RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES

SKRIPSI

Disusun oleh :

ARIYO YOUDO UTOMO

NIM : 06.12.656

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER & INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2012

HEARING

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Komputer dan Informatika Strata Satu (S-1)

Disusun Oleh :

ARIYO YOUDO UTOMO
NIM : 06.12.656

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda,MT
NIP.Y.1018800189

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Irmalia Suryani Faradisa,ST.MT	**Sotyohadi, ST.**
NIP.P. 103 0000 365	**NIP.Y.103 9700 309**

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2012

SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini :

Nama : ARIYO YOUDO UTOMO

NIM : 06.12.656

Program Studi : Tehnik Elektro

Konsentrasi : Tehnik Komputer Dan Informatika S1

Dengan ini menyatakan bahwa Skripsi yang saya buat adalah hasil karya sendiri, tidak merupakan plagiasi dari karya orang lain. Dalam Skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sumbernya sesuai dengan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat, dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsinya.

Malang, 21–02–2012

Yang membuat Pernyataan,

ARIYO YOUDO UTOMO

NIM : 06.12.656

Rancang Bangun Aplikasi Enkripsi Video 3GP Menggunakan VEA dan DES

Ariyo Youdo Utomo
(06.12.656)

Konsentrasi Komputer dan Informatika, Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Nasional Malang
Jln. Raya Karanglo Km 2 Malang
Email: riothelonely666@yahoo.com

ABSTRAK

Makalah ini membahas tentang rancangan dan implementasi model enkripsi pada video 3GP dengan tujuan keamanan pada penyimpanan file video. Proses enkripsi pada video akan menghasilkan video dengan gambar yang tidak bisa di buka sama sekali. Sebaliknya, proses dekripsi akan mengembalikan video yang tidak dapat di baca tersebut kembali menjadi video aslinya. Kunci yang digunakan pada proses enkripsi dan dekripsi ini harus sama, jika tidak proses dekripsi tidak akan mengembalikan video yang aslinya.

Pada model enkripsi ini dipilih algoritma VEA (VideoEncryption Algorithm) untuk diimplementasikan pada enkripsi video 3GP. Algoritma VEA umum digunakan karena dinilai ringan dan cocok untuk diterapkan pada video yang pada umunya berukuran besar. Untuk meningkatkan keamanan proses enkripsi, algoritma VEA yang ada dimodifikasi dengan menambahkan algoritma kriptografi DES dengan mode operasi CBC. Hal ini mengakibatkan operasi yang dilakukan bukan lagi bit per bit, melainkan per blok-blok slice dari gambar video. Blok-blok ini selanjutnya akan dienkripsi dengan menggunakan algoritma DES. Perangkat lunak ini dibangun dengan menggunakan bahasa pemograman Java dan kakas NetBeans 6.0.

***Kata kunci** : enkripsi video, 3GP, Data Encryption Standart, algoritma enkripsi video*

ABSTRACT

This paper discusses the design and implementation of the encryption model with 3GP video file storage purposes on the security video. The encryption process will produce a video with a video image that can not be opened at all. In contrast, the decryption process will restore the video that can not be read back into the original video. The key used in encryption and decryption processes must be equal, if not the decryption process will not restore the original video.

In the model chosen encryption algorithm VEA (VideoEncryption Algorithm) for encryption is implemented in 3GP video. VEA algorithm commonly used because it is considered mild and suitable to be applied to video in general large. To increase the security of the encryption process, the existing VEA algorithm is modified by adding a cryptographic algorithm DES with CBC mode of operation. This is not the result of surgery performed again bit by bit, but a slice blocks of video images. These blocks will then be encrypted using the DES algorithm. The software is built using the Java programming language and Kakas NetBeans 6.0.

***Keywords**: Video Encryption,3GP, Data Encrytion Standart, Video Encryption Algorithm*

KATA PENGANTAR

Dengan mengucap syukur kehadirat Tuhan YME yang dengan segala Kasih dan Anugerah – Nya, telah memberikan kekuatan, kesabaran, bimbingan dan perlindungan sehingga penulis dapat menyelesaikan laporan skripsi dengan judul : **" RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES"**

Pembuatan skripsi ini disusun guna memenuhi syarat akhir kelulusan pendidikan jenjang Strata I di Institut Teknologi Nasional Malang. Dalam penyusunan skripsi ini penulis banyak mendapat bantuan baik moril maupun materiil, saran dan dorongan semangat dari berbagai pihak, untuk itu penulis mengucapkan terima kasih kepada :

1. Bapak Ir. Soeparno Djiwo, MT selaku rektor ITN Malang
2. Bapak Ir. Sidik Noertjahjono, MT selaku Dekan Fakultas Teknologi Industri.
3. Bapak Ir.Yusuf Ismail Nakhoda,MT selaku Ketua Jurusan Teknik Elektro S-1 ITN Malang.
4. Ibu Irmalia Suryani Faradisa,ST.MT selaku Dosem Pembimbing I.
5. Bapak Sotyohadi, ST.MT selaku Dosem Pembimbing II.
6. Bapak Alm.Hasan dan Ibu Siani selaku orang tua penulis yang selalu memberi motivasi serta do'a yang mereka panjatkan untuk penulis sehingga dapat menyelesaikan skripsi.
7. Suci Nurlina Sari Nasution dan teman – teman Garda12 yang selalu membantu dan memberikan motivasi bagi penulis untuk menyelesaikan skripsi ini.
8. Semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini yang tidak bisa penulis sebutkan satu persatu.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat diharapkan.

Akhir kata, penulis mohon maaf kepada semua pihak bilamana selama penyusunan skripsi ini penyusun membuat kesalahan secara tidak sengaja dan semoga skripsi ini dapat bermanfaat bagi kita semua.

Malang, Januari 2012

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring berkembangnya era globalisasi, keamanan data multimedia sangat penting dalam bisnis komersil maupun tradisional saat ini. Contohnya, pada aplikasi *Video On Demand*, hanya orang yang membayar yang dapat menonton video tersebut. Selain itu juga pada aplikasi *Video Conferencing*, hanya orang yang berkepentingan saja yang dapat ikut serta dalam konferensi tersebut untuk mendapatkan datanya. Begitu pun dengan enkripsi video *3gp* ini, hanya yang mengetahui kunci saja yang dapat mendekripsi *file* video *3gp* tersebut. Untuk itu diperlukan tehnik enkripsi yang dapat menjaga keamanan data multimedia tersebut sehungga tidak dapat di putar oleh pemutar apapun juga.

Faktor penting yang harus diperhatikan dalam enkripsi video adalah efisiensi dan tingkat keamanan. Tidak seperti *plainteks*, *enkripsi* data *multimedia* memiliki ukuran data yang sangat besar. Sedangkan faktor keamanan yang di maksud ada dua, yaitu keamanan dari sisi algoritma serta keamanan dari sisi gambar video hasil enkripsi. Untuk itu di perlukan suatu algoritma enkripsi video yang dapat mengatasi hal-hal tersebut. Terdapat beberapa algoritma enkripsi video yang telah di bangun hingga saat ini, salah satunya adalah *Video Encryption Algorithm,* atau sering disebut dengan *VEA.*

Makalah ini membahas tentang rancangan dan implementasi model enkripsi pada video *3gp* dengan tujuan keamanan pada penyimpanan file video. Proses enkripsi pada video akan menghasilkan video yang tidak dapat di putar. Sebaliknya, proses dekripsi akan mengembalikan video tersebut kembali menjadi video aslinya. Kunci yang digunakan pada proses enkripsi dan dekripsi ini harus sama, jika tidak proses dekripsi tidak akan mengembalikan video aslinya.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka permasalahan yang diangkat pada Skripsi ini adalah :

1. Bagaimana proses pengolahan dan pengenalan pola pada video yang akan di enkripsi.
2. Bagaimana kualitas dari video yang telah di enkripsi serta keamanan dari video yang telah di enkripsi.

1.3 Tujuan Penelitian

Tujuan dari pelaksanaan skripsi ini adalah sebagai berikut :

1. Memahami cara kerja algoritma *Video Encryption.*
2. Memahami dan mencari solusi untuk menangani masalah enkripsi video, seperti keamanan video setelah di enkripsi .
3. Membangun aplikasi yang dapat mengenkripsi video.

1.4 Batasan Masalah

Berdasarkan rumusan masalah yang telah diuraikan sebelumnya, maka pada pelaksanaan skripsi ini batasan masalah adalah sebagai berikut:

1. *File* video yang di enkripsi dan hanya *file* video *3GP* dengan kapasitas *1Gb.*
2. Data yang di enkripsi adalah *segment bit* dari *3GP* video.

1.5 Metodologi

1. Studi literatur

   Dengan cara mempelajari dari berbagai macam sumber referensi maupun texbook mengenai *video-encryption* (enkripsi video) serta metode yang

akan dipakai dalam konversi video. Sumber pustaka dapat berupa buku ataupun website.

2. Analisa kebutuhan sistem

   Menganalisa masalah dan mencari solusi pada masalah yang dihadapi dalam enkripsi dan dekripsi video.

3. Perancangan dan Implementasi Perangkat Lunak

   Dalam tahap ini perancangan perangkat lunak meliputi perancangan dan antarmuka dari perangkat lunak yang akan dibangun.

4. Pengujian

   Pada tahapan pengujian untuk menganalisa performansi dari sistem enkripsi video .

1.6. Sistematika Pembahasan

Penyusunan laporan skripsi ini menggunakan kerangka pembahasan yang terbentuk dalam susunan bab, yang dapat dijelaskan sebagai berikut :

Bab I : Pendahuluan

Bab ini merupakan dasar penyusunan laporan skripsi yang di dalamnya berisi tentang latar belakang masalah, rumusan masalah, tujuan skripsi, batasan masalah, metodologi pengembangan sistem, dan sistematika pembahasan skripsi.

Bab II : Dasar Teori

Bab ini berisi tentang permasalahan yang berhubungan dengan penelitian yang dilakukan pada skripsi ini.

Bab III: Analisis

Bab ini berisi tentang analisis terhadap keamanan video yang telah di enkripsi dan di dekprisi. Perangkat lunak ini merupakan aplikasi yang berbasis desktop. Data yang digunakan dalam sistem secara umum dibagi menjadi dua, yaitu data untuk proses enkripsi dan data untuk proses dekripsi.

Bab IV: Analisis dan Perancangan

Bab ini berisi tentang tahap analisis yaitu identifikasi dan analisis masalah dan analisis kebutuhan sistem untuk menyelesaikan masalah yang dihadapi berdasarkan teori pada Bab II dan Bab III. Bab ini juga berisi hasil perancangan yaitu proses kelanjutan dari tahap analisis meliputi proses akusisi pengetahuan.

Bab V : Implementasi dan Pengujian

Bab ini berisi tentang implementasi hasil perancangan pada Bab IV dan penyesuaian kebutuhan sistem serta meliputi hasil pengujian sistem.

Bab VI : Penutup

Bab ini berisi tentang kesimpulan dan saran dari hasil penyusunan laporan skripsi yang telah disusun.

BAB II

DASAR TEORI

Pada bab II laporan tugas akhir ini di paparkan tentang dasar teori yang mendukung pelaksanaan tugas akhir yaitu penjelasan singkat tentang *kriptografi*, *stream chiper*, *block chiper*, *DES*, *Video Codec Standart*, *Video*, dan Algoritma Enkripsi Video. *Video Codec Standart* berisi penjelasan beberapa format video yang di tangani di tugas akhir ini. Algoritma enkripsi video yang akan dibahas adalah *Video Encryption Algorithm*. Landasan teori ini akan memberikan pemahaman yang lebih detail mengenai topik – topik tersebut sehingga akan memmudahkan proses analisis penyelesaian maslah pada bab selanjutnya

2.1 Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikannya kedalam bentuk yang tidak dapat dimengerti lagi maknanya. Algoritma *kriptografi* berisi aturan – aturan untuk mengenkripsi dan mendekripsi pesan dengan menggunakan fungsi matematika. Misalkan P menyatakan *plainteks* dan C menyatakan chiperteks, maka fungsi enkripsi E memetakan P ke C.

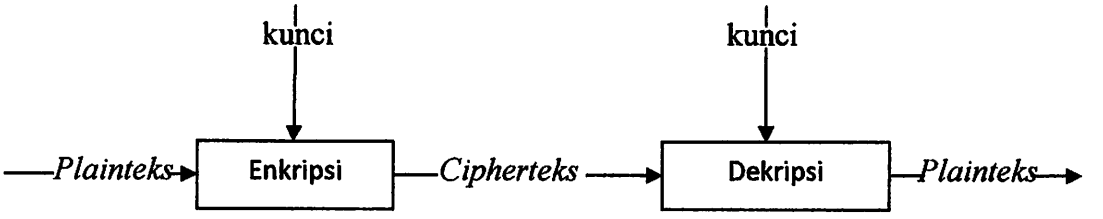
$$E(P) = C \tag{2.1}$$

Dan fungsi dekripsi D memetakan C ke P,

$$D(C) = P \tag{2.2}$$

Karena proses enkripsi kemudian dekripsi mengembalikan pesan ke pesan asal, maka kesamaan berikut harus benar,

$$D(E(P)) = P \tag{2.3}$$



Gambar 2.1 Enkripsi dan Dekripsi

Gambar II-2 merupakan ilustrasi dari proses enkripsi dan dekripsi suatu *plainteks*. Contoh dari enkripsi dan dekripsi dari pada data *text* adalah :

Plaintext : "Suci sedang belajar"

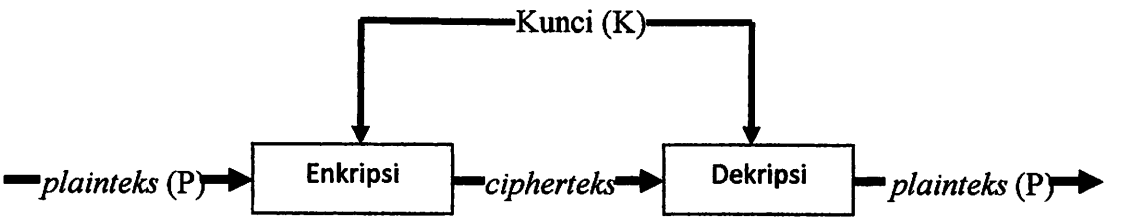
Chipertext : "mkja iswqro komidug"

Jika dikembalikan chipertext tersebut harus sama dengan *plaintext* yaitu "Suci sedang belajar"

Berdasarkan kunci yang digunakan untuk enkripsi dan dekripsi kriptografi dibedakan menjadi dua yaitu kriptografi dengan kunci simetri dan kriptografi dengan kunci asimetri. Pada skripsi kali ini, algoritma yang digunakan adalah *Data Encryption Standart* (*DES*). Algoritma *DES* memiliki kunci yang simetri.Kunci adalah parameter yang di gunakan untuk transformasi *enchipering* dan *dechipering.*

2.1.1 Algoritma Simetri

Algoritma simetri adalah algoritma yang menggunakan kunci enkripsi yang sama dengan kunci dekripsinya. Algoritma simetri sering juga disebut sebagai algoritma kunci rahasia, algoritma tunggal, atau algoritma satu kunci, dan mengharuskan pengirim dan penerima menyetujui suatu kunci tertentu sebelum mereka dapat berkomunikasi dengan aman. Keamanan algoritma simetri tergantung pada kunci, membocorkan kunci berarti orang lain dapat mengenkrip dan mendekrip pesan. Agar komunikasi tetap aman, kunci harus tetap di rahasiakan. Dua kategori algoritma yang termasuk pada algoritma simetri ini adalah algoritma *block chiper* dan *stream chiper. DES* merupakan salah satu contoh dari algoritma simetri *block chiper.* Ilustrasi dari algoritma kriptografi simetri dapat dilihat di gambar II-2.



Gambar 2.2 Algoritma kriptografi simetri

2.1.1.1 *Stream Cipher* (cipher aliran)

Cipher aliran beroperasi pada *plainteks/cipherteks* dalam bentuk bit tunggal, yaitu proses enkripsi dan dekripsi dilakukan bit per bit. *Chiper* akan mengenkripsi data bit per bit setiap kali transformasi, yang dalam hal ini rangkaian bit dienkripsikan/didekripsikan bit per bit. Cipher aliran mengenkripsikan *plainteks* menjadi *cipherteks* bit per bit (1 bit setiap kali transformasi).

$$c_i = p_i \oplus k_i \tag{2.4}$$

dan proses dekripsi menggunakan persamaan :

$$p_i = c_i \oplus k_i \tag{2.5}$$

Pada cipher aliran, bit hanya mempunyai dua buah nilai, sehingga proses enkripsi hanya menyebabkan dua keadaan pada bit tersebut : berubah atau tidak berubah. Dua keadaan tersebut ditentukan oleh kunci enkripsi yang disebut aliran-bit-kunci (*keystream*). Aliran-bit-kunci di-XOR-kan dengan aliran bit-bit *plainteks*, $p_1,p_2,...p_i$, untuk menghasilkan aliran bit-bit *cipherteks* :

$$c_i = p_i \oplus k_i \tag{2.6}$$

Disisi penerima, bit-bit *cipherteks* di-XOR-kan dengan aliran-bit-kunci yang sama untuk menghasilkan bit-bit *plainteks* :

$$p_i = c_i \oplus k_i \tag{2.7}$$

karena

$$c_i \oplus k_i = (p_i \oplus k_i) \oplus k_i = p_i \oplus (k_i \oplus k_i) = p_i \oplus 0 = p_i \tag{2.8}$$

Cipher aliran cocok untuk mengenkripsikan aliran data yang terus menerus melalui saluran komunikasi, misalnya :

1. Mengenkripsikan data pada saluran yang menghubungkan antara dua buah komputer.
2. Mengenkripsikan suara pada jaringan telepon *mobile* GSM.
3. Video *Streaming*.

