PERBANDINGAN ALGORITMA *BLOCK CIPHER* RC5 DAN RC6

Roland L. Bu'ulölö – NIM: 135 04 072

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-Mail: if14072@students.if.itb.ac.id

Abstrak

Makalah ini secara garis besar membahas tentang algoritma kriptografi kunci simetri RC5 dan RC6 dan perbedaannya. Mula-mula, akan dibahas mengenai serangan-serangan umum terhadap algoritma kriptografi kunci simetri dan kemudian akan dilihat secara lebih mendalam mengenai masing-masing algoritma (RC5 dan RC6) dan kemudian akan dibahas secara garis besar mengenai serangan-serangan dan analisis-analisis yang dilakukan terhadap kedua algoritma ini. Makalah ini akan lebih menekankan pada hasil analisis-analisis yang telah dilakukan terhadap algoritma ini.

RC5 dan RC6 keduanya adalah merupakan algoritma kriptografi kunci simetri yang terparameterisasi. Artinya, melalui parameter yang diberikan dapat dihasilkan *ciphertext* yang berbeda-beda tingkat keamanannya dan juga berbeda performa proses untuk menghasilkannya. Keduanya juga sangat bergantung pada penggunaan *data-dependent rotation* dalam proses enkripsi. Fitur ini sangat berguna dalam mencegah serangan-serangan kriptanalisis yang umum pada algoritma kriptografi kunci simetri. Dalam makalah ini akan juga ditunjukkan perbedaan implementasi *data-dependent rotations* yang ada pada RC5 dan RC6 yang mengakibatkan perbedaan keamanan dan perbedaan tingkat ketahanan terhadap serangan di antara keduanya.

Kata kunci: *RC5*, *RC6*, *differential cryptanalysis*, *linear cryptanalysis*, *exaustive key search*, *statistical test*, *block cipher*.

1. Pendahuluan

Dalam dunia digital sekarang ini, keamanan sudah merupakan kebutuhan yang umum dalam komunitas digital. Demi menjamin keamanan ini, telah diciptakan suatu metode penyandian (*encryption* dan *decryption*) terhadap data yang ingin dilindungi. Data yang dienkripsi tidak akan dapat dibaca/dimengerti oleh orang yang tidak berhak. Oleh karena itu, telah banyak pengembangan-pengembangan yang dilakukan untuk membentuk algoritma enkripsi-dekripsi (algoritma kriptografi) yang cepat dan aman. Salah satu dari jenis algoritma kriptografi yang populer adalah algoritma kriptografi kunci simetri.

Algoritma kriptografi kunci simetri adalah algoritma kriptografi yang kunci untuk enkripsi dan dekripsinya sama. Terdapat 2 jenis algoritma kriptografi (*cipher*) yaitu *cipher* aliran (*stream cipher*) dan *cipher* blok (*block cipher*). Namun, pada zaman sekarang algoritma *cipher* yang paling umum adalah yang bertipe *block cipher*.

RC5 dan RC6 keduanya adalah algoritma *cipher* kunci simetri. Perbedaan mendasar dari keduanya hanyalah pada algoritma yang diimplementasikan. Keduanya memiliki sifat-sifat umum algoritma *block cipher*, yaitu memiliki fungsi enkripsi dan dekripsi yang diaplikasikan pada 1 blok data dan menghasilkan 1 blok data hasil pemrosesan.

Jadi, faktor utama yang menentukan kekuatan (*strength*) dan kecepatan (*performance*) dari suatu algoritma *cipher* blok kunci simetri adalah dari implementasi algoritmanya. Algoritma yang baik akan sulit untuk dipecahkan dengan serangan-serangan kriptanalisis. Oleh karena itu, untuk merancang algoritma *cipher* yang aman adalah dengan mengetahui segala kemungkinan serangan yang dapat digunakan untuk menyerang algoritma tersebut dan merancangnya sedemikian rupa sehingga serangan-serangan tersebut dapat ditangkis.

2. Teknik Analisis (Serangan) terhadap *Block Cipher*

Beberapa teknik telah banyak dikembangkan untuk melakukan analisis terhadap *block cipher*. Diantaranya yang paling umum adalah *exaustive key search*, *statistical test*, *differential cryptanalysis*, dan *linear cryptanalysis*.

2.1 *Exaustive Key Search*

Serangan yang paling dasar yang selalu dapat digunakan untuk menganalisis *block cipher* adalah *exaustive key search*, atau disebut juga *brute force attack*. Dalam menggunakan serangan ini, umumnya penyerang memiliki pasangan *plaintext* dan *ciphertext* yang bersesuaian lalu mencoba setiap kandidat kunci yang mungkin sampai akhirnya ditemukan kecocokan sehingga kunci pun ditemukan. Jika panjang kunci adalah n bit maka akan ada 2^n kandidat kunci yang dicobakan, sehingga performa serangan ini sangat bergantung pada kunci rahasia. Jika ternyata diketahui bahwa panjang kunci rahasia lebih besar daripada panjang blok algoritma *cipher* yang digunakan, maka dibutuhkan lebih dari 1 pasangan *plaintext* dan *ciphertext* untuk menemukan kunci rahasia.

2.2 *Statistical Test*

Pengujian statistik dapat digunakan untuk menemukan sifat-sifat statistik dari *cipher* blok. Algoritma *cipher* blok yang kuat harus dapat menghasilkan *randomness* yang dapat menghilangkan hubungan-hubungan statistik antara *plaintext*, *ciphertext*, kunci rahasia, dan sebagainya. Namun, perlu ditekankan bahwa sifat statistik yang baik dari suatu algoritma *cipher*, yaitu sifat di mana korelasi antara elemen-elemen yang digunakan untuk melakukan proses enkripsi atau dekripsi dihilangkan, hanyalah merupakan syarat perlu untuk menrancang algoritmanya, karena walaupun suatu algoritma *cipher* telah berhasil melewati pengujian statistik belum tentu algoritma itu dapat dinilai aman.

