PENGAMANAN PESAN DALAM EDITOR TEKS MENGGUNAKAN *HYBRIDCRYPTOSYSTEM*

Yuli Fauziah[1)]
[1)]Jurusan Teknik Informatika UPN "Veteren" Yogyakarta
Jl. Babarsari No. 2 Tambakbayan Yogyakarta
e-mail : yuli_fa@yahoo.com

Abstrak

Editor teks merupakan tempat asal dari suatu pesan dibuat. Fasilitas keamanan data khususnya data tak terkoding (*plaintext*) menjadi suatu kebutuhan. *Plaintext* akan dikirimkan dalam bentuk *ciphertext*, sehingga diharapkan pemakai yang tidak berhak untuk membaca pesan yang terkirim tidak dapat dibaca oleh sembarang orang. Salah satu cara untuk mengamankan pesan tadi adalah dengan menggunakan teknik kriptografi, yang mana teknik ini ttelah mengalami perkembangan yang sangat pesat, yang dimylai dari sisi algoritmanya sampai dengan ukuran kunci yang digunakannya. Algoritma kriptografi yang telah ada kiranya dimungkinkan untuk digabungkan yang dimungkinkan akan menjadi suatu pengamanan berlapis. Hipotesis yang diajukan adalah penerapan proses tambahan pada editor teks ini tidak akan mengganggu kinerja *user*.

Penggunaan *hybrid cryptosystem* dalam penelitian ini merupakan gabungan kriptosistem RC4, RSA dan fungsi SHA-1 yang digunakan dalam tiga tahapan pengamanan data pada editor teks. Adapun tahapan pengamanan data meliputi : proses *setup key*, proses enkripsi data, dan proses dekripsi data.

Hasil penelitian ini menunjukkan bahwa pemakaian *secure* editor teks ini dapat mengamankan data. Pada penerapannya walaupun menambah sedikit waktu yang beragam dalam setiap proses, namun kunci yang dipakai tidak akan sama untuk setiap proses karena dilakukan secara acak.
Keyword: *plaintext, ciphertext, hybridcryptosystem, setup key, secure.*

1. PENDAHULUAN

Suatu editor teks adalah tempat kita melakukan pengolahan kata, mulai dari membuka file, memodifikasinya sampai dengan menyimpan kembali file teks. Untuk membaca file teks kita dapat menggunakan sembarang editor teks. Pada editor teks kiranya perlu menambah fasilitas pengamanan data sehingga dapat dikatakan suatu *secure* editor teks.

Kemudahan pengaksesan informasi, baik itu langsung atau secara terpisah (*remote access*), tentunya berdampak pada munculnya resiko dan ancaman keamanan dan integritas data. Ancaman yang diperkirakan akan terjadi adalah akses yang tidak berhak terhadap informasi atau sumber informasi, misalnya penduplikasian atau bahkan perusakan informasi itu sendiri, sehingga membawa kerugian. Untuk itu diperlukan suatu manajemen keamanan (*security management*) yang dapat melindungi atau paling tidak menahan suatu akses yang tidak berhak untuk durasi waktu tertentu.

Pengamanan file dengan menggunakan teknik kriptografi telah banyak dilakukan dalam berbagai penelitian dengan algoritma. Implementasi dengan menggunakan salah satu algoritma kriptografi saja kiranya sudah mulai ditinggalkan dan beralih pada penggunaan gabungan antara asimetrik kriptosistem dan simetrik kriptosistem. Gabungan antara asimetrik kriptosistem dan simetrik kriptosistem disebut sebagai *hybrid cyptosystem*.

Pembuatan suatu dokumen kadang diharapkan oleh pembuatnya untuk tidak dapat terbaca oleh orang lain. Namun, apabila komputer yang dipakai adalah multi *user*, maka perlu adanya suatu tindakan yang menuju ke pengamanan dokumen. (Ahyar, 2001)

2. TINJAUAN PUSTAKA

Keamanan data (*data security*) sering dibahas bersamaan dengan integritas data (*data integrity*). Keamanan lebih condong memiliki pengertian untuk keamanan data terhadap *unauthorized disclosure* (pengubahan data atau penghapusan data), sedangkan integritas lebih condong pada ketepatan atau validitas data. Secara sederhana, penjelasan kedua konsep itu adalah keamanan berarti mengamankan data terhadap user yang tidak berwenang dan integritas berarti mengamankan data terhadap user yang berwenang. (Dian, 2001)

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar aman. Kriptografi berasal dari bahasa Yunani yaitu "c*rypto*" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan). (Avon, 2004)

Data yang dapat dibaca dan dimengerti tanpa suatu cara khusus disebut dengan *plaintext* atau *cleartext*. Metode penyamaran *plaintext* sebagai suatu cara untuk menyembunyikan pesan disebut enkripsi (*encryption*). Penyamaran *plaintext* menghasilkan pesan yang tidak terbaca disebut dengan *ciphertext*. Kita dapat

menggunakan enkripsi untuk menjamin informasi disembunyikan dari mereka yang tidak berhak. Proses yang mengembalikan *ciphertext* ke dalam bentuk *plaintext* disebut juga dengan dekripsi (*decryption*). (Saskias, 1999)

