BAB VI

ANALISIS KINERJA PERANGKAT LUNAK

6.1 Penanganan Kesalahan

Dalam tahap ini akan dijelaskan mengenai pengujian program aplikasi yang digunakan pada enkripsi dan dekripsi *file* maupun *text*. Dengan pengujian ini diharapkan tingkat kesalahan baik dalam pengenkripsian *file* maupun sistem itu sendiri menjadi sangat minim bahkan tidak ada.

Pengujian kinerja pada enkripsi/dekripsi *file* ini dilakukan untuk mengetahui kesalahan tersebut. Penanganan kesalahan pada enkripsi/dekripsi *file* ini dilakukan dengan memberikan dalam bentuk pesan peringatan yang berisikan informasi tentang keharusan untuk mengisikan data tertentu.

Dalam hal ini pengujian kinerja untuk mendeteksi kesalahan pada enkripsi/dekripsi *file* terdiri dari pengujian normal dan pengujian tidak normal.

6.2 Pengujian Normal

Pengujian normal dilakukan dengan memberikan masukan sesuai dengan data yang dibutuhkan, dimana data yang dimasukan harus benar.

6.2.1 Pengujian Normal *Form Encrypt File*

Pada *form encrypt file* masukkan data berupa *key* proses enkripsi, nama *file* yang akan dienkripsi, *file* untuk menyimpan hasil dari proses enkripsi, nama *file* yang akan didekripsi dan *file* untuk menyimpan hasil dari proses dekripsi.

Hasil outputnya dapat dilihat di informasi *file* berupa nama *file* yang akan dienkripsi/dekripsi, ukuran *file* dan waktu proses enkripsi/dekripsi *file*. Misal data yang akan dimasukkan untuk proses enkripsi *file* adalah:

Key Proses Enkripsi : 1234

File yang akan dienkripsi : E:\Data Test\Nama PID.doc

Save Enkripsi *File* : E:\Data Test\Nama PID.doc.cry

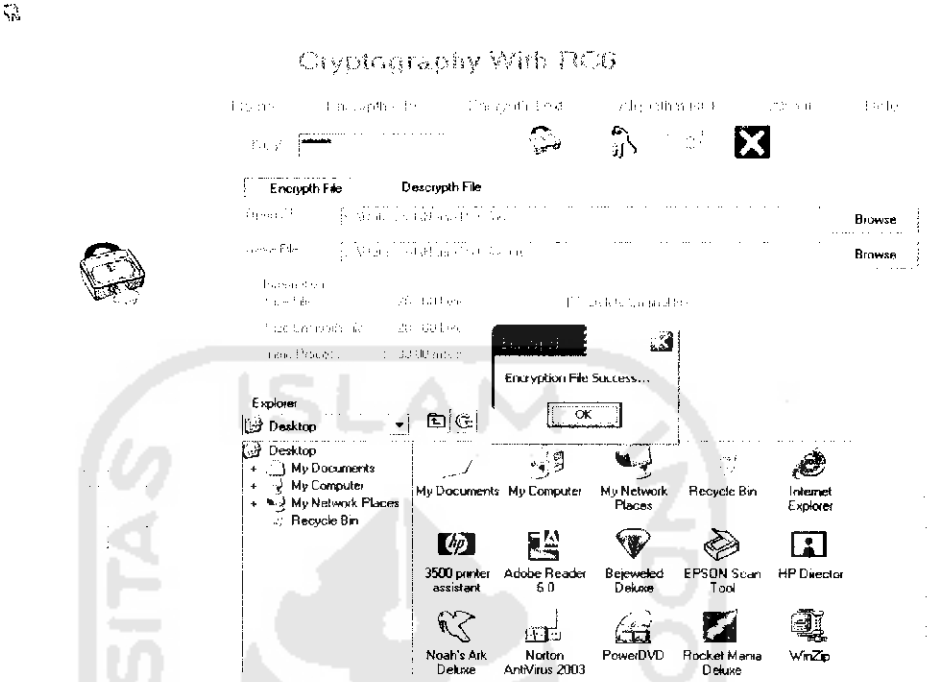
Untuk hasil dari proses enkripsi dapat dilihat pada informasi *file* yaitu:

