STUDI DAN PERBANDINGAN ALGORITMA RIJNDAEL DENGAN ALGORITMA SERPENT

Rizky Delfianto – NIM 13507032

Program Studi Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha 10 Bandung Indonesia 40132
e-mail: if17032@students.if.itb.ac.id

ABSTRAK

Kriptografi merupakan ilmu yang populer digunakan pada proses penyampaian pesan secara rahasia. Ilmu ini telah digunakan dari jaman Romawi dan berkembang hingga saat ini. Dan perkembangan yang belum lama ini terjadi adalah adanya standar enkripsi baru yang diadopsi oleh sebagian besar dunia. Standar yang baru ini dinamakan *Advanced Encryption Standard* (AES). Standar yang baru ini digunakan karena adanya kepentingan dimana standar enkripsi yang lama, dalam hal ini *Data Encryption Standard* (DES), sudah dianggap tidak aman lagi karena sudah banyak yang dapat merusak keamanannya.

AES memiliki beberapa standar yang harus dipenuhi. Diantaranya adalah algoritma enkripsi harus bisa mendukung panjang kunci 128, 192, dan 256 bit. Algoritma yang dirancang juga harus menggunakan metode algoritma kunci simetrik dan berbasiskan *cipher* blok. Dua algoritma yang sangat sesuai dengan standar yang baru ini adalah algoritma Rijndael dan Serpent. Kedua algoritma ini menggunakan algoritma kunci simetrik dan berbasiskan *cipher* blok. Kedua algoritma ini juga menggunakan sistem cipher berulang.

Rijndael memiliki keunggulan dibandingkan Serpent dalam hal performansi dan kecepatan proses. Desain Rijndael juga lebih sederhana. Sedangkan keunggulan Serpent dari Rijndael adalah di dalam hal keamanan karena jumlah putaran yang lebih banyak dari Rijndael. Desain Serpent dalam hal ini sangat konservatif namun tetap mendukung implementasi yang efisien dan mendukung *rapid avalanche* dan implementasi *bitslice* yang lebih efisien.

Kata kunci: Algoritma Kunci Simetrik, Algoritma *Cipher Block*, *Advanced Encryption Standard*, *Cipher* berulang, Rijndael, Serpent.

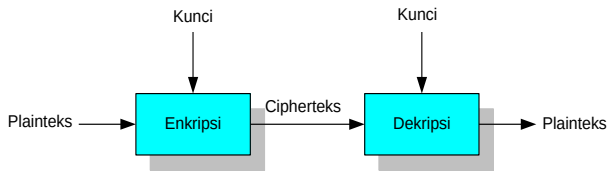
1. PENDAHULUAN

1.1 Kriptografi

Bruce Schneier, pada buku *Applied Cryptography*, mendeskripsikan kriptografi sebagai ilmu dan seni untuk menjaga kerahasiaan informasi. Sedangkan A. Menezes, P. van Oorschot, dan S. Vanstone menjelaskan pada buku *Handbook of Applied Cryptography*, "Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data." Namun secara umum, kriptografi adalah sebuah ilmu yang mempelajari tentang kerahasiaan pesan atau informasi.

Kriptografi memiliki empat tujuan mendasar yang juga merupakan aspek keamanan informasi, yaitu :

a. Kerahasiaan (confidentiality), merupakan layanan yang digunakan untuk menjaga isi pesan dari pihak yang tidak berhak membacanya. Dengan ini, hanya pihak yang memiliki otoritas atau kunci rahasialah yang dapat mengakses pesan.
b. Integritas data (data integrity), merupakan layanan yang menjamin bahwa pesan masih asli dan belum dimanipulasi selama pengiriman. Sistem yang dibuat harus memiliki kemampuan untuk mendeteksi manipulasi data yang dilakukan oleh pihak yang tidak berhak.
c. Otentikasi (authentication), merupakan layanan yang digunakan untuk mengidentifikasi kebenaran pihak-pihak yang berkomunikasi dan kebenaran sumber pesan.Pihak-pihak yang berkomunikasi harus saling memperkenalkan diri. Pesan yang dikirimkan melalui sebuah media harus diotentikasi keasliannya, isinya, waktu pengirimannya, dan informasi lain yang dirasa harus untuk diotentikasi.
d. Nirpenyangkalan (non-repudiation), merupakan layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan



