RANCANG BANGUN DIGITAL *AUDIO SHARE* DAN *PLAYER* DENGAN KEAMANAN DATA *ADVANCED ENCRYPTION STANDARD* BERBASIS ANDROID

Yuyon Alif Masruri[1], Dr. Eng. Herman Tolle[2] S.T, M.T, Aswin Suharsono[2] S.T, M.T

[1]Mahasiswa, [2]Dosen Pembimbing

Program Studi Ilmu Komputer/ Informatika Universitas Brawijaya

yuyonallif@gmail.com

ABSTRACT

Digital audio technology makes people easier to access music and audio that would be possessed and be heard. However, with such ease to access music and video allows people to do the copyright infringement by means distribute music and audio illegally. As we know, mobile device has feature to share files anywhere include digital audio just by using internet. Because of many illegal spread of digital audio, people needs a software that has security to save data. By using AES cryptographic technology is expected to provide security and pleasantly use software in daily. Audiosafe software design used a functional stage to design it based on specification requirement result and non-functional design by designing cryptography technology, encryption and decryption. Encryption and decryption performance in Audiosafe has been implemented by using three parameters such as cryptography performance, authenticity of data, and security. The software implementation was developing based on design of the previous development and it implements with Java programming language and uses Android Developer Tools named Android Studio to make an android software in smartphone android. Software testing in this software is using whitebox, blackbox, and testing data security. The result of testing unit can be concluded that implementation and functionality has qualified of software requirement. In performance testing can be concluded that performance time of cryptography tolerable based on APDEX with a minimum data size. Authenticity of data testing has a conclusion that size and file content can be returned to its original when it processes decryption from file cipher digital in encryption audio result. Security testing can be concluded that audio file cannot be played when the user doesn't have privilege of digital audio uploader

Keyword: AES, cryptography, digital audio, mobile application.

ABSTRAK

Teknologi *digital* audio memudahkan masyarakat mengakses musik maupun audio yang ingin dimiliki dan didengar. Namun dengan kemudahan tersebut, memungkinkan pelanggaran hak cipta dari karya seni terhadap masyarakat dalam menyebarluaskan teknologi *digital audio* yang berada dibawah perlindungan. Kemudahan penggunaan perangkat bergerak memungkinakan setiap pengguna bisa berbagi antar file kemana saja, tidak terlepas juga dengan file *digital audio* hanya dengan menggunakan jaringan internet. Oleh karena itu, perlu dikembangkan perangkat lunak perangkat bergerak yang memiliki keamanan data didalamnya. Dengan menggunakan teknologi kriptografi *AES*, diharapkan dapat memberikan keamanan dan kenyamanan dakam penggunaan perangkat lunak. Perancangan perangkat lunak Audiosafe menggunakan tahap perancangan fungsional berdasarkan hasil spesifikasi kebutuhan dan perancangan non fungsional atau keamanan data dengan merancang kinerja dari sisi kriptografi yaitu enkripsi dan dekripsi pada perangkat lunak berdasarkan 3 parameter yaitu waktu performansi kriptografi, keaslian data dan *security*. Implementasi dari perangkat lunak dikembangkan berdasarakan perancangan padah tahap sebelumnya dan diimplementasikan dalam bahasa pemrograman Java dengan development tool *Android Developer Tool* yaitu *Android Studio* dan diimplementasikan dalam *device smartphone Android*. Pengujian menggunakan teknik *Whitebox*, *Blackbox* dan Pengujian keamanan data. Dari hasil pengujian unit dapat disimpulkan bahwa algoritma dan unit perangkat lunak telah memenuhi kebutuhan fungsional. Hasil pengujian validasi dapat disimpulkan bahwa implementasi dan fungsionalitas memenuhi kebutuhan perangkat lunak. Pada pengujian waktu performansi kriptografi dapat disimpulkan bahwa performansi kriptografi menghasilkan waktu yang *tolerable* berdasarkan APDEX dengan syarat minimal ukuran data. Pengujian keaslian data menghasilkan kesimpulan bahwa ukuran dan konten *file* dapat kembali ke bentuk semula ketika mengalami proses dekripsi dari *file cipher digital audio* hasil enkripsi. Pengujian *security* dapat disimpulkan bahwa berdasarkan proses pada perangkat lunak, *file digital audio* tidak dapat diputar ketika belum diberikan *privilege* dari *user* pengunggah *digital audio*.

Kata kunci: *AES*, kriptografi, *digital audio*, perangkat lunak perangkat bergrerak.

1. Pendahuluan

1.1 Latar Belakang

Teknologi memungkinkan musik untuk direkam dan dihadirkan kembali dalam bentuk suara tanpa perlu menghadirkan pemusik. Kehadiran teknologi rekaman suara tersebut menghasilkan bentuk yang dikenal dengan istilah digital audio. *Digital audio* merupakan penyajian dari suara asli yang terrekam. Dengan kemunculan teknologi *digital audio*, maka masyarakat dimudahkan untuk mengakses musik-musik yang ingin dimiliki. Akan tetapi, persebaran musik dan rekaman suara dalam bentuk

digital audio memungkinkan masyarakat mengakses dan menyebarluaskan hingga melanggar hak cipta dari karya seni musik tersebut. Hal tersebut dikenal dengan istilah pembajakan. Pembajakan telah banyak terjadi di Indonesia dan sangat memperihatinkan. Menurut data ASIRI, angka pembajakan tercatat sekitar 96% sejak tahun 2008 [1]. Berdasarkan banyaknya kasus yang tercatat, dibutuhkan suatu keamanan penyebaran informasi berupa kriptografi. Kriptografi adalah salah satu cara untuk mengamankan data atau teknik yang prinsip utamanya adalah untuk melindungi keamanan informasi [2]. Dengan adanya kriptografi, maka kerahasiaan informasi akan tetap bisa terjaga tanpa dengan bisa mudah dicuri, dipastikan dikirimkan dengan benar, dan data dapat terjaga dari adanya pencurian data dari pihak ketiga.

Hal ini mendukung penulis merancang dan membangun *sistem digital audio share* dan *player* dengan kemanan data. Pada penelitian ini digunakan Kriptografi dengan algoritma AES (*Advanced Encryption Standard*) dan akan diimplementasikan pada perangkat bergerak (*mobile application*) untuk mengamankan data digital audio pada perangkat lunak music player, serta mengukur pengaruh pengimplementasian keamanan data dari segi waktu eksekusi kriptografi, perbandingan isi konten data enkripsi dan dekripsi (keaslian data), dan menguji tingkat *security* dengan cara dapat atau tidaknya dari persetujuan user untuk memutar satu musik digital tersebut terhadap perangkat lunak itu sendiri.

