Perbandingan Algoritma RC6 dengan Rijndael pada AES

Igor Bonny Tua Panggabean [1)]
1) Jurusan Teknik Informatika ITB, Bandung 40132, email: if14022@students.if.itb.ac.id

Abstract *– Data Encryption Standard (DES) telah menjadi standar algoritma enkripsi sejak lama. Namun seiring dengan perjalanan waktu, semakin lama standar ini semakin tidak aman dan mudah untuk diserang. Oleh sebab itulah National Institute of Standards and Technology (NIST) mengusulkan pembentukan standar baru yang diberi nama Advanced Encryption Standard (AES). Dalam perlombaan memperebutkan kandidat AES yang baru, terpilihlah algoritma Rijndael sebagai pemenang, dan dinobatkan sebagai standar algoritma enkripsi yang baru berdasarkan pemungutan suara. Namun NIST sendiri menyatakan pada putaran final bahwa setiap algoritma yang dipertandingkan memiliki kelebihan dan kekurangan masing-masing, dan tidak ada satupun algoritma yang jauh lebih baik daripada algoritma lainnya. Maka pada makalah ini, akan dilakukan perbandingan algoritma RC6 sebagai salah satu finalis AES pada putaran final dengan algoritma Rijndael yang dinobatkan sebagai AES dengan melihat dimana kelemahan, kemiripan dan kelebihan algoritma RC6 sebagai kandidat AES bila dibandingkan dengan algoritma Rijndael beserta implementasi pada algoritmanya. Kendati keduanya sudah mulai beredar sekitar tahun 2000, namun keduanya tidak mengalami perkembangan yang berarti hingga saat ini, bahkan minimal 10 tahun ke depan*

Kata Kunci*: RC6, Rijndael, AES, Block Cipher*

1. PENDAHULUAN

Advanced Encryption Standard telah menjadi standar baru dalam dunia algoritma. Sejarah mencatat bahwa dalam usaha mencari standar baru ini, 5 algoritma telah diikut sertakan dalam nominasi AES. Pada akhirnya, algoritma Rijndael yang ditemukan oleh Vincent Rijmen dan Joan Daemen dinobatkan sebagai standar AES.

Termasuk sebagai salah satu nominasi AES adalah algoritma RC6. RC6 dikembangkan oleh RSA Labs dengan berdasarkan kepada algoritma RC5. Walaupun RC5 sendiri belum pernah dikenai serangan, namun RSA Labs secara khusus sengaja membangun algoritma RC6 untuk dinominasikan sebagai AES.

Walaupun penobatan terhadap AES telah dilakukan, namun masih merupakan pembahasan yang menarik ketika ingin melihat perbandingan antara setiap nominasi AES. Namun pada makalah ini, perbandingan hanya dilakukn terhadap dua buah algoritma yaitu algoritma Rijndael dan algoritma RC6. Alasan memilih algoritma RC6 sebagai pembanding untuk pemenang AES adalah RC6 merupakan hasil pengembangan dari algoritma lama yaitu RC5 yang terbukti telah memiliki banyak sejarah dalam dunia kriptografi sejak ditemukannya algoritma RC1. Sehingga bisa dikatakan bahwa algoritma RC6 sebagai pemain lama dalam bidang kriptografi. Sedangkan Rijndael sebagai algoritma baru, menggunakan proses *Enciphering* yang sama sekali berbeda dengan jaringan *feistel* yang selama ini sudah banyak digunakan. Sangat menarik mengapa algoritma ini menjadi pemenang dalam AES, mengalahkan RC6 yang sudah lama berada dalam dunia kriptografi.

Bila ditelusuri lebih dalam lagi, kedua proses ini pasti memberikan banyak perbedaan yang sangat menarik untuk dicari. Mulai dari efisiensi, sampai bagian keamanan. Oleh sebab itulah pada makalah ini, kedua algoritma ini akan diperbandingkan satu sama lain untuk dapat melihat dimana kekurangan, kesamaan, dan kelebihan algoritma masing-masing. Diharapkan, pada akhirnya dapat dianalisa mana algoritma yang baik untuk digunakan untuk masalah tertentu.

2. DASAR TEORI

2.1. Rijndael

Adalah algoritma yang beroperasi dalam *byte*, bukan dalam *bit*. Algoritma ini mampu melakukan enkripsi terhadap *plain text* sebesar 16 *byte* atau 128 *bit*.

