Studi Implementasi Algoritma *Block Cipher* pada Platform *Android*

Hafid Inggiantowi - 13507094
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if17094@students.if.itb.ac.id

Abstrak— Sistem operasi untuk perangkat *mobile* semakin berkembang. Android merupakan salah satu sistem operasi *mobile* yang kian kini sangat populer dan banyak digunakan orang-orang. Android juga merupakan sistem operasi yang berbasis perangkat lunak yang dapat dikembangkan secara terbuka (*open source*) sehingga banyak pengembang yang kini turut serta ikut mengembangkan aplikasi untuk Android. Untuk keperluan aspek keamanan, Android juga telah menyediakan API khusus untuk fungsi-fungsi kriptografi, seperti enkripsi, dekripsi, *key agreement*, dan sebagainya. Pada makalah ini, penulis akan melakukan analisis dan studi dalam mengimplementasikan algoritma *block cipher* pada pengembangan aplikasi di Android, dalam hal ini salah satunya yaitu algoritma DES.

Kata Kunci—Android, API, kriptografi, *block cipher*, DES.

I. Latar Belakang

Sistem operasi untuk *smartphone* kini semakin beragam. Setelah hadirnya *smartphone* berbasis sistem operasi BlackBerry dan Apple, kini juga tersedia *smartphone* berbasis sistem operasi Android di Indonesia. Android merupakan sistem operasi *mobile* yang menggunakan versi modifikasi dari kernel Linux. Android mulanya dikembangkan oleh Android Inc,, sebuah perusahaan yang kemudian dibeli oleh Google, yang kemudian akhir-akhir ini dikembangkan oleh Open Handset Alliance.

Android kini berada dalam posisi yang baik untuk berada pada tingkat atas dalam *smartphone* selama dua sampai tiga tahun. Sistem operasi ini menjadi pilihan yang baik bagi para vendor *smartphone* karena biaya lisensinya yang lebih murah dan sifatnya yang semi *open source*.

Karena merupakan *open source*, dengan demikian Android ini merupakan *software* yang dapat didistribusikan secara terbuka sehingga *developer* dapat bebas mengembangkan aplikasi baru di dalamnya. Para *developer* kemudian dapat menerbitkan aplikasi yang dikembangkannya pada *Android Market*. *Android Market* menyediakan ribuan aplikasi baik yang gratis maupun berbayar.

Dalam kaitannya dengan makalah ini, kita ketahui bahwa perkembangan algoritma kriptografi modern kini didorong oleh penggunaan komputer digital untuk keamanan pesan. Kriptografi, yang berasal dari bahasa Yunani, yaitu "*kriptos*" yang berarti menyembunyikan dan "*graphias*" yang berarti tulisan, merupakan salah satu cabang ilmu yang menyediakan teknik-teknik yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, integritas data, dan sebagainya. Meski demikian, tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi.

Pada makalah berikut ini akan dilakukan studi mengenai implementasi algoritma kriptografi dalam suatu teknologi perangkat *mobile*, dalam hal ini yaitu platform Android, yang merupakan penggunaan teknologi yang kian semakin menjadi tren saat ini. Penerapan kriptografi dalam platform *mobile* telah cukup banyak digunakan berkaitan dengan aspek keamanan yang kian kali sangat diperlukan dalam aktivitas *mobile* tersebut. Demikian pula dengan Android, yang telah menyediakan beberapa implementasi algoritma kriptografi pada API yang telah tersedia dalam hal ini yaitu "javax.crypto". Dengan API ini, dapat diimplementasikan fungsi-fungsi standar daripada kriptografi seperti enkripsi dan dekripsi, dengan bermacam-macam metode mulai dari *cipher*, *key agreement*, *message digest*, *signature*, dan sebagainya.

Pada makalah ini, penulis melakukan studi penelitian khususnya terhadap implementasi algoritma *block cipher* yang merupakan salah satu penggunaan kriptografi yang populer, pada pengembangan di Android. Adapun *IDE* yang digunakan adalah *Eclipse Hellios* yang telah ditambahkan dengan plugin *Android SDK Tools*. Algoritma yang dicoba untuk diterapkan dalam studi analisis pengembangannya antara lain adalah algoritma DES yang beroperasi pada blok 64 bit.

II. Algoritma Block Cipher

Block Cipher merupakan algoritma dalam kriptografi yang beroperasi dengan memanfaatkan bit, lain halnya dengan algoritma kriptografi klasik yang beroperasi dengan memanfaatkan karakter. Oleh karenanya, kunci, plainteks, dan cipherteks akan diproses dalam rangkaian

bit. Operasi bit yang biasa digunakan antara lain adalah XOR. Implementasi daripada algoritma *block cipher* lebih rumit jika dibandingkan dengan algoritma klasik, namun tetap menggunakan beberapa dasar dari algoritma klasik seperti substitusi dan transposisi.

Adapun algoritma *block cipher* memiliki prinsip dalam melakukan perancangan yang kuat. Prinsip ini dinamakan prinsip Shannon. Prinsip ini terdiri dari dua, yaitu :

1) *Confusion*

Prinsip ini menyembunyikan hubungan apapun yang ada antara plainteks, cipherteks, dan kunci. Sebagai contoh, pada *cipher* substitusi seperti *caesar cipher*, hubungan antara cipherteks dan plainteks mudah diketahui, karena satu huruf yang sama pada plainteks diganti dengan satu huruf yang sama pada cipherteksnya.

Prinsip ini sesungguhnya akan membuat kriptanalis kesulitan untuk mencari pola-pola statistik yang muncul pada cipherteks. *Confusion* yang bagus membuat hubungan statistik antara plainteks, cipherteks, dan kunci menjadi sangat rumit.

2) *Diffusion*

Prinsip ini menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Sebagai contoh, pengubahan kecil pada plainteks sebanyak satu atau dua bit menghasilkan perubahan pada cipherteks yang tidak dapat diprediksi.