Ukuran *File* Original : 28,160 *bytes*

Ukuran *File* Enkripsi : 28,160 *bytes*

Waktu Proses Enkripsi : 20.00 mscd

Dari hasil proses enkripsi dapat diketahui bahwa untuk *file* Nama PID.doc dengan ukuran *file* sebesar 28,160 *bytes* waktu prosesnya yaitu 20 mscd. Hasil proses enkripsi ini disimpan dalam *file* Nama PID.doc.cry. Hasil dari proses enkripsi dengan data diatas dapat dilihat pada gambar 6.1, sedang untuk *file* yang telah terenkripsi dapat dilihat pada gambar 6.2



Gambar 6.1 Hasil proses enkripsi *file*

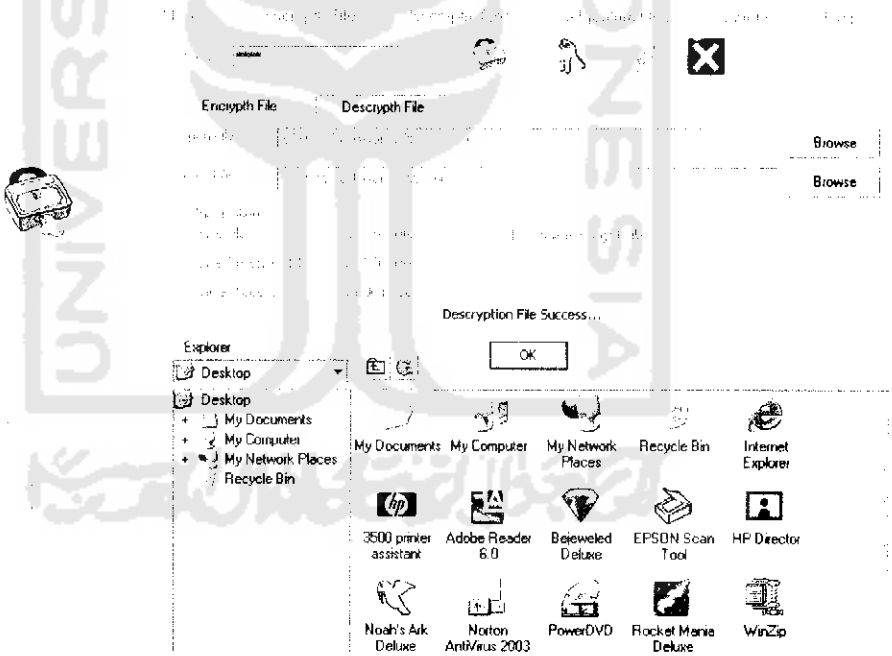
Gambar 6.2 *File* hasil proses enkripsi

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:\Data Test\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



Gambar 6.3 Hasil proses dekripsi *file*

6.2.2 Pengujian Normal *Form Encrypt Text*

Pada *form encrypt text* masukkan datanya adalah *key* proses cnkripsi dan *plaintext* yang akan dienkripsi. Informasi *file* berupa nama *file* yang akan dienkripsi , ukuran *file*, panjang karakter dan waktu proses. Misal data yang akan dimasukkan pada proses cnkripsi *text* adalah :

