KAJIAN MENGENAI *DIFFERENTIAL AND LINEAR ATTACKS* TERHADAP RC5

Irma Juniati – NIM : 13506088

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16088@students.if.itb.ac.id

Abstrak

Algoritma enkripsi RC5 pertama kali dipublikasikan pada Desember 1994. Sejak saat itu, banyak studi yang dilakukan terhadap algoritma tersebut, yang salah satunya akan dikaji pada makalah ini, yakni tingkat keamanan algoritma RC5 terhadap *differential attack* dan *linear attack*. Makalah ini akan mengkaji lebih dalam tentang tingkat keamanan algoritma RC5, mulai dari deskripsi singkat tentang algoritma RC5 dan fiturnya, *differential analysis* terhadap RC5, langkah apa saja yang dilakukan, sejauh mana analisis dapat dilakukan, serta perbandingannya jika serangan dilakukan dengan *linear analysis*, juga dengan *exhaustive search attack*. Dengan demikian, pada akhirnya dapat diambil kesimpulan mengenai tingkat keamanan dari algoritma RC5 ini. Selain itu juga ditambahkan analisis mengenai kekuatan dan kelemahan dari beberapa varian RC5.

***Kata kunci**:* RC5, *differential attack, differential cryptanalysis, linear attack, linear cryptanalysis*

1. PENDAHULUAN

RC5 merupakan salah satu pengembangan algoritma enkripsi *block cipher* yang cukup sederhana namun memiliki tingkat keamanan yang memadai. Salah satu pengembangannya adalah RC6, yang dikembangkan dengan berdasar pada algoritma RC5.

Serangan terhadap kriptografi dapat dibedakan menjadi enam metode. Dua di antaranya akan diuraikan dalam makalah ini, yaitu *differential* dan *linear attack*.

Differential cryptanalysis termasuk *chosen plaintext attack* yang dilakukan dengan mencari keterhubungan antar *ciphertext* yang dihasilkan oleh dua *plaintext* yang berhubungan. Metode serangan ini melakukan analisis statistik dari dua *input* dan dua *output* yang dihasilkan oleh sebuah algoritma enkripsi.

Sedangkan *linear cryptanalysis* termasuk *known plaintext attack* yang membutuhkan pasangan *plaintext* dan *ciphertext* dalam jumlah besar, yang dienkripsi dengan kunci yang tidak diketahui. Tipe serangan ini fokus kepada analisis statistik satu putaran dekripsi dengan *ciphertext* dalam jumlah besar.

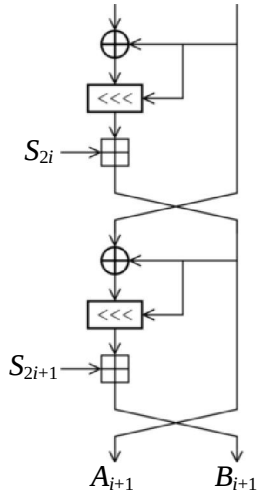
2. ISI

2.1 Deskripsi dan Fitur RC5

Berbeda dengan skema-skema lainnya, RC5 memiliki tiga parameter, yaitu ***w***–*word size* (32, 64, atau 128 bit), ***b***–*key size* (0–255); jumlah *byte* pada kunci, dan ***r***–*round* (0–255).

RC5 terdiri atas tiga komponen, yaitu algoritma *key expansion*, algoritma enkripsi, dan algoritma dekripsi. Ketiga komponen ini menggunakan tiga operasi primitif, yaitu:

- Penjumlahan *words* modulo 2^w, dinyatakan dengan "+".
- *XOR* bit dari *words*, dinyatakan dengan ⊕.
- Rotasi: rotasi x ke kiri sebanyak y dinyatakan dengan "x <<< y".
  Catatan: hanya $\log_2(w)$ *low-order* bit dari y yang mempengaruhi rotasi ini.



