IMPLEMENTASI ALGORITMA RC6 UNTUK PROTEKSI FILE MP3

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Strata Satu (S-1)

Disusun oleh:

NILUH KADEK KURNIA DEWI

NIM. 0610630072 - 63

KEMENTERIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2013

Pengantar

Puji syukur kepada Allah Tuhan YME yang telah memberikan segala nikmat dan rahmat-Nya sehingga penyusunan skripsi ini dapat diselesaikan. Karena hanya dengan pertolongan-Nya semata penulis mampu melewati segala kendala yang ada selama penyusunan skripsi ini.

Skripsi berjudul " Implementasi Algoritma RC6 untuk Proteksi File MP3" ini disusun sebagai salah satu syarat untuk mendapatkan gelar Sarjana Teknik di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Terselesaikannya skripsi ini tentunya tidak lepas juga dari bantuan berbagai pihak. Oleh sebab itu, dengan segala kerendahan hati penulis menyampaikan terima kasih kepada:

1. Bapak Dr. Ir. Soleh Hadi Pramono, M.S. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
2. Bapak M. Azis Muslim, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Muhammad Aswin, Ir., MT. selaku dosen pembimbing I yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.
5. Bapak Waru Djuriatno ST., MT. selaku dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.
6. Ibu Endah Budi Purnomowati, Ir., MT. selaku dosen pendamping akademik.
7. Bapak dan Ibu Dosen serta karyawan jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
8. Orang tua, kakak dan saudara-saudara terdekat yang telah memberikan dukungan serta doa dan semangat.
9. Ridwan Setiawan yang selalu mendampingi serta memberikan dukungan, doa dan semangat.

10. Teman-teman Ge-Force 2006 atas kebersamaan dalam suka maupun duka selama ini. Serta teman-teman di Ristie-HME periode kepengurusan 08/09 – 09/10.
11. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Penulis menyadari sepenuhnya bahwa skripsi ini masih banyak kekurangan. Segala kritik dan saran yang membangun akan penulis terima demi kesempurnaan skripsi ini. Harapan penulis semoga skripsi ini bermanfaat bagi pembaca dan khususnya mahasiswa jurusan teknik elektro dimasa yang akan datang.

Malang, Juli 2013

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Niluh Kadek Kurnia Dewi, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2013, *Implementasi Algoritma RC6 untuk Proteksi File MP3*, Dosen Pembimbing: Ir. Muhammad Aswin, MT. dan Waru Djuriatno, ST., MT.

Penggunaan *file* berupa *audio digital* berformat MP3 saat ini cukup populer dan mudah untuk dinikmati. *File* MP3 selain memberi kemudahan dalam penyebaran, juga memberi kemudahan dalam penggandaan yang kemudian dapat digunakan secara negatif tanpa memperhatikan aspek hak cipta. Skripsi ini membahas perancangan aplikasi untuk memproteksi *file* MP3 yaitu dengan enkripsi. *File* MP3 yang telah terenkripsi tidak dapat diputar/dimainkan dengan sempurna tanpa menggunakan aplikasi tertentu yang telah terdapat suatu cara untuk mendekripsikannya. Aplikasi dirancang dengan menggunakan algoritma RC6 untuk melakukan enkripsi dan dekripsi agar keamanan dapat ditingkatkan. Algoritma RC6 adalah suatu algoritma kunci privat yang dikenal dengan kesederhanaannya. Algoritma RC6 merupakan algoritma dengan parameter yang dapat bekerja pada panjang kunci yang beragam. Untuk aspek keamanannya, algoritma RC6 mengutamakan prinsip iterated cipher.

Kata Kunci—MP3, Enkripsi, Dekripsi.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Multimedia dapat diartikan sebagai teknologi yang menggabungkan berbagai sumber media (teks, grafik dan suara) untuk menyampaikan atau membuat sesuatu sebagai perantara atau suatu bentuk komunikasi. Multimedia seringkali digunakan dalam dunia hiburan. Salah satunya adalah penggunaan *file* berupa *audio digital* yang saat ini cukup populer dan mudah untuk dinikmati, yaitu *file* berformat MP3.

File MP3 selain memberi kemudahan dalam penyebaran, juga memberi kemudahan dalam penggandaan. Kemudahan tersebut akhirnya dapat digunakan secara negatif tanpa memperhatikan aspek hak cipta. *File* MP3 yang seharusnya menjadi properti legal dari produsen dan secara legal dimiliki oleh orang yang telah membelinya bisa dengan mudah disalahgunakan oleh pihak-pihak yang tidak bertanggung jawab. Untuk itu diperlukan suatu cara untuk memproteksi *file* tersebut, salah satunya adalah dengan enkripsi. *File* MP3 yang telah terenkripsi tidak dapat diputar/dimainkan dengan sempurna sehingga diperlukan aplikasi untuk mendekripsi *file* tersebut agar dapat dimainkan seperti semula.

Berdasarkan permasalahan tersebut, maka dibuat sebuah aplikasi yang dapat mengenkripsi *file* MP3 sekaligus mampu mendekripsi *file* MP3 terproteksi tersebut yang dalam prosesnya akan diterapkan algoritma kriptografi RC6. Untuk itu dalam Tugas Akhir ini, penulis mengangkat judul "Implementasi Algoritma RC6 untuk Proteksi *File* MP3".

1.2. Perumusan Masalah

Berdasarkan pada latar belakang pengambilan judul, maka yang menjadi masalah dalam Tugas Akhir ini adalah bagaimana merancang aplikasi yang dapat mengenkripsi *file* MP3 sehingga masih dapat diputar/dimainkan tetapi tidak dengan sempurna dan mendekripsi *file* yang telah terenkripsi tersebut dengan menggunakan algoritma RC6.

1.3. Pembatasan Masalah

Beberapa hal yang menjadi batasan masalah dalam pembuatan aplikasi ini antara lain.

1. Format *audio file* yang diujicobakan adalah MPEG-1 layer 3 (MP3).
2. Pembahasan difokuskan pada pembuatan aplikasi untuk enkripsi dan dekripsi *file* MP3.
3. Kunci di-*input*-kan oleh *user* secara manual dengan panjang maksimal 256 *byte*.
4. Pengujian dilakukan dengan menggunakan Windows Media Player.

1.4. Tujuan Penulisan

Tujuan penulisan Tugas Akhir ini adalah untuk merancang aplikasi yang dapat memproteksi *file* MP3 dengan input *file* MP3 dan player *file* MP3 terproteksi.

1.5 Manfaat

Adapun manfaat yang diperoleh dalam pengerjaan skripsi ini antara lain:

a) Bagi Penyusun

1. Memperoleh pemahaman mengenai kelebihan serta kekurangan aplikasi yang telah dibuat.
2. Menambah wawasan ilmu pengetahuan yang telah dipelajari sebelumnya, dan serta sebagai pelatihan berpikir kritis dalam menyelesaikan suatu masalah yang dihadapi.
3. Memberikan sumbangan bagi perkembangan ilmu pengetahuan khususnya dalam bidang Teknologi Informatika.

b) Bagi Pengguna

1. Memberikan solusi mengenai masalah penduplikasian file MP3 secara legal.

1.6 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Dalam bab ini akan dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat dan sistematika penulisan laporan tugas akhir.

BAB II Tinjauan Pustaka

Dalam bab ini akan dibahas dan dijelaskan mengenai dasar teoritis yang menjadi landasan dan mendukung pelaksanaan penulisan tugas akhir.

BAB III Metodologi Penelitian

Dalam bab ini akan membahas tentang metode yang dipakai penulis untuk menyelesaikan laporan tugas akhir.

BAB IV Perancangan dan Implementasi

Dalam bab ini menjelaskan langkah - langkah perancangan dan implementasi RC6 untuk proteksi *file* MP3.

BAB V Pengujian

Dalam bab ini akan disampaikan hasil pengujian dari aplikasi yang telah dibuat.

BAB VI Penutup

Dalam bab ini akan disampaikan kesimpulan dan saran.

BAB II
TINJAUAN PUSTAKA

2.1. Kriptografi

Kriptografi berasal dari bahasa Yunani, *crypto* dan *graphia*. *Crypto* artinya *secret* (rahasia) dan *graphia* berarti *writing* (tulisan). Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain (Ariyus, 2008).

Beberapa istilah yang terdapat dalam kriptografi.

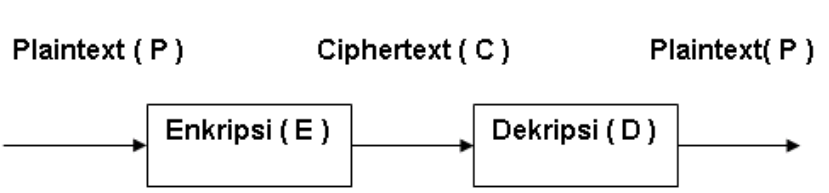
1. Pesan, *plaintext* dan *ciphertext.*

Pesan (*message*) adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan adalah *plaintext* atau teks-jelas (*cleartext*). Agar pesan tidak dapat dimengerti maknanya oleh pihak lain, maka pesan perlu disandikan ke bentuk lain yang tidak dapat dipahami. Bentuk pesan yang tersandi disebut *ciphertext. Ciphertext* harus dapat ditransformasikan kembali menjadi *plaintext* semula agar pesan yang diterima bisa dibaca.

2. Enkripsi dan dekripsi.

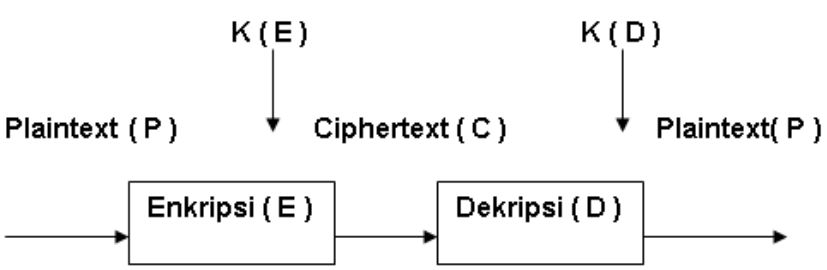
Enkripsi yaitu suatu proses pengamanan suatu data yang disembunyikan atau proses konversi data (*plaintext*) menjadi bentuk yang tidak dapat dibaca/dimengerti. Sedangkan dekripsi adalah kebalikan dari proses enkripsi yaitu proses konversi data yang sudah dienkripsi (*ciphertext*) kembali menjadi data aslinya (*Original Plaintext*) sehingga dapat dibaca/ dimengerti kembali.

Pesan yang akan dienkripsi disebut *plaintext* yang dimisalkan *plaintext* (P), proses enkripsi dimisalkan enkripsi (E), proses dekripsi dimisalkan dekripsi (D), dan pesan yang sudah dienkripsi disebut *ciphertext* yang dimisalkan *ciphertext* (C) maka dapat digambarkan pada gambar berikut ini :



Gambar 2.1. Proses Enkripsi dan Dekripsi
Sumber: www.agungnugroho.net/?p=28

Data atau informasi yang akan dienkripsi (*plaintext*) diacak oleh suatu kunci yang telah ditentukan kemudian *output* dari proses enkripsi (*ciphertext*) dikembalikan kebentuk aslinya oleh sebuah kunci yang sama.



Gambar 2.2. Proses Enkripsi dan Dekripsi dengan kunci K

Sumber: http://www.agungnugroho.net/?p=28

Fungsi enkripsi E dioperasikan dengan P kemudian menghasilkan C, yang digambarkan seperti notasi berikut :

$$E (P) = C$$

Pada proses dekripsi data yang sudah diproses pada enkripsi (*ciphertext*) melalui proses dekripsi data akan dikembalikan lagi ke dalam bentuk plaintext/ data aslinya, yang digambarkan seperti notasi berikut :

$$D (C) = P$$

Data atau informasi yang telah melalui proses enkripsi dan dekripsi, dimana data yang sudah diacak akan menghasilkan data atau informasi aslinya (*plaintext*), yang digambarkan seperti notasi berikut :

$$D (E (P)) = P$$

3. *Cipher* dan kunci

Algoritma kriptografi disebut juga *cipher* yaitu aturan untuk *enciphering* dan *deciphering* atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi.

Algoritma enkripsi digunakan pada saat melakukan proses enkripsi terhadap suatu plaintext dan algoritma dekripsi digunakan pada saat melakukan proses dekripsi terhadap suatu ciphertext.

Sedangkan dalam penerapannya algoritma enkripsi dan algoritma dekripsi harus menggunakan kunci untuk membuka dan menutup sandinya, hal ini untuk

menjaga keamanan data atau informasi tersebut. Kunci yang dimaksud dapat dilambangkan dengan K.

