Implementasi Pengamanan MP3 Menggunakan Advanced Encryption Standard

Ismaya Khusnu Wicaksana[*1)], Panji Wisnu Wirawan[*2)] dan Putut Sri Wasito[*3)]

[**]Jurusan Ilmu Komputer/ Informatika, Fakultas Sains dan Matematika,
Universitas Diponegoro
[1)]nuekiseki@live.com, [2)]maspanji@undip.ac.id, [3)]pututsw@undip.ac.id

Abstrak

Format MP3 merupakan salah satu format suara yang populer. Format ini tidak memiliki implementasi keamanan sehingga dapat menimbulkan beberapa dampak negatif. Salah satu dampak negatifnya adalah resiko terhadap keamanan MP3 yang berisi informasi rahasia. Kelemahan tersebut dapat dihilangkan dengan menerapkan proses enkripsi pada file MP3. Algoritma kriptografi Advanced Encryption Standard (AES) dipilih karena pada algoritma ini belum ditemukan celah keamanan, dipelihara dengan baik oleh NIST dan merupakan algoritma yang sering digunakan dalam implementasi keamanan. Implementasi Pengamanan MP3 Menggunakan AES merupakan suatu bentuk implementasi pengamanan dengan melakukan proses enkripsi terhadap file MP3 tanpa merusak struktur file MP3 sehingga file tersebut masih dapat dijalankan pada beragam MP3 player dan menghasilkan noise saja. File MP3 dapat dikembalikan ke bentuk yang sama seperti sedia kala dengan melakukan proses dekripsi. Hasil tersebut dapat dicapai dengan melakukan proses enkripsi selektif terhadap data yang terletak di dalam setiap frame pada file MP3. Implementasi ini berupa aplikasi yang dibuat dengan menggunakan bahasa pemrograman C++ dengan menggunakan pendekatan Pemrograman Berbasis Objek (PBO) serta penggunaan framework Qt. Aplikasi ini diuji dengan menggunakan beberapa MP3 dengan parameter-parameter yang berbeda serta metode pengujian blackbox testing. Dari hasil pengujian, aplikasi yang telah dibuat berfungsi sesuai dengan tujuan dan harapan yaitu dapat melindungi file MP3 serta mengembalikannya seperti sedia kala.

***Kata kunci** : Advanced Encryption Standard (AES), MP3, Keamanan, C++, Kriptografi*

Abstract

MP3 format is one of popular audio formats. Having no security implementation, this audio format has a few downsides. One of the downsides are security risk of classified digital audio data in the MP3. Those downsides could be eliminated by applying encryption process to the MP3. Advanced Encryption Standard (AES) encryption algorithm selected because it has no security flaw, well maintained by NIST and it's a more common algorithm used in security implementation. Implementation of MP3 Security using Advanced Encryption Standard (AES) is a form of MP3 security implentation where encryption process preserves the structure of MP3 file itself. It is resulting in an encrypted MP3 file which is still playable on various MP3 players although it only generates noises. The MP3 file can be decrypted to return to its original form. This implementation is taking form of a computer application that has been created using C++ language with Object-Oriented Programming (OOP) approach and using Qt framework. The application was tested with a few MP3 each with different parameters and blackbox testing method. From the testing, its concluded that the application working as expected from encryption process of MP3 data to decryption process.

***Keywords** : Advanced Encryption Standard (AES), MP3, Security, C++, Cryptography*

1 PENDAHULUAN

MP3 merupakan salah satu format data suara yang populer (International Federation of the Phonographic Industry, 2013). Selain ukurannya yang kecil, MP3 juga mempunyai reproduksi suara yang cukup bagus serta implementasi algoritma yang relatif mudah.