2.3 *Differential Cryptanalysis*

Differential Cryptanalysis – kriptanalisis diferensial – yang pertama sekali diimplementasikan oleh E. Bilham dan A. Shamir menghasilkan efek yang cukup revolusioner dalam perancangan dan analisis *cipher* blok. Kriptanalisis diferensial adalah termasuk ke dalam tipe *chosen-plaintext attact*. Ide dasar dalam teknik ini, yaitu pemilihan dua *plaintext* dengan adanya "perbedaan" P' antara keduanya. Pada umumnya, 'perbedaan'-nya diukur dengan eksklusif-OR ($\oplus$), tetapi untuk beberapa algoritma *cipher* yang lain, alternatif pengukuran yang lain mungkin dapat lebih baik. Kedua *plaintext* ini dienkripsikan untuk memberikan hasil yaitu dua *ciphertext* yang memiliki "perbedaan" C' antara keduanya. Pasangan nilai (P', C') disebut sebagai karakteristik. Bergantung pada algoritma *cipher* dan analis yang dilakukan, sifat nilai karakteristik ini dapat berguna untuk menurunkan beberapa bit-bit tertentu yang ada pada kunci.

2.4 *Linear Cryptanalysis*

Linear cryptanalysis, yang termasuk ke dalam tipe *chosen-plaintext attack*, pertama kali ditunjukkan oleh M. Matsui dan juga merupakan terobosan dalam teori kriptanalisis. Ide dasar teknik ini adalah mencari hubungan antara bit-bit tertentu di dalam *plaintext*, *ciphertext*, dan kunci yang memenuhi nilai probabilitas $p \neq 0{,}5$ ($|p - 0{,}5| > 0$). Hubungan yang demikian disebut sebagai aproksimasi linear. Proses kriptanalisis dilakukan dengan mencari aproksimasi linear yang dapat digunakan untuk mendapat informasi tentang kunci.

3. RC5 *Block Cipher*

Algoritma *cipher* simetri RC5 dirancang oleh Profesor Ron Rivest dari Laboratorium RSA MIT. RC5 dipublikasikan pada Desember 1994. Algoritma ini dirancang sedemikian rupa sehingga memenuhi syarat-syarat berikut:

1. RC5 harus dirancang menjadi algoritma *cipher* simetri.
2. RC5 harus cocok untuk digunakan pada *hardware* dan *software*. Hal ini berarti RC5 hanya boleh menggunakan primitif-primitif komputasi yang umum ditemukan pada *microprocessors*.
3. RC5 harus berkecepatan tinggi. Berarti algoritma RC5 harus berorientasi *word*. Operasi-operasi RC5 harus dapat memproses 1 *word* penuh data.
4. RC5 harus dapat beradaptasi pada berbagai panjang *word*. Contohnya, pada prosesor terbaru 64-bit, panjang *word*-nya lebih panjang daripada prosesor 32-bit. RC5 harus dapat memanfaatkan ini, oleh karena itu RC5 memiliki parameter w yang menandakan panjang *word*.

5. RC5 harus dapat beroperasi dalam berbagai jumlah ronde. Jumlah ronde yang bervariasi memungkinkan pengguna untuk memanipulasi RC5 untuk menjadi lebih cepat atau lebih aman.
6. RC5 harus dapat beroperasi dalam berbagai panjang kunci. Hal ini mengakibatkan panjang kunci b menjadi parameter dalam algoritma RC5.
7. RC5 harus berstruktur sederhana. Struktur yang sederhana belum tentu menghasilkan keamanan yang rendah. Struktur yang sederhana akan memungkinkan analisis dan evaluasi yang cepat untuk menentukan kekuatan algoritma RC5.
8. RC5 harus hemat dalam pemakaian memori. Hal ini akan memungkinkan implementasi RC5 ke dalam *smart-card* atau perangkat lain yang memiliki keterbatasan memori.
9. RC5 harus mengimplementasikan metode *data-dependent rotations*. Metode ini adalah primitif kriptografi yang merupakan sasaran pengkajian RC5. *Data-dependent rotations* adalah suatu teknik yang merotasi data yang sekarang diproses secara sirkuler sebanyak N, di mana besarnya N tergantung data yang lain.

Seperti telah ditunjukkan oleh poin-poin di atas, terlihat bawa RC5 adalah algoritma *cipher* blok simetri yang terparameterisasi dengan 3 parameter utama, yaitu: w, r, dan b. Oleh karena itu algoritma RC5 dapat juga ditulis secara lebih spesifik dengan penulisan:

RC5 – w / r / b

Nilai w adalah parameter pertama yang menyatakan jumlah bit dalam 1 *word*. RC5 memproses 1 blok yang terdiri dari 2-*word* sebagai input dan menghasilkan 1 blok yang juga terdiri dari 2-*word* sebagai output. Pada umumnya panjang *word* yang dipilih adalah 32 bit (w=32), sehingga panjang 1 blok *plaintext* dan panjang 1 blok *ciphertext* dalam RC5 adalah 64 bit. Sebenarnya, semua nilai w (w > 0) yang mungkin dapat digunakan pada algoritma RC5, tetapi nilai-nilai yang diperbolehkan hanya: 16,32, dan 64.

Nilai r adalah parameter kedua dalam RC5 yang menyatakan jumlah ronde. Semakin besar jumlah ronde, maka semakin aman. Tiap ronde, RC5 menggunakan kunci yang berbeda-beda untuk melakukan proses enkripsi atau dekripsi. Oleh karena itu, RC5 membangun tabel kunci S yang dibentuk berdasarkan kunci yang diberikan. Ukuran t dari tabel kunci S bergantung pada jumlah ronde. Tabel kunci S memiliki t = 2(r+1) *word*.
Oleh karena itu, semakin besar jumlah ronde, semakin banyak memori yang dibutuhkan.

Nilai b adalah parameter ketiga dalam RC5. Nilai b menyatakan panjang kunci rahasia yang diberikan dalam satuan bytes.

Berikut adalah kesimpulan dari parameter-parameter yang ada pada RC5:

w Menyatakan panjang *word*. Setiap *word* memiliki panjang (w/8) bytes (1 byte = 8 bit). Nilai yang diperbolehkan adalah 16,32, dan 64. RC5 mengenkripsi dengan panjang blok 2w. *Plaintext* dan *ciphertext* memiliki panjang blok yang sama yaitu 2w.

r Menyatakan jumlah ronde. Nilai ini juga menentukan ukuran tabel kunci S. Tabel kunci S memiliki ukuran t = 2(r+1) *word*. Nilai yang diperbolehkan: 0,1,...,255.

b Menyatakan jumlah bytes dalam kunci rahasia K. Nilai yang diperbolehkan: 0,1,...,255.

