Studi dan Implementasi Cryptography Package pada Sistem Operasi Android

Danang Tri Massandy (13508051)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
danangstei08161@students.itb.ac.id

Pada saat ini, penggunaan smartphone sudah menjadi trend di masyarakat dunia. Device telepon genggam yang semula hanya berfungsi untuk menerima atau menelpon dan menerima/mengirim SMS kini bertambah fungsi sebagai mini komputer yang memiliki banyak fungsi, misalnya jelajah internet, gaming, pemutar musik/video, memotret gambar/merekam video, bahkan sampai internet banking. Smartphone saat ini banyak dijalankan dengan menggunakan sistem operasi Android. Android adalah sistem operasi buatan Google yang bersifat open source. Hal ini membuat banyak developer mengembangkan aplikasi pada Android. Smartphone yang dijalankan oleh Android dapat menyimpan banyak data-data personal yang hanya boleh diketahui oleh pemiliknya saja misalnya akun email, akun jejaring sosial, akun kartu kredit, akun internet banking. Akan tetapi, keamanan yang disediakan oleh Android sendiri dirasa kurang cukup aman untuk melindungi privasi-privasi pemiliknya. Oleh karena itu, data-data yang disimpan dalam android seharusnya dienkripsi dengan aplikasi yang mengimplentasikan algoritma kriptografi tertentu sehingga pemilik merasa aman untuk menyimpan datanya. Mengembangkan aplikasi android sebenarnya tidak begitu rumit. Android sendiri menyediakan banyak package yang memiliki fitur-fitur salah satunya adalah package untuk kriptografi. Aplikasi android dibuat dengan basis bahasa java dan salah satu IDE yang banyak digunakan adalah eclipse. Selain itu, diperlukan juga SDK Android. Pada makalah ini, penulis akan mengimplementasikan package cryptography yang disediakan oleh Java. Aplikasi yang dibuat adalah aplikasi yang dapat mengenkripsi dan mendekripsi baik pesan ataupun file yang diberi nama AndroCrypt. Algoritma untuk enkripsi dan dekripsi yang diimplementasikan adalah algoritma 3DES dan AES.

Kata kunci : Android, kriptografi, 3DES, AES, enkripsi, dekripsi, java, Eclipse, Rijndael, AndroCrypt

I. PENDAHULUAN

Sistem operasi Android memang masih dalam tahap perkembangan. Namun, sistem operasi ini memiliki potensi yang cerah dan tidak akan kalah saingan dengan sistem operasi untuk mobile lainnya. Sifatnya yang open source membuat banyak developer software yang mulai mengembangkan berbagai aplikasi untuk Android. Jumlah aplikasi Android yang ada di Android Market saat ini mencapai lebih dari 200.000 (Desember 2010) baik aplikasi, games, ataupun widgets dengan rata-rata total download lebih dari dua setengah juta kali. Pengguna Android mempunyai kemudahan untuk mendapatkan aplikasi-aplikasi, misalnya dengan cara mendownload langsung dari Android Market ataupun mendownload file installer *.apk* dari internet. Namun, diutamakan untuk mendownload dari Android Market karena aplikasi disana lebih terjamin. Aplikasi untuk android bentuknya bermacam-macam, ada yang benar-benar gratis, gratis tapi berbayar, maupun berbayar. Aplikasi yang gratis untuk penggunanya mendapatkan keuntungan dengan pemasangan iklan-iklan *google AdMobs* yaitu layanan iklan berbasiskan *mobile* yang dibuat oleh *google*.

Sistem operasi Android membutuhkan SD Cards untuk diformat dalam format VFAT (FAT32). Hal ini menyebabkan ketidakamanan data-data yang disimpan di dalamnya. Padahal, android dapat menyimpan banyak data-data pribadi penggunanya, dari akun email, jejaring sosial, kartu kredit, akun internet banking. VFAT adalah standar file sistem yang lama yang tidak mendukung kontrol akses dari Linux, sehingga semua data yang disimpan digunakan bersama oleh semua program pada device. Keamanan untuk Android tidak hanya pada media penyimpanannya saja, namun lalu lintas data juga seharusnya dipastikan aman agar data pribadi kita tidak disadap oleh pihak yang tidak berkepentingan. Misalnya, ketika kita mengirim pesan yang cukup rahasia kepada pengguna lain, maka kita memerlukan sebuah cara pengamanan pesan tersebut agar pesan tersebut tidak diketahui oleh orang lain.

Oleh karena itu, baik untuk pengamanan pada media eksternal penyimpanan data maupun lalu lintas data, diperlukan adanya sebuah proses enkripsi agar data pengguna tidak dapat dibaca oleh orang lain. Bahasa java sendiri telah menyediakan fitur kriptografi di dalam package/library 'javax.crypto'. Untuk mengamankan data di dalam device, cukup dibuat aplikasi sederhana yang dapat mengenkripsi dan mendekripsi data. Algoritma yang paling banyak digunakan adalah AES (*Advanced Encryption Standard*) yang ditemukan oleh Vincent Rijmen dan Joan Daemen. Algoritma lain yang digunakan dalam enkripsi data adalah Triple-DES (*Triple Data Encryption Standard*) yang merupakan pengembangan lebih lanjut dari algoritma DES.

Library 'javax.crypto' menyediakan fitur untuk mengimplementasikan enkripsi dan dekripsi data dengan kedua algoritma tersebut. Library ini juga menyediakan fitur untuk *key agreement*.