Gambar 1. Skema umum proses penyandian

Kriptografi banyak melibatkan proses penyandian. Dalam proses penyandian, ada dua proses utama, enkripsi dan dekripsi. Enkripsi merupakan proses merubah plainteks menjadi cipherteks sedangkan dekripsi merupakan proses merubah cipherteks menjadi plainteks. Dalam kedua proses tersebut digunakan yang dinamakan kunci. Kunci merupakan sebuah informasi yang digunakan dalam proses enkripsi dan dekripsi seperti katalis dalam reaksi kimia. Jika kunci tidak digunakan, maka proses enkripsi dan dekripsi tidak akan berjalan. Jika kunci yang digunakan salah atau tidak sesuai dengan semestinya, maka hasil dari proses penyandian akan salah.

Proses penyandian sudah digunakan sejak zaman romawi. Saati itu, metode penyandian yang dilakukan adalah dengan melakukan penggeseran tiap huruf pada pesan sesuai dengan urutan alfabet. Proses penyandian ini dikenal dengan nama *Caesar Cipher*. Proses penyandian kemudian terus berkembang hingga saat Perang Dunia ke-2 mulai digunakan algoritma kunci simetrik dalam proses penyandian.

1.2 *Advanced Encryption Standard*

Advanced Encryption Standard (AES) merupakan sebuah standar enkripsi yang diadopsi oleh pemerintah Amerika Serikat yang saat ini sudah digunakan secara luas dan dianggap akan menggantikan standar enkripsi yang lama, *Data Encryption Standard* (DES) yang dianggap sudah tidak aman lagi. AES pertama kali dipublikasikan oleh *National Institue of Standards and Technology* (NIST) dengan mengadakan sayembara pencarian standar algoritma kriptografi. Sayembara atau lomba pencarian standar baru ini mempunyai beberapa kriteria :

a. Termasuk ke dalam kelompok algoritma kriptografi yang menggunakan kunci simetrik dan berbasis *cipher* blok.
b. Seluruh desain atau rancangan algoritma harus dapat dilihat oleh publik (tidak dirahasiakan).
c. Panjang kunci yang digunakan berukuran fleksibel : 128, 192, atau 256 bit.
d. Dalam sekali proses enkripsi, ukuran blok yang digunakan adalah 128 bit.
e. Algoritma dapat diimplementasikan baik sebagai perangkat lunak maupun perangkat keras.

AES tidak menggunakan metode *Feistel Network* seperti pada DES melainkan meggunakan *Subtitution Permutation Network*.

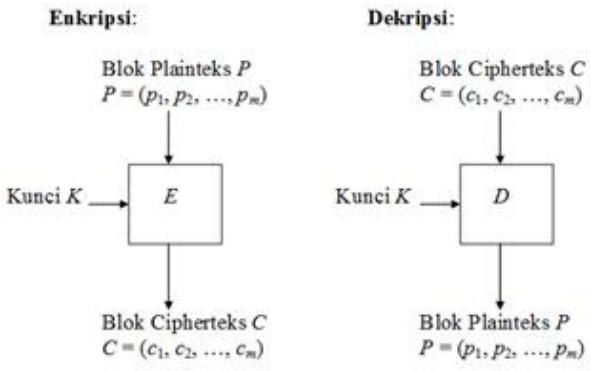
1.3 Algoritma Kunci Simetrik

Algoritma kunci simetrik merupakan algoritma kriprografi yang menggunakan kunci yang sama untuk kedua proses enkripsi dan dekripsi. Karena kunci merupakan satu-satunya jalan untuk proses penyandian, maka kerahasiaan kunci menjadi hal yang sangat utama. Untuk dapat mengirimkan kunci tersbut ke pihak yang akan menerima pesan tanpa ada pihak lain yang dapat memperolehnya merupakan masalah awal dari algoritma kunci simetrik.