Alasannya, jika bit *cipherteks* yang diterima mengandung kesalahan, maka hal ini hanya menghasilkan satu bit kesalahan pada waktu dekripsi, karena tiap bit *plainteks* ditentukan hanya oleh satu bit *cipherteks*. Kondisi ini tidak benar untuk *cipher* block karena jika satu bit *cipherteks* yang diterima mengandung kesalahan, maka kesalahan ini akan merambat pada seluruh blok bit *plainteks* hasil dekprisi (*error propagation*).

2.1.1.2 *Block Cipher*

Algoritma *block cipher* adalah algoritma yang masukan dan keluarannya berupa satu blok, dan setiap bloknya terdiri dari banyak bit (misalnya 1 blok terdiri dari 64 bit atau 128 bit). Enkripsi dilakukan terhadap blok bit *plainteks* menggunakan bit-bit kunci (yang ukurannya sama dengan ukuran blok *plainteks*). Algoritma enkripsi menghasilkan blok *cipherteks* yang berukuran sama dengan blok *plainteks*. Dekripsi dilakukan dengan cara yang sama dengan enkripsi.

Misalkan blok *plainteks* (P) yang berukuran m bit dinyatakan sebagai vektor

$$P = (p_1, p_2, \ldots, p_m)$$

Yang dalam hal ini p_i adalah 0 atau 1 untuk $i = 1,2,\ldots,m$ dan blok *cipherteks* (C) adalah

$$C = (c_1, c_2, \ldots, c_m)$$

Yang dalam hal ini c_i adalah 0 atau 1 untuk $i = 1, 2,\ldots,m$.

Bila *plainteks* dibagi menjadi n buah blok, barisan blok-blok *plainteks* dinyatakan sebagai

$$(P1, P2, \ldots, Pn)$$

Untuk setiap blok *plainteks* P_i, bit-bit penyusunnya dapat dinyatakan sebagai vektor

$$Pi = (p_{i1}, p_{i2}, \ldots, p_{im})$$

Enkripsi dan dekripsi dengan kunci K dinyatakan berturut-turut dengan persamaan

$$E_K(P) = C \qquad (2.9)$$

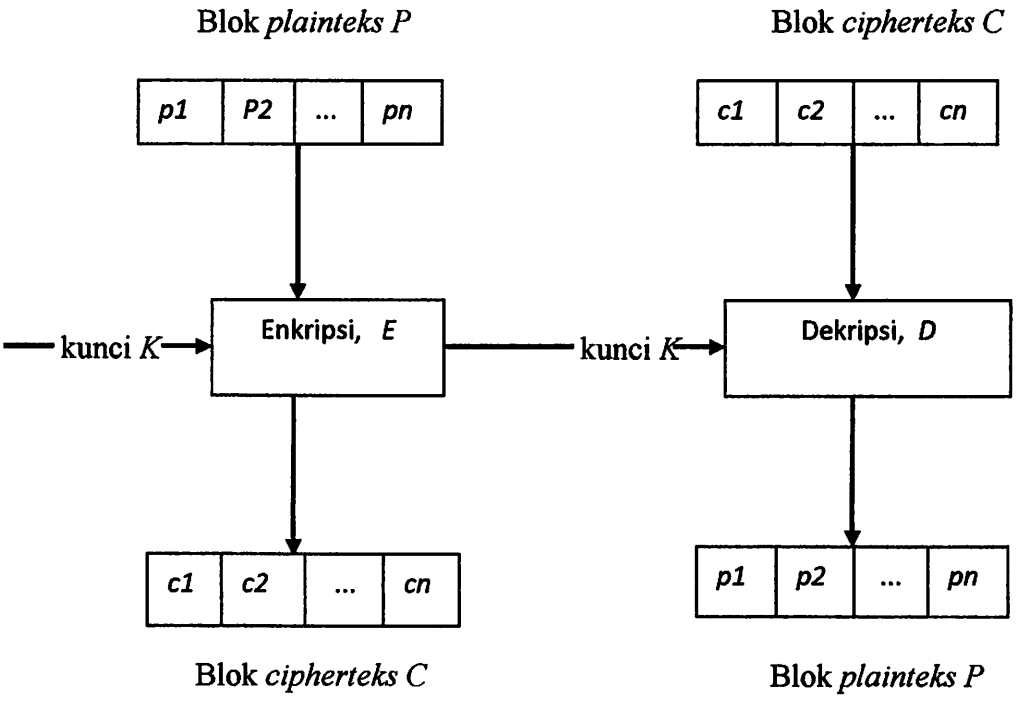
Untuk enkripsi, dan

$$D_K(C) = P \tag{2.10}$$

Fungsi E haruslah fungsi yang berkoresponden satu-ke-satu, sehingga

$$E^{-1} = D \tag{2.11}$$

Skema enkripsi dan dekripsi dengan *cipher block* di gambarkan pada gambar II-3. Fungsi E dan D dispesifikasikan oleh kriptografer.



Gambar 2.3 Skema enkripsi dan dekripsi pada cipherblok

2.1.1.3 Bitsream

Bitstream adalah data dalam suatu *file* baik yang berdampingan maupun yang tidak berdampingan (*contiguous* atau *non-contiguous*). Data ini memiliki ciri dan karakter (property) yang sama untuk kepentingan preservasi. Sebuah *bitstream* tidak dapat diubah menjadi *file* yang berdiri sendiri, kecuali kalau diubah strukturnya atau dilakukan format ulang (reformatting). Sebuah representation adalah serangkaian *files*, termasuk di dalamnya metadata struktural, yang diperlukan untuk interpretasi (rendition) sebuah Entitas Intelektual.

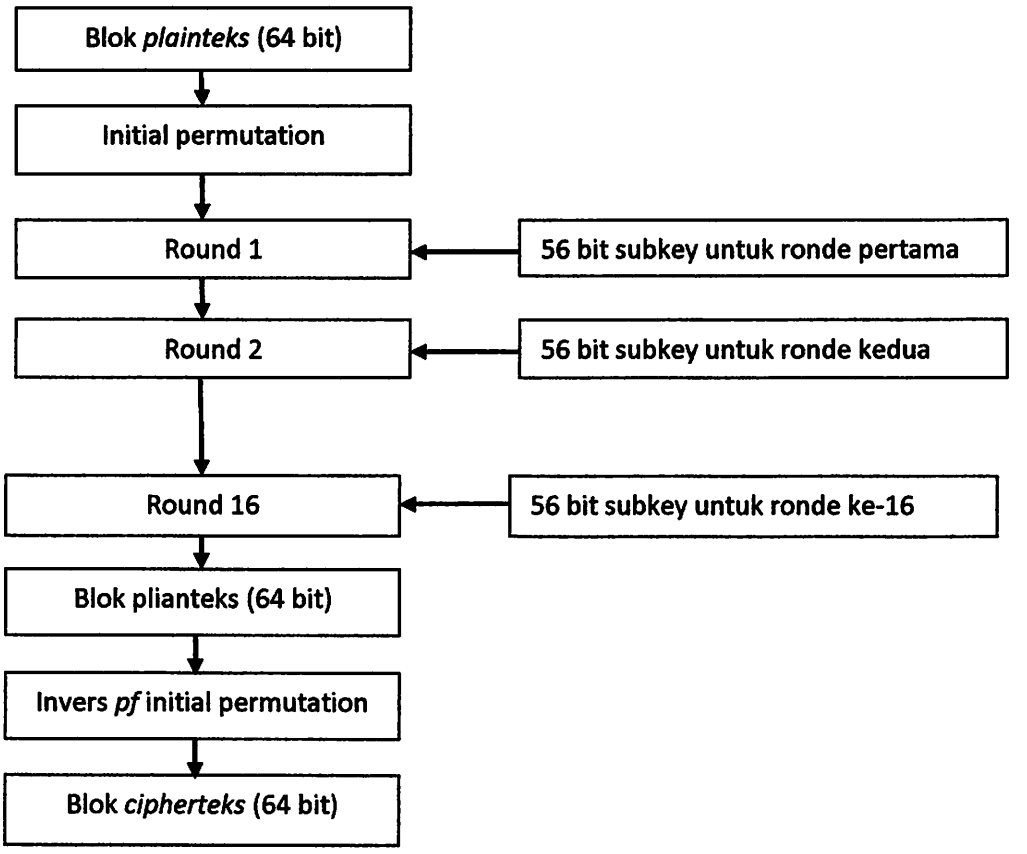
2.2 Data Encryption Standart (*DES*)

Data Encryption Standart adalah sebuah algoritma *chiper* yang termasuk dalam *block chiper* dan menggunakan kunci simetri.Algorima enkripsi yang paling banyak digunakan saat ini yaitu *Data Encryption Standart* (*DES*) yang di adopsi oleh *NIST* (*National Institut of Standart Technology*) sebagai standart pengolah informasi *Federal AS.* Data dienkrip dalam blok-blok 64 bit menggunakan kunci 56 bit. Dengan demikian, *DES* termasuk keluarga *block cipher.* Dengan tahapan dan kunci yang sama, *DES* digunakan untuk membalik enkripsi.

DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis *cipher* blok. *DES* beroperasi pada ukuran blok 64 bit. *DES* mengenkripsikan 64 bit *plainteks* menjadi 64 bit *cipherteks* dengan menggunakan 56 bit kunci internal (*internal key*) atau upa-kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) yang panjangnya 64 bit. Skema global dari algoritma *DES* adalah sebagai berikut :

1. Blok *plainteks* di permutasi dengan matriks permutasi awal (*initial permutation* atau IP).
2. Hasil permutasi awal kemudian diproses sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil *enciphering* kemudian di permutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP^{-1}) menjadi blok *cipherteks.*

Skema global lengkap dari *DES* dapat dilihat di gambar II-4.



Gambar 2.4 *DES* lengkap

2.3 Algoritma Enkripsi Video

Sama dengan enkripsi pada data teks, enkripsi video juga memiliki algoritma sendiri. Umumnya video dapat dienkripsi langsung dengan menggunakan algoritma enkripsi kunci rahasia yang telah banyak beredar saat ini seperti *DES*, *AES*, dan lain-lain, tetapi enkripsi seperti itu membutuhkan waktu yang cukup lama karena ukuran video yang cukup besar. Algoritma enkripsi video yang umum digunakan adalah *VEA*. Algoritma ini memiliki berbagai macam modifikasi yang disesuaikan berdasarkan kebutuhan, karena kemudahannya dalam implementasi. Salah satu tipe modifikasinya diberi nama *MVEA* (*Modified VEA*).

Tiga buah algoritma enkripsi video yang umum :

1. *VEA*. Memiliki berbagai modifikasi, karena kemudahannya dalam implementasi, salah satu tipe modifikasinya diberi nama *MVEA* (*Modified VEA*).

2. *MVEA*. Modifikasi dari algoritma *VEA*. Lebih sulit diimplementasikan dari *VEA* tetapi tingkat keamanan lebih tinggi dari *VEA*.

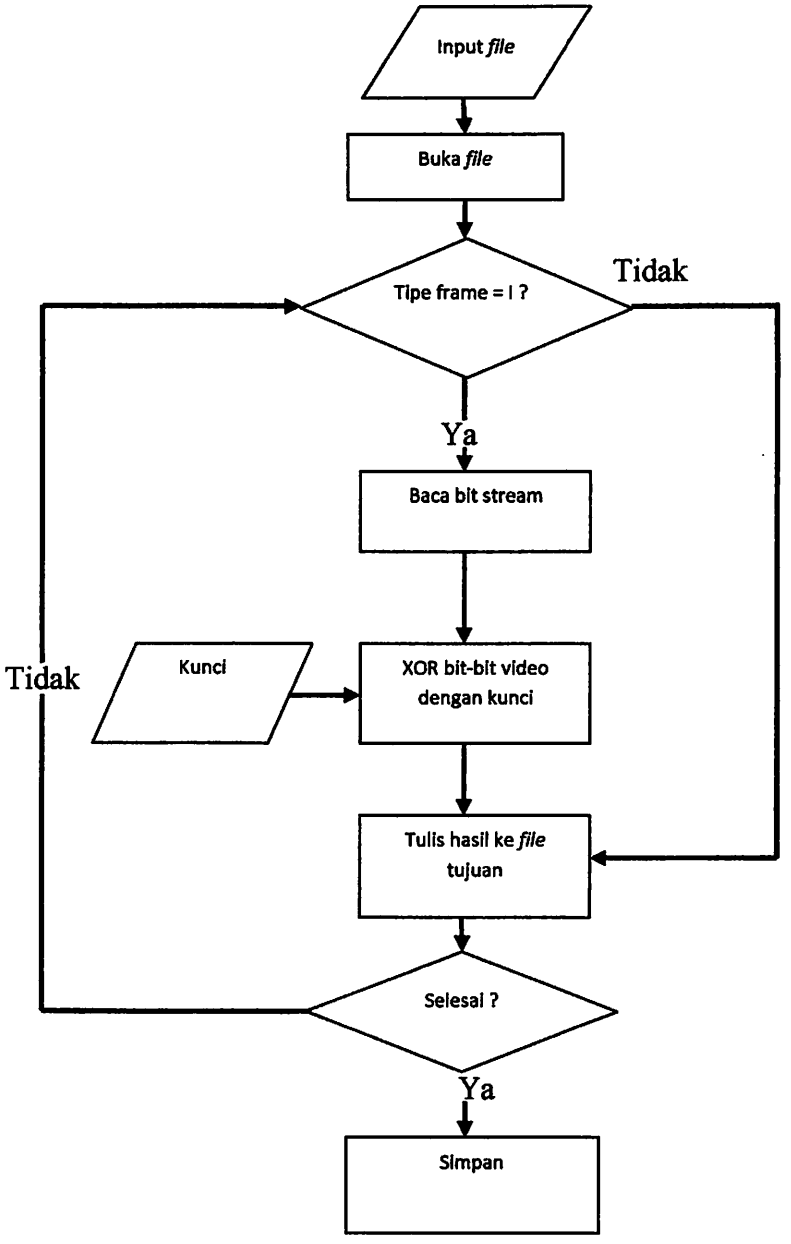
3. *RVEA*. Peningkatan dari *VEA*, tetapi implementasi dan komputasi yang cukup rumit, tetapi tingkat keamanan yang cukup tinggi.

2.3.1 Algoritma *VEA*

Berikut adalah skema global dari algoritma *VEA* :

1. Buka *file*.
2. Baca *frame file*, baca tipe *frame*-nya.
3. Baca *stream bit* dari *frame* tersebut.
4. Jika *frame* dari *stream bit* bukan *frame I*, maka *stream bit* langsung di tulis ke *file* tujuan.
5. Jika *stream bit* tersebut merupakan *stream bit* dari *frame I*, maka bit-bit tersebut di-XOR-kan dengan kunci.
6. Tulis hasil enkripsi ke *file* tujuan.
7. Baca *frame* selanjutnya, kembalilah ke langkah nomor dua sampai *End-of-file*.

Diagram alir dari algoritma *VEA* dapat dilihat di gambar II-5



Gambar 2.5 Diagram Alir Algoritma *VEA*

VEA (*Video Encrytion Algoritm*) merupakan sebuah algoritma enkripsi video yang berbasis pada *cipher* aliran (*stream* cipher). Kunci rahasia *VEA*, *k*, di-*generate* secara ramdom dalam bentuk *bitstream* dengan panjang *m*, yang dapat ditulis sebagai $k = b_1 b_2,...,b_m$. *Bitstream* dari video dapat direpresentasikan dengan :

$$S = ...s_1...s_2...s_m...s_{m+1}...s_{m+2}...s_{2m} \qquad (2.12)$$

Yang dalam hal ini $s_i (i = 1,2,....)$ adalah seluruh bit-bit dari video.

Fungsi dari enkripsi *VEA*, E_k dapat ditulis dengan :

$$E_k(S) = ...(b_1+s_1)...(b_m+s_{m+1})...(b_m+s_{2m}) \quad (2.13)$$

Yang dalam hal ini + adalah operasi XOR.

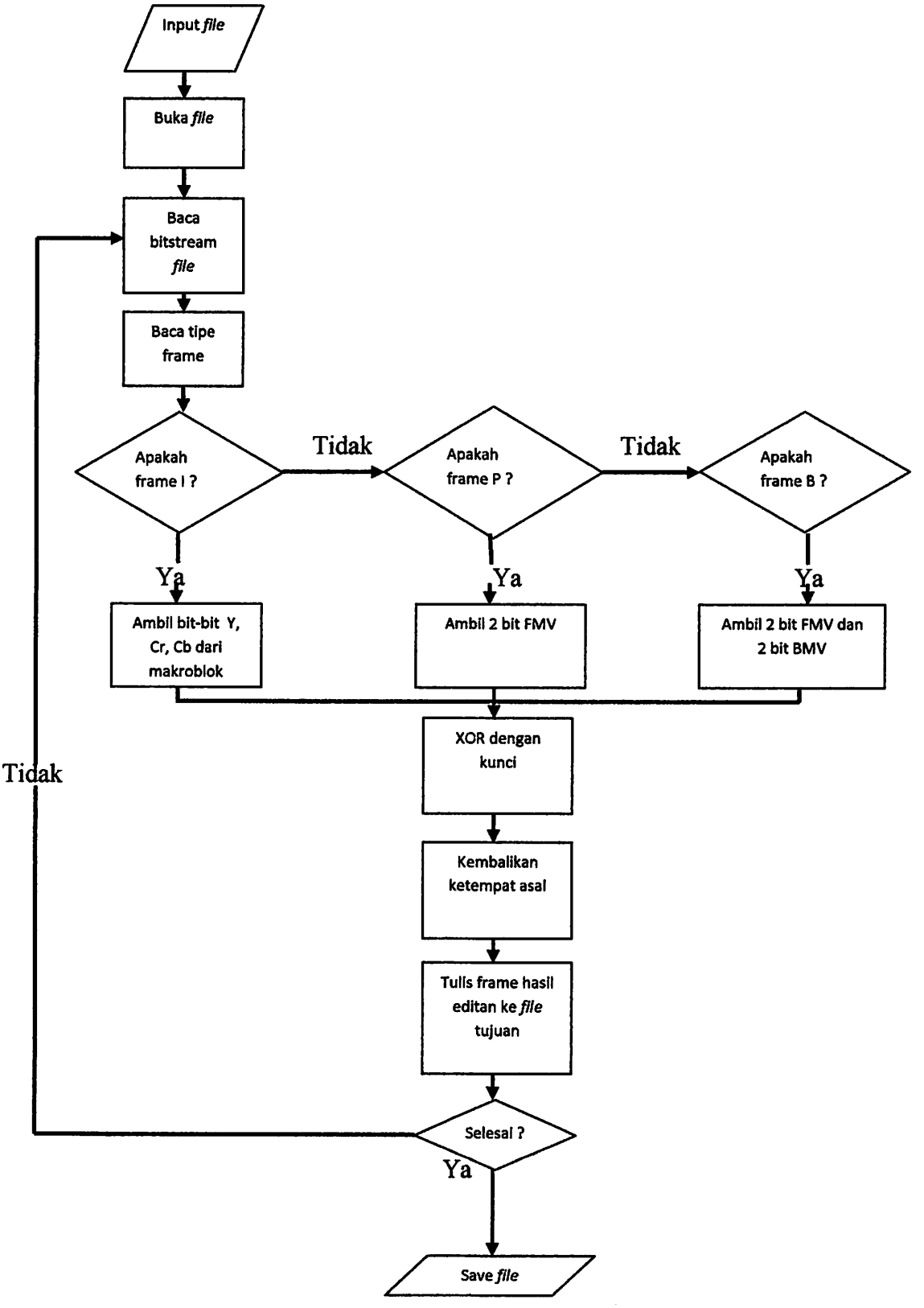
2.3.2 Algoritma *MVEA*

Algoritma *MVEA* (*Modified VEA*) merupakan salah satu modifikasi dari algoritma *VEA*, yang tetap mempertahankan metode enkripsi *cipher* aliran namun peningkatan berada pada keamanannya karena *frame* video yang dienkripsi pada algoritma *MVEA* adalah seluruh tipe *frame* video yaitu *frame* I, P, dan B.