Gambar 1 Enkripsi (1 putaran) RC5

Key Expansion

Algoritma pembangkitan kunci ini melakukan ekspansi terhadap kunci eksternal *K* dari *user*, menyimpan hasilnya ke dalam tabel *S*, sehingga ukuran *S* adalah 2(*r* + 1). Algoritma ini menggunakan dua "*magic constants*" dan terdiri atas tiga langkah algoritmik sederhana.

"*magic constants*" P_w dan Q_w didefinisikan sebagai berikut:

$$P_w = \text{Odd}((e - 2)\, 2^w)$$
$$Q_w = \text{Odd}((\phi - 2)\, 2^w)$$

di mana

e = 2.718281828459… (basis logaritma natural),

ϕ = 1.618033988749… (*golden ratio*),

dan Odd(*x*) mengembalikan bilangan *integer* ganjil yang terdekat dengan nilai *x*.

Tiga langkah algoritmik yang dimaksud adalah

- Menyalin kunci rahasia *K*[0, …, *b*–1] ke dalam senarai *L*[0, …, *c*–1], di mana $c = \lceil b / u \rceil$, $u = w / 8$ (*u* adalah jumlah *byte*s/*word*). Posisi *byte* yang tidak terisi di *L* diisi dengan 0. Untuk kasus $b = c = 0$, kita isi $c = 1$ dan $L[0] = 0$.

- Inisialisasi senarai *S* dengan pola bit yang *pseudo-random*, menggunakan dua *magic constants* yang telah didefinisikan.

```
S[0] = Pw;
for i = 1 to t – 1 do
  S[i] = S[i – 1] + Qw;
```

- Mengombinasikan kunci rahasia masukan *user* dalam tiga kali *pass* ke senarai *S* dan *L*. Karena ukuran *S* dan *L* yang berbeda, maka senarai yang lebih besar akan diproses tiga kali, sedangkan senarai lainnya akan ditangani lebih banyak kali. Berikut algoritmanya:

```
i = j = 0;
A = B = 0;
do 3 * max(t,c) times:
  A = S[i] = (S[i] + A + B) <<< 3;
B = L[j] = (L[j] + A + B) <<< (A +
B);
i = (i + 1) mod(t);
j = (j + 1) mod(c);
```

Enkripsi dan Dekripsi

Enkripsi dalam RC5 dijabarkan dalam *pseudo-code* berikut. Asumsikan blok masukan diberikan dalam *w*-bit *register A* dan *B*, dan keluarannya juga disimpan dalam *register A* dan *B*.

```
A = A + S[0];
B = B + S[1];
for i = 1 to r do
  A = ((A ⊕ B) <<< B) + S[2i];
  B = ((B ⊕ A) <<< A) + S[2i + 1];
```

Untuk dekripsi, dapat diturunkan dari algoritma enkripsinya, yaitu sebagai berikut.

```
for i = r downto 1 do
  B = ((B – S[2i + 1]) >>> A) ⊕ A
  A = ((A – S[2i]) >>> B) ⊕ B
B = B – S[1];
A = B – S[0];
```

Fitur dari algoritma RC5 ini dapat diuraikan sebagai berikut:

RC5 merupakan *block cipher* yang dirancang untuk implementasi perangkat lunak maupun keras. Dengan tiga variabel yang dimiliki (*w, r,* dan *b*), RC5 memberikan fleksibilitas yang tinggi baik dalam performansi maupun tingkat keamanannya.

Salah satu fitur RC5 yang paling signifikan adalah kesederhanaan; enkripsi hanya didasarkan pada tiga operasi (penjumlahan, *XOR*, dan rotasi). Oleh karena itu, RC5 menjadi mudah diimplementasikan. Selain itu, RC5 juga menggunakan rotasi yang bersifat *data-dependent* di dalam enkripsinya, yang sangat berpengaruh dalam mencegah *differential* dan *linear cryptanalysis*.

2.2 Serangan terhadap RC5

Dalam melakukan serangan terhadap RC5, kita dapat mencoba untuk menemukan kunci asli yang bersifat rahasia, atau menggunakan tabel *S* yang telah didenisikan sebelumnya. Dengan demikian, serangan bergantung pada panjang dari kunci rahasianya.

