Perbandingan Teknik Kriptografi menggunakan Metode Sapphire II Dan Metode RC4

Susi Japit[1)]**, Rudy**[2)]
STMIK IBBI
Jl. Sei Deli No. 18 Medan, Telp. 061-4567111 Fax. 061-4527548
e-mail: Susijapit@gmail.com[1)], Rudy@hotmail.com[2)]

Abstrak

Kerahasiaan data adalah merupakan hal yang sangat penting. Hal ini bertujuan untuk mencegah agar data tidak dapat dibaca dan dipergunakan oleh orang yang tidak berhak dan berkepentingan, seperti pencuri data atau penerima data yang keliru karena kesalahan pengirim. Untuk mengatasi masalah maka dikembangkan teknik pengamanan yang dikenal dengan kriptografi. Alasan tersebut yang mendorong ketertarikan untuk melakukan penelitian ini. Terdapat banyak sekali jenis dari algoritma kriptografi dan secara umum dikatagorikan atas dua kelompok yaitu block cipher dan stream cipher. Pada penelitian ini akan dilakukan perbandingan dua algoritma kriptografi, yaitu Sapphire II dan RC4. Pada prinsipnya kedua algoritma tersebut berbeda tetapi masih termasuk dalam kelompok stream cipher. Untuk itu peneliti meneliti kinerja kedua metode ditinjau dari avalanche effects dan waktu komputasi yang diperlukan untuk mengenkripsi dan mendekripsi suatu file. Kunci yang digunakan oleh algoritma Sapphire II dan RC4 hanyalah sebesar 256 bit. Perangkat lunak yang dirancang adalah berbasis under windows dengan menggunakan bahasa pemrograman Visual Basic. Perangkat lunak yang dirancang untuk mengenkripsi file biner (*.*) karena modus pembacaan file dilakukan secara binary meliputi file teks, audio, citra, dan video. Perbandingan yang dilakukan adalah dalam hal kecepatan proses enkripsi/dekripsi, Avalanche Effects, dan distribusi karakter (distribusi byte).

Kata kunci : kriptografi, Sapphire II, RC4

Abstract

*Confidentiality of data is a very important thing . It aims to prevent the data can not be read and used by unauthorized people and people who are not interested , such as data theft or receiving erroneous data due to shipper error . To overcome the problem of the developed technique known as cryptographic security . The reasons that drive interest to conduct this research . There are many types of cryptographic algorithms and is generally categorized into two groups: block ciphers and stream ciphers . This research will be conducted comparing two cryptographic algorithms , namely Sapphire II and RC4 . In principle, the algorithms are different but still belong to the stream cipher . To the researchers examined the performance of both methods in terms of avalanche effects and the computational time required to encrypt and decrypt a file . The key used by the RC4 algorithm and Sapphire II is of 256 bits . The software is designed based under windows using Visual Basic programming language . The software is designed to encrypt binary files (* . *) Because the file is done reading mode binary files include text , audio , image , and video . Comparisons are made in terms of the speed of encryption / decryption process , Avalanche Effects , and character distribution (distribution of bytes) .*

1. Pendahuluan

Dengan semakin pesatnya perkembangan teknologi komputer, dimana pertukaran data dan informasi semakin mudah dilakukan. Terkadang kemudahan dalam hal pertukaran data menjadi bermasalah jika data yang dipertukarkan tidak dilindungi dengan suatu teknik pengamanan sehingga data dapat dicuri atau dimanipulasi oleh orang yang tidak berhak. Untuk mencegah terjadinya pencurian data dari orang yang tidak berhak maka dikembangkanlah berbagai teknik pengamanan data, salah satu teknik yang dapat digunakan untuk menjaga keamanan data adalah dengan menggunakan perangkat lunak kriptografi. Peneliti tertarik untuk meneliti dalam bidang kriptografi dikarenakan bidang tersebut termasuk sub bidang matematika yang perkembangannya sangat pesat saat ini karena banyaknya aplikasi dalam bidang kriptografi yang dapat diterapkan seperti dalam komunikasi data dan suara, lalu lintas data dalam internet, standar keamanan dalam dunia perbankan hingga digunakan dalam bidang militer.