Berikut adalah skema global dari algoritma M*VEA* :

1. Buka *file*.
2. Baca *frame file*, baca tipe *frame*-nya.
3. Baca *stream bit* dari *frame* tersebut.
4. Jika *stream* bit tersebut merupakan *stream* bit dari *frmae* I, ambil bit-bit dari *macroblok Y, Cr,* dan *Cb* dari *frame* I tersebut saja, kemudian XOR-kan dengan kunci. Kembalikan hasil enkripsi ke tempat semula.
5. Jika *stream* bit tersebut merupakan *stream* bit dari *frame* B dan P, ambil *motion* vektor-nya saja, kemudian XOR-kan dengan kunci. Hasilnya kembalikan ke tempat semula.
6. Tulis hasil enkripsi ke *file* tujuan.
7. Baca *frame* selanjutnya, kembalilah ke langkah nomor dua sampai *End-of-file.*

Diagram alir dari algoritma *MVEA* dapat dilihat di gambar II-6.



Gambar 2.6 Diagram Alir Algoritma M*VEA*

2.3.3 Algoritma R*VEA*

RVEA (*Real-time Video Encryption Algorithm*) merupakan algoritma enkripsi video yang secara selektif beroperasi pada *sign bit,* terutama pada *koefisien DCT* dan *motion vector* dari video *MPEG-1* yang telah terkompresi. Berbeda dengan *VEA*, R*VEA* mengenkripsi *frame* I, P, dan B dengan keteraturan tertentu. R*VEA* dapat menggunakan algoritma kriptografi kunci rahasia apapun seperti *DES, IDEA*, atau *AES* untuk mengenkripsi *sign bit – sign bit* tersebut. Dengan hanya memperhatikan *sign bit*pada video *stream MPEG*, maka video *MPEG S* dapat dipresentasikan dengan :

$$S = s_1,...s_2,...sm,.... \qquad (2.14)$$

Dimana s_i (i = 1, 2, ...) adalah *sign bit* dari *motion vector* atau *koefisien DCT* (untuk *koefisien DC* dari *frame* I dan *motion vector* nilainya akan berbeda dengan *sign bit* lainnya, karena di-*encode* secara berbeda). Fungsi enkripsi *RVEA*, E_k (*E* adalah *DES*), mengenkripsi video *MPEG* perbagian (*slice*). *RVEA* mengganti *koefisien DCT* dan *motion vector* secara *random*. Berdasarkan kunci rahasia yang diberikan, sebuah *sign bit* dapat berubah menjadi nilai kebalikannya atau tidak berubah sama sekali.

RVEA memilih paling banyak 64 bit (8 *byte*) dari tiap *macroblok*, misalkan *bitstream* dari 8x8 blok video *MPEG* digambarkan sebagai–$\beta\alpha_1,\alpha_2,\alpha_3...\alpha_n$ dimana β adalah kode dari *koefisien DC*, dan α_i adalah kode untuk *koefisien AC* ke-*i* yang tidak kosong. Tiap 16x16 *macroblok* mengandung enam buah 8x8 blok. Mereka adalah Y_1, Y_2, Y_3, Y_4, C_r dan C_b. Alasan mengapa urutannya dibuat semacam itu karena *koefisien* DC lebih penting dibandingkan *koefisien AC* dengan frekuensi tinggi. R*VEA* mengurangi jumlah komputasi dengan membatasi jumlah bit yang dienkripsi, dan sebisa mungkin bit yang terpilih tersebut adalah bit-bit penting yang membangun video MPEG. Proses dekripsi dari algoritma *RVEA* sama dengan proses enkripsinya. Berikut ini adalah skema global dari algoritma *RVEA* :

1. Buka *file*.
2. Baca *frame file*, baca tipe *frame*-nya.
3. Jika *stream bit* tersebut bukan *macroblok*, maka tulis kembali ke *file* tujuan, baca *bit stream* selanjutnya.

4. Jika *stream bit* tersebut merupakan *macroblok*, cek tipe *frame* dari *macroblok* tersebut.

    a. Jika *frame* dari *macroblok* tersebut adalah *frame* I, maka ambil 64 bit awal *macroblok* dengan keteraturan.

    b. Jika *frame* dari *macroblok* tersebut adalah *frame* P, maka ambil 2 bit FMV dan 62 bit awal *macroblok* dengan keteraturan.

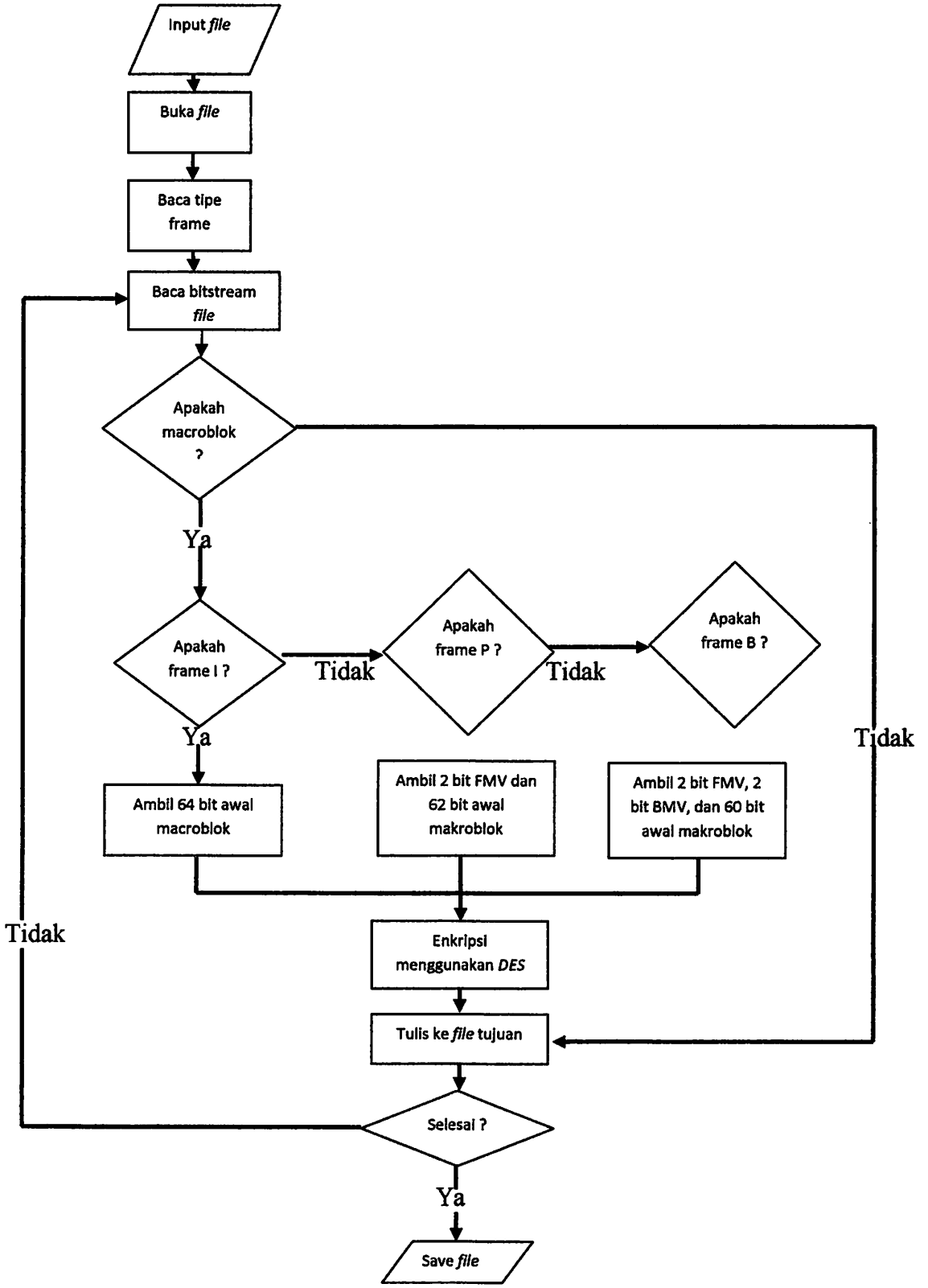
    c. Jika *frame* dari *macroblok* tersebut adalah *frame* B, maka ambil 2 bit FMV, 2 bit BMV, dan 60 bit awal *macroblok* dengan keteraturan.

5. Enkripsi 64 bit tersebut menggunakan *DES*.

6. Tulis hasil enkripsi kembali ketempat asalnya.

7. Baca *frame* selanjutnya, kembalilah ke langkah nomor dua sampai *End-of-file*.

Diagram alir dari algoritma *RVEA* dapat dilihat di gambar II-7.



Gambar 2.7 Diagram Alir Algoritma R*VEA*

2.4 *Video Codec Standart*

Beberapa *video codec standart* atau format video yang digunakan dalam tugas akhir ini adalah H.261, H.263, MJPEG, DivX, 3GP.

2.4.1 H.261

H.261 adalah *codec* khusus yang di buat oleh *ITU* pada tahun 1990 untuk keperluan global *video phone* dan aplikasi *video conferencing* melalui *ISDN*. H.261 didesain untuk *bit rate* rendah karena dengan asumsi perubahan gerak yang ada di dalam video untuk aplikasi *video phone* sangat rendah. *ITU* mengasumsikan juga bahwa ISDN akan digunakan luas di seluruh dunia. *Frame rate* dari H.261 adalah 7.5, 10, 15 atau 30 *fps* (*frame* per detik). H.261 telah diimplementasikan luas untuk keperluan *video conferencing* di Amerika Utara, Eropa, Jepang, dan merupakan awal mula dibuatnya standart *MPEG-1*.

2.4.2 H.263

H.263 dibuat oleh *ITU* pada tahun 1994 sebagai variasi dari H.261 dengan *bit rate* yang lebih rendah dari H.261. H.263 dibuat dengan tujuan untuk men-*support* aplikasi *videophone* yang menggunakan *modem PSTN* generasi baru dengan kecepatan 28.8 kbps atau lebih. Rentang *bit rate* dari H.263 adalah 8 kbps sampai 1.5 Mbps. H.263 merupakan basis dari *MPEG-4*.

2.4.3 JPEG dan MJPEG

JPEG merupakan singkatan dari "*Joint Photographic Experts Group*". Group ini mengembangkan sebuah kompresi standar untuk gambar fotografi 24-bit "*true colour*". JPEG bekerja dengan pertama kali mengkonversi sebuah gambar dari format RGB ke YcrCB untuk mengurangi ukuran *file* menjadi 1/3 sampai ½ kalinya.

MJPEG merupakan singkatan dari "*motion JPEG*" dan merupakan susunan dari gambar JPEG yang telah terkompresi untuk mempresentasikan gambar yang bergerak. Kekurangan dari MJPEG adalah tidak adanya penanganan terhadap audio.

2.4.5 DivX

DivX adalah sebuah *video codec* yang diciptakan oleh DivX, Inc. Yang menjadi terkenal karena dapat mengkompresi video menjadi ukuran yang lebih kecil dengan

kualitas yang relatif baik. DivX menggunakan kompresi *lossy MPEG-4 part 2.* Karena sering digunakan untuk membuat salinan dari DVD yang berhakcipta, DivX menjadi kontroversial. Format video kompresi berbasis MPEG-4 ini memiliki ukuran yang sangat kecil, bahkan dapat mencapai kurang dari seperlapan ukuran dari MPEG-2 dengan kualitas yang tetap terjaga. Format ini sering disebut "video MP3".

Biasanya berkas video DivX memiliki ekstensi .divx, .mp4, .avi. Ekstensi .avi milik DivX tentu berbeda dengan .avi standart. Agar perangkat lunak *player* yang digunakan dapat memutar berkas dengan format ini, harus di tambahkan *adds-on* atau *DivX codec* atau *player* khusus. DivX untuk *windows* secara khusus ditujukan untuk pengguna *Windows Operating System.* Perangkat lunak ini dapat menciptakan format video sekaligus dapat menampilkannya. DivX untuk *windows* dilengkapi dengan program tambahan seperti *DivX codec, DivX player, DivX converter,* dan *DivX Web player.*

2.4.6 3GP

3GP biasa juga disebut sebagai *3GPP* (*3rd Generation Partnership Project file format*). Format *file* ini merupakan format *file* hasil perjanjian dan kolaborasi sejumlah standar telekomunikasi terkemuka yang dikenal sebagai *"Organizational Partner".* Didalamnya termasuk ARIB, CCSA, EITSI, TTA, dan TTC. Format *file* 3GP merupakan format penampung multimedia (*media container*) yang digunakan untuk ponsel berteknologi 3G. Walau demikian format ini dapat juga dijalankan pada beberapa ponsel berteknologi 2G dan 4G, bahkan di komputer dan *notebook* dengan menggunakan aplikasi pemutar tertentu.

Format video 3GP dapat menampung *stream* video MPEG-4 dan H.263. Untuk audio, format *file* ini menampung *stream* audio bertipe *AMR* dan *AAC.* Ekstensi *file* ini dikenal dengan *ISO"MP4 family file".* Bahkan dalam beberapa ponsel menggunakan *file* berekstensi .mp4 untuk video 3GP.

File format 3GP ini banyak kelebihannya. Beberapa kelebihan itu antara lain :

1. Karena merupakan kesepakatan dari beberapa vendor & organisasi telekomunikasi terkemuka, format 3GP ini dapat diputar hampir disemua ponsel.

2. Kualitas gambar yang lumayan dengan ukuran *file* yang termasuk ringan.
3. Dukungan teknologi 3G yang marak saat ini memungkinkan kita dengan mudah mentransfer dan menyebarkan video jenis ini ke internet.
4. Banyaknya pengguna ponsel yang mendukung format ini sehingga kita dapat mudah berbagi *file* (*sharing*) yang dapat diakses atau diputar oleh banyak orang.
5. Tersedia banyaknya aplikasi pemutar dan konversi dari 3GP ke format yang lain ataupun sebaliknya yang dapat dimanfaatkan sesuai keperluan.

Selain kelebihan, *file* format 3GP juga mempunyai kekurangan. Ada beberapa kekurangan format *file* 3GP, yaitu :

1. *File* format 3GP biasanya memiliki kualitas yang sedikit rendah dan resolusi gambar yang kecil.
2. Rata – rata gambar per detiknya (*frame rate*) rendah, akibatnya gambar yang dihasilkan pada *file* 3GP dan MP4 terlihat patah – patah.
3. Kualitas suara yang tidak maksimal.
4. Karena mudah dan maraknya penggunaan *file* 3GP, banyak yang mempergunakan *file* berformat ini untuk tujuan negatif, seperti banyak beredarnya film porno dengan format 3GP.

2.6 Video (Gambar Bergerak)

Video adalah teknologi untuk menangkap, merekam, memproses, mentransmisikan dan menata ulang gambar bergerak. Biasanya menggunakan film seluloid, sinyal elektronik, atau media digital. Video juga dikatakan sebagai gabungan gambar-gambar mati yang dibaca berurutan dalam suatu waktu dengan kecepatan tertentu. Gambar-gambar yang digabung tersebut dinamakan frame dan kecepatan pembacaan gambar disebut dengan *frame rate*, dengan satuan *fps* (*frame per second*). Karena dimainkan dalam kecepatan yang tinggi maka tercipta ilusi gerak yang halus, semakin besar nilai *frame rate* maka akan semakin halus pergerakan yang ditampilkan.

