ISBN: 979-756-061-6

STUDI DAN ANALISIS ALGORITMA *RIVEST CODE 6* (RC6) DALAM ENKRIPSI/DEKRIPSI DATA

Yudi Prayudi dan Idham Halik
Jurusan Teknik Informatika, Fakultas Teknologi Industri,Universitas Islam Indonesia
Jl. Kaliurang Km. 14 Yogyakarta 55501
Telp. (0274) 895287 ext. 122, Faks. (0274) 895007ext. 148
E-mail: prayudi@fti.uii.ac.id

ABSTRAK

Keamanan dan kerahasiaan sebuah data atau informasi dalam komunikasi dan pertukaran informasi sangatlah penting. Seringkali data atau informasi yang penting, dalam komunikasi dan pertukaran informasi kadang tidak sampai kepada penerima atau tidak hanya diterima oleh penerima tetapi juga oleh pihak lain yang melakukan pembajakan atau penyadapan. Hal ini membuat data atau informasi tersebut menjadi tidak berguna lagi dan lebih parahnya lagi kadang data atau informasi tersebut oleh para pembajak digunakan untuk menjatuhkan pihak lain. Oleh karena itu kriptografi sangat dibutuhkan dalam menjaga kerahasiaan data atau informasi.

Algoritma kriptografi terdiri dari algoritma enkripsi (E) dan algoritma dekripsi (D). Enkripsi dimaksudkan untuk melindungi informasi agar tidak terlihat oleh orang atau pihak yang tidak berhak. Ada banyak model dan metode enkripsi, salah satu di antaranya adalah enkripsi dengan algoritma Rivest Code 6 (RC6). Model ini merupakan salah satu algoritma kunci simetris yang berbentuk block chiper yang merupakan salah satu kandidat dalam penentuan algoritma standart untuk kriptografi DES yang sekarang disebut AES.

Pada tulisan ini akan dibahas sejumlah aspek dari kriptografi serta konsep dasar dari algoritma RC6. Sebuah aplikasi dirancang untuk dapat mengimplementasikan algoritma RC6. Dengan bantuan aplikasi tersebut kemudian dilakukan sejumlah analisis menyangkut keterkaitan waktu dan ukuran file terhadap proses enkripsi dan deskripsi, panjang kunci serta kinerja algoritma RC6 dibandingan dengan algoritma RC4 dan Blowfish.

Kata kunci: *Kriptografi, RC6, Block cipher, stream cipher, asimetris, private key*

1. PENDAHULUAN

Keamanan dan kerahasiaan data pada jaringan komputer saat ini menjadi isu yang sangat penting dan terus berkembang. Beberapa kasus menyangkut keamanan jaringan komputer saat ini menjadi suatu pekerjaan yang membutuhkan biaya penanganan dan pengamanan yang sedemikian besar. Sistem-sistem vital, seperti sistem pertahanan, sistem perbankan, sistem bandara udara dan sistem-sistem yang lain setingkat itu, membutuhkan tingkat keamanan yang sedemikian tinggi. Hal ini lebih di sebabkan karena kemajuan bidang jaringan komputer dengan konsep *open system*-nya sehingga siapapun, dimanapun dan kapanpun, mempunyai kesempatan untuk mengakses kawasan-kawasan vital tersebut.Untuk menjaga keamanan dan kerahasiaan pesan, data, atau informasi dalam suatu jaringan komputer maka diperlukan beberapa enkripsi guna membuat pesan, data, atau informasi agar tidak dapat di baca atau dimengerti oleh sembarang orang, kecuali untuk penerima yang berhak.[KRI03]

Pengamanan pesan, data, atau informasi selain bertujuan untuk meningkatkan keamanan, juga berfungsi untuk:

a. Melindungi pesan, data, atau informasi agar tidak dapat di baca oleh orang-orang yang tidak berhak.
b. Mencegah agar orang-orang yang tidak berhak, menyisipkan atau menghapus pesan, data, atau informasi.

Informasi dibagi menjadi dua bagian yaitu informasi yang bersifat pribadi dan informasi yang bersifat umum. Informasi yang bersifat pribadi maksudnya informasi yang terkandung hanya untuk satu orang sedangkan informasi yang bersifat umum yaitu informasi yang dapat diketahui oleh orang banyak. Adapun perjalanan informasi tersebut tidak luput dari gangguan-gangguan pihak yang tidak berhak. Salah satu ilmu untuk menjaga keamanan dan kerahasiaan data atau informasi yaitu *Kriptografi.*

Algoritma kriptografi pertama kali dikembangkan untuk mengizinkan organisasi tertentu yang ditunjuk untuk mengakses suatu informasi. Pertama kalinya algoritma kriptografi ini digunakan untuk petunjuk dalam perang. Julius Caesar dikenal sebagai orang yang pertama kali telah mengembangkan algoritma kriptografi untuk mengirimkan pesan ke tentaranya.

Algoritma kriptografi terdiri dari algoritma *enkripsi (E)* dan algoritma *dekripsi (D).* Enkripsi adalah proses penguraian/pengacakan informasi yang mana menyandikan dari informasi aslinya agar tidak bisa dibaca atau tidak bisa dilihat. Sedangkan

ISBN: 979-756-061-6

dekripsi adalah proses mengembalikan informasi teracak ke bentuk karakter aslinya. Enkripsi dimaksudkan untuk melindungi informasi agar tidak terlihat oleh orang atau pihak yang tidak berhak.

Ada banyak model dan metode enkripsi, salah satu di antaranya adalah enkripsi dengan algoritma *Rivest Code 6* (RC6). Model ini merupakan salah satu algoritma kunci simetris yang berbentuk *block chiper*. Algoritma RC6 merupakan salah satu kandidat *Advanced Encryption Standard* (AES) yang diajukan *RSA Security Laboratories* kepada NIST. Algoritma ini merupakan pengembangan algoritma sebelumnya yaitu RC5 dan telah memenuhi semua kriteria yang diajukan oleh NIST.

2. TUJUAN

Pada penelitian ini akan dibahas tentang sejauh mana konsep dasar kriptografi serta implementasi dari algoritma RC 6 dalam proses enkripsi dan dekripsi data. Implementasi diwujudkan dalam sebuah aplikasi yang dapat digunakan untuk melakukan proses enkripsi dan dekripsi data/file. Untuk melihat sejuah mana kinerja dari algoritma RC6, maka akan dilakukan sejumlah analisa terkait dengan proses enkripsi dan dekripsi dengan sejumlah kondisi, antara lain : waktu proses, ukuran file, panjang kunci, proses dengan latar belakang sejumlah aplikasi yg berjalan serta analisa singkat terkait dengan perbandingan dengan algoritma RC4 dan Blowfish.

