APLIKASI ENKRIPSI DAN DEKRIPSI FILE MENGGUNAKAN ALGORITMA TWOFISH

Sry Yunarti
Program Studi Sistem Informasi
STMIK ProfesionalMakassar
yeye_rumbu@yahoo.co.id

Abstrak

Masalah keamanan dan kerahasiaan data merupakan hal yang sangat penting dalam suatu organisasi maupun pribadi. Apalagi jika data tersebut berada dalam suatu jaringan komputer yang terkoneksi dengan jaringan lain. Rawannya kejahatan dalam pencurian data dapat menyebabkan data yang berharga diakses oleh orang-orang yang tidak berhak, data diubahan secara legal, serta tidak jarang pula beberapa pihak menduplikanya. Jika hal tersebut sampai terjadi, kemungkinan besar akan merugikan pihak-pihak tertentu. Selain itu data yang dibajak tersebut akan memiliki kemungkinan rusak bahkan hilang yang akan menimbulkan kerugian material yang besar.

Tujuan penelitian ini adalah menerapkan algoritma twofish untuk mengamankan file sehingga file menjadi tidak dapat terbaca. Proses utama pada aplikasi perangkat lunak ini adalah melakukan enkripsi dan dekripsi.

Kata kunci: *Enkripsi, Dekripsi, Algoritma twofish*

PENDAHULUAN

Masalah keamanan dan kerahasiaan data merupakan hal yang sangat penting dalam suatu organisasi maupun pribadi. Apalagi jika data tersebut berada dalam suatu jaringan komputer yang terkoneksi dengan jaringan lain. Rawannya kejahatan dalam pencurian data dapat menyebabkan data yang berharga diakses oleh orang-orang yang tidak berhak, data diubahan secara legal, serta tidak jarang pula beberapa pihak menduplikanya. Jika hal tersebut sampai terjadi, kemungkinan besar akan merugikan pihak-pihak tertentu. Selain itu data yang dibajak tersebut akan memiliki kemungkinan rusak bahkan hilang yang akan menimbulkan kerugian material yang besar.

Oleh karena itu untuk menghindari agar hal tersebut tidak terjadi, digunakanlah sebuah program khusus proteksi/enkripsi data. Dewasa ini dengan arus informasi yang semakin global, kriptografi telah menjadi suatu bagian tidak dapat dipisahkan dari sistem keamanan data. Dalam kriptografi data di *scramble* menjadi kode yang tidak dapat di mengerti oleh orang lain. Kriptografi mengubah data asli menjadi data sandi yang tidak dapat dikenali. Proses mengubah data asli menjadi data sandi lebih dikenal dengan istilah Enkripsi, sebaliknya untuk mentransformasikan kembali data sandi menjadi data asli disebut Dekripsi.

DASAR TEORI

1.1 Keamanan Data

Keamanan data merupakan keadaan dimana catatan yang berisi fakta-fakta berada dalam keadaan aman dan tidak diakses oleh individu atau kelompok yang tidak berkepentingan. Berikut beberapa ancaman keamanan data, yaitu :

a. Kebocoran (*Leakage*) : pengambilan informasi oleh penerima yang tidak berhak.
b. *Tampering* : pengubahan informasi yang tidak legal atau tanpa sepengetahuan.
c. Perusakan (*Vandalism*) : adalah gangguan dari *system* operasi tertentu di mana si perusak tidak mengharapkan keuntungan apapun dari perusakan tersebut.

Dalam hal ini, keamanan data dapat dibedakan menjadi dua kategori, yaitu keamanan fisik dan keamanan sistem.

Keamanan fisik merupakan bentuk keamanan berupa fisik dari *server, terminal/ client router* sampai dengan *cabling.* Sedangkan keamanan sistem adalah keamanan pada sistem pengoperasiannya atau lebih khususnya pada lingkup perangkat lunak, misalnya dengan penggunaan kriptografi dan steganografi. Dalam laporan akhir ini akan dibahas tentang penggunaan kriptografi dalam memberikan keamanan pada data.

1.2 Kriptografi

Kriptografi (*Cryptography*) berasal dari bahasa Yunani, *Cryptos* dan *Graphien*. *Cryptos* yang berarti rahasia dan *Graphien* artinya tulisan. Schneier (1996) mendefenisikan kriptografi sebagai ilmu yang mempelajari teknik untuk menjaga keamanan pesan. Di lain pihak, Guritman (2003) mendefinisikan kriptografi sebagai studi teknik matematik yang berkaitan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, otentikasi entitas, dan otentikasi asal data. Dari penjelasan tersebut terlihat bahwa kriptografi tidak hanya sebagai alat yang memberikan keamanan informasi, melainkan juga berupa seperangkat teknik atau prosedur yang berhubungan dengan keamanan informasi.

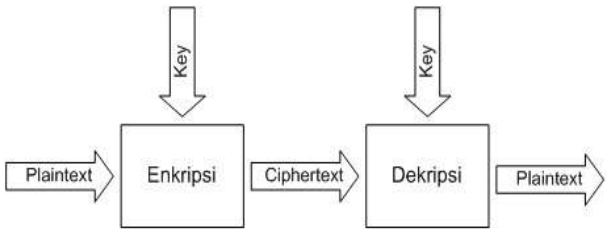
A. Enkripsi dan Dekripsi

Proses utama dalam suatu algoritma kriptografi adalah enkripsi dan dekripsi. 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 (*Chipertext*). Enkripsi bisa diartikan sebagai *Chiper* atau Kode.

Pada mode *ECB* (*Elekctronic Codebook*), sebuah blok pada plaintext dienkripsi ke dalam sebuah blok ciphertext dengan panjang blok yang sama. Blok cipher memiliki sifat bahwa setiap blok harus memiliki panjang yang sama (misalnya 128 bit). Namun apabila pesan yang dienkripsi memiliki panjang blok terakhir tidak tepat 128 bit, maka diperlukan mekanisme padding, yaitu penambahan bit-bit dummies untuk menggenapi menjadi panjang blok yang sesuai; biasanya padding dilakukan pada blok terakhir *plaintext*. Padding pada blok terakhir bisa dilakukan dengan berbagai macam cara, misalnya dengan penambahan bit-bit tertentu. Salah satu contoh penerapan padding dengan cara menambahkan jumlah total padding sebagai byte terakhir pada blok terakhir plaintext. Misalnya panjang blok adalah 128 bit (16 byte) dan pada blok terakhir terdiri dari 88 bit (11 byte) sehingga jumlah padding yang diperlukan adalah 5 byte, yaitu dengan menambahkan angka nol sebanyak 4 byte, kemudian menambahkan angka 5 sebanyak satu byte. Cara lain dapat juga menggunakan penambahan karakter *end-of-file* pada byte terakhir lalu diberi padding setelahnya.