BAB III

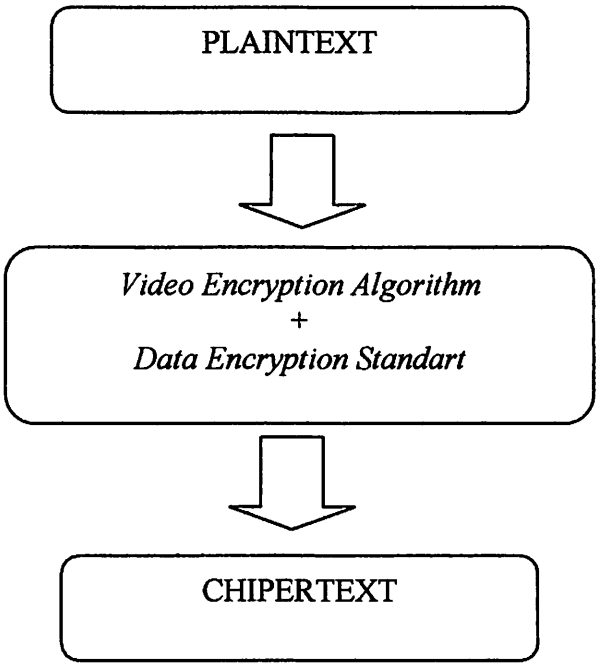
PERANCANGAN DAN ANALISA SISTEM

3.1 Analisa Masalah

Pada bab ini dijelaskan mengenai analisis dan perancangan sistem aplikasi. Analisis ditujukan untuk memberikan gambaran secara umum terhadap aplikasi dan memberikan solusi terhadap permasalahan yang dihadapi. Dalam merancang suatu sistem diperlukan analisis terhadap sistem yang akan di rancang tersebut terlebih dahulu. Tujuan dari analisis ini sendiri adalah agar sistem yang di rancang menjadi tepat guna dan ketahanan sistem tersebut akan lebih terjaga. Disamping itu dengan dilakukannya analisis akan mempermudah pekerjaan dalam membuat sistem, dan jika suatu saat nanti ada perbaikan atau penambahan dalam sistem tersebut, maka akan mudah diselesaikan.

Sistem yang dirancang terdiri dari dua proses secara garis besar yaitu proses *enkripsi* dan *deksripsi.* Dimana untuk proses *enkripsi* atau *dekripsi* menggunakan algoritma enkripsi video atau *Video Encryption Algorithm.* Algoritma ini memiliki berbagai macam modifikasi yang disesuaikan berdasarkan kebutuhan, karena kemudahannya dalam implementasi:

Model yang digunakan dalam pembuatan aplikasi *Enkripsi* video 3GP ini adalah perancangan dan pembuatan sistem. Informasi yang dikumpulkan berupa tata cara enkripsi video yg akan diterapkan. Metode yang digunakan dalam aplikasi enkripsi video ini adalah *Video Encryption Algorithm* dan *Data Encryption Standart.*



Gambar 3.1 Desain implementasi sistem.

Pada gambar diatas merupakan desain arsitektur dari implementasi sistem sesungguhnya. Dari gambar diatas dapat di lihat bahwa terrdapat dua penambahan metode atau cara dalam mengenkripsi video, yaitu *video encryption algorithm* dan *data encryption standart*.

Dasar tehnik *video encryption algorithm* yaitu kunci rahasia *VEA* , *k,* di-*generate* secara random dalam bentuk *bitstream* dengan panjang *m* yang dapat di tulis sebagai $k = b_1,...b_2,...b_m$. *bitstream* dari video dapat direpresentasikan dengan :

$$S = ...s_1...s_2...s_m...s_{m \oplus 1}...s_{m \oplus 2}...s_{2m}$$

3.2 Prosedur Perancangan

Pada bagian ini prosedur yang dilakukan yaitu menganalisis dan merancang aplikasi. Menganalisis terhadap semua masalah dan merancang aplikasi. Perancangan aplikasi disusun dengan membuat *flowchart* yang berfungsi untuk menggambarkan aliran data yang terjadi di dalam aplikasi. Kemudian membuat *flowchart* yang memberikan gambaran mengenai struktur enkripsi video secara keseluruhan.

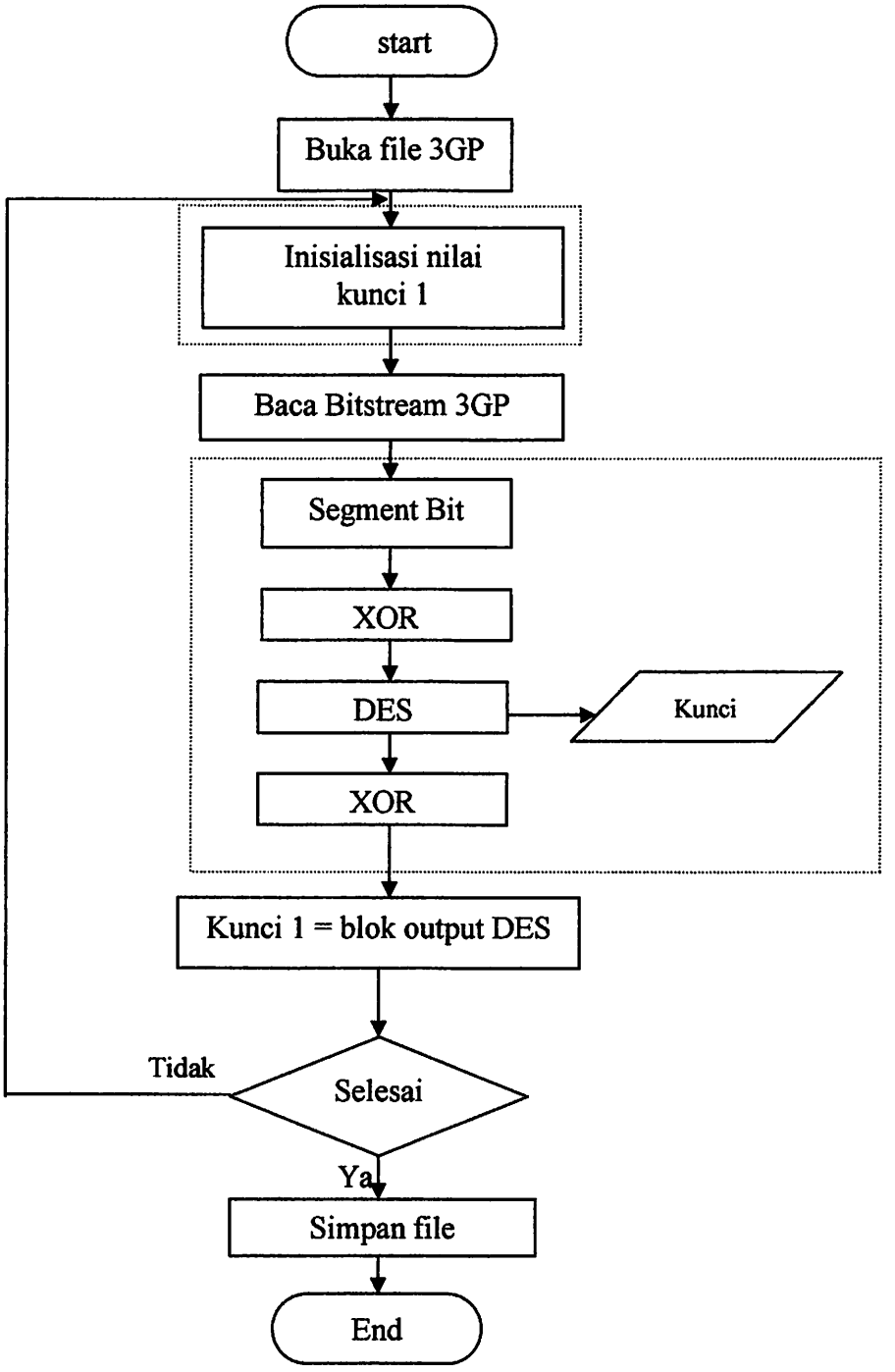
3.3 Identifikasi Permasalahan

Tahap *identifikasi* masalah merupakan tahapan paling awal untuk melakukan perancangan dan pembuatan aplikasi. Tahapan ini digunakan untuk melakukan *observasi* atau penelusuran permasalahan untuk mendapatkan permasalahan umum dari permasalahan yang dihadapi. Didalam tahapan ini juga dilakukan perumusan permasalahan yaitu merumuskan atau menetapkan permasalahan yang dihadapi, sehingga penelitian dapat lebih fokus untuk mencari dan memecahkan permasalahan yang ada.

Pada skripsi ini, fokus usaha penyelesaian masalah terdapat pada perancangan dan pembuatan aplikasi yang mengimplementasikan VEA dan DES dalam sebuah video dengan format 3GP.

3.4 Alur Proses Aplikasi

Untuk lebih memperjelas alur dari sistem, maka alur proses aplikasi akan digambarkan pada *flowchart* dibawah. *Flowchart* alur proses tersebut menjelaskan rancangan urutan proses yang terjadi pada aplikasi.



Gambar 3.2 Flowchart desain aplikasi enkripsi video

Algoritma dari *flowchart* pada aplikasi enkripsi video diatas dapat dijabarkan sebagai berikut :

1. **Start** atau memulai program aplikasi.
2. **Input file 3GP** merupakan masukan untuk memilih file video mana yang akan di gunakan dalam proses enkripsi.
3. **Inisialisasi nilai kunci 1** merupakan proses dalam menginisialisai nilai-nilai dari kunci.
4. **Baca bitstream 3GP** merupakan proses pembacaan bitstream dari file video 3GP yang akan di enkripsi. Dan kemudian di kelompok kan dalam setiap character.
5. **Segmen Bit** , setelah proses membaca bitstream maka dilakakukan pengelompok kan pada bit, yang di kelompokkan dalam 8 character.
6. **XOR,** jika pengecekan segment bit telah selasai maka baru di-XOR-kan.
7. **DES,** melakukan proses enkripsi yang di lakukan pada file video. DES mengenkripsi blok – blok dengan ukuran 64 bit.
8. **Kunci,** memasukkan kunci sebagai nilai untuk proses enkrisi agar video tidak dapat di jalankan.
9. **Selesai,** jika proses telah dirasa selasai maka tulis kedalam file keluaran, jika belum maka ulangi proses nomer 2 hingga selesai.

3.5 Perancangan Perangkat Lunak

3.5.1 Proses Enkripsi Video

Yang pertama dilakukan pada tahap ini yaitu inisialisasi nilai kunci sebelum dilakukan pembacaan bitsream pada file video 3gp. Berikut adalah listing program pada saat inisialisasi nilai kunci :

```
cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, e.toString() );
            System.out.println("Error :" +e.toString());
```

Kemudian setelah inisialisasi selesai di lakukan maka pada tahap selanjutnya yaitu dilakukan pembacaan bitsttream pada file video 3gp. Berikut ini adalah listing program untuk pembacaan bitstream dari file video 3gp :

```
fis = new FileInputStream(inputFcp);
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null, e.toString() );
            System.out.println("Cannot open file!");
            System.exit(-1);
```

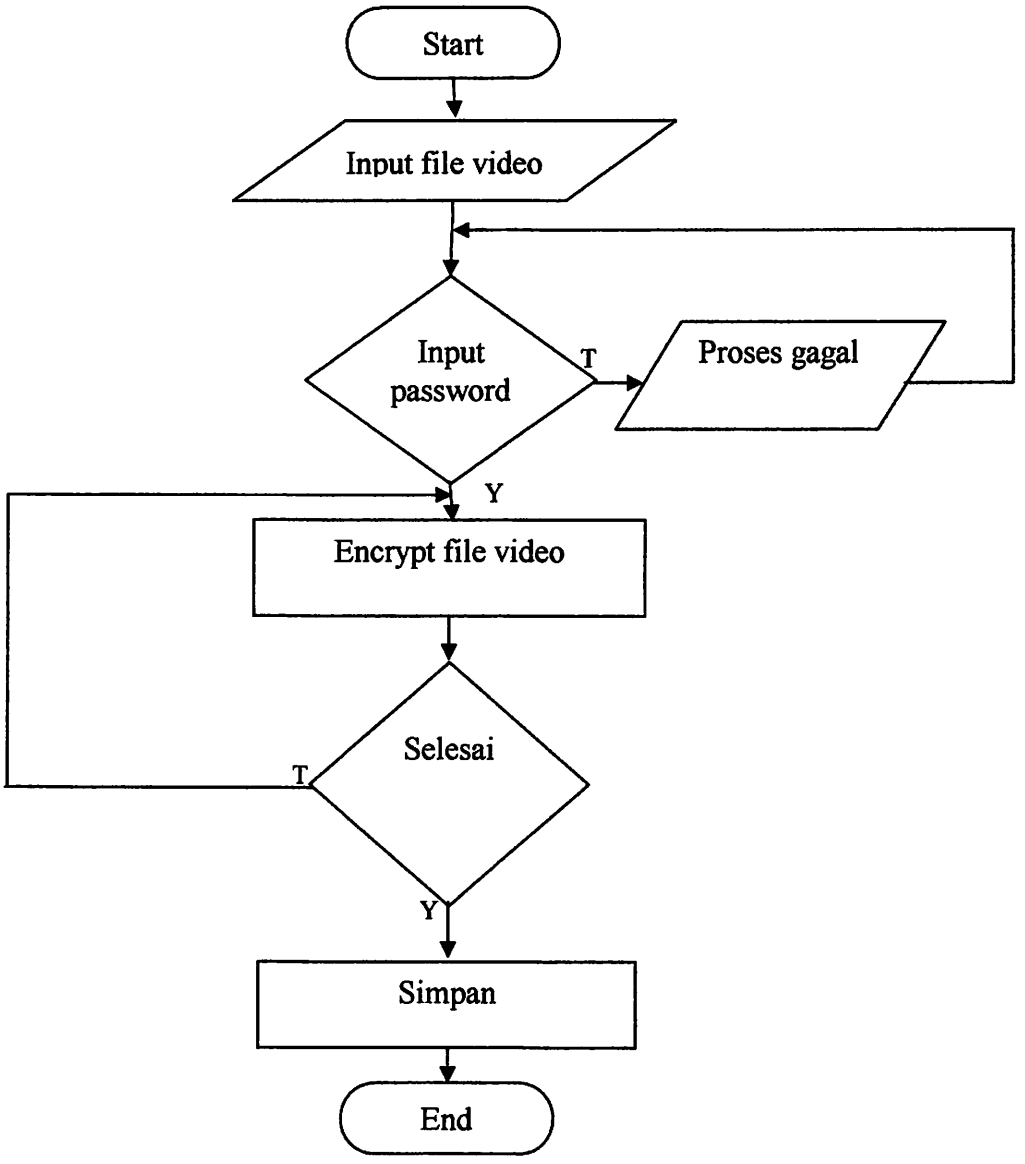
Pada saat file video di inputkan maka program akan mencoba untuk membaca bitstream dari file video tersebut, yang kemudian akan di kelompokkan menjadi beberapa kelompok segment bit. Berikut ini listing program pada saat mengelompokkan bitstream menjadi segment bit :

```
cis = new CipherInputStream(fis, cipher);
```

kemudian dilakukan perubahan dari plainteks menjadi chiperteks dengan DES pada segment bit tersebut yang setiap blok mempunyai ukuran 64 bit. Berikut ini adalah listing program pada saat enkripsi dilakukan :

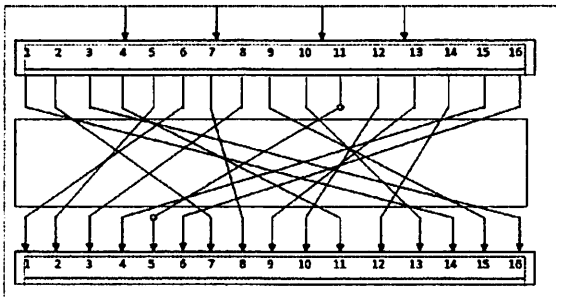
```
fos = new FileOutputStream(desFile);
        byte[] b = new byte[8];
        int i = cis.read(b);
        while (i != -1) {
            fos.write(b, 0, i);
            i = cis.read(b);
```

setiap blok di urutkan menjadi 8 bit yang kemudian di enkrip dengan DES. Hal ini dilakukan terus menerus hingga urutan blok yang terakhir.



Gambar 3.3 Flowchart proses enkripsi video

Pada proses enkripsi ini hanya bekerja pada file video yang telah di inputkan oleh user. User akan diminta untuk memasukkan *password* yang digunakan untuk proses enkripsi video tersebut. Sebelum video tersebut di enkripsi digunakan operasi *Data Encryption Standard* (DES) dimana DES adalah sebuah algoritma enkripsi sandi blok kunci simetrik dengan ukuran blok 64-bit dan ukuran kunci 56-bit. Dalam proses *DEScrypt* ini data di enkripsi menggunakan algoritma DES . *DEScrypt* adalah sebuah fungsi yang mengimplementasikan algoritma *Data Encryption Standart.* DES memerlukan minimal 8 bit key untuk mengenkripsi suatu data.



Pada tahap ini di lakukan pembacaan pada bitstream dari *file* video *3gp*. Pembacaan bitstream merupakan pengenalan dari stiap character *bitstream* yang terdapat pada *file* video. Kemudian *bitstream* tersebut di kelompok kan menjadi 8 character, karena *DES* beroperasi pada ukuran blok 64 bit. Setelah *bitstream* di kelompokkan dalam setiap segment bit maka di-XOR-kan. Proses enkripsi dilakukan tiap per segment bit.

Setelah semua parameter terpenuhi, maka program akan meng-enkripsi *file* video yang telah di inputkan. Setelah video tersebut berhasil di enkripsi maka video tersebut dapat di simpan dengan format *3gv* dengan nama *file* yang user telah tentukan. Jika *file* video tidak berhasil di enkripsi maka proses akan dilanjutkan dengan mengulangi proses memasukkan *password* yang user tentukan.

3.5.2 Proses Dekripsi Video

Dalam proses dekripsi ini hampir sama dengan proses enkripsi, hanya saja chiperteks dirubah kembali menjadi plainteks. dalam hal ini *chiperteks* akan di rubah kembali menjadi plainteks dengan nilai kunci yang sama pada saat dilakukan enkripsi pada plainteks.