2.2.1 *Differential Attack* terhadap RC5

Ide dasar dalam teknik ini adalah pemilihan dua *plaintext* dengan adanya "perbedaan" P' antara keduanya. Pada umumnya, 'perbedaan'-nya diukur dengan eksklusif-OR ( ), 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.

Half-round Characteristics

Secara intuitif, jika pasangan *input* dari setengah putaran memiliki jumlah rotasi yang berbeda, pasangan *output* yang dihasilkan tentu akan memberikan hasil yang sangat variatif. Oleh sebab

itu, kita akan fokus pada pasangan dengan jumlah putaran yang sama.

Karakteristik dinyatakan dengan $\Omega = (\Omega_P, \Omega_R)$, di mana

$\Omega_P = (L'_{i-1}, R'_{i-1}) = (L_{i-1} \oplus L^*_{i-1}, R_{i-1} \oplus R^*_{i-1})$,
$\Omega_T = (L'_i, R'_i) = (L_i \oplus L^*_i, R_i \oplus R^*_i)$.

Tabel berikut berisi *half-round characteristics* yang akan digunakan dalam *differential attack*. Ketika menghitung probabilitas, gunakan fakta bahwa untuk setiap *input* yang acak (x dan y), dengan $x \oplus y = e_s$ dan kunci *random* S_i, probabilitas $(x + S_i) \oplus (y + S_i) = e_s$ adalah minimal ½.

Ω	Ω_P	Ω_T	conditions	probability
Ω^1	$(0, e_s)$	(e_s, e_s)	$s \geq \lg(w)$	$p \geq \frac{1}{w} \cdot \frac{1}{2}$
Ω^2	(e_s, e_s)	$(e_s, 0)$	$s \geq \lg(w)$	$p = 1$
Ω^3	$(e_s, 0)$	$(0, e_t)$	$s, t \geq \lg(w)$	$p \geq \frac{1}{w} \cdot \frac{1}{2}$
Ω^4	$(0, e_s)$	(e_s, e_t)	$s, t \geq \lg(w), t \neq s$	$p \geq \frac{1}{w} \cdot \frac{1}{2}$
Ω^5	(e_s, e_t)	$(e_t, e_u \oplus e_v)$	$s, t \geq \lg(w), t \neq s, u > v$ $t - s = \pm(u - v) \bmod w$	$p \geq \frac{1}{w} \cdot \frac{1}{2} \cdot \frac{1}{2}$

Tabel 1 Karakteristik untuk *single half-round*

Terdapat tiga karakteristik dengan probabilitas 1:

- $\Omega^{1'}$: $\Omega_P = \Omega_T = (0, e_{w-1})$, yang bisa digabung dengan Ω^1
- $\Omega^{2'}$: $\Omega_P = \Omega_T = (e_{w-1}, e_{w-1})$, yang bisa digabung dengan Ω^2
- $\Omega^{3'}$: $\Omega_P = \Omega_T = (e_{w-1}, 0)$, yang bisa digabung dengan Ω^3

Sebuah *right pair* yang terkait dengan Ω_n terdiri atas dua *plaintext P*, *P** dan *ciphertext*-nya *C*, *C**, sedemikian rupa sehingga untuk semua $0 \leq i \leq n$, perbedaan yang muncul, yaitu (L'_i, R'_i) memiliki bentuk yang sama dengan salah satu karakteristik *half-round* di atas. *Right pair* inilah yang kemudian akan digunakan untuk menghitung *subkey* S_n. Berikut ini adalah *pseudo-code* untuk menghitung *subkey*.

```
for s = 0 to w – 1
  pilih pasangan plaintext/ciphertext
  (L0, R0)/(Ln, Rn) sehingga (b + Rn-1)
  mod w = s
  hitung Ln-1[b] dengan algoritma β
  if s = 0 then carry(0) = 0
  if s ≥ 1
    if Sn[s – 1..0] ≤ Rn[s – 1..0] then
      carry(s) = 0
    else
      carry(s) = 1
  Sn[s] = Ln-1[b] ⊕ Rn-1[b] ⊕ carry(s)
```

Karena *rigth pair* yang dihasilkan bisa berjumlah cukup banyak, maka kita perlu mereduksinya menjadi apa yang disebut dengan *good pair*. Secara formal, definisi *good pair* adalah dua *plaintext P*, *P** dan *ciphertext*-nya *C*, *C**, dengan perbedaan *input* dan *output* (*P'*, *C'*) yang memenuhi karakteristik yang sama dengan *right pair*.