Kunci yang digunakan dapat berupa sebuah angka bernilai kecil atau besar sesuai dengan angka-angka yang telah ditentukan untuk sebagai nilai transformasi matematis yang memetakan plaintext ke ciphertext dan sebaliknya. Ciphertext sangat dipengaruhi oleh keberadaan plaintext dan kuncinya, jadi nilai dari suatu kunci akan mempengaruhi fungsi enkripsi dan dekripsi, sehingga fungsi enkripsi tersebut dapat dinotasikan seperti berikut :

$$Ek (P) = C$$

Bila kunci yang dipakai untuk proses enkripsi sama dengan kunci yang dipakai untuk proses dekripsi, maka dapat digambarkan dengan notasi sebagai berikut :

$$Dk (Ek) = P$$

Keterangan :

K : Kunci

Ek : Kunci Enkripsi

Dk : Kunci Dekripsi

Konsep dasar inilah yang dipergunakan untuk teknik enkripsi dan dekripsi untuk menjaga Keamanan data dari pihak yang tidak bertanggung jawab atau pihak yang tidak berkepentingan (Nugroho, 2009).

2.1.1. Algoritma Kriptografi

Definisi terminologi algoritma adalah urutan langkah-langkah logis untuk menyelesaikan masalah yang disusun secara sistematis. Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut (Ariyus, 2008).

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya.

1. Algoritma simetri

Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Bila mengirim pesan dengan algoritma ini, si penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendeskripsikan pesan yang dikirim. Keamanan pesan yang menggunakan

algoritma ini tergantung pada kunci. Algoritma yang memakai kunci simetri diantaranya adalah *Data Encryption Standard* (DES), RC4, RC5, RC6, *International Data Encryption Algorithm* (IDEA), *Advanced Encryption Standard* (AES) dan sebagainya.

2. Algoritma asimetri

Algoritma asimetri sering juga disebut dengan algoritma kunci publik dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu kunci umum (*public key*) dan kunci rahasia (*private key*). Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci publik orang dapat mengenkripsi pesan tetapi tidak dapat mendekripsikannya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut.

3. Hash *function*

Fungsi Hash sering disebut dengan fungsi Hash satu arah yang merupakan fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap.

2.1.2. Algoritma RC6

RC6 merupakan salah satu dari algoritma simetri kriptografi yaitu algoritma yang menggunakan satu kunci untuk enkripsi dan dekripsinya. RC6 adalah algoritma blok kode yang sangat aman, padat, sederhana dan menawarkan performansi yang sangat bagus dan fleksibel, dikembangkan dari algoritma RC5. Parameter dari algoritma ini adalah ukuran blok, ukuran kunci eksternal dan jumlah putaran yang bervariasi dengan batasan sama seperti pada RC5 (Ariyus, 2008).

Algoritma RC6 adalah versi yang dilengkapi dengan beberapa parameter, sehingga dituliskan sebagai RC6-w/r/b, dimana parameter w merupakan ukuran kata dalam satuan bit, r adalah bilangan bulat bukan negatif yang menunjukan banyaknya iterasi selam proses enkripsi, dan b menunjukan ukuran kunci enkripsi dalam *byte*. Ketika algoritma ini masuk sebagai kandidat AES, maka ditetapkan

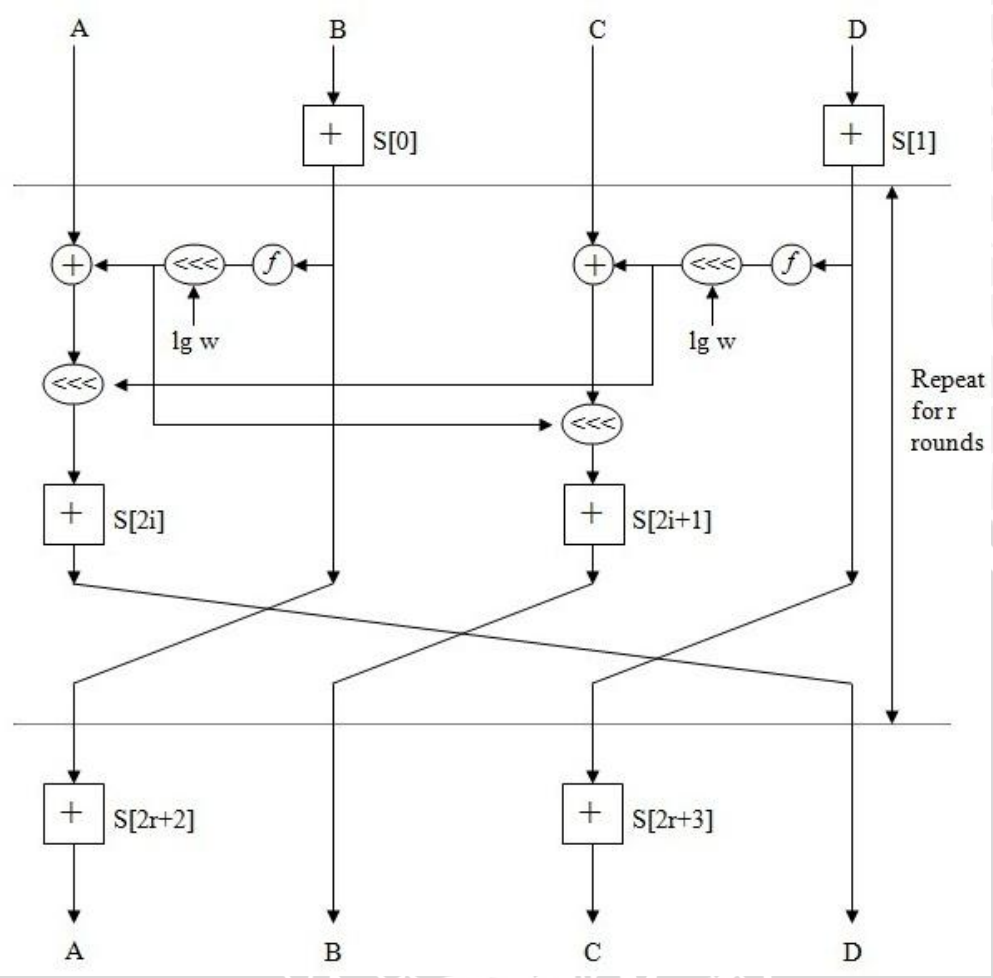
nilai parameter w = 32, r = 20 dan b bervariasi antara 16, 24, dan 32 *byte* (Abdurohman, 2002).

RC6-w/r/b memecah *block* 128 bit menjadi 4 buah *block* 32 bit, dan mengikuti enam aturan operasi dasar sebagai berikut :

A + B	Operasi penjumlahan bilangan integer.
A – B	Operasi pengurangan bilangan integer.
A ⊕ B	Operasi *exclusive*-OR (XOR)
A x B	Operasi perkalian bilangan integer.
A<<<B	A dirotasikan ke kiri sebanyak variabel kedua (B)
A>>>B	A dirotasikan ke kanani sebanyak variabel kedua (B)

2.1.2.1. Algoritma Enkripsi RC6

Karena RC6 memecah blok 128 bit menjadi 4 buah blok 32 bit, maka algoritma ini bekerja dengan 4 buah register 32-bit A, B, C, D. *Byte* yang pertama dari plaintext atau ciphertext ditempatkan pada *byte* A, sedangkan *byte* yang terakhirnya ditempatkan pada *byte* D. Dalam prosesnya akan didapatkan (A, B, C, D) = (B, C, D, A) yang diartikan bahwa nilai yang terletak pada sisi kanan berasal dari register disisi kiri. Diagram blok berikut akan lebih menjelaskan proses enkripsi yang terjadi pada algoritma RC6 :



Gambar 2.3. Proses Algoritma Enkripsi RC6

Sumber: Maman Abdurohman, Analisis Performansi Algoritma Kriptografi RC6. 2002

Algoritma RC6 menggunakan 44 buah sub kunci yang dibangkitkan dari kunci dan dinamakan dengan S[0] hingga S[43]. Masing-masing sub kunci panjangnya 32 bit. Proses enkripsi pada algoritma RC6 dimulai dan diakhiri dengan proses *whitening* yang bertujuan untuk menyamarkan iterasi yang pertama dan yang terakhir dari proses enkripsi dan dekripsi. Pada proses whitening awal, nilai B akan dijumlahkan dengan S[0], dan nilai D dijumlahkan dengan S[i]. Pada masing-masing iterasi pada RC6 menggunakan 2 buah sub kunci. Sub kunci pada iterasi yang pertama menggunakan S[2] dan S[3], sedangkan iterasi-iterasi berikutnya menggunakan sub-sub kunci lanjutannya. Setelah iterasi ke-20 selesai,

dilakukan proses whitening akhir dimana nilai A dijumlahkan dengan S[42], dan nilai C dijumlahkan dengan S[43].

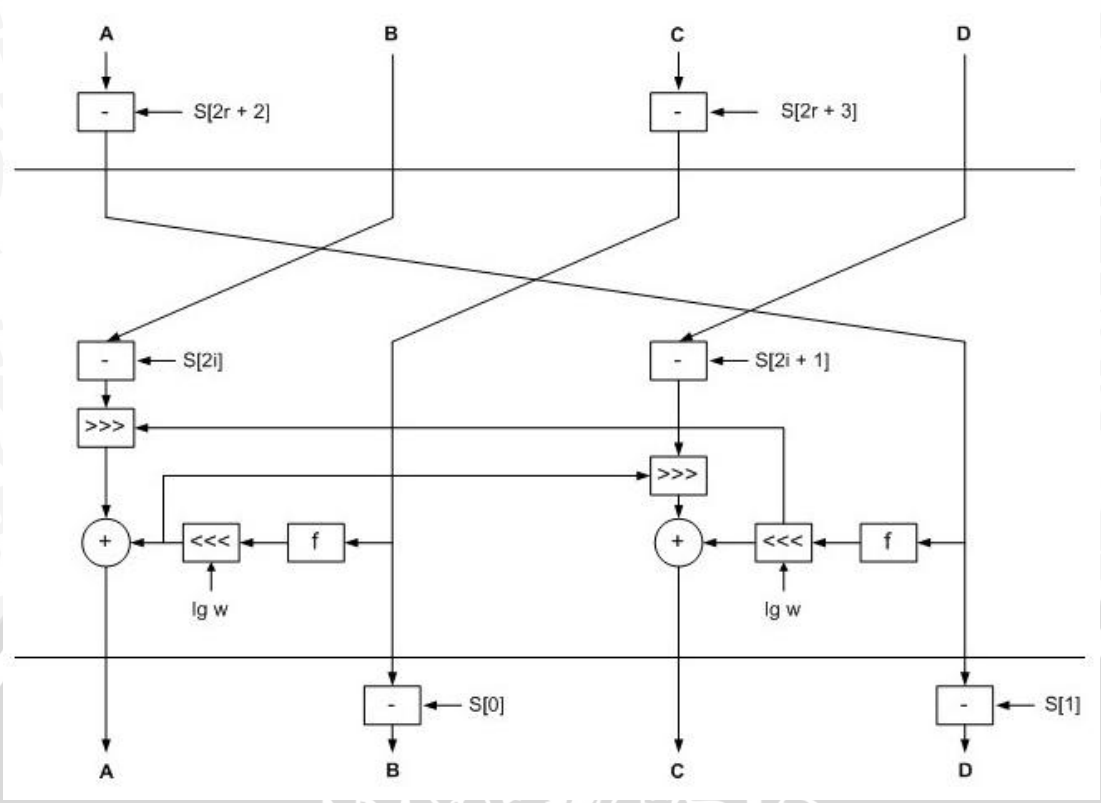
Setiap iterasi pada algoritma RC6 mengikuti aturan sebagai berikut, nilai B dimasukan ke dalam fungsi f, yang didefinisikan sebagai f(x) =x(2x+1) , kemudian diputar kekiri sejauh lg-w atau 5 bit. Hasil yang didapat pada proses ini dimisalkan sebagai u. Nilai u kemudian di XOR dengan C dan hasilnya menjadi nilai C.

Nilai t juga digunakan sebagai acuan bagi C untuk memutar nilainya kekiri. Begitu pula dengan nilai u, juga digunakan sebagai acuan bagi nilai A untuk melakukan proses pemutaran kekiri.

Kemudian sub kunci S[2i] pada iterasi dijumlahkan dengan A, dan sub kunci S[2i+1] dijumlahkan dengan C. keempat bagian dari blok kemudian akan dipertukarkan dengan mengikuti aturan, bahwa nilai A ditempakan pada D, nilai B ditempatkan pada A, nilai C ditempatkan apda B, dan nilai (asli) D ditempatkan pada C. demikian iterasi tersebut akan terus berlangsung hingga 20 kali (Abdurohman, 2002).

2.1.2.2. Algoritma Dekripsi RC6

Proses dekripsi ciphertext pada algoritma RC6 merupakan pembalikan dari proses enkripsi. Pada proses whitening, bila proses enkripsi menggunakan operasi penjumlahan, maka pada proses dekripsi menggunakan operasi pengurangan. Sub kunci yang digunakan pada proses whiteneing setelah iterasi terakhir diterapkan sebelum iteasi pertama, begitu juga sebaliknya sub kunci yang diterapkan pada proses whitening sebelum iterasi pertama digunakan pada whitening setelah iterasi terkahir. Akibatnya, untuk melakukan dekripsi, hal yang harus dilakukan semata-mata hanyalah menerapkan algoritma yang sama dengan enkripsi, dengan tiap iterasi menggunakan sub kunci yang sama dengan yang digunakan pada saat enkripsi, hanya saja urutan sub kunci yang digunakan terbalik (Abdurohman, 2002).



