Studi dan Perbandingan Sistem Penyandian Pesan dengan Algoritma RC2, RC4, RC5 dan RC6

Rika Safrina – NIM : 13503053

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10 Bandung
2006
E-mail : if13053@students.if.itb.ac.id

Abstrak

Kunci Simetri (*Secret Key*) adalah salah satu algoritma kriptografi yang digunakan untuk melakukan dekripsi dan enkripsi, dengan menggunakan kunci rahasia untuk setiap bit (*Stream Cipher*) dan bit per blok (*Block Cipher*). Salah satu ilmuan kriptografi yang mendalami metoda ini adalah Ronald Linn Rivest dari Laboratorium RSA, suatu badan riset di bidang kriptografi dan sistem keamanan dari RSA Security Inc., sebuah organisasi komersil yang merupakan hasil merger antara Security Dynamic dan RSA Data Security yang didirikan oleh 3 penemu algoritma kunci publik (*Public Key*) RSA, Ron Rivest, Adi Shamir dan Leonard Adleman [RIV03]. Berkat penemuan RSA ini, Rivest mendapat penghargaan Turing Award dari ACM (Association for Computing Machinery) pada tahun 2002 [RSA05]. Empat buah algoritma Kunci Simetri yang ditemukannya adalah RC2, RC4, RC5, dan RC6. Adapun RC1 tidak pernah dipublikasikan dan RC3 berhasil dipecahkan saat dikembangkan di RSA Security (RC merupakan singkatan dari Ron's Code atau Rivest's Cipher).

RC2 adalah block cipher yang menggunakan variabel berupa ukuran kunci. Dengan mengubah-ubah ukuran kunci ini, performansi RC2 dapat menjadi dua atau tiga kali lebih baik dibanding DES (*Data Encryption Standard*). Sama halnya dengan RC2, RC4 juga menggunakan variabel berisi ukuran kunci. Hanya saja, algoritma ini menggunakan *Stream Cipher* dan digunakan secara luas pada sistem enkripsi keamanan seperti protokol SSL (*Secure Sockets Layer*).

Sedangkan RC6, yang merupakan perkembangan dari RC5, adalah golongan *Block Cipher* yang menggunakan tiga parameter, yaitu: ukuran blok (32, 64 atau 128), ukuran kunci primer (0-2048 bit), dan *round number* (0-255). Dua algoritma ini menyediakan berbagai format pengenkripsian suatu pesan.

Keempat algoritma di atas tidak seluruhnya dipatenkan, namun sudah diimplementasikan di banyak aplikasi kriptografi dengan segala kekurangan dan kelebihan yang dimilikinya masing-masing.

Kata kunci: Kunci Simetri, *Stream Cipher, Block Cipher*, RC2, RC4, RC5, RC6.

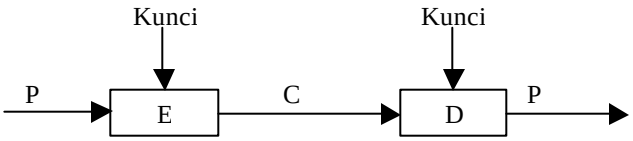
1. Pendahuluan

Kemajuan di bidang teknologi informasi telah memungkinkan institusi-institusi pendidikan atau komersil melakukan interaksi dengan konsumen melalui jaringan komputer. Kegiatan-kegiatan tersebut tentu saja akan menimbulkan resiko bilamana informasi yang sensitif dan berharga tersebut diakses oleh orang-orang yang tidak berhak (*unauthorized person*).

Bentuk ancaman yang dilakukan oleh penyusup dapat berupa ancaman pasif (*passive attack*) ataupun ancaman aktif (*active attack*). Ancaman pasif terjadi jika seorang penyusup secara sengaja mengambil, membaca dan menampilkan isi dari suatu pesan, Sedangkan ancaman aktif terjadi jika seorang penyusup tidak hanya mengetahui isi pesan melainkan juga memodifikasi atau bahkan memalsukan isi pesan dengan pesan yang baru [MEY82].

Untuk proteksi data yang cukup penting tidak ada jalan lain selain menggunakan metoda pengamanan pesan atau disebut juga teknik kriptografi. Secara umum, teknik kriptografi digunakan untuk melakukan penyandian pesan dan autentikasi pesan. Teknik penyandian pesan menekankan pada pencegahan terjadinya ancaman pasif, sedangkan teknik autentikasi pesan ditujukan untuk mencegah terjadinya ancaman aktif. Teknik yang dipelajari dalam studi ini adalah teknik penyandian pesan [SUG03].

Pada penyandian pesan, pesan awal yang ingin disandikan disebut plainteks. Jika plainteks digabung dengan suatu kunci yang dispesifikasikan khusus oleh *user*, pesan itu akan tersandi menjadi pesan yang tidak bermakna, atau disebut juga cipherteks. Adapun proses menyandikan pesan itu disebut enkripsi dan proses kebalikannya yaitu mengembalikan pesan tersandi menjadi semula disebut dekripsi. Algoritma yang mengkombinasikan plainteks dengan kunci sering disebut dengan *cipher*. Diagram proses kriptografi dapat dilihat pada Gambar 1.



Keterangan:
P : Plainteks
C : Cipherteks
E : Enkripsi
D: Dekripsi

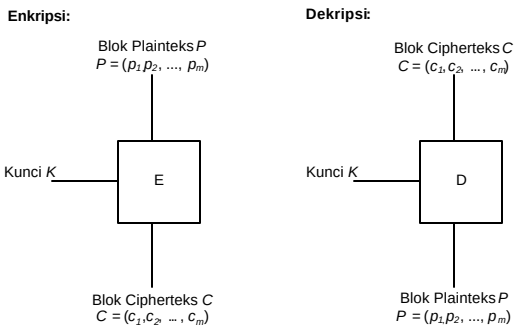
Gambar 1. Proses Enkripsi/Dekripsi Sederhana

Saat ini telah banyak beredar program khusus penyandian pesan atau proteksi data baik *freeware*, *shareware*, maupun komersial yang sangat baik. Pada umumnya program tersebut tidak hanya menyediakan satu metoda saja, tetapi beberapa jenis sehingga kita dapat memilih yang menurut kita paling aman[SUP03].

