Algoritma Enkripsi Baku Tingkat Lanjut

Anggrahita Bayu Sasmita
13507021
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
e-mail: if17021@students.if.itb.ac.id

Abstrak- *Makalah ini membahas tentang pemanfaatan metode enkripsi* block cipher *dalam suatu algoritma* Advanced Encryption Standard *(AES). Enkripsi ini merupakan algoritma kunci pribadi yang memadukan teknik permutasi-substitusi. Sebagai algoritma yang diadopsi menjadi standar enkripsi, AES memiliki kemampuan untuk mengolah panjang kunci sebesar 128 bit, dua kali lipat lebih besar dibandingkan algoritma standar predesesornya (DES) yang kemampuannya terbatas pada enkripsi 56 bit kunci.*

1. PENDAHULUAN

Dalam sejarah perkembangan komputer, terdapat suatu pernyataan yang dikenal dengan nama *Moore's Law*, yang menyatakan bahwa kemampuan komputer akan meningkat dua kali lipat dalam kurun waktu setiap 18 bulan. Dalam kehidupan sehari-hari, pernyataan tersebut memiliki nilai kebenaran yang cukup tinggi. Di satu sisi, hal ini merupakan suatu nilai positif bagi perkembangan kinerja komputer. Akan tetapi, di sisi lain, resiko terpecahkannya suatu enkripsi oleh serangan *brute force* akan menjadi semakin tinggi.

Sejak tahun 1977, algoritma *Data Encryption Standard* (DES) ditetapkan sebagai algoritma pemroses informasi baku di Amerika Serikat. Penggunaan algoritma ini telah merebak ke seluruh dunia karena kemampuannya yang dinilai sangat tinggi pada saat itu. Algoritma ini memiliki kemampuan untuk mengolah 56 bit kunci informasi sekaligus. Hal in berarti bahwa untuk 56 bit kunci tersebut akan terdapat 2^{56} kemungkinan kunci untuk dipecahkan. Jumlah tersebut dinilai sangat besar dan akan memakan waktu yang sangat lama bila hendak dipecahkan melalui metode *brute force* menggunakan komputer pada saat algoritma ini dibakukan.

Keamanan penggunaan DES sebagai algoritma enkripsi tidak lagi terjamin untuk masa-masa sekarang. Hal ini disebabkan ukuran kunci 56 bit yang dinilai ringan untuk dipecahkan dengan kemampuan □omputer saat ini. Pada bulan Januari 1999, kunci DES berhasil dipecahkan dalam waktu 22 jam 15 menit oleh suatu kolaborasi dari dua organisasi yang bergerak di bidang informatika.

Sebagai antisipasi serangan *brute force*, DES sempat dikembangkan menjadi *Triple DES.* Algoritma ini memperkuat kemampuan DES menjadi tiga kali lipat dalam hal panjang kunci serta perulangan eksekusi prosedur. Algoritma ini menawarkan keamanan yang beberapa tingkat lebih tinggi, tetapi memakan waktu yang lama untuk diaplikasikan dalam perangkat lunak.

Dari permasalahan yang telah disebutkan, dapat kita nyatakan bahwa algoritma DES memiliki kelemahan dalam tingkat keamanan serta performansi yang lambat dalam penerapan pada perangkat lunak. Oleh karena itu, dibutuhkanlah suatu algoritma baku lain sebagai pengganti algoritma standar DES yang dapat memenuhi kebutuhan tingkat keamanan serta keceptan performansi. Kemunculan AES sebagai algoritma baku dapat dikatakan mewakili pemenuhan kebutuhan-kebutuhan tersebut.