Format MP3 tidak mempunyai implementasi keamanan sehingga menimbulkan beberapa dampak negatif. Salah satu dampak negatifnya adalah resiko mengenai keamanan data suara jika file MP3 mengandung informasi yang bersifat rahasia. Dampak lainnya adalah meningkatnya pembajakan file MP3 yang disebabkan mudahnya proses duplikasi file ini. Kelemahan tersebut dapat dihilangkan dengan melakukan proses enkripsi pada file MP3. Proses enkripsi pada file MP3 merupakan dasar untuk membentuk sebuah sistem berbasis Digital Rights Management (DRM).

Proses enkripsi pada MP3 menggunakan algoritma Advanced Encryption Standard (AES) dapat melindungi data suara dengan mengacak setiap sampel suara sehingga keamanan data tersebut lebih terjamin. Pemilihan algoritma AES untuk digunakan pada proses enkripsi MP3 didasarkan pada tingkat keamanan yang diberikan oleh algoritma ini.

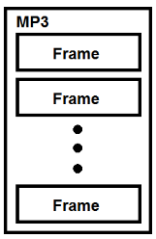
National Institute of Standards and Technology (NIST) secara terus menerus melakukan analisis dan evaluasi ulang terhadap algoritma AES dan akan melakukan revisi jika ditemukan kelemahan matematis pada algoritma ini [1]. Jadi algoritma AES merupakan suatu algoritma yang dipelihara dengan baik sehingga tingkat keamanan yang diberikan oleh algoritma ini tetap terjaga.

Dalam jurnal Bismita Gadayanak dan Chittaranjan Pradhan disebutkan bahwa aplikasi pengamanan MP3 yang beliau kembangkan melakukan proses enkripsi pada tahap kompresi [2]. Perbedaan antara metode jurnal tersebut dengan metode pada tugas akhir ini terletak pada tahap enkripsi. Penulis menggunakan pendekatan struktur file dan melakukan enkripsi pada setiap body frame pada MP3 untuk mempercepat proses enkripsi.

2 METODE

2.1 MP3

MP3 adalah suatu teknik kompresi suara dan merupakan singkatan dari "MPEG-1 Layer III" dan MPEG merupakan singkatan dari "Moving Picture Experts Group". Pada awalnya dikembangkan oleh anggota komunitas teknologi di Jerman bernama Fraunhofer dan Thomson, sekarang sudah dikodekan oleh International Standards Organization [3]. Struktur data pada file MP3 dijelaskan pada gambar 1.



Gambar 1. Struktur Data *File* MP3

Frame MP3

Semua file MP3 tersusun dari frame dan setiap frame terdapat 1152 sampel suara yang berakhir selama 26 ms. Data suara pada setiap frame dibagi menjadi 2 granule dan setiap granule mengandung 576 sampel suara.

Dekoder file MP3 melakukan tindakan tertentu jika terjadi kesalahan data pada suatu frame dengan mengganti frame yang rusak dengan frame sebelumnya atau frame kosong. Setiap frame dapat diidentifikasi menjadi dua bagian utama yaitu header dan body. Header mempunyai ukuran tetap sedangkan body mempunyai ukuran dan komponen yang bevariasi sesuai dengan parameter yang terdapat pada header.

Ukuran frame = ((144 * bitrate) / sample_frequency) + padding ……..(1)

Variabel-variabel yang digunakan dalam perhitungan rumus ukuran frame dapat dijelaskan sebagai berikut:

1. **Ukuran frame**: ukuran dari frame penyusun file MP3 dalam satuan byte
2. ***Bitrate***: ukuran besar arus data dengan satuan bit yang didapatkan dengan konversi bit-bit data bitrate yang terdapat pada header terhadap tabel ketetapan bitrate (tabel 1).
3. ***Sample frequency***: ukuran besar frekuensi sampel dengan satuan hz didapatkan dengan melakukan konversi bit-bit data frequency pada header terhadap tabel ketetapan frequency (tabel 2).
4. ***Padding***: informasi penambah ukuran frame. Jika bit padding bernilai 1 pada header maka besar ukuran frame bertambah 1 *byte*.

