, 1, 2, 6, 14, 9, 11, 3, 0, 8,
5, 12, 10}

'permutasi q0 berjalan pada 8 bit
        input = input Mod (2 ^ 8)

Dim a0, a1, a2, a3, a4 AsInteger
Dim b0, b1, b2, b3, b4 AsInteger

Dim y AsInteger = 0

        a0 = Math.Floor(input / 16)
        a0 = a0 Mod (2 ^ 4)                'supaya a0 juga 4 bit
        b0 = input Mod (2 ^ 4)           'b0 hasilnya 4 bit

'a1 = a0 _xor b0
        a1 = xor_4bit(a0, b0)
'b1 = a0 _xor ROR4(b0, 1) xor_ 8a0 mod 16
        b1 = xor_4bit(xor_4bit(a0,
binaryToDecimal(ROR(decimalToBinary4bit(b0), 1))), ((8 * a0) Mod
16))

'a2, b2 = t0[a1], t1[b1]
For i AsInteger = 0 To t0.Length - 1 Step 1
            a1 = xor_4bit(a1, t0(i))
Next
```

```
        a2 = a1

For i AsInteger = 0 To t1.Length - 1 Step 1
            b1 = xor_4bit(b1, t1(i))
Next
        b2 = b1

'a3 = a2 xor_ b2
        a3 = xor_4bit(a2, b2)
'b3 = a2 _xor ROR4(b2, 1) xor_ 8a2 mod 16
        b3 = xor_4bit(xor_4bit(a2,
binaryToDecimal(ROR(decimalToBinary4bit(b2), 1))), ((8 * a2) Mod
16))

'a4, b4 = t2[a3],t3[b3]
For i AsInteger = 0 To t2.Length - 1 Step 1
            a3 = xor_4bit(a3, t2(i))
Next
        a4 = a3

For i AsInteger = 0 To t3.Length - 1 Step 1
            b3 = xor_4bit(b3, t3(i))
Next
        b4 = b3

'y = 16 b4 + a4
        y = ((16 * b4) + a4)

Return y
EndFunction

PrivateFunction permutasi_q1(ByVal input AsLong) AsInteger
'Similarly, for q1 the 4-bit S-boxes are given by
't0 = [ 2 8 B D F 7 6 E 3 1 9 4 0 A C 5 ]
't1 = [ 1 E 2 B 4 C 3 7 6 D A 5 F 9 0 8 ]
't2 = [ 4 C 7 5 1 6 9 A 0 E D 8 2 B 3 F ]
't3 = [ B 9 5 1 C 3 D E 6 4 7 F 2 0 8 A ]

't0 = [ 2 8 B  D  F  7 6 E  3 1 9 4 0 A  C  5 ]
't0 = [ 2 8 11 13 15 7 6 14 3 1 9 4 0 10 12 5 ]
Dim t0 AsInteger() = {2, 8, 11, 13, 15, 7, 6, 14, 3, 1, 9, 4, 0,
10, 12, 5}

't1 = [ 1 E  2 B  4 C  3 7 6 D  A  5  F 9 0 8 ]
't1 = [ 1 14 2 11 4 12 3 7 6 13 10 5 15 9 0 8 ]
Dim t1 AsInteger() = {1, 14, 2, 11, 4, 12, 3, 7, 6, 13, 10, 5, 15,
9, 0, 8}

't2 = [ 4 C  7 5 1 6 9 A  0 E  D  8 2 B  3 F  ]
't2 = [ 4 12 7 5 1 6 9 10 0 14 13 8 2 11 3 15 ]
Dim t2 AsInteger() = {4, 12, 7, 5, 1, 6, 9, 10, 0, 14, 13, 8, 2,
11, 3, 15}

't3 = [ B  9 5 1 C  3 D  E  6 4 7 F  2 0 8 A  ]
't3 = [ 11 9 5 1 12 3 13 14 6 4 7 15 2 0 8 10 ]
```

```
Dim t3 AsInteger() = {11, 9, 5, 1, 12, 3, 13, 14, 6, 4, 7, 15, 2,
0, 8, 10}

'permutasi q1 berjalan pada 8 bit
        input = input Mod (2 ^ 8)

Dim a0, a1, a2, a3, a4 AsInteger
Dim b0, b1, b2, b3, b4 AsInteger

Dim y AsLong = 0

        a0 = Math.Floor(input / 16)
        a0 = a0 Mod (2 ^ 4)                    'supaya a0 juga 4 bit
        b0 = input Mod (2 ^ 4)                 'b0 hasilnya 4 bit

'a1 = a0 _xor b0
        a1 = xor_4bit(a0, b0)
'b1 = a0 _xor ROR4(b0, 1) xor_ 8a0 mod 16
        b1 = xor_4bit(xor_4bit(a0,
binaryToDecimal(ROR(decimalToBinary4bit(b0), 1))), ((8 * a0) Mod
16))

'a2, b2 = t0[a1], t1[b1]
For i AsInteger = 0 To t0.Length - 1 Step 1
            a1 = xor_4bit(a1, t0(i))
Next
        a2 = a1

For i AsInteger = 0 To t1.Length - 1 Step 1
            b1 = xor_4bit(b1, t1(i))
Next
        b2 = b1