Prinsip *diffusion* juga menyembunyikan hubungan statistik antara plainteks, cipherteks, dan kunci dan membuat kriptanalisis menjadi sulit dan rumit.

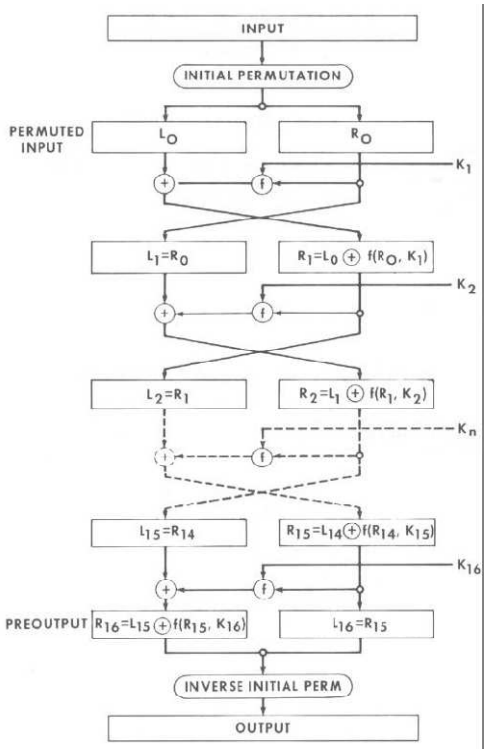
Untuk mendapatkan keamanan yang lebih tinggi, kedua prinsip inilah yang akan digunakan berkali-kali pada sebuah blok tunggal dengan kombinasi yang berbeda-beda.

II.1. DES (Data Encryption Standard)

Berikut ini merupakan spesifikasi daripada algoritma DES. *Data Encryption Standard* (DES) terdiri daripada algoritma *Data Encryption Algorithm* (DES) berikut dan *Triple Data Encryption Algorithm* (TDEA, sebagaimana dijelaskan pada ANSI X9.52). Algoritma ini harus didesain sedemikian rupa sehingga mereka dapat digunakan pada sistem komputer atau jaringan untuk menyediakan perlindungan kriptografi terhadap data biner. Metode daripada implementasi akan tergantung daripada lingkungan dan aplikasinya. Pengujian mungkin dapat dilakukan untuk memastikan keakuratan daripada algoritma tersebut. DES yang disebutkan disini merupakan salah satu algoritma block cipher yang akan penulis coba untuk diimplementasikan dalam perangkat *mobile* Android [5].

II.1.1 Cara Kerja DES (*Data Encryption Standard*)

Algoritma Data Encryption Standard (DES) ini didesain untuk enkripsi dan dekripsi blok data yang terdiri dari 64 bit dengan memanfaatkan kunci yang juga terdiri dari 64 bit. Dekripsi pun harus dilakukan dengan kunci yang sama yang digunakan untuk enkripsi, namun dengan langkah yang berbeda sehingga proses dekripsi adalah kebalikan daripada enkripsi. Blok yang akan dienkripsi mula-mula dilewatkan dengan fungsi *initial permutation* ***IP***, kemudian menuju perhitungan kompleks yang *key-dependent*, dan terakhir dilewatkan pada permutasi yang merupakan *inverse* daripada *initial permutation*, dengan kata lain adalah $\boldsymbol{IP^{-1}}$. Fungsi komputasi yang *key-dependent* tersebut dapat dibagi menjadi fungsi cipher, dapat disebut merupakan suatu fungsi ***f***, dan fungsi untuk *key scheduling yang* dapat kita sebut dengan fungsi ***KS***. Fungsi ***f*** merupakan fungsi yang memanfaatkan fungsi-fungsi seperti seleksi permutasi ***Si*** dan fungsi permutasi ***P***.



Gambar 1 Algoritma enkripsi DES

Proses enkripsi dapat dilihat pada Gambar 1. Suatu blok masukan sebesar 64 bit yang akan dienkripsi mula-mula dilewatkan pada operasi permutasi yang dinamakan *initial permutation* (***IP***) sebagai berikut.

Gambar 2 menunjukkan operasi *initial permutation* (***IP***) yang dapat dibaca sebagai berikut. Bit pertama hasil permutasi merupakan bit ke 58 daripada bit masukan, bit kedua hasil permutasi merupakan bit masukan ke 50, dan seterusnya sampai dengan bit terakhir yang merupakan bit ke 7 dari masukan. Blok yang telah dilakukan operasi permutasi ini kemudian dilewatkan lagi ke dalam fungsi kompleks yang *key-dependent*. Keluaran daripada fungsi kompleks tersebut, dapat kita sebut sebagai *pre-output*, yang kemudian kita lewatkan terhadap invers daripada fungsi *initial permutation* ($\boldsymbol{IP^{-1}}$).

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Gambar 2 Initial Permutation (IP)

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Gambar 3 Inverse Initial Permutation (IP^{-1})

Gambar 3 di atas menunjukkan operasi *inverse initial permutation* (***IP***) yang dapat dibaca sebagai berikut. Bit pertama hasil algoritma merupakan bit ke 40 daripada bit *pre-output*, bit kedua hasil algoritma merupakan bit *pre-output* ke 8, dan seterusnya sampai dengan bit terakhir yang merupakan bit ke 25 dari *pre-output*.

Fungsi komputasi rumit yang menggunakan blok yang telah dipermutasi sebagai masukannya menghasilkan blok *pre-output*. Pertukaran blok akan terjadi pada fungsi kompleks ini, dan dapat diiterasi sampai sekian kali, dalam hal algoritma ini adalah 16 iterasi. Fungsi ini akan bekerja terhadap dua blok, misalkan satu blok 32 bit dan satu blok 48 bit, yang akan menghasilkan blok keluaran sebesar 32 bit sebagaimana dapat dilihat ilustrasinya yang telah dijelaskan pada Gambar 1.

Misalkan kita ambil contoh sebuah blok masukan sebesar 64 bit, dilewatkan pada iterasi yang terdiri dari 32 bit blok ***L*** diikuti dengan 32 bit blok ***R***. Blok masukan kemudian dapat kita definisikan sebagai ***LR***.