K Merupakan kunci rahasia yang diperoleh dari pengguna. Memiliki ukuran b bytes. K adalah array yang terdiri dari b bytes: K[0], K[1], ..., K[255].

Parameter-parameter di atas menentukan tingkat keamanan dan kecepatan enkripsi dengan RC5. Kumpulan parameter-parameter di atas ditambah dengan data mengenai versi RC5 yang digunakan membentuk *control block* RC5. *Control block* RC5 adalah nilai parameter-parameter yang mempengaruhi proses kerja RC5. Oleh karena itu, *control block* RC5 terdiri dari:

v 1 byte. Angka versi RC5 dalam heksadesimal. Untuk versi 1.0 ditulis `10` (hex).

w 1 byte.

r 1 byte.

b 1 byte.

K b bytes.

Berdasarkan informasi di atas terlihat bahwa besar *control block* adalah (b+4) bytes. Sebagai contoh, *control block* berikut:

```
10 20 0c 0a 20 33 7d 83 05 5f 62 51 bb 09
```

Penjelasan *control block* tersebut adalah:

10 Versi RC5 yang digunakan, dalam hal ini versi 1.0.

20 20 (hex) sama dengan 32 (dec). Nilai ini menyatakan besaran w, yaitu panjang *word* yang adalah 32 bit.

0c 0c (hex) sama dengan 12 (dec). Nilai ini menyatakan besaran r, sehingga jumlah ronde adalah 12.

0a 0a (hex) sama dengan 10 (dec). Nilai ini menyatakan besaran b, yaitu panjang kunci yang adalah 10 bytes.

Setelah nilai b didapat (10 bytes), maka 10 bytes ke depan setelah nilai b menyatakan kunci rahasia K, yaitu:

```
20 33 7d 83 05 5f 62 51 bb 09
```

K merupakan *array of bytes*, sehingga pada contoh kunci ini nilai K[0]=20, K[1]=33, K[2]=7D, ..., K[9]=09.

3.1 Operasi-operasi Primitif

RC5 hanya menggunakan 3 operasi primitif (beserta inversnya), yaitu:

1. Operasi penjumlahan '+'. Operasi penjumlahan di sini adalah operasi penjumlahan *word*. Karena suatu *word* hanya terdiri dari bit-bit 1 dan 0 dan memiliki jumlah bit yang tetap, maka operasi ini adalah operasi penjumlahan modulo-2^w. Operasi inversnya adalah operasi pengurangan '-'.

2. Operasi ekslusif-OR, atau xor dengan simbol '$\oplus$'.

3. Rotasi *word* ke kiri (*left-spin*). Simbolnya dinyatakan sebagai $x \lll y$, yang berarti rotasi ke kiri terhadap *word* x sebanyak y, di mana y adalah modulo w, sehingga jika w genap maka hanya bit-bit orde bawah sebanyak $^2\log$ (w) dari y saja yang digunakan untuk menentukan banyaknya rotasi. Operasi inversnya adalah rotasi ke kanan (*right-spin*), dengan simbol $x \ggg y$.

Kekuatan RC5 sangat bergantung pada operasi ke 3, yaitu rotasi dimana jumlah rotasi bergantung pada data (*data-dependent rotations*).

3.2 Algoritma RC5

Algoritma RC5 terdiri dari 3 komponen, yaitu:

1. *Key expansion algorithm*, yaitu algoritma yang digunakan untuk membangun tabel kunci S.

2. *Encryption algorithm*, yaitu algoritma untuk melakukan enkripsi pada 1 blok *plaintext* menghasilkan 1 blok *ciphertext*.

3. *Decryption algorithm*, yaitu algoritma untuk melakukan dekripsi pada 1 blok *ciphertext* menghasilkan 1 block *plaintext*.

3.2.1 *Key Expansion Algorithm*

Algoritma ini berfungsi untuk membangkitkan kunci internal berdasarkan kunci rahasia K untuk mengisi tabel kunci S, di mana ukuran tabel kunci S adalah t = 2(r+1) *word*. Algoritma ini dalam mengisi tabel kunci S, menggunakan 2 konstanta 'ajaib' (*magic constants*). P_w dan Q_w yang didefinisikan sebagai:

$$P_w = Odd\left((e-2)\times 2^w\right)$$
$$Q_w = Odd\left((\phi-1)\times 2^w\right)$$

Dengan:

$e = 2.718281828459...$ (basis logaritma natural)

$\phi = 1.618033988749...$ (*golden ratio*)

Nilai ϕ dapat juga didapatkan dengan rumus:

$$\phi = \frac{1+\sqrt{5}}{2}$$

Sedangkan fungsi *Odd(x)* menghasilkan nilai integer ganjil yang paling dekat dengan x. Untuk nilai-nilai w yang diperbolehkan (16, 32 dan 64), nilai P_w dan Q_w adalah sebagai berikut (dalam heksadesimal):

```
P16 = b7e1
Q16 = 9e37

P32 = b7e15163
Q32 = 9e3779b9
```

P_{64} = b7e151628aed2a6b
Q_{64} = 9e3779b97f4a7c15

Langkah pertama untuk membentuk tabel kunci S adalah dengan menduplikasi kunci rahasia K yang masih berupa *array of bytes* ke dalam *array of word* L[0...c-1] di mana c = (b/u) dengan u = (w/8) yang menyatakan jumlah bytes dalam 1 *word*. Jika ternyata panjang kunci b bukan merupakan kelipatan w, maka *padding* dengan sejumlah bit 0 diperlukan agar panjang akhirnya menjadi kelipatan w.