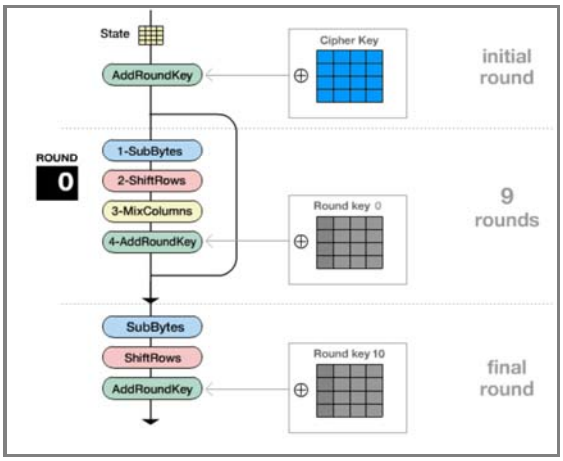
Selain itu, algoritma ini juga menggunakan kunci sebanyak 16 *byte*. Dengan kunci sepanjang 128 *bit*, maka terdapat 2^{128} = 3,4 x 10^{38} kemungkinan kunci. Dengan demikian, waktu yang dibutuhkan untuk menebak kunci yang ada dengan komputer yang cepat pun membutuhkan 10^{18} tahun[1]. Selain panjang kunci yang lumayan banyak, kunci internal pada algoritma ini juga selalu berubah pada setiap putarannya. Kunci internal ini disebut dengan *round key*. Pembangkitan *round key* diambil dari *cipher key*.

Mirip dengan DES, algoritma Rijndael juga melakukan putaran enkripsi (*enciphering*) sebanyak 10 putaran namun bukan putaran yang merupakan jaringan *Feistel*. *Enciphering* pada Rijndael melibatkan empat proses yaitu

1. *Sub Bytes*
2. *Shift Rows*
3. *Mix Columns*
4. *Add Round Key*

Secara umum, proses enkripsi dilakukan dengan *initial round* yaitu melakukan XOR antara *state* awal yang masih berupa *plain text* dengan *cipher key*. Kemudian melakukan keempat proses diatas sebanyak 9 kali putaran, dan terakhir adalah *final round* yang

melibatkan proses *sub bytes, shift rows,* dan *add round key*.



Gambar 1 : Proses Enkripsi Rijndael

2.1.1. Sub Bytes

Adalah proses subtitusi tiap *byte* pada *array state* dengan menggunakan tabel subtitusi *S-Box*. Sehingga terbentuk sebuah state baru yang tetap berukuran 16 *byte*. Proses subtitusi dilakukan terhadap *array state* dan *S-Box* yang berada pada urutan yang sama.
S-Box adalah tabel subtitusi non linear yang digunakan untuk mentransformasi subtitusi *byte* dan pada *key schedule* untuk melakukan subtitusi satu ke satu pada nilai *byte*-nya.

2.1.2. Shift Rows

Adalah proses melakukan pergeseran pada *array state* dengan cara siklik (*wrapping*). Namun pergeseran hanya dilakukan terhadap 3 baris terakhir dari *array state*.
Baris paling atas (baris r = 0) tidak dilakukan pergeseran, baris berikutnya (r = 1) dilakukan pergeseran sebanyak 1 kali. Baris berikutnya lagi (r = 2) dilakukan pergeseran sebanyak 2 kali, dan baris yang terakhir (r = 3) dilakukan pergeseran sebanyak 3 kali.

2.1.3. Mix Columns

Transformasi *mix columns* melakukan pengalian tiap kolom pada *array state* dengan polinom $a(x)$ mod $(x^4 + 1)$.
$a(x)$ yang ditetapkan adalah $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

2.1.4. Add Round Key

Adalah proses transformasi dengan melakukan operasi XOR terhadap *array state* dengan *round key* dan kemudian hasilnya disimpan sebagai *array state* yang baru.
Setelah ini, *round key* yang baru dibentuk lagi dengan melihat berdasarkan pada *round key* yang lama.

