ANALISIS ALGORITME DAN WAKTU ENKRIPSI VERSUS DEKRIPSI PADA *ADVANCED ENCRYPTION STANDARD (*AES)

Sugi Guritman [1], Ahmad Ridha [2], Endang Purnama Giri [3]

ABSTRAK

AES merupakan algoritme kriptografi yang didesain untuk beroperasi pada blok pesan 128 bit dan menggunakan tiga variasi blok kunci dengan panjang 128 bit, 192 bit, atau 256 bit. Empat proses utama algoritme terdiri atas satu proses permutasi (ShiftRows) dan tiga proses subtitusi (SubBytes, MixColumns, dan AddRoundKey). Analisis teori menyimpulkan, proses enkripsi AES didesain untuk melakukan proses penyandian secara rahasia dengan tingkat keamanan tak linear dengan kompleksitas waktu seefisien mungkin melalui penggunaan proses-proses transformasi yang ringan dalam implementasi. Akan tetapi, invers (proses kebalikan) dari proses-proses tersebut memiliki efisiensi yang rendah, akibatnya proses dekripsi AES menjadi lambat. Dengan analisis algoritme, didapatkan AES memiliki kompleksitas pada lingkup O(n) baik bagi proses enkripsi maupun dekripsi. Dari analisis keamanan AES dikategorikan memiliki level ketangguhan keamanan yang cukup memadai. Dari analisis hasil uji perbandingan kecepatan dapat disimpulkan bahwa AES memiliki kinerja waktu yang tinggi. Sedangkan melalui analisis uji implementasi enkripsi versus dekripsi menggunakan MatLab versi 6.5 dapat disimpulkan bahwa dari segi efisiensi, proses enkripsi tidak sama dengan proses dekripsi, dengan efisiensi dekripsi relatif lebih rendah. Dari hasil uji statistik (independent-samples T-test). dengan selang kepercayaan α=95% antara proses enkripsi dan dekripsi berbeda nyata dengan nilai signifikan 0,01. Hasil ini menunjukan bahwa AES bukan merupakan algoritme dengan struktur jaringan feistel sesuai dengan yang dikemukakan oleh Stallings (2003).

Kata kunci: AES, kriptografi, *block cipher*, enkripsi, dekripsi.

1. PENDAHULUAN

Perkembangan teknologi di bidang komputer berkaitan dengan semakin maraknya penggunaan sistem komputer dengan segala yang berkaitan terhadap konektifitas melalui jaringan sebagai media komunikasi bagi data dan informasi. Fenomena tersebut mengakibatkan gangguan dan usaha pencurian informasi terhadap aliran data ataupun pesan semakin meningkat sehingga para kriptografer dituntut untuk mendesain saluran komunikasi data seaman mungkin dari proses pencurian data oleh pihak pengganggu.

Secara umum, teknik kriptografi dapat digunakan untuk melakukan penyandian pesan dan autentikasi pesan. *Advanced Encryption Standard* (AES*)* merupakan teknik atau algoritme kriptografi penyandian pesan yang menggunakan teknik blok simetris. Algoritme ini dikembangkan oleh dua kriptografer dari Belgia, yaitu Dr. Joan Daemen dan Dr. Vincent Rijmen pada tahun 1997. Algoritme ini kemudian diajukan sebagai proposal Rijndael bagi AES, dan pada November 2001 disahkan sebagai proposal terpilih bagi AES oleh *National Institute of Standard and Technology* (NIST) (Stallings 2003). Pada penelitian ini dilakukan analisis teori, analisis algoritme, dan analisis uji perbandingan kinerja dari algoritme AES yang meliputi aspek keamanan dan kecepatan melalui telaah pustaka, serta implementasi algoritme AES dengan modus operasi *Electronic Codebook* (ECB) bagi analisis uji *running time* enkripsi *versus* dekripsi dalam MatLab versi 6.5.

2. DESKRIPSI ALGORITME AES

AES merupakan algoritme kriptografi yang didesain untuk beroperasi pada blok pesan 128 bit dan menggunakan tiga variasi blok kunci dengan panjang 128 bit, 192 bit, atau 256 bit. Khusus untuk penelitian ini, pengkajian akan dibatasi pada blok pesan 128 bit dengan ukuran blok kunci 128 bit. Empat proses utama algoritme terdiri atas satu proses permutasi (ShiftRows*)* dan tiga proses subtitusi (SubBytes, MixColumns, dan AddRoundKey*)*. Struktur algoritme secara umum cukup sederhana, dengan proses baik enkripsi maupun dekripsi diawali proses AddRoundKey, diikuti sembilan *round* yang masing-masing tersusun atas empat proses, dan diakhiri *round* kesepuluh yang terdiri atas tiga proses. Proses AddRoundKey membentuk *Vernam cipher*, sedangkan tiga proses lainnya menciptakan proses pengacakan dan penggabungan secara tak linear (Stallings 2003).