Gambar 2.4. Proses Algoritma Dekripsi RC6

Sumber: http://ilhuna.wordpress.com/tag/dekripsi-rc6/

2.1.2.3. Pembangkitan Kunci (*Key Expansion*)

Pengguna memasukkan sebuah kunci yang besarnya b *byte*, dimana 0≤b≤255. *byte* kunci ini kemudian ditempatkan dalam array c w-bit words L[0]…L[c-1]. *Byte* pertama kunci akan ditempatkan sebagai pada L[0], *byte* kedua pada L[1], dan seterusnya. (Catatan, bila b=0 maka c=1 dan L[0]=0). Masing-masing nilai kata w-bit akan dibangkitkan pada penambahan kunci round 2r+4 dan akan ditempatkan pada array S[0,…,2r+3].

Konstanta P32 = B7E15163 dan Q32 = 9E3779B9 (dalam satuan heksadesimal) adalah "konstanta ajaib"yang digunakan dalam penjadwalan kunci pada RC6. nilai P32 diperoleh dari perluasan bilangan biner e-2, dimana e adalah sebuah fungsi logaritma. Sedangkan nilai Q32 diperoleh dari perluasan bilangan biner ∅-1, dimana ∅ dapat dikatakan sebagai "golden ratio" (rasio emas) (Abdurohman, 2002).

2.2. MPEG-1 Layer 3

MPEG-1 Audio Layer 3 atau lebih dikenal sebagai MP3 dari singkatannya *Moving Picture Experts Group Audio Layer*-3 yang bila diartikan yaitu lapisan ketiga file audio MPEG. Secara keseluruhan definisi MP3 yaitu *file audio* yang terkompresi dengan menggunakan MPEG versi 1 dari lapisan suara yang ke - 3 berkualitas suara seperti CD Audio dan mempunyai kapasitas yang kecil. MPEG itu sendiri adalah sebuah organisasi internasional yang terdiri dari ahli-ahli pengkompres file.

MP3 pertama kali muncul pada pertengahan tahun 1987. Ketika itu, IIS (sebuah lembaga penelitian) beserta Thomson Multimedia memegang hak paten MP3 - melakukan kerja sama dengan *University of Erlangen* dalam mengembangkan sebuah proyek yang meneliti suatu formula matematis yang memotong file digital audio tanpa harus kehilangan sejumlah bagian atau informasi penting yang ada di dalamnya. Karena itu, MP3 sering juga disebut *lossy format* yang berarti kehilangan sebagian informasi. Meskipun bersifat *lossy*, MP3 merupakan salah satu format berkas pengodean suara yang memiliki kompresi yang baik sehingga ukuran berkas bisa memungkinkan menjadi lebih kecil. Berkas ini dikembangkan oleh seorang insinyur Jerman Karlheinz Brandenburg. MP3 memakai pengodean *Pulse Code Modulation* (PCM). MP3 mengurangi jumlah *bit* yang diperlukan dengan menggunakan model *psychoacoustic* untuk menghilangkan komponen-komponen suara yang tidak terdengar oleh manusia.

Teknik Psychoacoustic MP3 berhasil memanipulasi telinga dengan membuang bagian yang kurang penting pada suatu *file* musik. Sebagai contoh, apabila terdapat dua nada yang mirip, atau apabila nada tinggi dan rendah muncul secara bersamaan, otak hanya akan memproses salah satunya. Sehingga algoritma MP3 akan memilih sinyal yang lebih penting dan membuang sisanya. Inilah beberapa kelemahan dari sistem pendengaran manusia yang dimanipulasi teknik *psychoacoustic* MP3 (Murad, 2009).

1. Terdapat beberapa suara yang tidak dapat didengar oleh manusia (diluar jangkauan frekuensi 30-30.000 Hz).

2. Terdapat beberapa suara yang dapat terdengar lebih baik bagi pendengaran manusia dibandingkan suara lainnya.
3. Bila terdapat dua suara yang dikeluarkan secara simultan, maka pendengaran manusia akan mendengar yang lebih keras sedangkan yang lebih pelan tidak terdengar.

Seperti lazimnya file yang terkompresi, sebagian besar data MP3 mempunyai kapasitas berkisar dari 2 MB sampai dengan 10 MB. Hal itu disebabkan data MP3 hanya berisi informasi yang diperlukan saja, maksudnya adalah data MP3 hanya berisi data audio yang hanya bisa di dengar oleh manusia dan sebaliknya, data audio yang tidak bisa terdengar oleh manusia dipotong atau dibuang. Itulah mengapa kualitas suara MP3 tidak terlalu jauh dari kualitas CD-Audio.

2.2.1. *Bitrate*

Bitrate merupakan satuan data yang dibaca oleh program dalam satu kali waktu. Semakin besar, seharusnya kualitas *output* yang dihasilkan semakin baik. Jika pada citra terdapat *pixel* maka audio menggunakan *bitrate*, *bitrate* merupakan jumlah *bit* yang dikirim dalam satuan detik, contohnya 192 kbps, berarti dalam satu detik *file* audio tersebut membaca informasi sebesar 192kb atau 24 kB.

Bitrate menyatakan berapa jumlah *bit* yang disalurkan dalam satu detik. *Bit* menjadi suatu digit antara 1 atau 0 yang merupakan awal pembentukan sebuah *char* (huruf) atau bilangan. Dari sini kita ketahui bahwa setiap huruf atau bilangan itu dibentuk dari kumpulan bit-bit.

Bitrate pada audio adalah jumlah informasi (dalam *bit*) disampaikan atau diproses per unit waktu. Sebuah *bitrate* yang lebih tinggi memungkinkan kualitas output yang lebih baik. Tingkat *bit* rata-rata bervariasi antara 0 dan 5000 Kbits / s, nilai *default* adalah 800 Kbits / s untuk kualitas rendah, 1000 Kbit / s untuk kualitas menengah dan 1200 Kbits / s untuk kualitas tinggi.

2.2.2. Struktur MP3

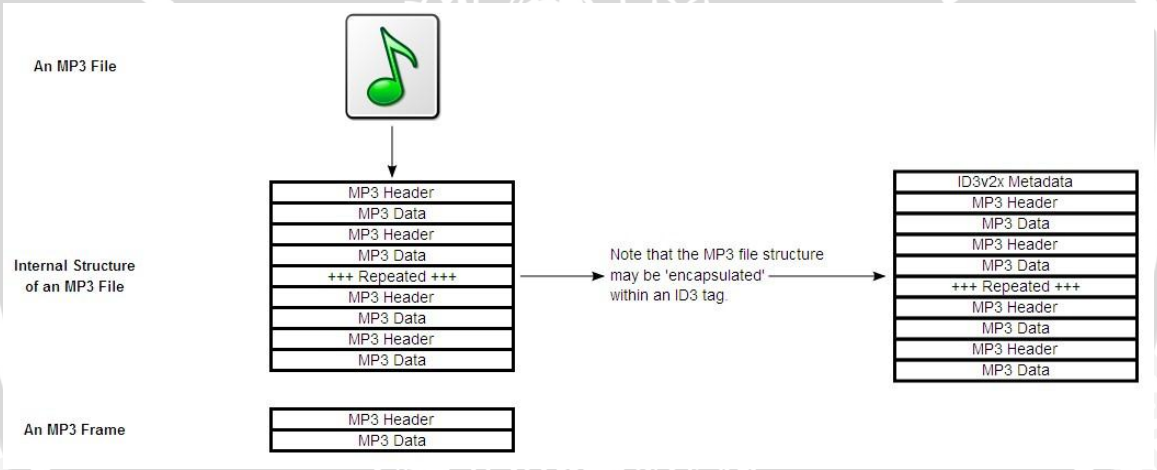
File MP3 tersusun dari banyak *frame* MP3, yang terdiri dari sebuah *header* dan sebuah blok data. Setiap frame secara umum menyimpan 1152

sampel audio selama 26 ms. Artinya frame rate yang dihasilkan sekitar 38 fps. Dengan tambahan setiap frame dibagi menjadi 2 unit yang menyimpan 576 sampel. Karena *bitrate* menentukan ukuran setiap sampel maka memperbesar *bitrate* akan memperbesar ukuran dari frame tersebut. Ukuran juga dipengaruhi oleh frekuensi sampel menurut formula dibawah :

$$Frame\ Size = \frac{144 * bitrate}{samplefrequency} + Padding\ [bytes]$$

Padding mengacu pada bit khusus yang dialokasi di awal frame. Hal itu digunakan di beberapa frame untuk menentukan kebutuhan *bitrate*. Jika terdapat padding bit maka frame akan ditambahkan 1 *byte*.

Berikut adalah gambar struktur file MP3 tanpa ID3 dan setelah dibungkus dengan ID3.



Gambar 2.5. Struktur *File* MP3

Sumber: http://en.wikipedia.org/wiki/RC6

2.2.2.1. ID3

File MP3 hanya berisi informasi audio. Struktur MP3 tidak memungkinkan untuk menyimpan deskripsi konten seperti judul lagu sehingga untuk menyimpan informasi tersebut digunakan ID3 (*Identify* MP3).

ID3 bukan bagian dari sebuah *file* MP3 dan mempunyai struktur yang berbeda dari MP3. Pada versi pertama ID3 (ID3v1), ID3 ditambahkan di akhir *file*. ID3 v1 yang dimulai dengan tulisan *TAG* ini merupakan *fixed size* ID3

artinya ID3 ini memiliki ukuran tetap 128 *byte* yang terbagi menjadi beberapa *field* antara lain untuk judul 30 *byte*, artis 30 *byte*, album 30 *byte*, tahun 4 *byte*, komentar 30 *byte* dan *genre* 1 *byte* serta 3 *byte* untuk tulisan *TAG*. Selanjutnya pada *field* komentar dikurangi ukurannya sebesar 2 *byte* untuk *track number* 1 *byte* dan 1 *byte* yang lain adalah biner 0 yang ditulis di antara *field* komentar dan *track*. Untuk lebih jelasnya dapat dilihat pada gambar. *Tag* ini dikenal dengan ID3 v1.1.

'TAG' 3 bytes	Title 30 bytes	Artist 30 bytes	Album 30 bytes	Year 4 bytes	Comment 28 bytes	'0'	Track 1 byte	Genre 1 byte

Gambar 2.6. ID3 v1.1 *Field*

Sumber: Raissi, Rassol. 2002. The Theory Behind MP3.

ID3 versi lain yaitu ID3 v2 adalah ID3 yang ditambahkan di awal *file* MP3 dan dimulai dengan *header* ID3. *Header* berisi informasi mengenai ID3 seperti panjang data ID3, beberapa *flag* dan beberapa informasi lainnya. ID3 v2 terdiri dari *frame-frame.* Tiap frame menyediakan informasi spesifik tentang *file*, misalnya ada *frame* untuk menyimpan judul lagu dan pada *frame* yang lain untuk menyimpan gambar *band*. Masing-masing *frame* memiliki informasi *header* sendiri. Panjang *frame* adalah variabel yang tersimpan di *header* .

Sebuah frame terdiri dari lima bagian yaitu *header* , CRC, side information, main data dan terakhir ancillary data.

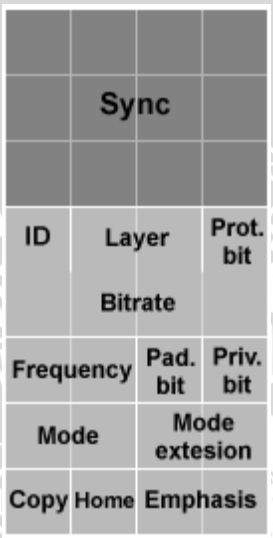
Header	CRC	Side Information	Main Data	Ancillary Data

Gambar 2.7. Struktur *Frame* MP3

Sumber: http://www.MP3-tech.org/programmer/docs/MP3_theory.pdf

2.2.2.2. *Header* Frame

Header berukuran 32 bit dan berisi synchronization word dan deskripsi dari frame itu. Synchronization word yang terdapat disetiap awal frame berguna bagi penerima MP3 untuk mengunci sinyal pada bagian manapun dari stream audio. Hal ini berguna untuk pemancaran *file* MP3. sebuah receiver yang melakukan tuning hanya perlu mencari synchronization word untuk menjalankan lagunya. Masalahnya disini adalah kemungkinan munculnya synchronization word salah dari bagian frame yang lain. Sebuah decoder harus melakukan validasi sync word dari dua frame yang berturut-turut atau mengecek keabsahan data dari bagian side information yang sedikit lebih rumit.



Gambar 2.8. Struktur *Header Frame* MP3

Sumber: http://www.mp3-tech.org/programmer/docs/mp3_theory.pdf

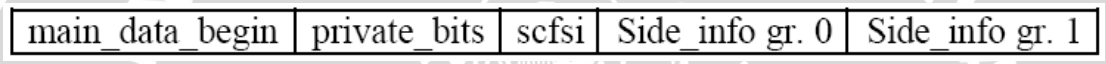
2.2.2.3. CRC

Field ini hanya muncul jika protection bit dalam *header* diset dan berguna untuk mengecek bagian paling sensitif dari data untuk kesalahan transmisi. Data sensitif didefinisikan oleh standar yaitu bit 16 sampai 31 di bagian *header* dan side information. Jika nilai dari bagian ini tidak tepat maka akan merusak semua frame sementara kerusakan pada data utama hanya akan mempengaruhi sebagian dari frame. Kerusakan frame dapat dinonaktifkan atau diganti oleh frame sebelumnya.