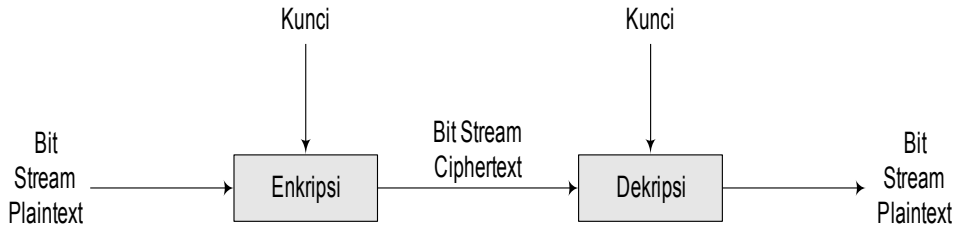
Terdapat banyak sekali metode kriptografi yang telah dikembangkan dimana untuk jenis *stream cipher* dua algoritma yang terkenal adalah Sapphire II dan RC4. Sapphire II diciptakan oleh Michael Paul Johnson pada tahun 1994 yang merupakan pengembangan dari *Sapphire Stream Cipher* original. Pada dasarnya metoda enkripsi ini mirip dengan RC4 tetapi menggunakan plainteks dan cipherteks *feedback*, sehingga lebih mudah digunakan secara aman. Perbedaan lain adalah operasinya yang lebih kompleks dan menggunakan indeks yang lebih banyak yaitu lima buah indeks sedangkan RC4 hanya dua buah indeks. Dengan kompleksnya operasi dan saling berkaitan operasi sebelum dan sesudahnya (sehingga tidak memungkinan dieksekusi secara paralel) maka kecepatan enkripsi/dekripsi dari Sapphire II tidak akan akan secepat RC4 (kurang lebih hanya sepertiganya). Kelebihan dari metoda ini adalah dapat digunakan sebagai generator bilangan *pseudorandom* (mestinya setiap metoda enkripsi dapat digunakan untuk hal ini), dapat juga digunakan sebagai *hash generation*. Panjang kunci seperti juga pada RC4 dapat mencapai 256 byte (2048 bit).

Dengan melihat sifat-sifat dari algoritma Sapphire II dan RC4 yang memiliki kesamaan maka peneliti berkeinginan untuk meneliti dan membandingkan sistem pengamanan data dengan menggunakan kedua metode tersebut yang diberi judul "Perbandingan Teknik Kriptografi Menggunakan Metode Sapphire II dan Metode RC4."

2. Metodologi Penelitian

Metodologi penelitian dilakukan dengan mengumpulkan data terlebih dahulu. Proses pengumpulan data dilakukan dengan studi kepustakaan. Peneliti mengambil bahan dan sumber-sumber yang berkaitan dengan topik yang dibahas dengan mencari di buku-buku, artikel, materi perkuliahan dan *website-website* yang ada di Internet. Adapun jenis data yang dikumpulkan terdiri atas: data primer dan data sekunder. Data primer yaitu data yang diperoleh berasal dari *website* yang membahas algoritma kompresi data khususnya kriptografi dan metode RC4 dan Sapphire II. Data sekunder yaitu data yang berasal dari buku yang membahas tentang kriptografi dan metode RC4 dan Sapphire II secara tidak langsung.

Dari skema umum proses enkripsi Sapphire II dan RC4 terlihat bahwa kedua algoritma *stream cipher* ini merupakan jenis kriptografi kunci privat dimana kunci yang sama harus digunakan kembali baik untuk proses enkripsi dan proses dekripsi. Secara umum kedua *stream cipher* merupakan jenis *stream cipher* yang dapat memproses *stream* dalam ukuran bit ataupun secara per byte.

