PENGAMANAN DATA FILE DENGAN MENGGUNAKAN ALGORITMA ENKRIPSI RIVEST CODE 5

[1)] Sayekti Harits Suryawan, [2)] Hamdani

[1,2)]Program Studi Ilmu Komputer, FMIPA Universitas Mulawarman
Email : [1)] zetsubou.harits@gmail.com, [2)] hamdani@unmul.ac.id

ABSTRAK

Data sebagai salah satu wadah untuk menyimpan atau menyampaikan informasi telah menjadi kebutuhan primer masyarakat. Permasalahan keamanan yang muncul dalam proses pertukaran file yang berisi data rahasia dapat diatasi dengan melakukan pengacakan data yang disebut enkripsi data. Dalam penelitian algoritma enkripsi yang digunakan adalah algoritma RC5 (*Rivest Code* 5). RC5 adalah algoritma enkripsi blok chiper simetris cepat yang sangat cocok diimplemantasikan pada *hardware* maupun *software*. Alasan digunakannya algoritma RC5 adalah karena dalam segi keamanannya yang baik selain itu algoritma RC5 juga ringan, cepat serta memiliki kompatibilitas yang baik.

Dengan melakukan enkripsi dan dekripsi pada bit data pada file, maka file yang dienkripsi tidak dapat dibuka jika tidak melakukan dekripsi terlebih dahulu dengan kunci dan metode yang sesuai. Dengan demikian diharapkan dapat melakukan pencegahan atas diaksesnya data pada file yang bersifat rahasia oleh orang-orang yang tidak berwenang. Jenis file yang dapat dienkripsi tidak dibatasi, dengan kata lain pada hasil penelitian dapat dilakukan proses enkripsi pada semua jenis file. Pada penelitian disimpulkan bahwa Enkripsi dapat dilakukan pada bit data file yang menyebabkan struktur file teracak sehingga file tidak dapat dibaca tanpa proses dekripsi terlebih dahulu menggunakan aplikasi tersebut dan kunci yang benar.

Kata Kunci : *Enkripsi, Data, File, RC5.*

PENDAHULUAN

Kebutuhan untuk data yang menjadi kebutuhan primer di kalangan masyarakat menjadikan data sebagai salah satu tempat untuk penyimpanan ataupun menyampaikan informasi. Data sendiri dapat berupa catatan-catatan di kertas, buku, atau tersimpan kedalam file di komputer maupun dalam database. File merupakan jenis data digital yang sering digunakan dalam melakukan penyimpanan atau penyaluran informasi.

Ada kalanya data dapat bersifat rahasia sehingga tidak semua pihak dapat melihat atau mengakses data tersebut. Hal tersebut membuat keamanan data menjadi sangat penting dalam proses penyimpanan maupun pendistribusian data. Penyaluran informasi dengan menggunakan data berupa file dapat dilakukan baik melalui media internet dengan fasilitas e-mail, transfer data antar perangkat mobile seperti smartphone dan flashdisk, transfer data melalui gelombang radio seperti Bluetooth dan GPRS (*General Packet Radio Service*), maupun melalui jaringan komputer.

Algoritma enkripsi RC5 (*Rivest Code* 5) adalah blok chiper simetris cepat yang sangat cocok untuk diimplementasikan pada *hardware* maupun *software* [1]. Salah satu fitur yang terkenal dari metode RC5 adalah penggunaan rotasi data-dependent yang cukup banyak. Selain algoritma yang cukup simpel dan mudah untuk dimengerti, berdasarkan laporan yang dituliskan oleh Burton dalam On the Security of the RC5 Encryption Algorithm RSA Laboratories Technical Report TR-602, tingkat keamanan RC5 sangat tinggi. Hal tersebut dikarenakan RC5 dengan 12 rounds dan 64 bit blok data memiliki tingkat keamanan yang sama dengan DES (*Data Encryption Standard*) dari serangan kriptoanalitik dari 244 pasangan plaintext yang telah dipilih untuk RC5 dan 243 untuk DES namun dengan kecepatan yang lebih dari DES. Rehman, S.U pada tahun 2012 dalam artikel yang berjudul Comparison Based Analysis of Different Cryptographic and Encryption Techniques Using Message Authentication Code (MAC) in Wireless Sensor Networks (WSN) pada International Journal of Computer Science Issues juga menyimpulkan bahwa RC5 merupakan algoritma enkripsi yang layak dan menggunakan resource atau sumber daya yang lebih sedikit dibandingkan dengan algoritma lain seperti AES (*Advanced Encryption Standard*), MD5 (*Message Digest 5*), SHA1 (*Secure Hash Algorithm* 1), IDEA (*International Data Encryption*

Algorithm) kecuali Skipjack dan XXTEA(XX-Tiny Encryption Algorithm).

