Implementasi Algoritma Rijndael pada Aplikasi Android Pengirim Short Message Service (SMS) Terenkripsi

Tegar Aji Pangestu / 13512061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512061@std.stei.itb.ac.id

***Abstrak* — Makalah ini membahas tentang penerapan algoritma Rijndael dalam pengiriman SMS secara terenkripsi. Untuk mengantisipasi celah keamanan pada SMSC saat mengirim SMS, perlu diterapkan sebuah algoritma yang dapat mengenkripsi pesan sehingga pesan yang dikirimkan hanya dapat diterjemahkan oleh penerima. Algoritma ini terbukti mempunyai tingkat sekuritas yang tinggi dan menjamin kerahasiaan informasi yang disimpan. Di makalah ini saya akan mengulas bagaimana algoritma tersebut dapan diimplementasikan dalam platform Android. Platform ini dipilih karena kustomisasi yang lebih luas untuk pengembang.**

***Kata Kunci*—Android, Kriptografi, Rijndael, SMS.**

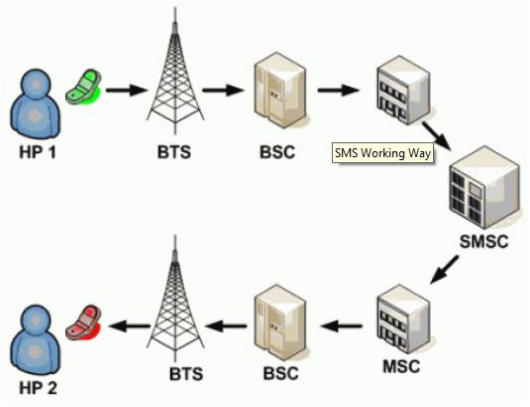
I. PENDAHULUAN

Dewasa ini, telepon selular *(Handphone)* merupakan perangkat komunikasi yang banyak digunakan oleh orang-orang untuk berkomunikasi satu sama lain. Mobilitas dan kecanggihan fiturnya menjadi alasan banyak orang menggunakan telepon selular. Berbagai korporat telah mengintegrasikan layanannya dengan telepon selular. Seperti contoh, penggunaan internet banking dengan SMS untuk transaksi, pemesanan tiket online dan lain-lain.

Dengan kemajuan teknologi yang sangat pesat, telepon selular telah bertransformasi menjadi *smartphone*. Smartphone adalah salah satu tipe perangkat *mobile* yang memiliki kompetensi melebihi telepon selular biasa. Sebuah smartphone berjalan dalam sebuah operating system yang memberikan antarmuka dan platform yang terstandarisasi untuk pengembangnya.

Salah satu platform yang banyak digunakan sebagai Operating System (OS) sebuah *smartphone* adalah Android. Android pertama kali dikembangkan pada 5 November 2007 dan pertama kali dirilis pada April 2009 dengan Android 2.1 Cupcake. Operating system ini banyak disukai oleh pengembang karena menggunakan bahasa *java* yang familiar dan kustomisasi yang lebih terbuka.

Fitur fundamental yang terdapat di setiap telepon selular sekarang ini salah satunya adalah *Short Message Service* (SMS). SMS merupakan salah satu fitur untuk berkirim pesan singkat antar pengguna dengan menggunakan provider selular. Meskipun banyak aplikasi *instant messaging* bermunculan, SMS masih menjadi salah satu fitur yang lebih disukai karena generalitasnya yang dapat mencakup semua jenis telepon selular. Dengan semakin banyaknya pengguna telepon selular dan banyaknya pengiriman SMS antar pengguna, sekuritas pengiriman SMS menjadi isu yang patut untuk diperhatikan. Industri SMS menjadi salah satu titik rawan untuk serangan ancaman sekuritas.