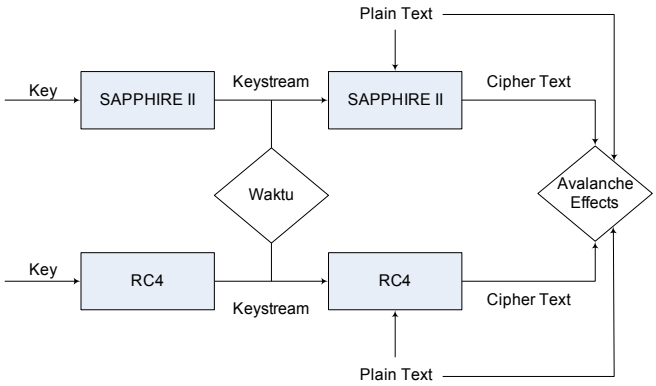


Gambar 1 Skema umum proses enkripsi dan dekripsi Sapphire II dan RC4

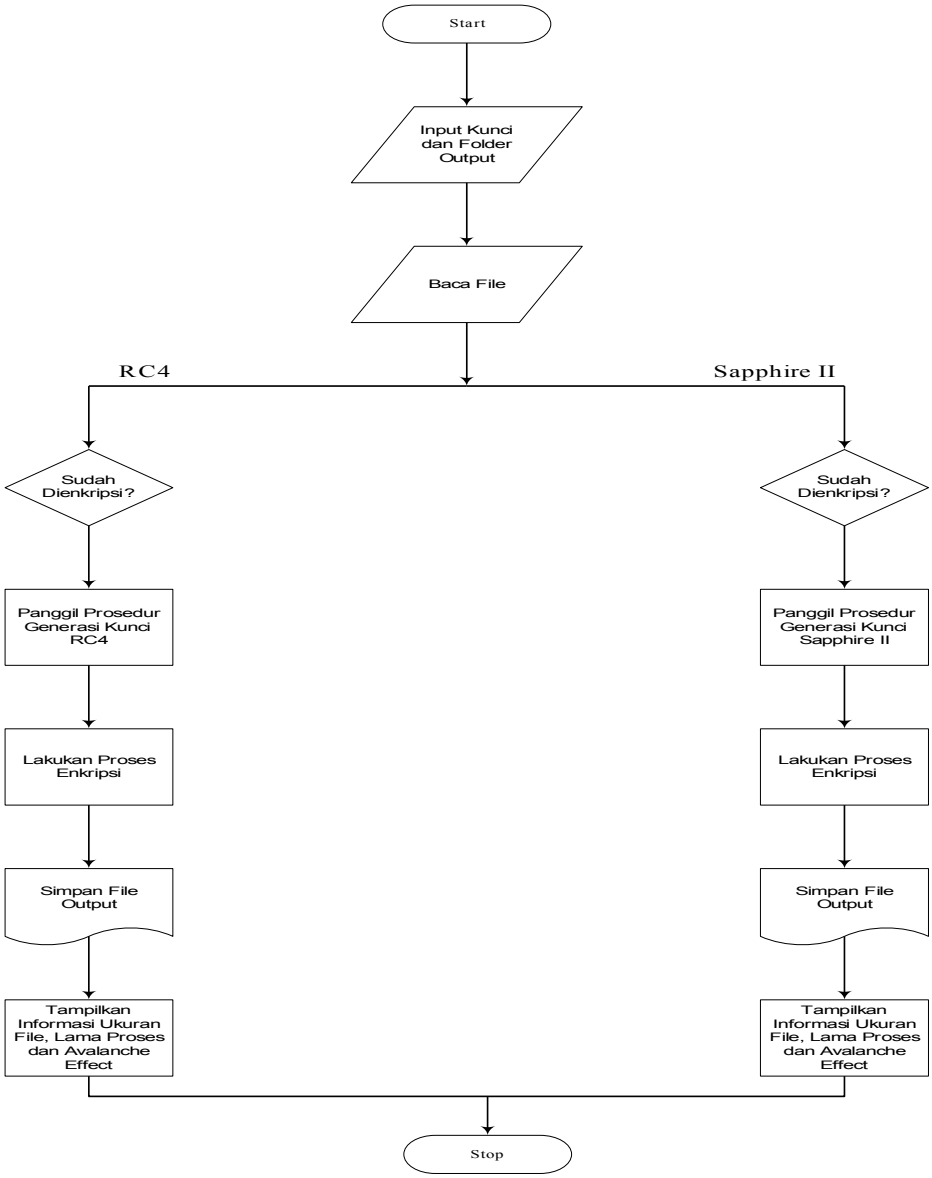
Secara khusus rangkaian proses algoritma Sapphire II dan RC4 sendiri terdiri atas dua tahapan yaitu tahap pertama adalah pembentukan *keystream* yang merupakan pembentukan sub kunci sedangkan tahap keduanya merupakan tahap enkripsi dan dekripsi. Pada algoritma Sapphire II *keystream* yang dihasilkan akan mengacak *array* CARDS sedangkan pada RC4 digunakan untuk mengacak nilai pada array Sbox. Seluruh rangkaian tahap tersebut adalah sama dimana pada tahap enkripsi dan dekripsi hanya *keystream* yang dihasilkan dilakukan proses *xor* dengan *plaintext* dan *ciphertext*.

Program yang dirancang nantinya dipergunakan untuk membandingkan dua buah algoritma *Stream Cipher* yaitu RC4 dan Sapphire II. Perbandingan yang dilakukan mencakup perbandingan waktu (kecepatan proses algoritma) serta perbandingan nilai *avalanche effects*. Khusus untuk perbandingan kecepatan algoritma akan dilakukan dengan membandingkan lama proses yang mencakup pembentukan kunci hingga proses enkripsi / dekripsi sedangkan perbandingan nilai *avalanche effects* dilakukan dengan cara membandingkan bit-bit *file* input dengan bit-bit *file output* dari tiap algoritma sesudah diproses.

Agar lebih jelasnya bagaimana proses perbandingan pada program ini maka dapat dilihat pada diagram perbandingan algoritma seperti terlihat pada gambar 2 dan Gambar 3 berikut ini.



Gambar 2 Diagram perbandingan kedua algoritma



Gambar 3 Diagram cara kerja program

3. Analisis dan Hasil

3.1 Analisis

analisis contoh perhitungan metode RC4 dan Sapphire II baik untuk proses enkripsi dan proses dekripsi.

Proses enkripsi dengan algoritma Sapphire II. *String* sampel dalam hal ini adalah Test.txt yang terdiri atas 7 *byte* dan merupakan *plain text* yang berisi string "NEMESIS". Untuk algoritma Sapphire II ini pertama sekali dibentuk 256 buah *array* yang bernama Cards dan masing-masing diisi dengan sebuah nilai yaitu 0 hingga 255. Proses ini pada algoritma Sapphire II disebut sebagai proses inisialisasi. Langkah berikut adalah mengacak CARDS berdasarkan *key* yang di-*input*. Dengan adanya 256 buah *Sbox* berarti Sapphire II dapat mendukung hingga 256 karakter untuk *key*-nya. Dalam pengujian ini digunakan *key* dengan panjang 3 karakter. Adapun kunci (*key*) yang dipakai adalah *string* "ABC".

key = "ABC"
key_length = 3
key[0] = 65; key[1] = 66; key[2] = 67 // dalam bentuk desimal