Tabel 1. Konversi kode *bitrate* menjadi besar arus data (Kb/s) [7]

Bit	MPEG-1 Layer III Bitrate (Kb/s)
0 0 0 1	32
0 0 1 0	40
0 0 1 1	48
0 1 0 0	56
0 1 0 1	64
0 1 1 0	80
0 1 1 1	96
1 0 0 0	112
1 0 0 1	128
1 0 1 0	160
1 0 1 1	192
1 1 0 0	224
1 1 0 1	256
1 1 1 0	320

Tabel 2. Konversi kode biner frekuensi

Bit	Frekuensi
0 0	44100 Hz
0 1	48000 Hz
1 0	32000 Hz
1 1	reservasi

Ukuran frame digunakan dalam aplikasi ini untuk melakukan pemisahan suatu frame dengan frame lainnya. Jika hasil perhitungan ukuran frame merupakan nilai desimal maka dilakukan pembulatan ke bawah sehingga hasilnya merupakan bilangan bulat.

2.2 KRIPTOGRAFI

Buku "Handbook of Applied Cryptography" menyebutkan bahwa "Kriptografi adalah studi mengenai teknik matematis yang berhubungan dengan keamanan informasi seperti kerahasiaan, integritas data, otentikasi entitas, dan otentikasi sumber data" [5]. Wade Trappe dan Lawrence C. Washington pada bukunya "Introduction to Cryptography with Coding Theory" mendefinisikan kriptografi modern sebagai area yang menitikberatkan pada matematika, ilmu komputer dan kecerdasan [8]. Jadi berdasarkan kedua sumber tersebut, dapat disimpulkan bahwa kriptografi modern merupakan penerapan teknik matematis dengan menggunakan teknologi komputer terhadap sumber data sehingga keamanan dan keaslian informasi pada data tersebut terjaga. Dalam kriptografi, data dalam bentuk aslinya disebut sebagai plaintext, sedangkan data yang telah terenkripsi disebut sebagai ciphertext.

Block Cipher

Block cipher adalah fungsi yang memetakan n-bit blok plaintext menjadi n-bit blok cipher-teks. Panjang blok

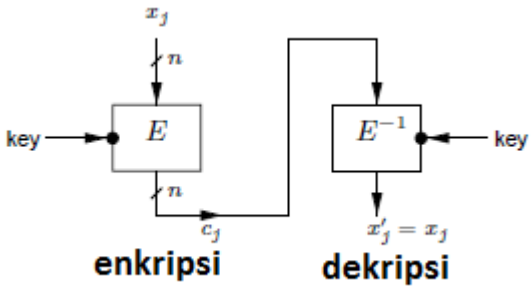
dilambangkan dengan simbol n. Dapat dipandang sebagai *substitution cipher* dengan ukuran karakter yang besar. Besar *plaintext* disesuaikan dengan kelipatan panjang blok sehingga tidak terjadi ekspansi data.

Fungsi enkripsi pada *block cipher* bersifat *one-to-one* untuk menghasilkan dekripsi data yang unik. Sebuah *block cipher* melakukan enkripsi terhadap blok plaintext yang berukuran tetap. Untuk *plaintext* yang berukuran lebih besar dari n maka pendekatan paling mudah adalah dengan membagi *plaintext* menjadi blok dan melakukan enkripsi terhadap masing-masing blok secara terpisah. Teknik ini dinamakan ECB (*Electronic Codebook*) dan memiliki banyak kelemahan pada sebagian besar terapannya sehingga memotivasi penggunaan metode lain. Empat mode yang paling umum digunakan adalah ECB, CBC, CFB, dan OFB dijelaskan sebagai berikut :

Electronic Codebook (ECB)

Pada mode operasi ECB, *plaintext* yang sama menghasilkan *ciphertext* yang sama. Dekripsi masing-masing blok tidak tergantung pada blok lain. Kesalahan enkripsi maupun dekripsi tidak merambat ke blok lain.