3. DASAR TEORI

3.1 Konsep Dasar Kriptografi

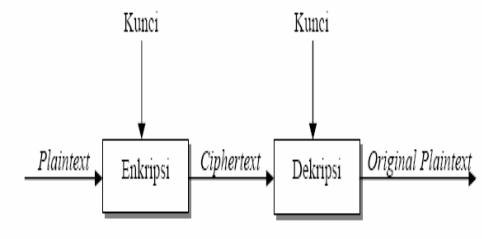
Kriptografi berasal dari bahasa Yunani yaitu *kryptos* yang artinya "yang tersembunyi" dan *graphein* yang artinya "tulisan", jadi kriptografi adalah seni dan ilmu untuk menjaga keamanan data. Dan ahlinya disebut sebagai *cryptographer. Cryptanalst* merupakan orang yang melakukan *cryptanalysis,* yaitu seni dan ilmu untuk membuka *ciphertext* menjadi *plaintext* tanpa melalui cara yang seharusnya. Data yang dapat dibaca disebut *plaintext* dan teknik untuk membuat data tersebut menjadi tidak dapat dibaca disebut *enkripsi.* Data hasil dari enkripsi disebut *ciphertext,* dan proses untuk mengembalikan *ciphertext* menjadi *plaintext* disebut *dekripsi.* Cabang matematika yang mencakup kriptografi dan *cryptanalysis* disebut *cryptology* dan pelakunya disebut *cryptologist.*

Sistem kriptografi atau *cryptosystem* adalah sebuah algoritma ditambah semua kemungkinan *plaintext*, *ciphertext* dan kunci [SCH96]. Dalam sistem ini, seperangkat parameter yang menentukan transformasi pengkodean tertentu disebut suatu set kunci. Proses enkripsi dan dekripsi diatur oleh satu atau beberapa kunci kriptografi. Secara umum, kunci-kunci yang digunakan untuk proses enkripsi dan dekripsi tidak perlu identik, tergantung pada sistem yang digunakan.

Setiap algoritma kriptografi terdiri algoritma enkripsi (E) dan algoritma dekripsi (D). Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu himpunan yang berisi elemen *plaintext* dan himpunan yang berisi elemen *ciphertext*. Enkripsi dan deskripsi merupakan fungsi tranformasi antara dua himpunan tersebut. Secara umum dapat digambarkan secara matematis sebagai berikut:

$E_k(P) = C \longrightarrow$ (Proses Enkripsi)
$D_k(C) = P \longrightarrow$ (Proses Dekripsi)
$D_k(E(P)) = P$ (Proses Dekripsi)

Dalam proses tersebut, *plaintext* disandikan dengan P dengan suatu kunci K lalu dihasilkan pesan C. Pada proses dekripsi, C diuraikan dengan menggunakan kunci K sehingga menghasilkan M yang sama dengan sebelumnya.



Gambar 1. Cryptosystem

Setiap *cryptosystem* yang baik memiliki karakteristik sebagai berikut:

a. Keamanan sistem terletak pada kerahasiaan kunci dan bukan pada kerahasian algoritma yang digunakan.
b. *Cryptosystem* yang baik memiliki ruang kunci (*keyspace)* yang besar.
c. *Cryptosystem* yang baik akan menghasilkan *ciphertext* yang terlihat acak dalam seluruh test statistik yang dilakukan.
d. *Cryptosystem* yang baik mampu menahan seluruh serangan yang telah dikenal sebelumnya.

Namun demikian, perlu diperhatikan bahwa bila suatu *cryptosystem* berhasil memenuhi seluruh karakteristik di atas, belum tentu ia merupakan sistem yang baik. Banyak *cryptosystem* lemah yang terlihat baik pada awalnya. Kadang kala untuk menunjukkan bahwa suatu *cryptosystem* kuat atau baik dapat dilakukan dengan menggunakan pembuktian matematika.[WAH03]

3.2 Aspek–Aspek Keamanan

Kriptografi tidak hanya memberikan kerahasiaan dalam telekomunikasi, namun juga melibatkan sejumlah aspek, yaitu: [KUR04]

a. *Authentication***.** Penerima pesan dapat memastikan keaslian pengirimnya. Penyerang tidak dapat perpura-pura sebagai orang lain.
b. *Integrity***.** Penerima harus dapat memeriksa apakah pesan telah dimodifikasi ditengah dijalan atau tidak. Seorang penyusup seharusnya tidak dapat memasukan tambahan ke dalam pesan, mengurangi atau mengubah pesan selama data berada di perjalanan.
c. *Nonrepudiation***.** Pengirim seharusnya tidak dapat mengelak bahwa dialah pengirim pesan yang sesungguhnya. Tanpa kriptografi, seseorang dapat mengelak bahwa dia yang mengirim email yang sesungguhnya.
d. *Authority***.** Informasi yang berada pada sistem jaringan seharusnya hanya dapat dimodifikasi oleh pihak yang berwenang.

Suatu algoritma dikatakan aman, bila tidak ada cara ditemukan *plaintext*nya. Karena selalu terdapat kemungkinan ditemukannya cara baru untuk menembus algoritma kriptografi, maka algoritma kriptografi yang dikatakan "cukup" atau "mungkin" aman, bila memiliki keadaan sebagai berikut: [KUR04]

a. Bila harga untuk menjebol algoritma lebih besar daripada nilai informasi yang dibuka, maka algoritma itu cukup aman.
b. Bila waktu yang digunakan untuk membobol algoritma tersebut lebih lama daripada lamanya waktu yang diperlukan oleh informasi tersebut harus tetap aman, maka algoritma tersebut mungkin aman.
c. Bila jumlah data yang dienkrip dengan kunci dan algoritma yang sama lebih sedikit dari jumlah data yang diperlukan untuk menembus algoritma tersebut, maka algoritma itu aman.