Ambil ***K*** adalah blok 48 bit yang dipilih dari kunci 64 bit. Maka, keluaran ***L'R'*** dari iterasi dengan masukan ***LR*** didefinisikan sebagai

(1) $L' = R$

$R' = L \oplus f(R,K)$

dimana $\oplus$ merupakan penambahan bit-per-bit modulo 2. Sebagaimana disebutkan sebelumnya, masukan dari iterasi pertama perhitungan adalah blok masukan yang telah dilakukan operasi permutasi awal (***IP***).

Jika ***L'R'*** adalah keluaran daripada iterasi ke 16 maka **R'L'** adalah blok pre-output. Pada tiap iterasi, blok K yang berbeda daripada bit kunci kemudian dipilih dari kunci 64 bit.

Fungsi ***KS*** juga dapat didefinisikan sebagai fungsi yang menerima kunci 64 bit blok, kemudian mengambil integer sembarang dari 1 sampai dengan 16, kemudian menghasilkan 48 bit blok K_n, yang merupakan seleksi permutasi daripada bit dari kunci [5].

III. Platform Android

Android merupakan sistem operasi untuk telepon seluler berbasis Linux. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai macam perangkat *mobile*. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat perangkat lunak pada ponsel. Kemudian, untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi, anatara lain seperti Google, HTC, Intel, Motorola, Qualcomm T-Mobile, dan Nvidia.

Gambar 4 Logo Android

Di dunia ini, terdapat dua jenis distributor sistem operasi Android. Pertama, yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua, adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau yang dikenal sebagai *Open Handset Distribution* (OHD).

Android memiliki berbagai keunggulan sebagai perangkat lunak yang menggunakan basis kode komputer yang dapat didistribusikan secara terbuka (*open source*) sehingga pengguna bisa membuat aplikasi baru di dalamnya. Android memiliki aplikasi *native* Google yang terintegrasi seperti *pushmail Gmail*, *Google Maps*, dan *Google Calendar* [4].

III.1 Perkembangan Android

Android telah melalui berbagai perkembangan sebagai berikut.

1) Android 1.1

Android versi ini dilengkapi dengan pembaharuan estetika pada aplikasi, jam alarm, *voice search* (pencarian dengan suara), pengiriman pesan dengan Gmail, dan pemberitahuan e-mail

2) Android 1.5 (Cupcake)

Android versi ini dilengkapi dengan pembaharuan yang mencakup penambahan beberapa fitur yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa secara langsung,

dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan.

3) Android 1.6 (Donut)

Android versi ini dilengkapi dengan penampilan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, camcorder, dan galeri terintegrasi, CDMA/ EVDO, 802.1x, VPN, Gestures, *text-to-speech engine*, kemampuan *dial contact*, teknologi *text to change speech*, resolusi VWGA.

4) Android 2.0/2.1 (Éclair)

Perubahan yang terjadi pada Android 2.0/2.1 adalah *hardware* yang lebih optimal, peningkatan Google Maps 3.1.2, perubahan UI dengan dukungan browser baru dan HTML5, daftar *contact* baru, dukungan *flash* untuk kamera 3.2 MP, digital zoom, dan Bluetooth 2.1

5) Android 2.2 (Froyo : Frozen Yogurt)

Perubahan yang terjadi pada Android 2.2 adalah adanya dukungan Adobe Flash 10.1, kecepatan kinerja aplikasi telah 2 sampai 5 kali lebih cepat, integrasi V8 Javascript Engine yang dipakai Google Chrome mempercepat kemampuan *rendering* pada browser, pemasang aplikasi dalam *SD card*, kemampuan WiFi *hotspot portable*, dan kemampuan *auto update* dalam aplikasi *Android Market*.

6) Android 2.3 (Gingerbread)

Perubahan yang terjadi pada Android 2.3 adalah adanya peningkatan kemampuan *gaming*, peningkatan fungsi *copy paste*, perubahan layar UI, dukungan format video VP8 dan WebM, efek audio baru (*reverb*, *equalization*, *headphone virtualization*, *bass boost*), dukungan kemampuan NFC (*Near Feld Communication*), dan dukungan jumlah kamera yang lebih dari satu.

7) Android 3.0 (Honeycomb)

Android 3.0 dirancang khusus untuk tablet yang telah mendukung ukuran layar yang lebih besar. Dengan demikian, UI (*user interface*)-nya pun berbeda karena memang didesain khusus untuk tablet. Android 3.0 juga mendukung multiprosesor dan juga akselerasi perangkat keras untuk grafis.

Kini dengan semakin berkembangnya Android, semakin bertambahnya jumlah perangkat *mobile* Android, semakin banyak pihak ketiga yang berminat untuk mengembangkan dan menyalurkan aplikasi mereka kepada Android. Adapun Android yang akan digunakan dalam melakukan studi implementasi pada makalah ini adalah Android versi 2.3.3, yaitu Android Gingerbread.

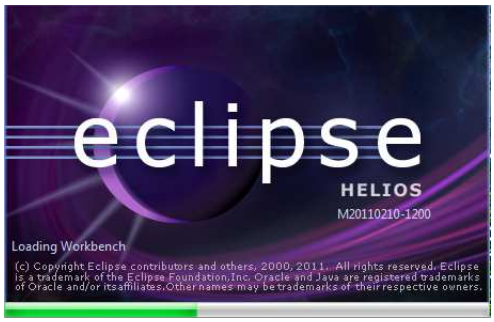
IV. PENGEMBANGAN IMPLEMENTASI ALGORITMA BLOCK CIPHER PADA ANDROID

Bagian ini akan membahas lebih rinci mengenai bagaimana pengembangan algoritma kriptografi dalam hal ini yaitu *block cipher* dapat diimplementasikan pada teknologi *mobile* Android. Adapun hal yang perlu dilakukan antara lain seperti instalasi Android SDK *tools*, penggunaan suatu *IDE* untuk pengembangan yang lebih mudah, dan berbagai pengaturan-pengaturan lainnya yang dibutuhkan.