2.2. RC6

2.2.1. Garis Besar RC6

Algoritma yang mulai diperkenalkan sekitar tahun 1998 ini adalah hasil pengembangan dari algoritma RC5. Algoritma ini, sama seperti RC1 sampai RC5, dikembangkan oleh Laboratorium RSA di San Mateo, USA. Algoritma ini lebih dispesifikasikan dengan sebutan RC6-*w/r/b* dengan *w* adalah panjang ukuran kata dalam bit, *r* adalah jumlah putaran (*rounds*) dari proses enkripsi dengan *r* tidak negatif, dan *b* adalah panjang kunci enkripsi dalam *bytes*. Putaran yang dimaksudkan disini adalah sama dengan pada DES. Satu kali putaran sama dengan satu kali jaringan *Feistel*. Agar sesuai dengan persyaratan dari AES, maka RC6 menggunakan ukuran $w = 32$ dan $r = 20$ dengan panjang kunci mulai dari 16, 24, sampai 32 *byte*.
RC6 beroperasi pada empat unit kata dengan masing-masing sepanjang *w bit*. Operasi yang dilakukan antara lain sebagai berikut.

$a + b$	Penambahan modulo 2^w
$a - b$	Pengurangan modulo 2^w
$a \oplus b$	*Exclusive-or* untuk *w bit* kata
$a \times b$	Multiplikasi modulo 2^w
$a \lll b$	Rotasi untuk kata *a* ke kiri sebanyak jumlah yang diberikan *least significant log w bits* dari *b*
$a \ggg b$	Rotasi untuk kata *a* ke kanan sebanyak jumlah yang diberikan *least significant log w bits* dari *b*

Dengan operasi logaritma berbasis dua

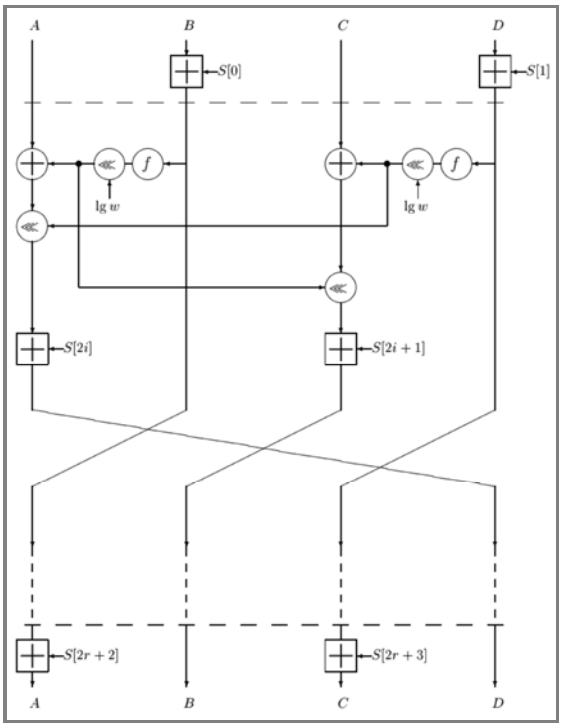
2.2.2. *Key Scheduling* pada RC6

RC6 juga melakukan proses enkripsi dengan menggunakan kunci yang berbeda pada setiap putarannya. Pembentukan kunci pada RC6 tidak terlalu berbeda dengan pembentukan kunci pada RC5. Secara garis besar, pembentukan kunci dilakukan dengan mengambil dari *b bytes* kunci yang dimasukkan oleh *user* dimana $0 \le b \le 255$. Sejumlah *byte* nol ditambahkan secukupnya untuk panjang kunci sama dengan jumlah bulat kata yang tidak nol. Kunci ini kemudian dimasukkan ke dalam array sepanjang *c* dimana tiap array diisi dengan satu *byte* dari kunci. Dari kunci ini, diambil sejumlah $2r + 4$ kata dengan panjang *w bits* tiap kata dan disimpan dalam array $S[0,...,2r + 3]$.

2.2.3. Operasi enkripsi dan dekripsi pada RC6

RC6 beroperasi menggunakan empat register *A, B, C, D* sepanjang *w bit* yang berisi masukan awal dari *plain text* dan bisa juga hasil akhir *cipher text* pada akhir proses enkripsi. *Byte* pertama dari *plain text* dimasukkan ke dalam *least significant byte A* dan *byte* terakhir dari *plain text* dimasukkan ke dalam *most significant byte D*.
Proses berikutnya adalah hasil enkripsi atau dekripsi

dengan menggunakan operasi pada jaringan *Feistel*.