'a3 = a2 xor_ b2
        a3 = xor_4bit(a2, b2)
'b3 = a2 _xor ROR4(b2, 1) xor_ 8a2 mod 16
        b3 = xor_4bit(xor_4bit(a2,
binaryToDecimal(ROR(decimalToBinary4bit(b2), 1))), ((8 * a2) Mod
16))

'a4, b4 = t2[a3],t3[b3]
For i AsInteger = 0 To t2.Length - 1 Step 1
            a3 = xor_4bit(a3, t2(i))
Next
        a4 = a3

For i AsInteger = 0 To t3.Length - 1 Step 1
            b3 = xor_4bit(b3, t3(i))
Next
        b4 = b3

'y = 16 b4 + a4
        y = ((16 * b4) + a4)
Return y
EndFunction
```

```
Function paddingkey(ByVal key AsString) AsString
If key.Length <= 16 Then'padding panjang kunci menjadi 128 bit
For i AsInteger = key.Length To 15 Step 1
                key &= "0"
Next
ElseIf key.Length <= 24 Then'padding panjang kunci menjadi 192 bit
For i AsInteger = key.Length To 23 Step 1
                key &= "0"
Next
ElseIf key.Length <= 32 Then'padding panjang kunci menjadi 256 bit
For i AsInteger = key.Length To 31 Step 1
                key &= "0"
Next
Else
            key = ""
EndIf

Return key ' & " " & key.Length
EndFunction

Function key_schedule(ByVal key AsString) AsLong()
Dim key_material AsString = paddingkey(key)
Dim N AsInteger = key_material.Length * 8
Dim k AsInteger = N / 64
Dim M() AsString
Dim Mi() AsLong
Dim M3() AsLong
Dim M0() AsLong
Dim nm AsInteger = 8 * k - 1
Dim iterasi AsInteger = 2 * k - 1

'   RS dalam Hexadecimal
'   RS =
'       01 A4 55 87 5A 58 DB 9E
'       A4 56 82 F3 1E C6 68 E5
'       02 A1 FC C1 47 AE 3D 19
'       A4 55 87 5A 58 DB 9E 03

Dim RS(,) AsLong = {
                                    {1, 168, 85, 135, 90, 88, 219,
158},
                                    {168, 86, 130, 243, 30, 198,
104, 229},
                                    {2, 161, 252, 193, 71, 174,
61, 25},
                                    {168, 85, 135, 90, 88, 219,
158, 3}
                                }

Dim si((k - 1), 3) AsLong
Dim S(k - 1) AsLong

Dim p AsLong = 0
Dim key_expand(39) AsLong

ReDim M(nm)
```

```
ReDim Mi(iterasi)
ReDim M3(Int(iterasi / 2))
ReDim M0(Int(iterasi / 2))

For i AsInteger = 0 To nm Step 1
            M(i) = key_material.Substring(i, 1)
Next

For i AsInteger = 0 To iterasi Step 1
Dim isiM AsLong = 0
For j AsInteger = 0 To 3 Step 1
                isiM += CLng(Asc(M(4 * i + j))) * 2 ^ (8 * j)
Next
            Mi(i) = isiM
Next

Dim cM3 AsInteger = 0
Dim cM0 AsInteger = 0

For i AsInteger = 0 To iterasi Step 1
If i Mod 2 = 0 Then
                M3(cM3) = Mi(i)
                cM3 += 1
ElseIf i Mod 2 = 1 Then
                M0(cM0) = Mi(i)
                cM0 += 1
EndIf
Next

'Pembuatan Vektor S --> (dibuat fungsi sendiri karena digunakan
pada fungsi h)
'For i As Integer = 0 To (k - 1) Step 1
'    Dim isiSi As Long = 0
'    For j As Integer = 0 To 3 Step 1
'        For jj As Integer = 0 To 7 Step 1
'            isiSi += RS(j, jj) * CLng(Asc(M(8 * i + jj)))
'        Next
'        si(i, j) = isiSi
'    Next
'Next

'For i As Integer = 0 To S.Length - 1 Step 1
'    Dim isiS As Long = 0
'    For j As Integer = 0 To 3 Step 1
'        isiS += si(i, j) * 2 ^ (8 * j)
'    Next
'    S(i) = isiS
'Next
'Array.Reverse(S)

        p = 2 ^ 24 + 2 ^ 16 + 2 ^ 8 + 2 ^ 0

For i AsInteger = 0 To 19 Step 1
Dim Ai AsLong = 0
Dim Bi AsLong = 0
            Ai = fungsi_h((2 * i * p), M3)
```

```
            Bi =
binaryToDecimal(ROL(decimalToBinary32bit(fungsi_h(((2I + 1) * p),
M0)), 8))
            key_expand(2 * i) = (Ai + Bi) Mod (2 ^ 32)
            key_expand(2 * i + 1) =
binaryToDecimal(ROL(decimalToBinary32bit((Ai + 2 * Bi) Mod (2 ^
32)), 9))
Next

Return key_expand
EndFunction

PrivateFunction fungsi_h(ByVal inputX AsLong, ByVal listL()
AsLong) AsLong
'input 32 bit
'banyaknya isi dalam list l merupakan panjang dari k (k = 2 list l
juga berisi 2)
Dim k AsInteger = listL.Length