Gambar 1.1

SMS bekerja dalam jaringan nirkabel. Dalam aplikasinya, pentrasmisian SMS membutuhkan beberapa komponen khusus untuk mengirimkan pesan sampai ke tujuan. Komponen yang diperlukan untuk melakukan komunikasi SMS diantaranya adalah :

a. BTS (*Base Transceiver Station*)
   BTS ini merupakan sebuah perangkat yang memfasilitasi komunikasi nirkabel antara perangkat user dengan jaringan. Perangkat user ini dapat meliputi telepon selular, komputer dengan koneksi internet nirkabel, dan lain-lain
b. MSC (*Mobile Switching Center*)
   MSC ini adalah sebuah noda layanan pengiriman utama bagi GSM/CDMA. Perangkat ini berfungsi untuk *routing* panggilan suara, SMS, FAX, maupun *conference call*.
c. SMSC (*SMS Service Center*)
   SMSC adalah sebuah perangkat yang terpasang pada jaringan utama SMSC ini berfungsi untuk menerima SMS dan menelusuri nomor tujuan, dan mengirimkannya ke perangkat tujuan (Telepon Seluler). SMSC ini juga berperan

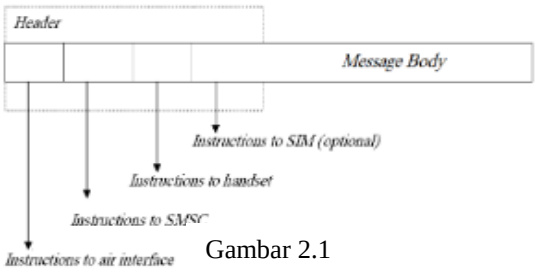
sebagai penyimpanan sementara untuk SMS. Jadi, jika nomor tujuan tersebut tidak aktif, SMS tersebut akan tersimpan pada SMSC dan SMSC akan mengirimkannya kembali jika perangkat tujuan telah aktif kembali. Sebagai tambahan, SMSC akan memberikan notifikasi kepada pengirim apakah pengiriman SMS tersebut berhasil ataupun tidak. Namun, karena keterbatasan memori penyimpanan, SMSC tidak dapat menyimpan SMS untuk jangka waktu yang lama.

Celah keamanan terbesar pada pengiriman SMS adalah pada saat SMS tersebut tersimpan pada SMSC. Sehingga, jika dilakukan serangan pada server SMSC, akan mengakibatkan pesan yang dikirim akan dapat dibaca orang lain yang tidak berhak. Maka, salah satu cara efektif untuk mengatasinya adalah dengan melakukan enkripsi pada pesan yang akan dikirimkan. Dengan melakukan enkripsi, maka pesan yang terbajak dari SMSC tidak dapat diterjemahkan langsung oleh pelaku. Dalam makalah ini, akan dijelaskan salah satu algortima yang digunakan untuk enkripsi pesan yaitu algoritma *Rijndael*. Algoritma ini merupakan pemenang lomba algoritma kriptografi yang diadakan *National Institue of Standarts and Technology* (NIST) pada tahun 2000. Algoritma ini memiliki kelebihan sebagai algoritma tercepat dan memiliki pemanfaatan hardware paling efisien.

II. TEORI

2.1 Struktur Pesan SMS

Struktur pesan dalam sebuah paket SMS dapat dilihat pada gambar dibawah berikut :



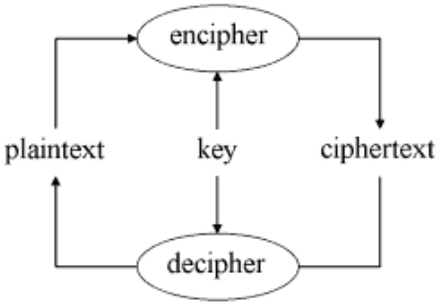
Gambar 2.1

Pada gambar diatas terdapat instruksi-instruksi yang diperlukan untuk mengirimkan SMS tepat pada tujuan. Pada bagian *message body*, terdapat isi pesan yang akan disampaikan