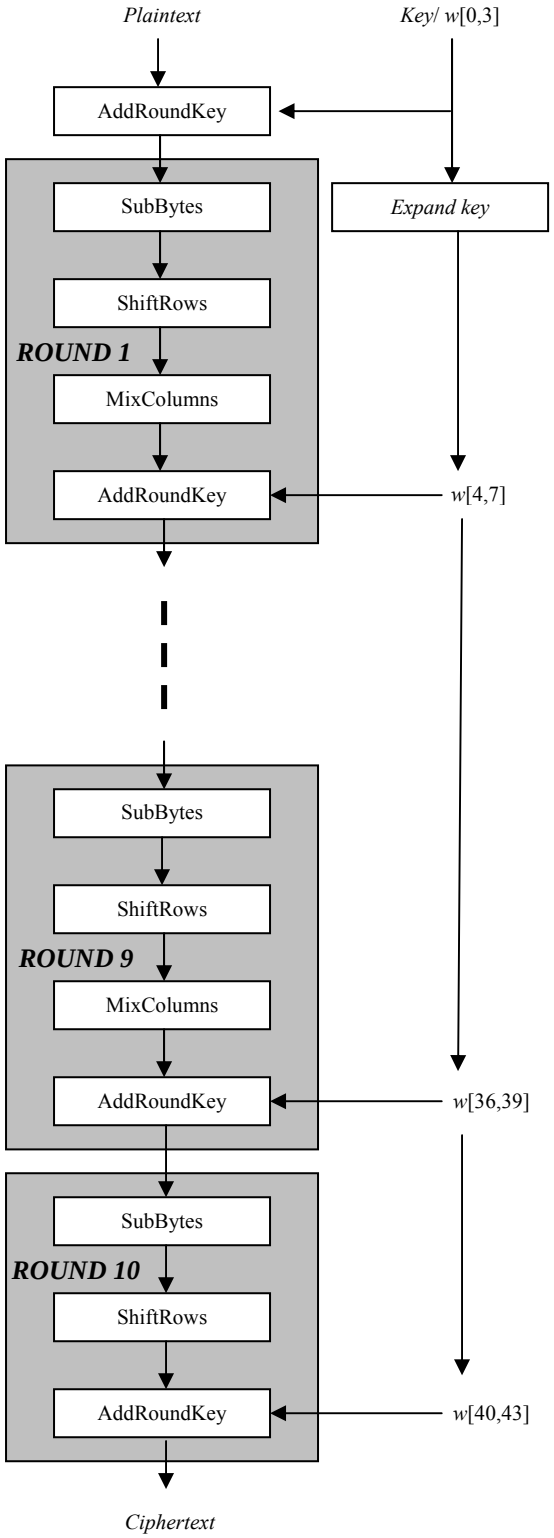
Proses Enkripsi

Struktur proses enkripsi AES digambarkan pada Gambar 1. Proses enkripsi AES diawali proses AddRoundKey diikuti sembilan *round* dengan arsitektur yang tersusun atas empat proses dan urutan identik yaitu SubBytes, ShiftRows, MixColumns, dan AddRoundKey. Akhir proses enkripsi digunakan *round* kesepuluh yang tersusun atas tiga proses terurut SubBytes, ShiftRows, dan AddRoundKey yang keseluruhan proses tersebut diiringi proses penjadwalan key (oleh *expand key*) bagi setiap *round*. Seluruh fungsi operasi (penjumlahan dan perkalian) yang tercakup dalam AES merupakan operasi-operasi yang didefinisikan dalam ruang lingkup *finite field* GF(2^8) dengan polinomial *irreducible* pembangkit $m(x)= x^8 + x^4 + x^3 + x + 1$.

Sebelum memasuki proses enkripsi, dilakukan partisi setiap blok *plaintext* 128 bit menjadi masing-masing delapan bit dan di representasikan dalam dua karakter heksadesimal, maka dihasilkan 16 buah *input*. *Input* tersebut direpresentasikan ke dalam matriks 4x4 yang disebut matriks *input.* Gambar 2 menunjukkan matriks *input* tersebut.

$$\begin{matrix} in_0 & in_4 & in_8 & in_{12} \\ in_1 & in_5 & in_9 & in_{13} \\ in_2 & in_6 & in_{10} & in_{14} \\ in_3 & in_7 & in_{11} & in_{15} \end{matrix}$$

Gambar 2. Matriks *input* (Stallings 2003).



Gambar 1. Proses enkripsi AES, *key* 128 bit (Stallings 2003).

Begitu pun pada blok kunci dilakukan partisi blok 128 bit dari kunci menjadi 16 nilai masing-masing dua karakter heksadesimal dan direpresentasikan ke dalam matriks 4x4, disebut matriks *key* sebagai berikut (Gambar 3):

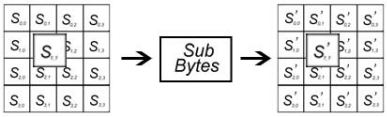
$$\begin{matrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{matrix}$$

Gambar 3. Matriks *key* (Stallings 2003).

Kolom pertama, kedua, ketiga dan, keempat dari matriks ini merepresentasikan *word* 0 (w_0), *word* 1 (w_1), *word* 2 (w_2), dan *word* 3 (w_3) yang masing-masing berukuran 4 byte atau 32 bit. Tahapan proses dimulai dengan proses meng-XOR-kan antara matriks *input* dengan matriks *key* yang hasilnya adalah matriks *state.* Proses XOR ini disebut juga proses AddRoundKey.

a. SubBytes

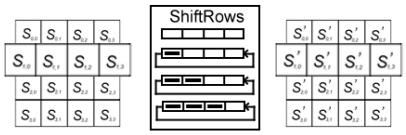
Forward subtitute byte transformation atau SubBytes merupakan proses subtitusi sederhana dari 16 nilai matriks 4x4 kepada 16 nilai baru mengikuti kaidah permutasi *S-Box*. Ilustrasi proses SubBytes diberikan pada Gambar 4.



Gambar 4. Proses SubBytes (Stallings 2003).

b. ShiftRows

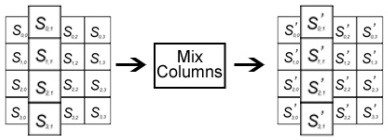
Forward shift rows atau ShiftRows merupakan proses permutasi sederhana dari 16 nilai matriks 4x4 kepada 16 nilai baru matriks 4x4. Ilustrasi proses ShiftRows diberikan pada Gambar 5.



Gambar 5. Proses ShiftRows (Stallings 2003).

c. MixColumns

Forward mix column atau MixColumns merupakan operasi terhadap setiap kolom secara terpisah (Gambar 6). Transformasi tersebut didefinisikan dengan proses mengalikan matriks *forward mix column transformation* yang berukuran 4x4 dengan matriks *state* berukuran 4x4.



Gambar 6. Proses MixColumns (Stallings 2003).