Dim l(k - 1, 3) AsLong
Dim x(3) AsLong
Dim yk(3) AsLong

Dim z(3) AsLong

'   01 EF 5B 5B
'   5B EF EF 01
'   EF 5B 01 EF
'   EF 01 EF 5B
Dim mds(,) AsLong = {
                                {1, 239, 91, 91},
                                {91, 239, 239, 1},
                                {239, 91, 1, 239},
                                {239, 1, 239, 91}
                            }


For i AsInteger = 0 To k - 1 Step 1
For j AsInteger = 0 To 3 Step 1
            l(i, j) = Math.Floor(listL(i) / 2 ^ (8 * j)) Mod
(2 ^ 8)
Next
Next

For j AsInteger = 0 To 3 Step 1
            x(j) = Math.Floor((inputX) / 2 ^ (8 * j)) Mod (2 ^ 8)
Next

For i AsInteger = 0 To 3 Step 1
            yk(i) = x(i)
Next

If k = 4 Then
'y(3,i)
```

```
            yk(0) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q1(yk(0))
), decimalToBinary32bit(l(3, 0))))
            yk(1) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q0(yk(1))
), decimalToBinary32bit(l(3, 1))))
            yk(2) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q0(yk(2))
), decimalToBinary32bit(l(3, 2))))
            yk(3) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q1(yk(3))
), decimalToBinary32bit(l(3, 3))))
EndIf

If k >= 3 Then
'y(2, i)
            yk(0) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q1(yk(0))
), decimalToBinary32bit(l(2, 0))))
            yk(1) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q1(yk(1))
), decimalToBinary32bit(l(2, 1))))
            yk(2) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q0(yk(2))
), decimalToBinary32bit(l(2, 2))))
            yk(3) =
binaryToDecimal(xorstring(decimalToBinary32bit(permutasi_q0(yk(3))
), decimalToBinary32bit(l(2, 3))))
EndIf

        yk(0) =
permutasi_q1(binaryToDecimal(xorstring(permutasi_q0(binaryToDecima
l(xorstring(decimalToBinary32bit(permutasi_q0(yk(0))),
decimalToBinary32bit(l(1, 0))))), decimalToBinary32bit(l(0, 0)))))
        yk(1) =
permutasi_q0(binaryToDecimal(xorstring(permutasi_q0(binaryToDecima
l(xorstring(decimalToBinary32bit(permutasi_q1(yk(1))),
decimalToBinary32bit(l(1, 1))))), decimalToBinary32bit(l(0, 1)))))
        yk(2) =
permutasi_q1(binaryToDecimal(xorstring(permutasi_q1(binaryToDecima
l(xorstring(decimalToBinary32bit(permutasi_q0(yk(2))),
decimalToBinary32bit(l(1, 2))))), decimalToBinary32bit(l(0, 2)))))
        yk(3) =
permutasi_q0(binaryToDecimal(xorstring(permutasi_q1(binaryToDecima
l(xorstring(decimalToBinary32bit(permutasi_q1(yk(3))),
decimalToBinary32bit(l(1, 3))))), decimalToBinary32bit(l(0, 3)))))

For i AsInteger = 0 To 3 Step 1
Dim isi AsLong = 0
For j AsInteger = 0 To 3 Step 1
                isi += mds(i, j) * yk(j)
Next
            z(i) = isi
Next

Dim Zz AsLong = 0
```

```
For i AsInteger = 0 To z.Length - 1 Step 1
            Zz += z(i) * (2 ^ (8 * i))
Next

Return Zz
EndFunction

Function make_S_vektor(ByVal key AsString) AsLong()
Dim key_material AsString = paddingkey(key)
Dim N AsInteger = key_material.Length * 8
Dim k AsInteger = N / 64
Dim nm AsInteger = 8 * k - 1
Dim M(nm) AsString
'   RS dalam Hexadecimal
'   RS =
'       01 A4 55 87 5A 58 DB 9E
'       A4 56 82 F3 1E C6 68 E5
'       02 A1 FC C1 47 AE 3D 19
'       A4 55 87 5A 58 DB 9E 03

Dim RS(,) AsLong = {
                                    {1, 168, 85, 135, 90, 88, 219,
158},
                                    {168, 86, 130, 243, 30, 198,
104, 229},
                                    {2, 161, 252, 193, 71, 174,
61, 25},
                                    {168, 85, 135, 90, 88, 219,
158, 3}
                                    }

Dim si((k - 1), 3) AsLong
Dim S(k - 1) AsLong

For i AsInteger = 0 To nm Step 1
            M(i) = key_material.Substring(i, 1)
Next

For i AsInteger = 0 To (k - 1) Step 1
Dim isiSi AsLong = 0
For j AsInteger = 0 To 3 Step 1
For jj AsInteger = 0 To 7 Step 1
                    isiSi += RS(j, jj) * CLng(Asc(M(8 * i + jj)))
Next
                si(i, j) = isiSi
Next
Next

For i AsInteger = 0 To S.Length - 1 Step 1
Dim isiS AsLong = 0
For j AsInteger = 0 To 3 Step 1
                isiS += si(i, j) * 2 ^ (8 * j)
Next
'Supaya hasil dari vektor s bernilai 4 byte karena hasil dari
perhitungan diperoleh 6 byte
```

```
            S(i) = isiS Mod (2 ^ 32)
Next
'membalik urutan vektor S
Array.Reverse(S)

Return S
EndFunction

Function sbox(ByVal input_kotak_s AsInteger(), ByVal vektor_s
AsLong()) AsInteger()
Dim output_kotak_s(3) AsInteger
Dim k AsInteger = vektor_s.Length