Dalam penelitian ini menggunakan fungsi hash satu arah SHA-1 (*Secure Hash Algorithm - 1*). Fungsi hash SHA-1 ini digunakan pada proses inisialisasi kunci, yaitu berupa pengacakan terhadap *password* yang dimasukkan pada saat awal proses *setup key* RSA untuk menghitung *private key* dan pada pembuatan tanda (*signature*) yang terletak pada *header file*, yaitu melakukan pengacakan terhadap hasil enkripsi kunci RC4 oleh *cryptosystem* RSA .

Fungsi hash SHA-1 dikatakan *secure* sebab komputasinya tidak mungkin dikerjakan dengan mudah untuk dapat menemukan atau menghasilkan kembali sebuah pesan yang cocok dengan *message digest*-nya, atau untuk menemukan dua perbedaan pesan yang menghasilkan *message digest* yang sama. (Eastlake, 2001)

Fungsi hash SHA-1 mempunyai data masukan sepanjang lebih kecil dari 2^{64} bit, dan menghasilkan sebuah keluaran sepanjang 160 bit. Fungsi hash SHA-1 memiliki *target round* sebanyak 80 tahapan, fungsi logik f(0), f(1),...,f(79) yang masing-masing fungsi f(t) dimana 0 <= t <= 79 dan memiliki *constant word* K(0),K(1),...,K(79) yang merupakan bilangan *hexadecimal*, dan adanya W(t) yang dapat digunakan sebagai identitas hasil. (Eastlake, 2001)

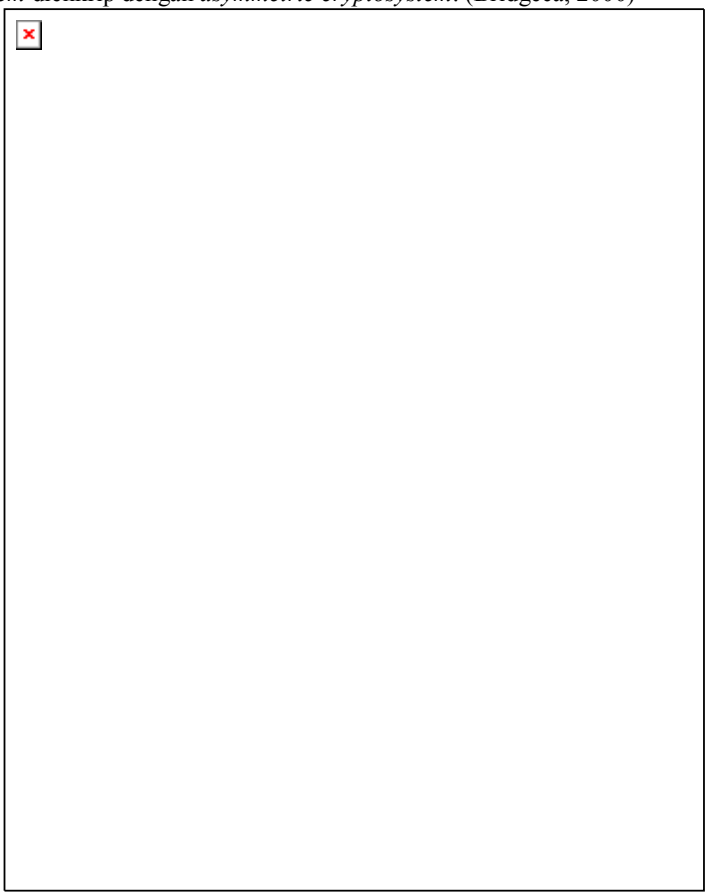
Cryptosystem adalah suatu fasilitas untuk mengkonversikan *plaintext* ke *ciphertext* dan sebaliknya. *cryptosystem* terdiri dari suatu algoritma seluruh kemungkinan *plaintext*, *ciphertext*, dan kunci. (Saskias, 1999)

Protokol kriptografi modern pada saat ini banyak menggunakan *hybrid cryptosystem* yaitu menggabungkan *asymmetric cryptosystem* dengan *symmetric cryptosystem* untuk memperoleh keunggulan-keunggulan pada masing-masing *cryptosystem*. (Bridgeca, 1999)

Hybrid cryptosystem merupakan gabungan antara *Cryptosystem* yang memakai *asymmetric Cryptosystem* dan *Cryptosystem* yang memakai *symmetric Cryptosystem* (Schneier, 1996)

Protokol kriptografi modern pada saat ini banyak yang menggabungkan *asymmetric cryptosystem* dengan *symmetric cryptosystem* untuk memperoleh keunggulan-keunggulan pada masing-masing *cryptosystem*. (Schneier, 1996)