IV.1 Instalasi SDK

Berikut ini adalah langkah-langkah dalam melakukan pengaturan SDK [2].

1) Lakukan instalasi *Eclipse Classic IDE Helios*. Dalam studi pengembangan, penulis menggunakan *Eclipse Classic Helios 3.6.2*.
2) Lakukan instalasi *SDK starter package*.
3) Lakukan instalasi *ADT Plugin* pada *Eclipse IDE*.
4) Lakukan penambahan platform Android dan komponen lainnya dalam SDK.
5) Melakukan setting AVD (*Android Virtual Device*).



Gambar 5 Eclipse Hellios IDE

Melakukan penambahan dan pembaharuan komponen pada SDK sangat mudah dan cepat. Untuk melakukan penambahan dan pembaharuan komponen dapat dilakukan dengan menggunakan *Android SDK dan AVD Manager* (termasuk dalam SDK Tools).

Semua kakas pengembangan yang diperlukan ini dapat diunduh melalui situs pengembang Android itu sendiri, yaitu http://developer.android.com/.

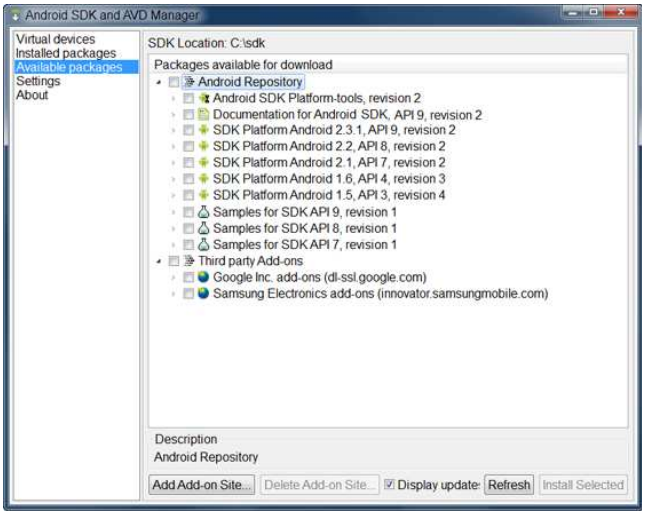
IV.2 Instalasi ADT Plugin dan Pengaturan AVD pada Eclipse

Adapun instalasi ADT Plugin pada Eclipse dilakukan dengan langkah-langkah sebagai berikut [2].

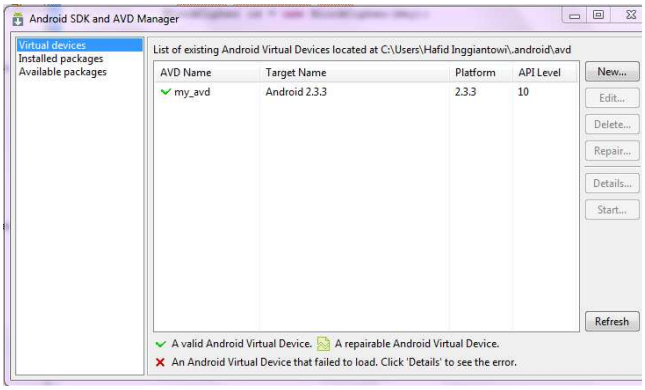
1) Jalankan Eclipse, kemudian pilih Help → Install New Software.
2) Pilih Add, pada pojok kanan atas.
3) Pada Add Repository, untuk *name* isikan dengan nama sembarang yang diinginkan, 'ADT Plugin' misalnya, sedangkan untuk *location* isikan dengan https://dl-ssl.google.com/android/eclipse/.

Kemudian pilih OK.

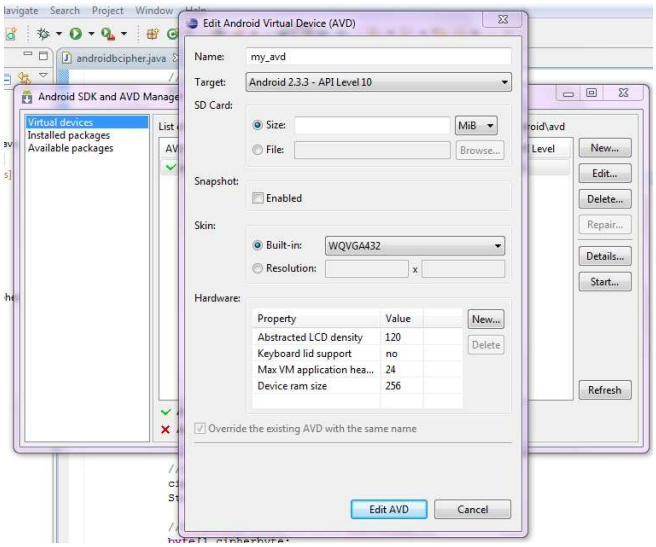
4) Pada Available Software, centang *checkbox* di samping *Developer Tools*, kemudian pilih Next
5) Pada jendela berikutnya kemudian dapat kita lihat daftar *tools* yang dapat diunduh, kemudian pilih Next.
6) Baca dan setujui semua persetujuan, kemudian pilih Finish
7) Lakukan restart Eclipse
8) Setelah berhasil, lakukan konfigurasi ADT Plugin
9) Konfigurasi ADT Plugin dilakukan dengan memilih *Window* → *Preferences*, kemudian memilih Android pada panel sebelah kiri
10) Pada *SDK Location*, lakukan *Browse* dan isikan dengan lokasi direktori SDK yang telah kita miliki. Kemudian, pilih Apply, dan pilih OK.
11) Listing AVD dilakukan untuk menentukan emulator yang akan digunakan dalam menampilkan aplikasi Android. Hal ini dilakukan pada Android SDK dan AVD Manager sebagaimana dapat dilihat pada Gambar 7 dan 8.