Setelah langkah pertama selesai, langkah ke-2 adalah melakukan inisialisasi tabel kunci S. Inisialisasi dilakukan sehingga tabel S berisi pola bit *pseudo-random* yang tetap. Pada tahap ini, pola yang dihasilkan tidak bergantung pada kunci rahasia, tetapi bergantung pada P_w dan Q_w. Algoritmanya adalah sebagai berikut:

$S[0] = P_w$
for i = 1 **to** (2r + 1) **do**
    $S[i] = S[i - 1] + Q_w$

Setelah inisialisasi selesai, langkah ke-3 adalah mencampurkan kunci rahasia pengguna ke dalam array S, yaitu dengan mencampurkan S dan L dalam 3 pas. Namun, karena biasanya panjang S dan L tidak sama (nilai t dan c tidak sama), maka array yang lebih panjang akan diproses 3 kali dan yang lain lebih dari 3 kali sesuai perbedaan panjangnya. Algoritmanya adalah sebagai berikut:

i = 0
j = 0
A = 0
B = 0
for 3 × max(c, 2r + 2) **times do**
    $S[i] = (S[i] + A + B) \lll 3$
    A = S[i]
    $L[i] = (L[j] + A + B) \lll 3$
    B = L[i]
    i = (i + 1) mod (2r + 2)
    j = (j + 1) mod c

Fungsi max(a,b) menghasilkan nilai yang terbesar antara a dengan b.

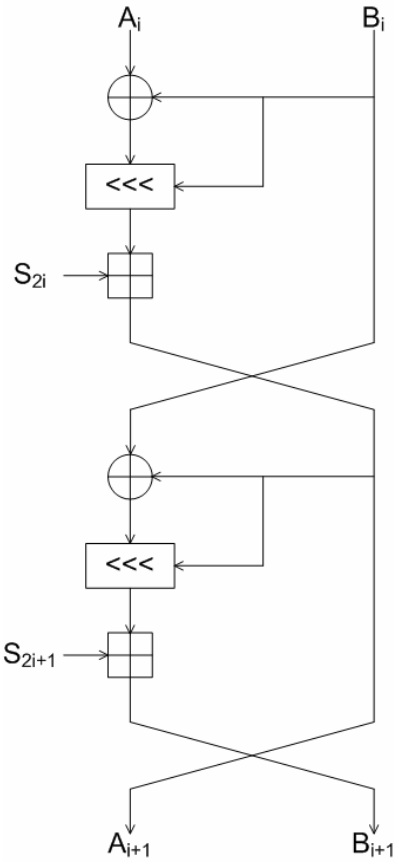
3.2.2 Algoritma Enkripsi

Fungsi enkripsi RC5 menerima input 1 blok sebesar yang terdiri dari 2 register sebesar w bit, A dan B. Untuk melakukan enkripsi diperlukan tabel kunci S[0..t – 1] yang telah diperoleh melalui proses sebelumnya (2.2.1). Algoritma enkripsinya adalah sebagai berikut:

A = A + S[0]
B = B + S[1]
for i = 1 **to** r **do**
    $A = ((A \oplus B) \lll B) + S[2i]$
    $B = ((B \oplus A) \lll A) + S[2i + 1]$

Hasil dari enkripsi 1 blok terdapat dalam kedua register yaitu A dan B. 1 blok *ciphertext* adalah gabungan dari kedua register A dan B, yaitu AB.

Diagram berikut menggambarkan proses enkripsi 1 ronde dengan algoritma RC5.



Gambar 1. Enkripsi 1 putaran dalam RC5

3.2.3 Algoritma Dekripsi

Fungsi dekripsi merupakan penurunan dari fungsi enkripsi. Algoritmanya adalah sebagai berikut:

for i = r **downto** 1 **do**
    $B = ((B - S[2 \times i + 1]) \ggg A) \oplus A$
    $A = ((A - S[2 \times i]) \ggg B) \oplus B$
B = B – S[1]
A = B – S[0]

4. RC6 Block Cipher

RC6 merupakan algoritma yang merupakan keturunan dari RC5 yang juga merupakan kandidat AES (Advanced Encryption Standard). Pada mulanya, perancangan RC6 diawali ketika RC5 dianggap dapat dijadikan kandidat untuk mengikuti kompetisi pemilihan AES. Modifikasi kemudian dibuat untuk meningkatkan keamanan dan performa dan juga untuk dapat memenuhi persyaratan AES.

RC6 dirancang untuk menghilangkan segala ketidakamanan yang ditemukan pada RC5, karena analisis pada RC5 menunjukkan bahwa ternyata jumlah rotasi yang terjadi pada RC5 tidak sepenuhnya bergantung pada data yang terdapat dalam blok. Selain itu, serangan kriptanalisis diferensial juga ternyata dapat menembus keamanan yang ditawarkan RC5.

RC6 juga dirancang untuk memenuhi persyaratan AES yang diantaranya adalah kemampuan untuk beroperasi pada mode blok 128 bit. Jika besar blok 128 bit langsung dipaksakan untuk diimplementasikan dengan algoritma RC5, maka akan dibutuhkan register kerja 64 bit. Spesifikasi arsitektur dan bahasa yang menjadi tempat implementasi algoritma yang ditentukan oleh AES belum mendukung pengoperasian 64 bit yang efisien. Oleh karena itu, daripada menggunakan 2 register 64 bit seperti pada RC5, RC6 menggunakan 4 register 32 bit. Karena menggunakan 4 register maka akan terdapat 2 operasi rotasi pada setiap *half-round* yang ada, dan juga akan lebih banyak bit-bit yang akan digunakan untuk mempengaruhi banyaknya bit yang dirotasi.

RC6 seperti juga RC5 mengeksploitasi penggunaan operasi-operasi primitif yang diimplementasikan secara efisien dalam prosesor-prosesor modern. RC6 juga selain menggunakan ketiga operasi primitif yang digunakan dalam RC5, juga menggunakan operasi perkalian 32-bit yang telah diimplementasikan secara efisien dalam prosesor modern saat ini. Primitif operasi perkalian ini sangat efektif dalam menghasilkan efek "*diffusion*" atau penyebaran yang tentu saja mengakibatkan RC6 lebih aman daripada RC5. Operasi perkalian ini digunakan untuk menghitung jumlah bit yang dirotasi sehingga konsep *data-dependent rotations* dapat dengan lebih sempurna diimplementasikan.