II. ALGORITMA TRIPLE-DES

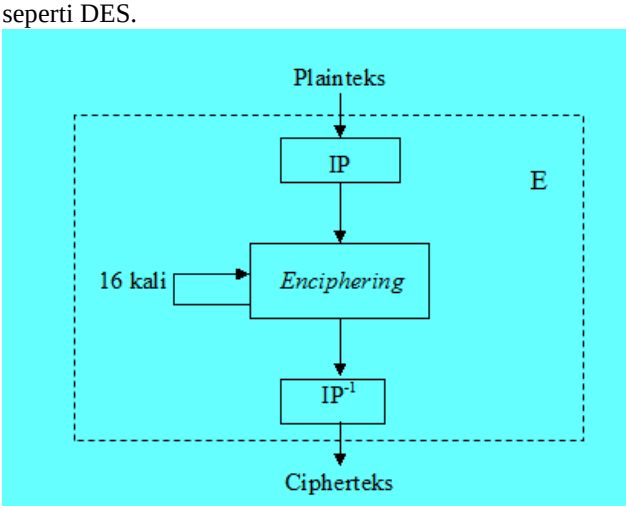
A. Deskripsi Umum Algoritma

Dalam kriptografi, 3DES sering kali disebut dengan Tripe Data Encryption Algorithm (TDEA atau Triple DEA) chiper blok, yang mengimplementasikan algoritma standar enkripsi data (DES) tiga kali untuk setiap blok data. Algoritma ini merupakan pengembangan dari algoritma DES karena algoritma DES yang asli mudah diserang melalui serangan *brute-force* ukuran panjang kuncinya sehingga diperlukan suatu metode untuk meningkatkan ukuran panjang kunci DES untuk memproteksi dari serangan-serangan tersebut tanpa harus medesain algoritma chiper blok yang baru.

Triple DES yang merupakan variasi dari standar DES tiga kali lebih lambat dari DES sendiri namun sangat lebih aman jika digunakan dengan benar. Triple DES banyak digunakan daripada DES karena saat ini DES dapat dipecahkan dengan waktu berbanding terbalik dengan kemajuan teknologi yang berkembang dengan cepat. Saat ini, dengan modal satu juta dolar memungkinkan untuk membuat sebuah perangkat keras yang dapat mencari semua kemungkinan kunci DES dalam 3,5 jam.

B. Cara Kerja Algoritma AES

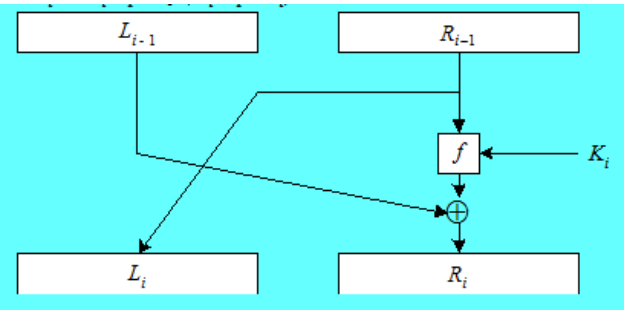
Triple DES menggunakan tiga kunci 64-bit, dengan total panjang key 192 bit. Dalam pengimplentasiannya, tiga kunci ini dimasukkan sebagai satu kunci saja. Kemudian algoritma ini memecah kunci tersebut menjadi tiga bagian dan menambahkan bit-bit jika panjangnya belum mencapai 64-bit. Prosedur untuk enkripsi sama seperti DES.



Gambar 1 Skema Umum DES

Setiap blok dienkripsi berulang 16 kali dan selalu menggunakan kunci internal yang berbeda. Kunci internal ini berukuran 56-bit dibangkitkan dari kunci eksternal. Setiap blok mengalami permutasi awal, 16 putaran enciphering, dan inverse permutasi awal (IP^{-1}). Permutasi awal digunakan untuk mengacak plainteks sehingga urutan bit-bit di dalamnya berubah. Untuk setiap enchipering, digunakan jaringan Feistel :

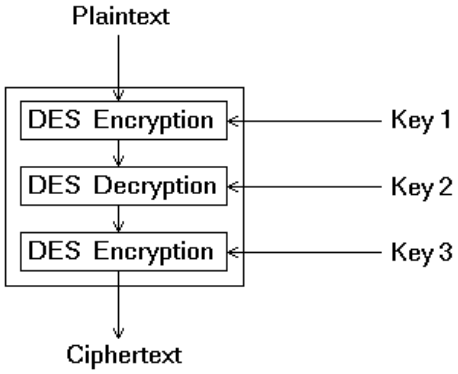
$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$



Gambar 2Jaringan Fiestel

Permutasi akhir dilakukan setelah 16 kali putaran terhadap gabungan blok kiri dan blok kanan. Permutasi akhir menggunakan matriks permutasi awal balikan (IP^{-1}). Untuk proses dekripsi terhadap chiperteks, digunakan kebalikan dari proses enkripsi.

Untuk Triple DES, prosedur untuk enkripsi maupun dekripsi diulang tiga kali. Urutan prosedur enkripsi adalah data dienkripsi dengan kunci pertama, data didekripsi dengan kunci kedua, kemudian data dienkripsi dengan kunci ketiga.



Gambar 3 Prosedure Enkripsi pada 3DES

Bentuk umum Triple DES :
Enkripsi: $C = E_{K3}(D_{K2}(E_{K1}(P)))$
Dekripsi: $P = D_{K1}(E_{K2}(D_{K3}(C)))$

3DES berjalan lebih lama daripada DES sebab 3DES merupakan tiga kali proses DES. Akan tetapi, 3DES ini lebih aman dibandingkan dengan DES jika penggunaan 3DES ini benar. 3DES mempunyai celah keamanan yaitu ketika kunci 1 sama dengan kunci 2 atau kunci 2 sama dengan kunci 3, maka prosedur enkripsi akan menjadi sama dengan DES aslinya. Kasus seperti ini menyebabkan 3DES lebih buruk daripada DES karena prosedur enkripsinya sama namun berjalan lebih lama daripada DES.

Mode operasi dari 3DES adalah Triple ECB (Electronic Code Book) dan Triple CBC (Chiper Block Chaining). Triple ECB adalah mode operasi yang paling banyak digunakan. Untuk Triple CBC, *initialization vector* menggunakan 64-bit kunci pertama.