Kemudian dalam proses mencoba memecahkan masalah pada algoritma kunci simetrik ini, muncul algoritma kunci asimetrik. Sebuah algoritma kunci asimetrik adalah algoritma yang menggunakan kunci yang berbeda saat proses enkripsi dan proses deskripsi. Dengan begini, permasalahan pengiriman kunci tanpa diketahui oleh pihak yang tidak berhak mengetahuinya dapat terselesaikan karena pihak yang akan menerima pesan telah mempunyai kunci sejak awal. Namun, terlepas kelebihannya yang dapat menyelesaikan permasalahan algoritma kunci simetrik, algoritma kunci asimetrik memiliki kekurangan dalam segi performansi karena sangat lamban dalam proses enkripsi dan dekripsi.

1.4 Algoritma *Cipher Block*

Cipher Block merupakan salah satu metode pengerjaan algoritma kunci simetrik. Perbedaan metode ini dengan metode *Cipher Stream* adalah pada metode *Cipher Block*, proses penyandian langsung melibatkan suatu kumpulan bit sekaligus sebagai satu unit berbeda dengan operasi bit per bit.



Gambar 2. Skema Algoritma *Cipher Block*

Dalam pembuatan algoritma *Cipher Block* perlu diperhatikan beberapa prinsip dasar algoritma *Cipher Block* diantaranya *Confussion* dan *Diffusion* dari Shanon, *Cipher* berulang, Jaringan Feistel, Kunci Lemah, Kotak-S, Kotak-P, Ekspansi, dan Kompresi. Prinsip dari *confussion*

adalah membuat kriptanalis kesulitan mencari pola statistik yang muncuk pada cipherteks. Prinsip dari *diffusion* adalah menjelaskan bahwa perubahan yang terjadi pada satu bit plainteks atau kunci akan menyebabkan banyak pengaruh pada cipherteks.

Cipher berulang pada dasarnya adalah merupakan fungsi transformasi sederhana yang mengubah plainteks menjadi cipherteks yang dilakukan sejumlah kali. Jaringan Feistel adalah sebuah model yang memungkinkan fungsi enkripsi dan dekripsi dalam proses penyandian tidak perlu berbeda atau dibalik. Model jaringan Feistel dirancang sedemikian sehingga fungsi enkripsi dapat dibuat serumit mungkin karena tidak perlu memikirkan bagaimana cara merancang fungsi dekripsinya karena seperti disebutkan sebelumnya, fungsi dekripsi dapat dibuat sama dengan fungsi enkripsinya.

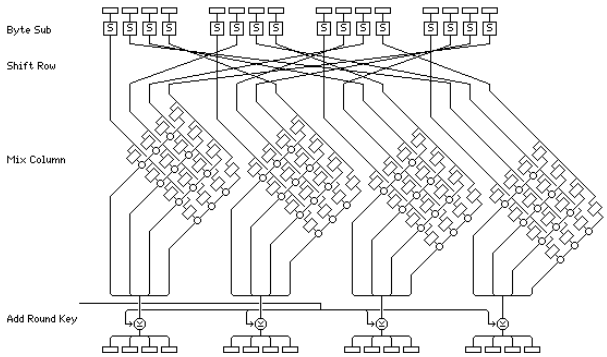
Kunci lemah adalah kunci yang menyebabkan tidak adanya perbedaan antara proses enkripsi dan dekripsi. Proses dekripsi terhadap cipherteks menghasilkan plainteks dan proses enkripsi dua kali juga menghasilkan kembali plainteks. *Cipher Block* yang baik tidak mempunyai kunci lemah. Prinsip kotak-S dan kotak-P adalah matriks subtitusi dan transformasi sederhana. Kotak-S akan memetakan suatu bit menjadi satu atau beberapa bit sedangkan kotak-P akan mengacak posisi bit dari suatu blok.

Prinsip ekspansi akan memperbanyak jumlah bit pada blok plainteks berdasarkan aturan tertentu. Sedangkan kompresi merupakan kebalikan dari prinsip ekspansi dimana jumlah bit pada blok dikurangi berdasarkan aturan tertentu seperti yang dilakukan oleh *Data Encryption Algorithm* (DEA).