Pengamanan data berupa file atau berkas dapat dilakukan dengan RC5 dapat dilakukan dengan implementasi RC5 pada data biner file tersebut. Sehingga, file data biner terenkripsi tidak dapat dibaca oleh komputer sebelum file tersebut didekripsikan kembali. Hal tersebut terjadi dikarenakan pada proses enkripsi, akan terjadi pengacakan data biner pada file tersebut sehingga informasi penting yang digunakan aplikasi pada komputer untuk membaca file tersebut tidak valid dan file akan dianggap corrupt (rusak), ataupun tidak dikenali oleh sistem yang akan membaca file tersebut pada komputer.

Pengamanan terhadap data berupa file yang dilakukan dengan cara melakukan enkripsi pada data binner file tersebut sudah selayaknya menggunakan metode yang dapat dieksekusi dengan cepat, ringan, dan aman. Sehingga metode RC5 sangat cocok untuk diimplementasikan membangun sistem keamanan data yang berupa file tersebut. Pada artikel Hamdani dalam pengujian Algoritma enkripsi RC5 (*Rivest Code* 5) pada file dokumen dapat digunakan untuk mengacak kode biner pada file tersebut, RC5 adalah sutu blok chiper simetris cepat yang sangat cocok untuk diimplementasikan pada *hardware* maupun *software* [2]. Sehingga pada penelitian ini dapat digunakan untuk mengenkripsi file agar file tidak dapat digunakan oleh sistem operasi atau aplikasi yang berjalan pada sistem komputer.

ALGORITMA ENKRIPSI RC5

Algoritma enkripsi RC5 didesain oleh Profesor Ronald Rivest dan pertama kali dipublikasikan pada Desember 1994. Sejak publikasinya, RC5 telah menarik perhatian banyak peneliti dalam bidang kriptografi dalam rangka menguji tingkat keamanan yang ditawarkan oleh algoritma RC5 (RSA Laboratory Technical Report TR-602). Pada dasarnya RC5 di desain dengan sedemikian rupa untuk memenuhi syarat-syarat sebagai, [1,2]:

1. RC5 harus merupakan blok chiper simetris. Kunci rahasia yang sama dalam kriptografi digunakan dalam enkripsi dan dekripsi. Teks awal dan teks terenkripsi memiliki panjang yang sudah ditentukan dalam blok.
2. RC5 harus cocok untuk *hardware* ataupun *software*. Hal tersebut berarti RC5 hanya akan menggunakan operasi perhitungan primitif yang sering kali ditemukan pada mikroprosessor.
3. RC5 harus cepat, hal tersebut lebih kurang dikarenakan RC5 merupakan algoritma *word-oriented* dengan kata lain pada operasi komputasi dasar yang digunakan harus dapat memproses data *word* secara penuh dalam satu waktu.
4. RC5 harus dapat beradaptasi pada berbagai panjang data word. Semisal pada prosessor 64 bit, panjang data *word* yang digunakan lebih panjang daripada prosesor 32 bit. RC5 harus dapat memanfaatkan hal tersebut, oleh karenanya RC5 memiliki parameter *w* yang yang menandakan panjang *word*.
5. RC5 harus dapat beroperasi dalam berbagai jumlah *round*. Jumlah *round* yang bervariasi memungkinkan pengguna untuk memanipulasi RC5 untuk menjadi lebih cepat dan aman.
6. RC5 harus dapat beroperasi dalam berbagai panjang kunci. Hal tersebut mengakibatkan panjang kunci *b* menjadi parameter dalam algoritma RC5.
7. RC5 haruslah berstruktur sederhana. Struktur yang sederhana tersebut belum tentu menghasilkan kemanan yang rendah. Namun, struktur sederhana akan memugkinkan analisis dan eveluasi yang cepat untuk menentukan kekuatan algoritma.
8. RC5 harus hemat dalam penggunaan memori. Hal tersebut memungkinkan implemantasi RC5 kedalam smart-card atau perangkat lain yang memiliki keterbatasan memori.
9. RC5 harus mengimplementasikan metode *data-dependent rotations*. Metode RC5 merupakan kriptografi primitif yang merupakan sasaran pengkajian RC5. *Data-dependent rotations* merupakan suatu teknik yang merotasi data secara sirkuler sebanyak N rotasi.