III. ALGORITMA AES

A. Deskripsi Umum Algoritma

AES atau *Advanced Encryption Standard* merupakan standar enkripsi kunci simetri yang pada awalnya diterbitkan dengan algoritma Rijndael. Algoritma ini dikembangkan oleh dua kriptografer Belgia, Joan Daemen dan Vincent Rijmen. AES diumumkan oleh Institut Nasional Standar dan Teknologi (NIST) sebagai standar pemrosesan informasi federal (FIPS) pada tanggal 26 November 2001.

Spesifikasi dari algoritma ini sebagai berikut,

- Mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit.
- Panjang kunci dan ukuran blok dapat dipilih secara independen.
- Setiap blok dienkripsi dalam sejumlah putaran tertentu, sebagaimana halnya pada DES.
- Karena AES menetapkan panjang kunci adalah 128, 192, dan 256, maka dikenal AES-128, AES-192, dan AES-256.

Jumlah putaran pada proses enkripsi/dekripsi dapat berubah-ubah sesuai dengan jumlah kunci dan ukuran blok yang dipilih secara independen. Jenis-jenis AES ada tiga yaitu,

JENIS AES	Panjang Kunci	Ukuran Blok	Jumlah Putaran
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tabel 1 Jenis AES dan keterangan

Panjang kunci, ukuran blok, dan jumlah putaran dalam satuan blok. Satu blok sama dengan 32 bit.

Secara de-fakto, hanya ada dua varian *AES*, yaitu *AES*-128 dan *AES*-256, karena akan sangat jarang pengguna menggunakan kunci yang panjangnya 192 bit. Dengan panjang kunci 128-bit, maka terdapat sebanyak

$$2^{128} = 3{,}4 \times 10^{38}$$ kemungkinan kunci.

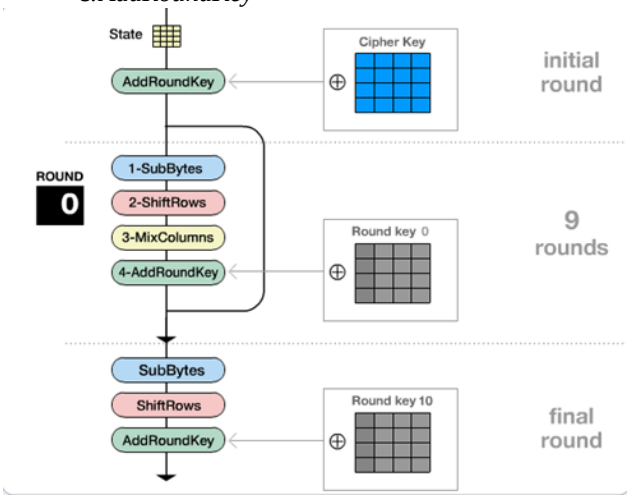
Jika komputer tercepat dapat mencoba 1 juta kunci setiap detik, maka akan dibutuhkan waktu $5{,}4 \times 10^{24}$ tahun untuk mencoba seluruh kunci. Jika tercepat yang dapat mencoba 1 juta kunci setiap milidetik, maka dibutuhkan waktu $5{,}4 \times 10^{18}$ tahun untuk mencoba seluruh kunci.

B. Cara Kerja Algoritma AES

Algoritma ini berbeda dengan DES yang berorientasi bit, Rijndael beroperasi dalam orientasi byte. Pada setiap putarannya digunakan kunci internal yang berbeda (disebut dengan *round key*). *Enchipering* melibatkan operasi subtitusi dan permutasi.

Garis besar algoritma Rijndael yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitan *round key*).

1. *AddRoundKey* : melakukan XOR antara state awal (plainteks) dengan *chiper key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah :
   a. *SubBytes* : subtitusi *byte* dengan menggunakan tabel subtitusi (*S-box*)
   b. *ShiftRows* : pergeseran baris-baris *array state* secara *wrapping*
   c. *MixColumns* : mengacak data di masing-masing kolom *array state*
   d. *AddRoundKey* : melakukan XOR antara state sekarang *round key*.
3. *Final round* : proses untuk putaran terakhir.
   a. *SubBytes*
   b. *ShiftRows*
   c. *AddRoundKey*



Gambar 4 Skema operasi algoritma Rijndael

Pada umumnya, Algoritma Rijndael mempunyai 3 parameter sebagai berikut:

1. plainteks : array yang berukuran 16 byte, yang berisi data masukan.
2. cipherteks : array yang berukuran 16 byte, yang berisi hasil enkripsi.
3. key : array yang berukuran 16 byte, yang berisi kunci ciphering (disebut juga cipher key).

Dengan 16 *byte*, maka blok data dan kunci yang berukuran 128-bit dapat disimpan di dalam *array* (128 = 16 × 8).

IV. PACKAGE KRIPTOGRAFI PADA JAVA

Java menyediakan library untuk kriptografi pada *javax.crypto*. Package ini tersedia baik untuk pengembangan aplikasi berbasis desktop ataupun android. Package ini menyediakan kelas-kelas dan interface untuk aplikasi kriptografi yang mengimplementasikan algoritma untuk enkripsi, dekripsi atau *key agreement*. Dalam stream chiper yang akan dipakai, disediakan baik asimetri, simetri ataupun chiper blok. Implementasi chiper dari provider yang berbeda dapat diintegrasikan dengan menggunakan SPI (Service Provider Interface) abstract

kelas. Kelas yang sering dipakai untuk enkripsi dan dekripsi adalah kelas Chiper dan SecretKeySpec.