Adapun matriks *forward mix column transformation* (Polimat) dapat dilihat pada Gambar 7 berikut:

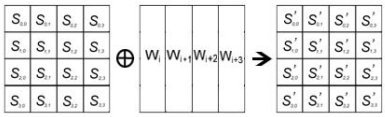
$$\begin{matrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{matrix}$$

Gambar 7. Matriks Polimat (Stallings 2003).

Setiap operasi penjumlahan dan perkalian dalam operasi perkalian matriks merupakan operasi yang terdefinisikan dalam ruang lingkup *finite field* GF(2^8).

d. AddRoundKey

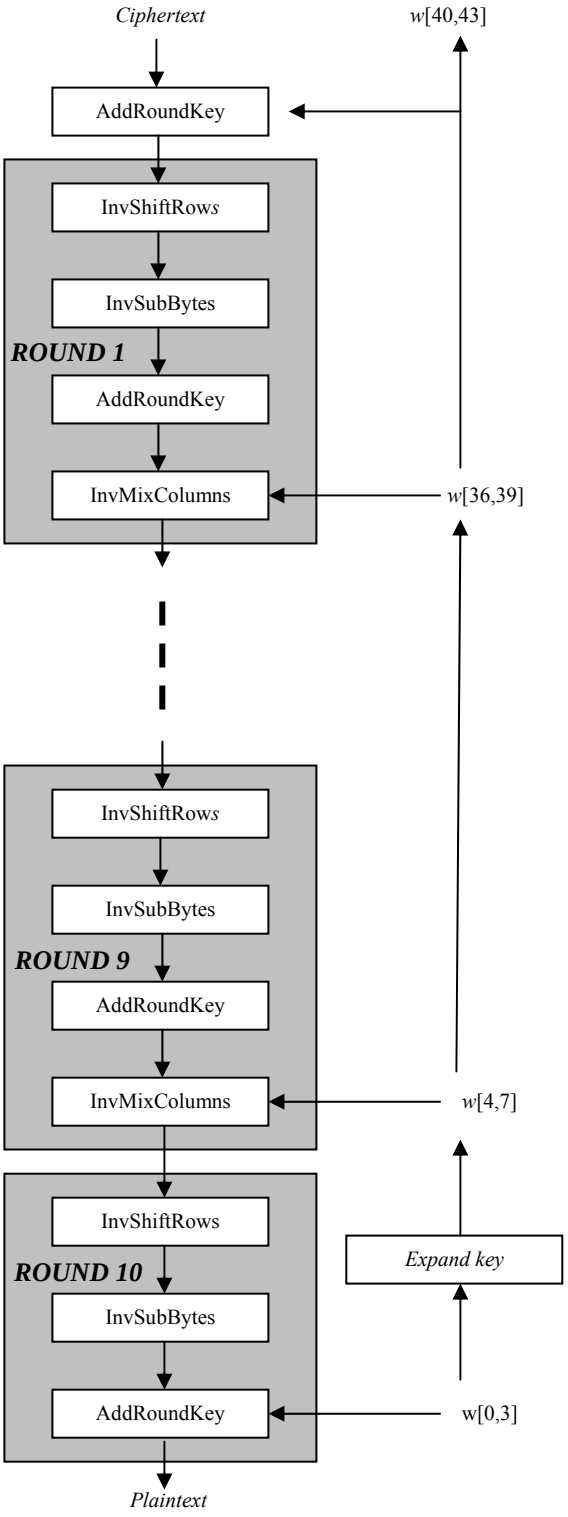
Forward add round key atau AddRoundKey hanya merupakan proses XOR sederhana antara matriks *state* 4x4 dengan matriks 4x4 yang merupakan *key* dari *round* bersesuaian hasil ekspansi kunci. Ilustrasi dari proses AddRoundKey diberikan pada Gambar 8.



Gambar 8. Proses AddRoundKey (Stallings 2003).

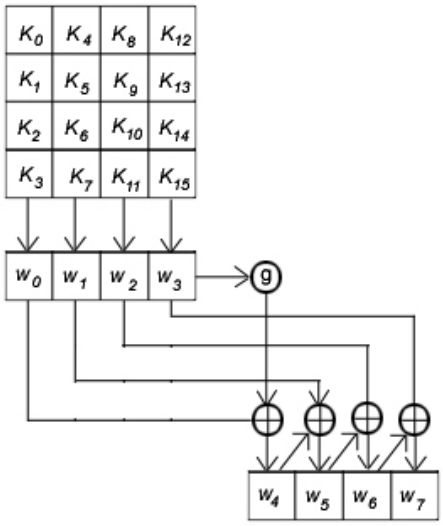
Proses Dekripsi

Seperti yang telah disebutkan, struktur proses dekripsi secara umum sama dengan proses enkripsi, akan tetapi proses dekripsi AES memiliki urutan proses transformasi penyusun tiap *round* yang berbeda. Selain itu, transformasi yang digunakan pun merupakan transformasi kebalikan dari proses transformasi penyusun setiap *round* pada proses enkripsi. Meskipun proses *expand key* pada proses dekripsi dan enkripsi identik, akan tetapi penjadwalan penggunaan *key* bagi setiap *round* pada dekripsi berkebalikan dengan proses enkripsi. Penjadwalan *key* pada proses dekripsi dimulai dari *word* ke-43 menuju *word* ke-0. Sehingga secara skematis proses dekripsi dapat digambarkan seperti Gambar 9.



Gambar 9. Proses dekripsi AES, *key* 128 bit (Stallings 2003).