Karena karakteristiknya ECB tidak cocok digunakan untuk *plaintext* yang ukurannya lebih panjang dari ukuran blok. Proses enkripsi dan dekripsi pada ECB dijelaskan pada gambar 2.

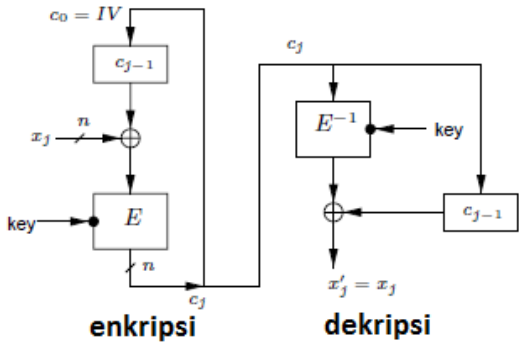


Gambar 2. Proses Enkripsi dan Dekripsi ECB [1]

Cipher-Block Chaining (CBC)

Pada mode operasi CBC, *plaintext* yang sama dapat menghasilkan *ciphertext* yang berbeda karena adanya input dari blok lain. Dekripsi masing-masing blok membutuhkan hasil dekripsi blok sebelumnya. Kesalahan enkripsi merambat ke blok selanjutnya, sedangkan kesalahan dekripsi hanya berdampak pada blok tersebut dan satu blok selanjutnya.

Proses enkripsi dan dekripsi pada mode operasi CBC dapat dijelaskan pada gambar 3.

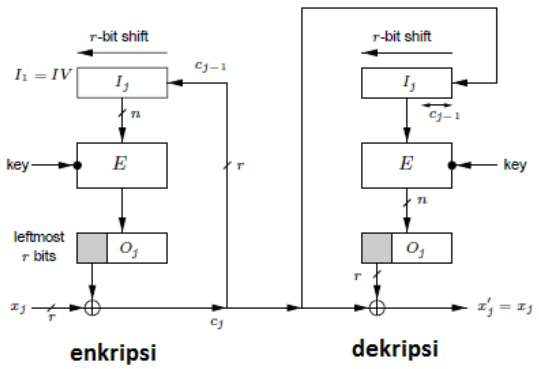


Gambar 3. Proses Enkripsi dan Dekripsi CBC [1]

Cipher Feedback (CFB)

Pada terapan tertentu dimana data harus dikirimkan dengan jeda waktu minimal maka penggunaan CBC tidak dapat digunakan karena CBC menunggu data sejumlah n sebelum memprosesnya. Sedangkan pada mode CFB data sejumlah r dapat dienkripsi meskipun ukuran r < n.

Proses enkripsi dan dekripsi pada mode operasi CFB dapat dijelaskan pada gambar 4.

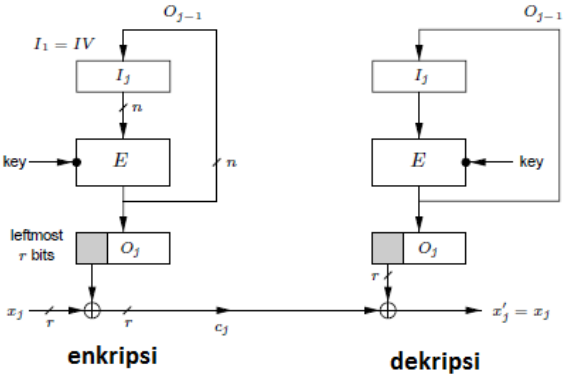


Gambar 4. Proses Enkripsi dan Dekripsi CFB

Output Feedback (OFB)

Mode operasi OFB digunakan saat perambatan kesalahan harus dihindari. Mode ini merupakan pengembangan dari mode CFB dimana proses enkripsi dapat dilakukan pada ukuran blok yang bervariasi.

Proses enkripsi dan dekripsi pada mode operasi OFB dapat dijelaskan pada gambar 5.