4.1.1 Operasi-operasi Primitif

Selain ketiga operasi primitif yang digunakan pada RC5, RC6 juga menggunakan operasi perkalian modulo-2^w. Berikut adalah daftar operator-operator primitif (termasuk invers) yang digunakan dalam RC6:

$a + b$	penjumlahan integer modulo 2^w
$a - b$	pengurangan integer modulo 2^w
$a \oplus b$	operasi eksklusif-OR (xor) dari w-bit *word*
$a \times b$	perkalian integer modulo 2^w
$a \lll b$	rotasi terhadap a ke kiri sebanyak nilai yang diperoleh dari bit-bit orde bawah sejumlah $^2\log(w)$ dari b
$a \ggg b$	rotasi terhadap a ke kanan sebanyak nilai yang diperoleh dari bit-bit orde bawah sejumlah $^2\log(w)$ dari b

4.1 Algoritma RC6

Algoritma RC6 seperti juga RC5 merupakan algoritma *cipher* yang terparameterisasi. RC6 secara tepat ditulis sebagai:

RC6 – w / r / b

Nilai parameter w, r, dan b menyatakan hal yang sama seperti yang ditunjukkan dalam algoritma RC5. Algoritma RC6 yang dipakai sebagai kandidat AES adalah RC6-32/20/b, yang berarti ukuran *word* 32 bit, jumlah ronde 20 kali, dengan panjang kunci b ditentukan pengguna.

4.1.1 *Key Expansion Algorithm*

Algoritma untuk membangkitkan kunci internal sama seperti pada RC5. Nilai konstanta P_w dan Q_w yang digunakan juga sama, tetapi ukuran array S tidak sama dengan yang seperti RC5. Ukuran t dari array S dalam RC6 adalah t = 2(r+2), yang berarti terdapat lebih banyak kunci internal yang dibangkitkan daripada jumlah kunci internal RC5. Berikut algoritmanya:

```
S[0] = Pw
for i = 1 to (2r + 3) do
    S[i] = S[i – 1] + Qw

i = 0
j = 0
A = 0
B = 0
for 3 × max(c, (2r + 4)) times do
    S[i] = (S[i] + A + B) <<< 3
    A = S[i]
    L[i] = (L[j] + A + B) <<< 3
    B = L[i]
```

i = (i + 1) mod (2r+4)
j = (j + 1) mod c

4.1.2 Algoritma Enkripsi

Fungsi enkripsi menerima input 1 blok *plaintext* yang terbagi dalam 4 register yang masing-masing berupa w-bit *word*, yaitu A, B, C, dan D. *Ciphertext* hasil proses terbagi dan disimpan dalam A, B, C, dan D. Dalam proses enkripsi diperlukan tabel kunci S yang dianggap telah didapat dari proses sebelumnya.

Secara lebih detil, proses enkripsi dengan RC6 dapat dibagi dalam beberapa langkah. Dalam penjelasan berikut, notasi (A,B,C,D) = (B,C,D,A) berarti adalah operasi *assignment* yang dilakukan paralel (bersamaan) untuk setiap elemen di ruas kanan ke ruas kiri yang berkorespondensi. Langkah-langkahnya adalah sebagai berikut:

1. Mula-mula lakukan *half-round loop* yang seperti pada RC5:

   for i = 1 **to** r **do**
   A = ((A ⊕ B) ⋘ B) + S[i]
   (A,B) = (B,A)

2. Lakukan dua proses RC5 secara paralel, yang satu untuk register A, B dan yang lain untuk register C, D.

   for i = 1 **to** r **do**
   A = ((A ⊕ B) ⋘ B) + S[2i]
   C = ((C ⊕ D) ⋘ D) + S[2i+1]
   (A,B) = (B,A)
   (C,D) = (D,C)

3. Pada tahap pertukaran, daripada menukar A dengan B, dan C dengan D, lakukan permutasi antar keempat register (A,B,C,D) = (B,C,D,A), sehingga komputasi AB bercampur dengan komputasi CD.

   for i = 1 **to** r **do**
   A = ((A ⊕ B) ⋘ B) + S[2i]
   C = ((C ⊕ D) ⋘ D) + S[2i+1]
   (A,B,C,D) = (B,C,D,A)

4. Campurkan komputasi AB dengan CD lebih jauh, yaitu dengan mempertukarkan kedua nilai yang menyatakan jumlah rotasi pada masing-masing komputasi.

   for i = 1 **to** r **do**
   A = ((A ⊕ B) ⋘ D) + S[2i]
   C = ((C ⊕ D) ⋘ B) + S[2i+1]
   (A,B,C,D) = (B,C,D,A)

5. Daripada menggunakan nilai B dan D secara langsung, RC6 menggunakan hasil transformasi kedua register ini. Hal ini dilakukan untuk tidak mengulangi masalah rotasi seperti pada RC5 di mana tidak seluruh bit dalam data yang berpengaruh dalam rotasi. Oleh karena itu, fungsi transformasi yang dipilih harus dapat memanfaat seluruh bit di dalam data untuk mengatur jumlah bit yang dirotasikan. Fungsi yang dipilih adalah $f(x) = x(2x + 1) \pmod{2^w}$ yang kemudian diikuti dengan rotasi ke kiri sebanyak 5 bit. Transformasi ini terpilih karena fungsi f(x) yang merupakan fungsi satu-ke-satu memiliki bit-bit orde atas yang menentukan jumlah rotasi yang akan digunakan yang sangat bergantung pada x.

   for i = 1 **to** r **do**
   p = (B × (2B + 1)) ⋘ 5
   q = (D × (2D + 1)) ⋘ 5
   A = ((A ⊕ p) ⋘ q) + S[2i]
   C = ((C ⊕ q) ⋘ p) + S[2i+1]
   (A,B,C,D) = (B,C,D,A)

6. Setelah *loop* di atas selesai, akan terdapat hasil di mana *plaintext* bisa menunjukkan bagian input ronde pertama dalam enkripsi dan *ciphertext* bisa menunjukkan bagian input ronde terakhir dalam enkripsi. Oleh karena itu perlu ditambahkan langkah-langkah di awal dan di akhir *loop* untuk menyamarkan hubungan ini. Sehingga, terbentuklah algoritma enkripsi RC6 yang sebagai berikut:

   B = B + S[0]
   D = D + S[1]
   for i = 1 **to** r **do**
   p = (B × (2B + 1)) ⋘ 5
   q = (D × (2D + 1)) ⋘ 5
   A = ((A ⊕ p) ⋘ q) + S[2i]
   C = ((C ⊕ q) ⋘ p) + S[2i+1]
   (A,B,C,D) = (B,C,D,A)
   A = A + S[2r + 2]
   C = C + S[2r + 3]

Perlu diketahui juga, dalam varian baru RC6 jumlah rotasi ke kiri yang mengikuti fungsi kuadrat bukan 5 bit tetapi adalah $^2\log(w)$ bit.