Gambar 6 Android SDK dan AVD Manager



Gambar 7 Virtual Devices



Gambar 8 Pengaturan AVD

IV.3 Pustaka Kriptografi pada Android

Android telah menyediakan API yang mendefiniskan fungsi-fungsi kriptografi. API ini dapat digunakan dengan melakukan import "javax.crypto" pada bagian atas kode program.

API yang terkait dengan implementasi kriptografi dapat dilihat rinciannya sebagai berikut.

javax.crypto	Menyediakan kelas dan interface untuk aplikasi kriptografi yang dapat mengimplementasikan enkripsi, dekripsi, maupun key agreement.
javax.crypto.interfaces	Menyediakan interface untuk mengimplentasikan algoritma key agreement Diffie-Hellman (DH) sebagaimana ditetapkan PKCS#3.
javax.crypto.spec	Menyedikan kelas dan interface yang dibutuhkan untuk menetapkan kunci dan parameter-parameter untuk enkripsi

Pustaka javax.crypto yang akan kita gunakan ini menyediakan kelas dan *interface* untuk aplikasi kriptografi yang dapat mengimplementasikan berbagai fungsi kriptografi seperti enkripsi, dekripsi, maupun *key agreement*.

Pustaka ini mendukung *stream cipher*, dan juga *asymmetric* serta *symmetric block cipher*. Implemetasi *cipher* dari penyedia yang berbeda dapat diintegrasikan menggunakan kelas abstrak SPI (*Service Provider Interface*). Dengan kelas *Sealed Object* juga, *programmer*

dapat mengamankan suatu objek dengan mengenkripsinya menggunakan cipher.

Autentikasi yang dapat dilakukan berbasis MAC (*Message Autenthication Code*) seperti HMAC (*Hash MAC*, sebagai contoh dengan fungsi hash SHA-1).

Berikut ini adalah rincian *interface*, kelas, serta *exception*, dan sebagainya yang dapat digunakan dari API "javax.crypto" [1].

Interfaces

SecretKey	A cryptographic secret (symmetric) key.

Classes

Cipher	This class provides access to implementations of cryptographic ciphers for encryption and decryption.
CipherInputStream	This class wraps an InputStream and a cipher so that read() methods return data that are read from the underlying InputStreamand processed by the cipher.
CipherOutputStream	This class wraps an output stream and a cipher so that write methods send the data through the cipher before writing them to the underlying output stream.
CipherSpi	This class defines the *Service Provider Interface* (**SPI**) for cryptographic ciphers.
EncryptedPrivateKeyInfo	This class implements the EncryptedPrivateKeyInfo ASN.1 type as specified in PKCS #8 - Private-Key Information Syntax Standard.
ExemptionMechanism	This class implements the functionality of an exemption mechanism such as *key recovery*, *key weakening*, or *key escrow*.
ExemptionMechanismSpi	The *Service Provider Interface* (**SPI**) definition for the ExemptionMechanism class.
KeyAgreement	This class provides the functionality for a key exchange protocol.
KeyAgreementSpi	The *Service Provider Interface* (**SPI**) definition for the KeyAgreement class.
KeyGenerator	This class provides the public API for generating symmetric cryptographic keys.
KeyGeneratorSpi	The *Service Provider Interface* (**SPI**) definition for the KeyGenerator class.
Mac	This class provides the public API for *Message Authentication Code* (MAC) algorithms.
MacSpi	The *Service-Provider Interface* (**SPI**) definition for the Mac class.
NullCipher	This class provides an identity cipher that does not transform the input data in any way.
SealedObject	A SealedObject is a wrapper around a serializable object instance and encrypts it using a cryptographic cipher.
SecretKeyFactory	The public API for SecretKeyFactory implementations.
SecretKeyFactorySpi	The *Service Provider Interface* (**SPI**) definition for the SecretKeyFactory class.

Exceptions

BadPaddingException	The exception that is thrown when a padding mechanism is expected for the input data, but the input data does not have the proper padding bytes.
ExemptionMechanismException	This is the base class for ExemptionMechanismException.
IllegalBlockSizeException	The exception, that is thrown when the data length provided to a block cipher does not match the block size of the cipher.

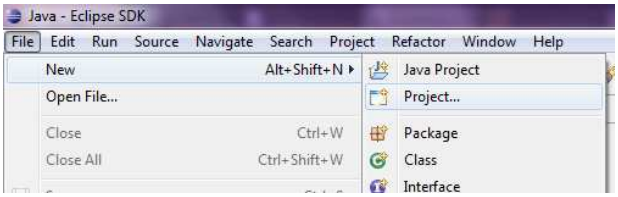
NoSuchPaddingException	The exception that is thrown when the requested padding mechanism is not supported.
ShortBufferException	The exception that is thrown when the result of an operation is attempted to store in a user provided buffer that is too small.

IV.4 Implementasi Block Cipher pada Android

Penulis membuat sebuah aplikasi berbasis Java dengan menggunakan *Eclipse Hellios IDE* yang telah ditambahkan plugin Android SDK.

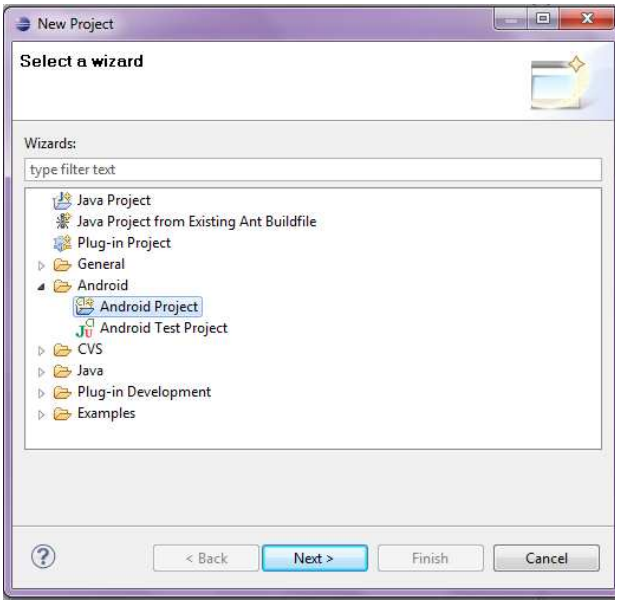
Berikut ini adalah langkah sederhana yang dapat digunakan dalam membuat aplikasi enkripsi dekripsi *block cipher* pada Android. Adapun sebelum memulai langkah implementasi ini, langkah yang telah dijelaskan pada bab IV.1 dan IV.2 harus telah dilakukan terlebih dahulu.