2,2 Algoritma Kriptografi

2.2.1 Definisi Kriptografi dan Istilah Umum

Istilah kriptografi diturunkan dari istilah Yunani *cruptos* yang berarti tersembunyi dan *logos* yang berarti ilmu. Kriptografi adalah ilmu atau studi tentang pembuatan suatu sistem dan penggunaannya untuk berkomunikasi secara rahasia dalam *channel* yang tidak aman. Pengirim menjaga kerahasiaan pesannya dengan mentransformasi data yang disebut *plaintext* ke dalam bentuk yang tidak dapat dimengerti secara langsung, yang disebut *cipher text*. Proses tersebut disebut enkripsi. Penerima menerjemahkan kembali *cipher text* menjadi *plain text* kembali. Proses ini disebut dekripsi. Sebuah kunci kriptografi mengendalikan proses enkripsi dan dekripsi data. Hal ini ditunjukkan pada gambar 1.1



Gambar 2.2

Dalam era digital sekarang, kebutuhan akan sekuritas pengiriman data menjadi hal yang sangat penting. Dengan menggunakan berbagai algoritma kriptografi, orang-orang dapat mengamankan transfer informasi yang dilakukannya agar tidak diketahui oleh orang yang tidak berkepentingan.

2.2.2 Klasifikasi Algoritma

Algoritma Kriptografi dapat diklasifikasikan sebagai berikut :

a. Algoritma Simetri
   Algoritma Simetri adalah salah satu jenis algoritma kriptografi yang memiliki kunci enkripsi dan dekripsi yang sama. Metode ini lebih cepat diimplementasikan daripada algoritma asimetri, karena pengirim dan penerima menggunakan kunci yang sama untuk mengirimkan dan menerima informasi. Namun, algorima ini lebih rawan daripada algoritma asimetri karena jika salah satu kunci diketahui, semua orang dapat mengetahui isi informasi yang terenkripsi. Contohnya adalah IDEA (International Data Encryption Algorithm), FEAL (Fast Data Encipherment Algorithm), DES (Data Encryption Standard), Triple DES, LOKI
b. Algoritma Asimetri
   Algoritma Asimetri adalah algoritma yang didesain sehingga kunci untuk enkripsi berbeda dengan kunci untuk dekripsi. Pengirim dapat mempublikasikan kunci enkripsi. Namun hanya penerima yang dapat mendekripsi informasi yang diterimanya. Metode ini lebih lambat daripada algoritma simetri. Contohnya adalah RSA dan Diffle-Hellman.

2.2.3 Cipher Block

Block cipher adalah sebuah fungsi yang memetakan n-bit *plain text* menjadi n-bit *cipher text blocks.* Ukuran blok cipher ditentukan oleh pertimbangan kompleksitas kriptografik dan ukuran tersebut harus

cukup besar untuk menghindari serangan *message exhaustion* sederhana. Sebagai contoh, dengan mencoba-coba semua kemungkinan kombinasi plainteks dengan suatu kunci yang dipilih, suatu lawan dapat membuat suatu kamus yang berisi *cipher text* dan *plain text* yang bersesuaian.

Walaupun demikian, jika ukuran blok cukup besar, maka kamus akan terlalu besar untuk dibuat atau disimpan. Serangan lainnya juga harus dipertimbangkan sebelum menentukan ukuran blok yang dapat digunakan. Sebagai contoh, model serangan dapat diambil dari fakta bahwa beberapa blok data lebih sering muncul dari pada yang lain. Tipe dari serangan ini disebut analisis frekuensi blok dan menggunakan metode statistik. Hal tersebut mirip dengan suatu analisis yang akan dilakukan pada suatu substitusi cipher dengan melakukan perhitungan frekuensi huruf.