Teknik enkripsi menggunakan *asymmetric cryptosystem* lebih lambat dibandingkan enkripsi menggunakan *symmetric cryptosystem*, karena pada *asymmetric cryptosystem* menggunakan teknik matematika yang cukup rumit, langkahnya *plaintext* dienkripsi dengan *symmetric cryptosystem* kemudian kunci privat *symmetric cryptosystem* dienkrip dengan *asymmetric cryptosystem*. (Bridgeca, 2000)



Gambar 1. Proses enkripsi dan proses dekripsi dengan *Hybrid Cryptosystem* (Bridgeca, 2000)

Cryptosystem Rivest Code 4 (RC4) merupakan model dan metode enkripsi *secret key*. *Cryptosystem* RC4 menggunakan panjang kunci dari 1 sampai 256 bit yang digunakan untuk menginisialisasikan tabel sepanjang 256 bit dan kunci RC4. RC4 merupakan salah satu jenis *stream cipher*, yaitu memproses unit atau input data pada satu saat. Dengan cara ini enkripsi dan dekripsi dapat dilaksanakan pada panjang yang bervariasi. RC4 tidak perlu *byte* tambahan untuk mengenkrip. (Slamet, 2003)

Sama halnya dengan *asymmetric cryptosystem* lainnya, RSA juga menggunakan dua buah kunci, yaitu kunci publik yang dapat dipublikasikan dan kunci privat yang harus dirahasiakan. Untuk menghasilkan kunci publik dan kunci pribadi dobuat algoritma yang berfungsi untuk membangkitkan kunci. Kemudian kunci publik digunakan untuk proses enkripsi, dan kunci pribadi digunakan untuk proses dekripsi.. Proses pemilihan kunci ini dikatakan sebagai kelemahan dari RSA karena membutuhkan waktu yang tidak sedikit, namun sekaligus merupakan suatu kekuatan dari RSA yaitu menambah segi keamanan hasil enkripsinya. (Schneier, 1996)

3. METODE PENELITIAN

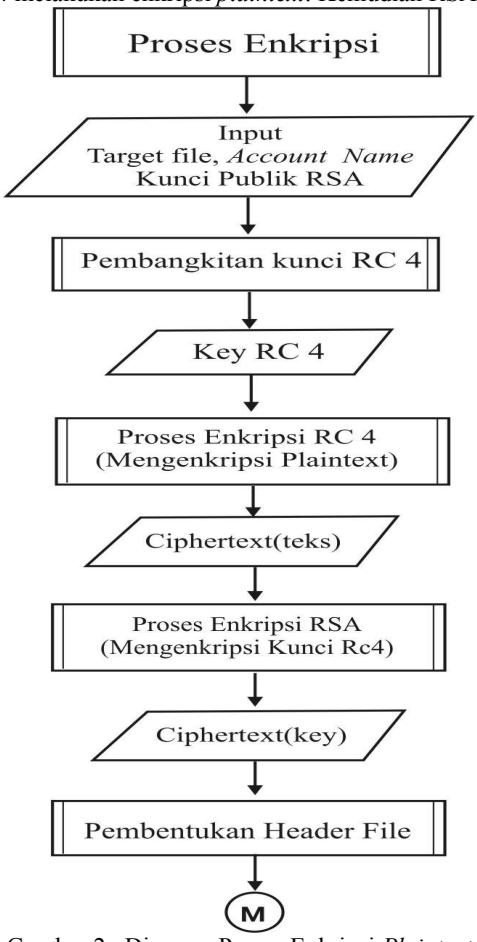
3.1. Analisis Sistem

1. Editor yang dimaksud di sini adalah berupa suatu *window* yang dipakai untuk menuliskan naskah/teks yang pendek dimana dilengkapi dengan suatu fasilitas penyandian data bila diperlukan.
2. File tersimpan menggunakan editor di atas, memakai *extensi file* .RFT, yaitu merupakan suatu metode enkoding dengan format teks dan struktur dokumen menggunakan sekumpulan karakter ASCII. Penggunaan tipe file ini karena dokumen diharapkan dapat dibaca oleh hampir semua *word processor* oleh *user* lain (sesuai otorisasinya) yang berbeda *platform* komputer (*Linux*, *Macintosh*) dan program. Target hasil enkripsi akan mengubah *extensi file* menjadi .Hbr, dan kembali dalam bentuk .RTF bila dilakukan proses dekripsi, dimana file hasil dekripsi ini akan terhapus bila kita keluar dari program editor teks.

3.2. Perancangan Sistem

Perancangan sistem terdiri dari perancangan prosedur operasi enkripsi, proses dekripsi dan *header file*.