1) Mula-mula pilih File, New Project (lihat Gambar 9).



Gambar 9 Membuat project baru

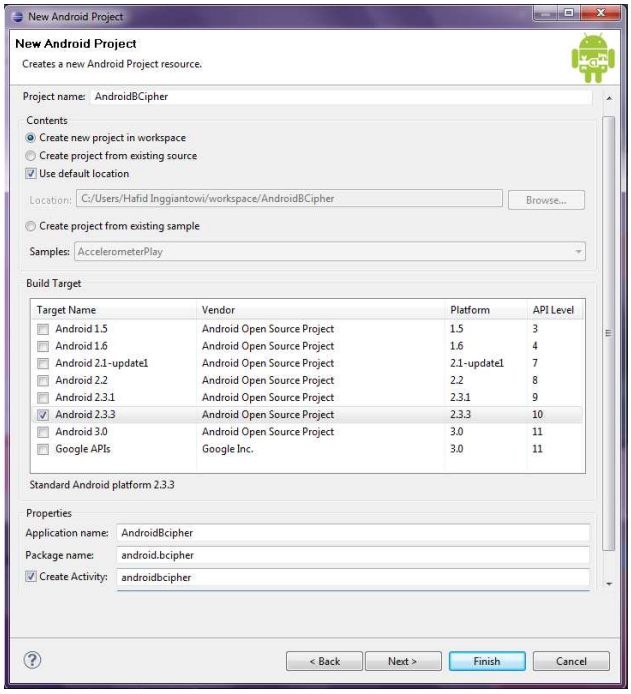
2) Selanjutnya pilih Android, kemudian pilih Android Project.



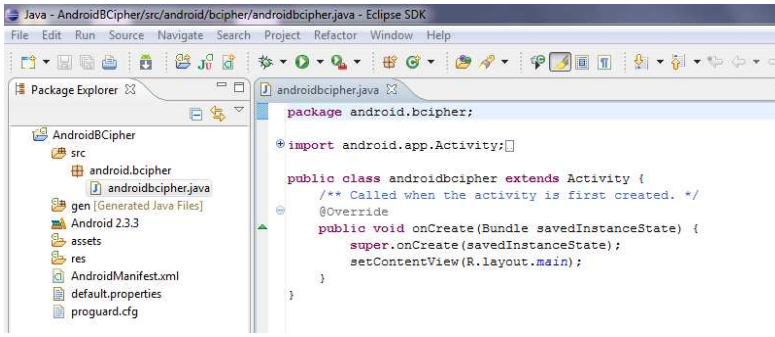
Gambar 10 Membuat project Android

3) Pada jendela New Android Project, lakukan pengaturan mulai dari nama project, build target Android yang akan digunakan (dalam makalah ini penulis menggunakan Android 2.3.3), nama activity, package, min SDK, dan sebagainya.

4) Selanjutnya akan didapatkan project otomatis yang telah di-*generate* sebagaimana dapat dilihat pada Gambar 12.



Gambar 11 New Android Project pada Eclipse IDE



Gambar 12 Android Project Package

Berikut ini adalah *source code* sederhana yang penulis buat dalam menganalisis implementasi API kriptografi pada pustaka Android. *Source code* dibuat dalam bentuk pembuatan aplikasi enkripsi dekripsi sederhana yang menggunakan algoritma *block cipher* DES pada Android.

File Blockcipher.java mendefinisikan algoritma enkripsi dan dekripsi *block cipher* DES yang digunakan, sedangkan untuk file layoutandroid.xml mendefinisikan antarmuka daripada tampilan di perangkat *mobile* Android, dan kemudian, androidbcipher.java

mendefinisikan penanganan dalam menerima masukan teks daripada pengguna pada perangkat *mobile* serta menghasilkan enkripsi maupun dekripsinya. Pada pengembangan Android akan terdapat juga *auto generated file* yang akan didefinisikan secara otomatis untuk menghubungkan antarmuka dan implementasi program, dalam hal ini yaitu R.java.

BlockCipher.java

```
package android.bcipher;

import
java.io.UnsupportedEncodingException;
import
java.security.GeneralSecurityException;
import javax.crypto.Cipher;
import javax.crypto.spec.DESKeySpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class BlockCipher {
	public static int MAX_KEY_LENGTH =
DESKeySpec.DES_KEY_LEN;
	private static String
ENCRYPTION_KEY_TYPE = "DES";
	private static String
ENCRYPTION_ALGORITHM =
"DES/CBC/PKCS5Padding";
  private final SecretKeySpec keySpec;

  public BlockCipher(String passphrase) {
    byte[] key;

    try {
       //Ambil representasi byte dari
kunci
      key = passphrase.getBytes("UTF8");
    } catch (UnsupportedEncodingException
e) {
      throw new
IllegalArgumentException(e);
    }

    key = padKeyToLength(key,
MAX_KEY_LENGTH);
    keySpec = new SecretKeySpec(key,
ENCRYPTION_KEY_TYPE);
  }

  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;
  }

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

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

  private byte[] doCipher(byte[] original,
int mode) throws GeneralSecurityException {
        //Misalkan mode CBC, dengan
padding yang didefinisikan
        Cipher cipher =
Cipher.getInstance(ENCRYPTION_ALGORITHM);
    // IV = 0
    IvParameterSpec iv = new
IvParameterSpec(new byte[] { 0, 0, 0, 0, 0,
0, 0, 0 });
    cipher.init(mode, keySpec, iv);
    return cipher.doFinal(original);
  }

      //Fungsi-fungsi konversi
      public String toHex(String txt) {
            return toHex(txt.getBytes());
      }

      public String fromHex(String hex) {
            return new
String(toByte(hex));
      }

      public byte[] toByte(String
hexString) {
            int len =
hexString.length()/2;
            byte[] result = new
byte[len];
            for (int i = 0; i < len; i++)
                  result[i] =
Integer.valueOf(hexString.substring(2*i,
2*i+2), 16).byteValue();
            return result;
      }

      public String toHex(byte[] buf) {
            if (buf == null)
                  return "";
            StringBuffer result = new
StringBuffer(2*buf.length);
            for (int i = 0; i <
buf.length; i++) {
                  appendHex(result,
buf[i]);
            }
            return result.toString();
      }
      private final static String HEX =
"0123456789ABCDEF";
      private static void
appendHex(StringBuffer sb, byte b) {

      sb.append(HEX.charAt((b>>4)&0x0f)).a
ppend(HEX.charAt(b&0x0f));
      }

}
```