Langkah selanjutnya adalah melakukan *swapping* pada cards, yang hasil adalah *variable* Cards. Setelah itu proses enkripsi dilakukan pada tiap karakter hingga panjang pesan atau *string* terakhir. Panjang string yang akan dienkripsi: LEN("NEMESIS") = 7, Jika dinyatakan dalam bentuk desimal maka:

N = 78 // message[1] = 78
E = 69 // message[2] = 69
M = 77 // message[3] = 77
E = 69 // message[4] = 69
S = 83 // message[5] = 83
I = 73 // message[6] = 73
S = 83 // message[7] = 83

Hasil dari proses enkripsi dapat diperlihatkan pada Tabel 1 di bawah ini:

Tabel 1 Tabel Hasil Enkripsi String "NEMESIS" dengan Sapphire II

Plain(i)	Biner	Heksa desimal	Desimal	Karakter
1	11011110	DE	222	▌
2	00100001	21	33	!
3	10110000	B0	176	░
4	11110001	F1	241	±
5	00000000	00	0	NULL
6	00010010	12	18	↕
7	01100101	65	101	e

Perhitungan *Avalanche Effects*-nya adalah sebagai berikut:
Biner pada *string* "NEMESIS" adalah:
01001110 01000101 01001101 01000101 01010011 01001001 01010011
Biner pada string hasil enkripsi dengan Sapphire II:
11011110 00100001 10110000 11110001 00000000 00010010 01100101

Dari hasil di atas terlihat bahwa perbedaan bit pada posisi yang sama sebanyak 29 bit jadi nilai *avalance effects*-nya adalah sebagai berikut:

$$\frac{Jumlah\ bit\ beda}{Jumlah\ bit\ yang\ dibandingkan} x\ 100\% = \frac{29}{56} x\ 100\% = 51{,}78\%$$

Untuk proses dekripsi pada Sapphire II urutan pengacakan nilai pada CARDS sama seperti dengan proses enkripsi termasuk juga algoritma dekripsinya mirip dengan algoritma enkripsinya.

Proses dekripsi dilakukan pada tiap karakter hingga panjang pesan atau *string* terakhir.
Panjang string yang akan didekripsi: LEN(“ ▌!▒±NULL↕e”) = 7
Jika dinyatakan dalam bentuk desimal maka:

```
▌       = 222   //  cipher[1] = 222
!       = 33    //  cipher[2] = 33
▒       = 176   //  cipher[3] = 176
±       = 241   //  cipher[4] = 241
NULL    = 0     //  cipher[5] = 0
↕       = 18    //  cipher[6] = 18
e       = 101   //  cipher[7] = 101
```

Hasil dari proses dekripsi dapat diperlihatkan pada Tabel 2 di bawah ini:

Tabel 2 Tabel Hasil Dekripsi String “ ▌!▒±NULL↕e” dengan Sapphire II

Cipher(i)	Biner	Heksa desimal	Desimal	Karakter
1	01001110	4E	78	N
2	01000101	45	69	E
3	01001101	4D	77	M
4	01000101	45	69	E
5	01010011	53	83	S
6	01001001	49	73	I
7	01010011	53	83	S

Proses enkripsi dengan algoritma RC4. *String* sampel dalam hal ini adalah Test.txt yang terdiri atas 7 *byte* dan merupakan *plain text* yang berisi string "NEMESIS". Untuk algoritma RC4 ini pertama sekali dibentuk 256 buah SBox yang akan dipakai sebagai kunci dalam RC4. Hasil dari proses enkripsi dapat diperlihatkan pada Tabel 3 di bawah ini:

Tabel 3 Tabel Hasil Enkripsi String "NEMESIS" dengan RC4

Cipher(i)	Biner	Heksa desimal	Desimal	Karakter
1	10000000	80	128	Ç
2	01100001	61	97	A
3	00010011	13	19	!!
4	11010110	D6	214	╥
5	00011100	1C	28	(cursor right)
6	11100111	E7	231	τ
7	00010101	15	21	§

Perhitungan *Avalanche Effects*-nya adalah sebagai berikut:
Biner pada *string* "NEMESIS" adalah:
01001110 01000101 01001101 01000101 01010011 01001001 01010011
Biner pada string hasil enkripsi dengan RC4:
10000000 01100001 00010011 11010110 00011100 11100111 00010101

Dari hasil di atas terlihat bahwa perbedaan bit pada posisi yang sama sebanyak 29 bit jadi nilai *avalance effects*-nya adalah sebagai berikut:

$$Avalanche\ Effects = \frac{Jumlah\ bit\ beda}{Jumlah\ bit\ yang\ dibandingkan} x\ 100\% = \frac{29}{56} x\ 100\% = 51{,}78\%$$

Setelah itu proses dekripsi dilakukan pada *string* tersebut dengan terlebih dahulu sama seperti halnya dengan proses enkripsi dibentuk 256 buah SBox melalui proses *swapping* dengan key "ABC" (key length = 3).