2. PERKEMBANGAN *BLOCK CIPHER*

Dalam kajian kriptografi, *block cipher* merupakan suatu metode enkripsi yang mengolah sekaligus satu blok plainteks. Sebagai contoh, misalkan terdapat sebuah masukan plainteks sebesar 128 bit dienkripsi menjadi keluaran cipherteks yang besarnya sama dan mewakili plainteks yang dienkripsi. Enkripsi diperoleh melalui suatu prosedur enkripsi standar yang dilengkapi suatu kunci. Kunci ini merupakan masukan sekunder sebagai pengendali transformasi plainteks.

Pada dasarnya *block cipher* memiliki dua pasang algoritma dengan kegunaan masing-masing untuk enkripsi dan dekripsi. Masukan baku untuk keduanya adalah kunci transformasi serta plainteks (untuk enkripsi) atau cipherteks (untuk dekripsi). Algoritma dekripsi dapat dinyatakan sebagai invers dari algoritma enkripsi dengan syarat bahwa kunci transformasi yang digunakan sama.

Prinsip kerja dari algoritma yang memanfaatkan kunci tersebut sesungguhnya adalah pemetaan bijektif (korespondensi satu-satu) terhadap seluruh anggota himpunan komponen masukan. Untuk masing-masing kunci, terdapat kemungkinan permutasi sebanyak 2^n dengan n adalah jumlah digit masukan.

Hingga pertengan 1990an, jumlah digit masukan pada umumnya sebesar 64 bit. Seiring perkembangan kemampuan komputer, desain *block cipher* saat ini telah mampu menangani 128 bit masukan. Adapun masukan yang memiliki ukuran lebih besar daripada ukuran baku akan diolah dengan mode operasi khusus. Beberapa mode operasi khusus tersebut, yaitu:

- *Electronic Codebook,* metode ini memproses masukan dan dienkripsi/dekripsi sebagai blok yang terpisah.
- *Cipher Block Chaining,* metode ini memanfaatkan vektor inisialisasi khusus yang memproses satu blok masukan sebelum dienkripsi. Vektor inisialisasi blok masukan berikutnya diambil dari blok cipherteks yang telah dienkripsi.
- *Cipher Feedback,* metode ini merupakan inversi *Cipher Block Chaining*. Posisi vektor inisialisasi ditukar dengan posisi plainteks pada algoritma enkripsi.

Masih terdapat beberapa mode-mode operasi *block cipher* lainnya yang merupakan variasi dari mode-mode di atas. Secara umum, sesungguhnya mode-mode operasi tersebut memanfaatkan vektor inisialisasi sebagai pemroses awal sebelum enkripsi dilakukan. Vektor inisialisasi hanya akan digunakan sebagai kunci pra-enkripsi untuk blok masukan awal. Untuk blok masukan berikutnya, algoritma tersebut akan memanfaatkan keluaran enkripsi sebagai pengganti vektor inisialisasi.

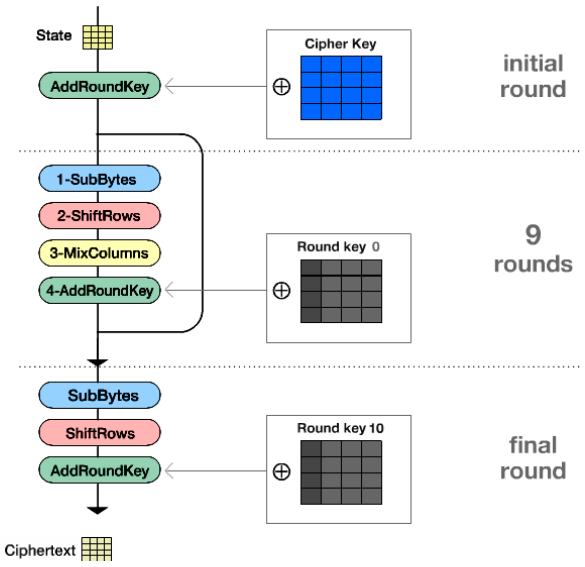
3. ALGORITMA AES

AES merupakan salah satu implementasi dari desain *block cipher*. Enkripsi AES memiliki kemampuan untuk mengolah 128 bit blok plainteks sekaligus. AES membutuhkan dua parameter masukan untuk melakukan prosedur enkripsi maupun dekripsi. Parameter masukan tersebut adalah plainteks dan *cipher key*. Baik kunci maupun plainteks merupakan suatu blok dengan besar minimal 128 bit.