2.2.2.4. *Side Information*

Bagian side information dari frame berisi informasi yang dibutuhkan untuk men-decode bagian data utama. Ukurannya bergantung pada channel mode dari data yang dikodekan. Jika modenya adalah single channel bitstream maka ukurannya sebesar 17 *byte*. Jika tidak maka dialokasikan sebesar 32 *byte*.

Panjang dari tiap bagian akan dispesifikasikan di sebuah parenthesis bersama nama field diatas deskripsi sebenarnya. Jika sebuah nilai yang tertulis maka ukuran field-nya akan konstan. Jika dispesifikasikan dua nilai maka nilai yang pertama akan digunakan dalam mode mono dan nilai yang kedua akan digunakan untuk mode yang lainnya yang menyebabkan ukuran field yang beragam. Gambar di bawah ini mengasumsikan mode mono dimana untuk mode yang lain akan dibutuhkan nilai yang terpisah.



Gambar 2.9. Struktur Informasi Tambahan *Frame* MP3

Sumber: http://www.mp3-tech.org/programmer/docs/mp3_theory.pdf

2.2.2.5. *Main Data*

Bagian data utama dari frame berisi *scalefactor*, Huffman *coded bit*, dan *ancillary data*.

2.2.2.5.1. *Scalefactors*

Kegunaan dari *scalefactor* adalah untuk mengurangi *Quantization* noise. Jika sampel audio dalam *scalefactor* band tertentu diukur dan ditentukan secara tepat maka *Quantization* noise akan sepenuhnya tertutupi. Sebuah *scalefactor* untuk tiap *scalefactor* band ditransmisikan. Field scfsi menentukan jika *scalefactor* tersebut didistribusikan antar granul atau tidak. Jumlah bit sebenarnya yang dialokasikan untuk tiap *scalefactor* bergantung pada field scalefac_compress. Pembagian divisi dari spektrum kedalam *scalefactor* band itu ditentukan untuk semua panjang windows dan frekuensi sample.

2.2.2.5.2. *Huffman code bits*

Bagian dari frame ini berisi bit untuk kode Huffman. Informasi mengenai bagaimana men-dekode bagman ini dapat ditemukan pada bagian *side information*. Untuk tiga sub-bagian dalam big_value pengkodean selalu dilakukan secara berpasangan.

2.2.2.6. Ancillary Data

Ancillary data adalah opsional dan banyaknya bit yang dialokasikan tidak ditentukan secara eksplisit. Field ini terletak setelah bit kode Huffman hingga letak poin main_data_begin pada frame selanjutnya.

BAB III

METODE PENELITIAN

Dalam penyusunan skripsi ini, dirancang suatu aplikasi yang merupakan implementasi algoritma RC6 untuk memproteksi *file* berformat MP3. Metode penelitian yang digunakan pada penyusunan skripsi ini adalah:

3.1. Studi Literatur

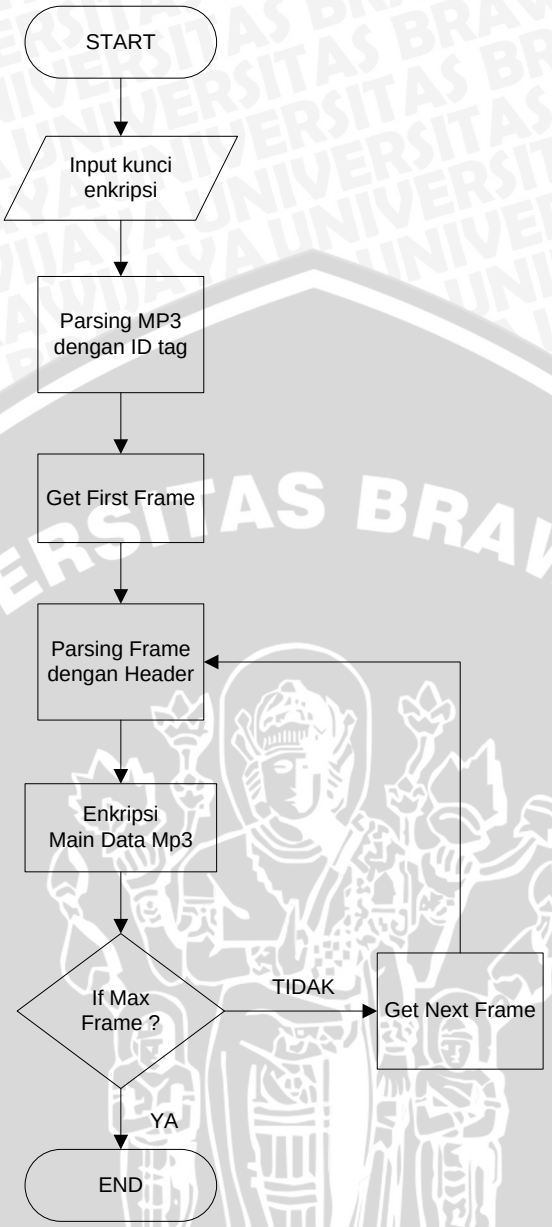
Studi literatur dilakukan dengan mencari dan mempelajari teori-teori yang berhubungan dengan *enkripsi*, *file* MP3, bahasa pemrograman Delphi serta literatur-literatur lainnya yang terkait.

3.2. Analisa Kebutuhan

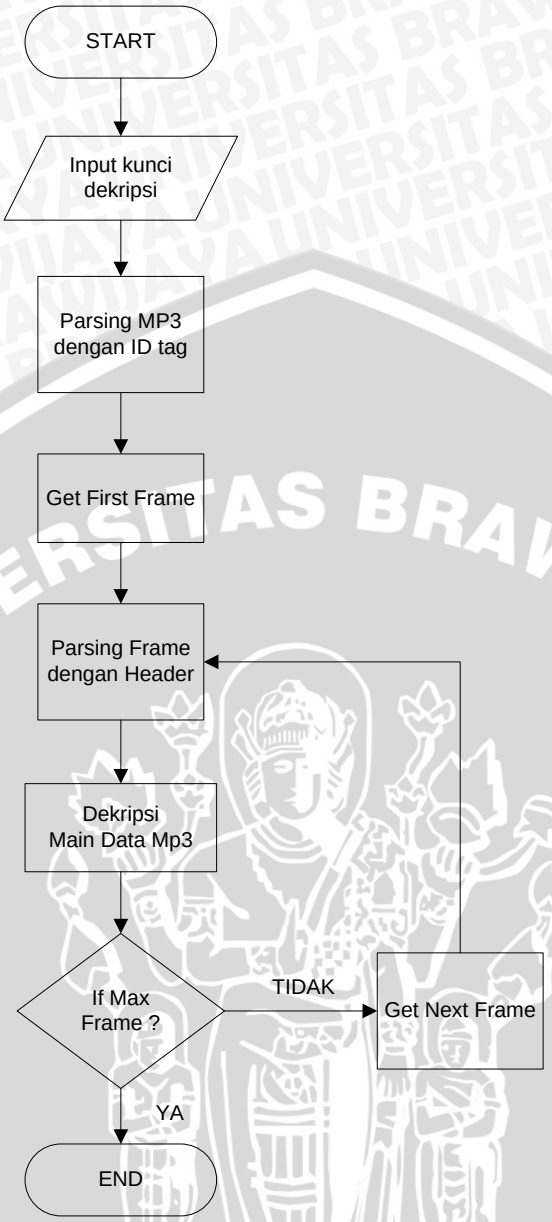
Analisa kebutuhan dilakukan untuk mendapatkan informasi mengenai perangkat-perangkat atau data yang diperlukan dalam pembuatan aplikasi, antara lain: PC atau Laptop, sampel *file* MP3, sistem operasi Windows dan bahasa pemrograman Delphi.

3.3. Diagram Alir Sistem

Berikut merupakan diagram alir dari proses enkripsi dan dekripsi *file* MP3. Diagram pertama memperlihatkan langkah-langkah pengenkripsian *file* MP3 sehingga didapatkan hasil berupa *file* MP3 yang terproteksi yang berformat .mp3. diagram kedua menunjukkan proses pendekripsian *file* dimana *output* - nya berupa *file* yang tersimpan dalam *memory* yang langsung dimainkan oleh *player*.



Gambar 3.1. Proses Enkripsi *File* MP3



Gambar 3.2. Proses Dekripsi *File* MP3

3.4. Perancangan dan Implementasi Sistem

1. Mengimplementasikan algoritma RC6 pada perancangan aplikasi.
2. Merancang *interface* dari aplikasi.
3. Merancang aplikasi dengan menggunakan bahasa pemrograman Delphi.

3.5. Pengujian Sistem

Pengujian dilakukan untuk menjamin dan memastikan bahwa sistem yang telah dirancang memiliki tingkat kesalahan yang kecil. Untuk mengetahui apakah

sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkain pengujian. Pengujian yang dilakukan adalah sebagai berikut:

1. Pengujian keberhasilan program dalam mengenkripsi *file* MP3 sehingga *file* masih dapat di-*play*.
2. Pengujian keberhasilan program dalam mendekripsi *file* yang telah terproteksi.

3.6. Kesimpulan dan Saran

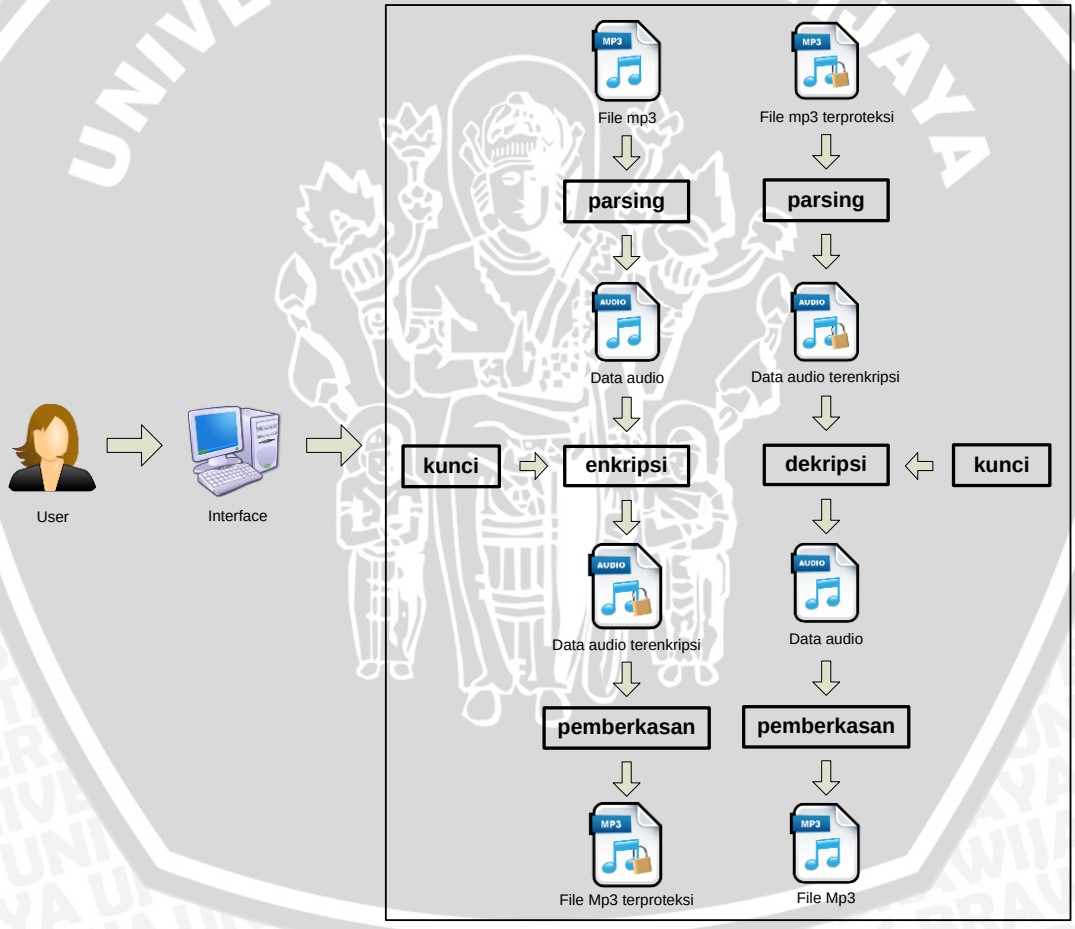
Kesimpulan diambil dari hasil pengujian dan analisa terhadap aplikasi yang telah dirancang. Selanjutnya adalah pengambilan saran untuk pengembangan aplikasi lebih lanjut serta dapat menyempurnakan kekurangan yang ada.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas mengenai perancangan dan implementasi aplikasi untuk memproteksi *file* MP3 dengan menggunakan Algoritma RC6 yang meliputi proses pembangkitan kunci, enkripsi dan dekripsi.