***Expand Key* (Ekspansi Kunci)**



Gambar 10. Ilustrasi proses ekspansi kunci (Stallings 2003)

Ekspand key atau ekspansi kunci adalah proses untuk membangkitkan kunci bagi *round* pada setiap iterasi. Ilustrasi ekspansi kunci diberikan pada Gambar 10. Karena setiap *key* merupakan representasi dari empat *word* maka proses ini dapat juga dikatakan sebagai proses ekspansi *word*. Pada kasus ini dibangkitkan 44 buah *word*, yaitu *word* ke-0 (w_0) hingga *word* ke-43 (w_{43}). Empat *word* pertama (w_0-w_3) dihasilkan dari penyalinan langsung *key* asli, sedangkan 40 *word* selanjutnya dibangkitkan dengan menggunakan $w_{(i-1)}$ sebagai *input word* serta urutan langkah transformasi sebagai berikut:

1. Proses *rot word* yaitu transformasi *input word* (b_0, b_1, b_2, b_3) menjadi (b_1, b_2, b_3, b_0).
2. Proses *sub word* yaitu proses subtitusi hasil dari *rot word* menggunakan kaidah *S-box* (SubBytes).
3. Proses XOR hasil *sub word* dengan suatu nilai konstan *Rcon*[j] bersesuaian bagi setiap *round*.

   $$Rcon[j] = (RC[j], 0, 0, 0)$$

   dan

   $$RC[j] = 2 * RC[j-1]$$
   $$RC[1] = 01$$

   Dengan sifat operasi * merupakan perkalian yang terdefinisikan pada *field* GF(2^8).
4. Lakukan proses XOR hasil langkah 3 dengan $w_{(i-4)}$.

3. HASIL DAN PEMBAHASAN

Analisis Teori

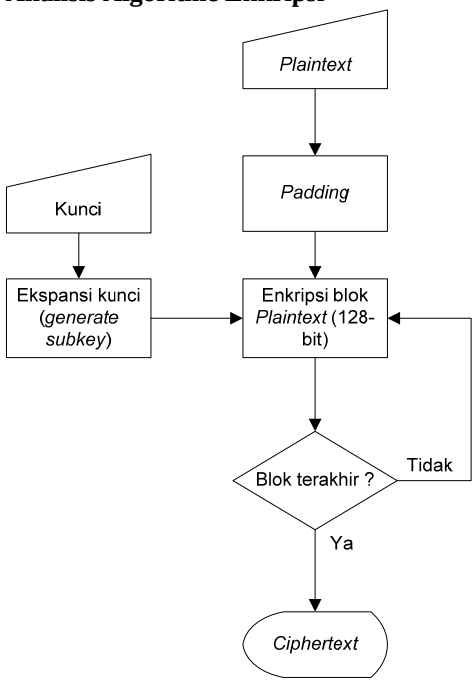
AES merupakan algoritme kriptografi yang didesain untuk beroperasi pada blok pesan 128 bit dan menggunakan tiga variasi blok kunci dengan panjang 128 bit, 192 bit, atau 256 bit. Variasi ukuran kunci mengakibatkan pemilihan penggunaan kunci sulit untuk diprediksi oleh kriptanalis. Ukuran kunci minimum yang mencapai 128 bit mengakibatkan *range* minimum bagi ruang kunci sebesar 2^{128}, dibandingkan dengan DES yang menggunakan 56 bit kunci, ruang kunci sebesar 2^{56} mengindikasikan AES lebih tahan terhadap serangan *brute-force search.*

Empat proses utama algoritme terdiri atas satu proses permutasi (ShiftRows) dan tiga proses subtitusi (SubBytes, MixColumns, dan AddRoundKey).

Pada SubBytes,. Untuk desain *S-box* digunakan kaidah pemetaan dalam lingkup *finite field* GF(2^8) yang memiliki kemungkinan 30 polinomial *irreducible* sebagai pembangkit. Sebagai pengembangan lanjut desain *S-box* dapat memiliki 30 bentuk desain. Subtitusi merupakan proses *confusion* yang menggunakan sumber daya kecil, sehingga implementasi algoritme menjadi efisien. Proses kedua, transformasi ShiftRows, menggerakan byte-byte secara bebas dari suatu kolom dengan kolom lain dalam satu baris. Proses ini menghasilkan perubahan (*distance*) bit secara signifikan. Pada transformasi MixColumns, pemilihan koefisien-koefisien matriks Polimat yang memiliki nilai nyata kecil yaitu {01}, {02}, atau {03}selain atas pertimbangan efisiensi implementasi juga didasari pada pembentukan transformasi yang menghasilkan jarak (perubahan) maksimal membentuk proses pengacakan byte-byte dalam satu kolom. Menurut Stallings (2003) proses perkalian matriks dengan menggunakan nilai-nilai koefisien tersebut merupakan kontruksi operasi yang paling banyak menggunakan operasi pergeseran dan operasi XOR. Operasi pergeseran dan operasi XOR merupakan operasi dengan sumber daya ringan pada implementasi. Kombinasi transformasi MixColumns dan ShiftRows setelah iterasi beberapa *round* menjamin setiap bit-bit *output* secara keseluruhan tergantung pada nilai dari bit-bit *input*. Adapun kelemahan dari proses ini adalah bentuk invers transformasinya (InvMixColumns) yang cenderung berat dalam implementasi. Proses yang keempat yaitu AddRoundKey melakukan efek transformasi bagi setiap bit-bit secara menyeluruh dengan operasi ringan XOR, meningkatkan keamanan secara tak linear dengan kompleksitas yang efisien. Proses invers dari proses ini adalah proses ini sendiri (XOR) maka masalah peningkatan kompleksitas pada proses dekripsi dapat diabaikan.

Analisis Algoritme

a. Analisis Algoritme Enkripsi



Gambar 11. Diagram alir proses enkripsi AES dengan menggunakan modus ECB.

Algoritme enkripsi AES diimplementasikan dengan menggunakan modus *Electronic Codebook* (ECB). Diagram alir proses enkripsi AES dengan ECB dapat dilihat pada Gambar 11.

Langkah untuk melakukan proses enkripsi modus ECB pada AES adalah:

a. *Padding plaintext.*
b. Ekspansi kunci.
c. Enkripsi blok *plaintext* 128 bit.
d. *Output.*

Waktu eksekusi pada langkah a, b, dan d adalah konstan (misalkan α) karena tidak dipengaruhi ukuran *input*. Sedangkan langkah c (misalkan waktu eksekusinya = ε) dipengaruhi jumlah blok *input plaintext* 128 bit, sehingga untuk *input* berukuran *n* blok diperlukan waktu ε*n*.