Blok sebesar 128 bit tersebut diolah sebagai blok berukuran 16 byte yang disusun dalam bentuk tabel berukuran 4x4. Plainteks dan kunci kemudian akan diolah secara terpisah untuk menghasilkan cipherteks yang dienkripsi secara bertingkat.

Secara umum, prosedur enkripsi AES dapat dinyatakan dalam langkah-langkah berikut:

- *KeyExpansion*; ekspansi cipher key menjadi round key menggunakan *Rijndael's Key Schedule*.
- *Initial Round* yang berisi *AddRoundKey*; plainteks diolah dengan mengimplementasaikan *bitwise XOR* dengan komponen *round key*. Pada keadaan awal, *cipher key*-lah yang menjadi *round key*.
- *Rounds;* iterasi prosedur yang terbagi menjadi empat tahapan:
    a. *SubBytes*; substitusi masing-masing byte dengan merujuk pada suatu tabel acuan.
    b. *ShiftRows*; pemindahan kedudukan byte dalam tabel dalam beberapa langkah tertentu.
    c. *MixColumns*; operasi perkalian matriks pada Rijndael's Galois Field terhadap kolom tabel.
    d. *AddRoundKey*; operasi *bitwise XOR* terhadap tabel dengan *round key* yang merupakan ekspansi *cipher key*.
- *Final Round* yang serupa dengan operasi *Rounds*, namun tanpa prosedur *MixColumn*.



Gambar 1: Diagram kedudukan algoritma AES dengan iterasi 9 kali tahapan *Rounds*

3.1 Prosedur Key Expansion

Ekspansi *cipher key* digunakan untuk membentuk *round key* yang akan digunakan pada langkah-langkah enkripsi plainteks (*state*). Ekspansi kunci ini memiliki tahapan khusus yang dikenal sebagai *Rijndael's key schedule*.

Secara umum, *Rijndael's key schedule* memanfaatkan beberapa operasi:

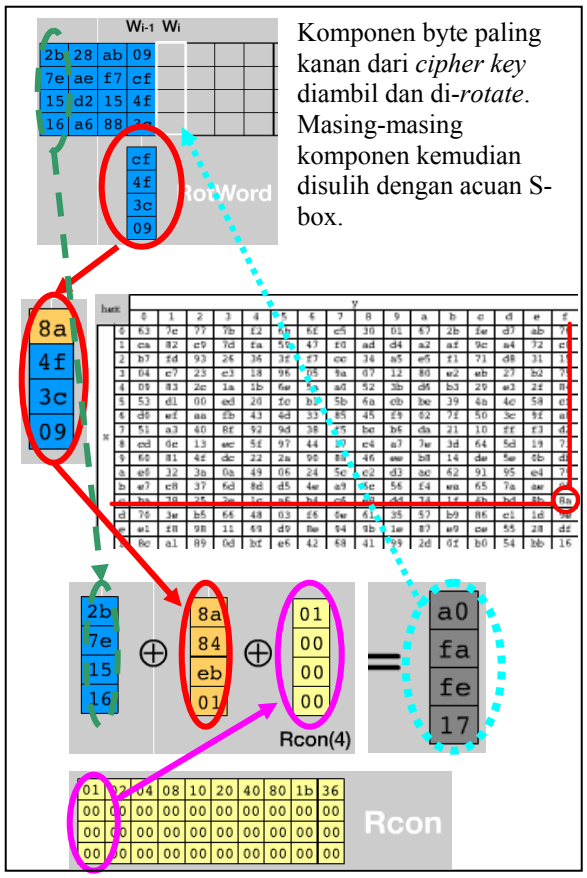
- *Rotate*; operasi ini merupakan operasi pemindahan byte. Byte terujung dipindahkan ke ujung lainnya tanpa mengubah keterurutan komponen lain.
- *Rcon*; operasi ini merupakan eksponensiasi dari bilangan bulat 2 dalam semesta *Rijndael's finite field*.