1.2 Rumusan Masalah

1. Bagaimana rancangan dan hasil implementasi perangkat lunak pemutar dan berbagi digital audio dengan pengamanan data menggunakan kriptografi *Advanced Encryption Standard* (AES) berbasis perangkat bergerak Android?
2. B Bagaimana kinerja perangkat lunak ditinjau dari sisi waktu eksekusi kriptografi, keaslian data dan *security* data?

2. Landasan Kepustakaan

Kriptografi, berguna untu kerahasiaan informasi akan tetap bisa terjaga tanpa dengan bisa mudah dicuri, dipastikan dikirimkan dengan benar, dan data dapat terjaga dari adanya pencurian data dari pihak ketiga. Salah satu algoritma kriptografi adalah *Advanced Encryption Standard* (AES). *AES* menggunakan memori yang rendah dan hasil keamanan yang bagus dalam proses kriptografinya [3]. Dipillih algoritma AES dibandingkan RSA dan DES karena didapatkan hasil dari uji coba dan pengomparasian bahwa algoritma AES lebih baik dari DES dan RSA [4].

2.1 *Media Player*

Media Player merupakan *software* yang memainkan *digital audio*, video atau animasi di dalam komputer. *Media player* menyediakan beberapa fitur, diantaranya pengguna dapat mengatur koleksi multimedia, memutar lagu dan film *rip CD* lagu ke dalam format MP3 dan format *digital audio* lainya, *burn-ing CD*, mendengarkan radio internet, download kontern dari toko musik *online* dan *streaming* konten dari internet. [5]

Salah satu *media player* untuk melakukan tugas terutama untuk memutar *digital audio* adalah *Android Multimedia*, dengan *support* untuk memutar berbagai jenis tipe media sehingga dengan mudah diintegrasikan dalam perangkat lunak. *Android Multimedia* dapat memutar *file* dari local storage maupun dari internet.

Dalam Sistem Operasi Android sudah disediakan *mediaplayer library*, yang memungkinkan untuk memutar dan mengendalikan seriap file audio di *background*, dengan menggunakan metode kontrol mediaplayer seperi *play(), stop()* dan *seekto()*. Pengembang hanya harus mengaplikasikannya dengan benar di dalam Komponen UI.

2.2 *Library* Kriptografi AES dalam bahasa *Java*

Advanced Encryption Standard (AES) merupakan penyempurnaan dari *Data Encryption Standard* (DES) yang bagian dari kode simetris dan pertama kali dipublikasikan oleh NIST (*National Institute of Standard and Technology*) pada tahun 2001 [6, p. 2, 6]. AES menggunakan kunci simetris untuk melakukan proses enkripsi dan dekripsi. Ukuran kunci dalam algoritma AES bervariasi yaitu 128, 192, dan 256 bit. Perbedaan panjang kunci nantinya mempengaruhi jumlah perputaran yang akan diimplementasikan dalam AES.

Dalam *Java* sudah disediakan library untuk penggunaan kriptografi yang bernama *Java Cryptography Extenseion* (*JCE*) yang sudah diintegrasikan pada *Java Development Kit* (*JDK*)versi 1.4.x keatas. Sehingga tdak diperlukannya lagi dalam membangun dan membuat algoritma AES mulai dari awal. Lokasi *JCE* terletak pada paket *javax.crypto* Untuk pengimplimentasianya dapat dilakukan dengan memanggil fungsi kriptografi dari *JCE* dengan Kode Sumber 1.

Kode Sumber 1 Import Kriptografi di JCE

```
1  import javax.crypto.*;
2  import javax.crypto.spec.*;
```

dua baris Kode Sumber 2.2 menjelaskan untuk mengimport fungsi kriptografi dari *JCE* kedalam klas untuk pengimplementasiannya.

Kemudian dalam pengimplementasiannya, diperlukan inisiasi atau pemberian nilai awal algoritma apa yang akan digunakan untuk kriptografi yang disediakan oleh *JCE*. Dalam penggunaan kriptografi AES maka inisiasi dilakukan seperti Kode Sumber 2.

Kode Sumber 2 Inisiasi kriptografi AES

```
1  Chiper chiper = Chiper.getInstace ("AES");
2  cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
```

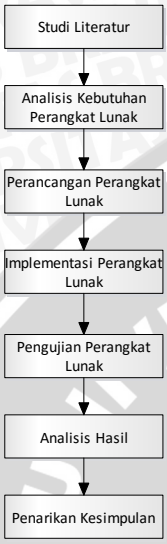
Library AES dalam bahasa *Java* mencakup proses standar enkripsi dan dekripsi, dan penggunaan fungsi tipe kriptografi apa yang akan digunakan.

3. Metodologi

Metodologi penelitian menjelaskan metode yang digunakan dalam perancangan dan pembangunan *Audio Share* dan *Player* dengan Keamanan Data *Advanced Encryption Standard* Berbasis Android. Tahapan metodologi penelitian dapat dilihat pada Gambar 1.

Dimulai dari

1. Studi Literatur
2. Analisa Kebutuhan Perangkat Lunak
3. Perancangan Perangkat Lunak
4. Implementasi Perangkat Lunak
5. Pengujian Perangkat Lunak
6. Analisis Hasil
7. Penarikan Kesimpulan

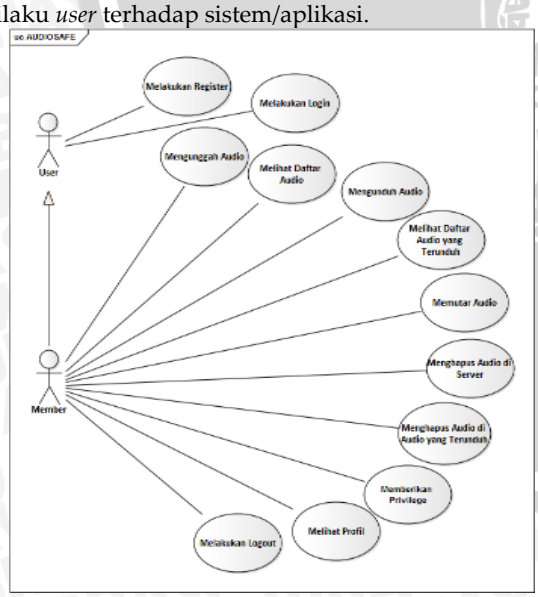


Gambar 1 Metodologi Penelitian

4. Spesifikasi Kebutuhan dan Perancangan Perangkat Lunak.

4.1 *Use Case*

Use case adalah salah satu diagram untuk memodelkan perilaku *user* terhadap sistem/aplikasi.