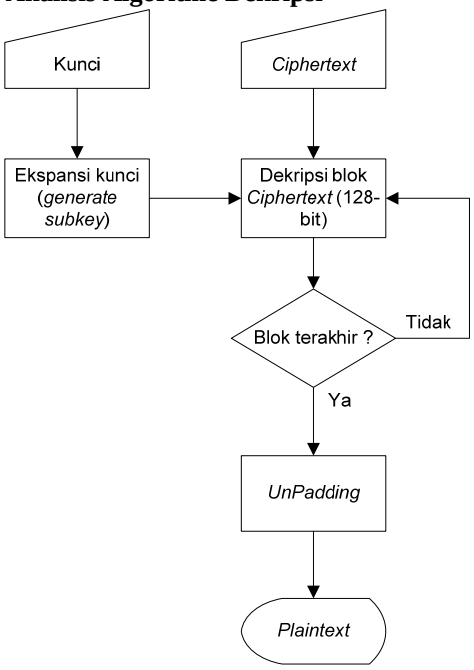
Secara keseluruhan, waktu eksekusi enkripsi AES dengan modus ECB adalah ε*n* + α, dengan ε dan α adalah suatu konstanta dan *n* adalah jumlah

blok *input*. Jadi notasi *O* untuk kasus terburuk proses enkripsi AES dengan modus ECB adalah:

$$E_{(AES)} = \varepsilon n + \alpha$$

Didapatkan kompleksitas $E_{(AES)}$ adalah dalam lingkup $O(n)$.

b. Analisis Algoritme Dekripsi



Gambar 12. Diagram alir proses dekripsi AES dengan menggunakan modus ECB.

Seperti pada algoritme enkripsi, algoritme dekripsi AES juga diimplementasikan dengan menggunakan modus ECB. Diagram alir proses dekripsi AES dengan ECB dapat dilihat pada Gambar 12.

Langkah untuk melakukan proses dekripsi modus ECB pada AES adalah:

a. Ekspansi kunci
b. Dekripsi blok *ciphertext* 128 bit
c. *Unpadding*
d. *Output*

Waktu eksekusi pada langkah a, c, dan d adalah konstan (misalkan β) karena tidak dipengaruhi ukuran *input.* Sedangkan langkah b (misalkan waktu eksekusinya = δ) dipengaruhi jumlah blok *input ciphertext* 128 bit, sehingga untuk *input* berukuran *n* blok diperlukan waktu δ*n*.

Secara keseluruhan, waktu eksekusi dekripsi AES dengan modus ECB adalah δ*n* + β, dengan δ dan β adalah suatu konstanta dan *n* adalah jumlah blok *input*. Jadi notasi *O* untuk kasus terburuk proses dekripsi AES dengan modus ECB adalah:

$$D_{(AES)} = \delta n + \beta$$

Didapatkan kompleksitas $D_{(AES)}$ adalah dalam lingkup $O(n)$.

Analisis Keamanan

Tabel 1. Data uji AES dengan *reduced round variants* (Nechvatal *et al.* 2000)

Algorithm, Round	***Rounds (Key Size)***	***Type of Attack***	***Texts***	***Mem. Bytes***	***Ops.***
MARS 16 *Core* (C) 16 *Mixing* (M)	11C	*Amp. Boomerang*	2^{65}	2^{70}	2^{229}
	16M, 5C	*Meet-in-Midle*	8	2^{236}	2^{232}
	16M, 5C	*Diff. M-i-M*	2^{50}	2^{197}	2^{247}
	6M, 6C	*Amp. Boomerang*	2^{69}	2^{73}	2^{197}
RC6 20	14	*Stat. Disting*	2^{118}	2^{112}	2^{122}
	12	*Stat. Disting*	2^{94}	2^{42}	2^{119}
	14 (192,256)	*Stat. Disting*	2^{110}	2^{42}	2^{135}
	14 (192,256)	*Stat. Disting*	2^{108}	2^{74}	2^{160}
	15 (256)	*Stat. Disting*	2^{119}	2^{138}	2^{215}
AES (Rijndael) 10 (128) 12 (192) 14 (256)	4	*Truncated Diff.*	2^{9}	small	2^{9}
	5	*Truncated Diff.*	2^{11}	small	2^{40}
	6	*Truncated Diff.*	2^{32}	$7*2^{32}$	2^{72}
	6	*Truncated Diff.*	$6*2^{32}$	$7*2^{32}$	2^{44}
	7 (192)	*Truncated Diff.*	$19*2^{32}$	$7*2^{32}$	2^{155}
	7 (256)	*Truncated Diff.*	$21*2^{32}$	$7*2^{32}$	2^{172}
	7	*Truncated Diff.*	$2^{128}-2^{119}$	2^{61}	2^{120}
	8 (256)	*Truncated Diff.*	$2^{128}-2^{119}$	2^{101}	2^{204}
	9 (256)	*Related Key*	2^{77}	NA	2^{224}
	7 (192)	*Truncated Diff.*	2^{32}	$7*2^{32}$	2^{184}
	7 (256)	*Truncated Diff.*	2^{32}	$7*2^{32}$	2^{200}
	7 (192,256)	*Truncated Diff.*	2^{32}	$7*2^{32}$	2^{140}
Serpent 32	8 (192,256)	*Amp. Boomerang*	2^{113}	2^{119}	2^{179}
	6 (256)	*Meet-in-Middle*	512	2^{246}	2^{247}
	6	*Differential*	2^{83}	2^{40}	2^{90}
	6	*Differential*	2^{71}	2^{75}	2^{103}
	6 (192,256)	*Differential*	2^{41}	2^{45}	2^{163}
	7 (256)	*Differential*	2^{122}	2^{126}	2^{248}
	8 (192,256)	*Boomerang*	2^{128}	2^{133}	2^{163}
	8 (192,256)	*Amp. Boomerang*	2^{110}	2^{115}	2^{175}
	9 (256)	*Amp. Boomerang*	2^{110}	2^{212}	2^{252}
Twofish 16	6 (256)	*Impossible Diff.*	NA	NA	2^{256}
	6	*Related Key*	NA	NA	NA

Salah satu parameter yang dapat diukur untuk menentukan level ketangguhan aspek keamanan algoritme kriptografi adalah perbandingan antara derajat *round* keseluruhan algoritme kriptografi tersebut dibandingkan dengan jumlah *round* yang memiliki kemungkinan terkena serangan (Nechvatal *et al.* 2000). Melalui skenario uji serangan menggunakan *reduced round variants* didapatkan data seperti pada Tabel 1.