Dim vektor_s_bin(vektor_s.Length - 1) AsString
Dim vektor_s_8bit(vektor_s.Length - 1, 3) AsInteger

'vektor s dalam 32 bit ditampung dalam string
For i AsInteger = 0 To vektor_s_bin.Length - 1 Step 1
            vektor_s_bin(i) = decimalToBinary32bit(vektor_s(i))
Next

For i AsInteger = 0 To vektor_s_bin.Length - 1 Step 1
For j AsInteger = 0 To 3 Step 1
                vektor_s_8bit(i, j) =
binaryToDecimal(vektor_s_bin(i).Substring(j * 8, 8))
Next
Next

If k = 4 Then
'output = q0(S0,0 xor q1(S1,0 xor q1(S2,0 xor q0(S3,0 xor
q1(input))))
'output = q1(S0,1 xor q1(S1,1 xor q0(S2,1 xor q0(S3,1 xor
q0(input))))
'output = q0(S0,2 xor q0(S1,2 xor q1(S2,2 xor q1(S3,2 xor
q0(input))))
'output = q1(S0,3 xor q0(S1,3 xor q0(S2,3 xor q1(S3,3 xor
q1(input))))

            output_kotak_s(0) =
permutasi_q0(xor_8bit(vektor_s_8bit(0, 0),
permutasi_q1(xor_8bit(vektor_s_8bit(1, 0),
permutasi_q1(xor_8bit(vektor_s_8bit(2, 0),
permutasi_q0(xor_8bit(vektor_s_8bit(3, 0),
permutasi_q1(input_kotak_s(0)))))))))
            output_kotak_s(1) =
permutasi_q1(xor_8bit(vektor_s_8bit(0, 1),
permutasi_q1(xor_8bit(vektor_s_8bit(1, 1),
permutasi_q0(xor_8bit(vektor_s_8bit(2, 1),
permutasi_q0(xor_8bit(vektor_s_8bit(3, 1),
permutasi_q0(input_kotak_s(1)))))))))
            output_kotak_s(2) =
permutasi_q0(xor_8bit(vektor_s_8bit(0, 2),
permutasi_q0(xor_8bit(vektor_s_8bit(1, 2),
permutasi_q1(xor_8bit(vektor_s_8bit(2, 2),
permutasi_q1(xor_8bit(vektor_s_8bit(3, 2),
permutasi_q0(input_kotak_s(2)))))))))
```

```
            output_kotak_s(3) =
permutasi_q1(xor_8bit(vektor_s_8bit(0, 3),
permutasi_q0(xor_8bit(vektor_s_8bit(1, 3),
permutasi_q0(xor_8bit(vektor_s_8bit(2, 3),
permutasi_q1(xor_8bit(vektor_s_8bit(3, 3),
permutasi_q1(input_kotak_s(3)))))))))

ElseIf k = 3 Then
'output = q0(S0,0 xor q1(S1,0 xor q1(S2,0 xor q0(input)))
'output = q1(S0,1 xor q1(S1,1 xor q0(S2,1 xor q0(input)))
'output = q0(S0,2 xor q0(S1,2 xor q1(S2,2 xor q1(input)))
'output = q1(S0,3 xor q0(S1,3 xor q0(S2,3 xor q1(input)))

            output_kotak_s(0) =
permutasi_q0(xor_8bit(vektor_s_8bit(0, 0),
permutasi_q1(xor_8bit(vektor_s_8bit(1, 0),
permutasi_q1(xor_8bit(vektor_s_8bit(2, 0),
permutasi_q0(input_kotak_s(0)))))))
            output_kotak_s(1) =
permutasi_q1(xor_8bit(vektor_s_8bit(0, 1),
permutasi_q1(xor_8bit(vektor_s_8bit(1, 1),
permutasi_q0(xor_8bit(vektor_s_8bit(2, 1),
permutasi_q0(input_kotak_s(1)))))))
            output_kotak_s(2) =
permutasi_q0(xor_8bit(vektor_s_8bit(0, 2),
permutasi_q0(xor_8bit(vektor_s_8bit(1, 2),
permutasi_q1(xor_8bit(vektor_s_8bit(2, 2),
permutasi_q1(input_kotak_s(2)))))))
            output_kotak_s(3) =
permutasi_q1(xor_8bit(vektor_s_8bit(0, 3),
permutasi_q0(xor_8bit(vektor_s_8bit(1, 3),
permutasi_q0(xor_8bit(vektor_s_8bit(2, 3),
permutasi_q1(input_kotak_s(3)))))))

ElseIf k = 2 Then

'output = q0(S0,0 xor q1(S1,0 xor q1(input))
'output = q1(S0,1 xor q1(S1,1 xor q0(input))
'output = q0(S0,2 xor q0(S1,2 xor q1(input))
'output = q1(S0,3 xor q0(S1,3 xor q0(input))

            output_kotak_s(0) =
permutasi_q0(xor_8bit(vektor_s_8bit(0, 0),
permutasi_q1(xor_8bit(vektor_s_8bit(1, 0),
permutasi_q1(input_kotak_s(0))))))
            output_kotak_s(1) =
permutasi_q1(xor_8bit(vektor_s_8bit(0, 1),
permutasi_q1(xor_8bit(vektor_s_8bit(1, 1),
permutasi_q0(input_kotak_s(1))))))
            output_kotak_s(2) =
permutasi_q0(xor_8bit(vektor_s_8bit(0, 2),
permutasi_q0(xor_8bit(vektor_s_8bit(1, 2),
permutasi_q1(input_kotak_s(2))))))
            output_kotak_s(3) =
permutasi_q1(xor_8bit(vektor_s_8bit(0, 3),
```