4.1.3 Algoritma Dekripsi

Sama seperti pada RC5, algoritma dekripsi RC6 juga merupakan penurunan dari algoritma enkripsi. Algoritmanya sebagai berikut:

```
C = C – S[2r + 3]
A = A – S[2r + 2]
for i = r downto 1 do
    (A,B,C,D) = (D,A,B,C)
    p = (D × (2D + 1)) <<< 5
    q = (B × (2B + 1)) <<< 5
    C = ((C – S[2i + 1]) >>> q) ⊕ p
    A = ((A – S[2i]) >>> p) ⊕ q
D = D – S[1]
B = B – S[0]
```

5. Perbandingan RC5 dengan RC6

Berdasarkan penjelasan mengenai algoritma RC5 dan RC6 di atas, dapat diketahui beberapa perbedaan mendasar antara RC6 dengan RC5.

1. RC6 menggunakan 4 register berukuran b/4-bit, sedangkan RC5 menggunakan 2 register berukuran b/2-bit, dengan b menyatakan panjang blok. RC6 menggunakan 4 register karena dirancang untuk memenuhi spesifikasi AES, yaitu kemampuan beroperasi dalam mode 128 bit. Dengan 4 register kerja, maka besar masing-masing register kerja yang diperlukan hanya 32 bit (4 × 32 = 128 bit)

2. RC6 memiliki operasi primitif tambahan, yaitu perkalian integer. Operasi perkalian ini digunakan untuk meningkatkan penyebaran bit yang dicapai setiap ronde. Hal ini tentunya berakibat pada tidak diperlukannya banyak ronde untuk mencapai tingkat penyebaran bit yang optimal, sehingga meningkatkan kecepatan pemrosesan.

Namun, selain perbedaan di atas terdapat juga perubahan yang signifikan pada RC6 dibandingkan dengan RC5, yaitu digunakannya fungsi kuadrat dalam menentukan jumlah rotasi. Maka,

3. RC6 menggunakan fungsi transformasi $f(x) = x(2x + 1) \pmod{2^w}$ yang kemudian diikuti oleh rotasi ke kiri sebanyak 5 bit untuk menentukan banyaknya bit yang akan dirotasi dalam setiap ronde.

Pemilihan fungsi transformasi ini adalah demi memberikan RC6 ketahanan terhadap serangan kriptanalisis diferensial. Fungsi kuadrat yang digunakan dapat memberikan tingkat penyebaran bit yang lebih cepat yang oleh karena itu akan memungkinkan RC6 menggagalkan proses menemukan karakteristiknya. Selain itu, dengan menggunakan fungsi kuadrat, maka RC6 dapat mengurangi sifat linearitas yang dapat dieksploitasi dalam kriptanalisis linear. Selain itu, rotasi sebanyak 5 bit yang mengikuti fungsi kuadrat juga memiliki peran dalam meningkatkan kompleksitas penyerangan RC6 dengan kriptanalisis diferensial dan kriptanalisis linear.

6. Keamanan RC5 dan RC6

Seperti telah ditunjukkan, RC5 dan RC6 adalah algoritma *cipher* yang terparameterisasi. Oleh karena itu, keamanan algoritma ini sangat bergantung pada besaran-besaran yang dimasukkan dalam parameter. Salah satu elemen yang sama antara RC5 dan RC6 adalah algoritma pembangkitan kunci internal (*key expansion algorithm*). Keamanan algoritma ini dapat dinilai dengan ada tidaknya kunci lemah (*weak keys*). Kunci lemah [1] adalah kunci yang menyebabkan tidak adanya perbedaan antara enkripsi dan dekripsi. Dekripsi terhadap *ciphertext* tetap menghasilkan *plaintext* semula, tetapi enkripsi dua kali berturut-turut terhadap *plaintext* akan menghasilkan kembali *plaintext*-nya. Sejak publikasi RC5, belum ada laporan mengenai adanya kunci lemah dalam algoritma ini. Karena RC6 juga menggunakan algoritma pembangkitan kunci internal yang sama, maka dapat dibilang bahwa sejauh ini RC5 dan RC6 tidak memiliki kunci lemah.

Penilaian terhadap keamanan, dapat dilihat dari tingkat kesulitan penyerangan terhadap algoritma. Banyak jenis serangan yang dapat dilakukan terhadap algoritma *cipher* (Poin 1). Serangan terbaik tetapi juga yang paling memakan usaha terhadap algoritma *cipher* adalah dengan *exaustive key search*. Seperti telah dijelaskan sebelumnya, performa serangan ini sangat bergantung pada panjang key yang ada, maka sebagai contoh RC5-w/r/b dengan domain nilai b adalah 0 sampai 255, dengan tabel kunci RC5 dengan jumlah ronde R memiliki $2^{(2r + 2) w}$ bit untuk ukuran blok sebesar 2w. Maka usaha yang diperlukan untuk melakukan serangan ini adalah $\min(2^{8b}, 2^{(2r + 2) w})$. Oleh karena panjang kunci rahasia dan jumlah ronde cukup besar, RC5 aman dari serangan ini. Demikian juga dengan RC6, yang malahan memiliki jumlah kunci internal yang lebih banyak. Tentu saja lebih aman.

Serangan dengan pengujian statistik terhadap RC5 dan RC6 cukup sulit untuk dilakukan. Hal ini disebabkan karena fitur kedua algoritma ini yang menggunakan *data-dependent rotations* yang mengakibatkan penyebaran bit-bit yang tidak dapat diprediksi. Hal ini tentunya menyebabkan korelasi antara *plaintext*, *ciphertext*, dan kunci menjadi sama sekali sulit untuk ditemukan.