3.3 Algoritma Kriptografi

Perkembangan algoritma kriptografi dapat kita bagi menjadi dua, yaitu: [KUR04]

- Kriptografi Klasik
- Kriptografi Modern

3.3.1 Kriptografi Klasik

Pada algoritma klasik, diterapkan teknik enkripsi konvensional (simetris). Algoritma ini merupakan algoritma kriptografi yang biasa digunakan orang sejak berabad-abad yang lalu. Dua teknik dasar yang biasa digunakan, yaitu: [KUR04]

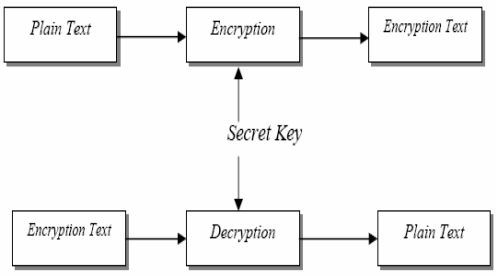
a. Teknik Subsitusi: penggantian setiap karakter *plaintext* dengan karakter lain.
b. Teknik Transposisi: Teknik ini menggunakan permutasi karakter.

Kombinasi subsitusi dan transposisi yang komplek menjadi dasar pembentukan algoritma-algoritma kriptografi modern.

3.3.2 Kriptografi Modern

Algoritma modern selain memfokuskan diri pada tingkat kesulitan algoritma juga pada kunci yang digunakan. Macam-macam algoritma menurut kuncinya adalah algoritma simetris dan algoritma asimetris.

Algoritma simetris disebut juga sebagai algoritma konvensional, yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan deskripsinya. Keamanan algoritma simetris tergantung pada kuncinya. Algoritma simetris sering juga disebut algoritma kunci rahasia, algoritma kunci tunggal atau algoritma satu kunci. Dua kategori yang termasuk pada algoritma simetris ini adalah algoritma *block cipher* dan *stream cipher*.



Gambar 2. Algoritma kriptografi simetris

Algoritma *block cipher* adalah algoritma yamg masukan dan keluarannya berupa satu *block*, dan setiap *block*nya terdiri dari banyak bit. Beberapa mode operasi enkripsi *block cipher*: [KUR04]

a. Data Enkripsi Standard (DES)
   Algoritma enkripsi yang paling banyak digunakan di dunia adalah DES yang telah diadopsi oleh NIST (*Nasional Institude of Standard and Tecnology*) sebagai standard pengolahan informasi Federal AS. Data dienkrip dalam *block-block* 64 bit menggunakan kunci 56 bit. Algoritma DES berasal dari algoritma Lucifer buatan IBM. Algoritma ini ditawarkan kepada NIST dan menjadi DES tahun 1977. Akan tetapi terdapat dua masalah besar pada algoritma ini. Pertama, kunci yang hanya 56 bit, sehingga sangat rawan terhadap serangan *brute force*. Kedua, desain struktur internal DES dimana bagian subsitusinya (S-box) masih dirahasiakan.

ISBN: 979-756-061-6

b. AES (Advanced Encrytion Standard)
Sekitar tahun1990-an ketika semakin banyak komputer yang dapat menembus kunci DES yang disebabkan terlalu pendeknya panjang kunci. Dalam kriptografi modern, panjang kunci dalam ukuran jumlah bit yang digunakan, merupakan salah satu faktor yang sangat penting. Hal ini dikarenakan penggunaan komputer yang sangat intensif dalam dunia kriptografi. Untuk itu NIST mengadakan sebuah kontes untuk mencari sebuah algoritma standard baru untuk menggantikan DES. Pada bulan oktober 2000, *Rijndael* dipilih menjadi pemenang kontes AES. Algoritma *Rijndael* dipilih bukan karena yang paling aman, melainkan karena keseimbangan antara keamanan dan fleksibilitas dalam berbagai platform *software* dan *hardware*. Algoritma menyingkirkan saingan terdekatnya yaitu Algoritma RC6 buatan RSA.

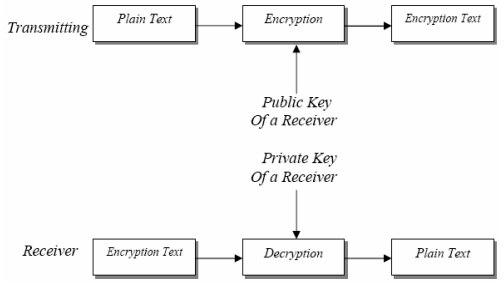
c. *Blowfish*
Blowfish merupakan *block cipher* 64-bit dengan panjang kunci variabel. Algoritma ini terdiri dari dua bagian: *key expansion* dan enkripsi data. *Key expansion* merubah kunci yang dapat mencapai 448 bit menjadi beberapa array subkunci (*subkey*) dengan total 4168 *byte*. Enkripsi data terdiri dari iterasi fungsi sederhana sebanyak 16 kali. Setiap putaran terdiri dari permutasi kunci-*dependent* dan substitusi kunci dan data *dependent*. Semua operasi adalah penambahan dan XOR pada variable 32-bit. Tambahan operasi lainnya hanyalah empat penelusuran tabel (*table lookup*) array berindeks untuk setiap putaran. *Blowfish* menggunakan subkunci yang besar. Kunci ini harus dihitung sebelum enkripsi atau dekripsi data. [SCH01]

Algoritma *Stream cipher* adalah *cipher* yang berasal dari hasil XOR antara bit *plaintext* dengan setiap bit kuncinya. *Stream cipher* sangat rawan terhadap *attack* pembalikan bit. Beberapa model algoritma *stream cipher* antara lain:

a. One Time Pad (OTP)
Dalam OTP terdapat teknik enkripsi yang sempurna. Ditemukan oleh Mayor J Maugborne dan G Verman tahun 1917. Setiap kunci hanya digunakan untuk sekali pesan. Teknik ini dikatakan sempurna di karena kunci yang acak dan hanya digunakan sekali. Untuk membangkitkan kunci OTP diperlukan pembangkit bilangan acak yang tidaklah mudah.