```
permutasi_q0(xor_8bit(vektor_s_8bit(1, 3),
permutasi_q0(input_kotak_s(3))))))

EndIf

Return output_kotak_s
EndFunction

Function MDS(ByVal yi AsInteger()) AsLong
Dim Zi(3) AsLong
Dim matrik(,) AsDouble = {
                                    {1, 239, 91, 91},
                                    {91, 239, 239, 1},
                                    {239, 91, 1, 239},
                                    {239, 1, 239, 91}
                                }
Dim Z AsLong = 0

For i AsInteger = 0 To 3 Step 1
Dim temp_zi AsLong = 0
For j AsInteger = 0 To 3 Step 1
                temp_zi += matrik(i, j) * yi(j)
Next
            Zi(i) = temp_zi
Next

For i AsInteger = 0 To Zi.Length - 1 Step 1
            Z += Zi(i) * (2 ^ (8 * i))
Next

Return Z
EndFunction

Function PHT(ByVal T0 AsLong, ByVal T1 AsLong, ByVal K0 AsLong,
ByVal K1 AsLong) AsLong()
Dim output(1) AsLong
        output(0) = (T0 + T1 + K0) Mod (2 ^ 32)
        output(1) = (T0 + T1 + T1 + K1) Mod (2 ^ 32)
Return output
EndFunction

Function fungsi_g(ByVal key AsString, ByVal input_X AsLong) AsLong
Dim Vektor_S() AsLong = make_S_vektor(paddingkey(key))

Dim xi(3) AsInteger
For i AsInteger = 0 To xi.Length - 1 Step 1
            xi(i) = Math.Floor(input_X / 2 ^ (8 * i)) Mod (2 ^ 8)
Next

'S-Box
Dim yi() AsInteger = sbox(xi, Vektor_S)
'MDS
Dim Z AsLong = MDS(yi)

Return Z
EndFunction
```

```
Function fungsi_f(ByVal key AsString, ByVal R0 AsLong, ByVal R1
AsLong, ByVal K0 AsLong, ByVal K1 AsLong) AsLong()
'T0 = g(R0)
'T1 = g(ROL(R1; 8))

'PHT
'F0 = (T0 + T1 + K2r + 8) Mod 2 ^ 32
'F1 = (T0 + 2T1 + K2r+9) mod 2 ^ 32

Dim T1 AsLong = fungsi_g(key, R0)
Dim T2 AsLong = fungsi_g(key,
binaryToDecimal(ROL(decimalToBinary32bit(R1), 8)))

Dim F() AsLong = PHT(T1, T2, K0, K1)

Return F
EndFunction

PrivateFunction swap(ByVal RA AsLong, ByVal RB AsLong, ByVal RC
AsLong, ByVal RD AsLong) AsLong()
Dim output(3) AsLong
        output(0) = RC
        output(1) = RD
        output(2) = RA
        output(3) = RB
Return output
EndFunction


Function konversi_little_endian_p(ByVal filetext AsByte())
AsLong()
Dim hasil(3) AsLong
For i AsInteger = 0 To 3 Step 1
Dim temp AsLong = 0
For j AsInteger = 0 To 3 Step 1
            temp += filetext((4 * i) + j) * 2 ^ (8 * j)
Next
            hasil(i) = temp
Next

Return hasil
EndFunction

Function konversi_little_endian_c(ByVal input AsLong()) AsByte()
Dim chipertext(15) AsByte
For i AsInteger = 0 To 15 Step 1
            chipertext(i) = Math.Floor(input(Math.Floor(i / 4)) /
2 ^ (8 * (i Mod 4))) Mod (2 ^ 8)
Next
Return chipertext
EndFunction


Function enkrip_twofish(ByVal filetext AsByte(), ByVal key
AsString) AsByte()
```

```
Dim kunci AsLong() = key_schedule(paddingkey(key))
Dim R(16, 3) AsLong
Dim F(15, 1) AsLong
Dim input AsLong() = konversi_little_endian_p(filetext)
Dim C(3) AsLong

'INPUT WHITENING
For i AsInteger = 0 To 3 Step 1
            R(0, i) =
binaryToDecimal(xorstring(decimalToBinary32bit(input(i)), _

decimalToBinary32bit(kunci(i))))
Next

For putaran AsInteger = 0 To 15 Step 1
'(Fr,0, Fr,1) = F(Rr,0, Rr,1, r)
'Rr+1,0 = ROR((Rr,2 xor_ Fr,0), 1)
'Rr+1,1 = ROL(Rr,3, 1) xor_ Fr;1
'Rr+1,2 = Rr,0
'Rr+1,3 = Rr,1

Dim temp_round AsLong() = fungsi_f(key, R(putaran, 0), R(putaran,
1), _
                                                kunci((2 *
putaran) + 8), kunci((2 * putaran) + 9))
            F(putaran, 0) = temp_round(0)
            F(putaran, 1) = temp_round(1)