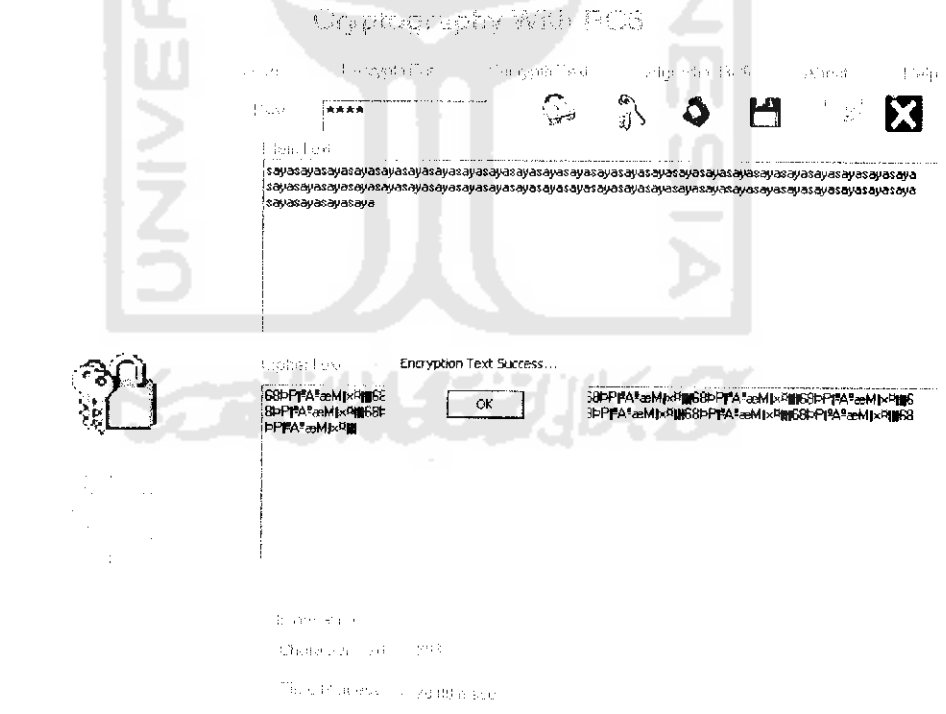
Key Proses Enkripsi : 1234

Nama *File* Enkripsi : E:\Data Test\test3.txt

Untuk informasi *file* yang diperoleh adalah :

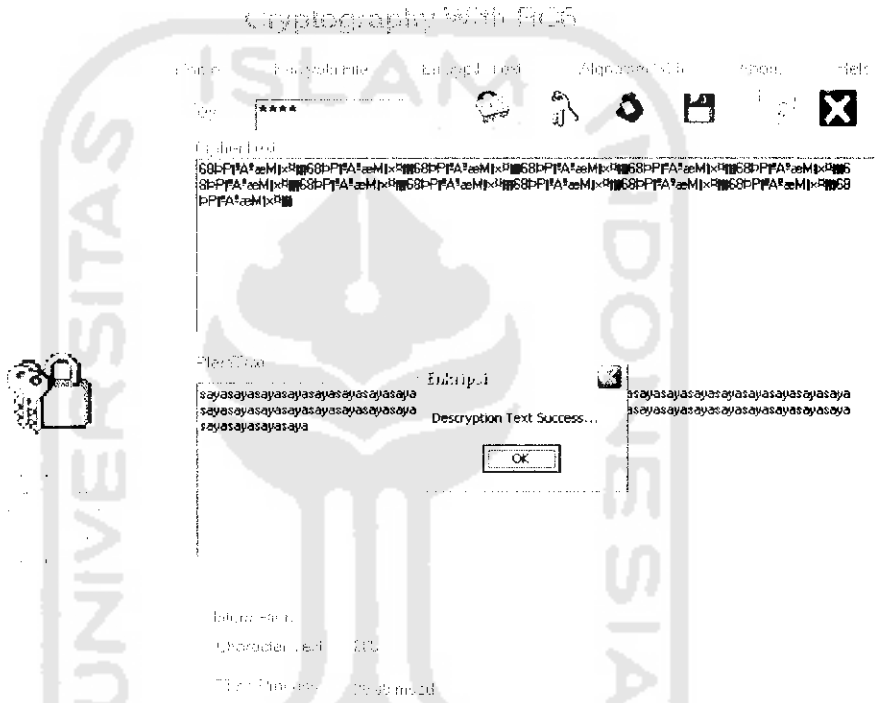
Panjang Karakter *Encryptext* : 208 karakter