Plainteks :00110011 10100010 11100010

Misalkan nilai kunci yang dimasukkan yaitu : 010

Chiperteks : 00110010 10100011 11100010

Sama seperti pada proses enkripsi, proses yang pertama yaitu inisialisasi nilai kunci. Berikut adalah listing program pada saat inisialisasi nilai kunci :

```
try{
            cipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
        }catch(Exception e){
         // JOptionPane.showMessageDialog(null, e.toString() );
          System.out.println("Error : " +e.toString());
        //   System.exit(-1);
```

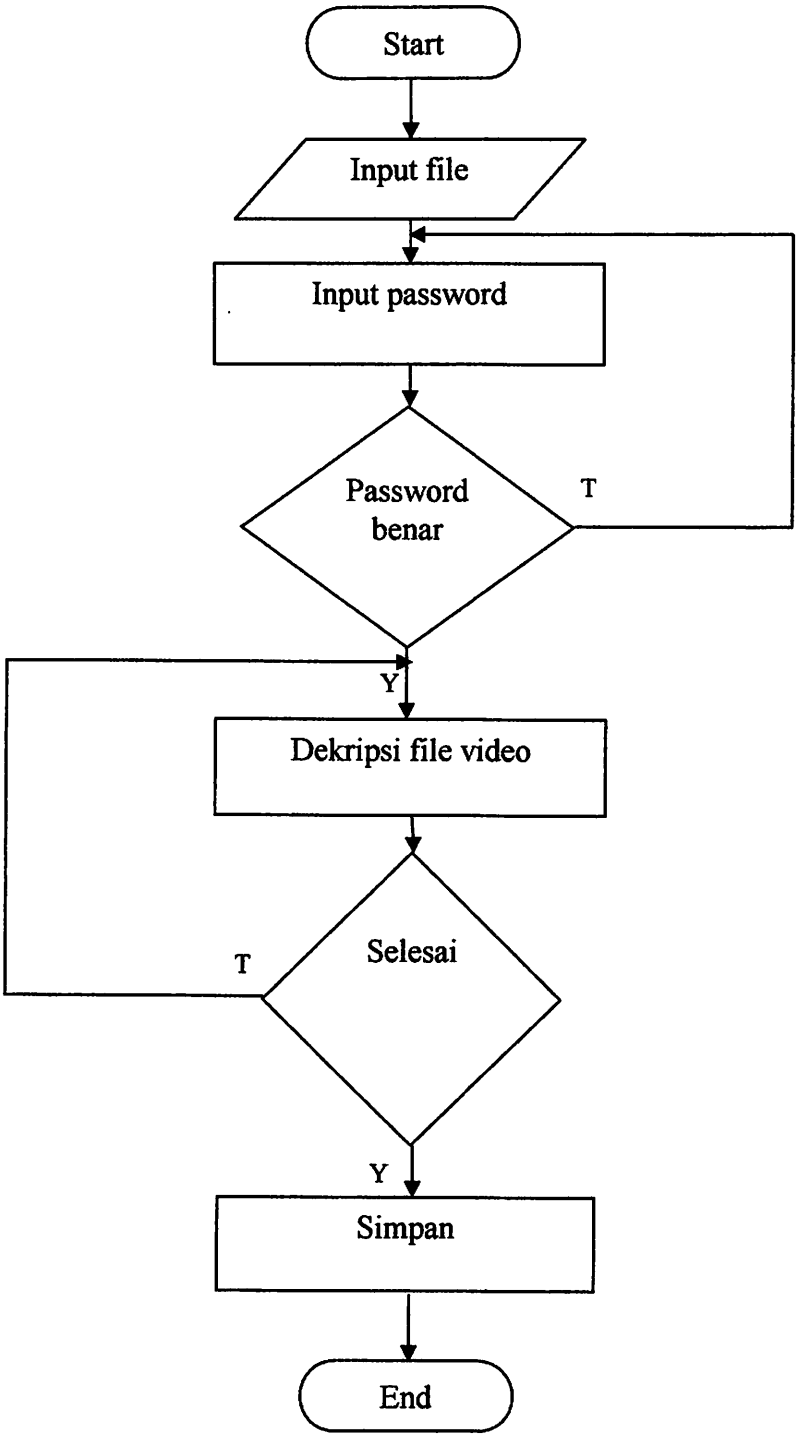
Kemudian dilakukan pembacaan bitsream pada *file* video tersebut. Berikut adalah listing program pada saat di lakukan pembacaan *bitsreeam* pada *file* video :

```
//read bitstream
            fis = new FileInputStream(desFile);
            //do decription
            cis = new CipherInputStream(fis, cipher);
```

setelah pembacaan *bitsream* selesai maka akan dilakukan proses perubahan dari *chiperteks* menjadi *plainteks*. setiap segment bit yang ada pada file video tersebut akan

di rubah kembali berdasarkan nilai kunci yang. Berikut ini adalah listing program ketika proses dekripsi dari *chiperteks* menjadi *plainteks* :

```
fos = new FileOutputStream(desFileBis);
        byte[] b = new byte[8];
        int i = cis.read(b);
        while (i != -1) {
            fos.write(b, 0, i);
            i = cis.read(b);
```



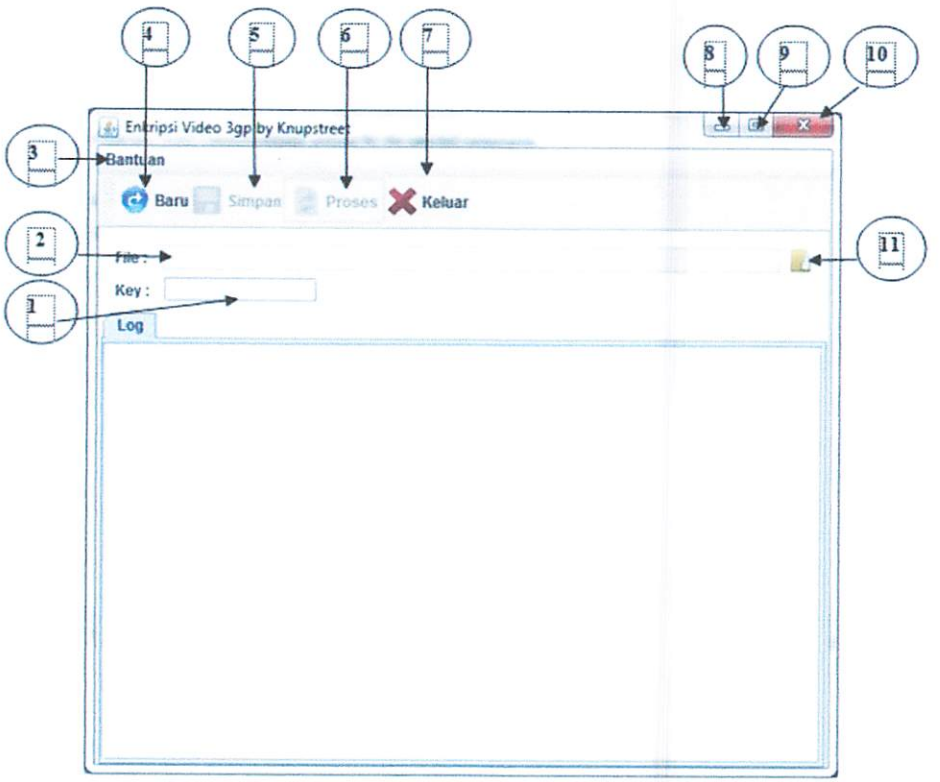
Gambar 3.4 Flowchart Dekripsi video

Sebuah *file* video yang telah di enkripsi akan dapat di kenali dengan cara melihat format atau ekstensi dari *file* video tersebut. Sebuah video *3gp* jika di enkripsi dengan aplikasi ini maka *file* outputnya akan berubah menjadi *3gv*. Sebuah file video telah di enkripsi dengan sempurna maka tidak dapat di putar atau dibaca oleh semua player

video. Fungsi *Decrypt* memanggil fungsi *DESdecrypt* untuk mendekrpisi *file* video tersebut. Fungsi ini juga berada dalam kelas yang sama dengan fungsi *DEScrypt*. Output dari proses dekripsi berupa sebuah *file* video *3gp*. Kemudian *file* video yang telah di dekripsi tersebut dapat di simpan oleh user dengan nama yang ditentukan oleh user.

3.6 Perancangan Aplikasi

Aplikasi harus di desain sesederhana mungkin tanpa mengurangi fungsi dan kinerja perangkat lunak tersebut. Pada perangkat lunak ini di buat halaman *interface* yang mudah untuk berinteraksi dengan *user* dengan beberapa halaman saja, yaitu halaman Menu Utama, halaman Pencarian *file* video, dan halaman Penyimpanan *file* video.



Gambar 3.5 Desain Rancangan Aplikasi

Perangkat lunak aplikasi ekripsi dan dekripsi video *3gp* ini dirancang dengan menggunakan bahasa pemrograman *Java* dengan beberapa komponen *standard* seperti:

1. *Textfield key* (untuk memasukkan *password* sebelum proses enkripsi atau proses dekripsi)
2. *Textfield* nama *file* video (memberitahukan user nama *file* video yang akan diproses).
3. *Jmenu Bantuan,*
4. *Button* Baru (berfungsi untuk me-*refresh* aplikasi).
5. *Button* Simpan (berfungsi untuk memanggil form *save*).
6. *Button* Proses (berfungsi untuk memproses *file* video).
7. *Button* Keluar (berfungsi untuk menghentikan aplikasi atau keluar dari aplikasi).
8. *Button hide* (Untuk menyembunyikan form)
9. *Button Maximize* (untuk memperbesar form)
10. *Button close* (untuk menutup aplikasi)
11. *Button browse* (berfungsi untuk memanggil form pencarian *file* video).

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan dijelaskan mengenai implementasi dan uji coba dari Aplikasi Enkripsi video *3GP*. Setelah melalui proses analisis dan perancangan perangkat lunak, dilakukan implementasi sistem. Bab ini membahas tentang implementasi sistem.

4.1 Kebutuhan Sistem

Sebelum menjalankan program atau aplikasi secara *standalone*, ada beberapa hal yang perlu diperhatikan, antara lain kebutuhan sistem akan perangkat keras (*hardware*) dan perangkat lunak (*software*), serta langkah-langkah yang harus dilakukan untuk dapat melakukan instalasi aplikasi agar dapat berfungsi sebagaimana mestinya. Dalam perancangan dan pembuatan aplikasi ini ada beberapa perangkat keras dan lunak komputer yang dibutuhkan antara lain :

4.1.1 Perangkat Keras

Perangkat keras komputer adalah komponen – komponen fisik peralatan yang membentuk suatu sistem komputer, serta peralatan-peralatan lain yang mendukung komputer dalam menjalankan tugasnya. Adapun perangkat keras yang diperlukan dalam pengujian aplikasi ini adalah :

1. CPU dengan processor 1200 Mhz atau lebih
2. Monitor LCD
3. Memory 1 GB atau lebih.
4. VGA Card dengan memory 128 MB atau lebih.
5. Printer

6. Mouse, Keyboard dan CDROM.

4.1.2 Perangkat Lunak

Perangkat lunak yang diperlukan adalah program komputer yang diperlukan untuk mengoperasikan fungsi dari perangkat keras. Adapun perangkat lunak yang diperlukan dalam perancangan dan pembuatan aplikasi ini adalah :

1. Sistem Operasi Windows XP, atau Windows 7
2. NetBeans IDE 6.9.1

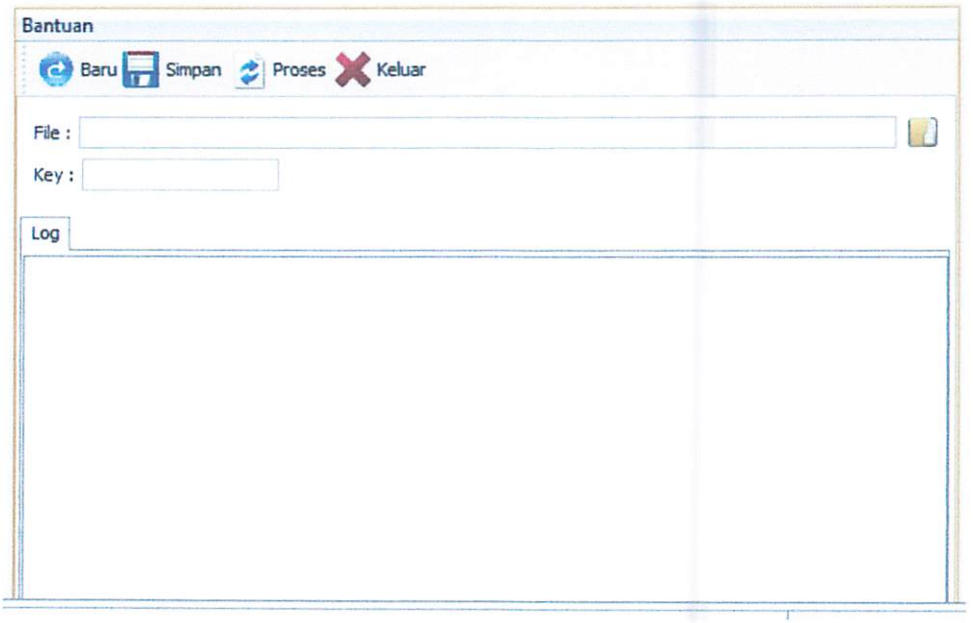
4.2 Penggunaan Aplikasi

Pada sub bab akan dijelaskan tentang penggunaan aplikasi per sistem menu, mulai dari tampilan aplikasi, fungsi dan cara penggunaannya.

4.3 Tampilan Aplikasi Enkripsi Video

4.3.1 Tampilan Menu Utama

Tampilan menu utama merupakan tampilan pembuka pada saat aplikasi pertama kali dijalankan. Di dalamnya terdapat menu utama dan sub menu dari masing-masing menu utama.



Gambar 4.1 Tampilan Menu Utama

Di dalam halaman ini kita dapat melakukan enkripsi video. Parameter dalam aplikasi ini yaitu sebuah *file* video dengan format ekstensi *3GP*.

Di halam ini juga terdapat tools untuk membantu user dalam melakukan enkripsi video, diantaranya adalah :

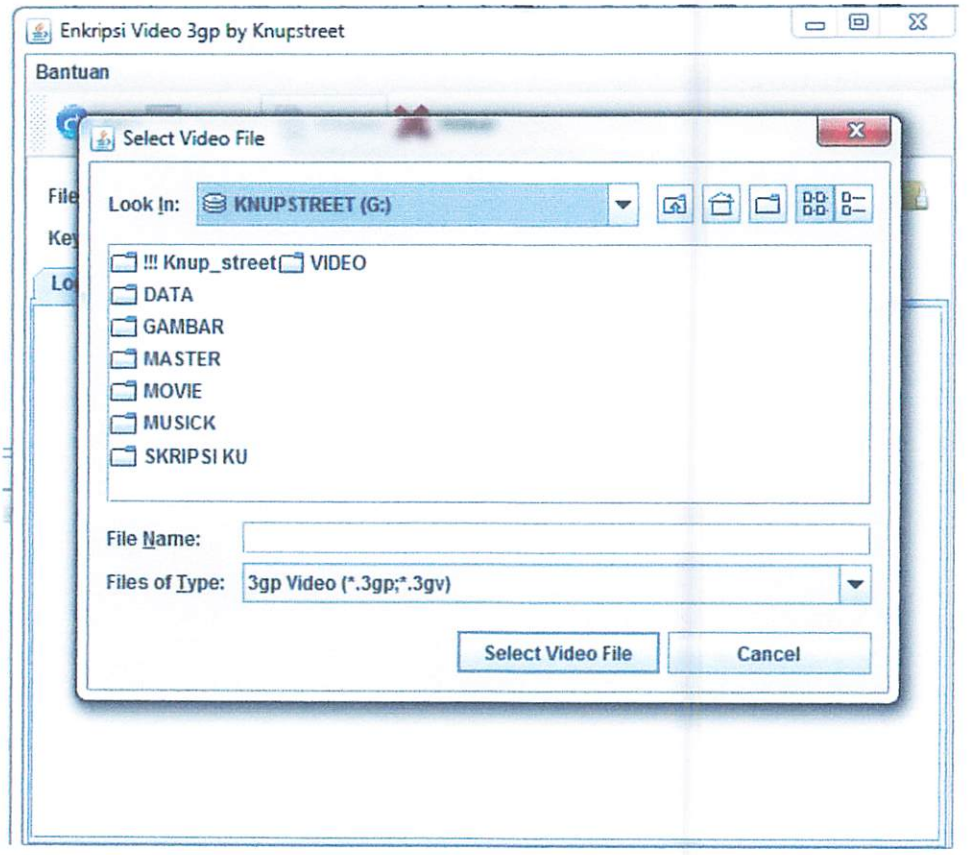
a. Baru : Untuk me-refresh aplikasi.

b. Simpan : Untuk menyimpan file video.

c. Proses : Untuk memproses file video.

d. Keluar : Untuk keluar dari aplikasi.

4.3.2 Tampilan Browse File

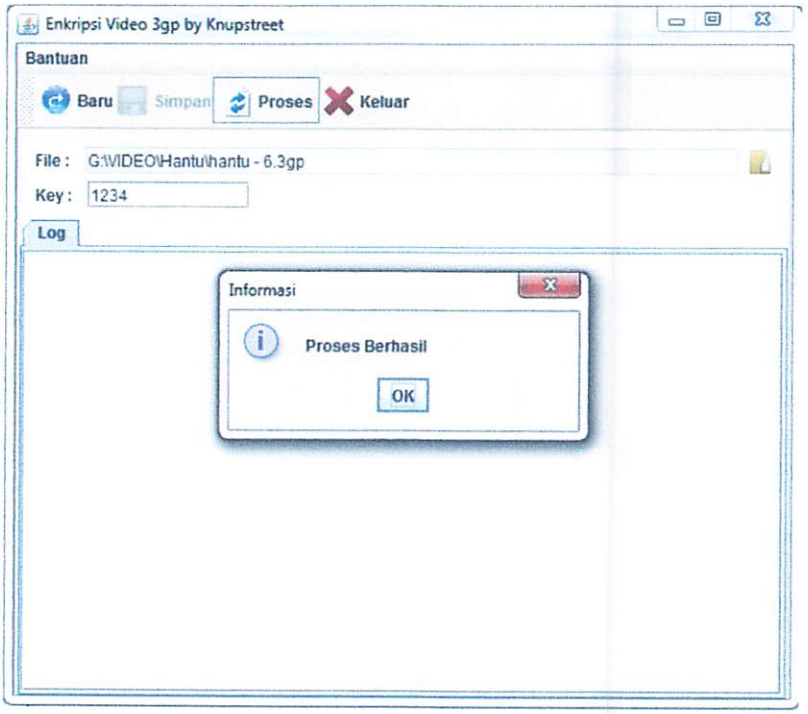


Gambar 4.2 Tampilan *browse file*.