Kemudian barulah proses dekripsi dilakukan pada tiap karakter hingga panjang *cipher* terakhir. Hasil dari proses dekripsi dapat diperlihatkan pada Tabel 4 di bawah ini:

Tabel 4 Tabel Hasil Dekripsi String "NEMESIS" dengan RC4

Message(i)	Biner	Heksa desimal	Desimal	Karakter
1	01001110	4E	78	N
2	01000101	45	69	E
3	01001101	4D	77	M
4	01000101	45	69	E
5	01010011	53	83	S
6	01001001	49	73	I
7	01010011	53	83	S

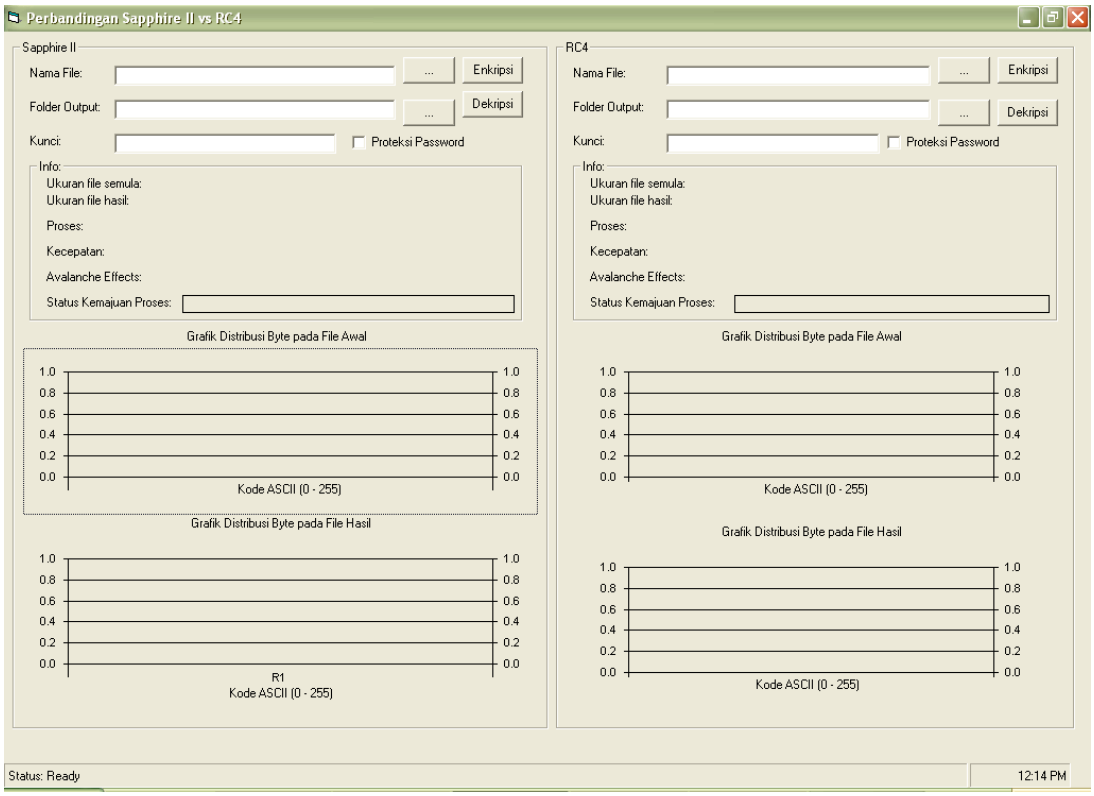
3.2 Hasil Implementasi

Secara umum program dapat melakukan dua tipe operasi yaitu yang pertama melakukan proses enkripsi dan dekripsi langsung pada *file* dan bagian kedua menampilkan hasil distribusi *byte* dalam bentuk grafik. Khusus untuk proses pada *file* tidak akan ditunjukkan langkah enkripsi atau dekripsi.Pada operasi *file* dalam satu proses hanya dapat memproses satu *file* saja dan operasi ini akan bersifat menimpa *file* yang sama jika nama *folder* yang diberikan terdapat nama *file* yang sama.

Setelah itu jika program dijalankan maka akan tampak tampilan seperti gambar berikut ini.

Gambar 4 Tampilan *Splash Screen*

Tampilan Gambar 4 merupakan bentuk tampilan *splash screen* dari program ini. Untuk menghilangkan tampilkan di atas maka dapat ditekan tombol OK. Berikutnya akan ditampilkan tampilan utama dari program perbandingan *stream cipher* ini seperti terlihat pada gambar 5.



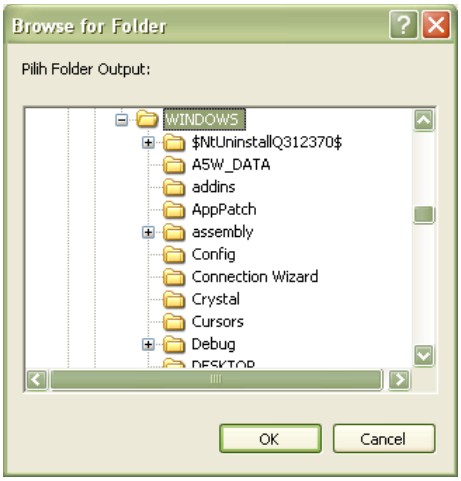
Gambar 5 Tampilan Program

Seperti terlihat pada gambar 5 tampilan program ini terdiri atas dua bagian yaitu yaitu bagian proses enkripsi dan dekripsi untuk algoritma Sapphire II dan bagian sisi lainnya untuk algoritma RC4.