4.1. Perancangan Secara Umum

Perancangan sistem secara umum merupakan tahap awal sebagai acuan dalam perancangan aplikasi yang akan dibuat. Berikut adalah blok sistem diagram dan cara kerja sistem.



Gambar 4.1. Diagram Blok Sistem Secara Umum

Sumber: Perancangan

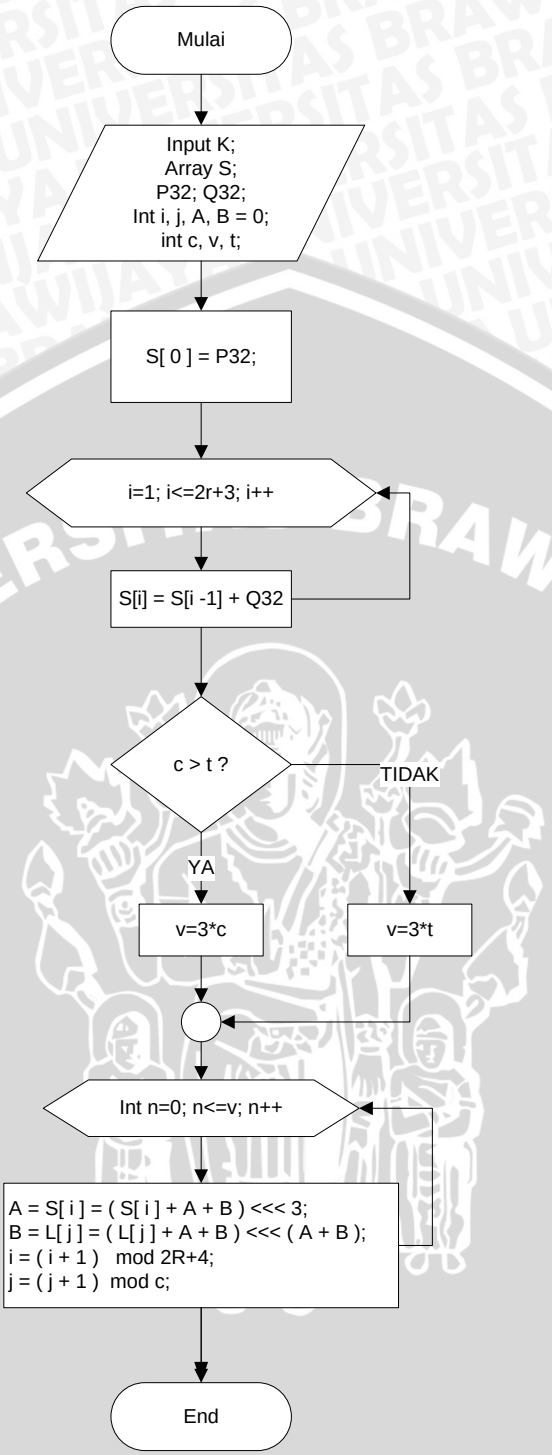
User menjalankan aplikasi proteksi *file* MP3 kemudian memasukkan *file* MP3 yang ingin diproteksi lalu menentukan dan meng-*input* kunci/*password*. Proses selanjutnya terjadi dalam aplikasi, kunci/*password* yang dimasukkan oleh *user* dibangkitkan menjadi sub kunci kemudian digunakan dalam proses enkripsi yang kemudian menghasilkan keluaran berupa *file* MP3 terproteksi (*chiper*). Apabila *user* ingin mengembalikan *file* MP3 yang terproteksi tersebut seperti semula, *user* kembali memasukkan *file* MP3 tersebut ke dalam aplikasi kemudian meng-*input* kunci/*password* maka proses dekripsi akan terjadi dan menghasilkan keluaran berupa *file* MP3 asli (*plain*).

4.2. Perancangan Aplikasi

Perancangan aplikasi dibangun dengan menggunakan bahasa pemrograman Delphi. Perancangan ini meliputi pembangkitan kunci, proses enkripsi dan proses dekripsi. Pada pembangkitan kunci (*key expansion*), kunci akan diproses menjadi sub kunci yang kemudian digunakan dalam proses enkripsi dan dekripsi. Proses enkripsi berupa proses perhitungan antara data *audio* yang berbentuk hexadesimal dengan sub kunci yang telah dibentuk yang kemudian akan menjadi *cipher*. Sedangkan proses dekripsi akan mengembalikan semua *cipher* ke dalam bentuk yang seperti semula. Dalam setiap proses ini akan dijelaskan tahapan tahapan yang membentuk semua proses sehingga menjadi program yang utuh.

4.2.1. Proses Pembangkitan Kunci

Proses ini dilakukan untuk membangkitkan kunci internal berdasarkan kunci rahasia K untuk mengisi tabel sub kunci S. Berikut adalah diagram alir proses pembangkitan kunci.



Gambar 4.2. Diagram Alir Proses *Key Expansion*

Sumber: Perancangan

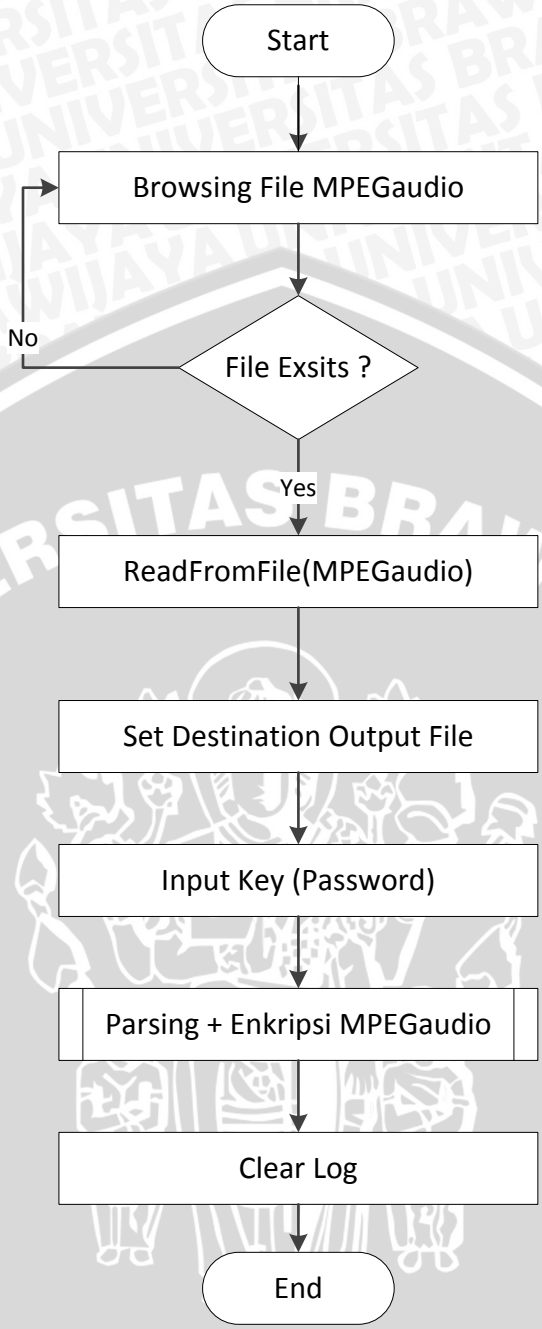
Berikut penjelasan proses pembangkitan kunci :

1. Inisialisasi P32 dan Q32 dalam bentuk hexadesimal sebagai konstanta ajaib, input kunci K dalam bentuk *byte*, *array* S dan variabel lain yang diperlukan dalam proses pencampuran kunci.
2. Menduplikasi kunci rahasia K yang masih berupa larik *byte* ke dalam larik kata L[0...c-1] dimana c=(b/u) dengan u=(w/8) yang menyatakan jumlah *byte* dalam 1 kata.
3. Inisialisasi tabel kunci S sehingga tabel S berisi pola bit *pseudo-random* yang tetap dengan panjang array S adalah t = 2r + 4. Array ke 0 dari S diisikan P32 dan array S ke 1 hingga array S terakhir (2r+3) didapat dengan Array S sebelumnya ditambahkan dengan Q32.
4. Pembandingan antara panjang Array L dan panjang Array S. Dan yang paling panjang dikalikan dengan 3 untuk dijadikan batas iterasi.
5. Proses pencampuran kunci rahasia dengan menambahkan tiap array L dan S dengan A dan B merotasikan kekanan sebanyak 3 lalu dicari modulus untuk tiap looping sedemikian hingga didapatkan table S sebagai table subkunci.

4.2.2. Proses Enkripsi

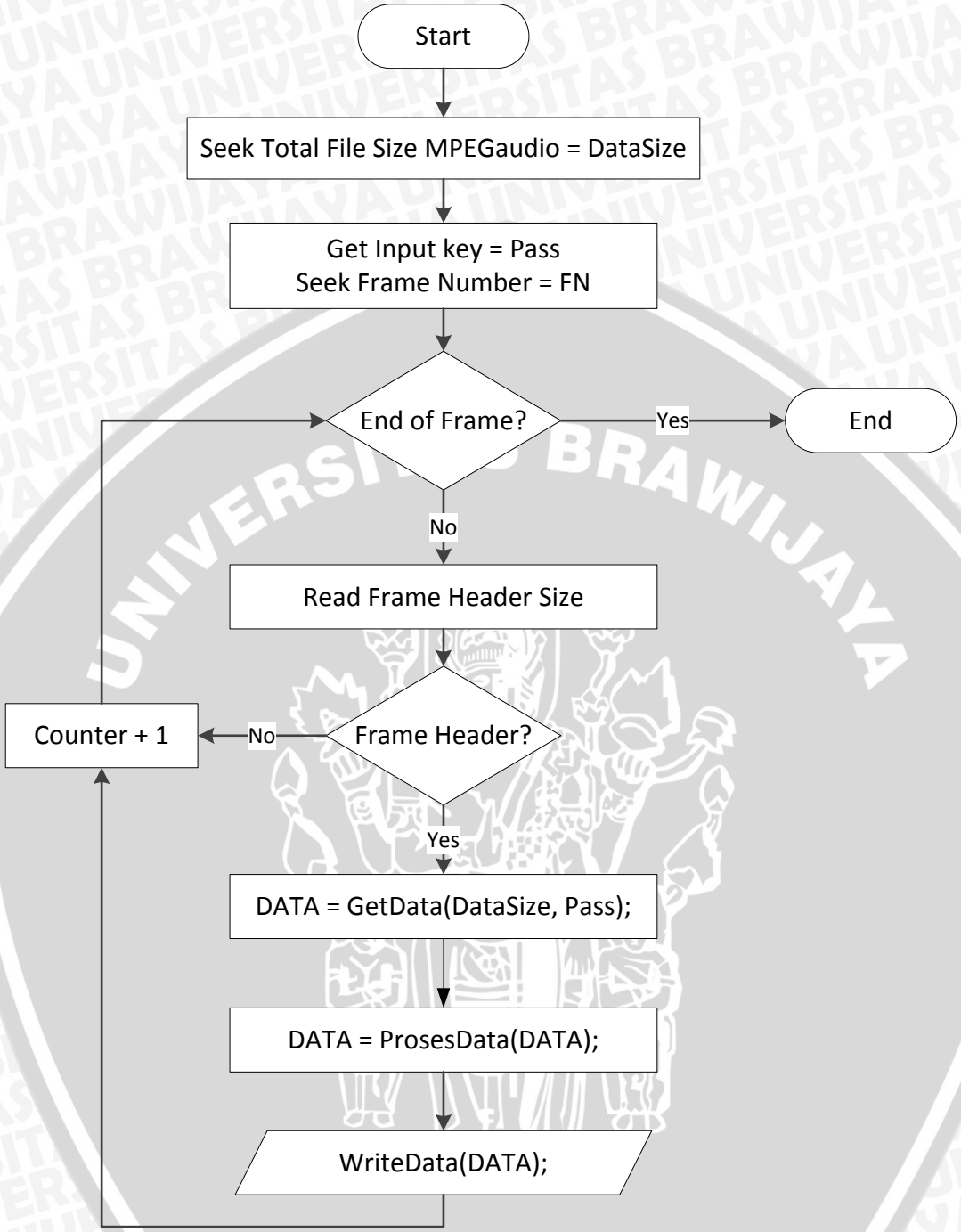
Proses enkripsi merupakan cara untuk mendapatkan *cipher* dari *plaintext* asli. Pada proses sebelumnya diasumsikan bahwa tabel sub kunci yang baru telah didapatkan sehingga tabel sub kunci yang baru tersebut akan digunakan dalam proses perhitungan untuk mengacak *plaintext* asli agar dapat diubah menjadi *cipher*.

Dalam perancangan aplikasi enkripsi ini, tahap awal yang dilakukan yaitu *parsing* yang bertujuan untuk memisahkan *header frame* dengan *main data* MP3. Tahap selanjutnya yaitu enkripsi pada *main data* MP3. Berikut adalah diagram proses enkripsi secara keseluruhan.