Tombol *close* berfungsi untuk membatalkan *file* video yang akan di pilih. Tombol *close* mempunyai fungsi yang sama dengan tombol *cancel* yaitu membatalkan pemilihan video. Sedangkan tombol pencarian lokasi digunakan untuk mencari di *drive* mana *file* video yang akan di enkripsi tersebut. Tombol format *file* video memberitahukan bahwa hanya *file* video dengan ekstensi *3gp* dan *3gv* yang bisa digunakan dalam aplikasi ini. Sedangkan pada kolom nama *file* video berfungsi untuk memberitahukan nama *file* video yang kan digunakan dalam enkripsi, serta mengingatkan kepada *user* apakah user telah memilih *file* video benar dan sesuai dengan nama dari *file* video yang dimaksud oleh *user*. Tombol *select* video *file* mempunyai fungsi untuk memilih *file* video yang sudah di tetapkan oleh user untuk di

enkripsi. Tombol *view* berfungsi untuk merubah tampilan dari tampilan *list* menjadi tampilan *icon*.

4.3.3 Tampilan Proses Enkripsi



Gambar 4.3 Tampilan setelah proses enkripsi berhasil

Berikut adalah listing program untuk untuk proses enkripsi pada inisialisasi kunci dan *file* yang akan di enkripsi :

```
public void EncriptFile(String inputFcp,String outputDes,String strKey)
  {
    try {
        int iterationCount = 2;
        Cipher cipher;
        //generate key
        KeySpec keySpec = new PBEKeySpec(strKey.toCharArray(), salt,
iterationCount);
```

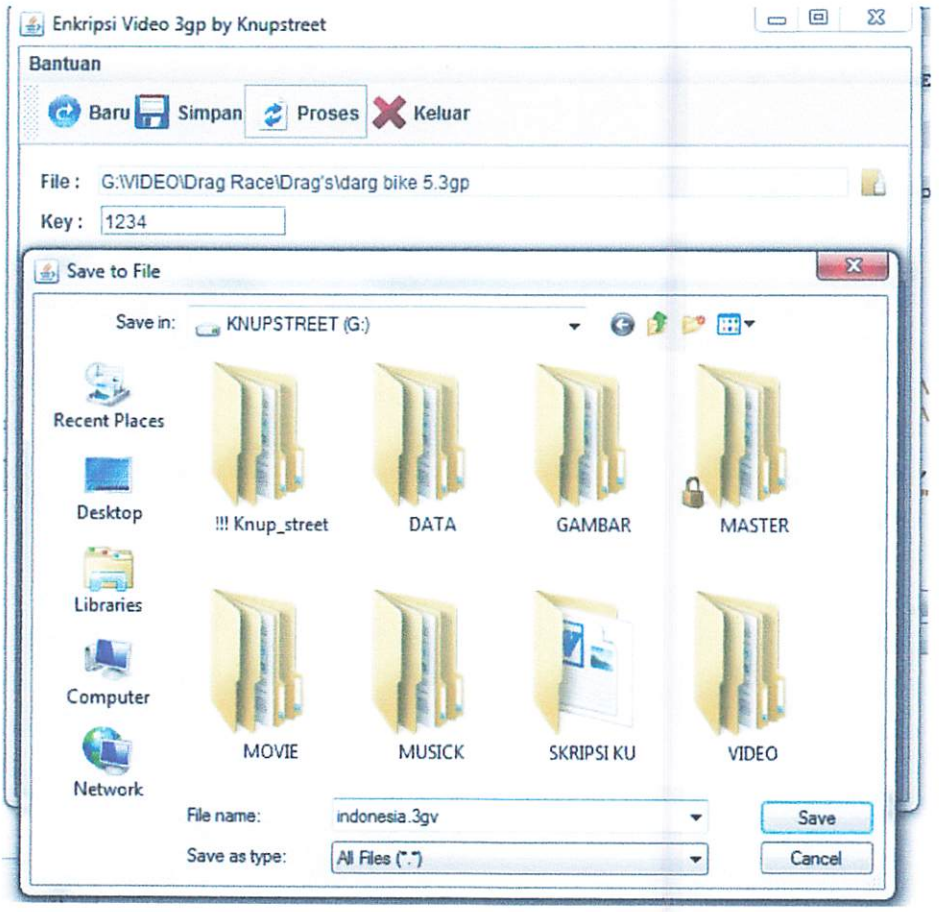
```
        SecretKey key =
SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);

        cipher = Cipher.getInstance(key.getAlgorithm());
        AlgorithmParameterSpec paramSpec = new PBEParameterSpec(salt,
iterationCount);
        //create output file
        File desFile = new File(outputDes);
        FileInputStream fis=null;
        FileOutputStream fos;
        CipherInputStream cis;
        try {
            // init mode ecryption
            cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
        }catch(Exception e){
            JOptionPane.showMessageDialog(null, e.toString() );
            System.out.println("Error :" +e.toString());
         //   System.exit(-1);
        }
        try {
           //read bitstream
            fis = new FileInputStream(inputFcp);
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null, e.toString() );
            System.out.println("Cannot open file!");
```

```
            System.exit(-1);
        }
        //do encryption bitstream
        cis = new CipherInputStream(fis, cipher);

        // Write to the Encrypted file
        fos = new FileOutputStream(desFile);
        byte[] b = new byte[8];
        int i = cis.read(b);
        while (i != -1) {
            fos.write(b, 0, i);
            i = cis.read(b);
        }
        //close file handle
        fos.flush();
        fos.close();
        cis.close();
        fis.close();
    } catch (Exception e) {
      e.printStackTrace();
      return;
    }
}
```

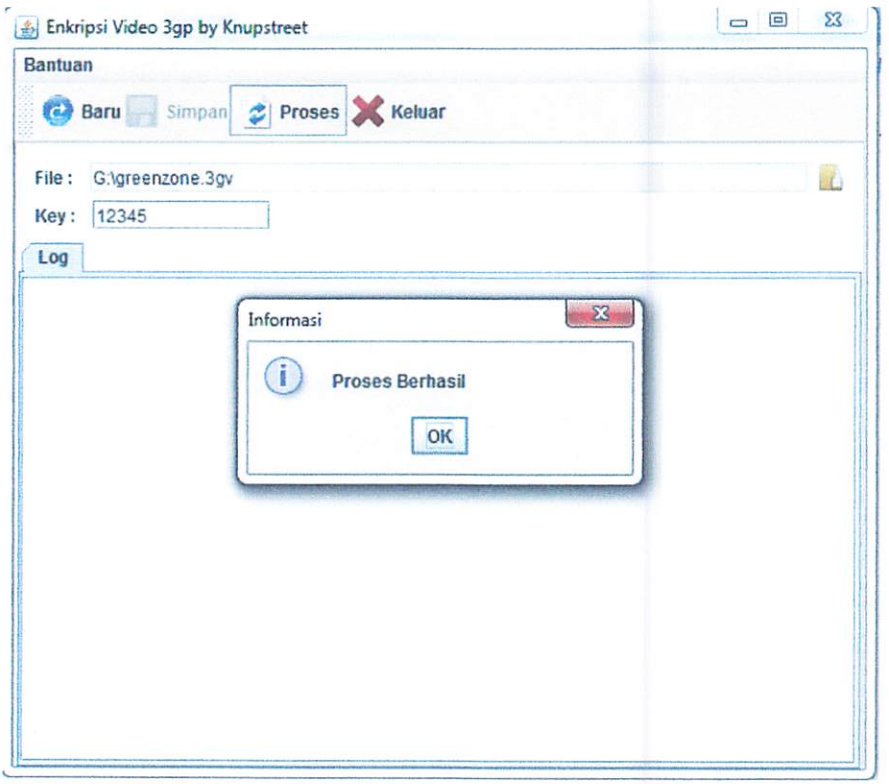
4.3.4 Tampilan Proses Save Enkripsi



Gambar 4.4 Tampilan proses *save file*

Pada halaman desain penyimpanan ini terdapat beberapa fungsi yang hampir sama dengan desain menu pencarian *file* video. Tombol *close* berfungsi untuk membatalkan penyimpanan *file* video dan kembali ke menu utama. Tombol *cancel* mempunyai fungsi yang sama dengan tombol *close* yaitu membatalkan *file* yang akan disimpan. Pada *textbok file name* digunakan untuk memberi nama *file* video yang akan disimpan oleh *user*. Pada *combo box save file type* berfungsi untuk memberitahukan *user* ekstensi dari *file* video yang akan disimpan tersebut.

4.3.5 Tampilan Proses Dekripsi



Gambar 4.5 Tampilan proses dekripsi berhasil

Berikut adalah listing program untuk untuk proses dekripsi pada inisialisasi kunci dan *file* yang akan di enkripsi :

```
public void DecriptFile(String inputDes,String outputOri,String strKey)
  {
    try {
          int iterationCount = 2;
          Cipher cipher;
          //generate key

          KeySpec keySpec = new PBEKeySpec(strKey.toCharArray(), salt,
iterationCount);
```

```
            SecretKey key =
SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);

            cipher = Cipher.getInstance(key.getAlgorithm());

            AlgorithmParameterSpec paramSpec = new PBEParameterSpec(salt,
iterationCount);

          //   cipher.init(Cipher.DECRYPT_MODE, key, paramSpec);

            //create output file
            File desFile = new File(inputDes);
            File desFileBis = new File(outputOri);

            FileInputStream fis;
            FileOutputStream fos;
            CipherInputStream cis;

            //init key cipher
            try{
               cipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
            }catch(Exception e){
             // JOptionPane.showMessageDialog(null, e.toString() );
              System.out.println("Error : " +e.toString());
            //   System.exit(-1);
```

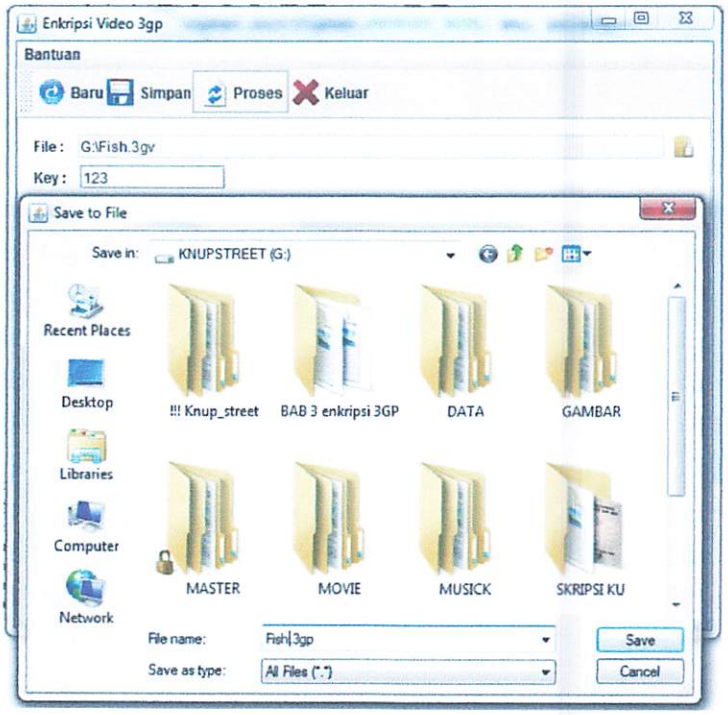
```
        }

        //read bitstream
        fis = new FileInputStream(desFile);
        //do decription
        cis = new CipherInputStream(fis, cipher);

        //create new file result
        fos = new FileOutputStream(desFileBis);
        byte[] b = new byte[8];
        int i = cis.read(b);
        while (i != -1) {
            fos.write(b, 0, i);
            i = cis.read(b);
        }
        fos.flush();
        fos.close();
        cis.close();
        fis.close();
    }
    catch (Exception e)
    {

      return;
    }
```

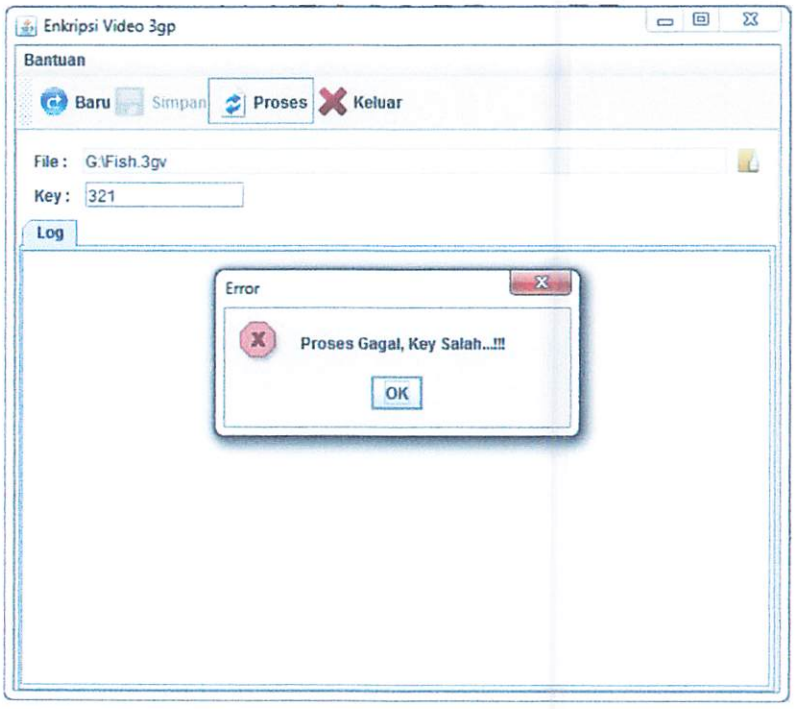
4.3.6 Tampilan Proses Save Dekripsi



Gambar 4.6 Tampilan simpan video setelah proses dekripsi

Pada gambar (gambar 4.6) ini adalah tampilan ketika terjadi proses dekripsi yang berhasil dilakukan. Tampak dalam gambar bahwa *file* telah kembali ke ekstensi *3gp*, kemudian *file* dapat disimpan dengan format *3gp*.

4.3.7 Tampilan Proses Gagal



Gambar 4.7 Tampilan setelah proses dekripsi gagal

4.4 Pengujian Aplikasi

Pengujian dilakukan bertujuan untuk mengetahui apakah aplikasi yang telah dibangun telah berjalan dengan baik dan memenuhi spesifikasi yang telah ditentukan. Pengujian pada aplikasi enkripsi video *3gp* berfokus pada keamanan *file* yang telah di enkripsi maupun di dekripsi.

4.4.1 Tujuan Pengujian.

Tujuan dilakukan pengujian adalah :

1. Mengetahui apakah rancangan model enkripsi video pada video dengan format *3GP* dapat direalisasikan.

2. Mengetahui apakah model enkripsi yang menggunakan algoritma *VEA* dapat berjalan pada video *3GP*.

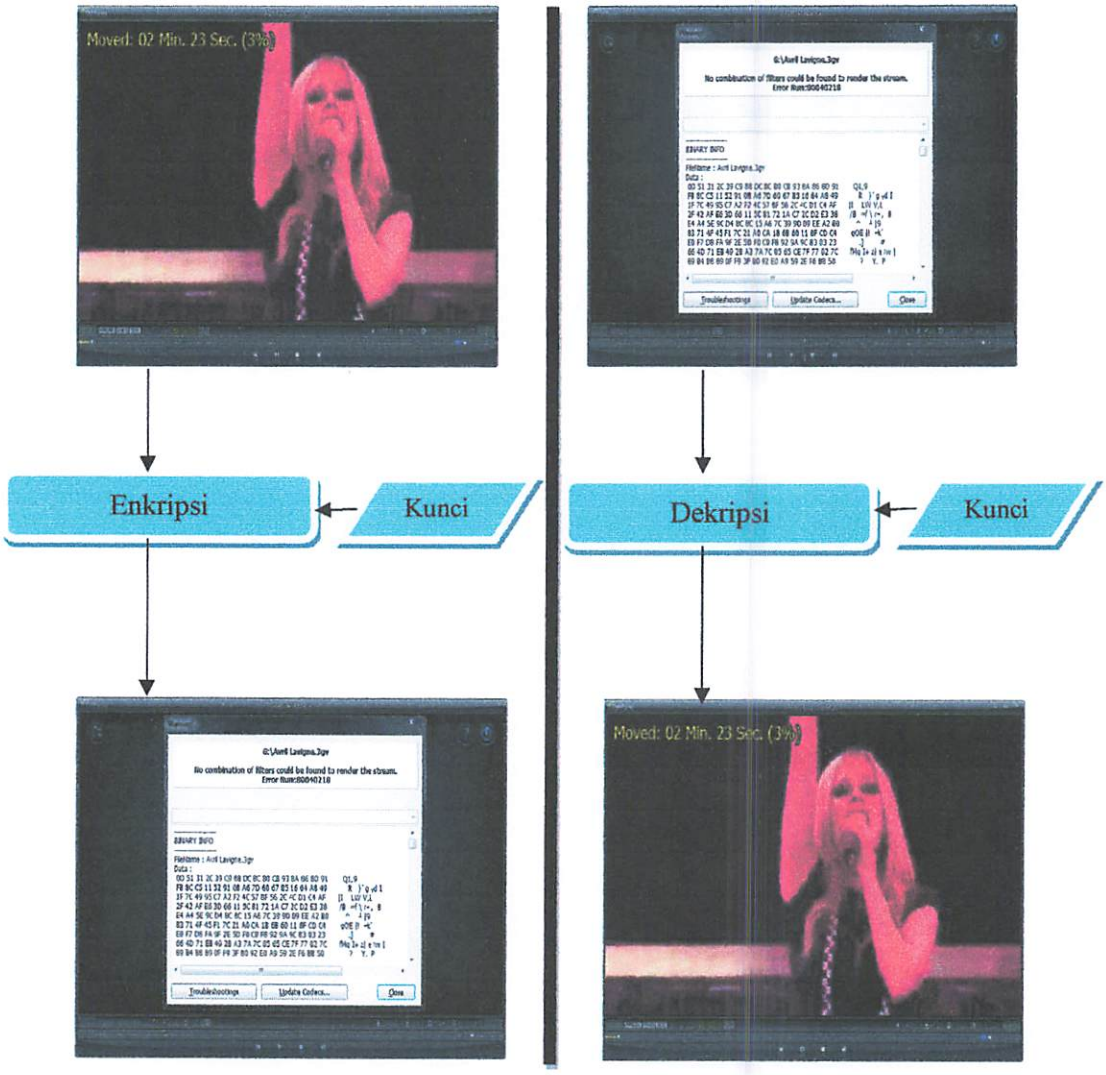
3. Mengetahui apakah model enkripsi menggunakan algoritma *DES* dapat di terapkan pada video *3GP*.