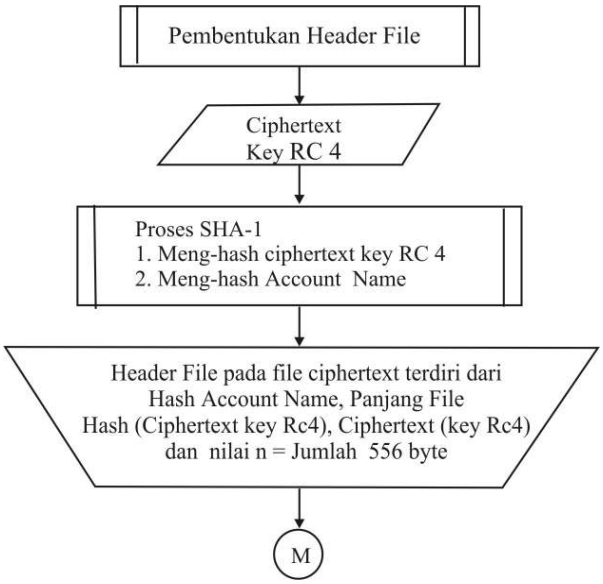
1. Perancangan prosedur operasi enkripsi dan deskripsi
   a. Proses *setup key* menggunakan fungsi hash SHA1 untuk menghasilkan kunci publik RSA, proses enkripsi diawali dengan RC4 melakukan enkripsi *plaintext*. Kemudian RSA mengenkripsi kunci RC4.



Gambar 2. Diagram Proses Enkripsi *Plaintext*

b. Apabila kita ingin melihat kembali *plaintext* yang terenkripsi, maka kita melakukan proses dekripsi. Proses dekripsi diawali dengan memasukkan *password* dan *account name* sebagai pembangkit kunci *private* RSA yang akan menghasilkan kunci/*secret key* RC4. Selanjutnya kunci RC4 akan mendekripsi *ciphertext* untuk menghasilkan *plaintext* yang kita maksud.

2. Perancangan *Header file*
   a. *Header file* atau *meta data* merupakan suatu keterangan file yang menampung informasi yang dibutuhkan oleh file yang bersangkutan.
   b. *Header file* memiliki ukuran maksimal 556 byte sebagai informasi yang terdiri dari 20 byte untuk menyimpan *account name* penerima, 4 byte keterangan panjang file terenkripsi, 20 byte kemudian menyimpan enkripsi kunci RC4 yang telah di-SHA-1 (*hash*) dan ditambah 256 byte menyimpan enkripsi kunci RC4 dan nilai *n* sebesar 256 byte.



Gambar 3. Diagram alir Prosedur Pembentukan *Header file*

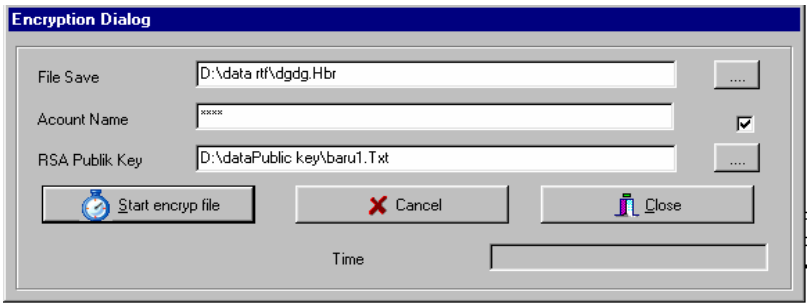
4. HASIL DAN PEMBAHASAN

Menjalankan editor teks secara umum adalah relatif seragam dalam pemakaiannya dan dapat membaca dokumen/filenya dengan menggunakan sembarang *word processor*. Setiap editor teks juga dilengkapi dengan fasilitas *secure*. Mengingat fasilitas ini sangat penting, maka konfigurasi standar dari editor pun diperlukan.

Untuk menentukan kunci atau proses *setup key* terletak pada menu *tool*. Proses ini dapat dilakukan sebelum atau sesudah menulis suatu dokumen/pesan, dengan terlebih dahulu kita memasukkan sebuah *password*, seperti gambar berikut :

Kunci publik yang dihasilkan disimpan sebagai kunci enkripsi dan *password* perlu kita ingat untuk dipakai saat kita mengdekripsi filenya. Bersamaan dengan proses *setup key* sebenarnya kita telah dapatkan kunci privat, namun setelah proses ini kunci tersebut tidak disimpan.

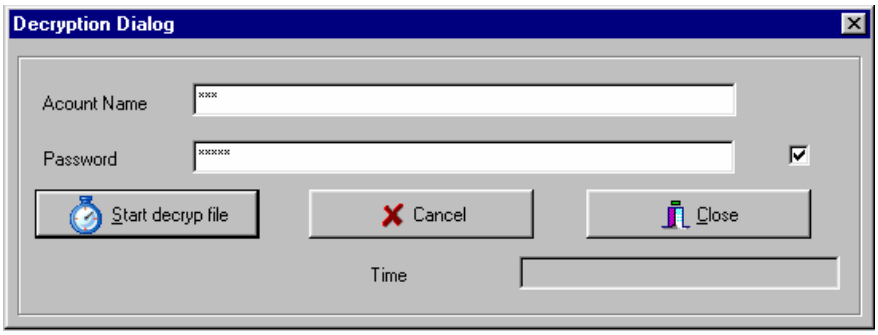
Pada saat kita akan menyimpan file maka kita dapat melakukan proses enkripsi. Proses ini diawali dengan menentukan target file enkripsi dengan memasukkan file tujuan yang berektensi .hbr, *accound name* si penerima, dan kunci publik.