Gambar 4.3 Diagram Alir Proses Enkripsi *File* MP3

Sumber: Perancangan

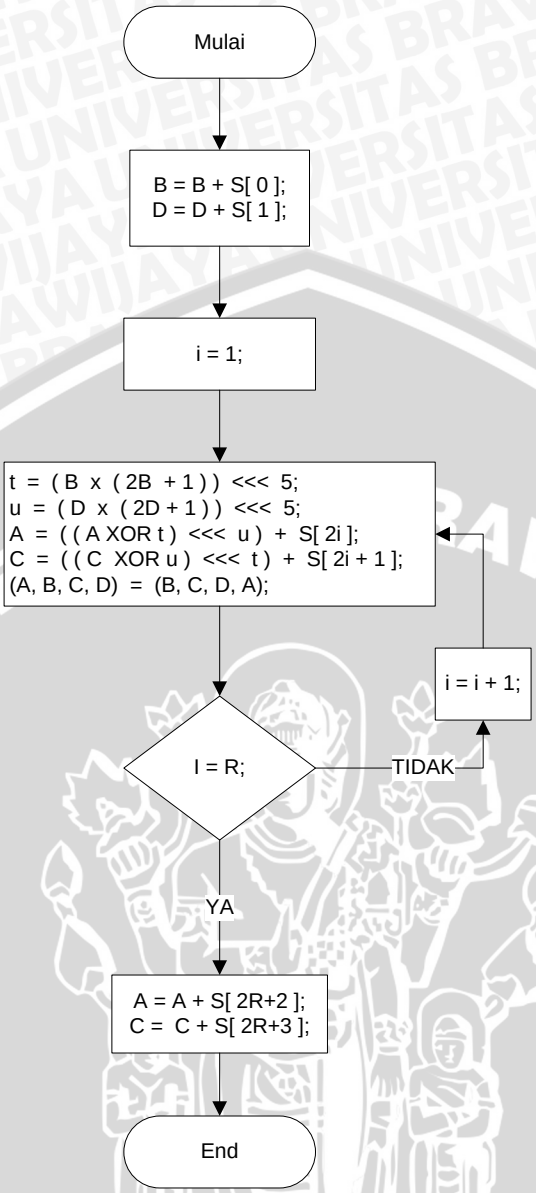


Gambar 4.4 Diagram Alir Proses *Parsing* dan Enkripsi *File* MP3

Sumber: Perancangan

Berikut proses penjelasan enkripsi pada *file* MP3:

1. Tahap awal dalam aplikasi enkripsi yaitu inisialisasi fungsi pengenalan *file* MP3.
2. Tahap selanjutnya aplikasi akan membaca informasi yang terdapat pada *file* MP3 (ukuran, *bitrate, sample rate, channel mode,* durasi dan *encoder*) dan menampilkannya kepada *user*.
3. Selanjutnya aplikasi akan meminta *user* untuk memasukkan tempat tujuan untuk menyimpan hasil enkripsi *file* MP3 dan menuliskan kunci/*password* sebagai salah satu tahap untuk melakukan enkripsi terhadap *file* MP3.
4. Setelah seluruh data terkumpul, tahap selanjutnya yaitu parsing *file* MP3 untuk memisahkan *header* dan *main data* pada tiap-tiap *frame*. Jika data bernilai 0xFF dan data selanjutnya bernilai 0xFA atau 0xFB maka data tersebut dan 2 byte data berikutnya merupakan *frame header* dan data setelah *frame header* adalah *main data.*
5. Tahap akhir dari aplikasi ini yaitu proses enkripsi *main data* MP3 dengan menggunakan algoritma RC6. Proses tersebut berulang hingga mencapai *frame* terakhir.
6. Setelah proses enkripsi berhasil dan telah selesai, maka aplikasi akan membersihkan seluruh *history* hasil enkripsi *file* sebelumnya.



Gambar 4.5 Diagram Alir Proses Algoritma Enkripsi RC6

Sumber: Perancangan

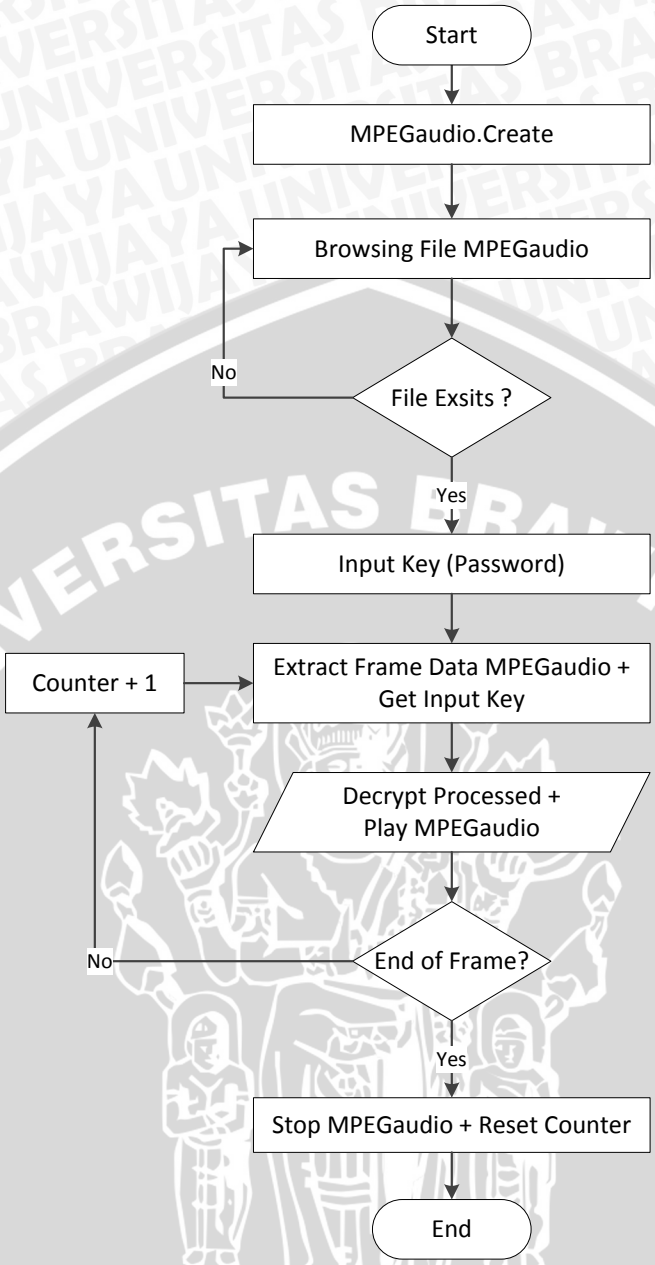
Berikut penjelasan proses enkripsi menggunakan algoritma RC6:

1. Data di-*input* ke dalam masing – masing register A , B , C dan D. *Byte* yang pertama dari *plaintext* ditempatkan pada *byte* A sedangkan *byte* yang terakhir ditempatkan pada *byte* D.
2. Algoritma RC6 menggunakan 44 buah subkunci yaitu S[0] hingga S[43] dengan panjang masing-masing 32 bit yang dibangkitkan dari kunci.

3. Proses enkripsi dimulai dengan proses *whitening* awal. B=B+S[0] dan D=D+S[1] dimana B adalah register B, D adalah register D, S[0] dan S[1] merupakan sub kunci.
4. Masuk dalam proses iterasi, r=20. Saat i=1 (iterasi ke-1), nilai B dimasukkan ke dalam fungsi f(x)=x(2x+1) sehingga menjadi f(B)=B(2B+1)=t lalu digeser ke kiri sejauh *log* w= *log* 32 atau 5 bit. t kemudian di-XOR dengan A lalu digeser sejauh u dimana u=f(D)=D(2D+1) dan ditambahkan dengan S[2i].
5. Nilai keempat blok tersebut kemudian ditukar dengan aturan (A,B,C,D) = (B,C,D,A).
6. Proses tersebut dilakukan berulang-ulang hingga r=20 (20 kali).
7. Setelah proses iterasi selesai dilanjutkan dengan proses *whitening* akhir dimana A=A+S[2r+2] dan C=C+S[2r+3].

4.2.3. Proses Dekripsi

Proses dekripsi merupakan kebalikan dari proses enkripsi yaitu cara untuk mendapatkan *plaintext* dari *ciphertext*. Pada proses sebelumnya di asumsikan bahwa tabel sub kunci yang baru telah didapatkan dari kunci yang dimasukan sehingga tabel sub kunci yang baru tersebut akan digunakan dalam dalam proses perhitungan untuk mengembalikan *plaintext* asli dari *ciphertext*. Berikut ini adalah perancangan dekripsi :

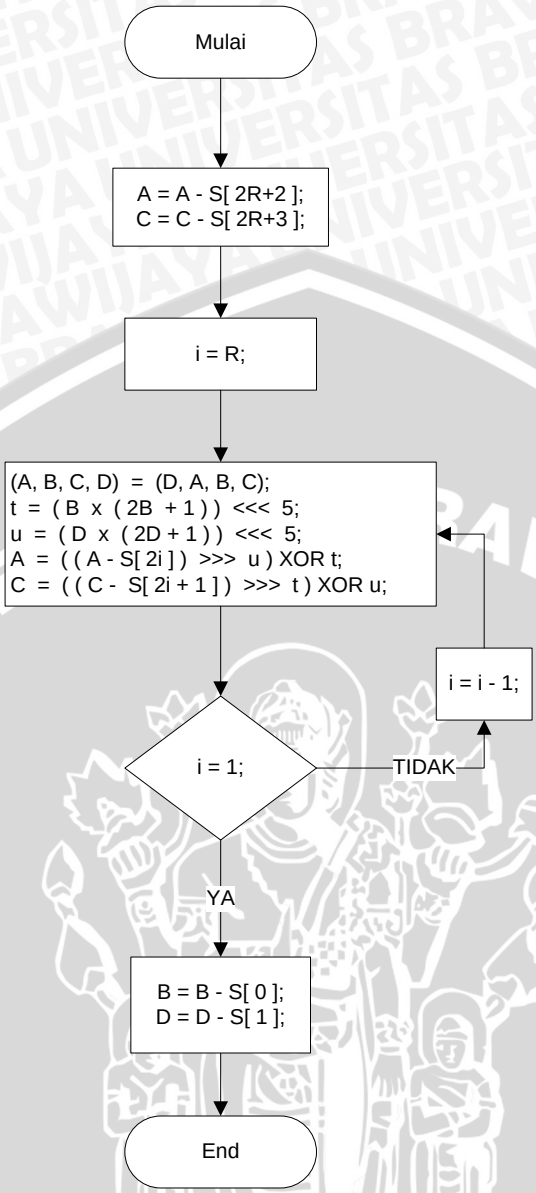


Gambar 4.6 Diagram Alir Proses Dekripsi

Sumber: Perancangan

Berikut penjelasan proses dekripsi:

1. Tahap awal dalam aplikasi dekripsi yaitu inisialisasi fungsi pengenalan *file* MP3 kemudian mengambil *file* MP3 yang sudah terenkripsi.
2. Tahap selanjutnya, apabila *file* MP3 tersedia maka *user* diminta untuk memasukkan kunci/*password* yang digunakan untuk melakukan proses dekripsi.
3. Setelah fungsi generator *key* mendapatkan *key* dari *user*, tahap selanjutnya aplikasi akan membaca keseluruhan data dari *file* MP3 yang pada akhirnya aplikasi akan menemukan *header*, *frame* dan *key* untuk melakukan dekripsi.
4. Setelah aplikasi mendapatkan *key*, tahap selanjutnya aplikasi akan melakukan proses dekripsi dengan menggunakan algoritma dekripsi RC6 dan sekaligus memutar *file* MP3 (*buffer*). *File* MP3 dimainkan setelah aplikasi mendekripsi 100 *frame* data MP3.
5. Ketika aplikasi membaca akhir dari *file* MP3 (*end frame*) maka aplikasi akan menghentikan pemutar audio dan mereset logika penghitung pada proses deksripsi.



Gambar 4.7 Diagram Alir Proses Algoritma Dekripsi RC6

Sumber: Perancangan

Berikut penjelasan proses dekripsi algoritma RC6:

1. Pengambilan data cipher sebagai data input ke dalam masing – masing register A , B , C dan D.
2. Proses pengembalian putaran dan pengembalian *whitening* dengan A = A - S[2r + 2] dan register C = C - S[2r + 3].

3. Mulai putaran ke 1 sampai putaran maksimal dilakukan perhitungan putaran dan dekripsi. Proses tersebut untuk mendapatkan plain dari cipher di masing masing register adalah :

   (A,B,C,D) = (D,A,B,C)

   u = (D × (2D + 1)) <<< 1g w

   t = (B × (2B + 1)) <<< 1g w

   C = ((C − S[2i + 1] >>> t) ⊕ u

   A = ((A − S[2i]) >>> u) ⊕ t

4. Dari proses pertukaran isi register tersebut didapatkan register A ,B , C dan D yang baru. Di sini akan dikembalikan lagi proses *whitening* awal untuk register A dan C yaitu register D = D − S[1] dan B = B − S[0].

4.3. Implementasi Sistem

Setelah tahap perancangan sistem tahap selanjutnya adalah tahap implementasi. Tujuan dari tahap implementasi ini merupakan proses transformasi hasil perancangan aplikasi. Pembahasan terdiri dari lingkungan implementasi (spesifikasi aplikasi dan perangkat keras), implementasi antarmuka aplikasi dengan sintaks dari bahasa pemrograman yang digunakan.