Dekripsi merupakan proses kebalikan dari proses enkripsi, dimana pesan yang telah dienkripsi (*Chipertext*) ditransformasikan kembali ke bentuk asalnya (*Plaintext*). Untuk menghilangkan padding yang diberikan pada saat prpses enkripsi, dilakukan berdasarkan informasi jumlah padding yaitu angka pada byte terakhir.

Skema yang mengilustrasikan enkripsi dan dekripsi terlihat pada gambar 1.



Gambar 1. Skema Enkripsi dan Dekripsi

Gambar 1 mengilustrasikan sebuah *Plaintext* dienkripsi menggunakan Kunci Enkripsi sehingga menjadi *Chipertext* dan *Chipertext* didekripsi kembali menggunakan Kunci Dekripsi sehingga menjadi *Plaintext*.

B. Tujuan Kriptografi

Terdapat empat tujuan mendasar pada ilmu Kriptografi yaitu:

1. Kerahasiaan, merupakan layanan yang digunakan untuk menjaga isi informasi dari siapapun kecuali yang memiliki kunci

rahasia untuk membuka informasi yang telah di enkripsi.

2. Integritas data, berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan dan pensubtitusian data lain ke dalam data yang sebenarnya.
3. Autentikasi, berhubungan dengan identifikasi/ pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang berkmunikasi harus memperkenalkan. Informasi yang harus diautentikasi keaslian, isi datanya, waktu pengiriman dan lailn-lain.
4. *Non-repudiasi* atau penyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan/membuat.

Menurut Shannon terdapat 2 hal penting yang harus diperhatikan dalam kripto sistem yang baik yaitu :

1. Difusi : Apabila karakter dari *plaintext* ada yang diubah maka beberapa karakter dalam *ciphertext* akan berubah, sebaliknya apabila karakter dalam *ciphertext* ada yang berubah maka karakter dalam *plaintext* ikut berubah
2. Membingungkan : *Key* tidak berhubungan dengan sederhana ke *ciphertext*, yang mana setiap karakter dalam *ciphertext* tergantung dari beberapa bagian *key*.

C. Algoritma Kriptografi

Algoritma Kriptografi adalah langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut. Algoritma kriptografi berdasarkan jenis kunci dibagi menjadi:

1. Algoritma Simetris

   Algoritma simetris adalah algoritma kriptografi dimana kunci yang digunakan untuk enkripsi dan dekripsi sama. Sebelum melakukan pengiriman pesan, pengirim dan penerima harus memilih suatu kunci tertentu yang sama untuk dipakai bersama, dan kunci ini haruslah rahasia bagi pihak yang tidak berkepentingan sehingga algoritma ini disebut juga algoritma kunci rahasia.
2. Algoritma Asimetris

   Algoritma asimetris adalah algoritma kriptografi dimana kunci yang digunakan untuk enkripsi berbeda dengan kunci yang digunakan untuk dekripsi. Pada algoritma ini menggunakan dua kunci yakni kunci publik dan kunci privat. Kunci publik disebarkan secara umum sedangkan kunci privat disimpan secara rahasia oleh si pengguna. Walau kunci publik telah diketahui namun akan sangat sukar mengetahui kunci privat yang digunakan. Pada umumnya kunci publik *(public key)* digunakan sebagai kunci enkripsi sementara kunci privat *(private key)* digunakan sebagai kunci dekripsi.

Algoritma kriptografi berdasarkan mode bit dibagi menjadi :

1. *Block Cipher*

   Algoritma kriptografi ini bekerja pada suatu data yang berbentuk blok/ kelompok data dengan panjang data tertentu (dalam beberapa *byte*), jadi dalam sekali proses enkripsi atau dekripsi data yang masuk mempunyai ukuran yang sama.
2. *Stream Cipher*

   Algoritma yang dalam operasinya bekerja dalam suatu pesan berupa bit tunggal. *Stream Chiper* mengenkripsi karakter secara individu (biasanya dalam bilangan biner) dari suatu *plaintext*, menggunakan transformasi enkripsi yang bergantung pada waktu.

1.3 Algoritma Twofish

Algoritma Twofish diciptakan oleh Bruce Schneier, sebelumnya ia menciptakan algoritma blowfish dengan 64 bit *block chiper* dan kunci 128 bit. Twofish merupakan algoritma kunci simetris *blok cipher* dengan blok masukan 128 bit dan kunci 128 bit,192 bit dan 256 bit. Menurut Schneier berikut adalah unsur pembangun twofish, yaitu :

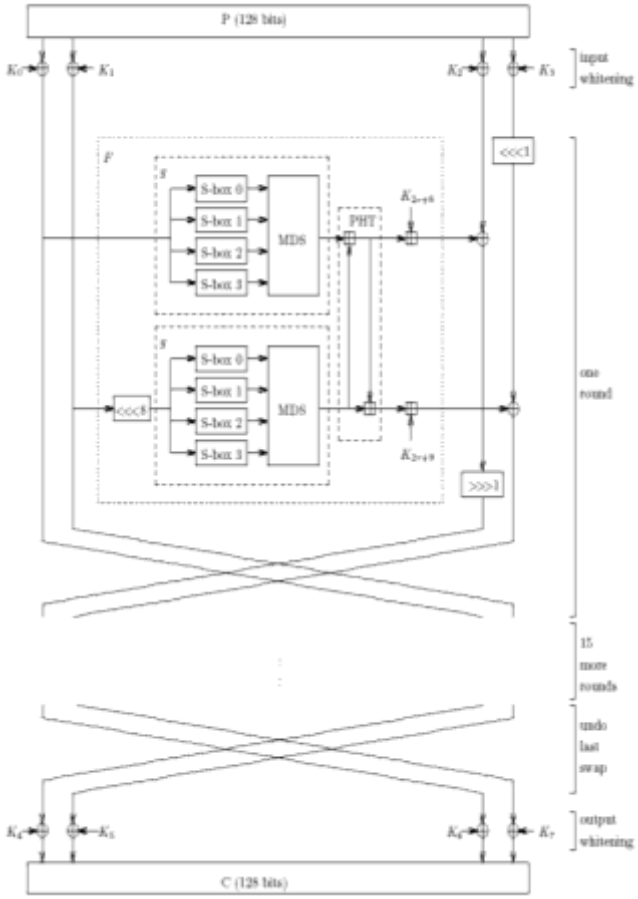
1. *Feistel Network* (Jaringan Feistel)

   Feistel Network adalah metode umum untuk mentransformasi suatu fungsi menjadi bentuk permutasi. Bagian paling fundamental dari Jaringan Feistel adalah

fungsi F yaitu sebuah pemetaan key-dependent dari suatu input string menjadi output string. Dalam twofish dilakukan *Feistel Network* sebanyak 16 kali. Pada twofish, jaringan feistel terdiri dari *Input Whitening*, S-boxes, *Transformasi Pseudo Hadamard*, dan *Output Whitening*.