Gambar 4. Dialog Enkripsi

Dokumen/file yang terenkripsi dilengkapi dengan adanya *header file* yang menampung informasi mengenai *accound name* penerima yang berhak, panjang dokumen/file, hash dari enkripsi kunci RC4, dan enkripsi kunci RC4. Jika data pada *header* ini terhapus maka file terenkripsi tersebut tidak dapat dibuka. Hal ini mempunyai pengaruh besar terhadap keamanan data kita, bahwa data kita tidak dapat sebatas *password* saja yang melindungi tapi juga tergantung *accound name* penerima.

Penerima hasil enkripsi atau orang yang mempunyai hak untuk dapat membuka file terenkripsi telah memiliki *password* yang dipakai membuka file yang bersangkutan atau melakukan proses dekripsi. Proses ini berada pada menu *open file*. Untuk membuka file yang terenkripsi akan muncul dialog sebagai berikut



Gambar 5. Dialog Dekripsi

Proses di atas akan menghasilkan file ~temp yang mana hanya terlihat pada saat program editor teks ini masih berjalan.

4.1 Pengaruh Secure Editor Terhadap Kinerja User

Kinerja *user* dalam menjalankan tugas pengolahan data dipengaruhi oleh berbagai factor antara lain, kecepatan prosesor komputer yang digunakan, pemilihan *device* pendukung, *software* dalam hal ini editor teks. Dengan penambahan fungsi keamanan data, tentunya akan menambah sedikit waktu dan cara pengolahannya.

Fasilitas *secure* pada editor teks ini terbagi dalam tiga bagian proses, yaitu pada proses *setup key*, proses enkripsi dan proses dekripsi.

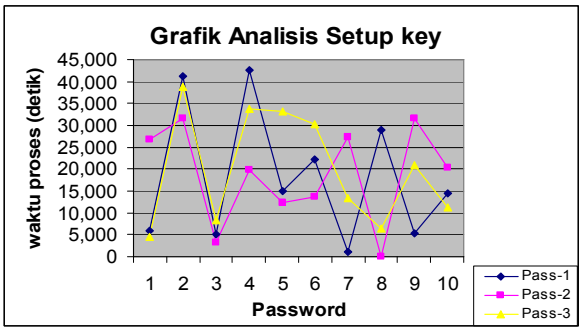
1. Proses Setup Key

   Waktu yang dibutuhkan untuk melakukan proses *setup key* pada editor teks HBR Crypto Versi 1.0, dijelaskan dengan menggunakan table 1. Pada tabel ini diuraikan lama proses *setup key* dengan *password* dengan panjang yang sama, dan *password* yang berbeda namun panjangnya sama, serta *password* yang berbeda-beda panjangnya.

   Tabel 1. Waktu proses *Setup key*

No	Password	Proses Setup Key (detik)	Password	Proses Setup Key (detik)	Password	Proses Setup Key (detik)
1	pwd	05,972	kursi	26,701	p	04,641
2	pwd	41,319	kasur	31,527	pw	38,773
3	pwd	05,023	mejaK	19,710	pwd	08,206
4	pwd	42,534	almri	12,395	pwd1	33,692
5	pwd	14,999	dapur	13,611	pwd12	33,252
6	pwd	22,314	mulia	31,724	pwd123	30,324
7	pwd	01,018	spatu	20,277	pwd1234	13,356
8	pwd	28,865	kacaR	27,395	pwd12345	06,423
9	pwd	05,462	lensa	05.914	pwd123456	20,914
10	pwd	14,560	mejaR	03,240	pwd1234567	11,307

Dari tabel di atas kiranya dapat lebih jelas dengan grafik berikut :



Gambar 6. Grafik Analisis *Setup Key*

Dari grafik di atas terlihat bahwa *password* tidak mempengaruhi lama proses *setup key*, lama proses ini tergantung pada proses random yang terjadi di dalam program untuk menentukan nilai p dan q sebagai variabel yang dipakai untuk menghitung/menentukan kunci pada *cryptosystem* RSA. Kiranya untuk lebih mempercepat proses diperlukan *device* pendukung yang mempunyai performa yang lebih tinggi. Komputer yang digunakan untuk analisis di atas adalah komputer PC dengan spesifikasi prosesor Intel Pentium III 600 MHz, RAM Apacher 128 MB/133 MHz, dengan harddisk Quantum berukuran 10.2 GB 7200 rpm.