2.3 Algoritma Rijndael

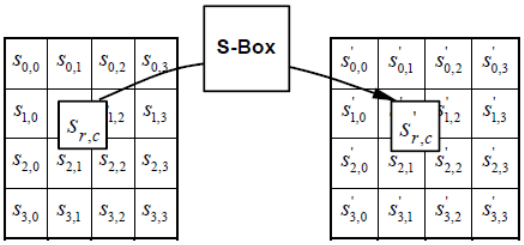
Algoritma Rijndael dapat diimplementasikan dengan panjang kunci 128 bit sampai dengan 256 bit. Karena panjang kunci yang bervariasi antara 128, 192, dan 256, maka dikenal AES-128, AES-192, AES-256, dimana AES merupakan singkatan dari *Advanced Encryption Standard*. Algoritma ini menggunakan sejumlah ronde enkripsi, dimana output dari ronde enkripsi saat ini akan menjadi input ronde enkripsi selanjutnya. Output dari round terakhir disebut dengan *cipher text*. Input yang diberikan oleh user dientrikan kedalam matriks yang disebut *State Matrix*

2.3.1 Langkah-langkah pada Tahap Enkripsi

Berikut ini langkah-langkah yang digunakan pada tahap enkripsi :

2.3.1 SubBytes

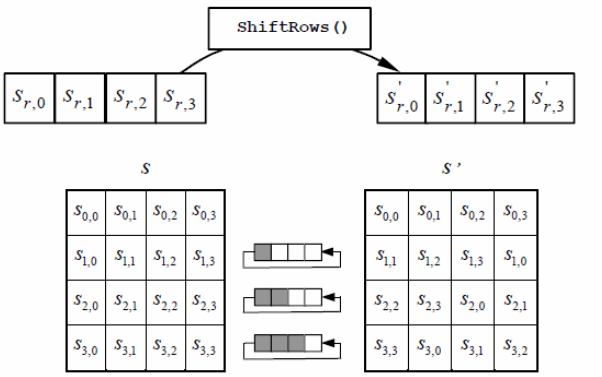
Pada tahap SubBytes ini terjadi substitusi byte dengan menggunakan tabel substitusi.



Gambar 2.3

2.2.2 ShiftRows

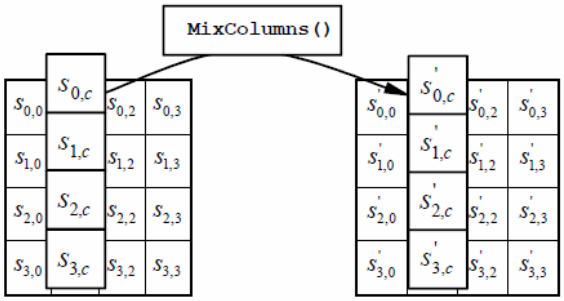
Pada tahap ShiftRows ini dilakukan pergeseran posisi byte dalam suatu baris secara siklik dengan offset tertentu. Pada baris pertama, bit tidak digeser. Pada baris kedua, masing-masing byte digeser sejauh 1 byte dari posisi semula. Pada baris ketiga, byte digeser sejauh 2 byte dari posisi semula. Begitu pula dengan baris selanjutnya, baris n digeser sejauh n-1 byte.



Gambar 2.4

2.2.3 MixColumns

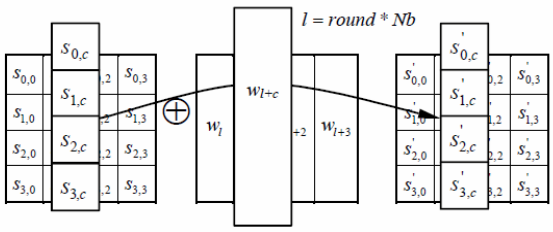
Pada tahap MixColumns ini, 4 byte pada masing-masing kolom di *state matrix* dikalikan dengan array polimom 4*4 yang dibangkitkan secara acak. Dilakukan operasi XOR pada setiap kolom di *state matrix* dengan array polinom tersebut. Hasil dari operasi itu dientrikan ke AddRoundKey



Gambar 2.5

2.2.4 AddRoundKey

Sebuah *round key* dibangkitkan dengan melakukan operasi XOR dengan setiap byte pada *state matrix*. Pada setiap round, *round key* yang baru dibangkitkan kembali dengan tahap ini.