User Interface layoutandroid.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/a
```

```
pk/res/android"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

android:orientation="vertical" >
    <TextView android:id="@+id/text"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
              android:text="Plaintext /
Ciphertext" />
    <EditText
android:layout_width="match_parent"

       android:id="@+id/editText1"

       android:layout_height="wrap_content"
             android:text=" "/>
    <TextView android:text="Key"

android:id="@+id/textView2"

android:layout_width="wrap_content"

android:layout_height="wrap_content"/>
    <EditText
android:layout_width="match_parent"

android:id="@+id/editText2"

android:layout_height="wrap_content"
                  android:text=" " />
    <LinearLayout
android:id="@+id/linearLayout1"
android:layout_height="wrap_content"
android:layout_width="match_parent">
        <Button
android:layout_width="wrap_content"
        android:id="@+id/button"
android:text="Encrypt"

android:layout_height="wrap_content"></Butt
on>
    </LinearLayout>
    <TextView  android:id="@+id/textView1"

android:layout_width="wrap_content"

android:layout_height="wrap_content"
                  android:text= "Encrypted
Text"/>
    <EditText
android:layout_width="match_parent"
    android:layout_height="wrap_content"
android:id="@+id/editText3"
    android:text=" "></EditText>
    <Button android:text="Decrypt"
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"></Butt
on>
    <TextView android:text="Decrypted
Text" android:id="@+id/textView3"
    android:layout_width="wrap_content"
android:layout_height="wrap_content">
    </TextView>
    <EditText
android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editText4"
android:text=" "></EditText>
</LinearLayout>
```

Androidbcipher.java

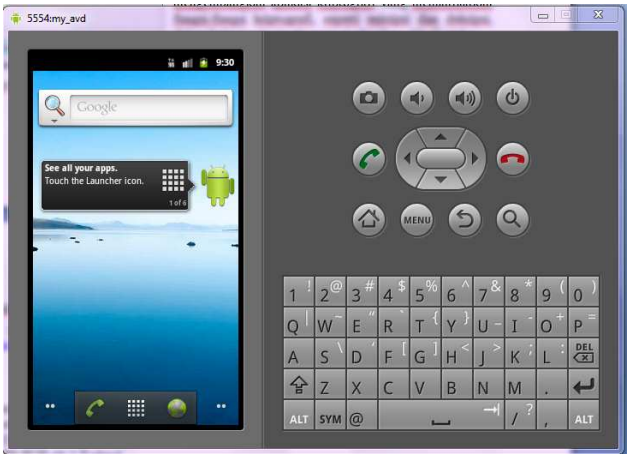
```
package android.bcipher;

import
java.security.GeneralSecurityException;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class androidbcipher extends
Activity{
       // Button Encryption Listener
    private OnClickListener EncryptListener
= new OnClickListener() {
        public void onClick(View v) {
          // do something when the button
is clicked
              EditText plaint;
       //plaintext dalam EditText
              EditText keyt;
       //kunci dalam EditText

             //Mengambil string plainteks
                    plaint =
(EditText)findViewById(R.id.editText1);

                    String plaintext =
plaint.getText().toString();

                    //Mengambil array of
byte plainteks
                    byte[] plaintbyte;

                    //Mengambil string
kunci
                    keyt = (EditText)
findViewById(R.id.editText2);
                    String key =
keyt.getText().toString();

                    //Membuat instansiasi
kelas BlockCipher
                    BlockCipher cs = new
BlockCipher(key);

                    //Menjadikan byte
                    plaintbyte =
plaintext.getBytes();

                    try {
                          //Melakukan
enkripsi
                          String
encryptresult =
cs.toHex(cs.encrypt(plaintbyte));

                          //Menampilkan
```

```
hasil enkripsi pada TextView
                        EditText enct;
                        enct =
(EditText)findViewById(R.id.editText3);

      enct.setText(encryptresult);

                    } catch
(GeneralSecurityException e) {

      e.printStackTrace();
                    }
       }
    };

      // Button Decryption Listener
    private OnClickListener DecryptListener
= new OnClickListener() {
        public void onClick(View v) {
          // do something when the button
is clicked
              EditText ciphert;
      //ciphertext dalam EditText
              EditText keyt;
      //kunci dalam EditText

            //Mengambil string cipherteks
                    ciphert =
(EditText)findViewById(R.id.editText3);

                    String ciphertext =
ciphert.getText().toString();

                    //Mengambil array of
byte cipherteks
                    byte[] cipherbyte;

                    //Mengambil string
kunci
                    keyt = (EditText)
findViewById(R.id.editText2);
                    String key =
keyt.getText().toString();

                    //Membuat instansiasi
kelas BlockCipher
                    BlockCipher cd = new
BlockCipher(key);

                    //Menjadikan byte
                    cipherbyte =
cd.toByte(ciphertext);

                    try {
                        //Melakukan
dekripsi
                        String
decryptresult =
cd.fromHex(cd.toHex(cd.decrypt(cipherbyte))
);