2. METODE

2.1 Algoritma Rijndael

Algoritma Rijndael dikembangkan oleh Vincent Rijmen dan Joan Daemen dari Belgia. Algoritma ini pertama kali dirancang pada sebuah bank di Belanda yang mempunyai banyak sekali mesin Anjungan Tunai Mandiri (ATM) dan pada saat itu menunjukkan banyak sekali kesalahan. Algoritma ini dirancang untuk memenuhi tiga kriteria, yaitu tahan terhadap semua jenis serangan, kecepatan dan ukuran kode yang sangat kecil serta dapat digunakan pada banyak *platform*, dan kesederhanaan yang design namun sulit untuk dipecahkan.



Gambar 3. Skema umum Algoritma Rijndael

Rijndael menerima masukan array byte 1 dimensi yang menampung 8 bit dan dijadikan sebagai satu blok data. Fitur utama dari Rijndael adalah fleksibilitas ukuran kunci dan blok dimana Rijndael mendukung pankang kunci dan blok 128, 192, dan 256 bit. Dengan begini, kemungkinan kunci yang ada akan jauh lebih banyak sehingga proses pemecahannya pun memerlukan waktu yang jauh lebih lama. Rijndael menggunakan sebuah *sub key* yang didapatkan dari kunci eksternal yang dikenakan proses *key* schedule. Proses yang dilakukan oleh algoritma Rijndael diawali dan diakhiri dengan proses mencampur *sub key* dengan blok data. Proses enkripsi dan dekripsi dengan algoritma Rijndael diawali dengan menjalankan fungsi Add Round Key (meng-XOR-kan *sub key* dengan blok data) diikuti dengan sembilan putaran proses transformasi, dan pada putaran kesepuluh dilakukan proses Mix Column.

Tiap tahapan transformasi dilakukan beberapa langkah, yaitu Byte Sub, Shift Row, Mix Column, dan Add Round Key. Byte Sub adalah langkah subtitusi byte non-linear yang dilakukan pada tiap byte menggunakan tabel subtitusi.

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

S-BOX

Gambar 4. Kotak Subtitusi Algoritma Rijndael

Langkah Shift Row digunakan untuk menjadikan sebuah byte tidak selalu dan hanya berinteraksi dengan sebuah byte tertentu pada row lain. Shift Row dilakukan

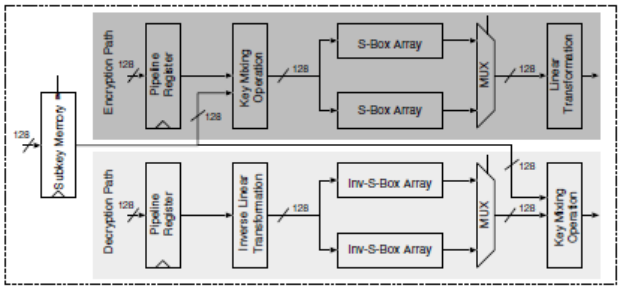
dengan cara row ke-0 dari state tidak digeser sama sekali. Row ke-1 digeser sebanyak satu byte. Row ke-2 digeser sebanyak dua byte. Row ke-3 digeser sebanyak tiga byte.

Tahapan Mix Column adalah mengoperasikan byte-byte hasil Shift Row. Dengan begini, tiap byte pada kolom akan mempengaruhi byte lain. Prosedurnya adalah mengalikan tiap kolom pada state dengan polinom a(x) mod (x^4+1). Setiap kolom diperlakukan sebagai polinom 4-suku pada GF(2^8). Langkah Add Round Key akan membuat round key baru untuk sebuah tahapan transformasi seperti yang telah dilakukan padaawal proses enkripsi atau dekripsi.

Gambar 5. Matriks yang digunakan dalam Mix Column

2.2 Algoritma Serpent

Algoritma Serpent merupakan hasil kerjasama internasional yang melibatkan tiga negara Inggris, Israel, dan Norwegia melalui tiga buah universitasnya. Inggris dalam hal ini diwakili oleh Cambridge University sedangkan Israel diwakili oleh Haifa Israel dan Norwegia diwakili oleh University of Bergen. Algoritma ini dikembangkan oleh Ross Anderson dari Inggris, Eli Biham dari Israel, dan Lars Knudsen dari Norwegia. Secara umum, algoritma Serpent merupakan pengembangan dari DEA. Algoritma ini mendukung kunci dengan panjang mulai dari 40 bit hingga 256 bit.