A. SecretKey

SecretKey adalah kunci simetri dalam kriptografi yang rahasia. *SecretKeySpec* adalah kunci yang lebih spesifik dan sering digunakan untuk menampung kunci saat enkripsi/dekripsi. Untuk membuat instance dari kelas ini digunakan konstruktor sebagai berikut,

```
public SecretKeySpec (byte[] key, String algorithm)
```

Key adalah data kunci dalam array byte, algorithm adalah jenis algoritma yang digunakan. Contoh nilai dari parameter algorithm adalah "DESede" atau "AES".

B. Chiper

Kelas ini menyediakan akses ke implementasi dari kriptografi chiper untuk enkripsi maupun dekripsi. Kelas Chiper ini tidak bisa diinstansiasi secara langsung, melainkan dengan cara memanggil method getInstance dengan parameter nama transformasi yang diinginkan, opsional dengan provider. Transformasi menentukan operasi sebagai bentuk string :

"algorithm/mode/padding"

Algoritma adalah nama sebuah algoritma kriptografi, mode adalah nama mode umpan balik dan padding adalah nama sebuah skema padding. Jika mode dan/atau nilai padding dihilangkan, penyedia nilai default tertentu akan digunakan.

Setelah diinstansiasi, chiper harus memanggil method init.

```
public final void init (int opmode, Key key, AlgorithmParameterSpec params)
```

Opmode diisi dengan Chiper.ENCRYPT_MODE atau Chiper.DECRYPT_MODE, key adalah kunci yang akan digunakan (bisa berupa `SecretKeySpec`), params dapat berupa ivParamaterSpec yang isinya Initialization Vector.

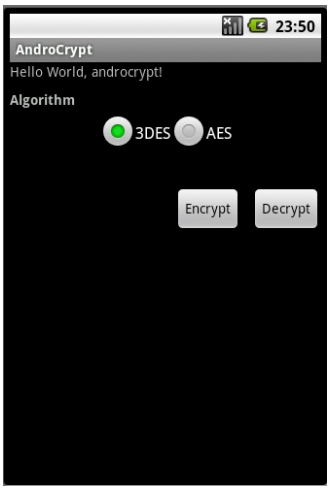
Langkah terakhir, chiper memanggil method doFinal() untuk menjalankan enkripsi/dekripsi dengan parameter byte plain/chiperteks.

V. IMPLEMENTASI PADA APLIKASI ANDROCRYPT

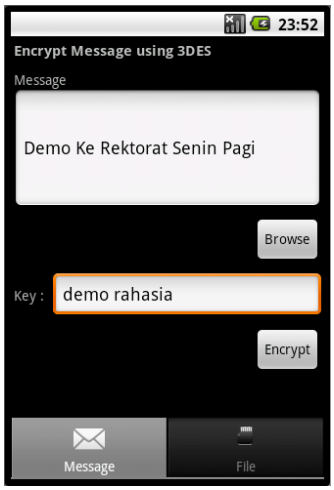
Aplikasi Android yang dibuat diberi nama AndroCrypt. Androcrypt adalah aplikasi kriptografi yang menggunakan pilihan algoritma 3DES dan AES. Androcrypt yang dibuat memiliki spesifikasi dan fitur-fitur sebagai berikut :

1. Dijalankan di lingkungan sistem operasi Android 2.1 Éclair ataupun versi di atasnya.
2. Memiliki fitur enkripsi dan dekripsi dari pesan atau file yang ada di memori eksternal.
3. Fitur enkripsi dan dekripsi menggunakan pilihan algoritma 3DES atau AES.
4. Pesan yang dienkripsi ditampilkan dalam bentuk string hexadesimal.
5. Pesan yang telah dienkripsi atau didekripsi dapat disimpan ke dalam memori eksternal ataupun dapat dikirim ke nomor-nomor ponsel tertentu sebagai Sort Message Service (SMS).
6. Untuk fitur mengirim pesan, disediakan menu untuk memilih beberapa kontak yang dibaca dari list kontak dari *device*. Biaya tetap dikenakan sesuai operator yang digunakan.
7. Dapat mengenkripsi dan mendekripsi file yang ada pada memori eksternal serta menyimpan hasilnya ke dalam memori eksternal.

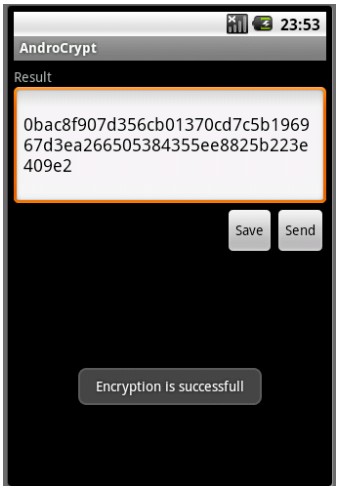
Berikut adalah tampilan dari androcrypt beserta fitur-fiturnya.