Terdapat empat buah tampilan grafik dimana grafik yang pertama dari masing-masing algoritma untuk menampilkan grafik distrubis *byte* pada *file* yang di-*load*. Sedangkan grafik di bawahnya untuk menampilkan grafik distribusi *file* setelah diproses.

Agar informasi perbandingan dapat diketahui seperti kecepatan proses dan persen *Avalanche Effects* maka bagian ini akan ditampilkan pada bagian info pada kedua algoritma.

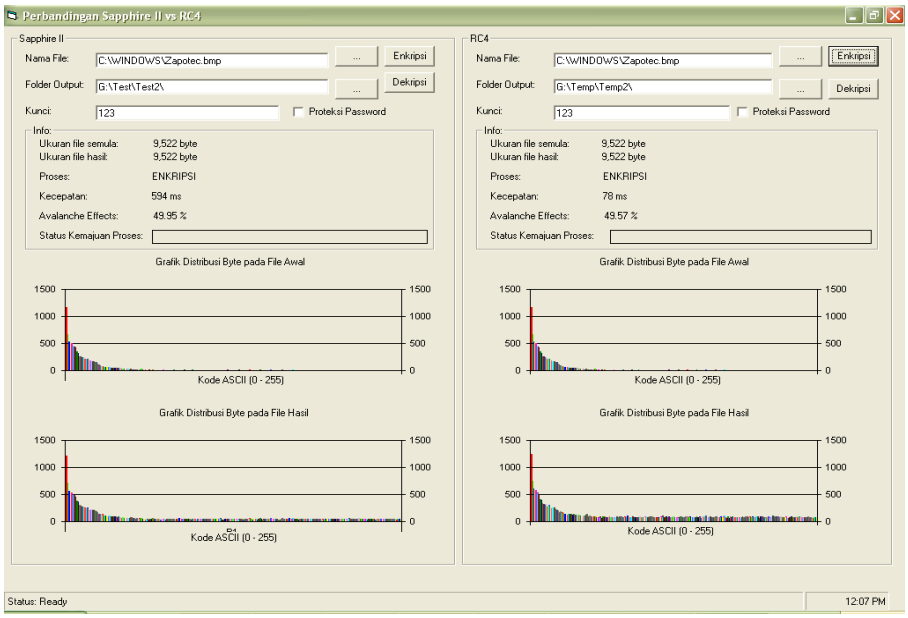
Untuk menjalankan program ini maka tentukan terlebih dahulu *file* input yang akan diproses dengan mengklik pada tombol bertanda … pada bagian kanan. Selanjutnya adalah memilih *folder output* juga dengan cara mengklik pada tombol bertanda … pada bagian kanan. Setelah itu program akan menampilkan sebuah kotak dialog untuk menentukan *folder output* seperti terlihat pada gambar 6.



Gambar 6 Tampilan *Folder Output*

Setelah selesai maka masukkan *password* atau kunci dengan panjang maksimum sebanyak 256 karakter (256 *byte*). Agar *password* yang dimasukkan tidak kelihatan maka dapat ditandai bagian "Proteksi *Password*". Setelah selesai maka klik pada tombol "Enkripsi" untuk menjalankan proses enkripsi dan tombol "Dekripsi" untuk menjalankan proses dekripsi.

Lakukan hal yang sama seperti di atas pada bagian lain *frame* RC4 ataupun pada *frame* Sapphire II. Setelah selesai maka akan ditampilkan hasilnya seperti tampilan Gambar 7 berikut ini.



Gambar 7 Tampilan Program Saat Enkripsi

3.3 Pengujian

Untuk mengetahui hasil pengujian program ini dengan perbandingan algoritma Sapphire II dan RC4 yang telah diimplementasikan maka dilakukan pengujian pada beberapa jenis *file* seperti *file* teks, *audio*, *graphic*, dan *file word processsor* dengan menggunakan kunci (*password*) "**123**". Tabel 5 merupakan tabel hasil pengujian yang telah dilakukan pada algoritma Sapphire II dan RC4.

Tabel 5 Tabel Hasil Pengujian Proses Enkripsi

Jenis File	Ukuran File (*byte*)	Ukuran File Output (*byte*)	Sapphire II			RC4		
			Lama Proses (ms)	Kecepatan Proses (*byte*/ms)	Avalanche Effects	Lama Proses (ms)	Kecepatan Proses (*byte*/ms)	Avalanche Effects
Gambar	16.730	16.730	140	119,50	50,20%	47	355,96	49,89%
Gambar	107.260	107.260	250	429,04	49,97%	232	462,33	49,78%
Text	42.503	42.503	156	272,45	50,00%	78	544,91	49,92%
Text	297	297	20	14,85	50,72%	18	16,5	47,47%
Audio	55.776	55.776	172	324,28	49,94%	78	715,07	49,95%
Audio	24.253	24.253	108	224,56	49,88%	31	782,35	49,88%
EXE	2.288.953	2.288.953	4.250	538,57	49,98%	2.235	1.024,14	50,01%
EXE	1.000.960	1.000.960	1.891	529,32	49,96%	1.000	1.000,96	50,00%

Dari hasil pengujian proses enkripsi dengan algoritma Sapphire II ini didapat kecepatan rata-ratanya sebesar 318,03125 *byte*/ms sedangkan untuk nilai *avalanche effects*-nya didapat rata-rata 49,9775 %.