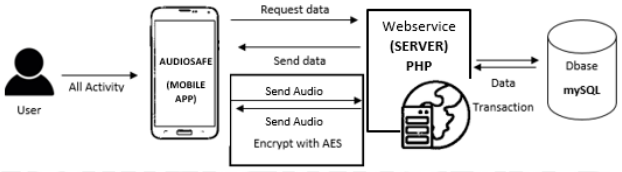


Gambar 2 Diagram *use case Audiosafe*

4.2 Pemodelan dan Perancangan Perangkat Lunak

4.2.1 Perancangan sistem

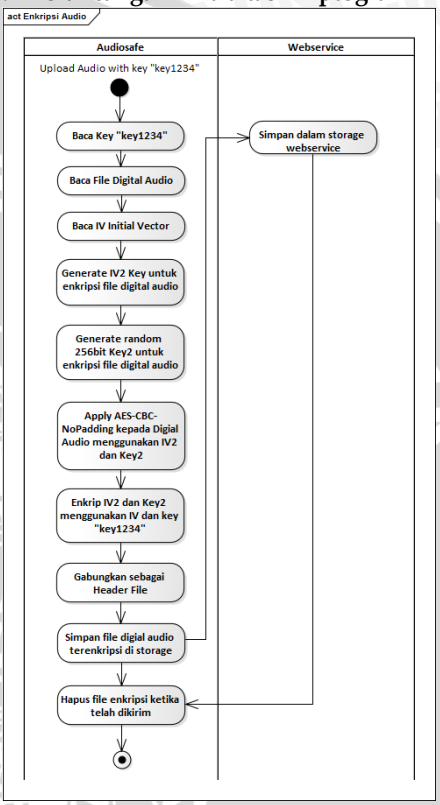
Pada Gambar 3 dijelaskan bagaimana aplikasi *Audiosafe* adalah perangkat lunak yang dijalankan dalam *smartphone* untuk menjalankan segala kebutuhan fungsionalitas *user*.



Gambar 3 Perancangan Sistem

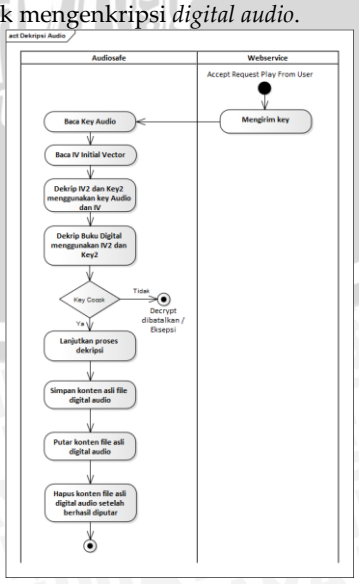
4.2.2 Perancangan Keamanan Data

4.2.1.1 Perancangan Aktivitas Kriptografi *AES*



Gambar 4 Perancangan Enkripsi Kriptografi *AES*

Perancangan keamanan data gambar 4 digunakan untuk mengenkripsi *digital audio*.



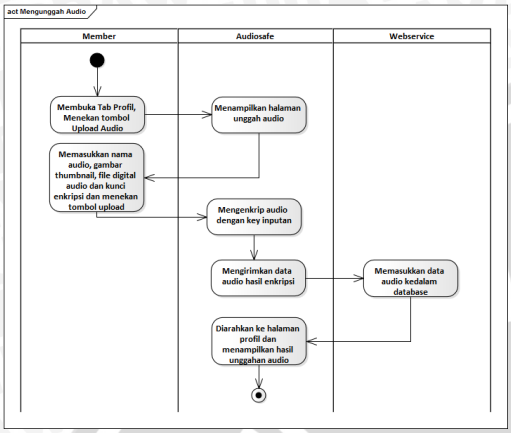
Gambar 5 Perancangan Dekripsi Kriptografi *AES*

Perancangan keamanan data gambar 5 digunakan untuk mendekripsi *file cipher digital audio.* Perancangan

keamanan data *digital audio* tersebut menggunakan *AES-256 bit, CBC Mode*.

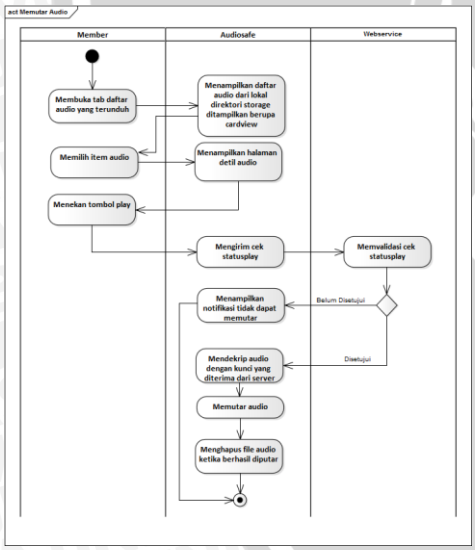
4.3 Perancangan Aktivitas Sistem

Pada Gambar 6, dijelaskan alur aktivitas saat user akan mengunggah audio. *User* melakukan input beberapa parameter dalam activity *uploadaudio*, selanjutnya *user* menekan tombol *upload* dan proses enkrispi dijalankan kemudian file dan data dikirimkan ke sever.



Gambar 6 Diagram aktivitas mengunggah audio

Pada Gambar 7, dijelaskan alur aktivitas saat *user* akan memutar audio. Setelah sebelumnya *user* melakukan activity melihat *detail downloaded audio* yang dimiliknya. *User* dapat memutar audio dengan menekan tombol *play*.