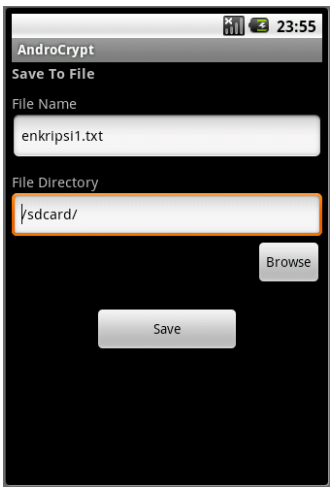
Gambar 5Tampilan Menu Awal



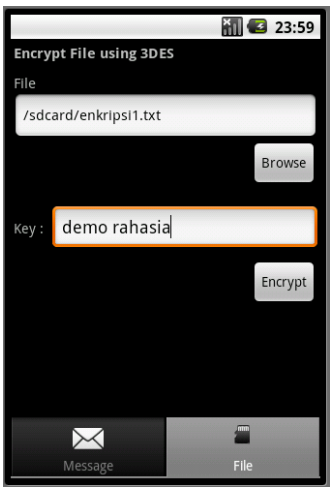
Gambar 6 Tampilan Enkripsi Pesan



Gambar 7 Tampilan Hasil Enkripsi Pesan



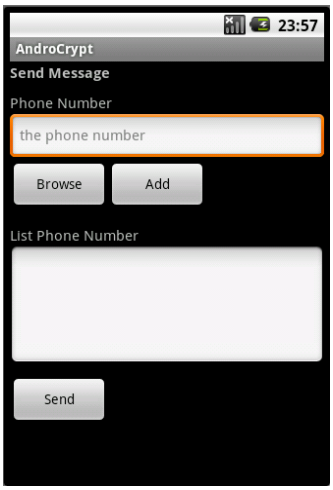
Gambar 8 Tampilan Simpan Hasil Ke File



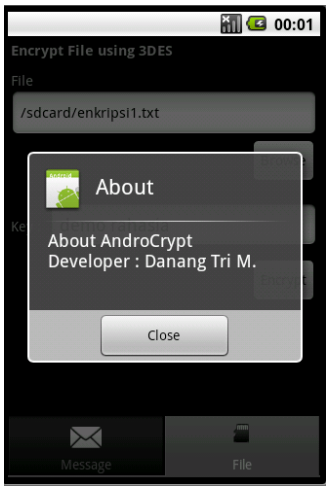
Gambar 11 Tampilan Enkripsi File

Gambar 9 Tampilan Membuka Folder/Memilih File

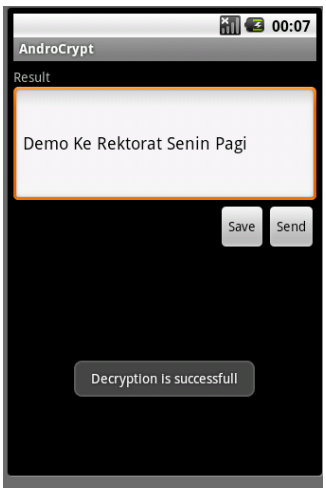
Gambar 12 Tampilan Menu Mengganti Pilihan Algoritma



Gambar 10 Tampilan Mengirim Pesan Ke Kontak Telefon



Gambar 13 Tampilan About



Gambar 14 Tampilan Hasil Dekripsi

Pada pembuatan program AndroCrypt, digunakan IDE Eclypse dengan bahasa Java. Berikut adalah pembagian modul-modul program beserta keterangannya :

Nama Modul	Keterangan
AES.java	Fungsi enkripsi dan dekripsi berdasarkan algoritma AES
androcrypt.java	Activity yang menampilkan menu di halaman utama/awal
Contact.java	Struktur data dari kontak telefon
ContactArrayAdapter.java	Kelas untuk menggenerate list kontak telefon
ContactListDemo.java	Menampilkan list kontak telefon untuk dipilih
DESede.java	Fungsi enkripsi dan dekripsi berdasarkan algoritma 3DES/DESede
FileActivity.java	Activity yang menampilkan menu untuk enkripsi/dekripsi file
FileArrayAdapter.java, FileChooser.java	Mengatur menu browser file/folder pada memori eksternal
FileManager.java	Mengatur baca dan tulis file
globals.java	Menyimpan variabel dan fungsi-fungsi global
MessageActivity.java	Activity yang menampilkan menu untuk enkripsi/dekripsi pesan
MessageResult.java	Activity yang menampilkan hasil pesan terenkripsi/terdekripsi
Option.java	Mengatur tampilan menu option
saveActivity.java	Activity yang menampilkan pilihan untuk menyimpan pesan/file ke memori eksternal
sendActivity.java	Activity yang menampilkan menu untuk mengirim pesan ke kontak
smsHandler.java	Kelas untuk handler mengirim pesan
TabActivity	Menampilkan pilihan menu tab message/file

Tabel 2 Modularisasi AndroCrypt

Struktur data yang dibuat disimpan di dalam file globals.java yang dapat digunakan di modul-modul lainnya.

```
public static int mode = 0;
// 0 encrypt, 1 decrypt

public static int algorithm = 0;
// 0 3DES, 1 AES

public static TextView LabelJudul;
public static TextView LabelJudul2;

// path chooser for browse file
public static String pathChooser;
// path chooser for browse folder
public static String pathfolderChooser;

// boolean to choose folder instead file
public static Boolean choose_folder=false;

// string phone number
public static String phonenumber;

// string input
public static String inputMessage;

// string result
public static String resultMessage;

// array of byte resultFile
public static byte[] resultFile;

// string key
public static String keyPhrase;

// boolean save message result
public static Boolean saveMessageResult=false;

// save the phone contact list
public static List<Contact> contacts;
```

Variabel resultMessage untuk hasil enkripsi nilainya dalam hexadesimal. resultMessage tidak ditampilkan dalam bentuk string byte nya karena ada beberapa karakter ASCII yang tidak terlihat. Untuk resultFile digunakan array of byte.

Pada aplikasi android tampilan diatur oleh file xml yang ada dalam folder res/layout. Daftar tampilan layout yang dibuat adalah sebagai berikut,

Nama File	Fungsi
file_view.xml	Mengatur layout file browser
filelayout.xml	Mengatur tampilan dari Activity FileActivity.java
listitemlayout.xml	Mengatur layout kontak browser
main_tab.xml	Mengaur layout tab file/message
main.xml	Mengatur tampilan pada

	halaman awal
messagelayout.xml	Mengatur tampilan pada MessageActivity.java
resultmessage.xml	Mengatur tampilan pada MessageResult.java
savelayout.xml	Mengatur tampilan pada SaveActivity.java
sendlayout.xml	Mengatur tampilan pada SendActivity.java

Tabel 3 Layout Manager

Pengaturan Activity yang berjalan dan permission yang digunakan diletakkan pada file AndroidManifest.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=
"http://schemas.android.com/apk/res/android"
      package="com.kripto.androcrypt"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name=
"android.permission.READ_CONTACTS">
    </uses-permission>
    <uses-permission android:name=
"android.permission.WRITE_EXTERNAL_STORAGE">
    </uses-permission>
    <uses-permission android:name=
"android.permission.SEND_SMS">
    </uses-permission>