Kolom "*Round* (*Key size*)" mengindikasikan jumlah *round* yang dapat terserang untuk kasus uji dengan ukuran kunci (*key*) yang bersesuaian. Kolom "*Texts*" mengindikasikan informasi yang dibutuhkan untuk memberikan efek serangan, secara spesifik, ukuran dari blok *plaintext* terhadap blok *ciphertext* yang bersesuaian dengan menggunakan suatu kunci rahasia. Kolom "*Mem. Bytes*" mengindikasikan ukuran terbesar dari byte memori yang digunakan saat mengeksekusi serangan. Sedangkan kolom "*Ops.*" mengindikasikan perkiraan dari ukuran jumlah operasi yang diperlukan untuk melakukan serangan. NA menyatakan informasi data yang tidak dapat disajikan.

Berdasarkan data tersebut didapatkan bahwa hasil uji perlakuan bagi algoritme AES (Rijndael) adalah sebagai berikut:

Untuk kunci 128 bit, 6 sampai 7 dari 10 *round* total dapat diserang (rasio=0,6-0,7). Untuk kunci 192 bit, 7 *round* dari 12 *round* total dapat diserang (rasio=0,583). Sedangkan untuk kunci 256 bit dengan 14 *round* total, 7 (rasio=0,5), 8 (rasio=0,571), hingga 9 (rasio=0,643) *round* dapat diserang. Adapun sumber daya yang dibutuhkan untuk melakukan serangan terhadap AES meliputi operasi dengan nilai *cost* yang besar. Level ketangguhan keamanan Rijndael dalam kategori cukup memadai, rasio terburuk 0,7.

Analisis Kecepatan

Berdasarkan hasil implementasi uji atas berbagai *platform* menggunakan blok kunci 128 bit yang dilakukan oleh divisi keamanan komputer (*Computer Security Division*) NIST didapatkan hasil yang dapat dilihat pada Tabel 2, Tabel 3, dan Tabel 4.

Tabel 2. Tingkat kinerja waktu enkripsi dan dekripsi berbagai *platform* (Nechvatal *et al.* 2000)

	MARS	RC6	AES (Rijndael)	Serpent	Two fish
32 bit (C)	II	I	II	III	II
32 bit (Java)	II	I	II	III	III
64 bit (C and Assembler	II	II	I	III	I
8 bit (C and Assembler	II	II	I	III	II
32 bit smartcard	II	I	I	III	III
Digital Signal Processor	II	II	I	III	I

Tabel 3. Tingkat kinerja waktu penjadwalan *key* berbagai *platform* (Nechvatal *et al.* 2000)

	MARS	RC6	AES (Rijndael)	Serpent	Two fish
32 bit (C)	II	II	I	III	III
32 bit (Java)	II	II	I	II	III
64 bit (C and Assembler	III	II	I	II	III
8 bit (C and Assembler	II	III	I	III	II
Digital Signal Processor	II	II	I	I	III

Tabel 4. Tingkat kinerja waktu keseluruhan (Nechvatal *et al.* 2000)

	MARS	RC6	AES (Rijndael)	Serpent	Two fish
Enc/ Dec	II	I	I	III	II
Key Setup	II	II	I	II	III

Pada tabel-tabel tersebut I menyatakan level kinerja waktu tinggi, II menyatakan level kinerja waktu rata-rata, dan III digunakan untuk menyatakan level kinerja waktu rendah. Dari data didapatkan algoritme kriptografi AES memiliki kinerja waktu tinggi yang konsisten untuk seluruh proses meskipun akan mengalami penurunan kinerja bagi blok kunci 192 bit dan 256 bit.

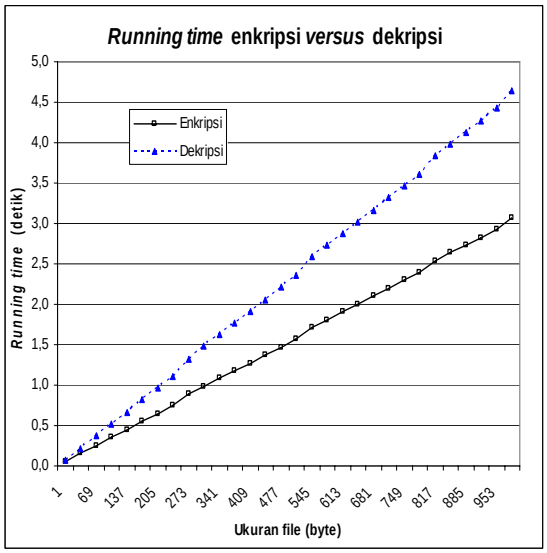
Analisis Uji Implementasi

Dari penelitian menggunakan 30 ukuran *file* teks berbeda dalam selang 1 byte sampai dengan 1000 byte sebagai objek kajian, dilanjutkan dengan pengukuran *running time* dari setiap perlakuan (ulangan setiap perlakuan sebanyak 10 kali) didapatkan rekapitulasi uji hasil implementasi proses enkripsi dan dekripsi yang tercantum pada Tabel 5.