2. Proses Enkripsi

Waktu yang dibutuhkan untuk melakukan proses enkripsi pada editor teks HBR Crypto Versi 1.0, dijelaskan dengan menggunakan tabel 2. Pada tabel ini diuraikan lama proses pada file teks yang sama dengan menggunakan kunci publik yang sama dan kunci publik berbeda.

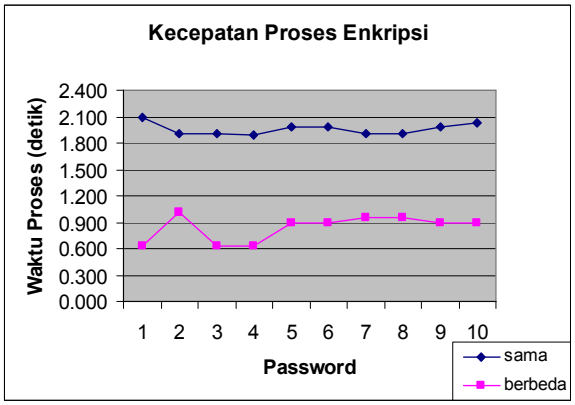
Tabel 2. Waktu Proses Enkripsi menggunakan kunci public yang sama

No	File Teks	Kunci public	Panjang kunci	Proses Enkripsi (detik)
1	data1	kunci-1	600 byte	2,094
2	data1	kunci-1	600 byte	1,909
3	data1	kunci-1	600 byte	1,909
4	data1	kunci-1	600 byte	1,898
5	data1	kunci-1	600 byte	1,979
6	data1	kunci-1	600 byte	1,979
7	data1	kunci-1	600 byte	1,909
8	data1	kunci-1	600 byte	1,909
9	data1	kunci-1	600 byte	1,979
10	data1	kunci-1	600 byte	2,025

Tabel 3. Waktu Proses Enkripsi dengan kunci berbeda

No	File Teks	Kunci public	Panjang Kunci	Proses Enkripsi (detik)
1	data1	a	434 byte	0,636
2	data1	ab	600 byte	1,018
3	data1	abc	434 byte	0,636
4	data1	abcd	434 byte	0,636
5	data1	abcde	468 byte	0,891
6	data1	abcdef	468 byte	0,891
7	data1	abcdefg	468 byte	0,949
8	data1	abcdefg1	468 byte	0,960
9	data1	abcdefg12	468 byte	0,891
10	data1	abcdefg123	468 byte	0,891

Dari tabel di atas diperoleh grafik sebagai berikut :



Gambar 7. Grafik perbandingan proses enkripsi dengan kunci sama dan kunci berbeda

Dari tabel 2, kunci-1 ditentukan oleh satu proses setup key maka besar file/panjang kunci adalah sama besar. Kemudian proses enkripsi waktu proses yang terjadi cukup bervariasi. Hal ini bisa dilihat pada algoritma proses enkripsi, bahwa plaintext dienkripsi menggunakan RC4, dalam pembuatan kunci RC4 terjadi proses random dalam menentukan *key* yang digunakan sebagai pembangkit kuncu RC4.

Dari tabel 3, kunci yang digunakan adalah berbeda-beda dan password yang dipakai sesuai dengan nama file kuci. Ternyata panjang kunci yang terjadi bervariasi tidak berdasarkan panjang password tetapi tergantung pada pemilihan karakter pada *password*. Terlihat pada tabel, bahwa semakin kecil panjang file kunci, maka semakin singkat waktu proses enkripsi data.

Selanjutnya dilihat pula pada tabel 4, lama proses enkripsi data pada file teks dengan panjang bervariasi dengan kunci publik yang sama.

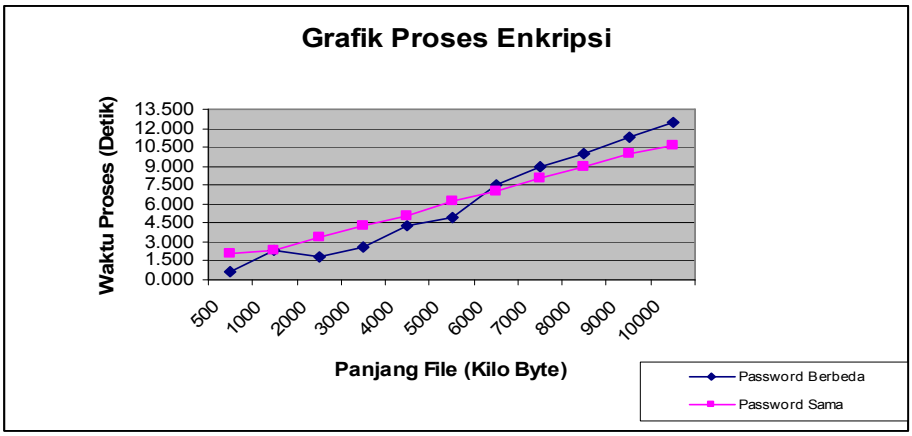
Tabel 4. Waktu Proses Enkripsi sesuai Panjang File dengan Kunci sama