Gambar 7 Diagram aktivitas memutar audio

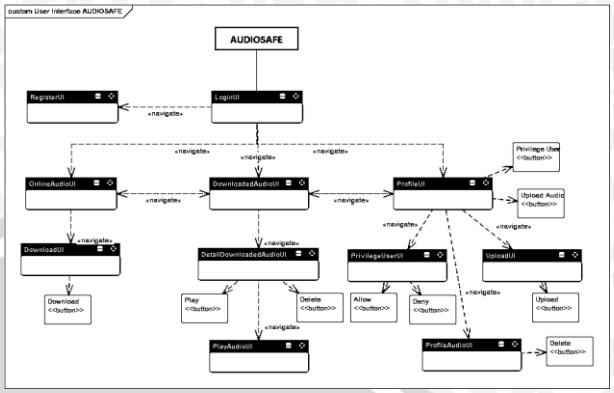
4.4 Perancangan *Database*

Berikut adalah tiga Tabel dalam basis data yang digunakan untuk memanajemen data dalam *webservice Audiosafe*:

- Tabel Members (*memberID, username, password, loginstatus)*
- Tabel ListAudio (*audioID, uploaderID, title, key, coverpath, audiopath*)
- Tabel Statusplay (*memberID, privilegememberID, status*)

4.5 Perancangan Antarmuka

Perancangan antarmuka membahas tentang perancangan antarmuka yang akan ditampilkan kepada pengguna yang bertujuan untuk menjelankan fungsi yang disediakan oleh sistem. Berikut adalah Gambar 8 *navigation map* keseluruhan dari *Audiosafe*.



Gambar 8 *Navigation map interface Audiosafe*

5. Implementasi

Dijelaskan bagaimana pengimplementasian perangkat lunak. Aplikasi diimplementasikan dengan bahasa pemrograman *Java* dan menggunakan *Android Developer Tool*

5.1 Implementasi Algoritma

Algoritma yang sudah dibentuk kemudian diimplementasikan dengan bahasa pemrograman java.

Berikut pada Kode Sumber 3 adalah algoritma bagaimana mengunggah *digital audio* dengan mengenkripsi *file* dengan *key* yang diinput oleh *user* kemudian dikrimkan kepada server.

Kode Sumber 3 Algoritma mengunggah audio

```
String responseString = null;
String audioname =
audionameText.getText().toString();
String key = keyText.getText().toString();
String fileimagePath =
tvimage.getText().toString();
String fileaudioPathtemp =
tvaudio.getText().toString();
String fileaudioPath =
"mnt/sdcard/Audiosafe/Audio/temp.aufe";

try :
aes = new AESSecHandler(key);

catch :
UnsupportedEncodingException e
GeneralSecurityException e

try :
Stopwatch timer = new Stopwarch();
timer= new Stopwatch();
timer.start();
aes.encrypt(fileaudioPathtemp, fileaudioPath);
timer.stop();

catch :
IOException e
GeneralSecurityException e

HttpClient httpclient = new
DefaultHttpClient();
HttpPost httppost = new
HttpPost(UPLOAD_AUDIO_URL);

try :
AndroidMultiPartEntity entity = new
AndroidMultiPartEntity(
    new
AndroidMultiPartEntity.ProgressListener() {
    public void transferred(long num) {
    publishProgress((int) ((num / (float)
totalSize) * 100));}});
```

```
File sourceaudioFile = new File(fileaudioPath);
File sourceimageFile = new File(fileimagePath);

entity.addPart("uploaderID", new
StringBody(memberID));
entity.addPart("audioname", new
StringBody(audioname));
entity.addPart("key", new StringBody(key));
entity.addPart("audio", new
FileBody(sourceaudioFile));
entity.addPart("image", new
FileBody(sourceimageFile));
totalSize = entity.getContentLength();
httppost.setEntity(entity);

HttpResponse response =
httpclient.execute(httppost);
HttpEntity r_entity = response.getEntity();

int statusCode =
response.getStatusLine().getStatusCode();
if (statusCode == 200) :
responseString =
EntityUtils.toString(r_entity);
responseprocess=1;

else :
responseString = "Error occurred! Http Status
Code: "+ statusCode;
responseprocess=0;

catch :
ClientProtocolException responseString =
e.toString();
IOException responseString = e.toString();

end
```

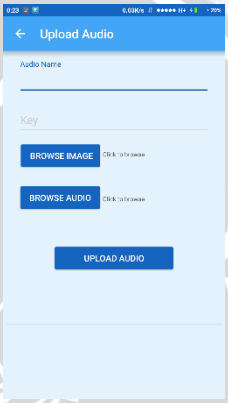
Berikut pada Kode Sumber 4 adalah algoritma bagaimana memutar *digital audio* dengan mendekripsi *file* yang sudah dienkripsi menggunakan *key* dari server yang diberikan persetujuan dari *user* pengunggah audio tersebut dan selanjutnya ditampilkan dalam *activity* untuk memutar audio.

Kode Sumber 4 Algoritma memutar audio

```
int response;
try:
List<NameValuePair> params1 = new
ArrayList<NameValuePair>();
params1.add(new BasicNameValuePair("audioname",
Filename));
params1.add(new BasicNameValuePair("memberID",
memberID));
JSONObject json =
jsonParser.makeHttpRequest(VIEW_DETAIL_STATUS_U
RL,"POST", params1);

if (json.length() > 0) :
JSONArray eventObject =
json.getJSONArray("list_event");
event = eventObject.getJSONObject(0);
status = event.getString("status");
if (Objects.equals(status, "allowed")):
response = 1;
String fileaudioPathtemp =
"mnt/sdcard/Audiosafe/Audio/"+Filename+".mp3";
try :
aes = new
AESSecHandler(event.getString("key"));

catch :
UnsupportedEncodingException e
GeneralSecurityException e

try :
timer= new Stopwatch();
timer.start();
aes.decrypt(Filepath, fileaudioPathtemp);
timer.stop();

catch :
IOException e
GeneralSecurityException e

pDialog.dismiss();

Intent intent = new
Intent(DetailDownloadedAudio.this,
PlayAudioExample.class);
i.putExtra("File", fileaudioPathtemp);
i.putExtra("Filename", Filename);
startActivity(intent);

else :
response = 0;

else :
"No data"

catch :
JSONException e

end
```

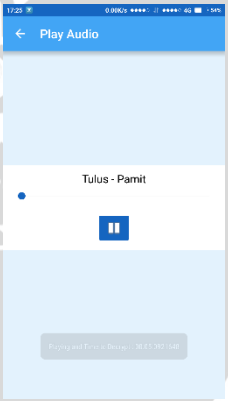
5.2 Implementasi Antarmuka

Gambar 9 menunjukan implementasi dari antarmuka mengunggah audio yang dirancang dalam pada tahap sebelumnya.