2. *S-Boxes*
   Sebuah *S-box* adalah matriks yang berisi subtitusi sederhana yang memetakan satu atau lebih bit dengan satu atau lebih bit yang lain. Pada kebanyakan algoritma *Chiper* blok, *S-box* memetakan *m* bit masukan dan *n* bit keluaran (*m x n*). Twofish menggunakan empat *bijective*, *key-dependent* dan 8-by-8-bit S-boxes. S-boxes ini dibuat menggunakan dua permutasi 8-by-8-bit dan material key.
3. *Matrik MDS*
   Code Maximum Distance Separable (MDS) melalui sebuah pemetaan linear dari elemen field a ke elemen field b, menghasilkan campuran dari vektor a+b elemen, dengan properti jumlah minimum angka tidak nol dalam vector tidak nol paling kurang b+1. Pemetaan MDS dapat direpresentasikan dengan matriks MDS berdimensi a x b. *Twofish* menggunakan 4 x 4 matriks MDS tunggal.
4. *Transformasi Pseudo-Hadamard* (PHT)
   Transformasi Pseudo-Hadamard (PHT) adalah operasi pencampuran sederhana. Diberikan 2 input, a dan b. Twofish menggunakan PHT 32 bit untuk melakukan pencampuran output dari 2 *double word* yang menghasilkan fungsi g.
5. *Whitening*
   Whitening merupakan teknik mengXOR *key* material sebelum ronde pertama dan sesudah ronde terakhir. Dalam serangan terhadap Twofish, terbukti bahwa *whitening* meningkatkan kesulitan menyerang *Chipper*, dengan jalan menyembunyikan input spesifik untuk awal dan akhir ronde dari twofish.
6. *Key schedule*
   Key schedule adalah suatu cara dimana bit-bit *key* diubah menjadi *key-key* bulat setiap putaran yang digunakan oleh *chipper*. Twofish memerlukan *material key* yang sangat banyak, dan memiliki *key schedule* yang rumit. Jadi secara singkat, *key schedule* (penjadwalan kunci) adalah proses pengacakan kunci untuk melakukan proses enkripsi sehingga tingkat kerumitannya menjadi tinggi.

Blok diagram *Twofish* terlihat pada Gambar 2



Gambar 2 Blok diagram *Twofish*

Twofish menggunakan sebuah Struktur *Feistel-like 16-round* dengan tambahan whitening pada masukan dan keluaran. Satu-satunya unsur *non-Feistel adalah 1-bit* rotasi. Perputaran dapat dipindah ke dalam fungsi F untuk membuat suatu struktur Feistel murni, tapi memerlukan suatu tambahan perputaran kata-kata yang tepat sebelum langkah keluaran *whitening*. *Plaintext* dipecah menjadi empat kata 32-bit. Pada langkah *whitening* masukan terdapat *xored* dengan empat kata kunci. Ini diikuti oleh 16 putaran. Pada setiap putaran, dua kata-kata pada sisi kiri digunakan sebagai masukan kepada fungsi g (salah satu darinya diputar pada 8 bit pertama). Fungsi g terdiri dari empat *byte-wide S-Box key-dependent*, yang diikuti oleh suatu langkah pencampuran linier berdasar pada suatu matriks *MDS*. Hasil kedua fungsi g dikombinasikan menggunakan suatu *Pseudo*

Hadamard Transform (PHT), dan ditambahkan dua kata kunci. Kedua hasil ini kemudian di-XOR ke dalam kata-kata pada sisi kanan (salah satunya diputar ke kanan 1 bit pertama, yang lainnya diputar ke kanan setelahnya). Yang kiri dan kanan dibelah dua kemudian ditukar untuk putaran yang berikutnya, pertukaran yang terakhir adalah dibalik, dan yang empat kata di-XOR dengan lebih dari empat kata kunci untuk menghasilkan *ciphertext*. Secara formal, 16byte *plaintext* $p_{0,}\ldots,p_{15}$ yang yang pertama dipecah menjadi 4 kata $P_0,\ldots,P_3$ dari 32bit masing-masing menggunakan konvensi little-endian.

Di dalam langkah *whitening, double word* ini di-XOR dengan 4 *word* dari kunci yang diperluas. Pada setiap 16 putaran, dua *word* pertama digunakan sebagai masukan kepada fungsi F, yang juga mengambil angka bulat itu sebagai masukan. *Word* yang ketiga di-XOR dengan keluaran pertama F dan kemudian diputar ke kanan satu bit. *Word* keempat diputar ke kiri satu bit kemudian di-XOR dengan *word* keluaran F Yang kedua . Akhirnya, keduanya saling ditukar menghasilkan persamaan :

$$
\begin{aligned}
(F_{r,0},F_{r,1}) &= F(F_{r,0},F_{r,1},r)\\
R_{r+1,0} &= ROR(R_{r,2} \oplus F_{r,0},1)\\
R_{r+1,1} &= ROL(R_{r,3},1) \oplus F_{r,1}\\
R_{r+1,2} &= R_{r,0}\\
R_{r+1,3} &= R_{r,1}
\end{aligned}
$$

Untuk r=0,…,15 di mana *ROR* dan *ROL* adalah berfungsi memutar argumentasi pertama (32-bit kata) ke kanan dengan angka bit-bit diindikasikan dengan argumentasi keduanya. Langkah whitening keluaran membatalkan pertukaran putaran terakhir dan meng XOR *double word* dengan 4 *word* dari kunci yang diperluas.

$$C_i=R_{16,(i+2) \bmod 4} \oplus K_{i+4} \quad I = 0,\ldots,3$$

Empat *word* dari *ciphertext* kemudian menulis seperti 16 byte $c_{o,\ldots,}c_{15}$ sama seperti menggunakan konversi *little-endian* untuk *plaintext*.