Gambar 2.6

2.3.2 Kekuatan dari Algortima Rijndael

Sebuah cipher key memiliki panjang 128 byte, dengan panjang 128 byte terdapat 2^{128} sekitar 3,4 x 10^{38} kemungkinan kunci atau yang tepat. Jika sebuah komputer tercepat .dapat mencoba 10^6 kemungkinan kunci, maka komputer tersebut membutuhkan 5,4 x 10^{24} tahun. Dalam

hal ini, algoritma *bruteforce* tidak dapat menembusnya.

III. ANALISIS

3.1. Implementasi Algoritma Rijndael pada Aplikasi Enkripsi SMS

3.1.1 Pseudocode

Makalah ini akan mengaji bagaimana implementasi algoritma Rijndael dalam platform Android. Pada Android, bahasa pemrograman yang digunakan adalah *java*. Namun, tidak tertutup kemungkinan algoritma ini diterapkan pada bahasa pemrograman lain. Berikut ini adalah beberapa notasi algoritmik untuk mengimplementasikan langkah-langkah pada cipher :

3.1.1.1 SubBytes

```
procedure SubByte (input/output state : array of byte)
{IS : array of byte terdefinisi}
{FS : elemen dari array of byte telah berganti sesuat pada step SubByte}

  Kamus Lokal
   row,col : integer

  Algoritma
  i traversal [0..3]
     j traversal [0..3]
        row ← GetFirstFourBits(state[i][j])
        col  ← GetSecondFourBits(state[i][j])
        state[i][j] = sBox[row][column]
```

3.1.1.2 ShiftRows

```
procedure ShiftRows (input/output state : array of byte)
{IS : array of byte terdefinisi}
{FS : elemen dari array of byte telah berganti sesuat pada step ShiftRows}

   Kamus Lokal
   i,j : integer
   offset : integer
   Algoritma
   i traversal [0..3]
      j traversal [0..3]
         if  (j-1<0) then
             offset ← j-1+4
         else
             offset = j-i
         state [i][j-offset] ← state[i][j]
```

3.1.1.3 MixColumns

```
procedure MixColumns (input/output state : array of byte)
{IS : array of byte terdefinisi}
{FS : elemen dari array of byte telah berganti sesuat pada step MixColumns}
     Kamus Lokal
     Mp : array of integer
     Mtemp : array of byte
     Mcol : array of byte
     GeneratePolynomToMatrix(Mp)
     Algoritma
      i traversal [0..3]
         j traversal :[0..3]
            Mtemp ← 0
            Mcol ← GetColumnState(state)
            k traversal [0..3]
                 Mtemp[i][j]←Mtemp[i][j]+(Mcol[i][k]*
                Mp[k][j])
```

3.1.1.4 AddRoundKey

```
procedure AddRoundKey (input/output state : array of byte)
{IS : array of byte terdefinisi}
{FS : elemen dari array of byte telah berganti sesuat pada step AddRoundKey}
    Kamus Lokal
    i,j : integer

    Algoritma
    i traversal [0..3]
       j traversal [0..3]
          state [i][j] ← state[i][j] XOR roundkey[i][j]
```

3.1.1.5 Program Utama

Berikut ini official pseudo-code yang dipublikasikan NIST tentang bagaimana algoritma ini bekerja pada program utama :