Gambar 5. Proses Enkripsi dan Dekripsi CFB

Algoritma *Advanced Encryption Standard (AES)*

Advanced Encryption Standard (AES) merupakan definisi standard dari algoritma kriptografi Rijndael, yang merupakan block *cipher* simetris yang melakukan proses kriptografi terhadap blok-blok data dengan ukuran 128 bit dan menggunakan kunci sepanjang 128 bit, 192 bit atau 256 bit. Rijndael didesain untuk menggunakan panjang blok dan panjang kunci yang lebih bervariasi tetapi tidak termasuk dalam standard AES.

Algoritma AES dapat menggunakan kunci dengan tiga ukuran berbeda dan dapat diacu sebagai "AES-128" untuk kunci 128 bit, "AES-192" untuk kunci 192 bit, dan "AES-256" untuk kunci 256 bit. [1]

Tabel 3. Spesifikasi Algoritma AES [1]

	Panjang Kunci	Panjang Blok	Jumlah Round
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

AES Cipher

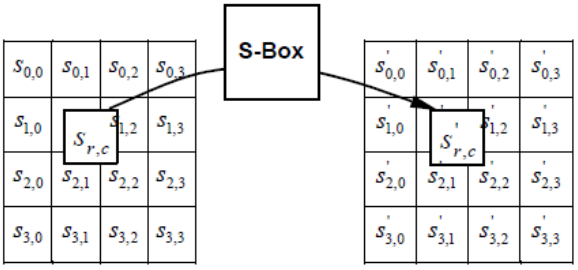
Pada awal *cipher*, data input dimasukkan ke dalam *state array*. *Cipher* dideskripsikan sesuai dengan gambar 6. Masing-masing transformasi yaitu *SubBytes*(), *ShiftRows*(), *MixColumns*(), dan *AddRoundKey*() melakukan transformasi terhadap *state a*rray, sedangkan *array* w[] berisi informasi *state key*.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])                  // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                              // See Sec. 5.1.1
      ShiftRows(state)                             // See Sec. 5.1.2
      MixColumns(state)                            // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Gambar 6. Algoritma Cipher AES [1]

Transformasi *SubBytes*()

Transformasi *SubBytes*() merupakan substitusi *byte* yang tidak linier dan dioperasikan pada masing-masing *byte* dari *state array* dengan menggunakan tabel substitusi (S-Box).



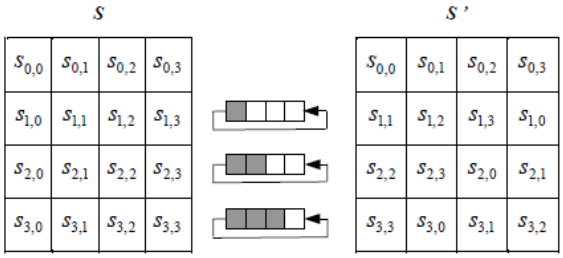
Gambar 7. Proses Subsitusi pada *SubBytes*() [1]

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar 8. S-Box

Transformasi *ShiftRows*()

Transformasi *ShiftRows*() melakukan pergeseran terhadap 3 baris terakhir dari *state array* yang jumlah pergeserannya bertambah sesuai dengan nomor baris. Transformasi perubahan state array dijelaskan pada gambar 9.



Gambar 9. Proses Transformasi *ShiftRows()* [1]

Transformasi *MixColumns*()

Transformasi *MixColumns*() beroperasi pada *state array* pada basis kolom. Proses perhitungan untuk setiap *byte* yang dilakukan dapat dijelaskan oleh formula pada gambar 10.

