Penggunaan AES pada Enkripsi Data Perangkat *Mobile* dengan Sistem Operasi Android

Erwin - 13511065
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511065@std.stei.itb.ac.id

***Abstraksi—Advanced Encryption Standard* (AES) merupakan salah satu algoritma yang kuat untuk digunakan dalam kriptografi. AES adalah salah satu dari beberapa algoritma kriptografi yang menggunakan kunci simetri. AES berbasis Rijndael cipher yang dikembangkan oleh 2 kriptografer Belgian, Joan Daemen dan Vincent Rijmen. AES telah diadopsi oleh pemerintah Amerika dan digunakan secara meluas. Pada bulan Mei 2002, AES menjadi standar yang digunakan untuk enkripsi informasi yang bersifat *top-secret*.**

AES saat ini banyak digunakan untuk enkripsi data-data yang bersifat rahasia, terutama pada perangkat elektronik seperti komputer. Pengaplikasian algoritma AES pada perangkat mobile masih kurang. Dengan meluasnya penggunaan perangkat mobile dan pertukaran informasi antar perangkat mobile, maka diperlukan algoritma enkripsi yang kuat seperti AES untuk melindungi informasi yang bersifat rahasia.

Oleh karena itu, pada paper ini akan dibahas implementasi algoritma Advanced Encryption Standard (AES) dalam perangkat mobile yang menggunakan sistem operasi Android.

Keywords—AES, state array, block, key

I. PENDAHULUAN

Advanced Encryption Standard (AES) awalnya adalah sebuah kompetisi yang diadakan oleh NIST untuk memilih algoritma block cipher yang paling tepat untuk digunakan sebagai standar. Rijndael, sebuah block cipher yang dikembangkan oleh Vincent Rijmen dan Joan Daemen memenangkan kompetisi ini dan algoritma yang mereka kembangkan digunakan dalam AES. AES pada awalnya dikembangkan untuk menggantikan algoritma Data Encryption Standard (DES) yang telah ditemukan kelemahannya dan dapat dijebol dalam beberapa jam saja dengan menggunakan komputer biasa. Kekuatan algoritma AES jauh melebihi DES, jika mesin yang digunakan untuk menjebol DES digunakan untuk menjebol AES, maka diperlukan waktu jutaan tahun hingga AES dapat terpecahkan.

AES yang menggunakan Rijndael dapat menerima pengaturan berupa panjang blok dan panjang kunci. Saat ini, AES menggunakan panjang blok 128 bit sedangkan panjang kunci dapat dibagi menjadi tiga variasi, yaitu 128,192, atau 256 bit. Makin panjang bit yang digunakan, hasil enkripsi yang dihasilkan oleh AES akan menjadi semakin kuat. AES menggunakan prinsip jaringan subsitusi dan permutasi. Teknis algoritma AES akan dijelaskan pada bab berikut.

Paper ini akan membahas teknik implementasi AES dalam perangkat mobile berbasis Android dengan panjang blok / kunci 128 bit untuk menghemat sumber daya dan waktu karena keterbatasan kemampuan memproses pada perangkat mobile.

II. ALGORITMA AES

Algoritma enkripsi AES terdiri dari 10 ronde dari pemrosesan kunci 128 bit. Setiap ronde diawali dengan subsitusi, kemudian permutasi, pencampuran kolom, dan diakhiri dengan penambahan kunci. Untuk mempermudah pemahaman, pemrosesan 128 bit AES dapat dimodelkan ke dalam matriks 4x4 yang terdiri dari 16 byte.

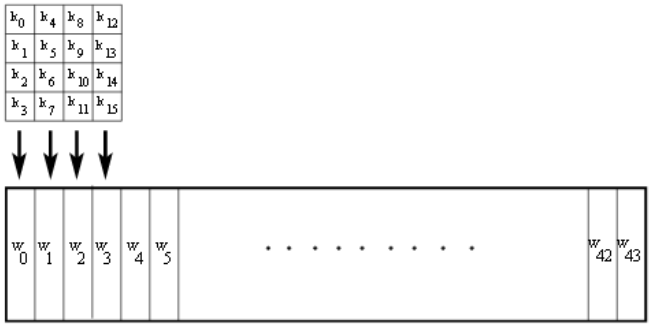
$$\begin{bmatrix} byte_0 & byte_4 & byte_8 & byte_{12} \\ byte_1 & byte_5 & byte_9 & byte_{13} \\ byte_2 & byte_6 & byte_{10} & byte_{14} \\ byte_3 & byte_7 & byte_{11} & byte_{15} \end{bmatrix}$$

Gambar 1State Array [1]

Dengan demikian, 4 byte pertama dari blok input mengisi kolom pertama dari matriks tersebut. 4 byte berikutnya mengisi kolom berikutnya dan seterusnya. Matriks tersebut sering disebut sebagai *state array* dalam AES. AES memiliki notasi *word*. Setiap *word* mengandung 4 byte (32 bit) sehingga setiap kolom dari *state array* merupakan *word*.