Seperti yang dijelaskan sebelumnya, algoritma RC5 merupakan metode enkripsi menggunakan metode simetrik dan pengolahan dalam bentuk blok *chiper*, jadi kata kunci yang sama digunakan untuk proses enkripsi dan dekripsi. Parameter-parameter yang digunakan dalam RC-5 adalah sebagai berikut:

1. *Round* atau jumlah putaran disimbolkan dengan *r* yang memiki nilai antara 1, 2, 3, 4, ..., 225.
2. Jumlah *word* dalam bit disimbolkan dengan w. Jumlah yang disupport adalah 16 bit, 32 bit, dan 64 bit.
3. Kata kunci (*key word*) disimbolkan dengan b dengan range 1, 2, 3, 4,..., 225.

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, [3]. Kunci internal kemudian digunakan dalam proses enkripsi dan dekripsi.

Proses enkripsi dibagi menjadi tiga, yaitu: penjumlahan integer, XOR dan rotasi. Untuk lebih jelasnya, adalah sebagai berikut:

1. Enkripsi

Diasumsikan terdapat dua buah blok input sebesar *w* bit, *A* dan *B*. Dan diasumsikan juga bahwa pembentukan kunci internal telah dilakukan, sehingga array *S*[0...*t*-1] telah dihitung. Sehingga *pseudecode* untuk proses enkripsi seperti di bawah:

```
A ← A + KI[0]
B ← B + KI[1]
for i ← 1 to r do
A←((A⊕B) <<< B) + KI[2i]
B←((B⊕A) <<< A) + KI[2i+1]
Endfor
```

2. 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 tersebut juga sama.

3. 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. Sebagai contoh:

```
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]
Endfor
```

Algoritma pembentukan kunci internal menggunakan konstanta P dan Q yang 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:

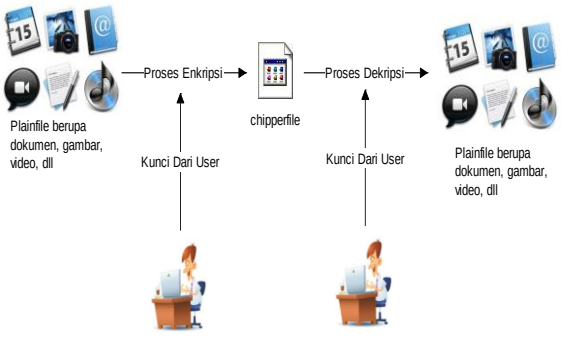
```
i ← 0
j ← 0
X ← 0
Y ← 0
n ← 3*max(r,c)
for k ← 1 to n do
KI[i]←(KI[i] + X + Y)<<< 3
X ← KI[i]
i ← (i + 1) mod t
L[j] ← (L[j] + X + Y)<<< 3
Y ← L[j]
j← (j + 1) mod c
endfor
```

Keterangan:

`max(r,c)` adalah fungsi menentukan bilangan terbesar antara r dan c. `c` adalah nilai maksimal dari panjang kunci `b` dibagi 4.

IMPLEMENTASI ALGORITMA TERHADAP FILE

Keamanan pada data yang terdapat dalam file diperlukan dalam proses penyimpanan maupun pengiriman data dalam yang bersifat rahasia. Hal tersebut disebabkan karena adanya kemungkinan pencurian data dalam file tersebut pada proses penyimpanan maupun pengiriman file tersebut. Maka file yang memiliki data yang akan diamankan harus dienkripsikan agar struktur dan data dalam file tersebut tidak dapat dibaca maupun dikenali tanpa proses dekripsi terebih dahulu. Sehingga proses pengamanan data pada file dapat diilustrasikan pada gambar 1.