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

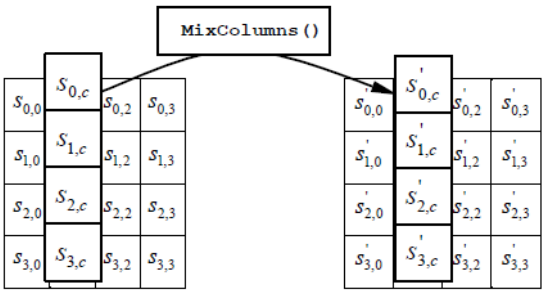
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

Gambar 10. Proses Operasi Pada Setiap Kolom *State Array* [1]

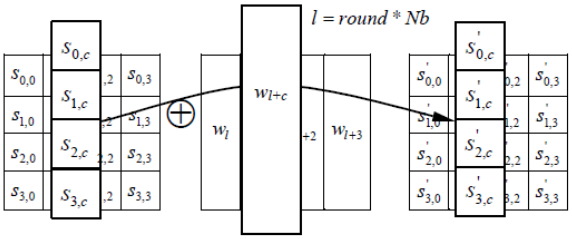
Gambar 11 menjelaskan bahwa *MixColumns*() beroperasi pada basis kolom per kolom.



Gambar 11.Transformasi *MixColumns*() [1]

Transformasi *AddRoundKey*()

Pada Transformasi *AddRoundKey*(), *Round Key* ditambahkan ke *State Array* dengan menggunakan operasi bit XOR. Proses transformasi *AddRoundKey*() dapat dijelaskan pada gambar 12.



Gambar 12. Transformasi *AddRoundKey*()

Key Expansion

Algoritma ekspansi kunci pada AES dapat dijelaskan pada gambar 13.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end