b. *Rivest Code 4* (RC 4)
RC4 merupakan salah satu algoritma kunci simetris yang berbentuk *stream cipher,* yaitu memproses unit atau *input* data pada satu saat. Unit atau data pada umumnya sebuah *byte* atau kadang-kadang bit. Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah *input* data tertentu sebelum diproses, atau menambahkan *byte* tambahan untuk mengenkripsi. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA. RC4 merupakan enkripsi *stream simetrik proprietary* yang dibuat oleh *RSA Data Security Inc* (RSADSI). Penyebarannya diawali dari sebuah source *code* yang diyakini sebagai RC4 dan dipublikasikan secara '*anonymously*' pada tahun 1994. Algoritma yang dipublikasikan ini sangat identik dengan implementasi RC4 pada produk resmi. RC4 digunakan secara luas pada beberapa aplikasi dan umumnya dinyatakan sangat aman. Sampai saat ini diketahui tidak ada yang dapat memecahkan/membongkarnya. RC4 tidak dipatenkan oleh RSADSI, hanya saja tidak diperdagangkan secara bebas (*trade secret*). Algoritma RC4 bekerja pada dua tahap, menyetem susunan (*key setup*) dan pengkodean (*ciphering*). Kunci susunan merupakan hal yang lebih awal dan merupakan tahap yang paling sulit dari algoritma ini. Selama menyetem susunan suatu N-bit (N menjadi panjangnya kunci), kunci enkripsi digunakan untuk menghasilkan suatu variabel enkripsi yang menggunakan dua arrays, state dan kunci, dan jumlah N dari operasi pencampuran. Operasi pencampuran terdiri dari menukar *byte*s, modulo operasi, dan rumusan lain. Suatu modulo operasi adalah proses sisa dari suatu hasil divisi. Sebagai contoh, 11/4 adalah 2 sisa 3; oleh karena itu 11 mod 4 sama dengan 3. [WAH04].

Algoritma asimetrik atau biasa disebut algoritma kunci publik dirancang sedemikian sehingga kunci yang digunakan untuk mengenkripsi dan mendekripsi berbeda. Selanjutnya kunci dekripsi tidak dapat dihitung dengan dari kunci enkripsi. Algoritma tersebut disebut *public-key* karena kunci enkripsi dapat dibuat secara *public*. Orang asing dapat menggunakan kunci enkripsi tersebut untuk mengenkripsi sebuah pesan, tetapi hanya orang tertentu dengan kunci dekripsi sepadan dapat mendekripsi pesan tersebut. Dalam sistem ini kunci enkripsi sering disebut *public key* sedangkan key dekripsi sering disebut *private key.*



Gambar 3. Algoritma kriptografi Asimetris

Beberapa algoritma asimetrik antara lain:

a. RSA
   RSA tidak pernah dibuktikan aman tidaknya, hanya karena sulitnya pemfaktoran bilangan yang sangat besar, maka RSA dianggap aman. Dalam pembangkitan kedua kunci, digunakan dua bilangan prima acak yang sangat besar.
b. Diffie-Hellman (DH)
   Diffie Helman dianggap merupakan algoritma asimetrik yang pertama kali ditemukan pada tahun 1976, meskipun NSA telah mengaku menemukan algoritma asimetrik jauh-jauh hari sebelumnya. Algoritma ini memperoleh keamanannya dari sulitnya menghitung logaritma diskrit pada bilangan yang amat besar. Akan tetapi algoritma ini hanya dapat digunakan untuk pertukaran kunci (simetris) dan tidak dapat digunakan untuk enkripsi/dekripsi maupun untuk tandat tangan digital.

3.4 Algoritma RC 6

Algoritma RC6 merupakan salah satu kandidat *Advanced Encryption Standard* (AES) yang diajukan oleh *RSA Laboratoriest* kepada NIST. Dirancang oleh Ronald L Rivest, M.J.B. Robshaw, R. Sidney dan Y.L. Yin, algoritma ini merupakan pengembangan dari algoritma sebelumnya yaitu RC5 dan telah memenuhi semua kriteria yang diajukan oleh NIST.

Algoritma RC6 adalah versi yang dilengkapi dengan beberapa parameter, sehingga dituliskan sebagai RC6-w/r/b, dimana parameter w merupakan ukuran kata dalam satuan bit, r adalah bilangan bulat bukan negatif yang menunjukan banyaknya iterasi selam proses enkripsi, dan b menunjukan ukuran kunci enkripsi dalam *byte*. Ketika algoritma ini masuk sebagai kandidat AES, maka ditetapkan nilai parameter w = 32, r = 20 dan b bervariasi antara 16, 24, dan 32 *byte*.[ABD02]

RC6-w/r/b memecah *block* 128 bit menjadi 4 buah *block* 32 bit, dan mengikuti enam aturan operasi dasar sebagai berikut :

A + B	Operasi penjumlahan bilangan integer.
A – B	Operasi pengurangan bilangan integer.
A B	Operasi *exclusive*-OR (XOR)
A x B	Operasi perkalian bilangan integer.
A<<<B	A dirotasikan ke kiri sebanyak variabel kedua (B)
A>>>B	A dirotasikan ke kanani sebanyak variabel kedua (B)

3.4.1 Enkripsi Algoritma RC 6

Karena RC6 memecah *block* 128 bit menjadi 4 buah *block* 32 bit, maka algoritma ini bekerja dengan 4 buah register 32-bit A, B, C, D. *Byte* yang pertama dari *plaintext* atau *ciphertext* ditempatkan pada *byte* A, sedangkan *byte* yang terakhirnya ditempatkan pada *byte* D. Dalam prosesnya akan didapatkan (A, B, C, D) = (B, C, D, A) yang diartikan bahwa nilai yang terletak pada sisi kanan berasal dari register disisi kiri.[ABD02]. Berikut ini adalah algoritma enkripsi RC6:

```
B = B + S[ 0 ]
D = D + S[ 1 ]
for  i  =  1  to  20  do
    {
        t  =  ( B  x  ( 2B  + 1 ) )
<<<  5
        u  =  ( D  x  ( 2D + 1 ) )
<<<  5
        A  =  ( ( A ⊕ t )  <<<  u )
+  S[ 2i ]
        C  =  ( ( C  ⊕ u )  <<<  t )
+  S[ 2i + 1 ]
        (A, B, C, D)  =  (B, C, D, A)
      }
A = A + S[ 42 ]
C =  C + S[ 43 ]
```

Algoritma RC6 menggunakan 44 buah sub kunci yang dibangkitkan dari kunci dan dinamakan dengan S[0] hingga S[43]. Masing-masing sub kunci panjangnya 32 bit. Proses enkripsi pada algoritma RC6 dimulai dan diakhiri dengan proses *whitening* yang bertujuan untuk menyamarkan iterasi yang pertama dan yang terakhir dari proses enkripsi dan dekripsi. Pada proses *whitening* awal, nilai B akan dijumlahkan dengan S[0], dan nilai D dijumlahkan dengan S[i]. Pada masing-masing iterasi pada RC6 menggunakan 2 buah sub kunci. Sub kunci pada iterasi yang pertama menggunakan S[2] dan S[3], sedangkan iterasi-iterasi berikutnya menggunakan sub-sub kunci lanjutannya. Setelah iterasi ke-20 selesai, dilakukan proses *whitening* akhir dimana nilai A dijumlahkan dengan S[42], dan nilai C dijumlahkan dengan S[43].[ABD02]