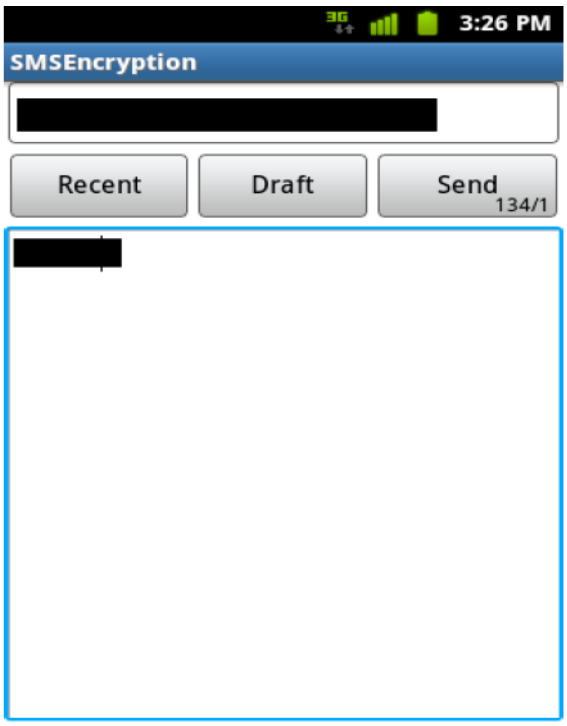
```
//Nb adalah blocksize
//Nr adalah banyaknya round

Cipher(byte in[4*Nb], byte out[4*Nb], word
w[Nb*(Nr+1)])
begin
          byte state[4,Nb]
          state = in
          AddRoundKey(state, w[0, Nb-1])
          for round = 1 step 1 to Nr–1
                    SubBytes(state)
                    ShiftRows(state)
```

```
                MixColumns(state)
                AddRoundKey(state, w[round*Nb,
                (round+1)*Nb-1])
        end for
        SubBytes(state)
        ShiftRows(state)
        AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-
        1])
        out = state
end
```

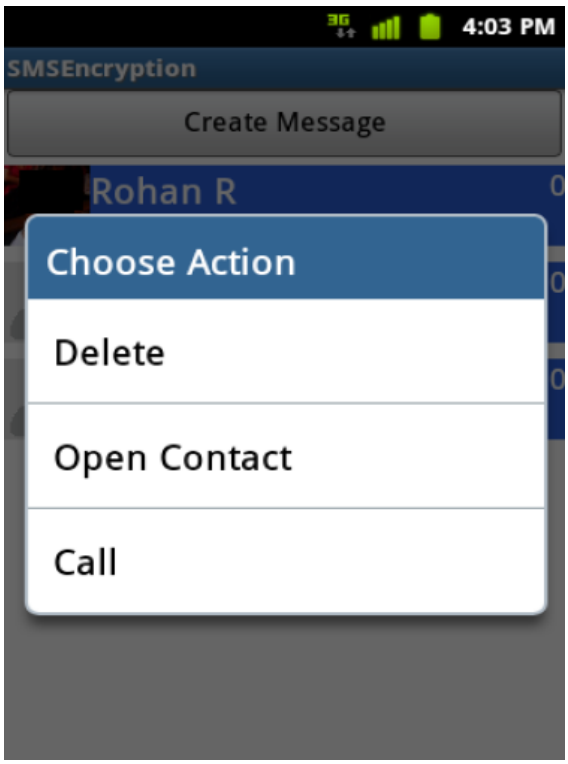
3.2 Contoh Aplikasi Pengirim SMS Terenkripsi pada Platform Android

3 orang dari University of Mumbai bernama Rohan Rayarikar, Sanket Upadhay, dan Priyanka Pimpale telah mengimplementasikannya ke dalam suatu aplikasi Android. Berikut beberapa screenshot aplikasi enkripsi SMS ini :