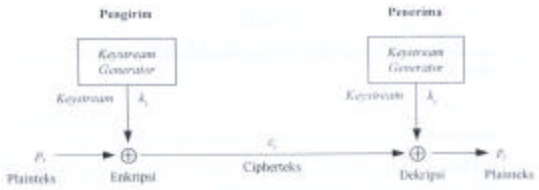
Sebuah *cipher* dikatakan aman jika tidak ada cara lain untuk memecahkan sebuah algoritma (memecahkan berarti menemukan plainteks atau kunci nya hanya dari cipherteks yang diberikan) selain dengan mencari di setiap kemungkinan kunci. Disinilah peranan kunci yang panjang. Semakin panjang kunci, semakin banyak kemungkinan kunci yang didapat, semakin sulit menemukan kunci yang tepat, maka semakin aman algoritma tersebut. Misalnya, untuk kunci 160 bit, maka akan terdapat 2^{160} kemungkinan kunci, yang jika dihitung kira-kira sekitar 1.46×10^{48} kunci yang harus dianalisis. Saat ini panjang kunci yang aman berkisar antara 128-256 bit [KREM00].

Salah satu metoda yang cukup aman adalah kunci simetri, yang mengimplementasikan kunci identik antara kunci untuk mengenkripsi data (disebut juga kunci enkripsi atau *enciphering key*) dan kunci untuk mendekripsi data (disebut juga kunci dekripsi atau *deciphering key*). Ataupun jika kedua kunci itu berbeda, kunci yang satu dapat dikomputasi untuk membangkitkan kunci yang lainnya [MEY82]. Algoritma RC2, RC4, RC5 dan RC6 termasuk dalam metoda ini.

Cipher simetrik dibedakan menjadi dua yaitu *cipher* aliran (*stream cipher*) yang mengoperasikan plainteks satu bit per satuan waktu (Gambar 2) dan *cipher* blok (*block cipher*) yang mengoperasikan plainteks dalam sekumpulan bit (blok) per satuan waktu (Gambar 3). RC4 merupakan *cipher* aliran sedangkan RC2, RC5 dan RC6 termasuk ke dalam *cipher* blok.



Gambar 2. Skema Enkripsi dan Dekripsi dengan *Cipher* Blok



Gambar 3. Skema Enkripsi dan Dekripsi dengan *Cipher* Aliran

2. RC2

RC2 adalah *cipher* blok yang menggunakan 64 bit sebagai ukuran per blok nya dengan kunci yang ukurannya bervariasi (0-1024 bit). Dengan mengubah-ubah ukuran kunci ini, performansi RC2 dapat menjadi dua atau tiga kali lebih baik dibanding DES (Data Encryption Standard), algoritma yang dikembangkan oleh NSA (National Security Agent) dan telah ditetapkan sebagai algoritma enkripsi standar oleh pemerintah AS pada tahun 1976-1997 (yang kemudian digantikan oleh AES, Advanced Encryption Standard).

```
sBox: array[0..255] of byte =

(D9,78,F9,C4,19,DD,B5,ED,28,E9,F
D,79,4A,A0,D8,9D,C6,7E,37,83,2B,
76,53,8E,62,4C,64,88,44,8B,FB,A2
17,9A,59,F5,87,B3,4F,13,61,45,6D
,8D,09,81,7D,32,BD,8F,40,EB,86,B
7,7B,0B,F0,95,21,22,5C,6B,4E,82,
54,D6,65,93,CE,60,B2,1C,73,56,C0
,14,A7,8C,F1,DC,12,75,CA,1F,3B,B
E,E4,D1,42,3D,D4,30,A3,3C,B6,26,
6F,BF,0E,DA,46,69,07,57,27,F2,1D
,9B,BC,94,43,03,F8,11,C7,F6,90,E
F,3E,E7,06,C3,D5,2F,C8,66,1E,D7,
08,E8,EA,DE,80,52,EE,F7,84,AA,72
,AC,35,4D,6A,2A,6,1A,D2,71,5A,15
,49,74,4B,9F,D0,5E,04,18,A4,EC,C
2,E0,41,6E,0F,51,CB,CC,24,91,AF,
50,A1,F4,70,39,99,7C,3A,85,23,B8
,B4,7A,FC,02,36,5B,25,55,97,31,2
D,5D,FA,98,E3,8A,92,AE,05,DF,29,
10,67,6C,BA,C9,D3,00,E6,CF,E1,9E
,A8,2C,63,16,01,3F,58,E2,89,A9,0
D,38,34,1B,AB,33,FF,B0,BB,48,0C,
5F,B9,B1,CD,2E,C5,F3,DB,47,E5,A5
,9C,77,0A,A6,20,68,FE,7F,C1,AD)
```

Gambar 4. S-Box RC2 berupa tabel satu dimensi berisi byte

Awalnya, kunci masukan dari *user* di-*padding* (jika kunci kurang dari 128 byte) menggunakan S-Box RC2 (lihat Gambar 4), dengan fungsi sebagai berikut:

```
for i ← Len to 127 do
   KeyB[i] ← sBox[(KeyB[i-Len] +
      KeyB[i-1])]
endfor
KeyB[0] ← sBox[KeyB[0]]
```

`Len` = panjang kunci masukan dari *user*

Kemudian hasilnya dikonversi menjadi 64 buah kunci dengan masing-masing berukuran 2 byte (16 bit).

Di sisi lain, blok-64-bit dibagi menjadi 4 *word* yang masing-masing berukuran 16 bit. *Word* ini kemudian dikomputasikan dengan kunci 16 bit di atas.

Algoritma RC2 ini tidak dipatenkan oleh RSA Security, Inc., namun nama algoritmanya (yaitu RC2) merupakan *copyright*.

Contoh aplikasi kriptografi yang menggunakan algoritma ini adalah HotCrypt, wodCrypt, FileBarricader 2004, AEPE PRO, EDCrypt serta SSLite dan CryptoLite dari IBM [SUR05].

3. RC4

Sama halnya dengan RC2, RC4 juga menggunakan kunci yang ukurannya bervariasi (0-2048 bit) namun pemerintah AS melarang penggunaan algoritma ini dengan kunci lebih besar dari 40 bit, kecuali bagi pemerintahan luar negeri dan perusahaan internasional (56 bit). Semakin besar kuncinya, semakin aman proses kriptografi, sehingga semakin sulit untuk memecahkannya.