Finite field tidak akan dibahas secara mendalam pada uraian makalah ini karena kajian mengenai hal tersebut tercakup dalam lingkup aljabar abstrak.

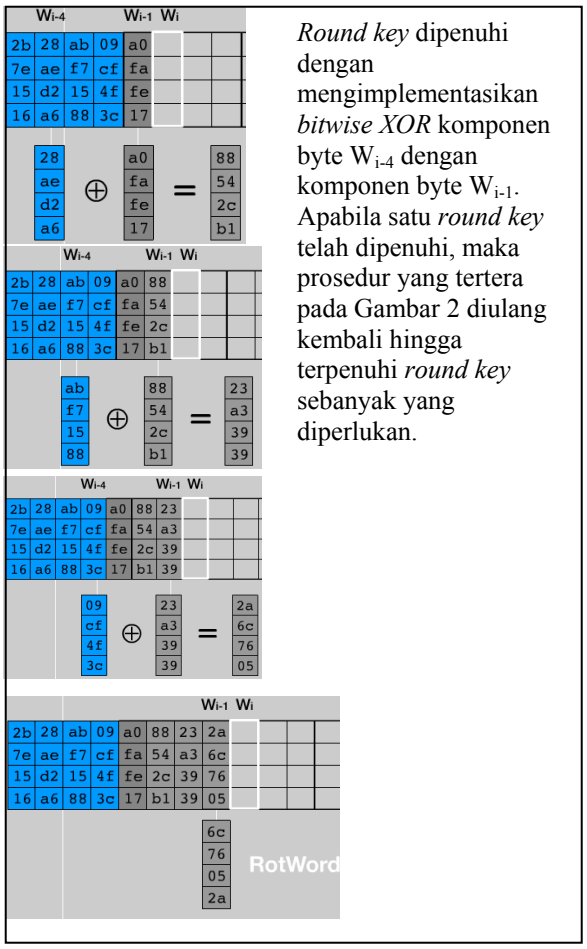
- *S-box*; Tabel ini bukan merupakan operasi dalam arti literal. *Substitution box* merupakan tabel yang menyulih nilai byte dengan nilai yang telah ditentukan.

Umumnya, iterasi tahapan *rounds* pada AES diulang sebanyak sembilan kali. Oleh karenanya, terdapat sebelas *round key* yang dibutuhkan dalam satu kali enkripsi *state*. *Round key* tersebut adalah sebuah *cipher key* pada *initial round*, sembilan lainnya pada tahapan *rounds,* serta satu *round key* yang digunakan pada *final round*.

Operasi-operasi di atas kemudian diimplementasikan pada *key schedule* dan diulang sebanyak kebutuhan. Gambar 2 dan 3 di bawah ini menjelaskan secara detil prosedur *key schedule* tersebut.



Gambar 2: Ilustrasi prosedur *Rijndael's key schedule*



Gambar 3: Lanjutan prosedur key schedule

Melalui *key schedule* tersebut, *cipher key* yang menjadi parameter masukan akan diekspansi menjadi beberapa *round key*. *Round key* ini kemudian akan diimplementasikan pada tahapan *AddRoundKey* yang merupakan enkripsi dari masukan *state* (plainteks).

3.2 Tahapan *Initial Round*

Tahapan *initial round* pada algoritma enkripsi AES sesungguhnya hanyalah operasi *bitwise XOR* terhadap komponen byte plainteks dengan acuan *cipher key*. Masing-masing komponen byte diubah ke dalam bentuk biner delapan bit untuk kemudian dioperasikan masing-masing bitnya dengan fungsi logika *exclusive or*. Bilangan biner 8-bit yang terbentuk kemudian dikonversi lagi menjadi komponen byte yang mewakili.