Dari hasil pengujian proses enkripsi dengan algoritma RC4 ini didapat kecepatan rata-ratanya sebesar 649,8325 *byte*/ms sedangkan untuk nilai *avalanche effects*-nya didapat rata-rata 49,875 %.

Dari hasil yang didapat (Tabel 6) ini terlihat bahwa kecepatan proses enkripsi dari algoritma RC4 dua kali lipat dibandingkan dengan algoritma Sapphire II. Sedangkan untuk nilai *avalanche effects*, algoritma Sapphire II hanya beberapa *point* di atas algoritma RC4.

Tabel 6 Tabel Hasil Pengujian Proses Dekripsi

Jenis File	Ukuran File (*byte*)	Ukuran File Output (*byte*)	SAPPHIRE II			RC4		
			Lama Proses (ms)	Kecepatan Proses (*byte*/ms)	Avalanche Effects	Lama Proses (ms)	Kecepatan Proses (*byte*/ms)	Avalanche Effects

Gambar	16.730	16.730	94	177,97	50,20%	31	539,67	49,89%
Gambar	107.260	107.260	265	404,75	49,97%	260	412,53	49,78%
Text	42.503	42.503	156	272,45	50,00%	78	544,91	49,92%
Text	297	297	24	12,375	50,72%	20	14,85	47,47%
Audio	55.776	55.776	157	355,26	49,94%	62	899,61	49,95%
Audio	24.253	24.253	94	258,01	49,88%	31	782,35	49,88%
EXE	2.288.953	2.288.953	4.281	534,67	49,98%	2.266	1.010,13	50,01%
EXE	1.000.960	1.000.960	1.906	525,16	49,96%	1.000	1.000,96	50,00%

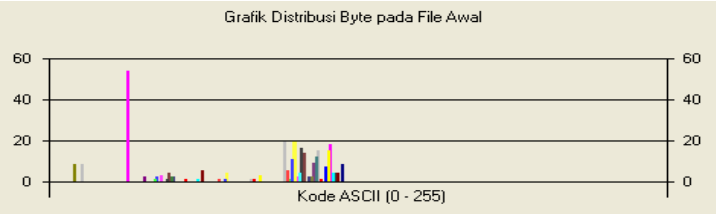
Dari hasil pengujian proses dekripsi dengan algoritma Sapphire II ini didapat kecepatan rata-ratanya sebesar 329,35 *byte*/ms sedangkan untuk nilai *avalanche effects*-nya didapat rata-rata 49,9775 %.

Dari hasil pengujian proses dekripsi dengan algoritma RC4 ini didapat kecepatan rata-ratanya sebesar 689,1475 *byte*/ms sedangkan untuk nilai *avalanche effects*-nya didapat rata-rata 49,875 %.

Dari hasil yang didapat ini terlihat bahwa kecepatan proses dekripsi dari algoritma RC4 dua kali lipat dibandingkan dengan algoritma Sapphire II.

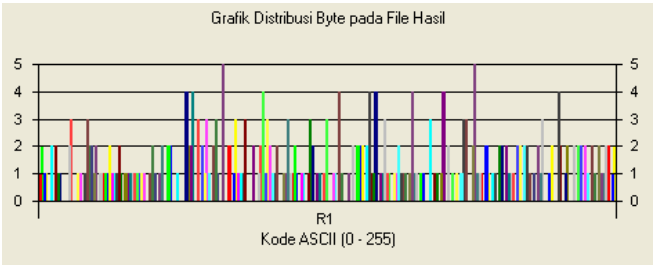
Untuk perbandingan proses enkripsi dan dekripsi Sapphire II maka proses dekripsinya lebih cepat dibandingkan dengan proses enkripsinya. Hal ini juga berlaku untuk algoritma RC4 dimana proses dekripsinya juga lebih cepat dibandingkan dengan proses enkripsinya.

Pada Gambar 8 hingga Gambar 13 merupakan tampilan dari grafik distribusi pada *file bitmap* Shed.bmp dan tampilan distrubsi *byte* pada proses enkripsi menggunakan algoritma Sapphire II dan RC4.