Waktu Proses Enkripsi : 20 mscd



Gambar 6.4 Hasil proses enkripsi *text*

Hasil dari proses dekripsi *text* ditampilkan pada memo *Plaintext* dengan karakter seperti semula (*plaintext*). Untuk informasi *file* yang diperoleh yaitu panjang karakter yang sama pada saat enkripsi serta waktu proses untuk dekripsi *text* yaitu 20 msed, hal ini dikarenakan *file* yang dienkripsi berukuran kecil.



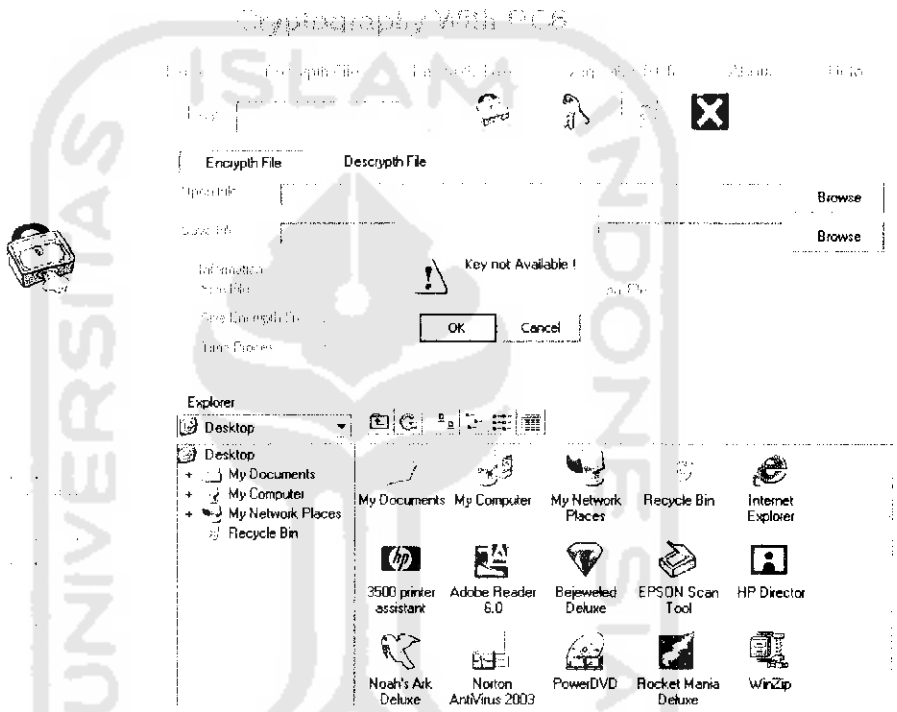
Gambar 6.5 Hasil proses dekripsi *text*

6.3 Pengujian Tidak Normal

Pengujian tidak normal (*robust testing*) ini dilakukan untuk penanganan kesalahan input data dengan memberikan pesan peringatan kepada *user*.