Dengan semua informasi yang sudah diperoleh, sekarang kita dapat menghitung jumlah *good pair* yang dibutuhkan untuk melakukan serangan terhadap RC5 dengan tingkat keberhasilan yang cukup tinggi. Jika kita memperoleh $2w$ *good pairs*, maka rata-rata akan ada sejumlah $2w \lg(w) / w = 2 \lg(w)$ *good pairs* yang berguna untuk memprediksi nilai setiap bit pada $S_n[s]$. Dengan probabilitas yang tinggi, lebih dari setengah *good pairs* adalah *right pairs*, sehingga $2w$ *good pairs* sudah cukup baik.

Tabel berikut menunjukkan jumlah *plaintext* yang dibutuhkan untuk melakukan *differential attack* terhadap RC5.

Round	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**
Chosen Plaintext	2^7	2^{16}	2^{28}	2^{36}	2^{44}	2^{52}	2^{61}	>>
Known Plaintext	2^{36}	2^{41}	2^{47}	2^{51}	2^{55}	2^{59}	2^{63}	>>

Tabel 2 Jumlah *round* untuk *differential attack*

Improvisasi *Differential Attacks* terhadap RC5

Beberapa perbaikan sudah dilakukan terhadap serangan tahap awal yang telah diuraikan, antara lain *Knudsen and Meier's attack*, *Biryukov and Kushilevitz's attack*.[3]

Batasan *Differential Attack* terhadap RC5

Batasan pada *differential attack* adalah kita hanya menggunakan *half-round characteristics* dengan pasangan *input* yang mempunyai jumlah rotasi yang sama, karena untuk jumlah putaran yang berbeda, akan dihasilkan *ciphertext* yang sangat bervariasi.

2.2.2 *Linear Attack* terhadap RC5

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. Caranya adalah dengan mencoba setiap 2^{32} *subkey* S_n yang mungkin, dengan mengambil satu *subkey* dengan nilai bias terbesar sebagai kunci yang benar.

Aproksimasi linier untuk *half-around* RC5

Aproksimasi linier untuk *half-round* disebut sempurna jika nilai bias = ½ (probabilitas 1 atau 0).

Dua persamaan dalam *half-round*:

$L_i = R_{i-1}$ (1)

$R_i = ((L_{i-1} \oplus R_{i-1}) <<< R_{i-1}) + S_i$ (2)

Untuk (1), terdapat banyak aproksimasi dengan probabilitas 1, salah satunya $L_i[0] = R_{i-1}[0]$.
Untuk (2), untuk menghasilkan aproksimasi yang baik, kita melakukan dekomposisi persamaan tersebut menjadi tiga bagian, yang masing-masing hanya mengandung satu operasi primitif.

$X = L_{i-1} \oplus R_{i-1}$ (3)

$Y = X <<< R_{i-1}$ (4)

$R_i = Y + S_i$ (5)

Nilai bias untuk persamaan (5) bergantung pada *subkey* S_i.

Analisis operasi tunggal

- **Operasi XOR**

  Persamaan $X = L_{i-1} \oplus R_{i-1}$ memiliki banyak aproksimasi linier yang sempurna. Semua aproksimasi yang melibatkan bit-bit yang sama untuk X, L_{i-1}, dan R_{i-1} adalah sempurna, sedangkan sisanya mempunyai nilai bias = 0.