Gambar 9 Implementasi antarmuka memutar audio

Gambar 10 menunjukan implementasi dari antarmuka memutar audio yang dirancang dalam pada tahap sebelumnya.



Gambar 10 Implementasi antarmuka memutar audio

6. Pengujian dan Analisis

Bagian ini perangkat lunak dilakukan pengujian dari sisi teknis rekayasa perangkat lunak seperti pengujian *Whitebox* dan *Blackbox*, dan pengujian keamanan data kinerja kriptografi yang diukur dengan performansi kecepatan eksekusi kriptografi, keaslian data dan *security* data dan dianalisa hasil masing-masing pengujiannya.

6.1 Pengujian Unit

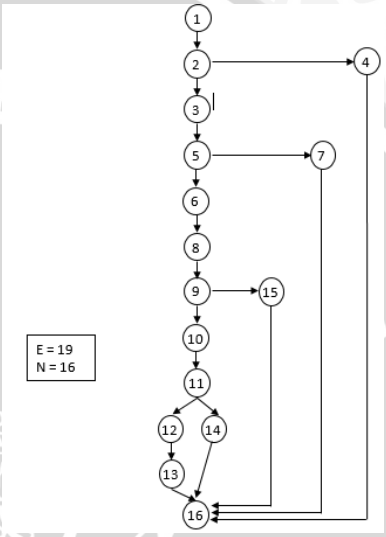
Pada pengujian unit untuk menguji aplikasi *Audiosafe* digunakan *Whitebox Testing* dilakukan dengan teknik *Basis*

Path Testing. Berikut adalah pseudocode algoritma mengunggah audio.

Kode Sumber 5 *Pseudocode* Algoritma mengunggah audio

```
1   String responseString = null;
2   String audioname =
3   audionameText.getText().toString();
4   String key =
5   keyText.getText().toString();
6   String fileimagePath =
7   tvimage.getText().toString();
8   String fileaudioPathtemp =
9   tvaudio.getText().toString();
10  String fileaudioPath =
11  "mnt/sdcard/Audiosafe/Audio/temp.au
12  fe";
13
14  try :
15  aes = new AESSecHandler(key);
16
17  catch :
18  UnsupportedEncodingException e
19  GeneralSecurityException e
20
21  try :
22  Stopwatch timer = new Stopwarch();
23  timer= new Stopwatch();
24  timer.start();
25  aes.encrypt(fileaudioPathtemp,
26  fileaudioPath);
27  timer.stop();
28
29  catch :
30  IOException e
31  GeneralSecurityException e
32
33  HttpClient httpclient = new
34  DefaultHttpClient();
35  HttpPost httppost = new
36  HttpPost(UPLOAD_AUDIO_URL);
37
38  try :
39  AndroidMultiPartEntity entity = new
40  AndroidMultiPartEntity(
41      new
42  AndroidMultiPartEntity.ProgressList
43  ener() {
44      public void transferred(long
45  num) {
46      publishProgress((int) ((num /
47  (float) totalSize) * 100));}});
48
49  File sourceaudioFile = new
50  File(fileaudioPath);
51  File sourceimageFile = new
52  File(fileimagePath);
53
54  entity.addPart("uploaderID", new
55  StringBody(memberID));
56  entity.addPart("audioname", new
57  StringBody(audioname));
58  entity.addPart("key", new
59  StringBody(key));
60  entity.addPart("audio", new
61  FileBody(sourceaudioFile));
62  entity.addPart("image", new
63  FileBody(sourceimageFile));
64  totalSize =
65  entity.getContentLength();
66  httppost.setEntity(entity);
67
68  HttpResponse response =
69  httpclient.execute(httppost);
70  HttpEntity r_entity =
71  response.getEntity();
72
73  int statusCode =
74  response.getStatusLine().getStatusC
75  ode();
76  if (statusCode == 200) :
77  responseString =
78  EntityUtils.toString(r_entity);
79  responseprocess=1;
80
81  else :
82  responseString = "Error occurred!
83  Http Status Code: "+ statusCode;
84  responseprocess=0;
85
86  catch :
87  ClientProtocolException
88  responseString = e.toString();
89  IOException responseString =
90  e.toString();
91
92  end
```

Algoritma Mengunggah Audio dimodelkan ke dalam *flowgraph* untuk menghitung kompleksitas siklomatis nantinya hasil dari perhitungan kompleksitas siklomatis akan menentukan banyaknya basis set / jalur independen pengujian dari algoritma Kode Sumber 5. Berikut Gambar 11 adalah konversi dari algoritma mengunggah audio menjadi *Flowgraph*:



Gambar 11 *Flowgraph* mengunggah audio

Melalui Flowgraph tersebut dapat dihitung kompleksitas siklmatis dengan menggunakan rumus:

1. V(G) = 5 regions
2. V(G) = 19E – 16N + 2 = 5
3. V(G) = 4P + 1 = 5

Berdasarkan dari nilai kompleksitas siklomatis yang telah didapatkan dari perhitungan maka ditentukan 4 buah basis set dari jalur independen, yaitu:

1. Jalur 1 = 1,2,4,16
2. Jalur 2 = 1,2,3,5,7,16
3. Jalur 3 = 1,2,3,5,6,8,9,15,16
4. Jalur 4 = 1,2,3,5,6,8,9,10,11,14,16
5. Jalur 5 = 1,2,3,5,6,8,9,10,11,12,13,16

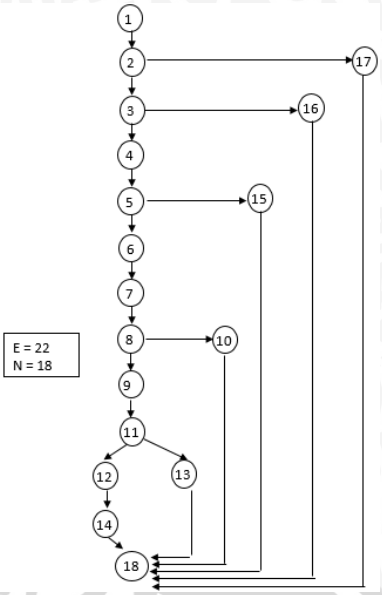
Berikut adalah pseudocode algoritma mengunggah audio.