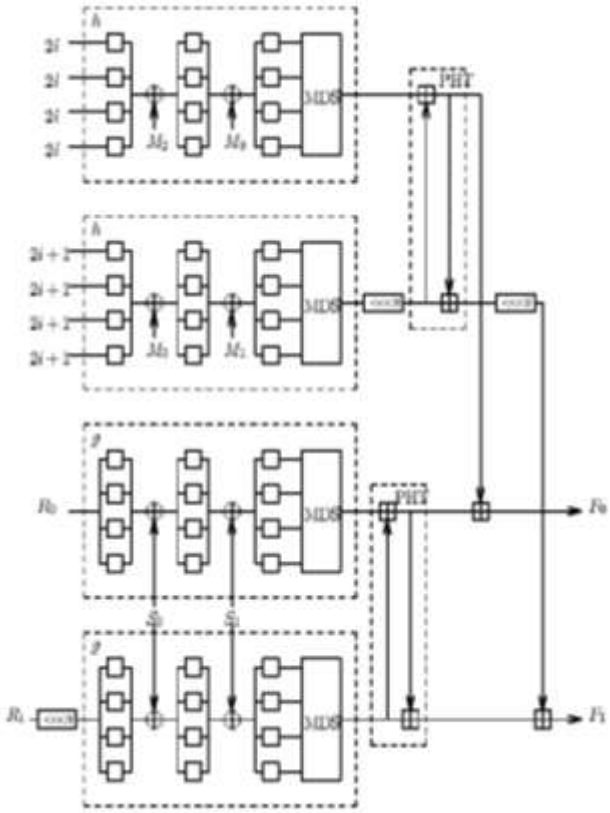
$$c_i=\left\lfloor \frac{C\lfloor i/4 \rfloor}{2^{8(i \bmod 4)}} \right\rfloor \bmod 2^8 \quad i = 0,\ldots,15$$

Fungsi F

Fungsi F adalah suatu permutasi *key-dependent* di atas nilai 64-bit. Untuk mengambil tiga argumentasi, dua kata masukan R_0 Dan R_1, dan angka bulat *r* digunakan untuk memilih *subkey* yang sesuai. R_0 yang dilewati fungsi g, menghasilkan T_0. *i* diputar 8 bit ke kanan kemudian melewati fungsi g untuk menghasilkan T_1. Hasil T_0 dan T_1 selanjutnya dikombinasikan dalam sebuah *PHT* danditambahkan 2 kata dari kunci yang diperluas menghasilkan persamaan :

$$
\begin{aligned}
T_0 &= g(R_0)\\
T_1 &= g(ROL(R_1,8))\\
F_0 &= (T_0 + T_1 + K_{2r+8}) \bmod 2^{32}\\
F_1 &= (T_0 + 2T_1 + K_{2r+9}) \bmod 2^{32}
\end{aligned}
$$

Di mana (F_0, F_1) adalah hasil dari *F*. Kita juga menggambarkan fungsi *F'* untuk menganalisa. *F'* adalah identik dengan fungsi *F*, kalau tidak tambahkan beberapa blok kunci pada keluaran. (*PHT* masih dilakukan.)



Gambar 3: Satu putaran fungsi *F* (kunci 128-bit)

Fungsi *g*

Fungsi g membentuk jantungnya Twofish. Kata masukan *X* dipecah menjadi empat byte. Masing-masing byte dijalankan melewati *S-box key-dependent.* Masing-masing S-box adalah *bijective*, mengambil 8 bit masukan, dan menghasilkan 8 bit keluaran. Ke empat hasil diinterpretasikan sebagai vektor yang panjangnya 4 di atas $GF(2^8)$, dan dikalikan dengan yang matriks *MDS* 4x4 (menggunakan bidang $GF(2^8)$ untuk perhitungannya). Untuk menghasilkan vektor diinterpretasikan sebagai 32-bit kata sebagai hasil dari g :

$x_i = \lfloor X / 2^{8i} \rfloor \bmod 2^8 \qquad i = 0,\ldots,3$

$y_i = s_i[x_i] \qquad i = 0,\ldots,3$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} . \\ .MDS \\ . \end{pmatrix} . \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$Z = \sum_{i=0}^{3} z_i . 2^8$$

Di mana s_i adalah *S-Box key-dependent* dan *Z* adalah hasil dari *g*. Untuk merumuskan dengan baik, kita harus menetapkan koresponden antara nilai-nilai byte dan elemen-elemen bidang $GF(2^8)$. Kita merepresentasikan $GF(2^8)$ sebagai $GF(2)[x]/v(x)$ di mana $v(x)=x^8+x^6+x^5+x^3+1$ adalah suatu polynomial primitif dari 8 tingkat di atas GF(2). Unsur Bidang $a = \sum_{i=0}^{7} a_i x^i$ dengan $a_i \in GF(2)$ adalah dikenal dengan nilai byte $\sum_{i=0}^{7} a_i x^i$. Ini adalah beberapa pengertian pemetaan "alamiah"; penambahan di dalam $GF(2^8)$ berkorespondensi dengan suatu *XOR* dari bytes. Matriks *MDS*-nya adalah sebagai berikut :

$$\text{MDS} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix}$$

Di mana elemen-elemen ditulis sebagai nilai-nilai byte hexadecimal.

Jadwal Kunci

Jadwal kunci harus menyediakan 40 kata dari kunci yang diperluas $K_{0,\ldots,}K_{39}$, dan 4 buah *S-Box key-dependent* yang digunakan di dalam fungsi g. *Twofish* didefinisikan untuk kunci-kunci dengan panjang N= 128, N= 192, dan N= 256. Beberapa kunci yang lebih pendek dari 256 bit dapat digunakan oleh lapisannya dari nol hingga yang lebih besar yang didefinisikan sebagai panjang kunci. Kita mendefinisikan k=N/64. Kunci M terdiri dari 8k byte $m_0,\ldots,m_{8k-1}$. Byte-byte adalah yang pertama diubah ke dalam 2k kata dari 32 bit masing-masing :

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

dan kemudian ke dalam dua vektor kata dari panjang k :

$M_e = (M_0, M_2, \ldots, M_{2k-2})$
$M_o = (M_1, M_3, \ldots, M_{2k-2})$

Sepertiga vektor kata dari panjang *k* adalah juga diperoleh dari kunci itu.