Tabel 5. Rekapitulasi *running time* enkripsi dekripsi

No	Ukuran *file* (byte)	Ukuran setelah *padding* (byte)	Enkripsi (detik)	Dekripsi (detik)
1	1	16	0,061	0,073
2	35	48	0,159	0,221
3	69	80	0,256	0,368
4	103	112	0,355	0,515
5	137	144	0,452	0,662
6	171	176	0,551	0,815
7	205	208	0,647	0,956
8	239	240	0,744	1,105
9	273	288	0,888	1,326
10	307	320	0,984	1,475
11	341	352	1,084	1,619
12	375	384	1,177	1,768
13	409	416	1,276	1,914
14	443	448	1,375	2,054
15	477	480	1,473	2,210
16	511	512	1,571	2,358
17	545	560	1,714	2,581
18	579	592	1,811	2,727
19	613	624	1,909	2,874
20	647	656	2,006	3,024
21	681	688	2,103	3,169
22	715	720	2,201	3,319
23	749	752	2,295	3,466
24	783	784	2,395	3,613
25	817	832	2,540	3,833
26	851	864	2,639	3,980
27	885	896	2,736	4,127
28	919	928	2,825	4,276
29	953	960	2,929	4,425
30	1000	1008	3,075	4,644

Berdasarkan Tabel 5, dengan meningkatnya ukuran *file*, *running time* eksekusi proses uji baik enkripsi dan dekripsi mengalami peningkatan. Hal ini terjadi karena semakin besar panjang blok pesan maka ukuran iterasi proses yang dilakukan-pun akan semakin besar. Untuk menggambarkan fenomena ini, diberikan grafik hubungan antara proses enkripsi-dekripsi terhadap ukuran *file* pada Gambar 13.



Gambar 13. Hubungan *running time* enkripsi-dekripsi terhadap ukuran *file*.

Dari grafik hubungan *running time* enkripsi-dekripsi terhadap ukuran *file* terlihat ukuran *running time* proses dekripsi selalu lebih besar dari *running time* proses enkripsi dengan *margin* yang semakin besar seiring membesarnya ukuran *file*. Untuk mengetahui besar perbedaan antara proses enkripsi dan dekripsi dilakukan analisis lanjut mengunakan uji statistik (*independent-samples T-test*). Asumsi analisis menyatakan bahwa antara proses enkripsi dan dekripsi sebagai dua proses berbeda yang hanya ditentukan oleh satu buah variabel bebas, yaitu ukuran *file* yang dieksekusi. Berdasarkan uji statistik didapatkan data seperti pada Tabel 6 dan Tabel 7.

Dari hasil uji statistik tersebut didapatkan nilai *Sig.*< 0,05 maka dapat disimpulkan bahwa dengan selang kepercayaan α=95%, antara proses enkripsi dan dekripsi berbeda nyata dengan nilai signifikan 0,01 serta nilai derajat bebas sebesar 50,17 dari 60 data contoh seluruhnya, 30 data enkripsi dan 30 data dekripsi. Representasi perbedaan sangat nyata antara enkripsi dan dekripsi ini digambarkan juga dengan nilai perbedaan rataan yang mencapai 0,7756. Nilai tersebut dapat dikatakan besar karena nilai rataan proses enkripsi dan dekripsi tercakup dalam suatu

nilai nyata yang kecil yaitu 1,5410 dan 2,3165. Melalui analisis regresi pada masing-masing data proses enkripsi dan dekripsi didapatkan hasil (Tabel 8 dan Tabel 9):

Tabel 6. *Group Statistics*

	GROUP	**N**	***Mean***	***Std. Error Mean***
DATA	enkripsi	30	1,5410	0,16610
	dekripsi	30	2,3165	0,25218

Tabel 7. *Independent-samples T-test* (α=95%)

Levene's Test for Equality of Variances	
F	7,158
Level of significant	
Sig.	0,01
t-test for Equality of Means	
t	-2,568
Degree of freedom	
df	50,17
***Sig.* (2-*tailed*)**	
Sig. (2-*tailed*)	0,13
Others	
Mean Difference	-0,7756
Std. Error Difference	0,30197
95% *Confidence Interval of the Difference*	
Lower	-1,38203
Upper	-0,16908

Tabel 8. Koefisien bagi grafik hubungan *running time* enkripsi terhadap ukuran *file*

	Unstandardized Coefficients	***Std. Error***	**t**	***Sig.***
(*Constant*)	4,200E-02	0,005	8,903	0,000
Ukuran *file*	3,032E-03	0,000	370,012	0,000

Tabel 9. Koefisien grafik hubungan *running time* dekripsi terhadap ukuran *file*

	Unstandardized Coefficients	***Std. Error***	**t**	***Sig.***
(*Constant*)	4,077E-02	0,008	5,399	0,000
Ukuran *file*	4,603E-03	0,000	350,942	0,000

Secara umum persamaan garis grafik hubungan *running time* enkripsi terhadap ukuran *file* dapat dituliskan sebagai E(x)= 3,032E-03(x) + 4,200E-02. Sedangkan persamaan garis grafik hubungan *running time* dekripsi terhadap ukuran *file* dapat dituliskan sebagai D(x)= 4,603E-03(x) + 4,077E-02, dengan x merupakan peubah bebas yang dalam hal ini adalah ukuran *file*. Dari dua persamaan garis tersebut maka untuk *file* uji berukuran besar dapat diasumsikan nilai *running time* dekripsi adalah 1,518 kali nilai *running time* enkripsi.

Analisis statistik lanjut pada setiap proses penyusun enkripsi-dekripsi didapatkan nilai data rataan (*mean*) bagi setiap proses seperti pada Tabel 10. Nilai rataan terbesar bagi proses enkripsi adalah MixColumns. Hal ini menyatakan bahwa kontribusi *running time* terbesar bagi proses enkripsi adalah MixColumns dengan persentase rata-rata kontribusi *running time* mencapai 95,283%. Sedangkan kontribusi *running time* terbesar bagi proses dekripsi adalah InvMixColumns (merupakan proses kebalikan MixColumns) yang mencapai persentase rata-rata kontribusi *running time* sebesar 87,906%.