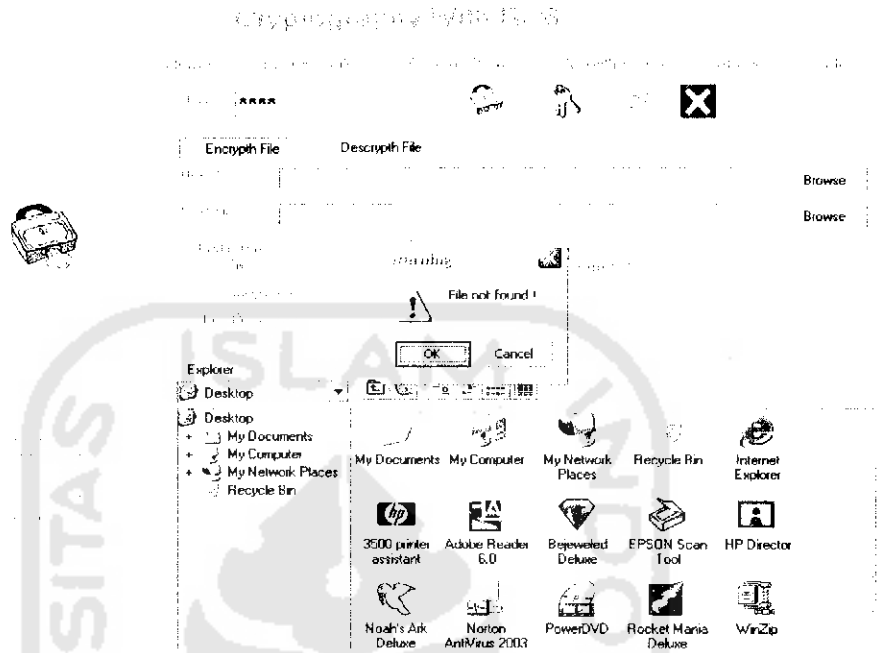
6.3.1 Pengujian Tidak Normal *Form Encrypt File*

Pada *form encrypt file* terdapat edit untuk memasukkan *key* proses enkripsi, jika edit ini tidak diisi maka akan menampilkan pesan peringatan untuk memasukkan *key* enkripsi. Untuk lebih jelasnya lihat gambar 6.6

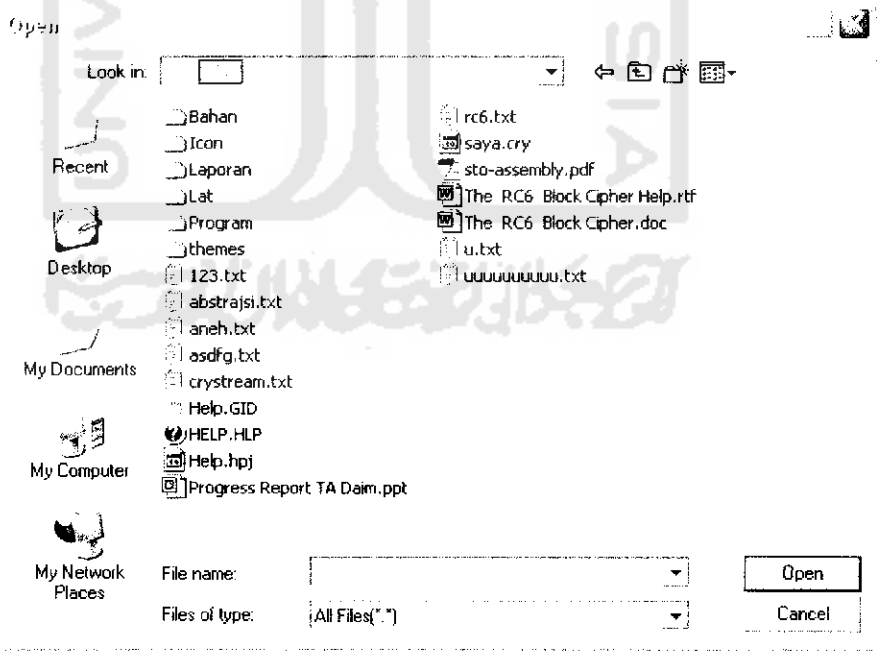


Gambar 6.6 Pesan peringatan *key* enkripsi belum terisi

Selain *key* enkripsi yang harus diisi, nama *file* yang akan dienkripsi juga harus diisi hal ini untuk memproses *file* mana yang akan dienkripsi. Setelah nama *file* yang akan dienkripsi dipilih kemudian memilih *file* untuk menyimpan hasil dari proses enkripsi. Pesan peringatan jika nama *file* yang akan dienkripsi tidak diisi dapat dilihat pada gambar 6.7