                        //Menampilkan
hasil dekripsi pada TextView
                        EditText dect;
                        dect =
(EditText)findViewById(R.id.editText4);

      dect.setText(decryptresult);
                    } catch
(GeneralSecurityException e) {

      e.printStackTrace();
                    }

        }
    };

      /** Called when the activity is
first created. */
    @Override
    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);

setContentView(R.layout.layoutandroidcipher
);

        Button buttondec =
(Button)findViewById(R.id.button1);

buttondec.setOnClickListener(DecryptListene
r);

        Button button =
(Button)findViewById(R.id.button);

button.setOnClickListener(EncryptListener);

    }
}
```

Auto generated file R.java

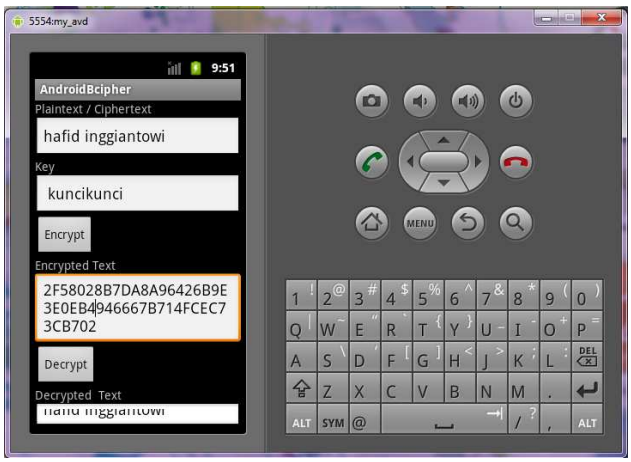
```
  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
   *
   * This class was automatically generated
by the
   * aapt tool from the resource data it
found.  It
   * should not be modified by hand.
   */

  package android.bcipher;

  public final class R {
      public static final class attr {
      }
      public static final class drawable {
          public     static     final     int
icon=0x7f020000;
      }
      public static final class id {
          public     static     final     int
button=0x7f050005;
          public     static     final     int
button1=0x7f050008;
          public     static     final     int
editText1=0x7f050001;
          public     static     final     int
editText2=0x7f050003;
          public     static     final     int
editText3=0x7f050007;
          public     static     final     int
editText4=0x7f05000a;
```

```
        public static final int
linearLayout1=0x7f050004;
        public static final int
text=0x7f050000;
        public static final int
textView1=0x7f050006;
        public static final int
textView2=0x7f050002;
        public static final int
textView3=0x7f050009;
    }
    public static final class layout {
        public static final int
layoutandroidcipher=0x7f030000;
    }
    public static final class string {
        public static final int
Decrypt=0x7f040002;
        public static final int
app_name=0x7f040001;
        public static final int
hello=0x7f040000;
    }
}
```



Gambar 13 Antarmuka depan

Gambar 14 Antarmuka memilih aplikasi



Gambar 15 Hasil tampilan aplikasi sederhana DES yang dibuat pada emulator AVD

Adapun hasil tampilan daripada implementasi program dapat dilihat pada emulator *Android Virtual Device* yang telah di-run dan dapat dilihat pada Gambar 15. Program yang dibuat merupakan program sederhana yang menerima masukan pengguna dari papan ketik *touchscreen* maupun *keyboard* dari perangkat *mobile*. Hasil enkripsi kemudian ditampilkan dengan menggunakan bilangan heksadesimal.

V. KESIMPULAN

Android merupakan salah satu teknologi *mobile* yang kian semakin menjadi tren saat ini. Banyak orang yang menggunakan perangkat *mobile* yang telah menggunakan sistem operasi Android. Android merupakan perangkat lunak berbasis *open source* sehingga pengembang dapat bebas mengembangkan aplikasi baru di dalamnya secara terbuka.

Pada Android SDK, API khusus kriptografi telah disediakan dan dapat digunakan dengan mudah dalam mengembangkan aplikasi kriptografi yang memanfaatkan fungsi-fungsi kriptografi, seperti enkripsi dan dekripsi, dalam hal ini yaitu implementasi algoritma *block cipher*. Dengan demikian, API ini dapat digunakan dalam meningkatkan aspek keamanan dalam teknologi *mobile* Android seperti kerahasiaan data, integritas data, keabsahan data, dan lain sebagainya.

PUSTAKA

[1] *Android Developers*. Diakses Maret 2011, http://developer.android.com/reference/javax/crypto/package-summary.html

[2] *Android Developers* : *Installing the SDK*. Diakses Maret 2011, http://developer.android.com/sdk/installing.html

[3] *Android Security : How to encrypt and decrypt strings?*. Diakses Maret 2011, Android Self.training();: http://www.tutorials-android.com/learn/How_to_encrypt_and_decrypt_ strings.rhtml

[4] *Indosat : Android*. Diakses Maret 2011. http://www.indosat.com/android

[5] *Data Encryption Standard (DES)*. (1999, October 25). Retrieved Maret 2011, from FIPS PUB 46-3 Federal Information Processing Standards Publication, U.S. Department of Commerce/National Institute

of Standards and Technology: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
[6] Munir, Rinaldi. *Algoritma Kriptografi Modern (Bagian I)*. Diakses Maret 2011, from http://www.informatika.org/~rinaldi/Kriptografi/2010-2011/Algoritma%20Kriptografi%20Modern_bag1%20(baru).ppt
[7] Rafael Pass, A. S. (n.d.). *A Course in Cryptography*. Diakses Maret 2011, http://www.cs.cornell.edu/ courses/cs4830/2010fa/lecnotes.pdf
[8] *The Story of Android, Cryptography and a crippled 3DES*. (2010, September 20). Diakses Maret 2011, The World According to Koto on Security, Malware, Cryptography, Pentesting, Javascript, Php, and Whatnots: http://blog.kotowicz.net/2010/09/story-of-android-cryptography-and.html.

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

Hafid Inggiantowi 13507094