Algoritma ini didasarkan pada kegunaan dari hasil permutasi yang acak. Langkah-langkah nya adalah sebagai berikut:

1. Melakukan inisialisasi pada tabel S satu dimensi:

```
S[0] = 0
S[1] = 1
   :
   :   ...dst
   :
```

```
S[255] = 255

for i← 0 to 255 do
      S[i] ← i
endfor
```

2. Jika panjang kunci U (masukan dari *user*) < 256 karakter (byte), kunci tersebut di-*padding*, yaitu penambahan bit-bit isian pada akhir kunci. Contoh: kunci dari user adalah `'kripto'`. Karena kurang dari 256 byte, maka kunci tersebut di-*padding* menjadi `'kriptokriptokripto.....'` hingga 256 karakter.

3. Melakukan permutasi terhadap setiap nilai tabel S:

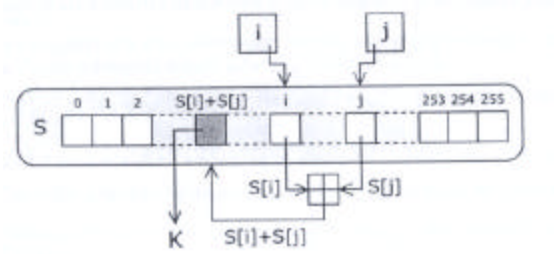
```
j ← 0
for i ← 0 to 255 do
     j←(j+ S[i] + U[i]) mod 256
     swap(S[i],S[j])
endfor
```

Dimana fungsi `swap` adalah proses pertukaran nilai `S[i]` dan `S[j]`.

4. Membangkitkan aliran kunci (*keystream*) dan mengenkripsi plainteks dengan *keystream* tersebut:

```
i ← 0
j ← 0
for idx←0 to PanjangPlainteks
– 1 do
     i ← (i+ 1) mod 256
     j ← (j+S[i]) mod 256
     swap(S[i],S[j])
     t ← (S[i]+S[j]) mod 256
     K ← S[t]
     C ← K XOR P[idx]
endfor
```

Keterangan:
`P` : *array* of karakter plainteks
`K` : *keystream* yang dibangkitkan
`C` : cipherteks



Gambar 5. Pembangkitan kunci aliran K

Proses pembangkitan kunci aliran K dalam algoritma RC4 ini dapat dilihat lebih jelas pada Gambar 5 di atas. Pada *cipher* aliran, semakin acak fungsi pembangkitan kunci alirannya, maka akan semakin baik pula performansi *cipher*-nya. Hal itu disebabkan karena algoritma menjadi tidak mudah dipahami oleh penyerang [AND95].

5. Dekripsi
Untuk mendapatkan pesan semula, kunci aliran yang dibangkitkan harus sama dengan kunci aliran pada proses enkripsi. Sehingga, algoritmanya:

```
i ← 0
j ← 0
for idx←0 to PanjangCipherteks
– 1 do
     i ← (i+ 1) mod 256
     j ← (j+S[i]) mod 256
     swap(S[i],S[j])
     t ← (S[i]+S[j]) mod 256
     K ← S[t]
     P ← K XOR C[idx]
endfor
```

Keterangan:
`C` : *array* of karakter cipherteks
`K` : *keystream* yang dibangkitkan
`P` : plainteks

RC4 digunakan untuk pengenkripsian file dalam banyak aplikasi seperti RSA SecurPC (salah satu produk RSA Security, Inc). Selain itu juga digunakan untuk keamanan komunikasi dalam lalu lintas pengiriman data dan keamanan keamanan website menggunakan protokol SSL (Secure Sockets Layer).

Contoh aplikasi kriptografi yang menggunakan algoritma ini adalah HotCrypt, wodCrypt serta SSLite dan CryptoLite dari IBM [IBM05].

4. RC5

RC5 adalah algoritma *cipher* blok yang bisa dibilang cukup cepat, dengan parameter-parameter:
1. Ukuran blok (32/64/128 bit)
2. Panjang kunci eksternal masukan dari *user* (0-2040 bit)
3. Jumlah putaran (0-255 bit).

Pemilihan 32 bit sebagai ukuran blok disarankan hanya untuk tujuan eksperimental dan evaluasi, karena mengenkripsi blok berukuran 32 bit sangat tidak aman untuk algoritma ini [RSA05]. RC5 dikembangkan oleh Ronald L. Rivest dari RSA Security, Inc. pada tahun 1994 dan diberikan hak paten pada tahun 1995 oleh pemerintah AS.

Ada 3 proses utama dalam RC5, yaitu perluasan kunci, enkripsi dan dekripsi. Perluasan kunci merupakan proses membangkitkan kunci internal dengan memanfaatkan komputasi rotasi *left regular shift* (<<) dan *right regular shift* (>>), dengan panjang kunci tergantung dari jumlah putaran [MUN06]. Kunci internal ini kemudian digunakan dalam proses enkripsi dan dekripsi. Proses enkripsi dibagi menjadi tiga, yaitu: penjumlahan *integer*, XOR dan rotasi. Untuk lebih jelasnya, adalah sebagai berikut:

Sebut saja ukuran blok adalah `w` bit, `r` putaran dan `b` karakter (byte) kunci eksternal.

1. Pembentukan kunci internal

   `K[0-1]...K[b]` disalin ke tabel `L[0-1]...L[b]` dengan aturan di-*padding* dengan karakter 0 hingga ukuran `L[i]` menjadi `w/2` bit.

   Contoh:

   Kunci eksternal K = ’kripto’ dan ukuran blok adalah 64 bit.

```
K[0] = k   L[0] = k000
K[1] = r   L[1] = r000
K[2] = i   L[2] = i000
K[3] = p   L[3] = p000
K[4] = t   L[4] = t000
K[5] = o   L[5] = o000
```

Kemudian, inisialisasi tabel kunci internal KI dengan ukuran `t = 2r + 2` seperti berikut:

```
KI[0] ← P
for i ← 1 to t – 1 do
    KI[i] ← KI[i – 1] do
endfor
```

dengan aturan sebagai berikut:

```
untuk w = 32:
P = B7E1
Q = 9E37

untuk w = 64 :
P = B7E15163
Q = 9E3779B9

untuk w = 128 :
P = B7E151628AED2A6B
Q = 9E3779B97F4A7C15
```

Adapun konstanta `P` dan `Q` di atas didapatkan dari fungsi yang melibatkan bilangan irasional sebagai berikut:

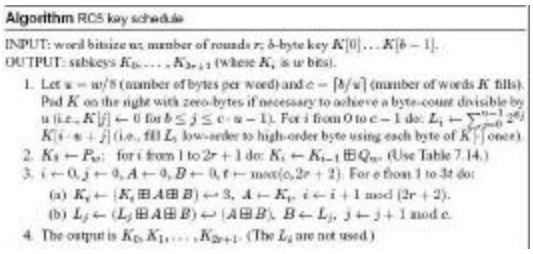
$P = Odd[(e - 2)2^w]$
$Q = Odd[(f - 1)2^w]$

Keterangan:
`e = 2.718281828459.....`
`F = 1.618033988749.....`

Kemudian `L` dan `S` digabungkan dengan algoritma berikut:

```
i ← 0
j ← 0
X ← 0
Y ← 0
n ← 3*max(r,c)
for k ← 1 to n do
    KI[i]←(KI[i] + X + Y)<<< 3
    X ← KI[i]
    i ← (i + 1) mod t
    L[j] ← (L[j] + X + Y)<<< 3
    Y ← L[j]
    j← (j + 1) mod c
endfor
```

Keterangan:
`max(r,c)` adalah fungsi menentukan bilangan terbesar antara r dan c.



Algorithm RC5 key schedule

INPUT: word bitsize w, number of rounds r, b-byte key $K[0] \ldots K[b-1]$.
OUTPUT: subkeys $K_0, \ldots, K_{2r+1}$ (where K_i is w bits).

1. Let $u = w/8$ (number of bytes per word) and $c = \lceil b/u \rceil$ (number of words K fills). Pad K on the right with zero-bytes if necessary to achieve a byte-count divisible by u (i.e., $K[j] \leftarrow 0$ for $b \le j \le c \cdot u - 1$). For i from 0 to $c-1$ do: $L_i \leftarrow \sum_{j=0}^{u-1} 2^{8j} K[i \cdot u + j]$ (i.e., fill L_i low-order to high-order byte using each byte of $K[\,]$ once).
2. $K_0 \leftarrow P_w$; for i from 1 to $2r+1$ do: $K_i \leftarrow K_{i-1} \boxplus Q_w$. (Use Table 7.14.)
3. $i \leftarrow 0, j \leftarrow 0, A \leftarrow 0, B \leftarrow 0, t \leftarrow \max(c, 2r+2)$. For s from 1 to $3t$ do:
   (a) $K_i \leftarrow (K_i \boxplus A \boxplus B) \lll 3$, $A \leftarrow K_i$, $i \leftarrow i+1 \bmod (2r+2)$.
   (b) $L_j \leftarrow (L_j \boxplus A \boxplus B) \lll (A \boxplus B)$, $B \leftarrow L_j$, $j \leftarrow j+1 \bmod c$.
4. The output is $K_0, K_1, \ldots, K_{2r+1}$. (The L_i are not used.)

2. Enkripsi

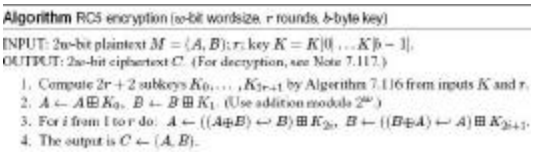
Mulanya, plainteks dibagi menjadi beberapa blok yang masing-masing memuat w bit. Dimulai dari sini, proses dikerjakan pada tiap blok.

Masing-masing blok dipecah menjadi dua *register* sama besar A dan B, yang berarti jika w = 64 bit, A dan B berisi 32 bit (4 karakter). Kemudian lakukan penjumlahan seperti berikut:

```
A ← A + KI[0]
B ← B + KI[1]
```

Selanjutnya dilakukan operasi XOR, rotasi dan penjumlahan lagi (sesuai dengan fungsi utama algoritma ini).

```
for i ← 1 to r do
    A←((A⊕B) <<< B) + KI[2i]
    B←((B⊕A) <<< A) + KI[2i+1]
endfor
```



Algorithm RC5 encryption (w-bit wordsize, r rounds, b-byte key)

INPUT: $2w$-bit plaintext $M = (A, B)$; r; key $K = K[0] \ldots K[b-1]$.
OUTPUT: $2w$-bit ciphertext C. (For decryption, see Note 7.11.)

1. Compute $2r+2$ subkeys $K_0, \ldots, K_{2r+1}$ by Algorithm 7.1 16 from inputs K and r.
2. $A \leftarrow A \boxplus K_0$, $B \leftarrow B \boxplus K_1$. (Use addition modulo 2^w.)
3. For i from 1 to r do: $A \leftarrow ((A \oplus B) \lll B) \boxplus K_{2i}$, $B \leftarrow ((B \oplus A) \lll A) \boxplus K_{2i+1}$.
4. The output is $C \leftarrow (A, B)$.

Secara umum, struktur algoritma di atas merupakan pendekatan dari prinsip jaringan Feistel. Hasil putaran terakhir akan mendapatkan cipherteks yang berupa gabungan dari A dan B Proses enkripsi ini dapat dilihat lebih jelas pada Gambar 6.