            R(putaran + 1, 0) =
binaryToDecimal(ROR(xorstring(decimalToBinary32bit(R(putaran, 2)),
_

decimalToBinary32bit(F(putaran, 0))), 1))
            R(putaran + 1, 1) =
binaryToDecimal(xorstring(ROL(decimalToBinary32bit(R(putaran, 3)),
1), _

decimalToBinary32bit(F(putaran, 1))))
            R(putaran + 1, 2) = R(putaran, 0)
            R(putaran + 1, 3) = R(putaran, 1)
Next

'UNDO SWAP
Dim temp AsLong() = swap(R(16, 0), R(16, 1), R(16, 2), R(16, 3))
        R(16, 0) = temp(0)
        R(16, 1) = temp(1)
        R(16, 2) = temp(2)
        R(16, 3) = temp(3)

'OUPUT WHITENING
For i AsInteger = 0 To 3 Step 1
Dim j AsInteger = (i + 2) Mod 4
            C(i) =
binaryToDecimal(xorstring(decimalToBinary32bit(R(16, j)), _

decimalToBinary32bit(kunci(i + 4))))
```

```
Next

Dim chipertext AsByte() = konversi_little_endian_c(C)

Return chipertext
EndFunction

Function dekrip_twofish(ByVal chipertext AsByte(), ByVal key
AsString) AsByte()
Dim kunci AsLong() = key_schedule(paddingkey(key))
Dim RD(16, 3) AsLong
Dim FD(15, 1) AsLong
Dim chiper AsLong() = konversi_little_endian_p(chipertext)
Dim P(3) AsLong

'outwhit
For i AsInteger = 0 To 3 Step 1
Dim j AsInteger = (i + 2) Mod 4
'R(16, j) =
binaryToDecimal(xorstring(decimalToBinary32bit(chiper(i)),
decimalToBinary32bit(kunci(i + 4))))
            RD(16, j) =
binaryToDecimal(xorstring(decimalToBinary32bit(chiper(i)), _

decimalToBinary32bit(kunci(i + 4))))
Next

Dim temp_swap AsLong() = swap(RD(16, 0), RD(16, 1), RD(16, 2),
RD(16, 3))
        RD(16, 0) = temp_swap(0)
        RD(16, 1) = temp_swap(1)
        RD(16, 2) = temp_swap(2)
        RD(16, 3) = temp_swap(3)

For putaran AsInteger = 15 To 0 Step -1
'(F(r,0), F(r,1)) = F(R(r,0), R(r,1), r)
'R(r+1,0) = ROR((R(r,2) xor_ F(r,0)), 1)
'R(r+1,1) = ROL(R(r,3), 1) xor_ F(r,1)
'R(r+1,2) = R(r,0)
'R(r+1,3) = R(r,1)
'================================================================
================================================================
======
            RD(putaran, 0) = RD(putaran + 1, 2)
            RD(putaran, 1) = RD(putaran + 1, 3)

Dim temp_round AsLong() = fungsi_f(key, RD(putaran, 0),
RD(putaran, 1), _
                                                    kunci((2 *
(putaran)) + 8), kunci((2 * (putaran)) + 9))
            FD(putaran, 0) = temp_round(0)
            FD(putaran, 1) = temp_round(1)

            RD(putaran, 2) =
binaryToDecimal(xorstring(ROL(decimalToBinary32bit(RD(putaran + 1,
0)), 1), _
```

```
decimalToBinary32bit(FD(putaran, 0))))
            RD(putaran, 3) =
binaryToDecimal(ROR(xorstring(decimalToBinary32bit(RD(putaran + 1,
1)), _

decimalToBinary32bit(FD(putaran, 1))), 1))


'INPUT WHITENING
For i AsInteger = 0 To 3 Step 1
            RD(0, i) =
binaryToDecimal(xorstring(decimalToBinary32bit(RD(0, i)), _

decimalToBinary32bit(kunci(i))))
            P(i) = RD(0, i)
Next

Dim plaintext AsByte() = konversi_little_endian_c(P)

Return plaintext
EndFunction
EndModule
```

C. Module LSB

```
Imports System.IO
Imports System.Drawing
Imports System.Drawing.Imaging

Modulelsb
Function lsb_insert(ByRef lokasi_citra AsString, ByVal isi_file AsByte())
AsBoolean
Dim wadah AsNewBitmap(lokasi_citra)

Dim panjang_pesan AsString = decimalToBinary16bit(isi_file.Length)
Dim pesan_rahasia AsByte()

'1 byte untuk identitas gambar yang disisipi pesan
Dim identitas AsInteger = 220
Dim counter_pesan AsInteger = 0
ReDim pesan_rahasia(counter_pesan)
        pesan_rahasia(counter_pesan) = identitas

'2 byte untuk panjang pesan yang disisipi
        counter_pesan += 1
ReDimPreserve pesan_rahasia(counter_pesan)
        pesan_rahasia(counter_pesan) =
binaryToDecimal(panjang_pesan.Substring(0, 8))
        counter_pesan += 1
ReDimPreserve pesan_rahasia(counter_pesan)
        pesan_rahasia(counter_pesan) =
binaryToDecimal(panjang_pesan.Substring(8, 8))