Hal ini dilakukan dengan mengambil byte-byte kunci di dalam kelompok 8, menginterpretasikannya sebagai vektor di atas $GF(2^8)$, dan mengalikannya dengan matrils 4x8 yang diperoleh dari suatu kode *R*. Masing-masing hasil dari 4 byte kemudian diinterpretasikan sebagai suatu kata 32 bit.. Kata-kata ini menyusun vektor yang ketiga :

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

$$S_i = \sum_{j=0}^{3} S_{i,j}.2^{8j}$$

untuk i = 0,…,k-1, dan

$$S = (S_k\text{-}1, S_{k-2}, \ldots, S_0)$$

Catat bahwa daftar S kata-kata itu di dalam order "terbalik". Karena Untuk perkalian matriks RS, *GF* (2^8) diwakili oleh *GF*(2)[x]/w(x), di mana w(x)= $x^8+x^6+x^3+x^2+1$ adalah polynomial primitif derajat dari 8 tingkat di atas GF(2). Pemetaan antara nilai-nilai byte dan elemen-elemen *GF*(2^8) menggunakan defnisi yang sama sebagaimana yang digunakan untuk perkalian matriks *MDS*. Dalam pemetaan ini, matriks RS ditunjukkan sebagai berikut :

$$RS = \begin{pmatrix} 01 & A4 & 55 & 87\ 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F\,31E & C6 & 68 & E5 \\ 02 & A1 & FC & C1\ 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A\ 58 & DB & 9E & 03 \end{pmatrix}$$

Ke tiga vektor M_e, M_o, dan *S* ini membentuk basis dari jadwal kunci.

Penambahan Panjang Kunci

Twofish dapat menerima kunci-kunci dengan panjang byte di atas 256 bit. Untuk ukuran-ukuran kunci yang tidak didefinisikan di atas, kunci diisi pada bagian akhir dengan nol byte kepada yang lebih panjang berikutnya. Sebagai contoh, suatu kunci 80-bit $m_0,\ldots,m_9$ akan diperluas dengan mengatur m_i=0 untuk i= 10,…,15 dan diperlakukan sebagaimana pada kunci 128-bit.

Fungsi *h*

Fungsi h adalah suatu fungsi yang mengambil dua input-a 32-bit kata *X* dan sebuah daftar $L = (L_0,\ldots,L_{k-1})$ dari kata-kata *X* 32 dengan panjang k- dan menghasilkan satu kata pada outputnya. Ini adalah pekerjaan fungsi di dalam langkah-langkah k. Pada setiap langkah, empat byte itu masing-masing melintasi suatu *fixed S-Box*, dan di-*XOR* dengan sebuah byte yang diperoleh dari daftar. Akhirnya, byte-byte sekali lagi digeser melewati sebuah fixed S-box dan empat byte itu dikalikan dengan matriks *MDS* seperti halnya dalam g. Lebih formal kita memisah-misahkan kata-kata itu ke dalam bytes :

$$l_{i,j} = \lfloor L_i / 2^8 \rfloor \bmod 2^8$$
$$x_j = \lfloor X / 2^{8j} \rfloor \bmod 2^8$$

untuk i=0,…, k-1 dan j=0,…,3. Kemudian urutan subtitusi dan *XOR* diterapkan:

$y_{k,j}=x_j \qquad j = 0,\ldots,3$

jika k≥ 3 didapatkan :

$y_{3,0} = q_1[y_{4,0}] \oplus l_{3,0}$
$y_{3,1} = q_0[y_{4,1}] \oplus l_{3,1}$
$y_{3,2} = q_0[y_{4,2}] \oplus l_{3,2}$
$y_{3,3} = q_1[y_{4,3}] \oplus l_{3,3}$

jika k= 4 didapatkan :

$y_{2,0} = q_1[y_{3,0}] \oplus l_{3,0}$
$y_{2,1} = q_1[y_{3,1}] \oplus l_{3,1}$
$y_{2,2} = q_0[y_{3,2}] \oplus l_{3,2}$
$y_{2,3} = q_0[y_{3,3}] \oplus l_{3,3}$

Dalam seluruh kasus didapatkan :

$y_0 = q_1[q_0[q_0[y_{2,0}] \oplus l_{1,0}] \oplus l_{0,0}]$
$y_1 = q_0[q_0[q_1[y_{2,1}] \oplus l_{1,1}] \oplus l_{0,1}]$
$y_2 = q_0[q_1[q_0[y_{2,2}] \oplus l_{1,2}] \oplus l_{0,2}]$
$y_3 = q_1[q_1[q_1[y_{2,3}] \oplus l_{1,3}] \oplus l_{0,3}]$

Di sini, q_0 dan q_1 ditetapkan permutasi di atas nilai 8-bit yang akan didefinisikan

segera. Menghasilkan vektor yang merupakan perkalian matriks *MDS*, seperti halnya dalam fungsi *g* :

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} . \\ .MDS \\ . \end{pmatrix} . \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$Z = \sum_{i=0}^{3} z_i . 2^{8i}$$

di mana *Z* adalah hasil dari fungsi *h*

S-Box Key-dependent

Sekarang dapat didefinisikan *S-Box* dalam fungsi *g* oleh :

$$g(x) = h(X; S)$$

Hal itu adalah karena i= 0,…,3, *S-Box Key-Dependent* s_i dibentuk oleh pemetaan dari x_i ke y_i di dalam fungsi *h*, di mana daftar Lsama dengan vektor *S* yang diperoleh dari kunci itu.

Kata-kata Kunci yang Diperluas K_j

Kata-kata dari kunci yang diperluas didefinisikan menggunakan fungsi *h* :

$P = 2^{24} + 2^{16} + 2^8 + 2^0$
$A_i = h(2_{i\rho}, M_e)$
$B_i = ROL(h((2_i + 1)\ \rho, M_o), 8)$
$K_{2i} = (A_i + B_i) \bmod 2^{32}$
$K_{2i+1} = ROL((A_i + 2B_i) \bmod 2^{32}, 9)$

Konstanta ρ digunakan untuk menduplikat byte yang mempunyai properti untuk i= 0,…,255, kata itu jika terdiri dari empat byte yang sama, masing-masing dengan nilai *i*. Fungsi h diberlakukan bagi kata-kata jenis ini. Untuk A_i nilai-nilai byte itu adalah 2i, dan argumentasi yang kedua dari h adalah M_e. B_i dihitung dengan cara yang sama menggunakan 2i+1 sebagai byte nilai dan M_o sebagai argumentasi yang kedua, dengan suatu putaran ekstra di atas 8 bit. Nilai-nilai A_i dan B_i Dua dikombinasikan dalam *PHT*. Salah satu dari hasil itu selanjutnya diputar dengan 9 bit. Kedua hasil tersebut membentuk dua kata kunci yang diperluas.