Setiap iterasi pada algoritma RC6 mengikuti aturan sebagai berikut, nilai B dimasukan ke dalam fungsi f, yang didefinisikan sebagai f(x) =x(2x+1) , kemudian diputar kekiri sejauh lg-w atau 5 bit. Hasil yang didapat pada proses ini dimisalkan sebagai u. Nilai u kemudian di XOR dengan C dan hasilnya menjadi nilai C. Nilai t juga digunakan sebagai acuan bagi C untuk memutar nilainya kekiri. Begitu pula dengan nilai u, juga digunakan sebagai acuan bagi nilai A untuk melakukan proses

pemutaran kekiri. Kemudian sub kunci S[2i] pada iterasi dijumlahkan dengan A, dan sub kunci S[2i+1] dijumlahkan dengan C. keempat bagian dari *block* kemudian akan dipertukarkan dengan mengikuti aturan, bahwa nilai A ditempakan pada D, nilai B ditempatkan pada A, nilai C ditempatkan pada B, dan nilai (asli) D ditempatkan pad C. demikian iterasi tersebut akan terus berlangsung hingga 20 kali.[ABD02]

3.4.2 Dekripsi Algoritma RC 6

Proses dekripsi *ciphertext* pada algoritma RC6 merupakan pembalikan dari proses enkripsi. Pada proses *whitening*, bila proses enkripsi menggunakan operasi penjumlahan, maka pada proses dekripsi menggunakan operasi pengurangan. Sub kunci yang digunakan pada proses *whitening* setelah iterasi terakhir diterapkan sebelum iterasi pertama, begitu juga sebaliknya sub kunci yang diterapkan pada proses *whitening* sebelum iterasi pertama digunakan pada *whitening* setelah iterasi terkahir. Akibatnya, untuk melakukan dekripsi, hal yang harus dilakukan semata-mata hanyalah menerapkan algoritma yang sama dengan enkripsi, dengan tiap iterasi menggunakan sub kunci yang sama dengan yang digunakan pada saat enkripsi, hanya saja urutan sub kunci yang digunakan terbalik.[ABD02]. Berikut ini adalah algoritma deskripsi RC6:

```
C =  C – S[ 43 ]
A =  A – S[ 42 ]
for  i  =  20  downto  1  do
    {
        (A, B, C, D)  =  (D, A, B, C)
        u  =  ( D  x  ( 2D + 1 ) )
<<<  5
        t  =  ( B  x  ( 2B  + 1 ) )
<<<  5
        C  =  ( ( C – S[ 2i + 1 ] )
>>> t ) ⊕ u
        A  =  ( ( A – S[ 2i ] ) >>> u
) ⊕ t
    }
 D = D – S[ 1 ]
 B = B – S[ 0 ]
```

3.4.3 Pembangkitan Kunci

Pengguna memasukkan sebuah kunci yang besarnya b *byte*, dimana 0≤b≤255. *byte* kunci ini kemudian ditempatkan dalam array c w-bit words L[0]…L[c-1]. *Byte* pertama kunci akan ditempatkan sebagai pada L[0], *byte* kedua pada L[1], dan seterusnya. (Catatan, bila b=0 maka c=1 dan L[0]=0). Masing-masing nilai kata w-bit akan dibangkitkan pada penambahan kunci *round* 2r+4 dan akan ditempatkan pada array S[0,…,2r+3].[ABD02]

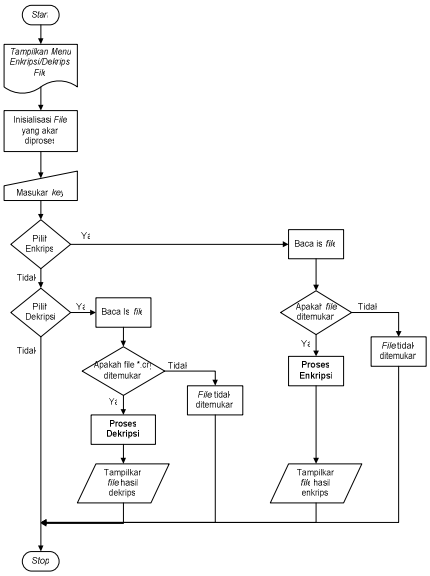
Konstanta P32 = B7E15163 dan Q32 = 9E3779B9 (dalam satuan heksadesimal) adalah "konstanta ajaib"yang digunakan dalam penjadwalan kunci pada RC6. Nilai P32 diperoleh dari perluasan bilangan biner e-2, dimana e adalah sebuah fungsi logaritma. Sedangkan nilai Q32 diperoleh dari perluasan bilangan biner ∅-1, dimana ∅ dapat dikatakan sebagai "*golden ratio*" (rasio emas). Algoritma untuk pembangkitan kunci RC6 adalah sebagai berikut:

```
S[ 0 ] = 0xB7E15163
for  i = 1  to  43  do S[i] = S[i-1]
+ 0x9E3779B9
A = B = i = j = 0
for  k = 1  to  132  do
    {  A = S[ i ] = ( S[ i ] + A + B
) <<< 3
       B = L[ j ] = ( L[ j ] + A + B
) <<< ( A + B )
       i = ( i + 1 )   mod 44
       j  =  ( j  +  1 )    mod  c
}
```