'byte selanjutnya diisi dengan pesan
For i AsInteger = 0 To isi_file.Length - 1 Step 1
            counter_pesan += 1
ReDimPreserve pesan_rahasia(counter_pesan)
            pesan_rahasia(counter_pesan) = isi_file(i)
```

```
Next


Dim ext AsString = Path.GetExtension(lokasi_citra)
Dim lokasi AsString = Path.GetDirectoryName(lokasi_citra)
Dim nama_file AsString = Path.GetFileName(lokasi_citra)

Try
Dim pesan AsString = ""
Dim counter AsInteger = 0

For i AsInteger = 0 To pesan_rahasia.Length - 1 Step 1
                pesan &= byteToBinary(pesan_rahasia(i))
Next

Dim done AsBoolean = False

For y AsInteger = 0 To wadah.Height - 1
For x AsInteger = 0 To wadah.Width - 1
Dim enough AsBoolean = False

Dim p AsColor = wadah.GetPixel(x, y)

'pixel merah
Dim merah AsString = decimalToBinary(p.R.ToString)
If enough = FalseThen
                        merah = merah.Remove(7, 1).Insert(7,
pesan(counter))
                        counter += 1
EndIf
If counter = pesan.Length Then
                        enough = True
EndIf

'pixel hijau
Dim hijau AsString = decimalToBinary(p.G.ToString)
If enough = FalseThen
                        hijau = hijau.Remove(7, 1).Insert(7,
pesan(counter))
                        counter += 1
EndIf
If counter = pesan.Length Then
                        enough = True
EndIf

'pixel biru
Dim biru AsString = decimalToBinary(p.B.ToString)
If enough = FalseThen
                        biru = biru.Remove(7, 1).Insert(7,
pesan(counter))
                        counter += 1
EndIf
If counter = pesan.Length Then
                        enough = True
EndIf

'penempatan kembali pixel
                       wadah.SetPixel(x, y,
Color.FromArgb(CInt(binaryToDecimal(merah)),
CInt(binaryToDecimal(hijau)), CInt(binaryToDecimal(biru))))

If enough Then
                        done = True
```

```
Exit For
EndIf
Next
If done ThenExit For
Next
Catch ex AsException
MessageBox.Show("FAILED! (penyisipan)")
'MessageBox.Show("Penyisipan pesan gagal!")
EndTry

Try
If ext = ".bmp"Or ext = ".BMP"Then
                wadah.Save(lokasi &"\_"& nama_file, ImageFormat.Bmp)
ElseIf ext = ".png"Or ext = ".PNG"Then
                wadah.Save(lokasi &"\_"& nama_file, ImageFormat.Png)

'ElseIf ext = ".jpeg" Or ext = ".JPEG" Or ext = ".jpg" Or ext = ".JPG"
Then
'    wadah.Save(lokasi & "\_" & nama_file, ImageFormat.Jpeg)
' tidak berjalan semestinya

EndIf
ReturnTrue
Catch ex AsException
ReturnFalse
EndTry
EndFunction

Function lsb_ekstrasi(ByVal lokasi_citra AsString) AsByte()
Dim citra_berpesan AsNewBitmap(lokasi_citra)
Dim lokasi AsString = Path.GetDirectoryName(lokasi_citra)
Dim pesan_rahasia AsString = ""
Dim hasil AsByte()
Dim counter AsInteger = -1
ReDim hasil(counter)
Dim header AsString = ""
Dim hanya_pesan AsByte()
Dim counter_pesan_asli AsInteger = -1
ReDim hanya_pesan(counter_pesan_asli)

Try
'cek identitas gambar yang disisipi
For y AsInteger = 0 To citra_berpesan.Height - 1
Dim cek_header AsBoolean = False
For x AsInteger = 0 To citra_berpesan.Width - 1
Dim p AsColor = citra_berpesan.GetPixel(x, y)
Dim merah AsString = decimalToBinary(p.R.ToString)
                header &= merah(7)

Dim hijau AsString = decimalToBinary(p.G.ToString)
                header &= hijau(7)

Dim biru AsString = decimalToBinary(p.B.ToString)
                header &= biru(7)

If header.Length = 24 Then
                    cek_header = True
Exit For
EndIf
Next
If cek_header ThenExit For
Next
```

```
'cek identitas gambar apakah disisipi pesan atau tidak
If binaryToDecimal(header.Substring(0, 8)) <> 220 Then
MessageBox.Show("Gambar tidak disisipi pesan!")
Else
'hitung panjang pesan untuk perulangan
'+24 karena 24bit --> untuk menyimpan identitas dan panjang pesan
Dim panjang_pesan_bit AsInteger = binaryToDecimal(header.Substring(8,
16)) * 8 + 24

Dim perulangan AsInteger = 0

For y AsInteger = 0 To citra_berpesan.Height - 1
Dim pesan_selesai AsBoolean = False
For x AsInteger = 0 To citra_berpesan.Width - 1
Dim p AsColor = citra_berpesan.GetPixel(x, y)

'pixel merah
Dim merah AsString = decimalToBinary(p.R.ToString)
                        pesan_rahasia &= merah(7)
If pesan_rahasia.Length = 8 Then
                            counter += 1
ReDimPreserve hasil(counter)
hasil(counter) = binaryToDecimal(pesan_rahasia)
                            pesan_rahasia = ""
EndIf
                        perulangan += 1
If perulangan = panjang_pesan_bit Then
                            pesan_selesai = True
Exit For
EndIf