<application android:icon="@drawable/icon"
android:label="@string/app_name">
<activity android:name=".androcrypt"
android:label="@string/app_name">
   <intent-filter>
        <action
android:name="android.intent.action.MAIN" />
        <category
android:name="android.intent.category.LAUNCHER"
/>
    </intent-filter>
</activity>
<activity android:name=".TabActivity"
android:theme="@android:style/Theme.NoTitleBar"
/>
<activity android:name=".MessageActivity" />
<activity android:name=".MessageResult" />
<activity android:name=".FileActivity" />
<activity android:name=".FileChooser" />
<activity android:name=".saveActivity" />
<activity android:name=".sendActivity" />
<activity android:name=".ContactListDemo" />
</application>
</manifest>
```

Implementasi algoritma 3DES ada pada modul file DESede.java

```
  package com.kripto.androcrypt;
  import java.io.UnsupportedEncodingException;
  import java.security.GeneralSecurityException;
  import javax.crypto.Cipher;
  import javax.crypto.spec.DESedeKeySpec;
  import javax.crypto.spec.IvParameterSpec;
  import javax.crypto.spec.SecretKeySpec;

  /**
   * @author
   *
   */
  public class DESede {
     public static int MAX_KEY_LENGTH =
DESedeKeySpec.DES_EDE_KEY_LEN;
     private static String ENCRYPTION_KEY_TYPE =
"DESede";
  private static String ENCRYPTION_ALGORITHM =
"DESede/CBC/PKCS5Padding";
  private final SecretKeySpec keySpec;

  public DESede(String passphrase) {
    byte[] key;
    try {
      key = passphrase.getBytes("UTF8");
    } catch (UnsupportedEncodingException e) {
      throw new IllegalArgumentException(e);
    }
    key = padKeyToLength(key, MAX_KEY_LENGTH);
    keySpec = new SecretKeySpec(key,
ENCRYPTION_KEY_TYPE);
  }

  public byte[] getKey() {
    return keySpec.getEncoded();
  }

  private byte[] padKeyToLength(byte[] key, int
len) {
    byte[] newKey = new byte[len];
    System.arraycopy(key, 0, newKey, 0,
Math.min(key.length, len));
    return newKey;
  }

  public byte[] encrypt(byte[] unencrypted)
throws GeneralSecurityException {
    return doCipher(unencrypted,
Cipher.ENCRYPT_MODE);
    }

  public byte[] decrypt(byte[] encrypted) throws
GeneralSecurityException {
    return doCipher(encrypted,
Cipher.DECRYPT_MODE);
    }

  private byte[] doCipher(byte[] original, int
mode)
  throws GeneralSecurityException {
    Cipher cipher =
Cipher.getInstance(ENCRYPTION_ALGORITHM);
    IvParameterSpec iv = new IvParameterSpec(new
       byte[] { 0, 0, 0, 0, 0, 0, 0, 0 });
       cipher.init(mode, keySpec, iv);
       return cipher.doFinal(original);
  }
}
```

Atribut MAX_KEY_LENGTH adalah panjang kunci maksimal dari algoritma DES yang sudah didefinisikan pada kelas DESedeKeySpec. Panjang maksimal kunci ini akan digunakan untuk *padding*. ENCRYPTION_KEY_TYPE menentukan jenis algoritma enkripsi/dekripsi yaitu "DESede". Dalam java, Triple DES memang dikenal dengan "DESede". ENCRYPTION_ALGORITHM menentukan Chiper yang akan digunakan yaitu Chiper dengan parameter "DESede/CBC/PKCS5Padding". DESede adalah algoritma yang dipilih. CBC (Chiper Block Chaining) adalah mode enkripsi/dekripsi yang akan digunakan. PKCS5Padding adalah jenis *padding* yang digunakan untuk kunci yang dimasukkan. PKCS5 dikembangkan untuk chiper blok dengan ukuran blok 8 byte. Kelas DESede memiliki konstruktor berparameter string passphrase yaitu kunci yang dimasukkan oleh user. Langkah-langkah dalam enkripsi adalah sebagai berikut :

1. Membentuk kunci bertipe SecretKeySpec. Konstruktor dari tipe ini adalah array byte kunci dan tipe enkripsi kunci("DESede"). Array byte kunci sebelumnya dipadding terlebih dahulu dengan menggunakan fungsi padKeyToLength yang akan menambahkan byte 0 sampai maksimum panjang kunci yaitu 24 byte.
2. Enkripsi dengan memanggil fungsi encrypt yang kemudian memanggil fungsi doChiper dengan mode Chiper.ENCRYPT_MODE.
3. doChiper akan membuat instance dari kelas javax.crypto.Chiper dengan parameter ENCRYPTION_ALGORITHM.
4. Instance chiper tersebut memanggil method init yang berparameter mode enkripsi/dekripsi, keySpec, dan Initialization vector. IV disini adalah array 8 byte dengan nilai byte 0.
5. Langkah akhir adalah chiper memanggil method doFinal() yang berparameter array byte plainteks/chiperteks.
6. Untuk dekripsi mode yang digunakan adalah Chiper.DECRYPT_MODE.

Implementasi algoritma AES ada pada modul file AES.java.