Serangan yang paling umum dilakukan oleh para peneliti dan kriptanalis untuk menganalisis RC5 dan RC6 adalah dengan menggunakan metode kriptanalisis diferensial dan metode kriptanalisis linear.

Serangan terhadap RC5

Hasil kriptanalisis terhadap RC5 pertama sekali dihasilkan oleh Kaliski dan Yin [3] pada tahun 1995. Dengan melakukan analisis terhadap struktur dasar algoritma enkripsi RC5 dan juga sifat-sifat *data-dependent rotations* mereka dapat membangun karakteristik diferensial dan aproksimasi linear dari RC5 yang berguna untuk melancarkan serangan diferensial dan linear. Hasil mereka menunjukkan bahwa penggunaan *data-dependent rotations* dan inkompatibilitas antara operasi-operasi aritmatik yang berbeda membantu pencegahan kedua jenis serangan tersebut.

Knudsen dan Meier berhasil menunjukkan perbaikan terhadap serangan diferensial Kaliski dan Yin dengan melakukan analisis terhadap relasi antara input, output, dan kunci internal dalam dua ronde yang pertama. Walaupun karakteristik yang mereka gunakan pada dasarnya sama dengan yang digunakan oleh Kaliski dan Yin, mereka berhasil memperbaiki kebutuhan *plaintext* dengan mengeksploitasi daftar karakteristik dengan lebih baik pada awal ronde dan pada akhir ronde ke r. Mereka juga menunjukkan adanya *weak-key* yang potensial yang memungkinkan peningkatan serangan.

Kaliski dan Yin selanjutnya juga mempelajari lebih jauh bagaimana *data-dependent rotations* dalam suatu ronde dapat menyebarkan perbedaan kecil dalam input tetapi menyebarkan perbedaan besar dalam output. Sifat yang demikian menyebabkan serangan diferensial menjadi tidak mungkin untuk RC5 yang memiliki jumlah ronde yang banyak.

Pada tahun 1998, Biryukov dan Kushilevitz menunjukkan perbaikan yang lebih baik terhadap serangan yang dilakukan oleh Knudsen dan Meier sebelumnya. Mereka mempelajari diferensial yang lebih kompleks dan mendefinisikan pasangan input/output yang baik (*good pairs*) secara lebih umum dengan memperhitungkan *data-dependent rotations*. Mereka memperkirakan RC5 dengan 12 ronde dengan ukuran blok 64 bit (RC5-32/12/b) bisa diserang dengan menggunakan 2^{44} *plaintext*. Hal ini menunjukkan bahwa RC5-32/12/b memiliki tingkat keamanan yang (hampir) sama dengan DES.

Namun, berbeda dengan serangan diferensial, RC5 cukup tahan dari serangan linear. Hal ini disebabkan karena RC5 memiliki operasi-operasi primitif yang diatur sedemikian rupa sehingga mempersulit dapatnya aproksimasi linear.

Tabel berikut ini menunjukkan jumlah *plaintext* yang dibutuhkan untuk melancarkan serangan diferensial terhadap RC5.

Ronde	4	6	8	10
Chosen plaintext	2^{7}	2^{16}	2^{28}	2^{36}
Known plaintext	2^{36}	2^{41}	2^{47}	2^{51}

Ronde	12	14	16	18
Chosen plaintext	2^{44}	2^{52}	2^{61}	>>
Known plaintext	2^{55}	2^{59}	2^{63}	>>

Serangan terhadap RC6

Serangan yang umum dan yang masih dilakukan untuk menguji terhadap RC6 adalah serangan diferensial dan linear. Berikut adalah tabel kebutuhan *plaintext* untuk melancarkan serangan diferensial dan linear terhadap RC6 (dengan panjang blok 64 bit).

	Jumlah ronde				
Att.	8	12	16	20	24
Dif.	2^{56}	2^{117}	2^{190}	2^{238}	2^{299}
Lin.	2^{47}	2^{83}	2^{119}	2^{155}	2^{191}

Dapat dilihat berdasarkan kebutuhan *plaintext*, bahwa RC6 jauh lebih kuat daripada RC5. Jumlah ronde yang dianjurkan untuk menggunakan RC6 adalah 20 ronde, karena pada ronde tersebut jumlah kebutuhan *plaintext* sudah melebihi 2^{128} untuk kedua jenis serangan.

RC6 memberikan pertahanan yang cukup baik terhadap serangan kriptanalisis diferensial [5].

Dengan digunakannya fungsi kuadrat yang diikuti dengan rotasi bit sejumlah tetap, RC6 benar-benar menyebabkan pembangunan diferensial yang efektif untuk melancarkan serangan diferensial menjadi sulit. Selain itu, properti RC6 yang demikian juga mengakibatkan sulitnya untuk membangun aproksimasi linear yang baik untuk mencarkan serangan linear yang efektif.

7. Kesimpulan

Kesimpulan yang dapat ditarik setelah pembahasan RC5 dan RC6 dalam makalah ini adalah:

1. Algoritma enkripsi RC5 dan RC6 merupakan algoritma enkripsi yang cukup aman dan memiliki performa tinggi oleh karena penggunaan operasi-operasi primitif prosesor yang telah diimplementasikan secara efisien pada zaman sekarang.

2. Algoritma enkripsi RC5 dan RC6 merupakan algoritma yang terparameterisasi. Hal ini memungkinkan RC5 dan RC6 untuk berubah secara dinamis sehingga dapat memenuhi kebutuhan penggunaan. Penggunaan parameter benar-benar memberikan fleksibilitas kepada pengguna untuk memilih *trade-off* antara performa dan keamanan. Parameterisasi juga memungkinkan RC5 dan RC6 untuk menjadi semakin aman dengan meningkatkan jumlah ronde.

3. Serangan-serangan yang umum terhadap algoritma kriptografi kunci simetri adalah serangan *brute-force*, pengujian statistik, kriptanalisis diferensial, dan kriptanalisis linear.