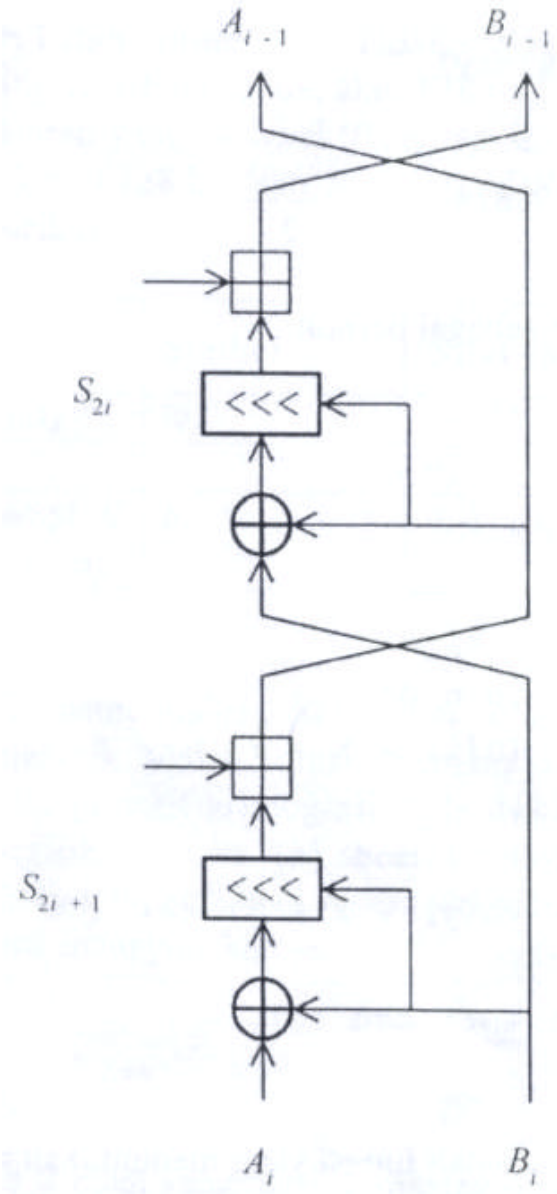
3. Dekripsi

Algoritma pada proses dekripsi merupakan kebalikan dari proses enkripsi. Jika tadinya digeser ke kiri, maka pada proses dekripsi dilakukan pergeseran ke kanan (*right regular shift*).

```
for i ← r downto 1 do
    B←((B – KI[2i+1])>>>A) ⊕ A
    A←((A – KI[2i])>>>B) ⊕ B
endfor
B ← B - KI[1]
A ← A - KI[0]
```

Untuk mendekripsi cipherteks, diperlukan KI yang sama dengan KI saat mengenkripsi. Proses pembangkitan KI pada kedua proses itu juga sama.

Contoh aplikasi kriptografi yang menggunakan algoritma ini adalah HotCrypt, wodCrypt serta SSLite dan CryptoLite dari IBM.



Gambar 6. Enkripsi satu putaran dalam algoritma RC5

5. RC6

RC6 adalah algoritma cipher blok yang sangat aman, padat dan sederhana dan menawarkan performansi yang sangat bagus dan fleksibel, dikembangkan dari algoritma RC5 oleh Ronald Linn Rivest, Ray Sidney, Matt J.B. Robshaw dan Yiquin Yin dari RSA Securiy, Inc pada tahun 1998. Seperti halnya RC5, parameter dari algoritma ini adalah ukuran blok, ukuran kunci eksternal dan jumlah putaran yang bervariasi dengan batasan sama seperti pada RC5.

Karena belum ditemukannya serangan yang dapat memecah RC5, beberapa penelitian mengusulkan beberapa serangan teoritis yang menarik, yang pada umumnya berdasarkan fakta bahwa jumlah rotasi di RC5 tidak bergantung pada seluruh bit dalam sebuah *register* (blok dibagi dua). RC6 didesain untuk mencegah serangan seperti ini, dan sekaligus untuk diajukan menjadi AES dengan mempertimbangkan keamanan, kesederhanaan, dan performansi yang baik [RIV98].

Algoritma ini merupakan salah satu finalis AES selain Rijndael (oleh Vincent Rijmen dan Joan Daemen), Serpent (oleh Ross Anderson, Eli Biham dan Lars Knudsen), Twofish (oleh Bruce Schneier), dan MARS (dari IBM) dari 15 kandidat, yaitu : Cast-256, Crypton, Deal, DFC, E2, Frog, HPC, Loki97, Magenta, Mars, RC6, Rijndael, Safer+, Serpent dan Twofish.

Karenanya RC6 memiliki karakteristik seperti ke-4 algoritma finalis lainnya yaitu termasuk *cipher* blok, seluruh rancangan algoritma harus dipublikasikan, panjang kunci fleksibel (seperti RC5), dan ukuran bloknya 128 bit (bisa juga untuk 32 dan 64 bit).

Pada RC5, seperti yang telah dijelaskan sebelumnya, plainteks berukuran 128 bit per bloknya akan dipisahkan menjadi dua *register* berukuran masing-masing 64 bit.

Namun arsitektur dan bahasa pemrograman yang dijadikan batasan dalam pengembangan AES belum mendukung operasi *register* berukuran 64 bit. Oleh karena itu, RC5 dimodifikasi dalam RC6 dimana penggunaan dua buah *register* 64 bit diganti menjadi empat buah *register* 32 bit. Hal ini memberikan keuntungan di sisi lain yaitu terdapat dua operasi rotasi untuk satu putaran, bukan satu rotasi untuk setengah putaran seperti pada *cipher* RC5.

Ada dua fitur utama dalam RC6 dibandingkan dengan RC5, yaitu: perkalian *integer* dan penggunaan empat buah *register* berukuran `w/4` bit, bukan `w/2` bit seperti pada RC5 (`w` adalah ukuran blok yang digunakan dalam satuan bit). Perkalian *integer* digunakan untuk menambah

pencapaian *diffusion*[1] untuk tiap putaran sehingga jumlah putaran dapat diperkecil dan kecepatan proses enkripsi menjadi bertambah. Selain itu, fitur utama lainnya masih berada di sekitar operasi rotasi.

Untuk lebih lengkapnya, berikut adalah enam buah operasi dasar yang digunakan oleh algoritma RC6 untuk melakukan enkripsi/dekripsi pesan:

1. $(a + b) \bmod 2^w$
2. $(a - b) \bmod 2^w$
3. $a \oplus b$
4. $(a \times b) \bmod 2^w$
5. $a <<< b$
6. $a >>> b$

1. Pembentukan kunci internal

`K[0-1]...K[b]` disalin ke tabel `L[0-1]...L[b]` dengan aturan di-*padding* dengan karakter 0 hingga ukuran `L[i]` menjadi `w/4` bit.

Contoh:

Kunci eksternal K = 'kripto' dan ukuran blok adalah 64 bit.

```
K[0] = k   L[0] = k0
K[1] = r   L[1] = r0
K[2] = i   L[2] = i0
K[3] = p   L[3] = p0
K[4] = t   L[4] = t0
K[5] = o   L[5] = o0
```

Kemudian, inisialisasi tabel kunci internal KI dengan ukuran `t = 4r + 2` seperti berikut:

```
KI[0] ← P
for i ← 1 to t – 1 do
     KI[i] ← KI[i – 1] do
endfor
```

dengan aturan sebagai berikut:

```
untuk w = 32:

P = B7E1
Q = 9E37

untuk w = 64 :

P = B7E15163
Q = 9E3779B9

untuk w = 128 :

P = B7E151628AED2A6B
Q = 9E3779B97F4A7C15
```

Adapun konstanta `P` dan `Q` di atas didapatkan dari fungsi yang melibatkan bilangan irasional sebagai berikut:

$P = Odd[(e - 2)2^w]$
$Q = Odd[(f - 1)2^w]$

Keterangan:
`e = 2.718281828459.....`
`F = 1.618033988749.....`

Kemudian `L` dan `S` digabungkan dengan algoritma berikut:

```
i ← 0
j ← 0
X ← 0
Y ← 0
n ← 3*max(r,c)
for k ← 1 to n do
     KI[i]←(KI[i] + X + Y)<<< 3
     X ← KI[i]
     i ← (i + 1) mod t
     L[j] ← (L[j] + X + Y)<<< 3
     Y ← L[j]
     j← (j + 1) mod c
endfor
```

Keterangan:
`max(r,c)` adalah fungsi menentukan bilangan terbesar antara r dan c.