4. PERANCANGAN DAN IMPLEMENTASI

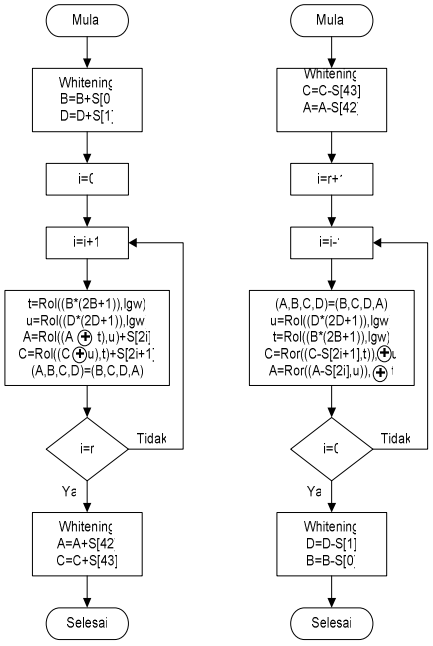
4.1 Desain dan Rancangan Aplikasi

Block diagram dari program implementasi secara umum algoritma RC6 yang akan dibuat diperlihatkan pada gambar 4.1. Pertama-tama yang dilakukan adalah inisialisasi data asli. Data ini akan disandikan/dienkripsi dengan menggunakan algoritma RC6 kemudian hasilnya akan di tampilkan, dari data yang telah disandikan akan dikembalikan ke data asli dengan proses dekripsi. Proses enkripsi/dekrispi memerlukan masukan/*input key* yaitu berupa karakter ASCII. *Key input* digunakan untuk masukan baik sebelum enkripsi maupun dekripsi. Operasi *key* dimasukan sebagai kunci untuk proses dekripsi atau pengubahan data sandi ke data asli, data yang disandikan tidak dapat dikembailkan ke data asli jika kita tidak mempunyai kunci atau *key* yang pernah dimasukan pada waktu proses enkripsi.



Gambar 4. Gambaran implementasi program secara umum

ISBN: 979-756-061-6

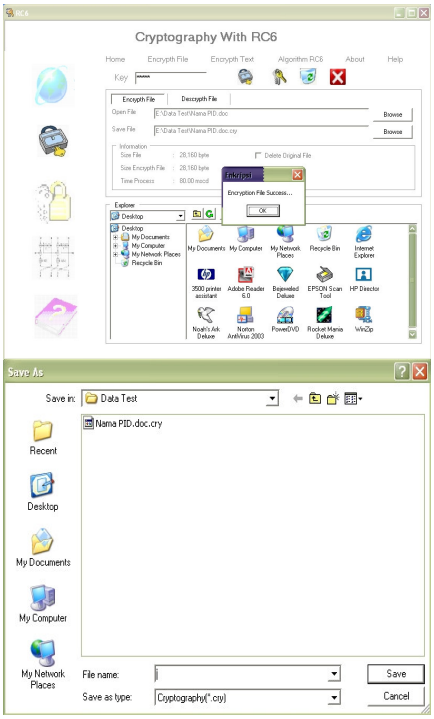


Gambar 5. *Flowchart* untuk proses enkripsi(a) dan dekripsi (b) *file*/*text*

4.2 Implementasi Hasil

Implementasi aplikasi terdiri dari *browse file* yang akan dienkripsi, *save file* hasil enkripsi, *browse file* enkripsi yang akan dideskripsi, *save file* hasil dekripsi, tombol enkripsi *file*, tombol dekripsi *file*, clear *file*, chek boks untuk pilihan hapus *file*, serta informasi *file* yang terdiri nama *file* yang akan dienkripsi dan dekripsi, ukuran *file* dan waktu proses enkripsi dan dekripsi *file*. Subrutin program enkripsi dan dekripsi *file* ini:

a. Proses Enkripsi *File*
   Proses ini merupakan proses untuk enkripsi *file*. Operasi program dimulai dari inisialisasi masukkan program berupa *file* atau data elektronik kemudian pemberian *key* (kunci) dan setelah proses enkripsi yang akan menyimpan *file* enkripsi tersebut kedalam *file* berekstensi *.cry
b. Proses Deskripsi *File*
   Proses ini adalah proses untuk dekripsi yaitu membuka kembali *file* yang telah dienkripsi. Operasi program dimulai dari inisialisasi *file* yang telah terenkripsi yaitu dengan format *.cry kemudian memasukkan *key* (kunci) untuk membuka *file* enkripsi tersebut dan menyimpannya ke dalam *file* semula
c. Informasi waktu proses dan informasi ukuran *file*.
   Waktu proses ditampilkan ketika mengenkripsi dan mendekripsi *file* yang memberikan informasi tentang kecepatan algoritma RC6, serta kapasitas *file* yang dapat dilihat pada informasi *file* atau pada *objek preview*.



Gambar 6. Interface Aplikasi Kriptografi Algoritma RC 6

Salah satu hasil dari proses enkripsi adalah dengan data sbb:

Key Proses Enkripsi : 1234
File yang akan dienkripsi :
E:\Data Test\Nama PID.doc
Save Enkripsi *File* :
E:\Data Test\Nama PID.doc.cry

Untuk hasil dari proses enkripsi dapat dilihat pada informasi *file* yaitu:

Ukuran *File* Original : 28,160 *bytes*
Ukuran *File* Enkripsi : 28,160 *bytes*
Waktu Proses Enkripsi : 20.00 mili second

Untuk data yang akan dimasukkan untuk proses dekripsi adalah:

Key Proses Dekripsi : 1234
File yang akan didekripsi:
E:\Data Test\Nama PID.doc.cry
Save Dekripsi *File*:
E:\DataTest\Nama PID.doc

Untuk hasil dari proses dekripsi dapat dilihat pada informasi *file* yaitu:

Ukuran *File* Enkripsi: 28,160 *bytes*
Ukuran *File* Dekripsi : 28,160 *bytes*
Waktu Proses Enkripsi: 71.00 mscd

5. ANALISA HASIL

5.1 Analisis Waktu Proses Terhadap Ukuran *File*

Waktu proses enkripsi/dekripsi *file* tergantung pada ukuran *file* yang akan dienkripsi/dekripsi. Pada uji coba proses enkripsi dengan menggunakan beberapa ekstensi *file* dengan ukuran yang berbeda–beda dapat disimpulkan bahwa semakin besar ukuran *file* maka waktu proses enkripsi/dekripsi *file* semakin lama. Hal ini disebabkan oleh efek *cache* dan efek penanganan *file* (*file handling*) oleh sistem operasi. Dapat dilihat pada Tabel .

Tabel 1. Analisis waktu proses enkripsi/dekripsi terhadap ukuran *file*

Nama *File*	Ukuran *File* (KB)	Waktu Proses Enkripsi (mscd)	Waktu Proses Dekripsi (mscd)
Nama PID.doc	28	20	20
Errorlog.log	45	80	50
Feb11^04.jpg	150	90	80
Laporan PIN.doc	250	130	120
Feb11^27.jpg	552	220	280
Progres Report TA.ppt	1,576	711	721
Hal1.jpg	2,687	1091	1222
Al Hadid.mp3	5,760	2444	2704
AR Ruum.mp3	7,913	3295	3636
Freeport.ppt	10,086	4186	4597

5.2 Analisis Ukuran *File* Terhadap Proses Enkripsi

Analisis dengan membandingkan ukuran *file* sebelum dienkripsi dan setelah *file* dienkripsi. Tabel 2 menunjukkan hasil proses enkripsi untuk beberapa jenis file.