4. Daya tahan RC5 dan RC6 terhadap serangan *brute-force* dan pengujian statistik, kurang lebih sama. Hal ini disebabkan karena algoritma RC5 dan RC6 keduanya menggunakan fitur *data-dependent rotations*, yang memungkinkan tingkat penyebaran bit yang tinggi. Yang berbeda di antara keduanya hanyalah tingkat penyebaran bitnya saja di mana RC6 memiliki kemampuan penyebaran bit yang lebih tinggi daripada RC5.

5. Daya tahan RC6 terhadap serangan kriptanalisis diferensial dan serangan kriptanalisis linear lebih besar daripada RC5. Hal ini disebabkan karena RC6 telah berhasil menerapkan fitur *data-dependent rotations* secara lebih baik daripada RC5, di mana RC5 ternyata tidak berhasil memanfaatkan keseluruhan bit-bit input untuk menentukan jumlah rotasi. Selain itu, RC6 juga menggunakan fungsi transformasi yang mengimplementasikan fungsi kuadrat, yang menyebabkan RC6 dapat menyebabkan tingginya tingkat penyebaran data dan sulitnya pembentukan karakteristik serta aproksimasi linear yang efektif untuk melancarkan serangan diferensial dan linear.

Daftar Pustaka

[1] Munir, Rinaldi. (2006) *Diktat Kuliah IF5054 Kriptografi*. Institut Teknologi Bandung.

[2] Rivest, R. L. (1997) *The RC5 Encryption Algorithm*. MIT Laboratory for Computer Science.

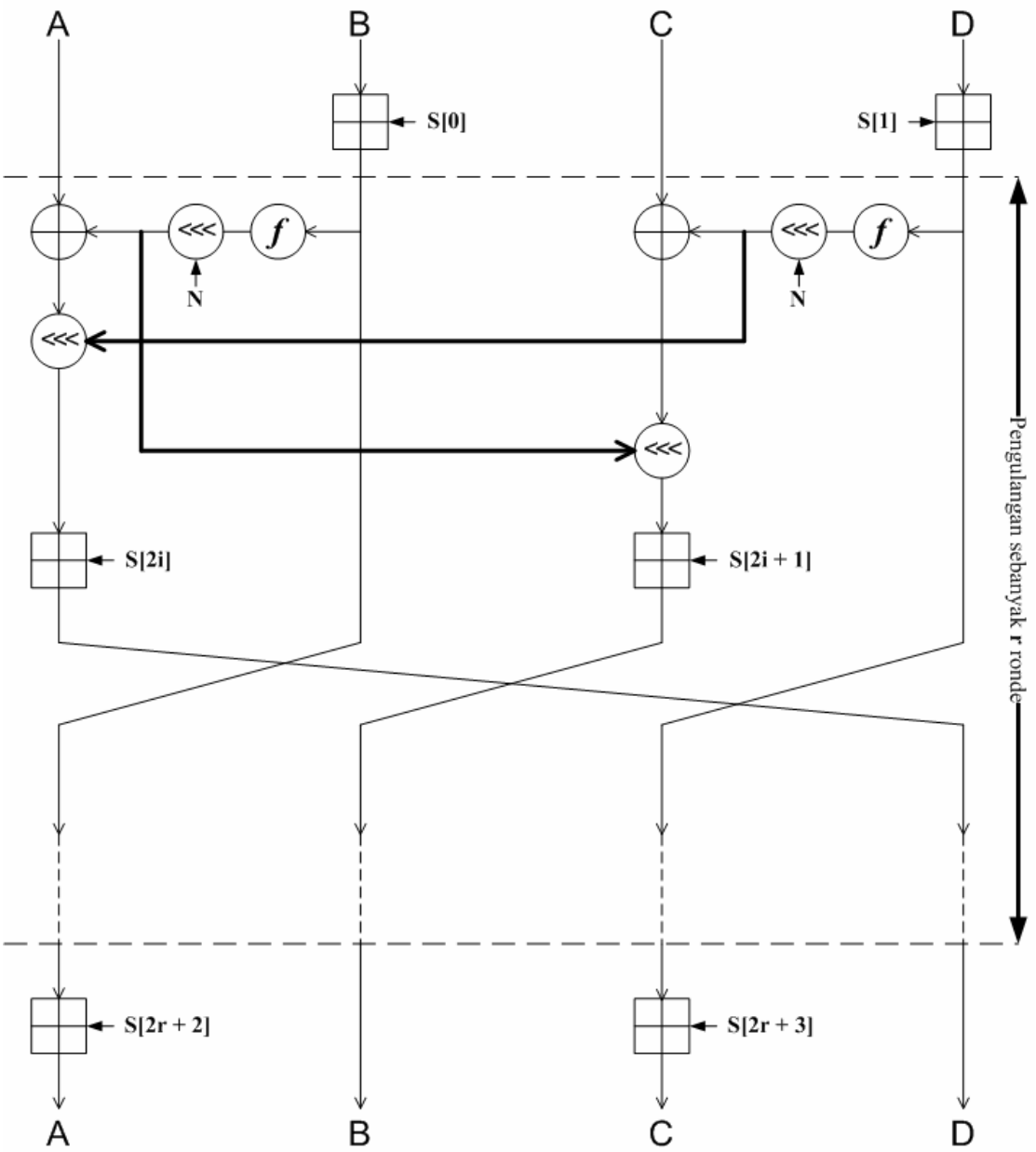
[3] Kaliski, B. S., Yin, Y. L. (1998) *On the Security of the RC5 Encryption Algorithm*. RSA Laboratories.

[4] Rivest, R. L., Robshaw M.J.B, Sydney, R., Yin, Y. L. (1998) *The RC6™ Block Chiper*. RSA Laboratories.

[5] Contini, S., Rivest, R. L., Robshaw, M.J.B, Yin, Y. L. (1998) *The Security of RC6™ Block Chiper*. RSA Laboratories.

Apendiks

Berikut ditampilkan diagram yang menjelaskan proses enkripsi dengan RC6. Pada diagram ini, *f* adalah fungsi kuadrat, dengan $f(x) = x(2x + 1) \pmod{2^w}$. Nilai N menyatakan jumlah bit yang dirotasi ke kiri yang operasinya dilakukan setelah fungsi *f*. Nilai N adalah sebesar 5 bit atau $^2\log(w)$ bit (tergantung pada varian).



Gambar 2. Enkripsi dalam RC6