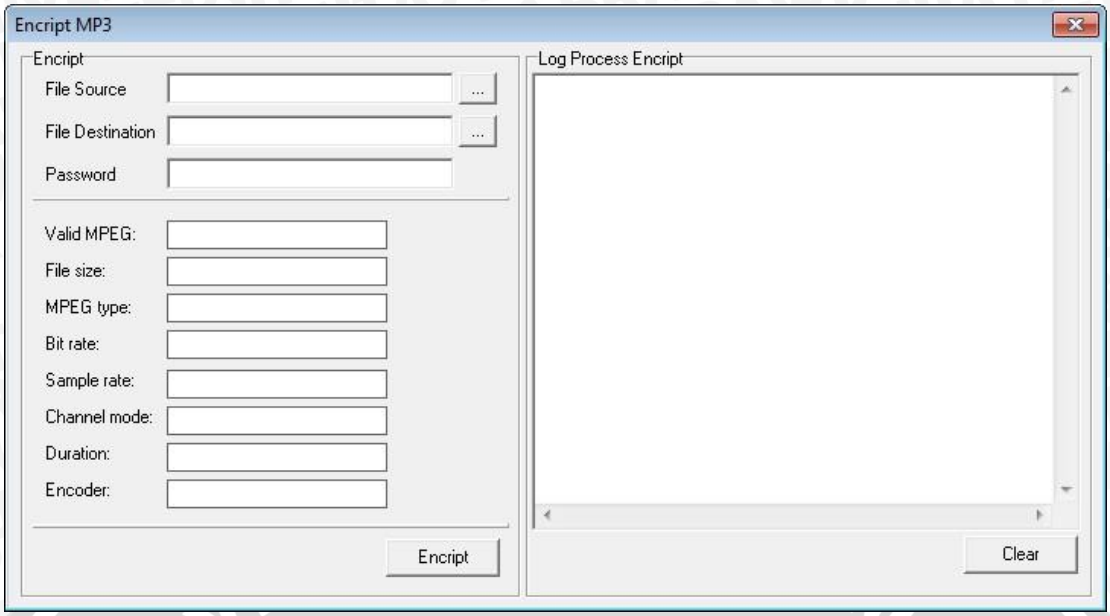
4.3.1. Lingkungan Implementasi

Sistem dibuat dengan menggunakan Delphi 7 diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

1. Perangkat keras :
   1.1. Komputer
   Spesifikasi :
   - Processor : Intel Atom N455 1,66 GHz
   - Memory : 1024MB RAM
   - VGA : Intel GMA 3150
   - Speaker : Realtek HDA
2. Aplikasi :
   2.1. Sistem operasi : Microsoft Windows 7 Starter
   2.2. Bahasa pemrograman: Delphi 7

4.3.2. Implementasi Antarmuka Aplikasi Enkripsi

Aplikasi enkripsi memiliki tampilan antarmuka yang berupa kolom *file source, file destination*, kolom *input password/key*, keterangan lengkap *file* MP3 serta keterangan proses enkripsi sebagai berikut.

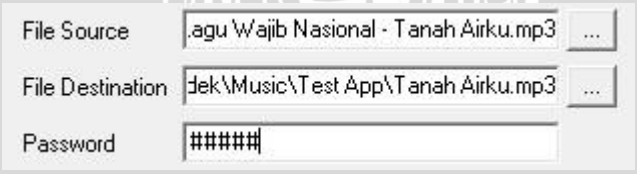


Gambar 4.8. Tampilan Antarmuka Aplikasi Enkripsi

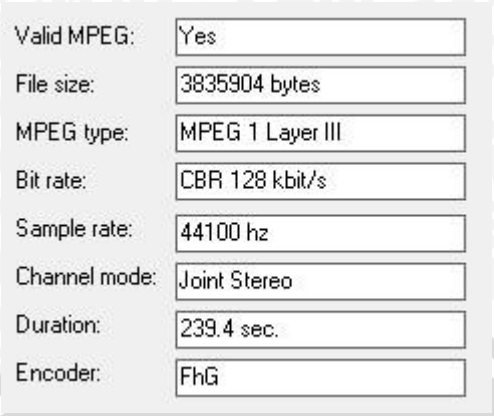
Pada aplikasi ini, antarmuka dibuat untuk memudahkan pengguna dalam menggunakan aplikasi yang telah dibangun. Antarmuka ini ditampikan ketika pertama kali program dijalankan.

4.3.2.1. Implementasi Antarmuka Masukan *File* MP3 dan Kunci

Untuk memudahkan pengguna dalam melakukan input *file* MP3, dibuat antarmuka yang dapat secara langsung mengambil data dari *harddisk* begitu pula untuk penyimpanan data yang telah dienkripsi. Sedangkan kunci dapat langsung diketik oleh pengguna. Implementasi sebagai berikut :



Gambar 4.9 User Interface Masukan *File* MP3 dan Kunci



Gambar 4.10 User Interface Keterangan *File* MP3

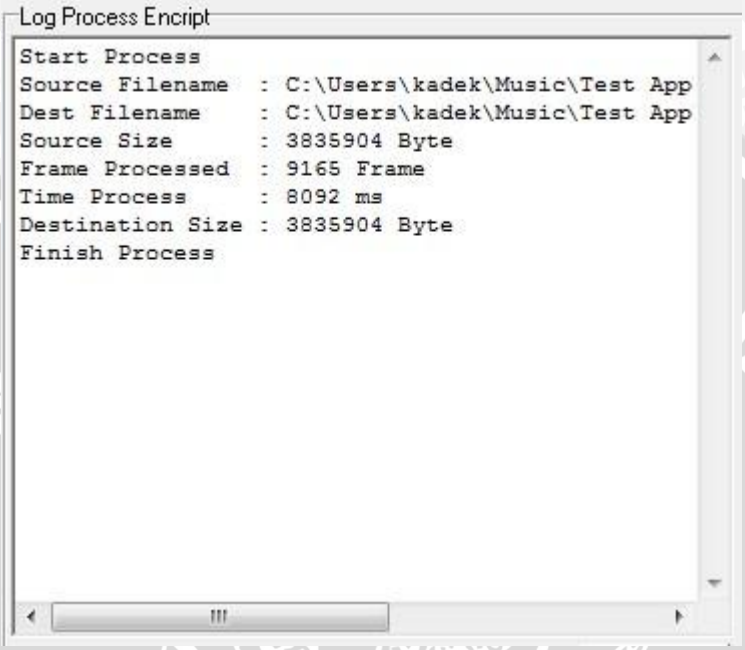
Setelah *file* MP3 dimasukkan akan muncul keterangan mengenail *file* tersebut antara lain ukuran, *bitrate, sample rate, channel mode, encoder* dan durasi. Berikut adalah potongan program :

```
procedure TFMain.Button1Click(Sender: TObject);
begin
 if OpenDialog1.Execute then
 begin
  { Clear captions }
  ClearAll;

  if FileExists(OpenDialog1.FileName) then
   { Load MPEG audio data  }
   if MPEGaudio.ReadFromFile(OpenDialog1.FileName) then
    if MPEGaudio.Valid then
    begin
     { Fill captions }
     ValidMPEGText.Text := 'Yes';
     FileSizeText.Text := IntToStr(MPEGaudio.FileLength) + ' bytes';
     MPEGTypeText.Text := MPEGaudio.Version + ' ' + MPEGaudio.Layer;
     if MPEGaudio.VBR.Found then
      BitRateText.Text := 'VBR ' + IntToStr(MPEGaudio.BitRate) + ' kbit/s'
     else
```

```
      BitRateText.Text := 'CBR ' + IntToStr(MPEGaudio.BitRate) + ' kbit/s';
      SampleRateText.Text := IntToStr(MPEGaudio.SampleRate) + ' hz';
      ChannelModeText.Text := MPEGaudio.ChannelMode;
      DurationText.Text := FormatFloat('.0', MPEGaudio.Duration) + ' sec.';
      EncoderText.Text := MPEGaudio.Encoder;

      EdSource.Text := OpenDialog1.FileName;
     end
     else
      { Not valid MPEG file }
      ValidMPEGText.Text := 'No'
    else
     { Read error }
     ShowMessage('Can not read data from the file: ' + OpenDialog1.FileName)
  else
   { File does not exist }
   ShowMessage('The file does not exist: ' + OpenDialog1.FileName);
 end;
end;

procedure TFMain.Button2Click(Sender: TObject);
begin
 if SaveDialog1.Execute then
  EdDestination.Text := SaveDialog1.FileName;
end;
```

4.3.2.2. Implementasi Antarmuka Keterangan Proses Enkripsi

Antarmuka keterangan proses enkripsi ini ditampilkan ketika pengguna menekan tombol encript untuk melakukan proses enkripsi. Implementasi sebagai berikut.



Gambar 4.11 User Interface Keterangan Proses Enkripsi

Berikut adalah potongan program :

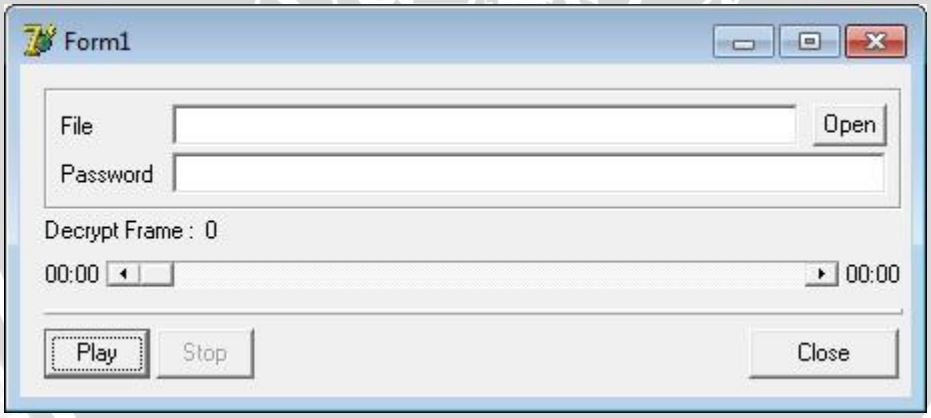
```
procedure TFMain.Button3Click(Sender: TObject);
var
  startTime64, endTime64, frequency64: Int64;
  fr_count : integer;
  waktu_ms : single;
begin
  if not FileExists(EdSource.Text) then
  begin
    ShowMessage('Source File must not Empty');
    exit;
  end;

  if Length(trim(EdDestination.Text)) < 1 then
```

```
  begin
    ShowMessage('Destination File must not Empty');
    exit;
  end;

  Source := TFileStream.Create(EdSource.Text, fmOpenRead or
fmShareDenyWrite);
  try
    Destination := TFileStream.Create(EdDestination.Text, fmCreate or
fmShareDenyRead);
    try
      Destination.CopyFrom(Source,Source.Size);

      tulisLog('Start Process');
      tulisLog('Source Filename  : ' + EdSource.Text);
      tulisLog('Dest Filename    : ' + EdDestination.Text);
      tulisLog('Source Size      : ' + IntToStr(Source.Size) +' Byte');
      tulisLog('Wait for Process Encryption ...');
      fr_count := MPEGaudio.Frames;

      QueryPerformanceFrequency(frequency64);
      QueryPerformanceCounter(startTime64);

      FrameProcessed := 0;
      MPEGaudio.ReadFrameByIndex(0,fr_count);

      QueryPerformanceCounter(endTime64);
      waktu_ms := (endTime64 - startTime64) / frequency64;

      MLog.Lines.Delete(MLog.Lines.Count-1);

      tulisLog('Frame Processed  : ' + IntToStr(FrameProcessed) + ' Frame');
```

```
    tulisLog('Time Process     : ' + IntToStr(round(waktu_ms*1000)) + ' ms');

    tulisLog('Destination Size : ' + IntToStr(Source.Size) +' Byte');
    tulisLog('Finish Process');
    tulisLog(' ');
   finally
    Destination.Free;
   end;
  finally
   Source.Free;
  end;
end;
```

4.3.3. Implementasi Antarmuka Aplikasi Dekripsi

Aplikasi dekripsi yang sekaligus difungsikan sebagai MP3 *player* ini memiliki tampilan yang lebih sederhana. Pada antarmuka ini terdapat kolom masukan *file* MP3 dan kunci tanpa ada pilihan untuk menyimpan hasil dekripsi.