Gambar 6. Skema Algoritma Serpent

Ide dasar dari algoritma Serpent adalah implementasi *bitslice* dimana seseorang dapat menggunakan prosesor 1 bit untuk dapat mengimplentasikan algoritma ini. Dengan begini, akan bisa dilakukan komputasi 32 blok secara parallel menggunakan prosesor 32 bit. Desain dari algoritma ini juga mempunyai implementasi yang efisien.

Algoritma Serpent terdiri dari beberapa tahapan. Initial permutation IP. Putaran transformasi sebanyak 32 putaran yang pada tiap putarannya dilakukan operasi *key mixing*, subtitusi dengan S-box pada tiap putaran kecuali putaran terakhir dan transformasi linear dimana transformasi ini akan digantikan dengan operasi *key mixing* pada putaran terakhir. Dan terakhir adalah permutasi kembali IP_{-1}. Cipherteks membutuhkan 132 kunci dengan panjang 32 bit yang diperoleh dari 256 bit kunci dari pengguna sehingga menghasilkan *prekey* sejumlah 132. *Round key* kemudian diperoleh dengan mentransformasi *prekey* menggunakan S-box.

S-box pada Serpent merupakan permutasi 4 bit yang memiliki beberapa karakteristik. Karakteristik differential dimana perubahan 1 bit input akan mengakibatkan perbedaan pada 1 bit output. Karakteristik linear serta non-linear order. S-box pada Serpent dihasilkan atau dibuat melalui proses *deterministic pseudorandom* dimana sebuah matix diinisialisasi dengan 32 baris dari DEA S-box dan kemudian diubah nilainya menggunakan sebuah kunci sedemikian sehingga kita dapatkan delapan buah S-box yang memenuhi karakteristik-karakteristik di atas. Transformasi linear dibuat sedimikian sehingga memaksimalkan efek *avalanche* dan sederhana. Efek *avalanche* adalah dimana perubahan 1 bit input yang dikenakan operasi S-box berefek pada berubahnya 2 bit output. Sederhana agar dapat digunakan pada prosesos moderen. Hasil cipherteks dari Serpent baru mulai terlihat merupakan hasil yang acak setalah sampai putaran ke empat dari proses keseluruhan.

2.3 Rijndael vs Serpent

Perbandingan yang dilakukan antara Rijndael dan Serpent adalah dalam hal keamanan dan algoritma. Keamanan dilihat dari bagaimana algoritma mampu menahan serangan kriptanalis dan tingkat keacakan hasil keluaran. Perbandingan algoritma dilihat dari fleksibilitas, kecocokan serta kesederhanaan perangkat keras dan lunak dari algoritma serta biaya dan kecepatan dari implentasi perangkat lunak.

Implentasi Rijndael sangat fleksibel karena adanya dukungan dengan bermacam-macam panjang kunci dan blok. Pada Rijndael dimungkinkan juga mengubah urutan dari beberapa tahapan tanpa merubah cipherteks karena cipherteks mempunyai struktur yang sederhana. Dengan kesederhanaan ini, algoritma ini mendapatkan keuntungan karena kemudahannya dalam menggunakan komponen sederhana yang terdefinisi secara baik. Dengan begini, akses terhadap cipherteks dapat dengan mudah bisa dilakukan dibandingkan dengan jika mencoba untuk emngakses ciperteks yang mempunyai tingkat kerumitan yang cukup tinggi.