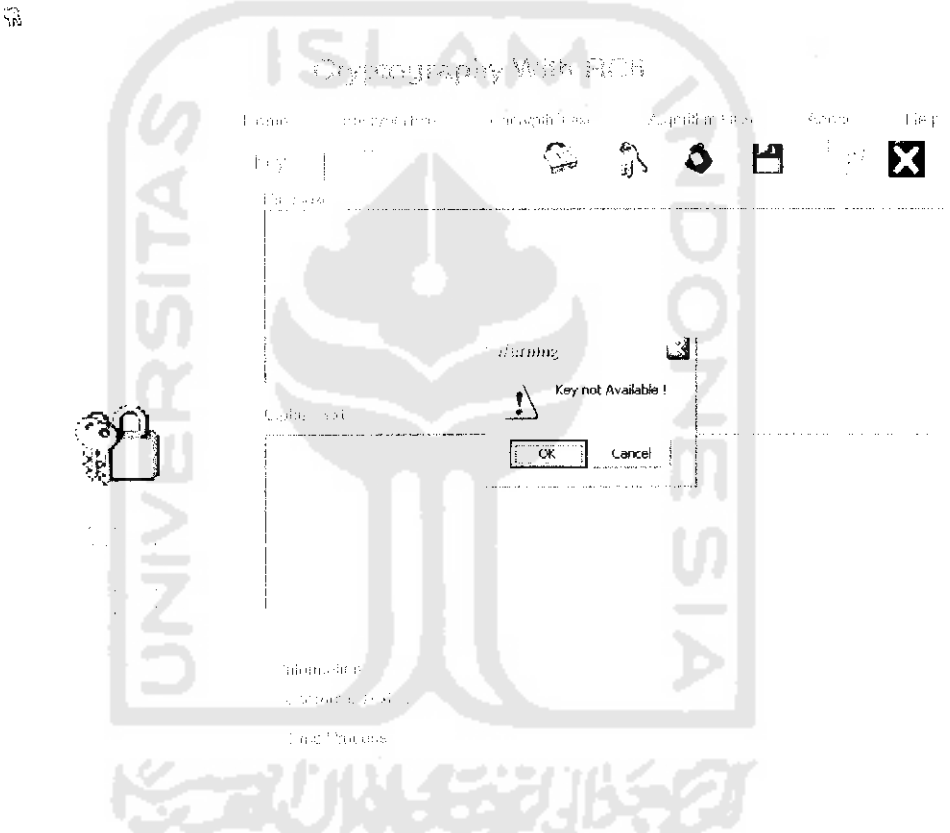
Gambar 6.7. Pesan peringatan *file* untuk enkripsi belum terisi