4. Mengetahui apakah perangkat lunak yang dibangun dapat diimplementasikan, terutama jika diimplementasikan pada *file* video *3GP* dengan ukuran *file* yang berbeda – beda .

5. Mengetahui apakah fungsi – fungsi yang terdapat dalam perangkat lunak dapat berjalan dengan baik.

4.4.2 Pengujian Keamanan Video Yang Terenkripsi

Di dalam pengujian kali ini akan dilakukan pemutaran video yang telah di enkripsi. Yang dijadikan perbandingan adalah sebuah video yang belum di enkripsi dan yang telah terenkripsi. Video dengan judul "Avril Lavigne.3gp" yang kemudian di enkripsi dengan perangkat lunak tersebut, setelah di enkripsi, *file* tersebut tidak dapat di putar karena *bitsream* pada video tersebut telah dirubah.



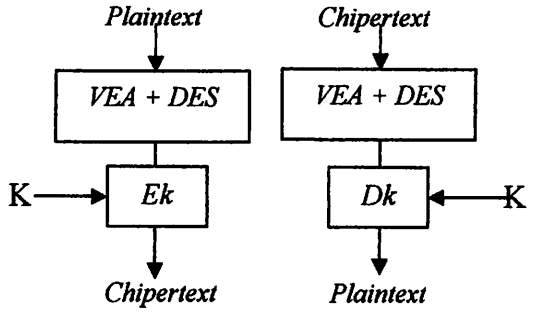
Gambar 4.8 Skema kerja aplikasi

Setelah dilakukan proses enkripsi dengan aplikasi ini maka sebuah *file* video tidak dapat di buka oleh semua *player*.

Tabel 4.1 Tabel keterangan data video

Ukuran file	Data
413 Mb	D5 91 04 16 2F 02 65 95 4D 2F 0D 43 07 C4 25 92 89 EF D8 9B 69 77 95 59 98 48 38 90 17 7F 0C 97 C1 A2 75 2A B9 A5 C4 F3 E0 44 21 9B 6F 39 8F C4 06 64 67 CC A3 E5 F4 C0 AE 3C B3 2F 20 37 CB 7D 79 0D 13 3D 7C DF 72 A5 08 07 3F 7A C3 B4 35 85 F6 B0 C6 4C 98 46 E5 FF 2A AB 38 47 D6 F5 24 81 FE 5E 29 56 E5 CE 38 B7 DF F5 24 DD 8E 9B 03 6C 1D 71 37 E2 CE 87 2A 49 E2 94 AF 39 F1 AE 35 02 CB 93 62 84 E7 C2 9F 92 03 21 25 D4 FA 1E 7B 6C AE 8B 29 77 12 87 BB 8C 2B 3D E0 1A 5E A5 BC 6B 1D 84 15 41 F5 C7 92 B4 35 BF 7F FA B1 86

Bitsream yang terdapat pada video tersebut telah di enkripsi menggunakan *VEA* dan *DES* sehingga bit – bit tersebut sudah di acak dan tidak dapat di baca oleh semua pemutar video.

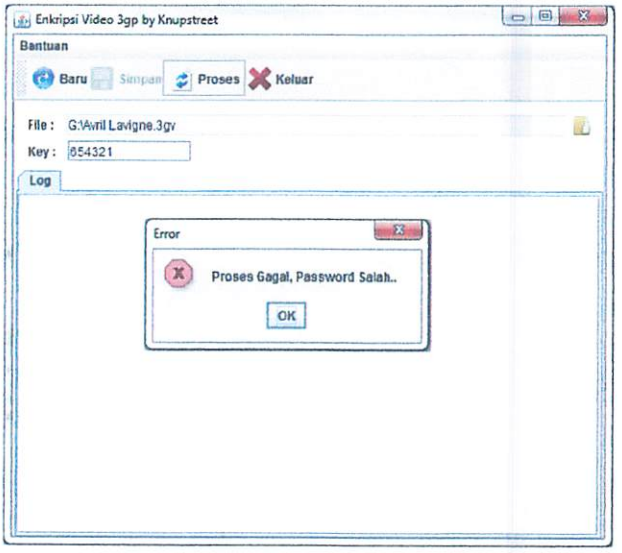


Gambar 4.9 Skema global enkripsi dan dekripsi.

4.4.3 Pengujian Kebenaran Proses Enkripsi Dan Dekripsi

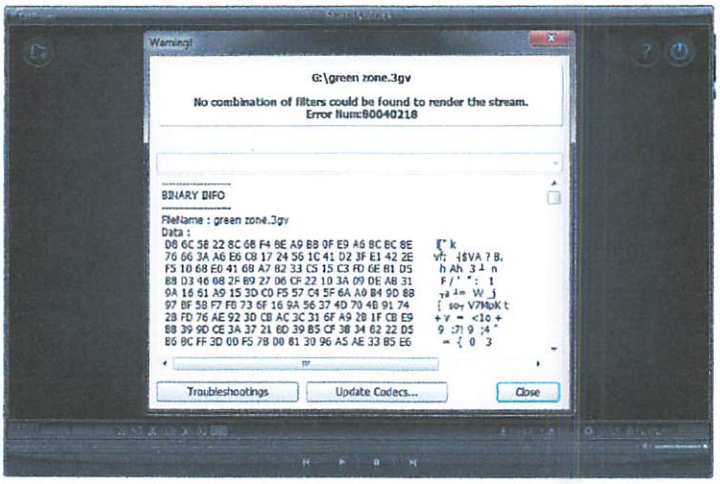
Pengujian ini dilakukan dengan cara memilih sebuah *file* video *3gp* untuk di enkripsi dengan kunci masukan *user* '123456' . Selanjutnya *file* video hasil enkripsi di

dekripsi dengan kunci yang sama dengan kunci proses enkripsi sebelumnya. Video hasil dekripsi ini menghasilkan video yang sama dengan video sebelum dienkripsi. Kemudian dilakukan pengujian dengan cara melakukan dekripsi dengan memasukkan kunci '654321'. Pengujian ini tidak dapat di proses di karenakan kunci yang di masukkan oleh user salah atau berbeda dengan kunci yang di masukkan pada saat enkripsi sebelumnya.



Gambar 4.10 Tampilan Proses gagal

Data yang akan di enkripsi berupa bitstream yang telah di kelompokkan menjadi beberapa *character*. *DES* membutuhkan 64-bit dalam pengoperasiannya maka setiap *segment bit* yang akan di enkripsi di kelompokkan menjadi 64-bit, maka di perlukan 8 *character*. Pada video nomer 1 dilakukan pengujian setelah di enkripsi. Setelah di enkripsi ukuran dari video tersebut tidak berubah sama sekali, hanya format video yang berubah menjadi *3gv*. Proses enkripsi dilakukan dengan cara merubah *segment bit* blok per blok.



Gambar 4.11 video setelah di enkripsi

Setelah video di enkripsi, data *bitream* pada video berubah menjadi acak dan tidak dapat di putar, hal ini di tunjukkan pada gambar 4.11. Pada percobaan tahap ini, ukuran *file* setelah di enkripsi tidak berubah sama sekali. Ini di karenakan pada proses enkripsi tidak terjadi pengurangan maupun pengambilan data pada *bitstream*.

4.4.4 Pengujian Terhadap Ukuran File dan Waktu Proses

Sebuah *file* video mempunyai ukuran yang bervariasi. Pengujian dalam tahap ini dilakukan pada beberapa ukuran *file* video yang telah disiapkan. Pada pengujian kali ini di lakukan pada proses enkripsi.

Tabel 4.2 Hasil Ukuran File Dan Waktu Proses

No	Nama File Plainteks	Ukuran file plainteks (byte)	Ukuran file chiperteks (byte)	Waktu proses enkripsi	Waktu proses dekripsi
1	Fish.3gp	3.594	3.594	00:00:06	00:00:06
2	Surga Dalam Bubuk Mesiu.3gp	21.303	21.303	00:00:19	00:00:19
3	Avril Lavigne.3gp	239.570	239.570	00:03:36	00:03:36
4	Green Zone.3gp	339.796	339.796	00:04:56	00:04:56
5	Muse.3gp	413.408	413.408	00:05:54	00:05:54
6	Dream Theater.3gp	605.166	605.166	00:08:32	00:08:32

Dari tabel 4.2 dapat dilihat bahwa besarnya ukuran *file* mempengaruhi waktu atau lamanya proses enkripsi maupun dekripsi. Dari ke enam video yang di uji, terlihat bahwa semakin besar ukuran *file*, maka semakin banyak waktu yang di perlukan untuk proses tersebut.

Pengujian di lakukan pada video dengan judul "Avril lavigne.3gp" kemudian dilakukan pengamatan, pada video ini tidak di dapati pengurangan kapasitas atau ukuran dari video tersebut. Pengujian kedua dilakukan pada *file* dengan judul "Dream Theater.3gp". Hal yang sama juga terjadi pada *file* tersebut, yaitu tidak di dapati pengurangan ukuran dari *file* asli video tersebut, hanya saja waktu yang di butuhkan

pada saat proses enkripsi dan dekripsi membutuhkan waktu yang lama di bandingkan dengan *file* video yang berukuran kecil.

Tabel 4.3 Hasil uji ukuran *file* enkripsi dan dekripsi

No	Nama file plainteks	Ukuran file plainteks	Ukuran file chiperteks
1	Fish.3gp	3.594	3.594
2	Surga Dalam Bubuk Mesiu.3gp	21.303	21.303
3	Avril Lavigne.3gp	239.570	239.570
4	Green Zone.3gp	339.796	339.796
5	Muse.3gp	413.408	413.408
6	Dream Theater.3gp	605.166	605.166

Algoritma *DES* pada perangkat lunak ini bekerja pada kunci dengan panjang 64 bit. Oleh karena itu, apabila terdapat sisa bit-bit pada *segment bit* yg berukuran kurang dari 64 bit, maka akan langsung ditulis ke *file* tujuan tanpa dilakukan enkripsi. Sehingga ukuran *file* video hasil enkripsi tetap sama dengan ukuran video yang sebenarnya.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah menyelesaikan Rancang Bangun Aplikasi Enkripsi Video 3GP ini, penulis menarik kesimpulan sebagai berikut :

1. Aplikasi ini telah berhasil dibangun dan dapat berfungsi sesuai tujuan, yaitu mengenkripsi *file* video *3gp* sehingga tidak dapat dibaca atau diputar.
2. Aplikasi ini juga telah berhasil mengembalikan *file* video *3gp* yang telah di enkripsi tersebut (*cipherteks*) seperti semula dengan menggunakan kunci yang sama sewaktu enkripsi.
3. Dalam proses enkripsi dan dekripsi tidak di dapati perubahan ukuran asli dari *file* video tersebut.

5.2 Saran

1. Aplikasi enkripsi dan dekripsi audio video menggunakan *VEA* dan *DES* ini masih belum begitu sempurna. Mungkin dalam pengembangannya dapat menggunakan metode penyandian atau komponen dan bahasa pemrograman lain dalam menjaga keamanan *file* video *3gp* yang lebih optimal.
2. Agar aplikasi enkripsi video ini dapat berjalan dengan baik maka pengguna diharapkan dapat mengetahui tatacara penggunaan aplikasi ini dengan baik dan benar.
3. Aplikasi enkripsi video ini dapat di kembangkan lebih jauh lagi karena pembuatannya masih banyak menggunakan batasan masalah. Dengan pertimbangan luasnya sistem dan sumber daya manusia yang menggunakan aplikasi ini.

DAFTAR PUSTAKA

1. Daniel, Socek. *Comparison and Analysis of Selected Video Encryption Algorithms Implemented for MPEG-2 Streams.* Department of Computer Science and Engineering Florida Atlantic University, U.S.A.

2. Anne Ahira, *file 3GP* dalam ponsel *GSM*, www.anneahira.com Indonesia, 2008.

3. Ramsky, Tessa. Perangkat lunak enkripsi video dengan modifikasi *Video Encryption Algorithm*, Institut Teknologi Bandung, 2004.

4. X. Liu and A. M. Eskicioglu, "*Selective Encryption of Multimedia Contents in Distribution Networks: Challenges and New Directions*" IASTED International Conference on Communications, Internet, and Information Technology (CIIT 2003), Scottsdale, AZ, November 17-19,2003.

5. Kilian, Joe and Phillip Rogaway. *How to protect DES against exhaustive key search.* Advances in *Cryptology - Crypto* '96, Springer-Verlag.

6. Dian Dwi Hapsari, Aplikasi video stenography, Universitas Gunadarma, Indonesia.

7. Dian Intania Safitri, Perancangan dan Implementasi Modifikasi Algoritma *VEA*, Institut Teknologi Bandung, 2005.

8. G. Simmons. *Contemporary Cryptology The Science of Information Integrity.* IEEE Press, 1992.

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : ARIYO YOUDO UTOMO
NIM : 06.12.656
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer dan Informatika
MASA BIMBINGAN : 1 Desember s/d 1 Mei 2012
JUDUL : **RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES**

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Rabu
Tanggal : 22 Februari 2012
Dengan Nilai : 73,95 (B+)

PANITIA UJIAN SKRIPSI

KETUA,

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

SEKRETARIS,

Dr. Eng. Aryuanto S, ST, MT
NIP.P.1030800417

ANGGOTA PENGUJI

Dosen Penguji I

Dr.Eng. Aryuanto S, ST. MT
NIP.P.1030800417

Dosen Penguji II

Bima Aulia F, ST
NIP.P.

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

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata Satu (S-1) Jurusan Teknik Elektro Konsentrasi Teknik Komputer dan Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : ARIYO YOUDO UTOMO
NIM : 06.12.656
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer dan Informatika
MASA BIMBINGAN : 1 Desember 2011 s/d 1 Mei 2012
JUDUL : **RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES**

Tanggal	Uraian	Paraf
Penguji I 22 - 02 - 2012	Latar belakang kurang jelas & terfokus	
	Penguasaan Konsep 3Gp, Vea dan Des	
	Penguasaan Java	
	Kesimpulan Di sempurnakan	
Penguji II 22 - 02 - 2012	~~Margin~~ Penulisan	
	Saran nomer 3 di hilangkan	
	Kesimpulan nomer 3 di perbaiki	

Mengetahui,

Dosen Pembimbing I

Irmalia Suryani Faradisa, ST. MT
NIP.P.1030000365

Dosen Pembimbing II

Sotyohadi, ST
NIP.Y.1039700309

Disetujui,

Dosen Penguji I

Dr.Eng. Aryuanto ST
NIP.P.1030800417

Dosen Penguji II

Bima Aulia F, ST
NIP.P.

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura – gura No.2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : ARIYO YOUDO UTOMO
Nim : 06.12.656
Masa Bimbingan : 1 Desember 2011 s/d 1 Mei 2012
Judul Skripsi : RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES

No	Tanggal	Uraian	Paraf Pembimbing
1	6/02 2012	Revisi BAB I, II	
2	8/02 2012	Revisi BAB II	
3	11/02 2012	Revisi BAB III	
4	14/02 2012	Acc BAB I	
5	15/02 2012	Acc BAB II	
6	17/02 2012	Acc BAB III	
7	18/02 2012	.Acc BAB I, BAB, II, BAB III	
8	21/02 2012	Acc BAB 4, BAB V	
9			
10			

Malang,

Dosen Pembimbing I

Irmalia Suryani Faradisa ST. MT
NIP. 1030000365

Form S-4b

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura – gura No.2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : ARIYO YOUDO UTOMO
Nim : 06.12.656
Masa Bimbingan : 1 Desember 2011 s/d 1 Mei 2012
Judul Skripsi : RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES

No	Tanggal	Uraian	Paraf Pembimbing
1	17-02-2012	Revisi Bab 3	
2	20/02/2012	Revisi Bab 4	
3	24/02/2012	ACC Bab 3, Bab 4, Bab 5	
4			
5			
6			
7			
8			
9			
10			

Malang,

Dosen Pembimbing II

Sotyohadi, ST
NIP. Y 1039700309

Form S-4b

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, Km. 2
MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang betanda tangan dibawah ini :

N a m a	: ARIYO YOUDO UTOMO
N I M	: 0912656
Semester	: .....
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: ~~TEKNIK ELEKTRONIKA~~ **TEKNIK ENERGI LISTRIK** **TEKNIK KOMPUTER DAN INFORMATIKA** ~~TEKNIK KOMPUTER~~ ~~TEKNIK TELEKOMUNIKASI~~
Alamat	: .....

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat ***SKRIPSI Tingkat Sarjana.*** Untuk melengkapi permohonan tersebut, bersama kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan-persyaratan pengambilan ***SKRIPSI*** adalah sebagai berikut :

1. Telah melaksanakan semua praktikum sesuai dengan konsentrasinya (.....)
2. Telah lulus dan menyerahkan Laporan Praktek Kerja (.....)
3. Telah lulus seluruh mata kuliah keahlian (MKB) sesuai konsentrasinya (.....)
4. Telah menempuh mata kuliah ≥ 134 sks dengan IPK ≥ 2 dan tidak ada nilai E (.....)
5. Telah mengikuti secara aktif kegiatan seminar skripsi yang diadakan Jurusan (.....)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenaran data tersebut diatas
Recording Teknik Elektro

(.....)