[1] Prinsip *diffusion*: pengubahan satu bit plainteks sebanyak satu atau dua bit menghasilkan perubahan pada cipherteks yang tidak dapat diprediksi.

Key schedule for RC6-$w/r/b$

Input: User-supplied b byte key preloaded into the c-word array $L[0,\ldots,c-1]$
Number r of rounds

Output: w-bit round keys $S[0,\ldots,2r+3]$

Procedure:

$S[0] = P_w$

for $i = 1$ **to** $2r+3$ **do**
$S[i] = S[i-1] + Q_w$

$A = B = i = j = 0$

$v = 3 \times \max\{c, 2r+4\}$
for $s = 1$ **to** v **do**
{
$A = S[i] = (S[i] + A + B) \lll 3$
$B = L[j] = (L[j] + A + B) \lll (A + B)$
$i = (i+1) \bmod (2r+4)$
$j = (j+1) \bmod c$
}

2. Enkripsi

Mulanya, plainteks dibagi menjadi blok-blok yang masing-masing memuat w bit. Dimulai dari sini, proses dikerjakan pada tiap blok.

Masing-masing blok dipecah menjadi empat *register* sama besar A, B, C dan D yang berarti jika w = 64 bit, masing-masing *register* berisi 16 bit (empat karakter). Kemudian lakukan penjumlahan seperti berikut:

```
B ← B + KI[0]
D ← D + KI[1]
```

Selanjutnya dilakukan operasi XOR, rotasi dan penjumlahan lagi (sesuai dengan fungsi utama algoritma ini).

```
for i ← 1 to r do
    t ← (B x (2B + 1))<<< lg w
    u ← (D x (2D + 1))<<< lg w
    A←((A⊕t) <<< t) + KI[2i]
    C←((C⊕u) <<< u) + KI[2i+1]
    (A,B,C,D)=(B,C,D,A) //swap
endfor
A ← A + KI[2r + 2]
C ← C + KI[2r + 3]
```

Keterangan:
$\text{lg w} = \log_2 w$

Hasil putaran terakhir akan mendapatkan cipherteks yang berupa gabungan dari A, B, C dan D.

Encryption with RC6-$w/r/b$

Input: Plaintext stored in four w-bit input registers A, B, C, D
Number r of rounds
w-bit round keys $S[0,\ldots,2r+3]$

Output: Ciphertext stored in A, B, C, D

Procedure:

$B = B + S[0]$
$D = D + S[1]$
for $i = 1$ **to** r **do**
{
$t = (B \times (2B+1)) \lll \lg w$
$u = (D \times (2D+1)) \lll \lg w$
$A = ((A \oplus t) \lll u) + S[2i]$
$C = ((C \oplus u) \lll t) + S[2i+1]$
$(A, B, C, D) = (B, C, D, A)$
}
$A = A + S[2r+2]$
$C = C + S[2r+3]$

3. Dekripsi

Algoritma pada proses dekripsi merupakan kebalikan dari proses enkripsi. Jika tadinya digeser ke kiri, maka pada proses dekripsi dilakukan pergeseran ke kanan (*right regular shift*).

```
C ← C – KI[2r + 3]
A ← A – KI[2r + 2]
for i ← r downto 1 do
    (A,B,C,D)=(D,A,B,C) //swap
    u ← (D x (2D + 1) <<< lg w
    t ← (B x (2B + 1) <<< lg w
    C←((C – KI[2i+1])>>>t) ⊕ u
    A←((A – KI[2i])>>>u) ⊕ t
endfor
D ← D - KI[1]
B ← B - KI[0]
```

KI dibangkitkan sama seperti pembangkitan KI pada proses enkripsi.

Decryption with RC6-$w/r/b$

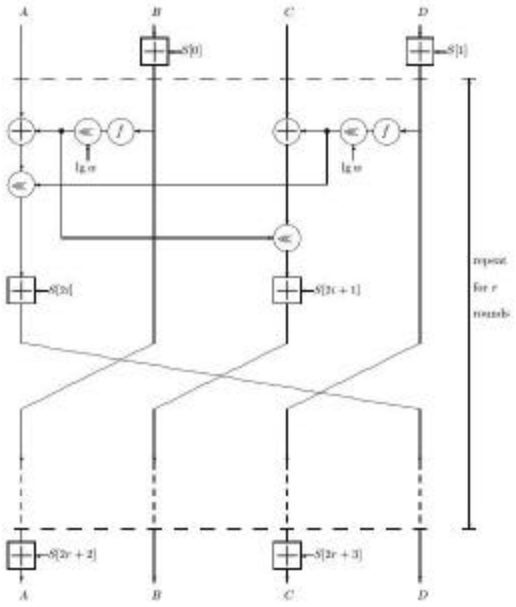
Input: Ciphertext stored in four w-bit input registers A, B, C, D
Number r of rounds
w-bit round keys $S[0,\ldots,2r+3]$

Output: Plaintext stored in A, B, C, D

Procedure:

$C = C - S[2r+3]$
$A = A - S[2r+2]$
for $i = r$ **downto** 1 **do**
{
$(A, B, C, D) = (D, A, B, C)$
$u = (D \times (2D+1)) \lll \lg w$
$t = (B \times (2B+1)) \lll \lg w$
$C = ((C - S[2i+1]) \ggg t) \oplus u$
$A = ((A - S[2i]) \ggg u) \oplus t$
}
$D = D - S[1]$
$B = B - S[0]$

Untuk lebih jelasnya, proses enkripsi untuk satu putaran dapat dilihat pada Gambar 7.