'pixel hijau
Dim hijau AsString = decimalToBinary(p.G.ToString)
                        pesan_rahasia &= hijau(7)
If pesan_rahasia.Length = 8 Then
                            counter += 1

ReDimPreserve hasil(counter)
hasil(counter) = binaryToDecimal(pesan_rahasia)
                            pesan_rahasia = ""

EndIf
                        perulangan += 1
If perulangan = panjang_pesan_bit Then
                            pesan_selesai = True

Exit For
EndIf

'pixel biru
Dim biru AsString = decimalToBinary(p.B.ToString)
                        pesan_rahasia &= biru(7)
If pesan_rahasia.Length = 8 Then
                            counter += 1
ReDimPreserve hasil(counter)
hasil(counter) = binaryToDecimal(pesan_rahasia)
                            pesan_rahasia = ""
EndIf
                        perulangan += 1
If perulangan = panjang_pesan_bit Then
                            pesan_selesai = True
Exit For
EndIf

Next
If pesan_selesai ThenExit For
```

```
Next

For i AsInteger = 3 To hasil.Length - 1 Step 1
                    counter_pesan_asli += 1
ReDimPreserve hanya_pesan(counter_pesan_asli)
                    hanya_pesan(counter_pesan_asli) = hasil(i)
Next

EndIf
Catch ex AsException
MessageBox.Show("FAILED! (ekstrasi gagal)")
EndTry

Return hanya_pesan
EndFunction
EndModule
```

D. Module Conveter

```
ModuleConveter
Function decimalToBinary4bit(ByVal des AsString) AsString
Dim dec AsInteger = CType(des, Integer)
Dim bin AsString = Convert.ToString(dec, 2)
Return bin.PadLeft(4, "0"c)
EndFunction

Function decimalToBinary(ByVal des AsString) AsString
Dim dec AsInteger = CType(des, Integer)
Dim bin AsString = Convert.ToString(dec, 2)
Return bin.PadLeft(8, "0"c)
EndFunction

Function decimalToBinary16bit(ByVal des AsString) AsString
Dim dec AsInteger = CType(des, Integer)
Dim bin AsString = Convert.ToString(dec, 2)
Return bin.PadLeft(16, "0"c)
EndFunction

Function decimalToBinary32bit(ByVal des AsString) AsString
Dim dec AsLong = CType(des, Long)
Dim bin AsString = Convert.ToString(dec, 2)
Return bin.PadLeft(32, "0"c)
EndFunction

Function decimalToBinary48bit(ByVal des AsString) AsString
Dim dec AsLong = CType(des, Long)
Dim bin AsString = Convert.ToString(dec, 2)
Return bin.PadLeft(48, "0"c)
EndFunction

Function decimalToBinary64bit(ByVal des AsString) AsString
Dim dec AsLong = CType(des, Long)
Dim bin AsString = Convert.ToString(dec, 2)
Return bin.PadLeft(48, "0"c)
EndFunction

Function byteToBinary(ByVal b AsByte) AsString
ReturnConvert.ToString(b, 2).PadLeft(8, "0"c)
EndFunction

PublicFunction binaryToDecimal(ByVal Bin AsString) AsDouble
Dim dec AsDouble = Nothing
Dim length AsInteger = Len(Bin)
```

```
Dim temp AsInteger = Nothing
Dim x AsInteger = Nothing
For x = 1 To length
            temp = Val(Mid(Bin, length, 1))
            length = length - 1
If temp <>"0"Then
                dec += (2 ^ (x - 1))
EndIf
Next
Return dec
EndFunction
EndModule
```

E. Module FungsiPendukung

```
Imports System.IO
Imports System.Text.RegularExpressions

ModuleFungsiPendukung
Function regexpass(ByVal password AsString) AsBoolean
If password.Length >= 8 _
AndAlsoRegex.IsMatch(password, "[a-z]") _
AndAlsoRegex.IsMatch(password, "[A-Z]") _
AndAlsoRegex.IsMatch(password, "[0-9]") Then
ReturnTrue
Else
ReturnFalse
EndIf
EndFunction

Function matchpass(ByVal password1 AsString, ByVal password2 AsString)
AsBoolean
If password1 = password2 Then
ReturnTrue
Else
ReturnFalse
EndIf
EndFunction

Function IsValidImage(ByVal filename AsString) AsBoolean
Try
Dim img AsImage = Image.FromFile(filename)
Catch generatedExceptionName AsOutOfMemoryException
' Image.FromFile throws an OutOfMemoryException. if the file does not
have a valid image format or GDI+ does not support the pixel format of
the file.
ReturnFalse
EndTry
ReturnTrue
EndFunction

Function split_byte(ByVal input AsByte()) AsByte(,)
Dim baris AsInteger = 0
If input.Length Mod 16 = 0 Then
            baris = input.Length / 16
Else
            baris = Math.Ceiling(input.Length / 16)
EndIf
Dim counter AsInteger = 0
Dim hasil AsByte(,)
ReDim hasil(baris - 1, 15)
```

```
For i AsInteger = 0 To baris - 1 Step 1
For j AsInteger = 0 To 15 Step 1
Dim temp AsByte
If counter < input.Length Then
                    temp = input(counter)
                    counter += 1
Else
                    temp = 0
EndIf
                hasil(i, j) = temp
Next
Next
Return hasil
EndFunction
EndModule
```