Malang, ...201
Pemohon

(ARIYO YOUDO UTOMO)

Disetujui
Ketua Jurusan Teknik Elektro

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 1018800189

Mengetahui
Dosen Wali

(Ir. YUSUF ISMAIL NAKHODAMT)

Catatan :
Bagi mahasiswa yang telah memenuhi persyaratan mengambil SKRIPSI agar membuat proposal dan mendapat persetujuan dari Ketua Jurusan/Sekretaris Jurusan T. Elektro S-1

1. IPK 3.76 / 136 = 2.76
2. .....
3. - 2 praktikum
   - MK pemrosesan Stn dgt.

Form. S-1a

JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

DAFTAR PRESTASI AKADEMIK PRAKTIKUM
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

Nama Mahasiswa	:	ARIYO YOUDO UTOMO
NIM	:	0612656
Tempat, Tanggal Lahir	:	JAKARTA, 3 NOVEMBER 1986
Jenjang	:	Strata 1 (S1)
Fakultas	:	Teknologi Industri
Jurusan / Program Studi	:	Teknik Elektro
Konsentrasi	:	Teknik Komputer dan Informatika

Praktikum Laboratorium	Kode	Nama Praktikum	SKS	Nilai
I 1.10 17	EL-2215 26	Fisika	1	B+
		Rangkaian Listrik		B
		Rangkaian Logika dan Digital		B
		Dasar Komputer dan Pemrograman		B+
II	EL-4216 1.10	Dasar Elektronika	1	B
		Dasar Sistem Telekomunikasi		B
		Mikrokontroler		B
		Sistem Pengukuran		C+
		Dasar Sistem Kendali		A
III 1.10	EL-5316 28	Basis Data	1	B
		Administrasi Jaringan		B
		Sistem Operasi		B
IV 1.10	EL-6317 29	Pemrograman Internet	1	B
		Pemrograman Objek		B
		Rekayasa Perangkat Lunak Sistem Informasi		B+
V 1.10	EL-7318	Peripheral dan Antar Muka	1	A
		Pemrosesan Sinyal Digital		A
		Multimedia		A
		Pemrograman Jaringan		B+

SS = 29

belum pengajuan TA

Malang, 14-6-2011

Recording
Jurusan Teknik Elektro S1

Puji Handayani

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika/ Teknik Komputer & Informatika*)

1.	Nama Mahasiswa: ARIYO. YOUDO. UTOMO			Nim: 0612656
2.	Keterangan Pelaksanaan	Tanggal: 6 DESEMBER 2011	Waktu:	Tempat Ruang:
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Judul Proposal yang diseminarkan Mahasiswa	RANCANG BANGUN APLIKASI ENKRIPSI VIDEO 3GP MENGGUNAKAN VEA DAN DES		
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian			
6.	Catatan:			
7.	Catatan: Persetujuan Judul Skripsi			
	Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II SAMSON MATOY	Disetujui, Dosen Keahlian III Bima Aulia F	
	Mengetahui, Ketua Jurusan, Ir. Yusuf Ismail Nakhoda,MT NIP.Y. 1018800189	Disetujui, Calon Dosen Pembimbing ybs Pembimbing I	Pembimbing II	

Perhatian:
1. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-3c

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
/*
 *
 *
 *
 */
import java.awt.FileDialog;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
/**
 *
 *
```

```
*/
public class V3GPDes extends javax.swing.JFrame {
    /** Creates new form FrmMain */
    public V3GPDes() {
        initComponents();
        btnSave.setEnabled(false);
        btnProcess.setEnabled(false);
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        pnlMain = new javax.swing.JPanel();
        txtVIdFile = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        btnOpen = new javax.swing.JButton();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jScrollPane1 = new javax.swing.JScrollPane();
```

```
txtlog = new javax.swing.JTextArea();

txtKey = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jToolBar1 = new javax.swing.JToolBar();

bNew = new javax.swing.JButton();

btnSave = new javax.swing.JButton();

btnProcess = new javax.swing.JButton();

btnExit = new javax.swing.JButton();

jMenuBar1 = new javax.swing.JMenuBar();

jMenu2 = new javax.swing.JMenu();

jMenu1 = new javax.swing.JMenu();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setTitle("Enkripsi Video 3gp");

setAutoRequestFocus(false);

setName("FrmMain"); // NOI18N

addWindowListener(new java.awt.event.WindowAdapter() {

   public void windowActivated(java.awt.event.WindowEvent evt) {

      formWindowActivated(evt);

   }

});

addComponentListener(new java.awt.event.ComponentAdapter() {

   public void componentResized(java.awt.event.ComponentEvent evt) {
```

```
        formComponentResized(evt);
    }
});
addWindowStateListener(new java.awt.event.WindowStateListener() {
    public void windowStateChanged(java.awt.event.WindowEvent evt) {
        formWindowStateChanged(evt);
    }
});
addPropertyChangeListener(new java.beans.PropertyChangeListener() {
    public void propertyChange(java.beans.PropertyChangeEvent evt) {
        formPropertyChange(evt);
    }
});

pnlMain.setBackground(new java.awt.Color(255, 255, 255));
pnlMain.setBorder(javax.swing.BorderFactory.createEtchedBorder());

txtVIdFile.setEditable(false);

jLabel1.setText("File :");

btnOpen.setBackground(new java.awt.Color(255, 255, 255));
```

```
        btnOpen.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/open_document_24.png"))); //
NOI18N
        btnOpen.setText("Add Video");
        btnOpen.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        btnOpen.setFocusPainted(false);
        btnOpen.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOpenActionPerformed(evt);
            }
        });

        jTabbedPane1.setBackground(new java.awt.Color(237, 235, 231));
        jTabbedPane1.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1,
1));

        txtlog.setColumns(20);
        txtlog.setRows(5);
        jScrollPane1.setViewportView(txtlog);

        jTabbedPane1.addTab("Log", jScrollPane1);

        jLabel2.setText("Key :");
```

```
        javax.swing.GroupLayout pnlMainLayout = new
javax.swing.GroupLayout(pnlMain);

        pnlMain.setLayout(pnlMainLayout);

        pnlMainLayout.setHorizontalGroup(


pnlMainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(pnlMainLayout.createSequentialGroup()

                .addContainerGap()

                .addComponent(jLabel1)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(txtVIdFile, javax.swing.GroupLayout.DEFAULT_SIZE,
451, Short.MAX_VALUE)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

                .addComponent(btnOpen)

                .addContainerGap())

            .addComponent(jTabbedPane1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 576, Short.MAX_VALUE)

            .addGroup(pnlMainLayout.createSequentialGroup()

                .addContainerGap()

                .addComponent(jLabel2)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(txtKey, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(416, Short.MAX_VALUE))
        );
        pnlMainLayout.setVerticalGroup(
pnlMainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(pnlMainLayout.createSequentialGroup()
                .addContainerGap()
.addGroup(pnlMainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.T
RAILING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
20, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(txtVIdFile,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnOpen))
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addGroup(pnlMainLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.T
RAILING)
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
20, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
            .addComponent(txtKey, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 14,
Short.MAX_VALUE)

            .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 361,
javax.swing.GroupLayout.PREFERRED_SIZE))

    );


    jToolBar1.setBackground(new java.awt.Color(237, 235, 231));

    jToolBar1.setRollover(true);


    bNew.setBackground(new java.awt.Color(237, 235, 231));

    bNew.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/redo_48.png"))); // NOI18N

    bNew.setText("Baru");

    bNew.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));

    bNew.setFocusPainted(false);

    bNew.addActionListener(new java.awt.event.ActionListener() {

      public void actionPerformed(java.awt.event.ActionEvent evt) {

        bNewActionPerformed(evt);

      }

    });
```

```
        jToolBar1.add(bNew);


        btnSave.setBackground(new java.awt.Color(237, 235, 231));
        btnSave.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/save_48.png"))); // NOI18N
        btnSave.setText("Simpan");
        btnSave.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
        btnSave.setFocusPainted(false);
        btnSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnSaveActionPerformed(evt);
            }
        });
        jToolBar1.add(btnSave);


        btnProcess.setBackground(new java.awt.Color(237, 235, 231));
        btnProcess.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/refresh_document_48.png")));
// NOI18N
        btnProcess.setText("Proses");
        btnProcess.setFocusPainted(false);
        btnProcess.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnProcessActionPerformed(evt);
```

```
        }
    });
    jToolBar1.add(btnProcess);

    btnExit.setBackground(new java.awt.Color(237, 235, 231));
    btnExit.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/delete_48.png"))); // NOI18N
    btnExit.setText("Keluar");
    btnExit.setBorder(javax.swing.BorderFactory.createEmptyBorder(1, 1, 1, 1));
    btnExit.setFocusPainted(false);
    btnExit.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            btnExitActionPerformed(evt);
        }
    });
    jToolBar1.add(btnExit);

    jMenu2.setText("Bantuan");

    jMenu1.setText("Tentang Saya");
    jMenu2.add(jMenu1);

    jMenuBar1.add(jMenu2);
```

```
        setJMenuBar(jMenuBar1);


        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jToolBar1, javax.swing.GroupLayout.DEFAULT_SIZE, 580,
Short.MAX_VALUE)
            .addComponent(pnlMain, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jToolBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0)
                .addComponent(pnlMain, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
```

```
        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-596)/2, (screenSize.height-527)/2, 596, 527);
    }// </editor-fold>
    private BufferedImage imageData;



    Integer Maxdata;
    public void fillLog(String isi){
        txtlog.append(isi+'\n');
    }
    private void btnOpenActionPerformed(java.awt.event.ActionEvent evt) {
     String dirc = System.getProperty("user.dir");



     btnSave.setEnabled(false);


     JFileChooser fc = new JFileChooser(dirc);
     fc.setFileFilter(new javax.swing.filechooser.FileFilter()
       {

            public final  String gpFile = "3gp";
            public final  String gvFile = "3gv";
```

```
public String getDescription() {
   return "Add 3gp Video (*.3gp;*.3vd)";
}

public boolean accept(java.io.File f) {
        if (f.isDirectory()) {
           return true;
        }

        String extension = getExtension(f);
        if (extension != null) {
           if ( extension.equals(gpFile) || extension.equals(gvFile)) {
                return true;
           } else {
              return false;
           }
        }

        return false;
     }
```

```
            }
        );


    int rc = fc.showDialog(null, "Select Video File");
    if (rc == JFileChooser.APPROVE_OPTION)
    {
      File file = fc.getSelectedFile();
      if (this.getExtension(file).equalsIgnoreCase("3gp") ||
this.getExtension(file).equalsIgnoreCase("3gv")    ){
        this.txtVIdFile.setText( file.getAbsolutePath());


        try {


          btnProcess.setEnabled(true);


        } catch ( Exception e) {
          btnProcess.setEnabled(false);
          JOptionPane.showMessageDialog(rootPane,"Load Video Failed");
        }
```

```
    }else{
      btnProcess.setEnabled(false);
      this.txtVIdFile.setText("");
      JOptionPane.showMessageDialog(rootPane,  "Choose Video File..");



    }
    btnSave.setEnabled(false);
  }
new File(System.getProperty("user.dir")+"\\temp.3gp").delete();
new File(System.getProperty("user.dir")+"\\temp.3gv").delete();

}

private void bNewActionPerformed(java.awt.event.ActionEvent evt) {
   // TODO add your handling code here:
    txtVIdFile.setText(null);
    // new proses
    txtlog.setText("");
    btnSave.setEnabled(false);
    btnProcess.setEnabled(false);
    new File(System.getProperty("user.dir")+"\\temp.3gv").delete();
```

```
        new File(System.getProperty("user.dir")+"\\temp.3gp").delete();
        jTabbedPane1.setSelectedIndex(0);



    }

    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        if ((JOptionPane.showConfirmDialog(rootPane, "Are You Sure To
Exit?","Confirm.",JOptionPane.YES_NO_OPTION))==JOptionPane.YES_OPTION){
            System.exit(0);
        }
    }

    private void btnSaveActionPerformed(java.awt.event.ActionEvent evt) {
     if (txtVIdFile.getText().endsWith(".3gp")) {



            String files =null;
            FileDialog fd=null;
            //call save dialog
            fd  = new FileDialog(this,"Save to File ",FileDialog.SAVE);
            fd.setFile("*.3gv");
```

```
        fd.setDirectory(".");
        fd.show();

        files = fd.getDirectory()+fd.getFile();
        if (!files.equalsIgnoreCase(null) ){
          //remove original file
          new File(txtVIdFile.getText()).delete();
          //create new file
           copyfile(System.getProperty("user.dir")+"\\temp.3gv",
fd.getDirectory()+fd.getFile() );


        }
      }else{
        String files =null;
        FileDialog fd=null;
        //call save dialog
        fd  = new FileDialog(this,"Save to File ",FileDialog.SAVE);
        fd.setFile("*.3gp");

        fd.setDirectory(".");
        fd.show();
```

```
        files = fd.getDirectory()+fd.getFile();
        if (!files.equalsIgnoreCase(null) ){
           //remove original file
            new File(txtVIdFile.getText()).delete();
            //create new file
            copyfile(System.getProperty("user.dir")+"\\temp.3gp",
fd.getDirectory()+fd.getFile() );


        }
      }
   }

   private void formPropertyChange(java.beans.PropertyChangeEvent evt) {


   }

   private void formWindowStateChanged(java.awt.event.WindowEvent evt) {


   }
```

```
    private void formWindowActivated(java.awt.event.WindowEvent evt) {

    }

    private void formComponentResized(java.awt.event.ComponentEvent evt) {

    }

    private void btnProcessActionPerformed(java.awt.event.ActionEvent evt) {
      try {

        if (txtKey.getText().trim().length()==0) return;

        if (txtVIdFile.getText().endsWith(".3gp")){

            new File(System.getProperty("user.dir")+"\\temp.3gp").delete();
            new File(System.getProperty("user.dir")+"\\temp.3gv").delete();
            copyfile(txtVIdFile.getText() ,
System.getProperty("user.dir")+"\\temp.3gp");

            //call decript method
            DESencrypt  encrypter = new DESencrypt();
            encrypter.EncriptFile(  System.getProperty("user.dir")+"\\temp.3gp",
```

```
            System.getProperty("user.dir")+"\\temp.3gv" ,txtKey.getText().toString());


        }else{
            new File(System.getProperty("user.dir")+"\\temp.3gp").delete();
            new File(System.getProperty("user.dir")+"\\temp.3gv").delete();
            //call decript method
            DESdecript   encrypter = new DESdecript() ;
            copyfile(txtVIdFile.getText() ,
System.getProperty("user.dir")+"\\temp.3gv");
            encrypter.DecriptFile( System.getProperty("user.dir")+"\\temp.3gv" ,
              System.getProperty("user.dir")+"\\temp.3gp" ,txtKey.getText().toString());
            //get marker from file
            FileInputStream resFile = new
FileInputStream(System.getProperty("user.dir")+"\\temp.3gp" );
            int a = ((int) resFile.read());
            int b = (int) resFile.read();copyfile(txtVIdFile.getText() ,
System.getProperty("user.dir")+"\\temp.3gv");
            int c = ((int) resFile.read());
            //check marker
             if (a != 0 || b != 0 || c != 0  ){
               JOptionPane.showMessageDialog(this, "Proses Gagal, Key Salah...!!!",
"Error",JOptionPane.ERROR_MESSAGE);
               System.out.println("Wrong Key");
               new File(System.getProperty("user.dir")+"\\temp.3gv").delete();
```

```
                new File(System.getProperty("user.dir")+"\\temp.3gp").delete();
                return;
            }

        }
        JOptionPane.showMessageDialog(this, "Proses Berhasil",
"Informasi",JOptionPane.INFORMATION_MESSAGE);
        btnSave.setEnabled(true);

    } catch (Exception e) {
        JOptionPane.showMessageDialog(this, "Proses Gagal",
"Error",JOptionPane.ERROR_MESSAGE);
        return;
    }

}

/**
* @param args the command line arguments
*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new V3GPDes().setVisible(true);
```

```
        }
    });


}

// Variables declaration - do not modify
private javax.swing.JButton bNew;
private javax.swing.JButton btnExit;
private javax.swing.JButton btnOpen;
private javax.swing.JButton btnProcess;
private javax.swing.JButton btnSave;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JToolBar jToolBar1;
private javax.swing.JPanel pnlMain;
public javax.swing.JTextField txtKey;
public javax.swing.JTextField txtVIdFile;
```

```
private javax.swing.JTextArea txtlog;
// End of variables declaration


public  String getExtension(File f) {
   String ext = null;
   String s = f.getName();
   int i = s.lastIndexOf('.');


   if (i > 0 &&  i < s.length() - 1) {
      ext = s.substring(i+1).toLowerCase();
   }
   return ext;
}
private static void copyfile(String srFile, String dtFile){
      try
   {

            File f1 = new File(srFile);
            File f2 = new File(dtFile);
            InputStream in = new FileInputStream(f1);



            OutputStream out = new FileOutputStream(f2);
```

```
            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) > 0){
                out.write(buf, 0, len);
            }
            in.close();
            out.close();
            System.out.println("File Copied.");
        }catch(FileNotFoundException ex){
            System.out.println(ex.getMessage() + " in the specified directory.");
            System.exit(0);
        }
            catch(IOException e){
            System.out.println(e.getMessage());
        }
    }
    private static void SetTipe(String srFile, String dtFile){
        try
      {
            File f1 = new File(srFile);
            File f2 = new File(dtFile);
            InputStream in = new FileInputStream(f1);
```

```
            OutputStream out = new FileOutputStream(f2);


            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) > 0){
                    out.write(buf, 0, len);
            }


            in.close();
            out.close();
            System.out.println("Operation completed.");
    }catch(FileNotFoundException ex){
            System.out.println(ex.getMessage() + " in the specified directory.");
            System.exit(0);
    }
            catch(IOException e){
            System.out.println(e.getMessage());
    }
}
```