Permutasi q_0 dan q_1

Permutasi q_0 dan q_1 adalah ditetapkan permutasi di atas nilai-nilai 8 bit. Mereka dibangun dari empat 4-bit permutasi yang masing-masing berbeda. Untuk nilai masukan x didefinisikan sebagai nilai output y sebagai berikut :

$a_0, b_0 = \lfloor x/16 \rfloor$, x mod 16
$a = a_0 \oplus b_0$
$b_1 = a_0 \oplus ROR_1(b_0,1) \oplus 8a_0 \bmod 16$
$a_2, b_2 = t_0[a_1], t_1[b_1]$
$a_3 = a_2 \oplus b_2$
$b_3 = a_2 \oplus ROR_1(b_2,1) \oplus 8a_2 \bmod 16$
$a_4, b_4 = t_2[a_3], t_3[b_3]$
$y = 16b_4 + a_4$

di mana ROR_4 adalah suatu fungsi yang serupa dengan *ROR* yang merupakan putaran nilai 4-bit. Pertama, byte dipecah menjadi dua bagian. Ini dikombinasikan dalam suatu bijective yang mencampur langkah. Masing-masing bagian kemudian melintasi *4-bitfixed S-Box*. Ini diikuti oleh yang lain. Akhirnya, dua bagian dikombinasikan kembali ke dalam satu byte. Untuk permutasi q_0 *S-Box* 4-bit :

t_0 = [8 1 7 D 6 F 3 2 0 B 5 9 E C A 4]
t_1 = [E C B 8 1 2 3 5 F 4 A 6 7 0 9 D]
t_2 = [B A 5 E 6 D 9 0 C 8 F 3 2 4 7 1]
t_3 = [D 7 F 4 1 2 6 E 9 B 3 0 8 5 C A]

di mana masing-masing *S-Box* 4-Bit diwakili oleh daftar masukan yang menggunakan notasi hexadecimal. (untuk masukan 0,…,15 didaftarkan dalam order.) Dengan cara yang sama, untuk q_1 *S-Box* 4-bit :

t_0 = [2 8 B D F 7 6 E 3 1 9 4 0 A C 5]
t_1 = [1 E 2 B 4 C 3 7 6 D A 5 F 9 0 8]
t_2 = [4 C 7 5 1 6 9 A 0 E D 8 2 B 3 F]
t_3 = [B 9 5 1 C 3 D E 6 4 7 F 2 0 8 A]

PERANCANGAN SISTEM

1.4 Tahapan Proses Enkripsi dan Dekripsi

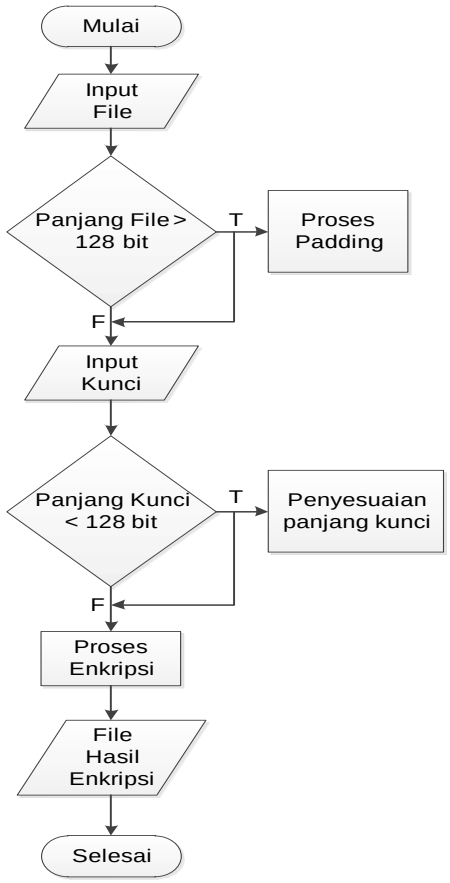
Tujuan pembuatan sistem ini adalah menerapkan algoritma twofish untuk mengamankan file sehingga file menjadi tidak dapat terbaca. Proses utama pada pembuatan aplikasi perangkat lunak ini adalah melakukan enkripsi dan dekripsi.

1. Proses Enkripsi

Langkah-langkah dalam melakukan proses Enkripsi File :

a. Memulai proses enkripsi.
b. *Input* file yang akan dienkripsi.
c. Jika *Plaintext* berukuran 128 bit maka tidak akan terjadi proses *Padding*. Jika *Plaintext* berukuran lebih dari 128 bit, maka proses *Padding* akan dilakukan.
d. *Input* kunci untuk mengenkripsi.
e. Jika panjang kunci yang di *input* kurang dari 128 bit maka akan dilakukan penyesuaian panjang kunci untuk mencapai 128 bit.
f. Proses enkripsi.
g. Menampilkan hasil enkripsi.
h. Selesai.

Digram alur proses enkripsi file terlihat pada gambar 4.



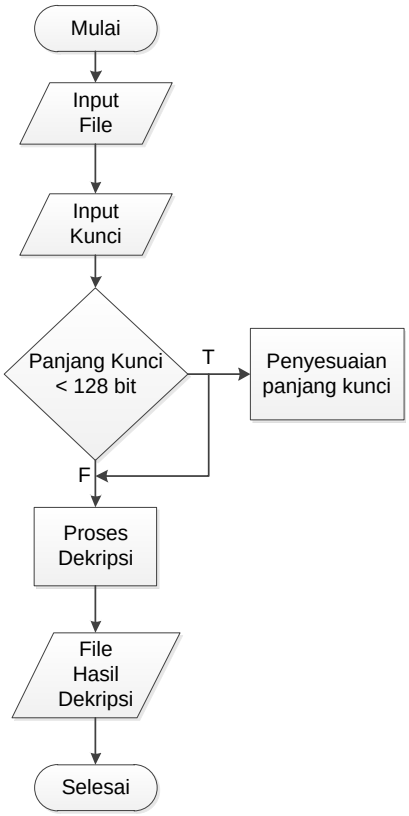
Gambar 4. *Flowchart* Enkripsi File

2. Proses Dekripsi

Langkah-langkah dalam melakukan proses Dekripsi File :

a. Memulai proses dekripsi.
b. *Input* file yang akan di dekripsi.
c. Input kunci untuk mendekripsi file.
d. Jika panjang kunci yang akan di *input* kurang dari 128 bit maka akan dilakukan penyesuaian panjang kunci untuk mencapai panjang 128 bit.
e. Proses dekripsi.
f. Menghapus padding.
g. Menampilkan hasil dekripsi.
h. Selesai.

Digram alur proses dekripsi file terlihat pada gambar 4.



Gambar 5. *Flowchart* Dekripsi File

1.5 Perancangan Proses

Proses yang dirancang pada perangkat lunak ini berdasarkan pada algoritma Twofish. Algoritma Twofish sendiri merupakan algoritma yang menggunakan kunci simetris. Secara lebih jelas tahapan-tahapan algoritma twofish akan dijabarkan sebagai berikut.