Kunci enkripsi AES dengan panjang 128 bit, diatur dengan format 4 *word* (1 kata = 32bit). 4 *word* tersebut kemudian diekspansi hingga didapatkan 44 *word*. Semua *word* ini akan digunakan dalam pemrosesan 10 ronde dari AES. Proses

penggunaan *word* ini dapat dilihat pada gambar di bawah :



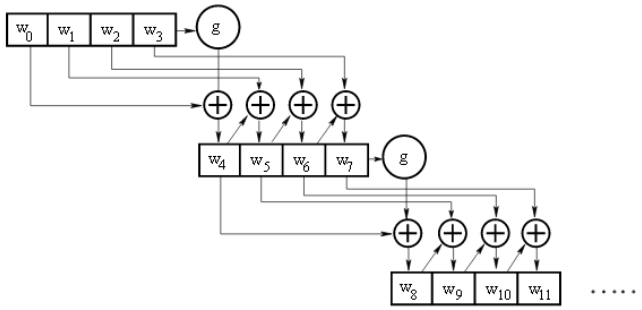
Gambar 2 Word usage [1]

Algoritma ekspansi kunci dari 4 *word* menjadi 44 *word* adalah sebagai berikut :

$$w_4 = w_0 \otimes g(w_3)$$
$$w_5 = w_4 \otimes w_1$$
$$w_6 = w_5 \otimes w_2$$
$$w_7 = w_6 \otimes w_3$$

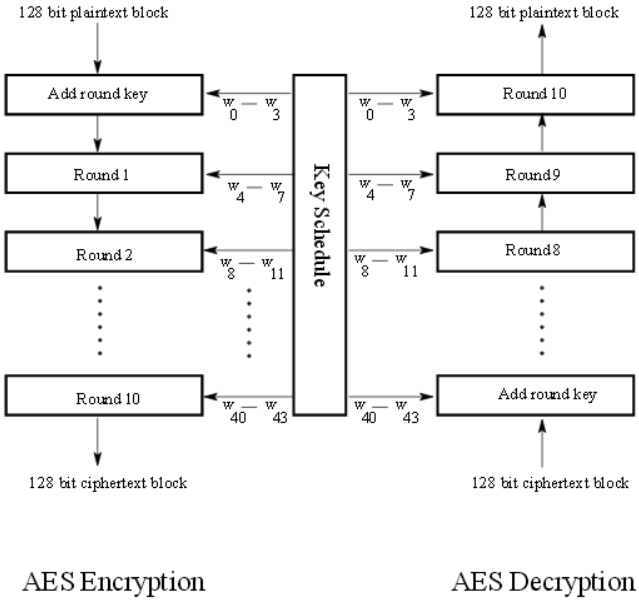
Algoritma di atas akan diulang sampai *word* ke-44. Fungsi g pada penanganan *word* pertama adalah sebuah fungsi yang terdiri dari 3 tahapan yaitu, rotasi kiri 1 byte, subsitusi byte, dan peng-XOR-an byte.

Proses *key expansion* lebih jelasnya dapat dilihat pada gambar di bawah :



Gambar 3 Key Expansion [1]

Garis besar struktur AES dapat dilihat pada gambar di bawah :



Gambar 4 AES Structure [1]

Penjelasan setiap ronde yang terjadi pada tahap enkripsi AES adalah sebagai berikut.

A. Substitution Bytes

Pada tahap ini, digunakan 16x16 *lookup table* untuk menemukan byte yang akan menggantikan byte yang ada pada *state array* masukan. Tujuan dari tahap ini adalah untuk mengurangi korelasi dari bit masukan dan bit keluaran.

Konstruksi *lookup table* untuk tahap ini adalah dengan cara mengisi setiap sel dari tabel tersebut dengan menggabungkan indeks baris dan kolom. Perhatikan gambar di bawah :

Gambar 5 Lookup Table [1]

Proses selanjutnya adalah dengan mengganti nilai dari setiap sel dengan invers multiplikatifnya sendiri dalam GF(2^8). Setelah itu dilakukan XOR untuk setiap bit dalam tabel seperti yang ditunjukkan dalam matriks di bawah :

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Gambar 6 Bit XOR [1]

Langkah selanjutnya adalah memanipulasi baris pada *state array* untuk meningkatkan keteracakan dari byte dalam *state array* tersebut.

B. Shifting Rows

Penggantian baris yang dilakukan pada tahap ini terdiri dari 4 bagian yaitu :

1. Tidak mengubah baris pertama dari *state array*
2. *Circular shifting* baris kedua 1 byte ke kiri
3. *Circular shifting* baris ketiga 2 byte ke kiri
4. *Circular shifting* baris keempat 3 byte ke kiri

Representasi dari keempat tahap di atas dapat dilihat pada gambar di bawah.

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} ===> \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{bmatrix}$$

Gambar 5 Shifting Rows [1]

Tujuan dari tahap ini adalah untuk mengacak aturan byte dari blok masukan. Oleh karena blok masukan dimasukkan per kolom, penggantian byte secara baris akan mengacak aturan byte tersebut.