Kode Sumber 6 *Pseudocode* Algoritma memutar audio

```
1   int response;
2   try:
3   List<NameValuePair> params1 = new
4   ArrayList<NameValuePair>();
5   params1.add(new
6   BasicNameValuePair("audioname",
7   Filename));
8   params1.add(new
9   BasicNameValuePair("memberID",
10  memberID));
11  JSONObject json =
12  jsonParser.makeHttpRequest(VIEW_DET
13  AIL_STATUS_URL,"POST", params1);
14
15  if (json.length() > 0) :
```

```
16  JSONArray eventObject =
17  json.getJSONArray("list_event");
18  event =
19  eventObject.getJSONObject(0);
20  status = event.getString("status");
21  if (Objects.equals(status,
22  "allowed")):
23  response = 1;
24  String fileaudioPathtemp =
25  "mnt/sdcard/Audiosafe/Audio/"+Filen
26  ame+".mp3";
27  try :
28  aes = new
29  AESSecHandler(event.getString("key"
30  ));
31
32  catch :
33  UnsupportedEncodingException e
34  GeneralSecurityException e
35
36  try :
37  timer= new Stopwatch();
38  timer.start();
39  aes.decrypt(Filepath,
40  fileaudioPathtemp);
41  timer.stop();
42
43  catch :
44  IOException e
45  GeneralSecurityException e
46
47  pDialog.dismiss();
48  Intent intent = new
49  Intent(DetailDownloadedAudio.this,
50  PlayAudioExample.class);
51  i.putExtra("File",
52  fileaudioPathtemp);
53  i.putExtra("Filename", Filename);
54  startActivity(intent);
55
56  else :
57  response = 0;
58
59  else :
60  "No data"
61
62  catch :
63  JSONException e
64
    end
```

Algoritma Memutar Audio dimodelkan ke dalam *flowgraph* untuk menghitung kompleksitas siklomatis nantinya hasil dari perhitungan kompleksitas siklomatis akan menentukan banyaknya basis set / jalur independen pengujian dari algoritma Kode Sumber 6. Berikut Gambar 12 adalah konversi dari algoritma memutar audio menjadi *Flowgraph*:



Gambar 12 *Flowgraph* memutar audio

Melalui *Flowgraph* tersebut dapat dihitung kompleksitas siklmatis dengan menggunakan rumus:

1. V(G) = 6 regions
2. V(G) = 22E – 18N + 2 = 6
3. V(G) = 5P + 1 = 6

Berdasarkan dari nilai kompleksitas siklomatis yang telah didapatkan dari perhitungan maka ditentukan 4 buah basis set dari jalur independen, yaitu:

1. Jalur 1 = 1,2,17,18
2. Jalur 2 = 1,2,3,16,18
3. Jalur 3 = 1,2,3,4,5,15,18
4. Jalur 4 = 1,2,3,4,5,6,7,8,10,18
5. Jalur 5 = 1,2,3,4,5,6,7,8,9,11,13,18
6. Jalur 6 = 1,2,3,4,5,6,7,8,9,11,12,14,18

6.2 Pengujian Validasi

Hasil uji validasi fitur perangkat lunak *Audiosafe* dijelaskan di jelas kan pada Tabel 1:

Tabel 1 Hasil pengujian validasi

No	Kasus Uji	Status.
1	*Digital audio* yang diunggah *member* dapat tersimpan didalam *server*.	*Valid*
2	*Member* dapat melihat daftar *digital audio* yang *member* miliki di *external storage device*.	*Valid*
3	Status dalam *privilege* yang *member* ubah sudah diupdate dalam server.	*Valid*

6.3 Pengujian Keamanan Data

Pengujian terkait keamanan data yang meliputi performansi waktu eksekusi proses enkrpsi dan dekripsi kriptografi, keaslian data dan *security* data.

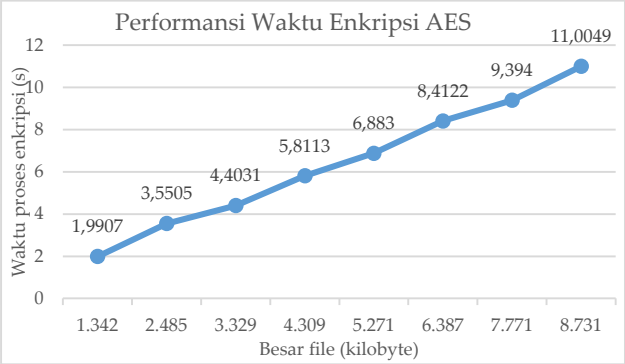
6.3.1 Pengujian Performasi Waktu Eksekusi Kriptografi

Pengujian performansi adalah pengujian yang dimaksudkan untuk menguji dan mengetahui tingkat performansi proses kriptografi dari sisi kecepatan proses enkripsi maupun dekripsi terhadap besarnya *file* yang terjadinya pada aktvitas kriptografi dalam aplikasi. Pengukuran kinerja waktu kriptografi juga didasarkan pada APDEX, yaitu index performa aplikasi yang

ditentukan dalam tiga ukuran, yaitu *satisfiying response time, tolerable response time* dan *frustated response time* [7].

Delapan sampel digunakan untuk mengukur performansi kriptografi *AES*. Berdasarkan masing-masing delapan sampel uji pengujian pada kriptografi *AES* di dapatkan hasil.

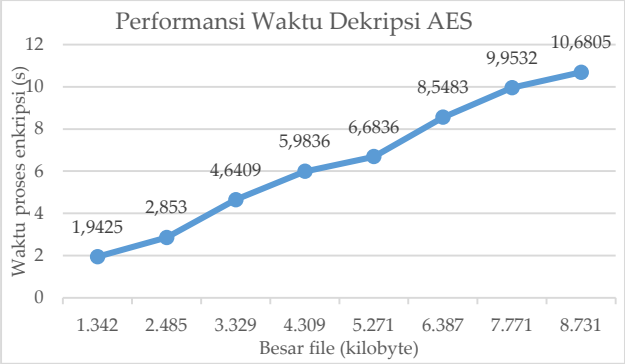
6.3.1.1 Waktu eksekusi enkripsi *AES* terhadap besar *file*



Gambar 13 Grafik waktu proses enkripsi *AES* terhadap besar *file*

Trendline dari pengaruh besar data terhadap waktu proses eksekusi enkripsi berbentuk pola linear yang terus naik. Dengan demikian dapat disimpulkan jika besar data sangat mempengaruhi performansi cepat atau lambatnya proses enkripsi. Jika waktu proses enkripsi yang didapatkan melalui sample tersebut terhadap expected value yaitu 4 detik, maka proses dekripsi data hingga 8,7 M*egaByte* adalah sesuatu yang tidak relevan dan performansinya sangatlah buruk jika mengacu pada index APDEX (Application Performance Index).