Tabel 1. Perbandingan performansi Serpent dan Rijndael pada dua arsitektur

	Rijndael	Serpent
IA64	504	2269
PA-RISC	666	2415

Ket : Satuan diatas adalah *average clock cycle*

Dalam hal performansi perangkat lunak dan perangkat keras, Rijndael secara konsisten memberikan kinerja yang sangat baik dan mempunyai lingkungan implementasi yang luas. Kebutuhan memori untuk Rijndael sangat kecil sehingga sangat cocok untuk lingkungan implementasi yang mempunyai batasan ruang. Ukuran algoritma ini pada prosesor 8 bit sangat luar biasa kecil, panjang kode hanya berukuran 1 kbit dan kebutuhan RAM untuk panjang kunci 256 bit hanya sebesar 52 byte. Algoritma ini mendukung panjang kunci mulai dari 128 bit sampai 256 bit dengan ukuran blok 32 bit pada tiap langkah. Kecepatan proses yang ditawarkan pada Pentium 200 adalah 27 megabit per detik dengan kunci 128 bit dan 19.8 megabit per detik untuk kunci 256 bit. Dengan menggunakan Visual C++, angka tersebut meningkat 250% menjadi 70.5 megabit per detik untuk kunci 128 bit dan 51.2 megabit per detik untuk kunci 256 bit. Algoritma Rijndael juga sangat tahan terhadap serangan yang mengandalkan kekuatan. Hal ini dibuktikan dengan sebuah skenario serangan Brute Force dimana jika komputer tercepat di dunia bisa mencoba 1 juta kunci tiap detiknya, maka komputer membutuhkan waktu lebih dari 10^{24} tahun untuk mencoba semua kemungkinan kunci pada Rijndael dengan 128 bit kunci karena pada Rijndael 128 bit ada 2^{128} kunci.

Namun, Rijndael cukup lemah terhadap serangan linear dan differential oleh kriptanalis. Rijndael juga secara teori lemah terhadap serangan bernama Square Attack. Serangan ini berbasiskan bagaimana cara kerja perkalian pada matriks dan karena Rijndael menggunakan tahapan perkalian matriks pada Mix Column, secara teori serangan ini dapat berhasil pada Rijndael. Namun pada prakteknya, serangan ini masih belum cukup untuk meruntuhkan keamanan dari algoritma Rijndael. Rijndael juga mempunyai keterbatasan pada *inverse cipher*. Karena Rijndael membutuhkan waktu untuk melakukan *inverse cipher* dikarenakan rumitnya kode untuk melakukan *inverse cipher*.

Algoritma Serpent mempunyai dasar prinsip-prinsip yang sudah sangat dipahami dan melalui banyak analisis karena juga didasari oleh yang sudah digunakan pada DEA. Algoritma ini mendukung kunci dengan panjang mulai dari 40 bit hingga 256 bit. Kecepatan proses algoritma Serpent pada Pentium 200 adalah 14.7 megabit per detik. Ukuran Serpent dalam prosesor juga cukup kecil sehingga akan dapat dengan baik diimplementasikan pada perangkat keras serta lingkungan yang mempunyai batasan ruang khususnya ruang memori.

Dalam hal performansi, Rijndael lebih baik dari Serpent dalam kedua proses enkripsi dan dekripsi ika digunakan pada prosesor 8 bit, 64 bit, 32 bit SmartCard, dan Digital Signal Processors. Sedangkan Serpent mengungguli Rijndael dalam prosesor 32 bit C dan Java.

Dalam hal performansi dapat dilihat bahwa Rijndael lebih baik dari Serpent, bahkan kecepatan pemrosesan Rijndael hampir dua kali dari kecepatan Serpent. Hal ini sedikit banyak dipengaruhi oleh jumlah putaran pada Rijndael yang hanya berjumlah 10 putaran dibandingkan dengan Serpent yang memiliki 32 jumlah putaran. Bila dalam hal performansi Rijndael dikatakan lebih baik, berbeda lagi dalam hal kemanan. Dalam hal keamanan, Serpent dapat dikatakan lebih baik dari Rijndael karena menggunakan 32 putaran yang tingkat keamanannya sangat tinggi. Kedua algoritma ini memang terbukti aman namun dilihat dari jumlah putarannya, Serpent akan lebih sulit diretas keamanannya.

Tabel 2. Perbandingan operasi algoritma Rijndael dan Serpent

	Rijndael	Serpent
Transformasi Awal	AddRoundKey	Permutasi IP
Putaran Transformasi	9x SubBytes ShiftRows MixColumns AddRoundKey	31x Key Mixing S-boxes Transformasi Linear
Tranformasi Akhir	SubBytes ShiftRows AddRoundKey	Key Mixing S-boxes Key Mixing Permutasi IP_{-1}