Gambar 2 : Proses Enkripsi RC6

Jaringan *Feistel* pada gambar 2 adalah proses yang dilakukan selama satu kali putaran enkripsi. Dalam RC6, proses enkripsi tersebut akan diulang kembali sebanyak *r* kali putaran sesuai masukan dari pengguna.
Untuk proses dekripsi, tinggal melakukan urutan pada jaringan *Feistel* tersebut dari bawah ke atas.

3. PERBANDINGAN DAN PEMBAHASAN

Secara umum, RC6 adalah algoritma dengan deskripsi yang mudah diingat. Dan dengan adanya fungsi tambahan $f(x) = x(2x + 1)$ membuat serangan *cryptanalitic attack* pada RC5 menjadi tidak berarti. Ditambah lagi dengan kemampuan rotasi RC6 yang bersifat *data-dependent* sehingga menyebabkan keamanannya semakin meningkat.[4]
Sementara Rinjdael memiliki elemen algoritma yang dapat dideskripsikan sebagai perhitungan matematika sederhana termasuk *S-box*, bukan sebuah elemen yang menggunakan operasi acak. Serangan yang mungkin terjadi pada Rinjdael adalah *square attack*.
Pada makalah ini, perbandingan akan dilakukan pada tiga sektor yang menjadi kriteria standar pada AES. Untuk performansi akan dilakukan perbandingan pada komputer 32 *bit* dan *smart card*.

3.1. Flexibilitas

Untuk urusan bahasa pemrograman, Rijndael maupun RC6 dapat diimplementasikan baik pada bahasa Assembly, C maupun Java tanpa mengalami perbedaan performansi yang berarti. Implementasi pada ketiga bahasa ini, paling jauh, hanya mengalami perubahan performansi hingga 2:1. Namun kebanyakan hanya hingga 1:1 meskipun pada akhirnya ada begitu banyak varian kode yang dimunculkan.
Selain itu, kedua algoritma ini juga dapat diimplementasikan untuk keamanan *smart card*.

3.1.1. Fleksibilitas Rijndael

Sebagai varian dari *Square Cipher*, Rijndael memiliki kemampuan untuk bekerja sangat baik pada platform apapun. Ditambah dengan operasi yang menggunakan *table lookup* dan operasi XOR membuat prosesnya menjadi tidak terlalu rumit. Walaupun proses enkripsi dan dekripsinya tidak terlalu identik, namun struktur umum dan performanya tidak dapat secara langsung dibedakan.
Rijndael, memiliki keuntungan yang tinggi dalam *smart card* dikarenakan penggunaan ROM yang cukup kecil. Namun untuk penggunaan *smart card*, proses dekripsi malah membuat ukuran kodenya menjadi bertambah dan kecepatan dekripsi lebih lambat dari pada kecepatan enkripsi, mencapai 2 kali lipat.

3.1.2. Fleksibilitas RC6

Kode RC6 yang sangat pendek merupakan sebuah kemampuan tersendiri dari algoritma ini, dan membuat algoritma ini sangat sepadan bila diimplementasikan ke dalam lingkungan *smart card*. Ditambah lagi karena RC6 tidak melakukan proses pembuat *sub key* atau yang kita sebut *key scheduling* pada saat proses enkripsi.
Namun untuk *platform* yang lain, RC6 tidak terlalu banyak memberikan hasil yang sebaik *platform* pentium. Bahkan, RC6 berjalan lebih lambat hingga 3 kali lipat pada *platform* pentium pro.

3.2. Performansi

Secara umum, performansi dilihat dengan menghitung lama pembuatan *key* serta proses enkripsinya. Perbandingan dilakukan dengan menggunakan panjang kunci yang sama yaitu 128 *bit* pada prosesor pentium.