6.3.1.2 Waktu dekripsi *AES* terhadap besar *file*



Gambar 14 Grafik waktu proses deskripsi *AES* terhadap besar *file*

Trendline dari pengaruh besar *file* terhadap waktu proses dekripsi tampak membentuk garis linear yang terus naik. Dengan demikian dapat disimpulkan jika besar *file chiper* sangat mempengaruhi performansi dari proses dekripsi, semakin besar atau kecilnya ukuran *file chiper* akan mempengaruhi cepat/lambatnya proses dekripsi. Jika waktu proses dekripsi yang didapatkan melalui sample tersebut terhadap expected value yaitu 4 detik, maka proses dekripsi data hingga 8,7 M*egaByte* adalah sesuatu yang tidak relevan dan performansinya sangatlah buruk jika mengacu pada index APDEX (Application Performance Index).

6.3.2 Pengujian Keaslian Data

Tabel 2 Hasil uji *AES* keaslian data *file size*

No	Besarnya File (*byte*)	Besarnya File Pasca Enkripsi (*byte*)	Besar File Pasca Dekripsi (*byte*)	Status
1	1.341.941	1.342.084	1.341.941	*Valid*
2	2.484.942	2.485.076	2.484.942	*Valid*
3	3.328.626	3.328.772	3.328.626	*Valid*
4	4.309.423	4.309.556	4.309.423	*Valid*
5	5.270.666	5.270.804	5.270.666	*Valid*
6	6.386.812	6.386.948	6.386.812	*Valid*
7	7.771.590	7.771.732	7.771.590	*Valid*
8	8.731.432	8.731.572	8.731.432	*Valid*

Tabel 3 Hasil uji *AES* keaslian data *file content checksum*

No	*Checksum* (i=*initial*, e=*encrypted*, d=*decrypted*)	Status
1	ci = 2eb92c093393867f55b79a0904c75a81 ce= b7660766eab29906b871ac878d2d563f cd= 2eb92c093393867f55b79a0904c75a81	*Valid*
2	ci = 12fc84d3feb3b0907a323786b8c5ec98 ce= e56aeec97d55c555e347715874df1b49 cd= 12fc84d3feb3b0907a323786b8c5ec98	*Valid*
3	ci = 398a142737e3844d3c2216e20b6c44a1 ce= f68d073f29e8ebd55dac7cec2b9b860d cd= 398a142737e3844d3c2216e20b6c44a1	*Valid*
4	ci = 216973dc69cd89c57af034d350c5d466 ce= 290da673085464a520cd7f56e08920ac cd= 216973dc69cd89c57af034d350c5d466	*Valid*
5	ci = 4106b90ce50fba18ac508cffc031ec9b ce= 6283fe87e5f13e801102d01c54abb0e7 cd= 4106b90ce50fba18ac508cffc031ec9b	*Valid*
6	ci= 6b979721f1331a3b5f1306c45e68180b ce= f3eec2e869eba900b0c805d8f9287174 cd= 6b979721f1331a3b5f1306c45e68180b	*Valid*
7	ci= 7d74d98b3010a059110572e398eda217 ce= 3c3f7ad3706f078e87885f7380d8735a cd= 7d74d98b3010a059110572e398eda217	*Valid*
8	ci= 21cdfbe666486246ddd1edfd8e3cb144 ce= 2c67b0f3fef12c068e65ee041e108d8f cd= 21cdfbe666486246ddd1edfd8e3cb144	*Valid*

Hasil dari pengujian keaslian data didasarkan pada keaslian data besarnya *file* dan konten *file* dengan menggunakan MD5 *Checksum* setelah didekripsi kembali dengan menggunakan perangkat lunak sehingga mendapatkan data yang sama dengan data awal sebelum *file* mengalami proses enkripsi. Melalui Tabel 6.22 di dapatkan status data kasus uji bernilai seluruhnya *valid* yang berarti besar *file* akan kembali seperti semula setelah mengalami proses dekripsi dari perangkat lunak. Tingkat keaslian data besar *file* dari delapan kasus uji bernilai hingga 100% yang berarti besar file yang terkena imbas dari enkripsi *AES*-256 dapat dikembalikan keukuran semula.

Sementara dari pengujian keaslian data yang didasarkan pada *file content checksum* pada Tabel 6.23 di atas, *checksum file* delapan kasus uji setelah mengalami proses dekripsi bernilai sama persis dengan *checksum file* asli sebelum mengalami proses enkripsi. *Checksum* digunakan untuk melihat apakah terjadi perubahan data terhadap suatu *file*, secara tidak langsung nilai *checksum* yang sama antara hasil dekripsi dan *file* asli membuktikan jika tidak ada konten *file* yang berubah selama proses enkripsi. Tingkat keaslian data *file content checksum* dari delapan kasus uji bernilai hingga 100% yang berarti konten dari sampel *file* uji yang terkena imbas dari enkripsi *AES*-

256 dapat dikembalikan kebentuk semula tanpa mengubah isi dari konten setelah dilakukan dekripsi.

6.3.3 Pengujian *Security*

Tabel 4 Data uji pengujian security *AES*

Nomor Kombinatorial	*username* pemberi *privilege*	*username* penerima *privilege*	status *privilege*	nama audio
1	usertest1	usertest2	*allow*	test1
2	usertest1	usertest2	*deny*	test1
3	usertest2	usertest2	*allow*	test2
4	usertest2	usertest2	*deny*	test2

Tabel 5 Hasil pengujian *security*

No	*Expected Value*	*Result Value*	*Status Validasi*
1	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*valid*
2	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	*valid*
3	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*valid*
4	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	*valid*

Dari Tabel 5 pengujian *security* yang didasarkan pada *expected value* dan *result value*, status validitas pada empat kasus uji pada Tabel 4 setelah mengalami proses pemutaran *digital audio* menunjukan semua valid. Tingkat *security* dari empat kasus uji bernilai hingga 100% valid yang berarti keamanan dalam mengukur seberapa efektif teknik *privilege member dan* teknik kriptografi untuk menekan penyebaran distribusi *digital audio* pada *member* yang tidak berwenang atau tidak diberi ijin untuk pemutaran benar-benar aman dalam penggunaan perangkat lunak ini.