Tabel 10. Statistika deskriptif setiap proses penyusun enkripsi dan dekripsi

	N	***Min***	***Max***	***Mean***	***Std.Dev***
Enkripsi					
SubBytes	30	0,002	0,019	0,01086	0,005006
ShiftRows	30	0,002	0,117	0,05626	0,032625
MixColumns	30	0,054	2,933	1,46828	0,873355
AddRoundKey	30	0,000	0,000	0,00000	0,000000
Dekripsi					
InvSubBytes	30	0,008	0,403	0,18668	0,114060
InvShiftRows	30	0,001	0,128	0,06765	0,039836
InvMixColumns	30	0,060	4,066	2,03635	1,217522
AddRoundKey	30	0,001	0,043	0,02045	0,012541

Dari statistika deskriptif setiap proses penyusun enkripsi dan dekripsi dapat juga disimpulkan bahwa penyebab menurunnya efisiensi proses dekripsi AES terhadap proses enkripsi AES diakibatkan menurunnya nilai efisiensi seluruh proses penyusun dekripsi secara umum, terlihat dari peningkatan bagi setiap nilai *mean* empat proses kebalikan (*inverse*) penyusun proses dekripsi AES terhadap nilai *mean* empat proses penyusun enkripsi AES. Pada penelitian ini didapatkan kecepatan maksimal bagi proses enkripsi mencapai 326 byte/detik, sedangkan bagi proses dekripsi hanya mencapai kurang lebih 217 byte/detik.

5. KESIMPULAN DAN SARAN

Kesimpulan

Proses enkripsi AES didesain untuk melakukan proses penyandian secara rahasia dengan tingkat keamanan tak linear dengan kompleksitas waktu seefisien mungkin melalui penggunaan proses-proses transformasi yang ringan dalam implementasi. Akan tetapi, invers (proses kebalikan) dari proses-proses tersebut terkadang memiliki efisiensi yang rendah,

akibatnya proses dekripsi AES menjadi lambat dalam implementasi.

Dengan analisis algoritme, didapatkan AES memiliki kompleksitas pada lingkup *O*(n). Hal ini berlaku bagi proses enkripsi dan dekripsi.

Dari analisis keamanan diperoleh level ketangguhan keamanan dilihat dari sisi rasio terburuk, antara jumlah *round* yang memiliki kemungkinan terserang terhadap derajat *round* keseluruhan, AES dikategorikan memiliki level ketangguhan keamanan yang cukup memadai.

Dari hasil analisis uji perbandingan kecepatan dapat disimpulkan bahwa AES memiliki kinerja waktu yang paling tinggi dibandingkan seluruh algoritme lain yang berlaku sebagai algoritme pembanding. Ini dapat dijadikan satu acuan AES dapat diandalkan bagi teknik penyandian pesan untuk masa mendatang menggantikan DES dan variannya.

Melalui analisis uji implementasi enkripsi *versus* dekripsi menggunakan MatLab versi 6.5 dapat disimpulkan bahwa dari segi efisiensi, proses enkripsi tidak sama dengan proses dekripsi, dengan proses dekripsi memiliki efisiensi yang relatif lebih rendah. Dari hasil uji statistik (*independent-samples T test*) dengan selang kepercayaan α=95% proses enkripsi dan dekripsi berbeda nyata dengan nilai signifikan 0,01. Hasil ini menunjukkan bahwa AES bukan merupakan algoritme dengan struktur jaringan Feistel sesuai dengan yang dikemukakan Stallings (2003).

Saran

Ruang lingkup penelitian ini dibatasi pada algoritme AES untuk panjang blok pesan 128 bit dengan panjang blok kunci 128 bit. Penelitian dapat dikembangkan dengan analisis lanjut menggunakan dua alternatif ukuran blok kunci yang lain yaitu blok kunci 192 bit dan 256 bit .

Dari segi algoritme perkalian modulo yang digunakan masih menggunakan algoritme perkalian modulo sederhana. Penelitian lebih lanjut dapat menggunakan algoritme perkalian modulo yang lebih efisien seperti *euclid algorithm.*

Modus operasi yang digunakan pada implementasi hanya modus operasi *Electronic Codebook* (ECB). Uji implementasi enkripsi *versus* dekripsi untuk penelitian selanjutnya dapat digunakan modus operasi yang lain seperti *Cipher Block Chaining* (CBC)*, Cipher Feedback* (CFB)*, Output Feedback* (OFB) dan *Counter* (CTR).

Dari segi bahasa pemrograman yang digunakan, jika bahasa pemrograman yang digunakan pada penelitian ini adalah MatLab maka pada penelitian selanjutnya dapat digunakan jenis bahasa pemrograman yang lain sebagai pembanding.

REFERENSI

Cormen T.H., Leiserson C.E. & Rivest R.L. 1990. Introduction to Algorithms. Massachusetts-London: The MIT Press.

Daemen J. & Rijmen V. 1999. AES Proposal: Rijndael.http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf. [6 Desember 2003].

Ireland D. 2004. Using Padding in Encryption. Sydney-Australia: DI Management Services. http://www.di-mgt.com.au/cryptopad.html#exampleecb. [12 Agustus 2004].

Menezes A., Van Oorschot P., & Vanstone S. 1996. Handbook of Applied Cryptography. CRC Press Inc. www.cacr.math.uwaterloo.ca/hac. [19 Februari 2004].

Nechvatal J. *et al*. 2000. Report on the Development of the Advanced Encryption Standard (AES). Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Administration U.S. Department of Commerce. http://csrc.nist.gov/rng/. [6 Desember 2003].

Schneier B. 1996. Applied Cryptography - Protocols, Algorithms and Source Code in C. Second Edition. New York: John Wiley & Sons, Inc.

Stallings W. 2003. Cryptography and Network Security Principles and Practice. Third Edition. New Jersey: Pearson Education.