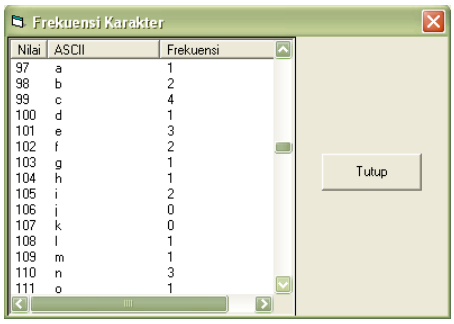
Gambar 8 Grafik Distribusi *Byte* pada *File Input*

Gambar 9 Tampilan Distribusi ASCII *File Input*

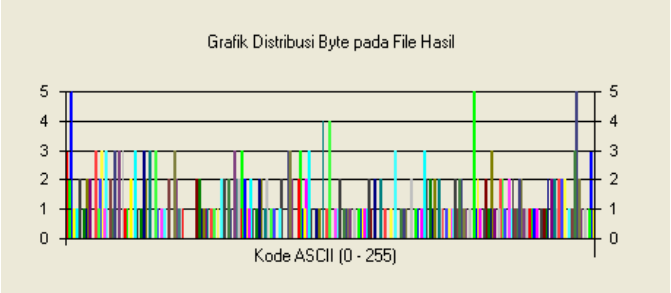


Gambar 10 Grafik Distribusi *Byte* Hasil Algoritma Sapphire II

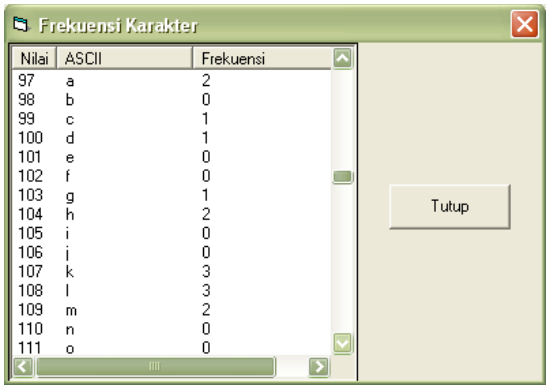
Pola distribusi *byte* ini dihasilkan dengan cara mencacah jumlah setiap karakter ASCII yang mempunyai indeks 0 hingga 255 untuk dicari berapa frekuensi kemunculan dari masing-masing indeks dan hasilnya kemudian ditampilkan dalam bentuk grafik batang seperti pada gambar di atas.



Gambar 11 Tampilan Distribusi ASCII Hasil Algoritma Sapphire II



Gambar 12 Grafik Distribusi *Byte* Hasil Algoritma RC4



Gambar 13 Tampilan Distribusi ASCII Hasil Algoritma RC4

4. Kesimpulan dan Saran

Hasil uji coba secara umum didapat metode RC4 mempunyai waktu komputasi yang cepat. Sedangkan Sapphire II secara umum mempunyai *avalanche effects* sedikit di atas RC4 tetapi mempunyai waktu komputasi yang lebih lama dibandingkan RC4. Dengan demikian untuk pemilihan algoritma *stream cipher* yang lebih aman dengan nilai *avalanche effects* yang baik maka metode Sapphire II lebih tepat digunakan dan bila faktor kecepatan yang menjadi pertimbangan maka metode RC4 dapat dipakai. *Avalanche Effects* algoritma Sapphire II secara umum unggul 0,1025% di atas dari algoritma RC4.Hasil uji coba secara umum didapat metode RC4 mempunyai waktu komputasi yang cepat yaitu 331,80125 ms yang disebabkan oleh algoritma RC4 lebih sederhana dibandingkan dengan Sapphire II.

Untuk pengembangan lebih lanjut program enkripsi dan dekripsi pada *file biner* ini, maka dapat diberikan beberapa saran sebagai berikut: program dapat ditambah dengan beberapa metode algoritma kriptografi yang lain agar perbandingan yang dilakukan tidak hanya terbatas dua metode saja. Perbandingan dilakukan dengan menggunakan format *file* yang lain dengan ukuran yang lebih bervariasi sehingga hasil pengujian yang didapat lebih akurat.

Daftar Pustaka

[1] Rivest R. The MD5 Message Digest Algorithm, Request for Comments: 132, Network Working Group, MIT Laboratory for Computer Science and RSA Data Security, Inc, April 1992,

[2] Ariyus, Dony. Kriptografi: Keamanan Data dan Komunikasi. Yogyakarta : Penerbit Graha Ilmu, 2006.
[3] Ariyus, Dony. Pengantar Ilmu Kriptografi: Teori, Analisis dan Implementasi. Yogyakarta: Penerbit Andi. 2008.
[4] Kurniawan Yusuf. Kriptografi Keamanan Internet dan Jaringan Komunikasi. Bandung: PT. informatika. 2004.
[5] Raharjo, Agus. CyberCrime, Bandung: Citra Aditya Bakti. 2002.
[6] Schneier, Bruce. Applied Cryptography, Edisi 2, New Jersey: John Wiley & Sons, Inc. 1996.
[7] Schneier, Bruce and Whiting, Doug. Fast Software Encryption: Designing Encryption
[8] Algorithms for Optimal Sofware Speed on the Intel Pentium Processor. 1997.
[9] Schneier, Bruce. Security Pitfalls in Cryptography. Counterpane Systems. 1998.
[10] Wei Dai, Crypto ++. C++ Class Library of Cryptographic Primitives Version 2.1, 1996.