Gambar 7. Enkripsi satu putaran dalam algoritma RC6

Contoh aplikasi kriptografi yang menggunakan algoritma ini adalah HotCrypt, wodCrypt serta SSLite dan CryptoLite dari IBM.

Percobaan[2] RC6 terhadap beberapa plainteks dapat dilihat sebagai berikut:

```
plaintext   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
user key    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ciphertext  8f c3 a5 36 56 b1 f7 78 c1 29 df 4e 98 48 a4 1e

plaintext   02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
user key    01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
ciphertext  52 4e 19 2f 47 15 c6 23 1f 51 f6 36 7e a4 3f 18

plaintext   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
user key    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                                    00 00 00 00 00 00 00 00
ciphertext  6c d6 1b cb 19 0b 30 38 4e 8a 3f 16 86 90 ae 82

plaintext   02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
user key    01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
                        89 9a ab bc cd de ef f0
ciphertext  68 83 29 d0 19 e5 05 04 1e 52 e9 2a f9 52 91 d4

plaintext   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
user key    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
            00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ciphertext  8f 5f bd 05 10 d1 5f a8 93 fa 3f da 6e 85 7e c2

plaintext   02 13 24 35 46 57 68 79 8a 9b ac bd ce df e0 f1
user key    01 23 45 67 89 ab cd ef 01 12 23 34 45 56 67 78
            89 9a ab bc cd de ef f0 10 32 54 76 98 ba dc fe
ciphertext  c8 24 18 16 f0 d7 e4 89 20 ad 16 a1 67 4e 5d 48
```

[2] Penulis menggunakan aplikasi HotCrypt

6. Kriptanalisis terhadap RC2, RC4, RC5 dan RC6

6.1 RC2

Related-key attack adalah kriptoanalisis differensial[3] yang mengasumsikan bahwa penyerang mempelajari pola enkripsi suatu plainteks tertentu bukan dari kunci aslinya, namun dari keterhubungan antara dua kunci yang digunakan untuk mengenkripsi plainteks yang sama (K' = (f(K)). Ada dua tipe serangan ini yaitu *chosen-related-key*, penyerang memilih kunci-kunci tertentu untuk melakukan kriptanalis, dan *known-related-key* dimana dua kunci yang berbeda itu diketahui. Bruce Schneier, John Kelsey dan David Wagner telah berhasil memecahkan *cipher* RC2 dengan menggunakan *chosen-related-key* [SCH98].

Algoritma RC2 membebaskan *user* memasukkan kunci dengan ukuran apapun (dengan batasan lebih kecil dari 1024 bit) yang kemudian dikembangkan menjadi 128 byte dengan bantuan S-Box, hasilnya adalah 64 buah subkunci yang masing-masing berukuran 16 bit. Bruce dan teman-temannya menemukan karakteristik unik dari bit tunggal.

Misal, kunci berukuran 64 byte:
`K = (x0, x1,..., x63)` dan
`K' = (x'0, x1,..., x'63)`
Dengan kata lain, K dan K' hanya berbeda di elemen pertama dan terakhir.

```
S-Box[x0+x63]=S-Box[x'0+x'63]
```

Kemudian kedua kunci itu diperluas menjadi 128 byte:

```
K[0]=x0...K[63]=x63  dan
K[i]=S-Box[K[i-1]+K[i-64]]

K'[0]=x'0...K'[63]=x'63  dan
```

[3] Kriptoanalisis differensial (*differential cryptoanalysis*) merupakan metode serangan yang diperkenalkan oleh Eli Biham dan Adi Shamir untuk menganalisa efek yang ditimbulkan oleh perbedaan pada suatu pasangan plainteks terhadap perbedaan yang terjadi pada resultan pasangan cipherteks yang dihasilkan [GUR03].

```
K'[i]=S-Box[K'[i-1]+K'[i-64]]
```

Keterangan:
```
i >= 64
```

K sekarang hanya berbeda di posisi 0, 63 dan 127 dengan K' yang sekarang.

Dengan menerka-nerka perbedaan sejauh t antara `K[0]` dan `K'[0]`, RC2 dapat dipecahkan dengan memiliki satu kunci di antara `K` dan `K'` dan 2^{34} kemungkinan plainteks.

6.2. RC4

Sampai saat ini *cipher* RC4 belum bisa dipecahkan kecuali menggunakan *exhaustive search (brute force)*[4]. Batas kunci RC4 yang bisa dipecahkan adalah 40 bit. Kini pemerintah AS sudah mengubah ketetapannya mengenai ekspor/penggunaan algoritma kriptografi yang lebih besar dari 40 bit di luar negerinya. Batasnya adalah 128 bit.

6.3. RC5

David Wagner, John Kelsey and Bruce Schneier telah menemukan kunci lemah (*weak keys*) pada RC5, dengan kemungkinan pemilihan kunci lemah tersebut adalah 2^{10r}, dimana r adalah jumlah putaran.

Kunci lemah adalah suatu kunci rahasia pada algoritma blok dengan suatu nilai tertentu yang dapat memperlihatkan suatu ketraturan yang terjadi pada proses enkripsi. Keteraturan ini akan mempermudah kerja seorang kriptoanalis yang mencoba untuk melakukan serangan terhadap pesan yang dienkripsi dengan menggunakan kunci tersebut.

Oleh karena itu, penggunaan r yang aman adalah yang lebih besar dari 10. Kundsen juga telah menemukan sebuah serangan terhadap *cipher* ini, namun belum dipublikasikan secara luas [KRE00].

6.4. RC6

RC6 adalah algoritma yang paling baru di antara keempat lainnya, dan belum ada catatan mengenai kriptanalisis terhadap algoritma ini. Perlu diingat bahwa algoritma ini hampir setara dengan algoritma Rijndael yang ditetapkan sebagai AES pada tahun 2001-2010. Jika Rijndael diperkirakan akan bertahan (tidak dapat dipecahkan) hingga 5 tahun lagi, maka RC6 juga begitu.

7. Perbandingan antara RC2, RC4, RC5 dan RC6

Di antara keempatnya, cipher RC4 yang berbeda karena termasuk cipher aliran, bukan cipher blok seperti yang lainnya. Secara umum, kelebihan dari sebuah cipher aliran (yang melakukan enkripsi per bit) adalah kerusakan pada satu bit tidak akan mempengaruhi keseluruhan isi pesan, hanya satu bit itu saja. Selain itu, cipher ini sangat cocok untuk aplikasi yang melakukan pemrosesan data per bit nya, mungkin dikarenakan untuk menghemat isi memori atau buffer [MEN96].