Tabel 1. Performansi Rijndael dan RC6

Cipher	**Rijndael**	**RC6**
Key setup Pentium Pro C (*Clock*)	2100	1700
Encrypt Pentium Pro C (*Clock*)	440	260
Encrypt Pentium Pro ASM (*Clock*)	320	250
Encrypt Pentium ASM (*Clock*)	320	700

3.2.1. Pada Kunci Enkripsi

Algoritma RC6 tidak tergantung pada panjang kunci. Sehingga, waktu yang dibutuhkan untuk membangun sebuah kunci adalah sama, meskipun panjang kuncinya adalah 128, 192, atau 256 *bit*. Sehingga proses enkripsi dan dekripsi tetap konstan. Sedangkan untuk Rijndael, melakukan enkripsi dan dekripsi lebih lama untuk kunci yang lebih panjang, dan membutuhkan waktu yang lama untuk membangun kunci. Sehingga, bila panjang kunci 128 *bit* membutuhkan 10 putaran, untuk kunci 192 *bit* akan membutuhkan waktu lebih lama 20%, dan kunci 256 *bit* membutuhkan waktu lebih lama 40%.

Tabel 2. *Clock Cycles per Bytes to Key and Encrypt Different Text Sizes on Pentium in Assembly*

Text Size (*Bytes*)	**Clock Cycles**	
	Rijndael	**RC6**
16	115	146
32	68	95
64	44	69
128	32	57
256	26	50
512	23	47
1024	21	45
2048	21	45

Pengukuran 16 *byte* adalah kecepatan algoritma bila dijadikan sebagai fungsi hash. Dari tabel 1 dapat dilihat bahwa Rijndael sangat baik untuk enkripsi data dengan blok yang kecil. Namun algoritma ini lebih cepat kembali ke kecepatan asalnya pada peningkatan selanjutnya.

3.2.2. *Minimal Secure Variant*

Minimal secure variant, sebuah konsep yang diperkenalkan oleh Biham, adalah cara perbandingan dengan melihat jumlah putaran minimal yang masih aman terhadap serangan, kemudian ditambahkan lagi dua putaran.

Tabel 3. Kecepatan enkripsi untuk MSV pada Assembly

	Clock Cyles	
	Rijndael	**RC6**
Pentium	256	700
Pentium Pro	256	250

Tabel 4. Kecepatan enkripsi untuk MSV pada Bahasa C

	Clock Cyles	
	Rijndael	**RC6**
Pentium	560	570
Pentium Pro	350	295

3.2.3. Pada *Smart Card*

Smart Card yang digunakan untuk pengujian adalah T6N55 dari Toshiba dengan Z80 mikroprosesor dan koprosesor, 48 KB ROM, 1 KB RAM, dan *internal clock frequency* 5 MHz.

Tabel 5. Perbandingan pada *Smart Card*

Cipher	**Rijndael**	**RC6**
RAM (*Bytes*)	66	156
ROM (*Bytes*)	980	1060
Encrypt (*Clock*)	25.494	34.736
Schedule (*Clock*)	10.318	138.851
Total	35.812	173.857

Penelitian lebih lengkap dapat dilihat pada [7].

3.3. Keamanan

Keamanan adalah salah satu faktor penting dalam dunia enkripsi. Melihat RC6 sebagai algoritma yang sedikit lebih sederhana dibandingkan dengan Rijndael, mengakibatkan keduanya membutuhkan proses kriptanalisis yang sangat cermat. Namun NIST mengatakan bahwa keduanya memiliki batas keamanan yang sangat cukup.

3.3.1. Keamanan Rijndael

Untuk Rijndael, tipe serangan *square attacks* cukup menjadi dikenal sebagai serangan terbaik terhadap Rijndael. *Square attacks* adalah serangan yang memanfaatkan struktur orientasi *byte*. Algoritma ini bekerja dengan baik pada *square cipher* yang bekerja dalam 6 putaran.

Untuk Rijndael dengan kunci sepanjang 128 *bit*, maka serangan ini lebih cepat dari pada *exhaustive search* hingga 6 kali iterasi Rijndael.

Namun untuk AES, jelas bahwa serangan ini tidak mungkin dipraktekkan karena jumlah putaran pada Rijndael, mengakibatkan batas keamanan untuk algoritma ini menjadi lebih besar.

3.3.2. Keamanan RC6

Serangan paling sederhana dan dapat menghasilkan pada RC6 diperkirakan adalah *exhaustive search* untuk kunci RC6. Serangan ini dapat digunakan pada *block cipher* apapun. Dengan RC6, operasi yang dibutuhkan untuk mencari *key* atau *expanded key array* adalah antara 2^8 hingga 2^{1408} operasi. Namun dapat digunakan *meet in the middle attack* sehingga dibutuhkan 2^{704} komputasi. Untuk ukuran panjang kunci yang bersesuaian dengan AES, serangan ini jelas tidak mungkin.[9]

4. KESIMPULAN

Dari hasil seluruh pembahasan di atas, dapat ditarik kesimpulan sebagai berikut.