Gambar 1. Proses Enkripsi / Dekripsi Pada File

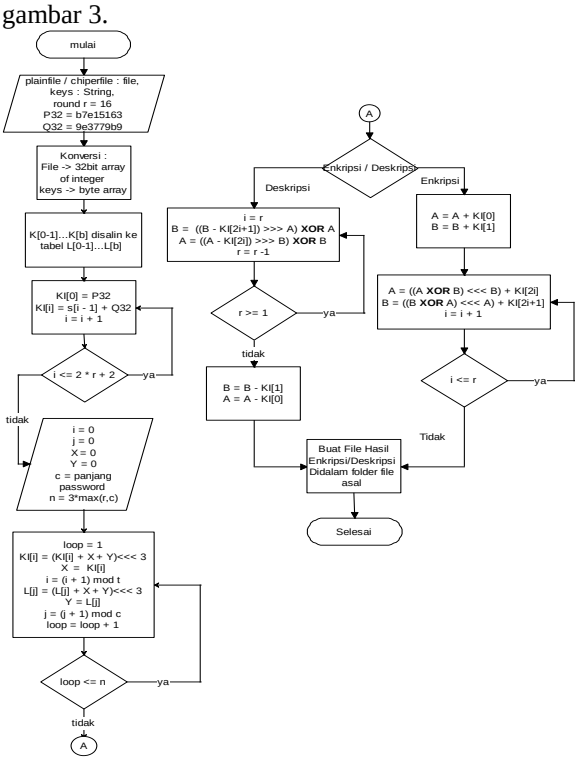
Pada dasarnya data berupa file terbentuk dari bit-bit data biner yang nantinya akan diproses oleh komputer sebagai perintah-perintah dan akan menghasilkan tampilan maupun data pada akhir proses tersebut. Bit data tersebut memiliki informasi berupa struktur file dan termasuk data yang terdapat dalam file itu sendiri. Jika bit pada data tersebut dikonversikan kedalam sebuah tabel yang berisi data berupa hexadecimal, maka hal tersebut dapat terlihat seperti pada gambar 2.

6D	FA	82	C4	B4	CD	BF	1F	65	27	46	DA	1A	71	5A	23	17	03	B6	CC
BB	55	CD	7E	2D	6F	CB	AF	AC	48	28	9C	12	C6	3B	A8	D9	16	12	BB
1F	C0	D1	4A	E3	A3	15	48	AF	71	C5	83	90	FF	C4	0A	F8	E7	D9	EC
80	15	2B	AE	BA	1F	B3	57	CD	B4	CD	02	ED	02	1F	8C	89	60	D2	8B
4F	84	CE	86	1D	F2	CA	73	A8	43	83	5D	DC	0F	4A	67	D4	0A	8A	3B
A1	8F	CF	6A	21	7A	09	29	51	9D	AC	A9	FA	15	2B	B4	DB	F3	0F	71
1C	0E	31	B4	B9	48	B8	35	90	26	5B	BF	CA	44	A7	7B	2C	05	07	F6
2A	D4	29	26	CF	AF	4E	39	56	F8	B4	53	1E	45	78	84	FB	9F	67	A3

Gambar 2. Potongan bit data pada file

Pada gambar 2, terdapat potongan bit data pada sebuah file dalam bentuk hexadecimal. Seluruh bagian dari bit file pada data dapat dirubah menjadi sebuah word dengan panjang 32 bit dengan melakukan konversi seluruh bit array pada stream data tersebut menjadi 32 bit Uinteger untuk dilakukan enkripsi atau dekripsi pada word tersebut.

Pada proses enkripsi file, diperlukan proses untuk membaca seluruh bit data pada file untuk diubah menjadi sebuah blok data yang diperlukan agar algoritma RC5 dapat diimplementasikan untuk melakukan enkripsi maupun dekripsi pada file tersebut. Proses yang dilakukan untuk melakukan enkripsi maupun dekripsi pada file dengan menggunakan algoritma RC5 dapat dilihat pada gambar 3.



Gambar 3. Flowchart Proses Enkripsi/Dekripsi File

Berdasarkan pada gambar 4.3, aplikasi pengamanan data akan dibagi menjadi dua bagian, yaitu enkripsi dan dekripsi. Dalam flowchart yang tertera pada gambar 4.3, aplikasi algoritma RC5 pada data berupa file terdapat pada bagian proses enkripsi dan proses dekripsi. Dalam perjalanan prosesnya, enkripsi dan dekripsi dengan algoritma RC5 selalu didahului dengan proses pembentukan kunci internal.