Penjadwalan Kunci :

a. Sebelum melalui tahapan enkripsi, maka harus melalui penjadwalan kunci. Panjang kunci yang didefiniksikan twofish sepanjang 128 bit, 192 bit dan 256 bit. Apabila input kunci yang dimasukkan kurang dari ketentuan tersebut, maka akan ditambahkan *zero byte* sampai panjang kunci memenuhi ketentuan.
b. Setelah itu kunci dibagi menjadi vector Me, Mo dan S. Vector Me dan Mo digunakan pada fungsi h sebagai *list*, sedangkan vector S digunakan untuk tahap enkripsi pada fungsi g.
c. Masukkan masing-masing *word* kunci K_j yang diekspansi yaitu 2i dan 2i+1 ke dalam fungsi h yaitu melalui pemutasi $_{q0}$ dan $_{q1}$ dilanjutkan dengan matrik MDS.
d. Hasil dari *word* 2i melalui proses PHT, sedangkan *word* 2i+1 sebelum melalui proses PHT dilakukan rotasi ke kiri sejauh 8 bit. Maka hasil dari proses tersebut menjadi kunci yang sudah terjadwal.

Tahap Enkripsi :

a. Input *Plaintext* sebesar 128 bit akan dibagi menjadi empat *word* yaitu P0, P1, P2, P3 yang masing-masing sebesar 32 bit. P0 dan P1 akan menjadi bagain kiri, sedangkan P2 dan P3 akan menjadi bagian kanan.
b. *Plaintext* akan melalui proses *input whitening* yaitu *input* akan di-XOR dengan empat *word* kunci yang telah terjadwal yaitu K0, K1, K2 dan K3.
c. Proses berikutnya input akan melalui proses pada fungsi F yang meliputi di dalamnya adalah fungsi g dan dilanjutkan dengan PHT, dan dilakukan penambahan hasil PHT dengan kunci. Proses fungsi F tersebut dilakukan secara bertahap . R0 dan R1 yang merupakan hasil whitening akan menjadi input untuk fungsi F.
d. R0 dan R1 akan dimasukkan ke dalam fungsi g yang merupakan bagian awal dari fungsi F. Untuk R1 sebelum dimasukkan ke dalam fungsi g akan dirotasi ke kiri sejauh 8 bit. R0dan R1 melalui S-Box dan selanjutnya akan dikalikan dengan menulis matriks MDS. Hasil dari fungsi g ini masing-masing menjadi T0 dan T1.
e. T0 dan T1 akan melalui proses PHT yang merupakan penggabungan T0 dan T1 dimana T0+T1 dan T0+2T1. Setelah itu hasil dari PHT tersebut masing-masing akan ditambahkan dengan kunci yang sudah terjadwal yaitu K2r+8 dan K2r+9. Hasil dari fungsi F adalah F0 dan F1, maka dengan demikian fungsi F telah terpenuhi.
f. Setelah itu F0 dan F1 masing-masing di-XOR dengan R2 dan R3. Hasil dari

R2 XOR F0 dirotasi ke kanan sejauh 1 bit. Sedangkan R3 XOR F1, sebelumnya R3 dirotasi ke kiri sejauh 1 bit.

e. Setelah itu, maka akan dilakukan iterasi sebanyak 16 kali. Setiap iterasi sama dengan proses sebelumnya.
f. Hasil dari swap blok terakhir adalah penukaran bagian kanan dan kiri yang di-*undo.*
g. Hasil dari 16 *round* enkripsikan melalui *output whitening* yaitu proses peng-XORan 16 *round* enkripsi dengan K4, K5, K6 dan K7.

Tahap Dekripsi :

Pada proses dekripsi, cara yang dilalui sama dengan proses enkripsi tetapi hanya arahnya saja yang berlawanan. Proses yang dilalui secara berurutan yaitu : *Output Whitening*, Swap Blok Terakhir, 16 Iterasi Dekripsi, dan *Input Whitening*. Inputnya adalah *chipertext* dan kunci untuk memperoleh *plaintext*. Kunci untuk dekripsi sama saja dengan kunci enkripsi, begitupun juga panjang maksimal kunci yaitu 256 atau 32 karakter.

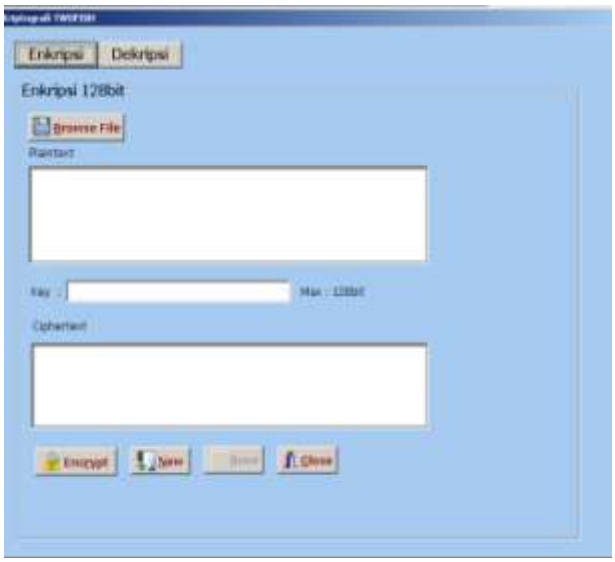
IMPLEMENTASI

Implementasi dari perangkat lunak merupakan tahap akhir dari proses perancangan. Agar proses implementasi dapat bekerja dengan baik, maka terlebih dahulu dilakukan pengujian untuk mengetahui kesalahan yang ada kemudian dievaluasi.

Jika program dijalankan maka tampilan yang muncul adalah tampilan *Form* Utama seperti pada Gambar 6.

Gambar 6. *Form* Menu Utama

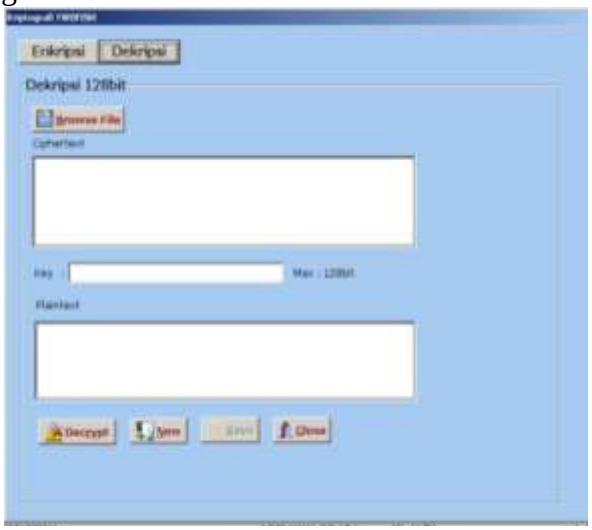
Pada *form* menu utama yang merupakan tampilan awal ini terdapat pilihan Encryption/ Decryption serta pilihan *Exit*. Jika memili Encryption/ Decryption maka akan tampil *form* Enkripsi dan Dekripsi. Untuk keluar dari menu utama maka dipilih tombol *Exit* yang merupakan pilihan untuk keluar secara keseluruhan dari Aplikasi. Tampilan *form* Enkripsi dapat dilihat pada gambar 7.