1. Kedua algoritma Rijndael dan RC6 termasuk algoritma *block cipher*.
2. RC6 dikembangkan dari RC5 dengan menambahkan 4 buah register.
3. Rijndael bekerja dalam ukuran *byte* dan tergolong dalam *square cipher*.
4. Rijndael memiliki fleksibilitas yang tinggi karena dapat diterapkan pada *platform* yang beragam sementara RC6 kurang memiliki kebebasan ini.
5. Kode untuk RC6 lebih sederhana dibandingkan dengan Rijndael, dan juga RC6 tidak menghabiskan penggunaan memori yang banyak bila diterapkan ke dalam *smart card*.
6. Secara umum, performa Rijndael pada *smart card* masih melebihi performa RC6.
7. Penggunaan *lookup table* pada Rijndael cukup menjadi kelemahan dalam ukuran kode.
8. RC6 memiliki kelebihan dalam bidang *data dependent* dalam enkripsinya dibandingkan Rijndael.
9. Dalam proses enkripsi, keduanya menggunakan *key* yang berbeda untuk setiap putaran, namun Rijndael membangkitkan *key* bersamaan dengan proses enkripsi, sementara RC6 tidak melakukan pembangkitan *key*.
10. Untuk lingkungan implementasi 8 *bit* dan *hardware*, Rijndael jauh lebih unggul dibandingkan RC6 pada performansinya. Namun tidak berarti RC6 tidak mampu mengeluarkan performa yang baik.
11. Untuk implementasi pada bahasa C maupun assembly, secara umum RC6 menggungguli Rijndael. Namun perbedaan performansi dipengaruhi oleh banyak faktor juga. Sebagai contoh, RC6 tidak memiliki performa sebaik Rijndael pada prosesor pentium 32 *bit*.
12. Untuk urusan keamanan, kedua algoritma ini masih bekerja dalam performa yang sangat baik, terlebih karena jumlah putaran semakin mempersulit proses kriptanalisis.
13. Melihat kelebihan, kekurangan, dan kemiripan RC6 dibandingkan dengan Rijndael, maka dapat ditarik kesimpulan bahwa RC6 juga layak untuk dipertimbangkan bila ingin melakukan enkripsi pada *platform* tertentu.

DAFTAR REFERENSI

[1] R. Munir, "Advanced Encryption Standard (AES)", IF5054 Kriptografi, Program Studi Teknik Informatika Institut Teknologi Bandung.

[2] R.L. Rivest, M.J.B. Robshaw, R. Sydney, YL. Yin, "The RC6™ Block Cipher", v.1.1, August 20, 1998, www.rsasecurity.com/rsalabs/aes/

[3] R. Munir, "*Diktat Kuliah IF5054 Kriptografi*", Program Studi Teknik Informatika Institut Teknologi Bandung, 2006.

[4] L. R. Knudsen, "*Recommendation to NIST for the AES*", May 15, 2000, http://csrc.nist.gov/

[5] J. Daemen, V.Rijmen, "*AES Proposal : Rijndael v.2*", AES Submissions.

[6] B. Schneier, D. Whiting, "*A Performance Comparation of the Five AES Finalist*", March 15, 2000, http://csrc.nist.gov/

[7] F. Sano, M. Koike, S. Kawamura, M. Shiba, "*Performance Evaluation of AES Finalists on the High End Smart Card*", http://csrc.nist.gov/

[8] B. Scheneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, "*Performance Comparation of The AES Submissions*" v.1.4, http://www.ussrback.com/

[9] S. Contini, R. L. Rivest, M. J. B. Robshaw, Y. L. Yin, "*The Security of RC6 Block Cipher*", www.rsasecurity.com/rsalabs/aes/

[10] M. J. B. Robshaw, "*RC6 and The AES*", ftp://ftp.rsasecurity.com/

[11] S. Lucks, "*Attacking Seven Rounds of Rijndael Under 192 bit and 256 bit Keys*", http://csrc.nist.gov/