- **Operasi Rotasi**

  Aproksimasi untuk rotasi dibedakan menjadi dua:

  (1) Tidak ada *bit* R_{i-1} yang terlibat
  Probabilitasnya adalah ½ + ½ w, karena untuk suatu jumlah rotasi, *bit-bit* yang terlibat dijamin sama, sedangkan untuk jumlah lainnya, probabilitasnya adalah ½ (*input* diasumsikan *random*).

  (2) Beberapa *bit* R_{i-1} terlibat
  Beberapa aproksimasi ini mempunyai nilai bias yang tidak 0. Misal $Y[0] = X[0] \oplus R_{i-1}[0]$, mempunyai probabilitas ½ + ½ w. Karena ketika jumlah rotasi = 0, $R_{i-1}[0] = 0$ dan $Y[0]=X[0]$, ketika jumlah rotasi bukan 0, probabilitas pada persamaan tersebut adalah ½.

- **Operasi Penjumlahan**

  Aproksimasi terbaik untuk persamaan $R_i = Y + S_i$ adalah $R_i = Y[0] + S_i[0]$, dengan probabilitas = 1 untuk semua *subkey* S_i.

Aproksimasi 1-*bit* linier

Dengan menggabungkan $X[0] = L_{i-1}[0] \oplus R_{i-1}[0]$, aproksimasi $Y[0] = X[0] \oplus R_{i-1}[0]$, dan $R_i = Y[0] + S_i[0]$, kita dapatkan $R_i[0] = L_{i-1}[0] + S_i[0]$.

Untuk *half-round* yang pertama, yang hanya melibatkan operasi penjumlahan, kedua aproksimasi berikut memiliki nilai probabilitas 1;
$L_1[0] = L_0[0] \oplus S_0[0]$ dan $R_1[0] = R_0[0] \oplus S_1[0]$.

Aproksimasi *multiple-bit* linier

Untuk operasi rotasi, kita mempertimbangkan aproksimasi yang tidak melibatkan *bit* R_{i-1}. Untuk t = 0, …, log(w), aproksimasi yang melibatkan sejumlah $k = 2^t$ *bit* dari X (dengan interval yang sama), dan 2^t *bit* dari Y (rotasi dari X) mempunyai nilai bias $2^t/2w$. Contohnya untuk w = 16 dan t = 2, aproksimasi untuk $X[0,4,8,12] = Y[1,5,9,13]$ mempunyai nilai bias = 1/8.

Untuk operasi penjumlahan, kita memilih operasi yang melibatkan *bit-bit* yang sama antara Y dan R_i sebagai aproksimasi untuk <<<. Contohnya $Y[1,5,9,13] = R_i[1,5,9,13]$.

Setelah kita memastikan aproksimasi untuk kedua operasi di atas, pilih aproksimasi untuk *XOR* yang sesuai. Misalnya:

$X[0,4,8,12] = L_{i-1}[0,4,8,12] \oplus R_{i-1}[0,4,8,12]$.

Semakin banyak jumlah *bit* k yang terlibat, nilai bias untuk <<< meningkat dan nilai bias rata-rata untuk operasi penjumlahan berkurang. Tabel berikut menunjukkan hasil percobaan yang dilakukan untuk *word-size*(w) = 4, 8, dan 16.

w = 4

k	Bias <<<	Avg. bias +	Total avg. bias
1	1/8	1/2	2/16
2	1/4	≤ 1/4	≤ 2/16
4	1/2	3/16	3/16

Tabel 3 Nilai bias untuk w = 4

w = 8

k	Bias <<<	Avg. bias +	Total avg. bias
1	1/16	½	$32/2^9$
2	1/8	≤ ¼	≤ $32/2^9$
4	1/4	≤ $27/2^8$	≤ $27/2^9$
8	1/2	$18/2^8$	$36/2^9$

Tabel 4 Nilai bias untuk w = 8

w = 16

k	Bias <<<	Avg. bias +	Total avg. bias
1	1/32	1/2	$2048/2^{16}$
2	1/16	≤ 1/4	≤ $2048/2^{16}$
4	1/8	≤ $4368/2^{16}$	≤ $1092/2^{16}$
8	1/4	≤ $1074/2^{16}$	≤ $537/2^{16}$
16	½	$608/2^{16}$	$608/2^{16}$

Tabel 5 Nilai bias untuk w = 16

Dari ketiga tabel di atas, tampak bahwa untuk *word-size* = 4, 8, aproksimasi yang melibatkan semua *w bit* mempunyai nilia bias rata-rata terbesar. Sedangkan untuk *w* = 16, aproksimasi yang melibatkan 1 *bit* mempunyai nilai bias rata-rata terbesar.