```
  package com.kripto.androcrypt;

  import java.io.InputStream;
  import java.io.OutputStream;
  import java.io.UnsupportedEncodingException;
  import java.security.GeneralSecurityException;
  import java.security.MessageDigest;
  import java.security.NoSuchAlgorithmException;

  import javax.crypto.Cipher;
  import javax.crypto.CipherInputStream;
  import javax.crypto.CipherOutputStream;
  import javax.crypto.spec.IvParameterSpec;
  import javax.crypto.spec.SecretKeySpec;

  public class AES {
    private       static       final       String
ENCRYPTION_KEY_TYPE = "AES";
    private       static       final       String
ENCRYPTION_ALGORITHM = "AES/CBC/PKCS5Padding";
    private final SecretKeySpec keySpec;
    private final byte[] iv;

    /**
     * Converts a user-entered pass phrase into
a hashed binary value which is
     * used as the encryption key.
     * @param hashAlgorithm
     */
     public static byte[] passPhraseToKey(String
passphrase, String hashAlgorithm, int numHashes)
{
     if (numHashes < 1) {
       throw new IllegalArgumentException("Need
a positive hash count");
     }

     byte[] passPhraseBytes;
     try {
       passPhraseBytes                          =
passphrase.getBytes("UTF8");
     } catch (UnsupportedEncodingException e) {
       throw new IllegalArgumentException(e);
     }
  // Hash it multiple times
  byte[] keyBytes = passPhraseBytes;
  for (int i = 0; i < numHashes; i++) {
    keyBytes         =         doDigest(keyBytes,
hashAlgorithm);
  }

    return keyBytes;
  }

  public AES(String keyPhrase) {
  // Use an MD5 to generate an arbitrary
initialization vector
    byte[]               keyBytes               =
passPhraseToKey(keyPhrase,"MD5",7);
    iv = doDigest(keyBytes, "MD5");
    keySpec    =    new    SecretKeySpec(keyBytes,
ENCRYPTION_KEY_TYPE);
    }

   public  byte[]  encrypt(byte[]  unencrypted)
throws GeneralSecurityException {
    return                  doCipher(unencrypted,
Cipher.ENCRYPT_MODE);
   }

   public   byte[]   decrypt(byte[]   encrypted)
throws GeneralSecurityException {
     return                   doCipher(encrypted,
Cipher.DECRYPT_MODE);
   }

   private byte[] doCipher(byte[] original, int
mode) throws GeneralSecurityException {
     Cipher cipher = createCipher(mode);
     return cipher.doFinal(original);
   }

   private  static  byte[]  doDigest(byte[]  data,
String algorithm) {
     try {
        MessageDigest            md            =
MessageDigest.getInstance(algorithm);
        md.update(data);
        return md.digest();
      } catch (NoSuchAlgorithmException e) {
        throw new IllegalArgumentException(e);
      }
   }