Proses enkripsi dan dekripsi pada algoritma RC5 tidak lepas dari pembentukan kunci internal *KI[]*. Kuci internal merupakan kunci berbentuk tabel yang digunakan untuk melakukan proses enkripsi dan dekripsi. Kunci ini dibentuk dari kunci yang diinputkan oleh *user* dan dirubah kedalam bentuk *array*. Dalam pembentukan kunci diperlulan konstanta P32 = b7e15163 dan Q32 = 9e3779b9 untuk panjang *word* sebesar 32 bit. Pembentukan tabel kunci diawali dari inisialisasi tabel *KI[]* dan diakhiri dengan pencampuran dua *array* kunci. Dalam pencampuran dua *array* kunci yang digambarkan pada gambar 4.3, akan dilakukan proses sebagai 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. `c` adalah nilai maksimal dari panjang kunci `b` dibagi 4.

Hasil akhir pada tabel *KI[]* merupakn kunci internal yang akan digunakan dalam proses enkripsi maupun dekripsi seperti yang ditunjukkan pada gambar 4.3.

Selanjutnya setelah kunci internal terbentuk, pada sistem terdapat dua pilihan fungsi yang dapat dipanggil, yaitu fungsi untuk melakukan proses enkripsi dan fungsi untuk melakukan proses dekripsi.

Pada gambar 4.3, proses enkripsi dan dekripsi dilakukan dengan menggunakan dua buah blok input *A* dan *B* sebagai *word* sebesar 32 bit yang telah dihasilkan dari proses konversi pada file input kedalam 32 bit *array of integer*. Selanjutnya, proses enkripsi akan dilakukan seperti pada algoritma berikut:

```
A ← A + KI[0]
B ← B + KI[1]
for i ← 1 to r do
   A←((A⊕B) <<< B) + KI[2i]
   B←((B⊕A) <<< A) + KI[2i+1]
Endfor
```

Sedangkan pada proses dekripsi akan dilakukan seperti pada algoritma berikut:

```
for i ← r downto 1 do
B←((B − KI[2i+1])>>>A) ⊕A
```

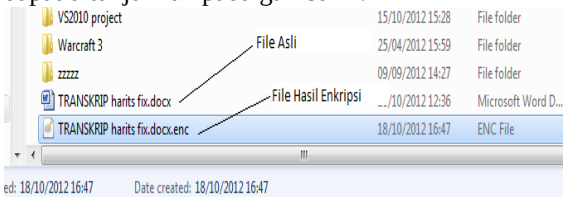
```
    A←((A - KI[2i])>>>B) ⊕B
endfor
B ← B - KI[1]
A ← A - KI[0]
```

Setelah dua buah blok input tersebut selesai di proses, maka sistem akan melakukan penggabungan kembali dan menuliskan hasil enkripsi atau dekripsi kedalam sebuah file baru.

PENGUJIAN SISTEM

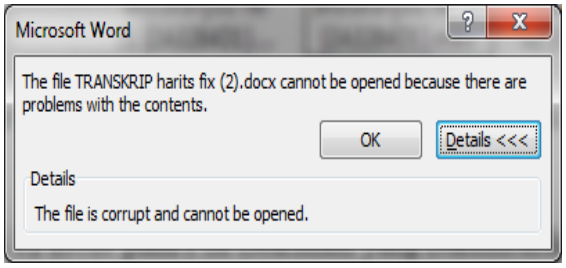
Pengujian pada aplikasi pengamanan data berupa file dengan metode RC5 difokuskan pada kemampuan aplikasi dalam melakukan pengamanan data. Hal ini difokuskan pada file yang telah dienkripsi dibandingkan dengan file original dan kemampuan aplikasi dalam mengembalikan file yang telah dienkripsi ke bentuk semula. Pengujian dilakukan pada dua jenis file, yaitu file dokumen Microsoft Word dan file berupa video.

File yang dienkripsikan akan mendapatkan ekstensi baru yaitu *.enc. File terenkripsi akan disimpan pada lokasi yang sama dengan file asli dan dinamai dengan nama yang sama dengan file asli ditambah dengan ekstensi file asli. Semisal file asli adalah "D:\ TRANSKRIP harits fix.docx" maka file terenkripsi akan disimpan sebagai "D:\ TRANSKRIP harits fix.docx.enc". Hal tersebut dapat ditunjukkan pada gambar 4.