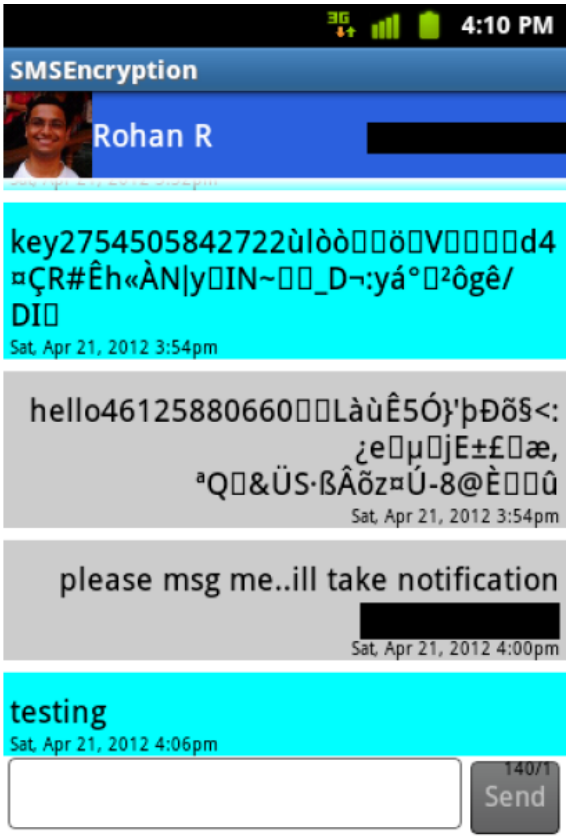
Gambar 3.1

Pada Gambar diatas, terdapat antarmuka berupa *textbox* untuk menentrikan penerima SMS. Kemudian terdapat 3 buah *button* yaitu *button recent, button draft* dan *button send*. *Button resent* berfungsi untuk melihat kontak yang digunakan beberapa saat yang lalu. *Button draft* berfungsi untuk menyimpan SMS. Kemudian *button send* berfungsi untuk mengirimkan pesan dengan meninputkan *cipher key*. Dibawah itu merupakan *textbox* untuk mengetikkan pesan yang akan dikirimkan.



Gambar 3.2

Gambar diatas merupakan fitur options yang akan muncul jika kita menyentuh dan menahan sebuah kotak.



Gambar 3.3

Pesan yang masuk dan terkirim ditampilkan dalam bentuk *thread*. Untuk memudahkan, pada pesan

pertama dan kedua yang tampak di layar merupakan hasil enkripsi dari pesan ketiga dan keempat secara berurutan.

IV. KESIMPULAN

Algoritma Rijndael telah terbukti dapat diimplementasikan untuk pembuatan aplikasi Android pengirim SMS terenkripsi. Dengan semakin maraknya aplikasi *instant messaging*, berkirim pesan pendek melalui SMS tetap lebih disukai oleh pengguna telepon selular karena generalitasnya yang dapat mencakup berbagai platform telepon selular. Kekuatan enkripsi dari algoritma ini merupakan salah satu keunggulan menggunakan algoritma ini untuk berkirim pesan secara lebih aman. Dengan kemajuan teknologi dan keterbukaan informasi seperti sekarang, aplikasi ini sangat memungkinkan untuk diimplementasikan ke platform lain selain Android. Aplikasi pengirim SMS terenkripsi hanyalah salah satu dari pemanfaatan algoritma Rijndael. Dengan kehandalannya dalam mengenkripsi pesan, algoritma ini dapat dimanfaatkan pada pengamanan data digital untuk kepentingan yang lebih luas.

REFERENSI

[1] Liem, Inggriani. 2007. Diktat Kuliah Dasar Pemrograman. Bandung

[2] Munir, Rinaldi. 2004. Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[3] http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf diakses 14 Desember 2013

[4] http://research.ijcaonline.org/volume50/number19/pxc3881038.pdf diakses 14 Desember 2013

[5] http://www.phonescoop.com/glossary/term.php?gid=131 diakses 14 Desember 2013

[6] http://www.computerworld.com/s/article/9178684/Developing_for_the_iPhone_and_Android_The_pros_and_cons diakses 14 Desember 2013

[7] http://www.eetimes.com/document.asp?doc_id=1275908 diakses 14 Desember 2013

[8] http://cacr.uwaterloo.ca/hac/about/chap7.pdf diakses 14 Desember 2013

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, 17 Desember 2013

Tegar Aji Pangestu
13512061