Gambar 4.12 User Interface Aplikasi Dekripsi

Berikut adalah potongan program :

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if OpenDialog1.Execute then
 begin
  if FileExists(OpenDialog1.FileName) then
```

```
    EdSource.Text := OpenDialog1.FileName
  else
    { File does not exist }
    ShowMessage('The file does not exist: ' + OpenDialog1.FileName);
 end;
end;

procedure TForm1.sb_posChange(Sender: TObject);
begin
 lb_pos.Caption := second_to_minute(sb_pos.Position);
end;
```

BAB V
PENGUJIAN DAN ANALISIS

Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkaian pengujian. Pengujian yang dilakukan dalam bab ini adalah sebagai berikut:

1. Pengujian validasi.
2. Pengujian kualitas suara.
3. Pengujian program kecepatan enkripsi dan dekripsi.

5.1. Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang perancangan. Item- item yang telah dirumuskan dalam perancangan menjadi acuan untuk melakukan pengujian validasi.

5.1.1. Pengujian Enkripsi *File* MP3

Nama Kasus Uji	: Kasus Uji Enkripsi *File* MP3
Tujuan Pengujian	: Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan enkripsi *file* MP3 menjadi acak
Prosedur Uji	: *Load file* MP3. Mengisi kunci pada form isian kunci dengan data private key “12345”. Berikut adalah tabel data masukan pada *offset* ke – 1200 dalam bentuk heksadesimal.

Tabel 5.1. Pengujian data masukan untuk proses enkripsi

No	Data Masukan Plain
1	8004035BD3D30F1B6A3B838A7965E849
2	8C34B566EC3D0B6186222CDD860974D7
3	0B848D57A7B1A961698F2B34F6252454

Hasil yang diharapkan : Aplikasi dapat melakukan proses pengamanan *file* dengan cara melakukan enkripsi *main data* MP3.

5.1.2. Pengujian Dekripsi *File* MP3

Nama Kasus Uji : Kasus Uji Dekripsi *File* MP3

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan dekripsi *file* MP3 acak menjadi *plaintext*

Prosedur Uji : *Load file* MP3 dengan data masukan adalah data keluaran dari uji kasus enkripsi *file* MP3. Memasukan private key “12345” untuk proses dekripsi.

Tabel 5.2. Pengujian data masukan untuk proses dekripsi

No	Cipher
1	7714C4FCFF24B9A8CA322FF181BAB98D
2	A152E0EFE1B0FEBC08F133F47FFC86DB
3	76DE28232E6AC25E2BEC5EA9A983C736

Hasil yang diharapkan : Aplikasi dapat melakukan proses dekripsi dan memainkan *file* MP3 yang telah terdekripsi.

5.1.3. Hasil Pengujian Validasi

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian maka didapatkan hasil sebagai berikut.

Tabel 5.3. *Test case* untuk pengujian validasi

<table>
<tr><th>No.</th><th>Kasus Uji</th><th>Hasil yang didapatkan</th><th>Status</th></tr>
<tr><td>1.</td><td>Enkripsi pesan rahasia</td><td>Aplikasi dapat melakukan proses dekripsi dengan hasil data pada tabel enkripsi
<table>
<tr><th>No</th><th>Cipher</th></tr>
<tr><td>1</td><td>7714C4FCFF24B9A8CA322FF181BAB98D</td></tr>
<tr><td>2</td><td>A152E0EFE1B0FEBC08F133F47FFC86DB</td></tr>
<tr><td>3</td><td>76DE28232E6AC25E2BEC5EA9A983C736</td></tr>
</table></td><td>Valid</td></tr>
<tr><td>4</td><td>Dekripsi pesan rahasia</td><td>Aplikasi dapat melakukan proses dekripsi dan mengembalikan ciphertext ke plaintext seperti pada data enkripsi
<table>
<tr><th>No</th><th>Hasil dekripsi</th></tr>
<tr><td>1</td><td>8004035BD3D30F1B6A3B838A7965E849</td></tr>
<tr><td>2</td><td>8C34B566EC3D0B6186222CDD860974D7</td></tr>
<tr><td>3</td><td>0B848D57A7B1A961698F2B34F6252454</td></tr>
</table></td><td>Valid</td></tr>
</table>

5.2. Pengujian Kualitas Suara

Pengujian kualitas suara ini ditujukan untuk membandingkan kualitas suara *file* MP3 setelah dienkripsi dan setelah didekripsi kembali, apakah dapat didengarkan dengan baik atau tidak. Pengujian ini dilakukan secara subyektif dengan menggunakan tiga sampel data yang berbeda.

5.2.1. Pengujian kualitas suara setelah proses enkripsi

Nama Kasus Uji : Kasus Uji Kualitas Suara *File* MP3 Setelah Proses Enkripsi

Tujuan Pengujian : Pengujian dilakukan untuk membandingkan kualitas suara *file* MP3 sebelum dan sesudah proses enkripsi.

Prosedur Uji : Subyek diperdengarkan beberapa sampel *file* MP3 yang belum dienkripsi. Kemudian subyek diperdengarkan sampel *file* MP3 yang sudah dienkripsi.

Hasil pengujian : Data hasil pengujian kualitas suara pada sistem ini diperlihatkan pada tabel 5.4.

Tabel 5.4. Data hasil pengujian kualitas suara setelah proses enkripsi

	Berkas 1	Berkas 2	Berkas 3
Subyek 1	Tidak dapat didengarkan	Tidak dapat didengarkan	Tidak dapat didengarkan
Subyek 2	Tidak dapat didengarkan	Tidak dapat didengarkan	Tidak dapat didengarkan
Subyek 3	Tidak dapat didengarkan	Tidak dapat didengarkan	Tidak dapat didengarkan
Subyek 4	Tidak dapat didengarkan	Tidak dapat didengarkan	Tidak dapat didengarkan
Subyek 5	Tidak dapat didengarkan	Tidak dapat didengarkan	Tidak dapat didengarkan

5.2.2. Pengujian kualitas suara setelah proses dekripsi

Nama Kasus Uji : Kasus Uji Kualitas Suara *File* MP3 Setelah Proses Dekripsi

Tujuan Pengujian : Pengujian dilakukan untuk membandingkan kualitas suara *file* MP3 sebelum proses enkripsi dan sesudah proses dekripsi.

Prosedur Uji : Subyek diperdengarkan beberapa sampel *file* MP3 yang belum dienkripsi. Kemudian subyek diperdengarkan sampel *file* MP3 yang sudah didekripsi.

Hasil pengujian : Data hasil pengujian kualitas suara pada sistem ini diperlihatkan pada tabel 5.5.

Tabel 5.5. Data hasil pengujian kualitas suara setelah proses dekripsi

	Berkas 1	Berkas 2	Berkas 3
Subyek 1	Terdengar sama	Terdengar sama	Terdengar sama
Subyek 2	Terdengar sama	Terdengar sama	Terdengar sama
Subyek 3	Terdengar sama	Terdengar sama	Terdengar sama
Subyek 4	Terdengar sama	Terdengar sama	Terdengar sama
Subyek 5	Terdengar sama	Terdengar sama	Terdengar sama

5.2.3. Hasil Pengujian Kualitas Suara

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian maka didapatkan bahwa terjadi penurunan kualitas suara setelah *file* MP3 dienkripsi sehingga tidak dapat didengarkan dengan baik oleh telinga manusia. Namun setelah didekripsi, *file* MP3 tersebut dapat didengarkan kembali dan memiliki kualitas suara yang hampir sama dengan *file* aslinya.

5.3. Pengujian Kecepatan Proses

Pengujian kecepatan proses pada aplikasi ini ditujukan untuk mengetahui proses enkripsi dan dekripsi sebagai acuan analisis terhadap kecepatan proses. Dalam percobaan, akan digunakan 5 sample data pesan dengan ukuran yang berbeda-beda yang ditunjukan pada Tabel 5.6.

Tabel 5.6 Data percobaan kecepatan proses

panjang plain	panjang kunci
161385 bytes	5 byte
321463 bytes	10 byte
481542 bytes	15 byte
641202 bytes	20 byte
801281 bytes	25 byte

5.3.1. Pengujian kecepatan proses enkripsi

Nama Kasus Uji : Kasus Uji Kecepatan Enkripsi *File* MP3

Tujuan Pengujian : Pengujian dilakukan untuk mengetahui kecepatan proses enkripsi dari program dengan menggunakan beberapa sampel data.

Prosedur Uji : Load file MP3 dan masukan kunci kemudian dapat dilihat keterangan waktu yg dibutuhkan untuk mengetahui proses yg terjadi.

Hasil pengujian : Data hasil pengujian kecepatan proses enkripsi data pada sistem ini diperlihatkan pada tabel 5.7

Tabel 5.7. Data hasil pengujian kecepatan proses enkripsi

panjang plain	panjang kunci	waktu (ms)	waktu (ms)	waktu (ms)	rata-rata
161385 bytes	5 bytes	776	701	720	732.3333
321463 bytes	10 bytes	1313	942	960	1071.6667
481542 bytes	15 bytes	1564	1529	1341	1478
641202 bytes	20 bytes	1782	1880	1544	1735.3333
801281 bytes	25 bytes	2287	1995	1949	2077

Sumber: Pengujian

Analisis : Proses enkripsi *file* MP3 pada aplikasi ini bekerja dengan kecepatan rata-rata 1418,8667 milisekon dengan data antara 150000-800000 bytes.

5.3.2. Pengujian Kecepatan Proses Dekripsi

Nama Kasus Uji : Kasus Uji Kecepatan Dekripsi *File* MP3

Tujuan Pengujian : Pengujian dilakukan untuk mengetahui kecepatan proses dekripsi dari program dengan menggunakan beberapa sampel data.

Prosedur Uji : Load file MP3 dan masukan kunci.

Hasil pengujian : Data hasil pengujian kecepatan proses dekripsi pada sistem ini , diperlihatkan pada tabel 5.8.

Tabel 5.8. Data hasil pengujian kecepatan proses dekripsi

chiper	panjang kunci	waktu (ms)	waktu (ms)	waktu (ms)	rata-rata
161385 bytes	5 byte	463	507	584	518
321463 bytes	10 byte	719	716	999	811.3333
481542 bytes	15 byte	1361	1394	1334	1363
641202 bytes	20 byte	1521	1552	1486	1519.6667
801281 bytes	25 byte	1741	1781	1913	1811.6667

Sumber: Pengujian

Analisis : Proses dekripsi pada aplikasi ini bekerja dengan kecepatan rata-rata 1204.7333 milisekon dengan data antara 150000-800000 byte.

5.3.3. Analisis Hasil Pengujian Kecepatan Proses

Dari hasil pengujian kecepatan proses enkripsi didapatkan bahwa waktu untuk melakukan enkripsi mempunyai *range* antara 732ms untuk nilai terendah dan tertinggi 2077ms. Jika dicari hasil kecepatan enkripsi tersebut secara rata-rata adalah sebesar 1418,9ms. Kecepatan proses ini dirasa cukup cepat terutama dalam mengenkripsi *file* MP3 berukuran besar misalnya berukuran di atas 800000byte.

Demikian pula dengan pengujian kecepatan proses dekripsi yang menunjukan *range* antara 518ms sampai 1812ms dengan hasil rata-rata kecepatan 1204,7ms. Baik untuk proses dekripsi dan enkripsi menunjukan bahwa semakin panjang plaintext yang dimasukan atau cipher yang digunakan untuk proses dekripsi dengan kunci yang semakin panjang pula maka waktu yang dibutuhkan akan semakin lama.

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian maka dapat diambil kesimpulan sebagai berikut:

1. Aplikasi ini dapat melakukan enkripsi pada *file* MP3 dan dari hasil pengujian yang telah dilakukan proses enkripsi membutuhkan waktu rata-rata 1418,87 ms dengan data antara 150000-800000 bytes.
2. Aplikasi ini dapat melakukan dekripsi sekaligus memainkan *file* MP3 terenkripsi. Dari hasil pengujian yang telah dilakukan proses dekripsi membutuhkan waktu rata-rata 1204.7 ms dengan data antara 150000-800000 byte.
3. Semakin panjang jumlah kunci pada algoritma RC6, maka tingkat keamanan akan semakin baik, namun waktu yang diperlukan untuk melakukan enkripsi dan dekripsi akan semakin besar.

6.2. Saran

Dalam perancangan dan pembuatan sistem keamanan ini masih terdapat kekurangan dan kelemahan, oleh karena itu masih diperlukan adanya penyempurnaan dalam rangka pengembangan kedepan. Adapun hal yang dapat disempurnakan antara lain:

1. Untuk pengembangan lebih lanjut, dapat dikembangkan pengamanan data pada *audio file* yang lain seperti WAV, AAC.
2. Untuk pengembangan lebih lanjut, dapat digunakan algoritma kriptografi yang telah dibangun ini untuk diimplementasikan pada aplikasi lain.
3. Untuk pengembangan lebih lanjut, *file* MP3 hasil enkripsi dapat diputar/dimainkan di berbagai media *player* selain *Windows Media Player*.

DAFTAR PUSTAKA

Abdurohman, Maman. 2002. *Analisis Performansi Algoritma Kriptografi RC6.* Bandung: Institut Teknologi Bandung.

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

Murad, Affandy. 2009. *Sejarah MP3.* http://my.opera.com/theaffandymurad/blog/show.dml/2878198.

Nugroho, Agung. 2009. Kriptografi. http://www.agungnugroho.net/?p=28.

Raissi, Rassol. 2002. *The Theory Behind Mp3.* http://www.mp3-tech.org/programmer/ docs/mp3_theory.pdf.

Rivest, Ronald L. 1998. *The RC6 Block Cipher.* USA.

Wikipedia. 2010. *MP3*. http://id.wikipedia.org/wiki/MP3.

Wikipedia. 2010. *RC6*. http://en.wikipedia.org/wiki/RC6.