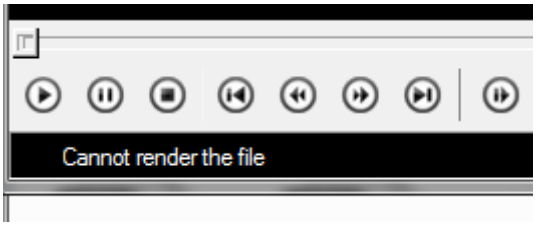


Gambar 4. Tampilan Perbandingan Antara File Asli dan Terenkripsi

Proses pengamanan data tidak hanya mengubah ekstensi file sehingga file tidak dapat dibuka, namun aplikasi pengaman data melakukan enkripsi pada bit data pada file tersebut sehingga struktur dari file akan berubah dan tidak dapat dibuka meskipun dilakukan pengembalian ekstensi secara manual. Berikut adalah contoh hasil File terenkripsi yang dikembalikan secara manual.

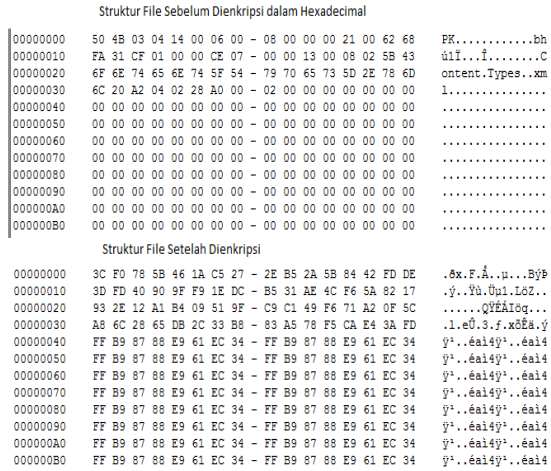


Gambar 5. *Error* pada File Dokumen yang Dikembalikan Secara Manual Saat dibuka



Gambar 6. *Error* pada File Dokumen yang Dikembalikan Secara Manual Saat dibuka

Pada gambar 5 dan gambar 6 dapat dilihat bahwa file yang ekstensinya dikembalikan ke bentuk asal dengan cara manual tidak dapat dibuka. Hal ini disebabkan oleh perbedaan struktur file pada data yang telah dienkripsi seperti yang ditunjukkan pada gambar 7.



Gambar 7. Perbandingan Struktur File Asli dan Terenkripsi

Untuk mengembalikan file kedalam bentuk semula, diperlukan proses dekripsi yang merupakan kebalikan dari proses enkripsi, dimana proses tersebut akan mengembalikan nilai dan struktur bit data pada file kedalam bentuk semula. Proses pengembalian data pada aplikasi keamanan data pada file memerlukan kunci yang tepat untuk dapat mengubah file menjadi ke bentuk semula. Kesalahan dalam memasukkan kunci pada proses enkripsi akan menyebabkan file menjadi rusak.

KESIMPULAN

Keamanan pada informasi yang terkandung dalam file dapat ditingkatkan dengan cara melakukan enkripsi guna mengacak bit data dalam file sehingga file tersebut tida dapat dibuka atau dibaca oleh aplikasi pada komputer. Algoritma RC5 dapat diimplementasikan dalam melakukan proses enkripsi bit data pada file dengan baik dikarenakan prosesnya yang ringan dan cepat sehingga akan lebih menghemat waktu dan *resource* dalam

melakukan enkripsi file yang memiliki ukuran data cukup besar.

DAFTAR PUSTAKA

[1] Rivest, R.L. 1997. *The RC5 Encryption Algorithm**. Cambridge : MIT Laboratory for Computer Science 545 Technology Square (http://people.csail.mit.edu/rivest/Rivest-rc5rev.pdf) diakses pada tanggal 1 Agustus 2012

[2] Hamdani, Suryawan, S.H., dan Septiarini, A. 2013. "Pengujian Algoritma Rivest Code 5 Untuk Enkripsi Struktur File Dokumen". *Prosiding STI 2013 Seminar Nasional Teknik Informatika, Prospek dan Tantangan Mobile Application*. Juni 2013,Universitas Ahmad Dahlan.

[3]Munir, R. 2004. Pengantar Kriptografi. Bandung : Institute Teknologi Bandung.

[4] Rehman, S.U. 2012. "Comparison Based Analysis of Different Cryptographic and Encryption Techniques Using Message Authentication Code (MAC) in Wireless Sensor Networks (WSN)". *International Journal of Computer Science Issues*. Vol 9. No. 2. Issue 1 January 2012. page 96 - 101.

[2] Kaliski, B.S Jr and Yin, Y.L. 1998. *On the Security of the RC5 Encryption Algorithm RSA Laboratories Technical Report TR-602*. RSA Laboratories, a division of RSA Data Security, Inc.