7. Penutup

Berisi penarikan kesimpulan dari penelitian yang dilakukan guna landasan pengetahuan penelitian selanjutnya dan saran yang bisa dijadikan penyempurnaan kegiatan penelitian selanjutnya. Berikut adalah kesimpulan dan saran bagi penelitian yang telah dilakukan oleh penulis.

7.1 Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian serta analisis yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut.

1. Perancangan perangkat lunak Audiosafe menggunakan tahap perancangan fungsional berdasarkan hasil spesifikasi kebutuhan yang menghasilkan 12 fitur dan perancangan non fungsional atau keamanan data dengan merancang kinerja dari sisi kriptografi yaitu enkripsi dan dekripsi pada perangkat lunak berdasarkan 3 parameter yaitu waktu performansi kriptografi, *recoverabilty* dan *security*.
2. Implementasi dari perangkat lunak dikembangkan berdasarakan perancangan padah tahap sebelumnya dan diimplementasikan dalam bahasa pemrograman Java dengan development tool *Android Developer Tool* yaitu *Android Studio* dan diimplementasikan dalam *device smartphone Android.* Dalam implementasi perangkat lunak menggunakan jenis pendekatan *Native Apps* karena proses dalam kriptografi membutuhkan resource yang besar, dan *native apps* dipilih karena dapat menggunakan semua *resource* dari sistem operasi tersebut.
3. Pengujian dilakukan dengan beberapa tahap yaitu, pengujian unit (*Whitebox*), pengujian validasi (*Blackbox*) dan pengujian pada keamanan data. Pengujian keamanan data menggunakan 3 parameter yaitu waktu performansi kriptografi, *recoverabilty* dan *security*.
4. Pengujian Unit (*Whitebox*) menghasilkan hasil jika fungsionalitas perangkat lunak disisi pengujian unit algoritma fungsi telah memenuhi hasil yang diharapkan. Pengujian Validasi (*Blackbox*) menghasilkan jika fungsionalitas perangkat lunak telah memenuhi kebutuhan yang diinginkan atau *valid* pada spesifikasi kebutuhan.
5. Pengujian waktu performansi kriptografi AES yang diimplementasikan dalam perangkat lunak menghasilkan waktu tolerable beradasarkan nilai APDEX dengan syarat kurang lebih *file* berukuran 3 MegaByte. Berdasarkan hasil pengujian untuk waktu enkripsi pada *file* yang berukuran kurang lebih 3 MegaByte menghasilkan waktu 4,4 detik untuk mendapatkan performa waktu eksekusi dalam rentang tolerable jika mengacu pada APDEX. Demikian juga pada proses dekripsi, untuk waktu yang dibutuhkan dalam *file* kurang lebih 3 MegaByte menghasilkan waktu 4,64 detik untuk mendapatkan performa waktu eksekusi dalam rentang tolerable jika mengacu pada APDEX.. Proses enkripsi dan dekripsi data lebih dari ukuran tersebut di dalam perangkat lunak mengakibatkan performa kecepatan waktu menjadi buruk atau *frustated* jika mengacu pada APDEX.
6. Pengujian keaslian data menunjukan hasil bahwa file yang mengalami proses enkripsi akan dapat kembali pada hasil yang sama saat mengalami proses dekripsi.

Berdasarkan hasil pengujian, tingkat keaslian data file dalam perangkat lunak bernilai 100%.

7. Pengujian *security* menunjukan hasil bahwa berdasarkan proses pada perangkat lunak, *file digital audio* tidak dapat diputar ketika belum diberikan *privilege* dari *user* pengunggah *digital audio*. Berdasarkan hasil pengujian, tingkat *security* file dalam perangkat lunak bernilai 100%.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan perangkat lunak ini bagi penelitian selanjutnya antara lain:

1. Implementasi perangkat lunak Audiosafe hanya dapat dijalankan pada versi SDK 21 keatas, diharapkan pada penelitian selanjutnya dapat diimplementasi juga pada versi sebelumnya.
2. Sistem operasi yang diimplementasikan apda perangkat lunak ini adalah Android, diharapkan implementasi dalam penelitian ini juga dilakukan pada sistem operasi lain seperti *IOS*, atau *Windows Phone* untuk mengukur performansi waktu eksekusi kriptografi, keaslian data dan *security* data pada *environment* berbeda.
3. Pada penelitian ini, *key* yang diberikan oleh server ke *device*, dikirimkan tampa melalui pengamanan data sehingga dapat menyebabkan *snipping* data pada pihak ketiga dan dapat melakukan dekripsi bebas pada *file*. Diharapkan pada penelitian selanjutnya dapat meneliti pengamanan data pada *key* dan data yang mengalami proses transaksi antara *server* dan *device*.

DAFTAR PUSTAKA

[1] A. B. Pramudyanto, "Media Baru dan Peluang Counter-Hegemony atas Dominasi Logika Industri Musik (Studi Kasus Perkmbangan Netlabel di Indonesia)," *Jurnal ILMU KOMUNIKASI*, pp. 63-82, 2013.

[2] X. Zhou and X. Tang, "Reasearch and Implementation of RSA Algorithm for Encryption and Decryption," *The 6th International Forum on Strategic Technology*, 2011.

[3] B. Gadanayak and C. Pradhan, "Encryption on MP3 Compression," *MES Journal of Technology and Management* , pp. 86-89, 2011.

[4] P. Mahajan and A. Sachdeva, "A Study of Encryption Algorithms AES, DES and RSA for," *Global Journal of Computer Science and Technology Network, Web & Security*, 2013.

[5] "Encyclopedia," 2016. [Online]. Available: http://www.pcmag.com/encyclopedia/term/46725/media-player.

[6] N. Endriani and E. Utami, "Implementasi Algoritma Enkripsi AES pada Aplikasi SMS (Short Message Service) Berbasis Android," 2014.

[7] I. Apdex Alliance, Application Performance Index - Apdex Technical Spesification, Apdex Aliance, Inc, 2007.