Gambar 6.8. Pilihan *file* yang akan dienkripsi

6.3.2 Pengujian Tidak Normal *Form Encrypt Text*

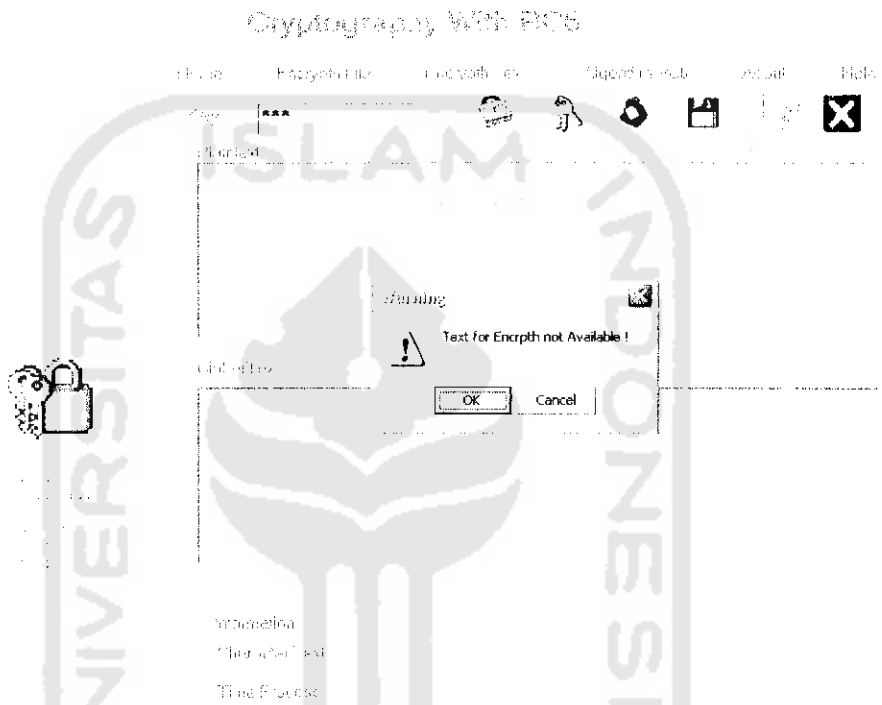
Pada *form encrypt text*, *key* enkripsi harus diisi untuk proses enkripsi, kemudian memo *plaintext* juga harus diisi sebagai pesan yang akan dienkripsi. Untuk *key* dan memo *plaintext* yang tidak diisi maka akan menampilkan pesan peringatan. Lihat gambar 6.9 dan gambar 6.10



Gambar 6.9 Pesan peringatan *key* enkripsi *text* belum terisi

Pada pengujian dan analisi perangkat lunak, pada pengujian enkripsi *text* terdapat beberapa kasus yang unik. Terdapat beberapa model *text* yang tidak dapat dilakukan proses enkripsi. Seperti kata "SAYA', harus dienkripsi dengan kunci minimal 5 karakter. Proses enkripsi dapat berjalan sukses, akan tetapi proses

dekripsi tidak menghasilkan karakter semula. Tetapi kata "saya', dengan sembarang jumlah karakter kunci proses enkripsi dan dekripsi sukses. Hal ini mungkin disebabkan adanya kesalahan *logical* pada program.



Gambar 6.10 Pesan peringatan *plaintext* belum terisi

6.4 Analisis Perangkat Lunak

Analisis perbandingan digunakan untuk melihat kinerja dari algoritma RC6. Perbandingan disini berupa analisis waktu proses enkripsi dan dekripsi terhadap ukuran *file* kemudian perbandingan ukuran *file* antara *file* yang belum dienkripsi dengan setelah dienkripsi, analisis panjang kunci terhadap waktu

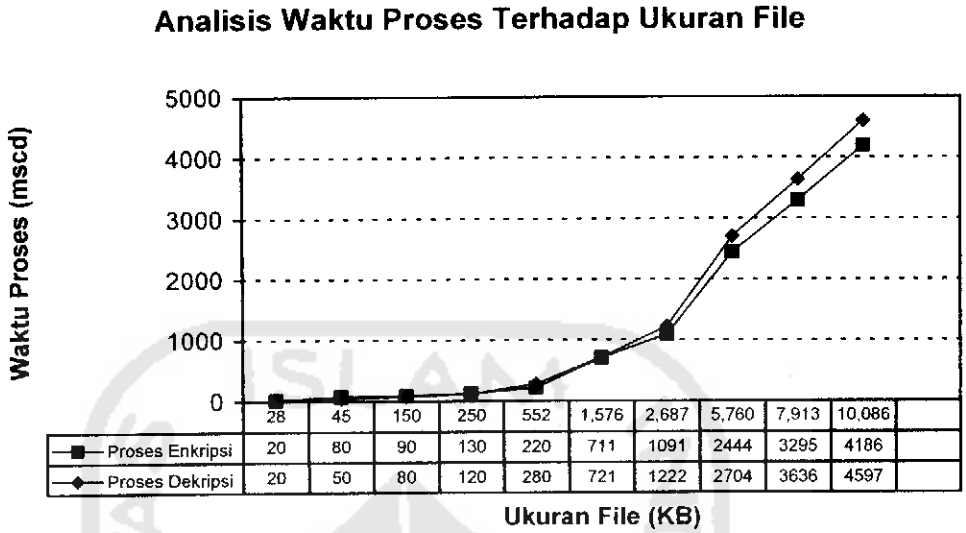
proses. Serta analisis algoritma RC6 terhadap algoritma *Blowfish* dan algoritma RC4.