Dalam hal desain, desain Rijndael lebih sederhana dari dibandingkan Serpent karena menggunakan komponen yang sederhana. Desain Rijndael juga mendukung *parallel processing* yang sangat menguntungkan dimana perkembangan komputer saat ini adalah ke arah komputer dengan prosesor yang dapat mengeksekusi banyak instruksi secara paralel. Sedangkan Serpent, walaupun desainnya tergolong konservatif, masih dapat mendukung implementasi yang efisien dan sangat aman terhadap semua jenis serangan kriptografi. Desain Serpent yang didasari dari prinsip-prinsip yang matang, maka tingkat keamanan Serpent akan sangat praktikal dan bukan hanya sekedar teori. Desain Serpent, khususnya struktur yang ada pada algoritma Serpent secara simultan mendukung *rapid avalanche* dan implementasi bitslice yang lebih efisien. Dalm perancangan kotak subtitusi (S-box), algoritma Serpent dirancang tidak bersesuaian dengan desain algoritmanya secara keseluruhan sedangkan pada Rijndael, kotak Subtitusi (S-box) dirancang sesuai dengan desain keseluruhan dari algoritmanya.

3. KESIMPULAN

Standar enkripsi baru memang sangat dibuthkan dalam proses pengiriman pesan karena standar yang lama sudah termakan oleh kemajuan zaman. Ilmu kriptografi semakin lama semakin berkembang ke arah pembaruan yang disebabkan oleh utamanya perkembangan teknologi dan informasi. Munculnya standar baru AES memang terbukti sangat baik karena langsung diadopsi oleh sebagian besar pihak di dunia. Dalam menentukan standar AES yang terbaik dilakukan perbandingan dan penilaian dalam banyak hal terutama keamanan, performansi, dan desain.

Pembahasan terhadap algoritma Rijndael dan Serpent dilakukan berdasarkan terutama pada performansi dan keamanan serta kaitannya dengan desain algoritma secara umum. Dalam hal performansi, Rijndael jauh mengalahkan Serpent dalam banyak prosesor dan arsitektur. Dan Serepent hanya mampu mengalahkan performansi Rijndael pada prosesor 32 bit. Dalam segi keamanan, dapat dikatakan Serpent lebih baik dari Rijndael karena lebih sulit untuk dibobol dan keamanan dari Serpent mampu mengatasi semua jenis serangan kriptanalis. Namun untuk keamanan, Rijndael juga tidak begitu buruk bahkan menurut standar AES sudah memenuhi standar keamanannya.

Dalam segi desain, dipastikan Rijndael memiliki desain yang lebih sederhana. Desain algoritma Rijndael mempunyai struktur sejumlah 10 putaran sedangkan Serpent mempunyai proses yang lebih rumit karena jumlah putarannya jauh lebih banyak yaitu 32 putaran. Desain Serpent yang konservatif mungkin dirasa kaku atau mirip dengan desain DEA, namun desain ini mendukung implementasi yang efisien, *rapid avalanche*, dan implementasi *bitslice* yang lebih efisien.

REFERENSI

[1] Wikipedia Indonesia
http://id.wikipedia.org/wiki/Kriptografi
Tanggal akses : 21 Maret 2010 pukul 13.35

[2] Kuliah Kriptografi
http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/PengantarKriptografi2006.pp
Tanggal akses : 21 Maret 2010 pukul 13.35

[3] English Wikipedia
http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
Tanggal akses : 2 Maret 2010 pukul 7.04

[4] Kuliah Kriptografi
http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Advanced%20Encryption%20Standard%20(AES).ppt
Tanggal akses : 21 Maret 2010 pukul 14.15

[5] Makalah Ilmiah
http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Makalah1/Makalah1-004.pdf
Tanggal akses : 10 Maret 2010 pukul 17.00

[6] Makalah Ilmiah
http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Makalah1/Makalah1-009.pdf
Tanggal akses : 21 Maret 2010 pukul 14.40

[7] Halaman Web
http://www.seifried.org/security/cryptography/20001002-aes.html
Tanggal akses : 2 Maret 2010 pukul 6.49

[8] Halaman Web
http://www.cs.mcgill.ca/~kaleigh/computers/crypto_rijndael.html
Tanggal akses : 2 Maret 2010 pukul 7.03

[9] Halaman Web
http://csrc.nist.gov/publications/nistbul/itl99-08.txt
Tanggal akses : 2 Maret 2010 pukul 6.59

[10] Halaman Web
http://www.cl.cam.ac.uk/~rja14/serpent.html
Tanggal akses : 2 Maret 2010 pukul 7.04