Note that Nk=4, 6, and 8 do not all have to be implemented;
they are all included in the conditional statement above for
conciseness.   Specific implementation requirements for the
Cipher Key are presented in Sec. 6.1.
```

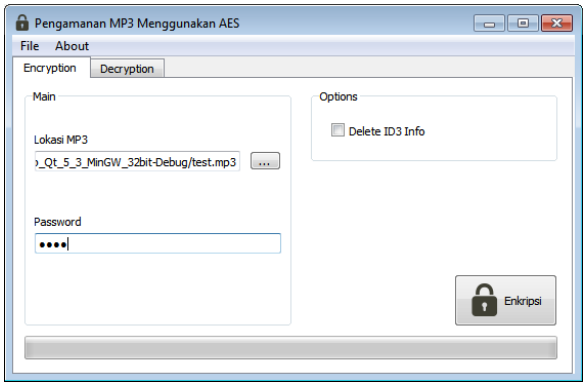
Gambar 13. Algoritma Ekspansi Kunci AES

3 IMPLEMENTASI

Implementasi pada aplikasi ini terdiri dari dua alur utama yaitu alur enkripsi dan alur dekripsi.

Menu Enkripsi

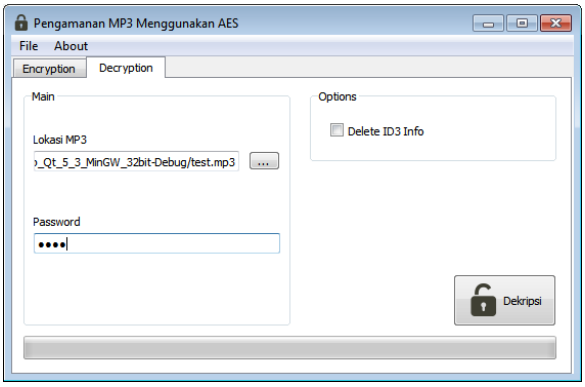
Pengguna dapat melakukan proses enkripsi pada berkas MP3 menggunakan menu sesua pada gambar 14.



Gambar 14. Menu Enkripsi

Menu Dekripsi

Pengguna dapat melakukan proses dekripsi pada berkas MP3 menggunakan menu sesuai pada gambar 15.



Gambar 15. Menu Dekripsi

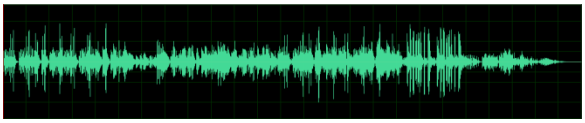
4 PENGUJIAN

Pengujian aplikasi dilakukan dengan metode blackbox testing sehingga data pengujian yang dihasilkan lebih mengarah dari sudut pandang pengguna. Dari hasil pengujian yang telah dilaksanakan dapat diketahui bahwa aplikasi berjalan sesuai harapan.

Pengujian Terhadap Test01.mp3 (TC_1)

Pada pengujian ini aplikasi melakukan proses enkripsi dan dekripsi terhadap "Test01.mp3" yang memiliki bitrate bervariasi (VBR) 192 Kbps, berisi suara instrumen musik saja dan memiliki informasi ID3 berupa judul, nama artis, serta nama album.

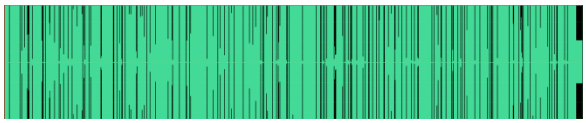
Pengujian dimulai dengan medapatkan bentuk gelombang suara awal dari *file* "Test01.mp3" dengan menggunakan aplikasi Adobe Audition, hasil yang didapatkan seperti pada gambar 16.



Gambar 16. Bentuk Gelombang *File* Test01.mp3

Pengujian dilanjutkan dengan melakukan enkripsi terhadap *file* "Test01.mp3" menggunakan dengan

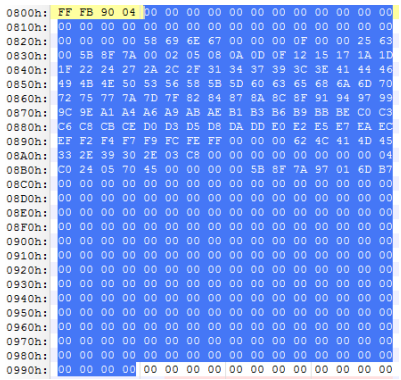
password "abc" dengan *output file* "Enk01.mp3". Setelah proses enkripsi selesai, *file* "Enk01.mp3" dibuka pada aplikasi Adobe Audition maka didapatkan gambar bentuk gelombang suara terenkripsi sesuai pada gambar 17.



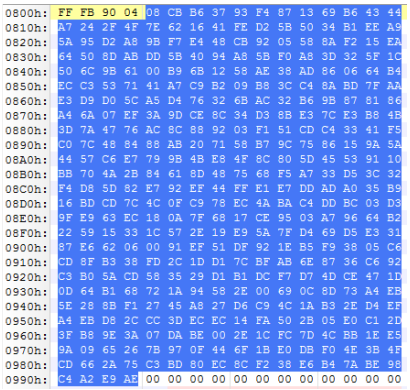
Gambar 17. Bentuk Gelombang *File* Enk01.mp3

Perbandingan antara gambar 16 dan gambar 17 menunjukkan bahwa gelombang suara yang terdapat dalam file "Enk01.mp3" telah mengalami pengacakan yang disebabkan oleh enkripsi terhadap setiap frame body. Berkas tersebut masih dapat dibuka dalam program pengolah suara Adobe Audition yang menandakan bahwa header yang terdapat pada setiap frame tidak mengalami pengacakan.

Potongan sampel data dalam bentuk kode heksadesimal dari *frame* pertama pada *file* awal "Test01.mp3" dan file terenkripsi "Enk01.mp3" digunakan untuk memperkuat bukti pengacakan sesuai pada gambar 18 dan 19. Kode heksadesimal tersebut diperoleh dengan membuka masing-masing berkas pada aplikasi Sweetscape 010 Editor.

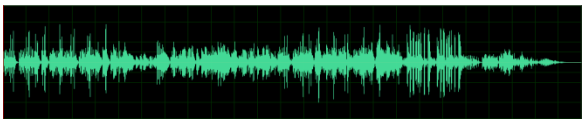


Gambar 18. Potongan *Frame* Pertama Pada Test01.mp3



Gambar 19. Potongan *Frame* Pertama Pada Enk01.mp3