3.3 Iterasi Tahapan *Rounds*

Bagian utama dari algoritma enkripsi AES terdapat pada perulangan tahapan *rounds*. Tahapan ini memiliki empat operasi unik yang diulang sebanyak sembilan kali (untuk

masukan 128 bit), sebelas kali (untuk masukan 192 bit) atau tiga belas kali (untuk masukan 256 bit) sehingga
menghasilkan enkripsi bertingkat pada plainteks awal. Sebagaimana telah dijelaskan di awal, keempat operasi tersebut ialah *SubBytes, ShiftRows, MixColumns,* dan *AddRoundKey*.

3.3.1 Operasi SubBytes

Operasi *SubBytes* merupakan operasi penyulihan komponen byte plainteks dengan menggunakan acuan *Rijndael's S-Box*. S-Box ini disusun dengan menentukan invers perkalian angka-angka tertentu yang terdapat pada *Rijndael's finite field* yang kemudian ditransformasi menggunakan metode *affine transformation.*

Hasil akhir dari penyusunan S-Box tersebut dapat dilihat pada Gambar 4.

hex	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

x

Gambar 4: *Rijndael's S-Box*

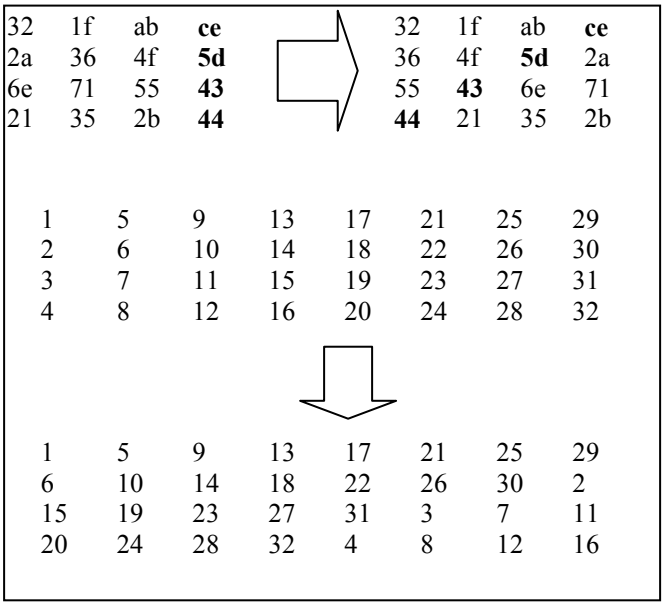
Masing-masing komponen byte pada *state* yang melalui tahapan ini disulih nilainya dengan nilai yang tertera pada S-Box. Nilai *state* yang berubah kemudian akan diproses lagi melalui tahapan selanjutnya.

3.3.2 Operasi ShiftRows

Operasi *ShiftRows* pada dasarnya bukanlah operasi yang mengimplementasikan persamaan matematika yang rumit. Operasi ini hanya menggeser komponen bit sebesar nilai tertentu.

Untuk panjang masukan sebesar 128 bit, *state* dinyatakan sebagai tabel berukuran 4x4. Baris kedua tabel tersebut kemudian digeser sebanyak satu sel ke kiri, baris ketiga sebanyak dua sel ke kiri dan baris keempat sebanyak tiga sel ke kiri.

Pergeseran ini tidak berbeda untuk *state* dengan ukuran 192 bit (yang diwakili dengan tabel berukuran 4x6). Penanganan khusus diberlakukan untuk *state* yang berukuran 256 bit. Dalam hal ini, pergeseran baris kedua tetap dilakukan sebanyak satu sel ke arah kiri. Akan tetapi, baris ketiga digeser sebanyak tiga sel ke arah kiri dan baris keempat digeser sebanyak empat sel dengan arah yang sama.