Batasan *Linear Attack*

Karena penggunaan kombinasi operasi pada RC5, aproksimasi linier yang baik dicegah untuk dapat tercapai. Dapat kita lihat dari bahasan sebelumnya bahwa operasi penjumlahan dan rotasi tidak *compatible* ketika kita mencoba melakukan aproksimasi linier untuk suatu *half-round* yang mempunyai nilai rata-rata bias terbesar; nilai bias untuk <<< semakin besar dan nilai rata-rata bias untuk + mengecil jika semakin banyak *bit* (*k*) yang terlibat.

Hasil percobaan memperlihatkan bahwa kriptanalisis linier standar hanya efektif untuk RC5 jika jumlah *round* (*r*) yang terlibat sangat kecil.

2.3 Perbandingan dengan Jenis Serangan Lain

Salah satu perbandingan yang dilakukan di sini adalah dengan *exhaustive search* terhadap RC5. Seperti yang telah kita ketahui, tingkat keamanan *block cipher* terhadap *exhaustive search* sangat bergantung pada ukuran kunci yang digunakan. Pada RC5, ukuran kunci sepanjang *b* dapat bervariasi dari 0 hingga 255 *bytes*, dan *key table* untuk RC5 dengan *r round*, untuk 2*w bit block-size* mempunyai ukuran $2^{(2r+2)w}$ *bit*.

Dengan demikian, jika menggunakan *exhaustive search attack* pada RC5-*w*/*r*/*b*, pencarian yang dilakukan minimal adalah sejumlah min(2^{8b}, $2^{(2r+2)w}$). Oleh karena itu, dengan kunci yang panjang dan jumlah *round* yang banyak, RC5 sangat amat terhadap serangan *exhaustive search*.

Selain itu, fleksibilitas dalam mengatur ketiga variabel dalam RC5 sangat memungkinkan bagi kita untuk melakukan *upgrade* terhadap tingkat keamanannya. Contohnya RC5 dengan panjang kunci 48 *bit* dapat dinaikkan hingga 88 *bit*. Seiring dengan perkembangan teknologi, keamanan RC5 dapat terus ditingkatkan.

2.4 Modifikasi dari RC5

Pada bagian ini, akan dibahas beberapa varian hasil modifikasi dari RC5 (yang dilakukan dengan mengubah sedikit operasi di dalamnya), dengan mempertimbangkan kekuatan dan kelemahan masing-masing.

2.4.1 RC5XOR

$$R_i = ((L_{i-1} \oplus R_{i-1}) <<< R_{i-1}) \oplus S_i$$

RC5XOR memiliki tingkat keamanan yang lebih rendah daripada RC5 dalam menghadapi serangan, baik *differential* maupun *linear attack*. Perubahan operasi penjumlahan menjadi *XOR* meningkatkan probabilitas *half-round characteristic* sekitar 2^t. Namun, keluaran yang dihasilkan oleh RC5XOR bagus digunakan sebagai langkah awal bagi seseorang untuk menganalisa RC5, karena RC5XOR masih tetap mempertahankan struktur dasar RC5, tetapi hanya membutuhkan *plaintext* yang tidak terlalu banyak.