Sedangkan cipher RC5 dan RC6, walaupun fitur utamanya hampir sama, namun RC6 jelas lebih baik dibanding RC5. Esensi dari *cipher* RC5 adalah memanfaatkan operasi rotasi yang diimplementasikan pada prosesor modern secara efisien. RC6 juga mengikuti langkah-langkah RC5 ini dan mengambil keuntungan dari fakta bahwa perkalian *integer* 32 bit telah diimplementasikan secara efisien di banyak prosesor sehingga *cipher* ini sangat cocok untuk digunakan di prosesor-prosesor tersebut. Perkalian *integer* ini adalah fungsi primitif *diffusion* yang sangat efektif dan digunakan oleh RC6 pada operasi rotasinya, sehingga jumlah rotasi sangat bergantung pada bit-bit *register* lainnya (sebelumnya) dibandingkan dengan operasi yang hanya berupa penjumlahan bit seperti pada RC5. sebagai hasilnya, RC6 lebih cepat dibanding RC5.

Di bawah ini adalah daftar karakteristik dari algoritma RC2, RC4, RC5 dan RC6.

[4] *Exhaustive key search* atau *brute force search* adalah suatu teknik dasar yang digunakan kriptoanalis untuk mencoba setiap kunci yang mungkin sampai ditemukan kunci yang sebenarnya.

Tabel 1. Perbandingan karakteristik RC2, RC4, RC5 dan RC6

Karakte-ristik	*Cipher*			
	RC2	**RC4**	**RC5**	**RC6**
Tahun	Dipub-likasi-kan 1998	1994	1994	1998
Jenis	cipher blok	cipher aliran	cipher blok	cipher blok
Hak paten	Tidak (*trade secret)*	Tidak *(trade secret)*	Ya	Ya
Kunci (bit)	0-1024	0-2048	0-2048	0-2048
Blok (bit)	64	-	32/64 / 128	32/64 / 128
Putaran	1	-	0-255	0-255

Di bawah ini adalah prinsip-prinsip *cipher* blok pada algoritma RC2, RC5 dan RC6. Dapat dilihat di sana bahwa algoritma RC6 adalah yang paling bagus performansi nya. Walaupun algoritma ini tidak menggunakan konsep substitusi, namun sudah dirancang dengan sangat rumit: ada *confussion*, *diffusion*, dan jaringan *feistel*.

Tabel 2. Perbandingan RC2, RC5 dan RC6 dilihat dari sudut pandang prinsip-prinsip perancangan *cipher* blok

Prinsip	*Cipher*		
	RC2	**RC5**	**RC6**
Confussion	v	v	v
Diffusion	X	v	v
Cipher berulang	v	v	v
Jaringan *Feistel*	X	v	v
Kunci Lemah	v	v	X
S-Box	v	X	X

8. Kesimpulan

Kekuatan dari metoda-metoda enkripsi adalah pada kunci (dari masukan *user*) sehingga walaupun algoritma metoda tersebut telah tersebar luas, orang yang tidak berkepentingan tidak akan dapat membongkar data tanpa kunci yang tepat. Walaupun tentunya untuk menemukan metoda tersebut diperlukan teori matematika yang cukup rumit. Tetapi intinya disini ialah bagaimana mengimplementasikan metoda-metoda yang telah diakui keampuhannya tersebut didalam aplikasi kriptografi sehingga dapat meningkatkan keamanan dari aplikasi yang dibangun.

Dari empat algoritma yang dibahas pada makalah ini, algoritma RC6 yang paling aman, rumit dan sederhana, karena melibatkan *confussion*, *diffusion*, jaringan *feistel* dan panjang blok dan kunci yang besar.

DAFTAR PUSTAKA

[MUN06] Munir, Rinaldi. 2006. *Diktat Kuliah IF5054 Kriptografi*. Bandung: Program Studi Teknik Informatika, Institut Teknologi Bandung.

[MEY82] Meyer, Carl H., dan Matyas, Stephen M. 1982. *Cryptography: A New Dimension in Computer Data Security*. New York: John Wiley & Sons.

[RSA05] RSA Securiti, Inc. 2005. *RSA Laboratories*. http://www.rsasecurity.com/rsalabs. [28 September 2006].

[SUK98] Sukmawan, Budi. 1998. *Keamanan Data dan Metode Enkripsi*. Info Komputer Internet edisi Jan 1998. http://www.infokomputer.com/arsip/internet/0698/cakra/cakrawa1.shtml. [28 September 2006].

[GUR03] Guritman, Sugi., dkk. 2003. *Algoritma Blowfish untuk Penyandian Pesan*. Jurnal Ilmiah Ilmu Komputer vol.1 no.1, edisi September 2003, hal 9-19. Bogor: Institut Pertanian Bogor.

[PRA98] Prasetya, ISWB. 1998. *Mengupas Rahasia Penyandian Informasi*. Info Komputer Internet vol II no.6, edisi Juni 1998, hal 10-16.

[RIV03] Rivest, Ronald L.,. *Ronald L. Rivest's Homepage*.

http://theory.csail.mit.edu/%7Erivest/homepage.html. [28 September 2006].

[SCH98] Schneier, Bruce., dkk. 1998. *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*. Paper.

[KRE00] KremlinEncrypt. 2000. *Concepts of Cryptography*. http://www.kremlinencrypt.com/crypto.html . [28 September 2006].

[SMA05] SmartComputing. 2005. *RC2, RC4, RC5 and RC6*. USA: Sandhills Publishing Company. http://www.smartcomputing.com. [28 September 2006].

[RIV98] Rivest, Ronald L., dkk. 1998. *The RC6 Block Cipher*. Paper.

[SUP03] Suprapti, Iswanti. 2003. *Studi Sistem Keamanan Data dengan Metode Public Key Cryptography*. Paper.

[MEN96] Menezes, A., dkk. 1996. *Handbook of Applied Cryptography*. CRC Press, Inc.

[IBM05] IBM. 2005. *IBM Cryptographic Toolkits*. http://www.ibm.com [28 September 2003].

[SUR05] Surfpack. 2005. *File Encryption for Your Important Files*. http://www.surfpack.com [28 September 2003].

[AND95] Anderson, R. J. 1995. *A Faster Attack on Certain Stream Cipher*. Cambridge: University Computer Laboatory.