C. Mixing Colums

Tahap ini bertujuan sama dengan tahap sebelumnya, yaitu untuk mengacak aturan byte dari blok masukan lebih jauh lagi. Tahap ini menggantikan setiap byte dari kolom dengan fungsi dengan masukan semua byte dari kolom yang sama. Fungsinya adalah perkalian 2 dengan byte yang sedang diproses ditambah 3 kali byte selanjutnya ditambah 2 byte setelahnya. Matriks perkalian tersebut dapat dilihat pada gambar di bawah :

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Gambar 6 Mixing Columns [1]

D. Add Round Key

Pada tahap ini, dilakukan penambahan kunci ronde kepada keluaran dari tahap sebelumnya. Kunci ronde ini akan di XOR kan dengan *state array* masukan sebelumnya sehingga dihasilkan *state array* yang baru.

III. IMPLEMENTASI

Implementasi AES pada Android akan menggunakan perangkat lunak Eclipse dengan bahasa pemrograman Java. Perancangan awal adalah dengan melakukan import pada *library* Java yang sudah tersedia untuk AES.

```
import javax.crypto.SecretKey;
import javax.crypto.SecretKeySpec;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.KeyGenerator;
import javax.crypto.IvParameterSpec;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.Cipher;
```

Langkah selanjutnya adalah melakukan pengkodean untuk enkripsi dan dekripsi. Kode implementasi adalah sebagai berikut.

```
public static byte[] encrypt(String plain, String Enkey){
        Cipher cipher = cipher.getInstance(“
         AES/CBC/NoPadding”, “SunJCE”);

        SecretKeySpec key = new SecretKeySpec(Enkey
        .getBytes(“UTF-8”),”AES”);

        cipher.init(cipher.ENCRYPT_MODE, key, new
        IvParameterSpec(IV.getBytes(“UTF-8”)));

        return cipher.doFinal(plain,getbytes(“UTF-8”));

}

public static byte[] decrypt(String cipherTx, String Enkey){
        Cipher cipher = cipher.getInstance(“
         AES/CBC/NoPadding”, “SunJCE”);

        SecretKeySpec key = new SecretKeySpec(Enkey
        .getBytes(“UTF-8”),”AES”);

        cipher.init(cipher.DECRYPT_MODE, key, new
        IvParameterSpec(IV.getBytes(“UTF-8”)));

        return cipher.doFinal(cipherTx,getbytes(“UTF-8”));

}
```

Implementasi AES dalam perangkat *mobile* dengan sistem operasi Android dapat dilihat pada tabel di bawah. Untuk data dengan ukuran 1MB diperlukan waktu enkripsi selama sekitar 50 detik. Waktu enkripsi berbanding lurus dengan ukuran data yang dienkripsi.

Berikut adalah tabel yang menunjukkan keterkaitan beberapa ukuran data yang diuji dengan waktu yang diperlukan untuk proses enkripsi :

Ukuran Data	Waktu (ms)
128 b	6,231
256 b	11,389
512 b	21,587
1 Kb	38,452
128 Kb	6251
256 Kb	11873
512 Kb	23091
1 Mb	50328
2 Mb	113871

Tabel 1 Perbandingan Ukuran Data dan Waktu

IV. ANALISIS HASIL

Dari tabel di atas terlihat bahwa untuk ukuran data yang kecil, waktu yang diperlukan untuk mengenkripsi cukup cepat dan dalam batas toleransi ($t < 1s$). Akan tetapi, untuk data yang berukuran besar (di atas 1Mb), waktu enkripsi menjadi cukup lama ($t > 50s$) yang menyebabkan pengguna akan merasa tidak nyaman / tidak sabar dalam menggunakan aplikasi ini di perangkat *mobile*.

V. KESIMPULAN

Kesimpulan yang dapat ditarik dari hasil di atas adalah penggunaan AES untuk enkripsi data berukuran kecil masih dapat diimplementasikan dalam perangkat *mobile*. Akan tetapi, jika ingin menggunakan AES untuk data yang lebih besar, algoritma AES harus dimodifikasi terlebih dahulu agar waktu yang dibutuhkan dalam proses enkripsi dapat ditoleransi oleh pengguna perangkat *mobile*.

REFERENSI

[1] Kak. Avinash, "Lecture Notes on Computer and Network Security". United States, 2015.

[2] Daemen, J., Rijmen, V. (2002) The Design of Rjindael: AES-the advanced encryption standard. Springer, Heidelberg

[3] X Zhang, KK Parhi, "Implementation approaches for the advanced encryption standard algorithm", *IEEE Circuits Syst. Mag.*, 2002

[4] C. C. Lu, S. Y. Tseng, "Integrated design of AES (advanced encryption standard) encrypter and decrypter", *Proc. IEEE Int. Conf. Application Specific Systems, Architectures Processors*, 2002

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, 11 Mei 2015

Erwin 13511065