   private Cipher createCipher(int mode) throws
GeneralSecurityException {
      Cipher               cipher               =
Cipher.getInstance(ENCRYPTION_ALGORITHM);
      cipher.init(mode,       keySpec,       new
IvParameterSpec(iv));
      return cipher;
   }
}
```

ENCRYPTION_KEY_TYPE menentukan jenis algoritma enkripsi/dekripsi yaitu "AES". ENCRYPTION_ALGORITHM menentukan Chiper yang akan digunakan yaitu Chiper dengan parameter "AES/CBC/PKCS5Padding". AES adalah algoritma yang dipilih. CBC (Chiper Block Chaining) adalah mode enkripsi/dekripsi yang akan digunakan. PKCS5Padding adalah jenis *padding* yang digunakan untuk kunci yang dimasukkan. PKCS5 dikembangkan untuk chiper blok dengan ukuran blok 8 byte. Kelas AES memiliki konstruktor berparameter string passphrase yaitu kunci yang dimasukkan oleh user. Langkah-langkah enkripsi/dekripsi menggunakan AES adalah sebagai

berikut,

1. Membentuk array bytes dari hasil fungsi hash input kunci dengan algoritma hash "MD5". Kunci di-hash sebanyak 7 kali. Kunci yang terbentuk menjadi 16 byte atau 128-bit.
2. Membentuk initialization vector dengan memanggil fungsi hash dodiggest berparamater algoritma "MD5" dan array bytes kunci yang telah dihash beberapa kali.
3. Membentuk kunci bertipe SecretKeySpec. Konstruktor dari tipe ini adalah array byte kunci yang telah dihash beberapa kali dan tipe enkripsi kunci("AES").
4. Enkripsi/dekripsi dengan memanggil fungsi encrypt/decrypt yang kemudian memanggil fungsi doChiper dengan mode Chiper.ENCRYPT_MODE/Chiper.DECRYPT_MODE.
5. doChiper akan membuat instance dari kelas javax.crypto.Chiper dengan parameter ENCRYPTION_ALGORITHM. Proses membuat instance dengan memanggil fungsi createChiper.
6. Method createChiper sendiri memanggil Chiper.getInstance sesuai algoritma yang dipilih dan menginisialisasi chiper dengan mode, keySpec, dan initialization vector.
7. Setelah itu, instance chiper melakukan enkripsi/dekripsi dengan memanggil method doFinal yang berparameter array byte plain/chiper teks.

VI. PENGUJIAN DAN ANALISIS

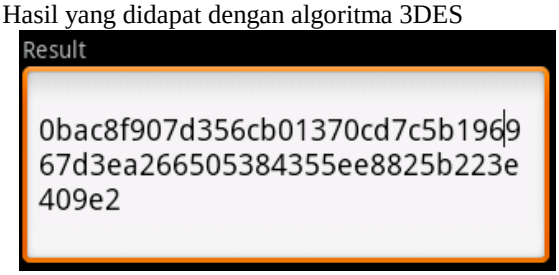
Pengujian pengenkripsian pesan yang berbunyi

Demo Ke Rektorat Senin Pagi

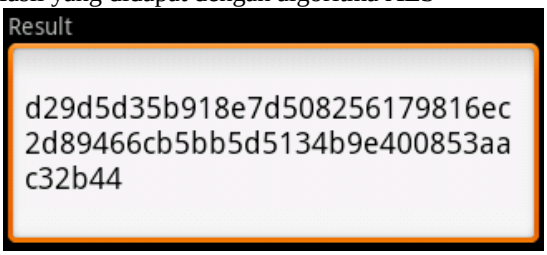
Kunci yang dimasukkan adalah

demo rahasia

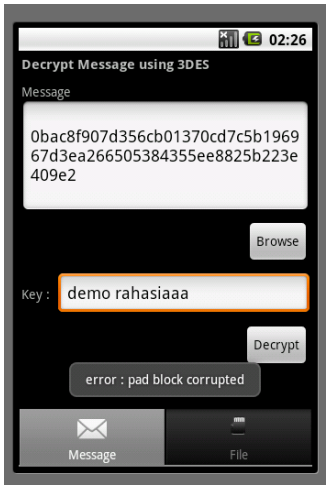
Hasil yang didapat dengan algoritma 3DES



Hasil yang didapat dengan algoritma AES



Prose pendekripsian dengan kunci yang sama menghasilkan plainteks yang tepat. Sedangkan jika kunci yang dimasukkan salah akan memunculkan pesan seperti pada gambar,



Gambar 15Proses Dekripsi dengan Kunci yang Salah

Exception : pad block corrupted berarti saat mekanisme *padding* ada kesalahan pada data input yang tidak sesuai dengan yang diinginkan/sesuai.

Pengujian dilanjutkan dengan memodifikasi input teks yang terenkripsi dengan menambahkan beberapa byte semu . Ketika didekripsi dengan kunci yang benar muncul *exception* "last block incomplete in decryption". Hal ini menunjukkan proses dekripsi gagal karena input teks yang tidak sesuai dengan kunci.

Untuk proses enkripsi dan dekripsi file telah diuji dengan menggunakan beberapa ekstensi file, misalnya, *.doc, .jpg, .mp3*. Pengujian dilakukan langsung pada device Sony Ericsson Xperia X8 dengan versi android Éclair. Lama waktu enkripsi dan dekripsi berbanding lurus dengan ukuran file. Semakin besar file tersebut maka semakin lama waktu yang dibutuhkan.

Algoritma 3DES yang diimplementasikan menggunakan cara *padding* kunci yang kurang efektif. Fungsi *padKeytoLength*() mengisi 24 byte array dengan byte 0 setelah array byte input kunci yang dimasukkan jika panjang kunci kurang dari 24 byte.

Misalkan kunci "demo rahasia" memiliki nilai hex,

64 65 6d 6f 20 72 61 68 61 73 69 61

Setelah di-*padding* menjadi 24 byte dan dipecah menjadi 3 bagian,

Kunci 1 64656d6f20726168
Kunci 2 6173696100000000
Kunci 3 0000000000000000

Sisa byte kunci diisi dengan byte 0. Dari contoh di atas belum tampak ada celah. Namun, jika kunci yang dimasukkan mempunyai panjang 8 byte, misalnya "demodemo", maka setelah di-*padding* dan dipecah,

Kunci 1 64656d6f64656d6f
Kunci 2 0000000000000000
Kunci 3 0000000000000000

Dalam contoh ini, kunci 2 = kunci 3. Bentuk umum dari 3DES adalah E = $E_{K3}(D_{K2}(E_{K1}(P)))$. Karena Kunci 2 = Kunci 3 dan 3DES adalah algoritma simetri maka enkripsi menggunakan kunci 3 dari hasil dekripsi menggunakan kunci 2, hasilnya adalah $E_{K1}(P)$ yang merupakan hasil yang sama jika dienkripsi dengan DES.

Hal ini membuat 3DES dapat dipecahkan dengan cara yang sama dengan memecahkan DES jika kunci yang digunakan hanya 8 byte.

Algoritma AES yang diimplementasikan menggunakan fungsi hash untuk *padding*. *Padding* yang dilakukan membuat kunci menjadi array byte dengan panjang 16 byte atau 128-bit. Sehingga, celah keamanan yang ada pada implementasi 3DES diatas telah ditanganani pada implementasi AES ini.

VII. KESIMPULAN

Aplikasi yang dibuat berjalan lancar pada lingkungan Android Éclair (2.1). Aplikasi telah diujicobakan pada simulator Android dan device Sony Ericsson Xperia X8. Fitur tambahan seperti mengirim pesan ke kontak telefon berfungsi dengan baik.

Implementasi algoritma 3DES memiliki celah keamanan jika kunci yang dimasukkan kurang dari 8byte karena fungsi dari 3DES akan menjadi sama seperti algoritma DES biasa. Untuk mengatasinya, *padding* kunci seharusnya dilakukan dengan menggunakan fungsi hash seperti pada implementasi algoritma AES.

Algoritma AES merupakan salah satu algoritma yang sangat kuat tingkat keamanannya. Algoritma yang diimplementasikan adalah AES jenis 128 bit. Pada JDK versi saat ini, terdapat batasan dalam pemilihan ukuran kunci 128 saja.

DAFTAR PUSTAKA

[1] http://www.informatika.org/~rinaldi/Kriptografi/kriptografi.htm
Tanggal akses : 2 Maret 2011

[2] http://developer.android.com/
Tanggal akses : 2 Maret 2011

[3] http://www.android-questions.com
Tanggal akses : 22 Maret 2011

[4] http://en.wikipedia.org/wiki/Android_(operating_system)
Tanggal akses : 22 Maret 2011

[5] http://securitymusings.com/article/2039/encrypt-stored-data-in-android
Tanggal akses : 22 Maret 2011

[6] http://www.tropsoft.com/strongenc/des3.html
Tanggal akses : 22 Maret 2011

[7] http://securitymusings.com/article/2039/encrypt-stored-data-in-android
Tanggal akses : 22 Maret 2011

[8] http://download.oracle.com/javase/1.4.2/docs/guide/security/CryptoSpec.html
Tanggal akses : 22 Maret 2011

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Maret 2011

Danang Tri Massandy
13508051