Gambar 5: Kedudukan *state* setelah melewati *ShiftRows*

3.3.3 Operasi MixColumns

Operasi *MixColumns* merupakan operasi yang dilakukan setelah pergeseran komponen byte pada *state*. Operasi ini merupakan perkalian matriks yang dilakukan dalam *finite field* terhadap masing-masing kolom pada *state*. Prosedur ini menggunakan matriks khusus sebagai pengali, yaitu matriks yang terdiri atas komponen berikut:

2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2

Dalam tahapan ini, masing-masing kolom dianggap sebagai polinom dalam *finite field*. Kolom tersebut kemudian dikalikan dengan matriks yang tertera sebelumnya. Apabila hasil perkalian melebihi 8-bit, maka terdapat suatu prosedur eliminasi bit dengan mengimplementasikan XOR pada hasil dengan suatu *string* 9-bit. Umumnya *string* tersebut adalah suatu bilangan biner dengan nilai 100011011.

3.3.4 Operasi AddRoundKey

Operasi ini merupakan operasi terakhir dalam satu tahapan *round*. *State* yang berada pada operasi ini diolah dengan operasi logika *bitwise XOR* terhadap *round key* yang telah disebutkan. *Round key* yang digunakan akan berbeda-beda untuk setiap *round.*

3.4 Tahapan *Final State*

Tahapan ini dapat dikatakan *round* tanpa *MixColumn*. Operasi-operasi seperti *SubBytes, ShiftRows,* dan *AddRoundKey* dilakukan sebagaimana pada tahapan *round*. *State* yang menjadi keluaran tahapan ini sudah merupakan cipherteks.

4.ANALISIS

Beberapa faktor yang menjadikan algoritma AES sebagai algoritma enkripsi baku adalah karena factor keamanannya. Kemampuannya membentuk permutasi terhadap kunci sepanjang 128 bit menjadikan cipherteks keluaran menjadi sangant sukar dipecahkan. Terdapat 2^{128} kemungkinan kunci yang bisa ditarik apabila cipherteks hasil enkripsi AES dipecahkan dengan serangan *brute force*. Untuk komputer saat ini, jumlah tersebut adalah jumlah yang sangat besar.

Salah satu keunggulan AES dibandingkan dengan Triple DES adalah kecepatan performansinya. Meskipun keduanya merupakan algoritma yang mengantisipasi kelemahan DES, pada dasarnya Triple DES adalah DES yang kemampuannya diperkuat tiga kali lipat dalam hal eksekusi prosedur serta jumlah *string* masukan. Hal ini sesungguhnya tidak menambah kerumitan enkripsi.

Beberapa hal lain yang menjadikan AES lebih aman adalah karena pergeseran dan penyulihan komponen yang dilakukan sebelum dioperasikan dengan kunci baik *round key,* maupun *cipher key*.

5.KESIMPULAN

Algoritma AES merupakan *block cipher* bersifat *symmetric* yang memanfaatkan operasi *substitution-permutation*. Kunci-kunci yang diolah dalam operasi algoritma tersebut memanfaatkan beberapa teori *finite field* yang banyak menggunakan operasi *bitwise XOR* dalam pengoperasiannya. Sebagai pengganti DES, AES memiliki tingkat keamanan yang lebih tinggi dan performansi yang lebih baik.

DAFTAR REFERENSI

[1] http://en.wikipedia.org/wiki/AES
waktu akses: 4 Januari 2009
[2] http:// en.wikipedia.org/wiki/Block_Cipher
waktu akses: 4 Januari 2009
[3] http:// en.wikipedia.org/wiki/DES
waktu akses: 4 Januari 2009
[4] http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf
waktu akses: 4 Januari 2009
[5] http://www.quadibloc.com/crypto/co040401.htm
waktu akses: 4 Januari 2009