Gambar 7. *Form* Enkripsi

Pada *form* tersebut, *plaintext* yang akan dienkripsi di *browse* dari komputer melalui tombol *Browse File*, kemudian pilih format *file* yang akan dienkripsi, dalam hai ini hanya file Dokumen (*.doc). Setelah

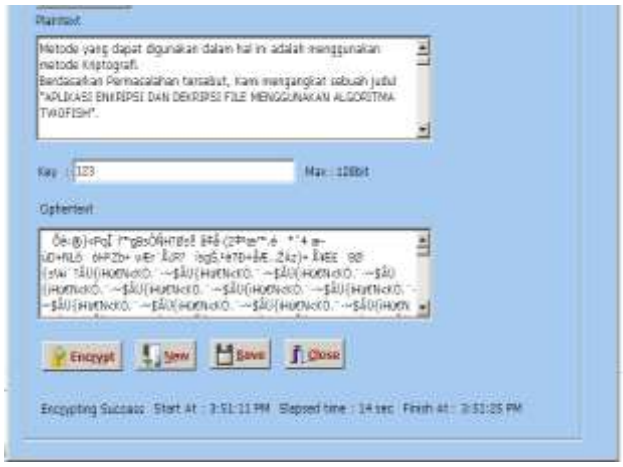
melakukan pemilihan *file*, *input* kunci yang akan digunakan pada proses. Klik tombol *Encrypt* untuk melakukan proses enkripsi. Tampilan *form* Dekripsi dapat dilihat pada gambar 8.



Gambar 8. *Form* Dekripsi

Tampilan tersebut ditampilkan ketika Pengguna hendak melakukan proses Dekripsi. Proses dekripsi hanya bisa dilakukan jika kunci yang di input oleh user bernilai Benar. File yang akan didekripsi di browse dari komputer dan untuk melakukan proses dekripsi haruslah memasukkan kunci yang sama pada saat melakukan proses enkripsi. Klik tombol *Decrypt* untuk melakukan proses dekripsi.

Saat proses Enkripsi dijalankan akan terlihat di sebelah pojok kiri bawah terdapat label "*Encrypt*" dengan nilai persen disebelahnya yang memiliki maksud untuk memberitahu sampai seberapa jauh proses enkripsi telah dilaksanakan. Keterangan "*Success*" berarti proses enkripsi telah berhasil dijalankan. Terlihat pada gambar 9.



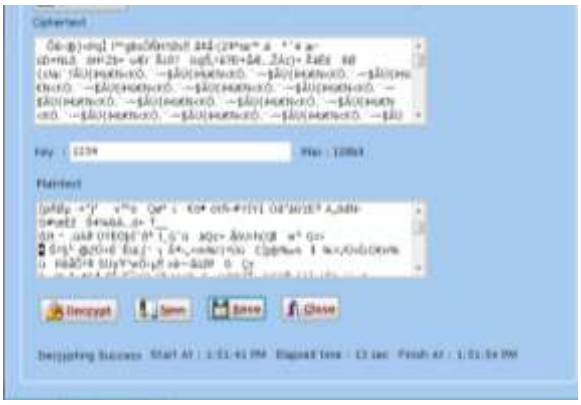
Gambar 9. Proses Enkripsi

Proses ini juga terjadi saat proses dekripsi akan tetapi label yang tertulis adalah "*Decrypt*". Terlihat pada gambar 10. Sama seperti proses enkripsi setelah proses dekripsi dijalankan hasil dekripsi akan diletakkan pada *textbox* bagian bawah yang berjudul "*Plaintext*", dengan waktu mulai tercatat dibawahnya dengan label yang berjudul "*Start At*" dan waktu selesai tercatat dengan label "*Finish At*" dan terdapat label yang berjudul "*Elapsed time*" yang menandakan seberapa lama proses dekripsi dilakukan. Jika file didekripsi menggunakan kunci berbeda dengan yang digunakan pada saat proses enkripsi, maka akan menampilkan hasil yang tidak sesuai dengan file asli. File hasil dekripsi tidak sama dengan file asli. File masih terdeteksi sebagai file rusak.

Gambar 10. Proses Dekripsi

Jika file didekripsi menggunakan kunci

berbeda dengan yang digunakan pada saat proses enkripsi, maka akan menampilkan hasil yang tidak sesuai dengan file asli seperti gambar 11.



Gambar 11. Proses Dekripsi Dengan Kunci Berbeda

File hasil dekripsi tidak sama dengan file asli. File masih terdeteksi sebagai file rusak.

KESIMPULAN

Dari hasil perancangan, pembahasan dan pengujian aplikasi, data dapat dibuat menjadi data yang tidak dapat terbaca dengan menggunakan algoritma twofish, sehingga kriptosistem yang dirancang dapat memenuhi prinsip kriptografi. Dengan demikian data dapat lebih terjaga keamanaannya. Sekalipun terjadi pencurian data, setidaknya pihak yang tidak berhak tersebut tidak dapat mengetahui isi dari file.

DAFTAR PUSTAKA

Ariyus, Dony. 2008. Pengantar Ilmu Kriptografi Teori, Analisis, dan Implementasi. Yogyakarta : Andi.

J. Alam, M. Agus. 2003. Belajar Sendiri Mengolah Database Dengan Borload Delphi 7. Jakarta : PT Elex Media Komputindo.

Kurniawan J., Ir. , M.T., Kriptografi, Keamanan Internet dan Jaringan Komunikasi, Penerbit Informatika Bandung, April 2004.

Mudeng, Denny. *Kriptografi Twofish*, (Online), (budi.insan.co.id/courses/el7010/dikmenjur-2004/denny-report.doc)

Schneier B., Applied Crytography, Second Edition, John Wiley & Sons, Inc., 1996. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., dan Ferguson, N.(1998). *Twofish: A 128-Bit Block Cipher*, (online) (http://www.schneier.com/paper-twofish- paper.pdf)

Wahana Komputer. The Best Encryption Tools. Jakarta : PT Elex Media Komputindo. 2004.