Pengujian dilanjutkan dengan melakukan proses dekripsi terhadap *file* "Enk01.mp3" dengan menggunakan *password* "abc" dan *output* "Dek01.mp3". Setelah proses dekripsi selesai, *file output* "Dek01.mp3" dibuka pada aplikasi Adobe Audition maka didapatkan gambar bentuk gelombang terdekripsi seperti pada gambar 20.



Gambar 20. Gelombang Berkas Dek01.mp3

Perbandingan antara gambar 16 dan gambar 20 menunjukkan bahwa *file* awal "Test01.mp3" dan *file* hasil dekripsi "Dek01.mp3" memiliki bentuk gelombang suara yang sama. Hal ini membuktikan bahwa proses enkripsi dari *file* awal "Test01.mp3" menjadi *file* terenkripsi "Enk01.mp3" dan proses dekripsi dari *file* terenkripsi "Enk01.mp3" menjadi *file* terdekripsi "Dek01.mp3" berjalan dengan benar.

Tabel Hasil Pengujian

Hasil pengujian dapat disajikan dalam bentuk yang lebih ringkas berupa tabel. Tabel 4 menjelaskan mengenai jumlah frame yang mengalami perubahan akibat proses

enkripsi maupun dekripsi dibandingkan dengan frame pada berkas awal.

Tabel 4. Jumlah Perbedaan *Frame* Terhadap Data Awal

Nama	Jml. Frame	Beda Awal	Beda Hasil Enkrip	Beda Hasil Dekrip
Test01	9572	0	9572	0
Test02	3922	0	3922	0
Test03	8180	0	8180	0
Test04	2753	0	2753	0
Test05	10771	0	10771	0

5 KESIMPULAN

Setelah melaksanakan semua proses dalam pembuatan aplikasi tugas akhir ini dapat diambil kesimpulan sebagai berikut:

1. Aplikasi yang dapat melindungi MP3 dengan cara melakukan proses kriptografi menggunakan algoritma AES telah selesai dibangun.
2. Teknik enkripsi pada MP3 yang diterapkan aplikasi ini tidak merusak struktur data header pada setiap frame sehingga berkas MP3 masih diterima oleh media player meskipun hanya menghasilkan noise saat proses pemutaran.
3. Aplikasi telah diuji dan memenuhi target pengujian dengan menggunakan parameter input berupa lima buah file MP3 dengan spesifikasi yang berbeda.

REFERENSI

[1]. Federal Information Processing Standard Publication, 2001. *Advanced Encryption Standard*. [Online] Available At: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf [Accessed 18 April 2013].

[2]. Gadanayak, B. & Pradhan, C, 2011. *Selective Encryption of MP3 Compression*. International Journal of Computer Applications.

[3]. Hacker, S., 2000. *Mp3: The Definitive Guide*. California: O'Reilly.

[4]. International Federation of the Phonographic Industry, 2013. *IFPI Digital Music Report 2013*. [Online] Available At: http://www.ifpi.org/content/library/DMR2013.pdf [Accessed 18 April 2013].

[5]. Menezes, A.J., Oorschot, P.C. & Vanstone, S.A., 1996. *Handbook of Applied Cryptography*. CRC Press.

[6]. Pressman, R.S., 2010. *Software Engineering: A Practitioner's Approach Seventh Edition*. New York: McGraw-Hill.

[7]. Raissi, R., 2002. *The Theory Behind Mp3*. [Online] Available At: http://www.mp3-tech.org/programmer/docs/mp3_theory.pdf [Accessed 18 April 2013].

[8]. Trappe, W. & Washington, L.C., 2006. *Introduction to Cryptography with Coding Theory: 2nd Edition*. New Jersey: Pearson Prentice Hall.