Tabel 2. Analisis ukuran *file* terhadap proses enkripsi

Nama *File*	Ukuran *File* Original (KB)	Ukuran *File* Enkripsi (KB)
Nama PID.doc	28	28
Errorlog.log	45	45
Feb11^04.jpg	150	150
Laporan PIN.doc	250	250
Feb11^27.jpg	552	552
Progres Report TA.ppt	1,576	1,576
Hal1.jpg	2,687	2,687
Al Hadid.mp3	5,760	5,760
AR Ruum.mp3	7,913	7,913
Freeport.ppt	10,086	10,086

Dari tabel di atas dapat disimpulkan bahwa ukuran *file* sebelum proses enkripsi dengan ukuran *file* setelah proses enkripsi tidak mengalami perubahan yang mencolok. Perubahan yang terjadi sangat kecil, hal ini terjadi karena adanya prosess *padding*.

5.3 Analisis Proses Enkripsi Terhadap Panjang kunci

Pada Tabel 3 ditunjukkan hasil analisis terhadap *file* yang sama dengan menggunakan ukuran panjang kunci yang berbeda-beda untuk mengetahui pengaruh panjang kunci terhadap kecepatan proses enkripsi.

Tabel 3. Analisis proses enkripsi terhadap panjang kunci

Ukuran *File* (KB)	Panjang Kunci (karakter)	Waktu Proses (mscd)
1,576	1	661
1,576	5	631
1,576	10	641
1,576	15	651
1,576	25	621
1,576	32	631

Dari tabel di atas, terlihat bahwa dengan menggunakan *file* yang sama dengan ukuran kunci yang berbeda-beda dari 1 karakter sampai panjang kunci maksimal 32 karakter terdapat perbedaan kecepatan proses enkripsi. Kecepatan proses enkripsi dan dekripsi tidak dipengaruhi oleh panjang kunci . Hal yang mempengaruhi terjadinya perbedaan kecepatan proses yaitu, kecepatan dari sistem komputer dan adanya aplikasi atau proses lain yang sedang berjalan.

5.4 Analisis Waktu Proses Terhadap Pengaruh Aplikasi Yang Berjalan

Pada Tabel 4 akan digambarkan lebih lanjut pengaruh adanya aplikasi yang berjalan pada waktu yang dibutuhkan pada saat proses enkripsi.

Tabel 4. Analisis waktu proses terhadap pengaruh aplikasi yang berjalan

Nama *File*	Ukuran *File* (KB)	Panjang Kunci	Aplikasi Yang Berjalan			Waktu Proses (mscd)
			Winap	Windows Media	Explorer	
10.ppt	10,086	1234				3054
10.ppt	10,086	1234	●			4736
10.ppt	10,086	1234	●	●		4967
10.ppt	10,086	1234	●	●	●	5187

Dari Tabel 4 di atas, dapat terlihat jelas bahwa banyaknya aplikasi yang berjalan sangat mempengaruhi kecepatan dari proses enkripsi

maupun deskripsi. Hal ini disebabkan adanya fungsi *scedulling* pada sistem operasi yang mengatur processor untuk membagi waktu proses tiap aplikasi yang sedang berjalan. Jadi semakin banyak aplikasi yang berjalan pada sebuah komputer, maka kecepatan proses dari enkripsi akan semakin berkurang.

5.5 Analisis Perbandingan Algoritma RC6 dengan Algoritma *Blowfish* dan Algoritma RC4

Terdapat sejumlah penelitian sejenis yang telah dilakukan terkait dengan algoritma enkripsi/dekripsi data, antara lain oleh Anton Nugroho yaitu aplikasi enkripsi *file* dengan menggunakan algoritma *Blowfish* yang berbentuk *block cipher*. [NUG04],dan Dian Wahyudi dengan menggunakan algoritma RC4 yang berbentuk *stream cipher* dimana dalam proses enkripsi dan dekripsi berbeda.[WAH04].

Untuk enkripsi dengan menggunakan algoritma *blowfish* merupakan blok *cipher* 64-bit dengan panjang kunci variabel. Algoritma ini terdiri dari dua bagian: *key expansion* dan enkripsi data. *Key expansion* merubah kunci yang dapat mencapai 448 bit menjadi beberapa array subkunci (*subkey*) dengan total 4168 *byte*.[NUG04]

Model enkripsi dengan menggunakan algoritma RC4 merupakan *stream cipher* dengan panjang kunci variabel. Algoritma RC4 bekerja pada dua tahap, menyetem susunan (*key setup*) dan pengkodean (*ciphering*). Untuk proses enkripsi, setelah proses inisialisasi *S-boxes* dan *key*stream generator maka diperoleh *keystream,* keystream inilah yang kemudian di XORkan dengan *plaintext* untuk menghasilkan *ciphertext*.[WAH04]

Pada algoritma RC6 sendiri yang merupakan *block cipher* 128-bit dengan panjang kunci max128 bit. Algoritma RC6 bekerja 3 tahap yaitu *key setup, Whitening* yang berfungsi menyamarkan iterasi, *ciphering*. Karena RC6 memecah blok 128 bit menjadi 4 buah blok 32 bit, maka algoritma ini bekerja dengan 4 buah register 32-bit A, B, C, D. *Byte* yang pertama dari *plaintext* atau *ciphertext* ditempatkan pada *byte* A, sedangkan *byte* yang terakhirnya ditempatkan pada *byte* D. Dalam prosesnya akan didapatkan (A, B, C, D) = (B, C, D, A) yang diartikan bahwa nilai yang terletak pada sisi kanan berasal dari register disisi kiri.

Untuk algoritma yang berbentuk block cipher seperti *Blowfish*, *Rinjdael* (AES) dan RC6 hasil dari proses *ciphertext*nya berpola, jika ada teks yang berulang maka hasil *ciphertext*nya sama sedang untuk algoritma yang berbentuk *stream cipher* seperti RC4 hasil dari *ciphertext*nya terlihat *random* dimana untuk teks yang berulang hasilnya tidak sama dengan teks yang sebelumnya karena setiap karakter dari *plaintext* di XORkan dengan *keystream* yang berbeda. Berikut ini adalah perbandingan waktu proses enkripsi dengan menggunakan RC6, RC4 dan *Blowfish*.