2.4.2 RC5P

$$Ri = ((Li–1 + Ri–1) <<< Ri–1) + Si$$

RC5P melakukan perubahan terhadap operasi yang sebelumnya adalah *XOR* menjadi operasi penjumlahan. Perubahan ini sedikit mengurangi probabilitas sejumlah *half-round characteristic*, jika penghitungan perbedaan antar *output* yang dihasilkan dihitung dengan *XOR*. Namun, karena operasi penjumlahan dilakukan sebanyak dua kali, penghitungan perbedaan antar *output* dapat diukur secara sederhana dengan menggunakan pengurangan *integer*. Dengan demikian, dapat dibandingkan tingkat keamanan RC5P dan RC5XOR terhadap *differential attack*. Sedangkan untuk *linear attack*, RC5P dan RC5 memiliki tingkat keamanan yang setara.

2.4.3 RC5PFR

$$Ri = ((L_{i-1} \oplus R_{i-1}) <<< r_i) + S_i,$$

di mana r_i merupakan jumlah rotasi yang bersifat tetap. Nilai r_i dapat dipublikasikan sebagai parameter dari *cipher*.

Meskipun kedua jenis serangan (*differential* dan *linear*) tidak berjalan dengan baik pada varian RC5 ini karena nilai jumlah putaran yang tetap, RC5PFR bukan merupakan sebuah *cipher* yang kuat. Meskipun diberikan *input* yang berbeda-beda, satu-satunya ketidakpastian yang dihasilkan hanya terletak pada *carry*. Dengan demikian, akan terdapat karakteristik-karakteristik dengan nilai probabilitas yang cukup tinggi.

2.4.4 RC5KFR

$$R_i = ((L_{i-1} \oplus R_{i-1}) <<< r_i(K)) + S_i,$$

di mana $r_i(K)$ merupakan jumlah rotasi yang dilakukan, yang nilainya diturunkan dari kunci rahasia *K*. Dengan kata lain, nilai $r_i(K)$ bergantung pada kunci yang digunakan dan bersifat tetap.

Untuk varian RC5 ini, jika penyerang melakukan tebakan yang benar terhadap jumlah rotasi yang dilakukan, makan RC5KFR tereduksi menjadi RC5PFR. Jumlah tebakan yang mungkin adalah 2^{10r}, yang menjadi sangat tidak mungkin dilakukan untuk nilai *r* yang besar.

2.4.5 RC5RA

$$Ri = ((L_{i-1} \oplus R_{i-1}) <<< f(R_{i-1})) + S_i,$$

di mana $f(R_{i-1})$ bergantung pada semua *bit* R_{i-1} (tidak hanya bergantung pada lima *bit* yang paling tidak signifikan).

Karena semua *differential attacks* terhadap RC5 yang ada memiliki karakteristik bahwa pasangan *input* mempunyai jumlah rotasi yang sama, serangan yang sama akan menjadi lebih tidak efektif terhadap RC5RA. RC5RA sangat potensial untuk menjadi *cipher* yang kuat karena adanya tambahan fungsi tersebut.

3. KESIMPULAN

Dari uraian dalam keseluruhan makalah ini, dapat diambil beberapa kesimpulan:

- RC5 merupakan suatu bentuk *block cipher* yang beroperasi dengan cepat (dalam satuan *word*).
- RC5 sederhana; hanya terdiri atas tiga operasi utama yaitu penjumlahan, rotasi, dan *XOR*.
- RC5 fleksibel; memiliki tiga parameter yaitu *w*, *r*, dan *b* yang dapat divariasikan untuk memperkuat tingkat keamanan enkripsi atau mendapatkan tingkat keamanan sesuai kebutuhan.
- RC5 memiliki tingkat keamanan yang tinggi jika parameter yang dimasukkan sesuai.
- *Differential cryptanalysis* terhadap RC5 lebih mudah dilakukan dibandingkan dengan *linear cryptanalysis*. Hal ini disebabkan oleh RC5 yang memiliki operasi-operasi primitif yang diatur sedemikian rupa sehingga mempersulit penyerang untuk melakukan aproksimasi linier.

DAFTAR PUSTAKA

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

[2] Types of Cryptography Attacks – GIAC Research in the Common Body of Knowledge, http://www.giac.org/resources/whitepaper/cryptography/57.php

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

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