6.4.1 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 6.1.

Tabel 6.1 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



Gambar 6.11 Grafik waktu proses terhadap ukuran *file*

6.4.2 Analisis Ukuran *File* Terhadap Proses Enkripsi

Analisis dengan membandingkan ukuran *file* sebelum dienkripsi dan setelah *file* dienkripsi. Dapat dilihat pada tabel 6.2 bahwa ukuran *file* logo.doc dengan ukuran *file* logo.doc.cry adalah sama. Untuk lebih jelasnya lihat tabel 6.2

Tabel 6.2 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*.

6.4.3 Analisis Proses Enkripsi Terhadap Panjang kunci

Pada tabel 6.3 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 6.3 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 diatas, 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. Akan tetapi apabila waktu proses diukur dalam hitungan detik, perbedaan waktu proses ini tidak akan nampak. Untuk lebih jelasnya, akan dijelaskan lebih lanjut.

6.4.4 Analisis Waktu Proses Terhadap Pengaruh Aplikasi Yang Berjalan

Pada tabel 6.4 akan digambarkan lebih lanjut pengaruh adanya aplikasi yang berjalan pada waktu yang dibutuhkan pada saat proses enkripsi.

Tabel 6.4 Analisis waktu proses terhadap pengaruh aplikasi yang berjalan

Nama *File*	Ukuran *File* (KB)	Panjang Kunci	Aplikasi Yang Berjalan			Waktu Proses (msed)
			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 diatas, 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.

6.4.5 Analisis Waktu Proses Terhadap Spesifikasi Komputer

Pada tabel 6.5 berikut akan digambarkan pengaruh spesifikasi komputer terhadap kecepatan waktu proses enkripsi.

Tabel 6.5 Analisis waktu proses terhadap spesifikasi komputer

Nama *File*	Ukuran *File*	Panjang Kunci	Waktu Proses Pada Tiap PC (msed)			
			A	B	C	D
10.ppt	10,086KB	1234	4637	4621	4574	4156
Al-Hadid.mp3	5,897	1234	1822	1798	1732	1684
Hall.jpg	2,687	1234	1282	1253	1207	1172

Keterangan :

A : PC dengan spesifikasi ; AMD Duron 750 MHz, 256 SDRAM

B : PC dengan spesifikasi ; P3 933 MHz, 256 SDRAM

C: PC dengan spesifikasi ; P4 1,6 GHz, 256 DDRAM pc 2700

D : PC dengan spesifikasi ; P4 2,8 GHz, 256 DDRAM pc 3200

Perbedaan spesifikasi PC dengan kecepatan processor dan kapasitas memori yang berbeda, merupakan salah satu penyebab perbedaan kecepatan waktu proses enkripsi. Hasil yang didapat pada tabel 6.5 diatas bukan merupakan nilai pasti. Nilai tersebut dapat berubah unutk setiap kali pengujian

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

Penelitian yang dilakukan oleh saudara Anton Nugroho yaitu aplikasi enkripsi *file* dengan menggunakan algoritma *Blowfish* yang berbentuk *block cipher*, sedang penelitian yang lakukan saudara Dian Wahyudi adalah aplikasi enkripsi *file* dengan menggunakan algoritma RC4 yang berbentuk *stream cipher* dimana dalam proses enkripsi dan dekripsi berbeda.

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 6.6 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
Hall.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*.

6.5 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.