No	File Teks	Panjang File (byte)	Kunci public	Proses Enkripsi (detik)
1	data1	500 Kbyte	kunci-1	2,037
2	data2	1.000 Kbyte	kunci-1	2,280
3	data3	2.000 Kbyte	kunci-1	3,368
4	data4	3.000 Kbyte	kunci-1	4,259
5	data5	4.000 Kbyte	kunci-1	5,081
6	data6	5.000 Kbyte	kunci-1	6,168
7	data7	6.000 Kbyte	kunci-1	7,048
8	data8	7.000 Kbyte	kunci-1	8,020
9	data9	8.000 Kbyte	kunci-1	8,900
10	data10	9.000 Kbyte	kunci-1	9,849
11	data11	10.000 Kbyte	kunci-1	10,682

Tabel 5. Waktu Proses Enkripsi sesuai Panjang File dengan Kunci yang Berbeda

No	File Teks	Panjang File (byte)	Kunci public	Proses Enkripsi (detik)
1	data1	500 Kbyte	a	0,636
2	data2	1.000 Kbyte	ab	2,361
3	data3	2.000 Kbyte	abc	1,782
4	data4	3.000 Kbyte	abcd	2,546
5	data5	4.000 Kbyte	abcde	4,259
6	data6	5.000 Kbyte	abcdef	4,953
7	data7	6.000 Kbyte	abcdefg	7,569
8	data8	7.000 Kbyte	abcdefg1	8,958
9	data9	8.000 Kbyte	abcdefg12	9,976
10	data10	9.000 Kbyte	abcdefg123	11,249
11	data11	10.000 Kbyte	abcdefg1234	12,453

Dari kedua tabel di atas diperoleh grafik sebagai berikut :



Gambar 8. Grafik Kecepatan proses Enkripsi pada Panjang File yang bervariasi dengan Kunci yang Sama dan Berbeda

Dari tabel 4 dan tabel 5 serta grafik pada gambar 8 dapat diambil suatu kesimpulan, bahwa kecepatan proses enkripsi tergantung dari besarnya file kunci public yang dihasilkan oleh proses *setup key*. Sedangkan panjang file kunci tergantung pada karakter yang terpilih untuk pasangan *password* bukan pada panjang *password* yang digunakan. Dan terlihat untuk pemakaian kunci yang sama dan berbeda, menunjukkan kenaikan yang konstan.

3. Proses Dekripsi

Waktu yang dibutuhkan untuk melakukan proses dekripsi pada editor teks HBR Crypto Versi 1.0, dapat dilihat pada tabel 6 Pada tabel ini diuraikan lama proses pada file teks yang sama dengan *password* yang berbeda.

Tabel 6. Waktu Proses Dekripsi dengan Kunci yang Sama

No	File Teks	Panjang File	Password	Proses Dekripsi (detik)
1	data1	500 Kbyte	kunci-1	1,527
2	data2	1000 Kbyte	kunci-1	1,840
3	data3	2000 Kbyte	kunci-1	2,673
4	data4	3000 Kbyte	kunci-1	4,201
5	data5	4000 Kbyte	kunci-1	5,590
6	data6	5000 Kbyte	kunci-1	8,645
7	data7	6000 Kbyte	kunci-1	8,703
8	data8	7000 Kbyte	kunci-1	9,097
9	data9	8000 Kbyte	kunci-1	10,995
10	data10	9000 Kbyte	kunci-1	13,090
11	data11	10000 Kbyte	kunci-1	18,877

Dari tabel di atas diperoleh grafik sebagai berikut :

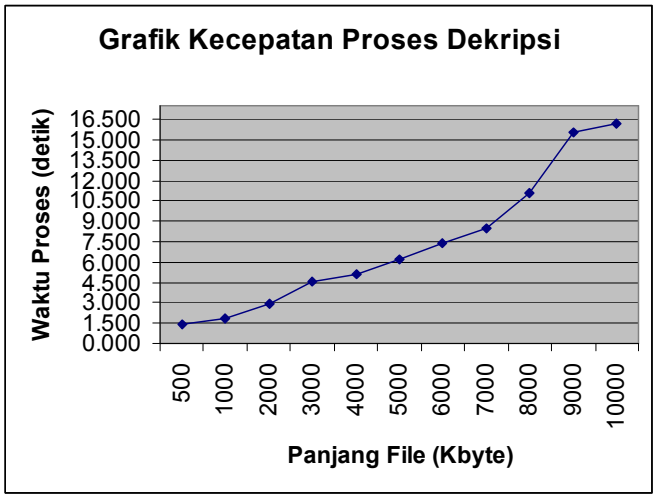
Gambar 9. Grafik Kecepatan Proses Dekripsi pada Panjang File yang berbeda dan Kunci yang Sama

Selanjutnya dilihat pula pada tabel 7 lama proses dekripsi data pada file teks dengan panjang bervariasi dengan *password* yang sama.