Tabel 5. Perbandingan waktu proses RC6, RC4, *Blowfish*

Nama *File*	Ukuran *File*	Kunci	Waktu Proses (dtk)		
			RC6	RC4	*Blowfish*
Test.ppt	10,086 KB	1234	2,955	511	5,357
Al-Hadid.mp3	5,897 KB	1234	1,822	281	3,064
Hal1.jpg	2,687 KB	1234	1,282	134	1,442

Dari tabel diatas dapat terlihat dengan jelas kecepatan proses enkripsi algoritma RC6 lebih cepat dibandingkan dengan algoritma RC4 dan algoritma *Blowfish*.

5.6 Keamanan Algoritma RC6

Dalam algoritma enkripsi, panjang kunci yang biasanya dalam ukuran bit, juga menentukan kekuatan dari enkripsi. Kunci yang lebih panjang biasanya lebih aman daripada kunci yang pendek. Jadi enkripsi dengan menggunakan kunci 128-bit lebih sukar dipecahkan dengan algoritma enkripsi yang sama tetapi memiliki kunci 56-bit. Semakin panjang sebuah kunci, semakin besar *key*space yang harus dijalani untuk mencari kunci dengan cara *brute force attack* atau coba-coba karena *key*space yang harus dilihat merupakan pangkat bilangan dari 2. Jadi kunci 128-bit memiliki *key*space 2^{128}, sedangkan kunci 56-bit memiliki *key*space 2^{56}. Artinya semakin lama kunci baru bisa ditemukan.[AND03].

Pada intinya, keamanan suatu pesan tidak tergantung pada sulitnya algoritma tetapi pada kunci yang digunakan. Pada RC6 dengan adanya fungsi $f(x)=x(2x+1)$ yang diikuti pergeseran lima bit kekiri dapat memberi tingkat keamanan data yang tinggi. Adanya *avalanche effect* juga memberikan cukup kesulitan kepada kriptanalis untuk melakukan serangan.

Adanya konstanta P32 dan Q32 yang dikenal sebagai "kostantata ajaib" dapat membantu algoritma RC6 sehingga tidak terjadi *weak key*, yaitu terjadinya nilai yang sama pada dua entri dari s-box. Algoritma selain aman juga cepat dan mudah. Banyaknya jumlah iterasi yang berjumlah 20 *round*/iterasi memberikan azas keseimbangan pada algoritma RC6, karena jika *round* terlalu banyak akan menyebabkan kecepatan proses enkripsi dan deskripsi menjadi lambat. Dan dengan jumlah *round* yang sedikit akan menyebabkan *cipher* menjadi mudah untuk dipecahkan.

Kekurangan umum dari algoritma yang berbentuk simetris atau kunci pribadi adalah pada kunci itu sendiri. Kelemahan ini timbul jika terdapat banyak orang yang ingin saling berkomunikasi, karena setiap pasangan maupun *file* enkripsi

mempunyai *key* berbeda yang harus disepakati sehingga *key* tiap orang maupun *file* harus menghafal banyak kunci dan menggunakannya dengan tepat**.**

6. PENUTUP

Dari hasil analisis studi dan implementasi algoritma RC6 untuk enkripsi dan dekripsi data dapat diambil beberapa kesimpulan, yaitu:

a. RC6 adalah algoritma enkripsi dengan model *private key*/kunci pribadi yang mempunyai key dekripsi sama dengan key enkripsi..
b. Algoritma RC6 merupakan *block cipher* dengan ukuran *block* hingga128 bit dan parameter yaitu RC6-w/r/b dengan nilai w=32 sebagai ukuran kata dalam bit, r=20 sebagai banyaknya iterasi/*round* dan b ukuran kunci yang bervariasi antara 16, 24 dan 32 *byte.*
c. Algoritma RC6 terdiri dari 3 bagian yaitu *key setup*, *whitening* dan *ciphering.*
d. Waktu proses enkripsi dan deskripsi tergantung pada besarnya *file* yang akan dienkripsi/dekripsi. Hal ini disebabkan oleh efek *cache* dan efek penanganan *file* (*file handling*) oleh sistem operasi.
e. Ukuran *file* pada proses enkripsi/dekripsi mengalami sedikit perubahan, hal ini dikarenakan terjadinya proses *padding*.
f. Dengan menggunakan panjang kunci yang berbeda pada proses enkripsi yang dilakukan pada *file* yang sama, waktu proses tidak mengalami perubahan yang besar
g. Dibandingkan dengan algoritma RC4 dan Blowfish, maka algoritma RC6 merupakan algoritma enkripsi yang lebih *simple, fast, and secure.*

DAFTAR PUSTAKA

[ABD02] Abdurohman, Maman. *Analisis Performansi Algoritma Kriptografi RC6.* Fakultas Teknologi Informasi, T. Elektro, ITB. Bandung, 2002.

[KRI03] Kristanto, Andri. *Keamanan Data pada Jaringan Komputer*, Gava Media, Yogyakarta, 2003.

[KUR04] Kurniawan, Yusuf, Ir. *Kriptografi Keamanan Internet dan Jaringan Komunikasi.* Bandung: Informatika Bandung, 2004.

[NUG04] Nugroho, Anton, 2004, Studi dan Implementasi Algoritma Blowfish untuk Keamanan File, *Skripsi*, FTI, T. Informatika, UII. Yogyakarta. 2004

[PRA03] Pranata, Antony. *Pemrograman Borland Delphi 6.* Yogyakarta: Andi Offset, 2003.

[RIV98] Rivest L, Ronald., Robshaw, M.J.B., Sidney, R., and Yin, Y.L. *The Rc6 Block Cipher***.** San Mateo, USA: RSA Laboratories, 1998. www.rsasecurity.com/rsalabs/rc6/

[SCH96] Scheier, Bruce. *Applied Cryptography, Second Edition*. New York: John Wiley & Son, 1996.

[SCH01] Scheier, Bruce. *Decryption of A New Variable-Length Key, 64/128-Bit Block Cipher (Blowfish).* New York: John Wiley & Son, 2001.

[WAH03] Wahana. *Memahami Model Enkripsi dan Security Data*. Yogyakarta, Andi Offset, 2003.

[WAH04] Wahyudi, Dian. Implementasi Enkripsi/Dekripsi Data Pada File.Txt Dengan Menggunakan Algoritma Rivest Code 4 (Rc4)**,** *Skripsi*, FTI, T.Informatika, UII. Yogyakarta, 2004.