Tabel 7. Waktu Proses Dekripsi sesuai Panjang File dengan Kunci yang Berbeda

No	File Teks	Panjang File	Password	Proses Dekripsi (detik)
1	data1	500 Kbyte	a	1,458
2	data2	1000 Kbyte	ab	1,840
3	data3	2000 Kbyte	abc	2,986
4	data4	3000 Kbyte	abcd	4,571
5	data5	4000 Kbyte	abcde	5,081
6	data6	5000 Kbyte	abcdef	6,226
7	data7	6000 Kbyte	abcdefg	7,430
8	data8	7000 Kbyte	abcdefg1	8,518
9	data9	8000 Kbyte	abcdefg12	11,064
10	data10	9000 Kbyte	abcdefg123	15,567
11	data11	10000 Kbyte	abcdefg1234	16,145

Dari tabel di atas diperoleh grafik sebagai berikut :



Gambar 10. Grafik Kecepatan proses Dekripsi pada Panjang File yang berbeda dan Kunci yang Berbeda

Dari tabel 6 dan tabel 7 serta grafik pada Gambar 9 dan Gambar 10 dapat diambil suatu kesimpulan bahwa pada editor HBR Crypto 1.0, kecepatan proses dekripsi tergantung dari besarnya file kunci public yang dihasilkan oleh proses *setup key*. Sedangkan panjang/besarnya file kunci tergantung pada karakter yang terpilih untuk pasangan *password* bukan pada panjang *password* yang digunakan. Pada proses dekripsi waktu prosesnya sedikit lebih lama dibandingkan pada proses enkripsi mengingat pada proses dekripsi terjadi pemeriksaan pada beberapa atribut.

5. KESIMPULAN

Berdasarkan pada hasil penelitian, analisis dan pembangunan sistem pengamanan pesan menggunakan teknik *hybridcryptosystem* dapat ditarik beberapa kesimpulan sebagai berikut :

1. Keamanan data di antaranya berada pada sisi kunci, semakin besar/panjang kunci semakin lama waktu yang dibutuhkan untuk membongkarnya. Kunci yang besar/panjang tidak memungkinkan *user* untuk menghafalkannya, maka pada HBRCrypto versi 1.0 kunci dibangkitkan oleh suatu *password*. *Password* dimungkinkan mudah untuk tersadap oleh *user* lain yang tidak berhak, maka pada HBRCrypto versi 1.0 pun dibatasi penerimanya dengan menyebutkan *account name* penerima. Kedua atribut ini harus valid untuk dapat membuka file terenkripsi menjadi bentuk *plaintext* yang asli.
2. Pada proses *setup key* menghasilkan kunci publik yang disimpan dalam *public key file*, besar/panjang file sangat beragam, namun keberagamannya tidak ditentukan oleh panjangnya *password* akan tetapi tergantung pada pemilihan karakter pada *password*.

3. Pada proses enkripsi mempunyai variasi kecepatan proses yang sangat beragam, walaupun menggunakan kunci yang sama ataupun kunci yang berbeda. Karena pada proses enkripsi juga terdapat proses *random* dalam pembangkitan kunci RC4. Sehingga kecepatan pada proses enkripsi tidak tergantung pada panjang *password*. Dan proses *random* dalam pembangkitan kunci RC4 dapat menepis kelemahan dari algoritma ini karena tidak dimungkinkan kunci RC4 akan sama pada setiap proses enkripsi.

5. DAFTAR PUSTAKA

Ahyar, S., 2001, *Sistem Keamanan File Dan Folder Data Menggunakan Algoritma Blowfish Dengan Kunci Simetrik*, Paper Mata Kuliah Keamanan Sistem Informasi, Bandung : TE – ITB.

Bridgeca., 1999, *Cryptography and Public Key Infrastruktur*, http://www.secude.com/bridgeca/crypt_and_pki_guide_e.pdf

Dian E., 2001, *Mekanisme Discretionary Access Control untuk Database Security*, PAPER Keamanan Sistem Lanjut, Bandung, TE - ITB

Saskia Groenewegen and Andreas Buchner, 1999, *The Basics of Cryptography*, http://www.crypto-nuitari-de.htm

Avon B., 2004, *Enkripsi Data Kunci Simetris Dengan Algoritma Kriptografi Loki97* , Paper Keamanan Sistem Lanjut, Bandung, TE - ITB

Eastlake, D., September 2001, *US Secure Hash Algorithm 1 (SHA1)*, RFC 3174

Schneier, Bruce., 1996, *Applied Cryptography : Protocols, Algorithms, and Source Code in C*, 2nd Edition John Wiley & Sons Inc.

Slamet., 2003, *Keamanan Jaringan Komputer*, Paper Keamanan Sistem Informasi, Bandung, TE - ITB