APLIKASI PENGAMANAN DOKUMEN *OFFICE* DENGAN ALGORITMA KRIPTOGRAFI BLOWFISH

EKKY PRATAMA
Program Studi Teknik Informatika
Fakultas Ilmu Komputer Universitas Dian Nuswantoro
Email : pratamaaa@hotmail.com

ABSTRAK

Teknologi informasi dan komunikasi yang berkembang semakin pesat melalui dunia maya, sangat berperan penting untuk mengefisienkan waktu dan mempermudah pekerjaan. Komunikasi data seperti halnya pengiriman dokumen penting, sering kali dilakukan tanpa memikirkan segi kemanan data yang dikirim. Dari permasalahan tersebut, dibutuhkan penyandian pada dokumen terutama dokumen *office*, karena sebagian besar terbiasa dengan aplikasi *Microsoft Office* yang sangat memudahkan siapa saja ketika menggunakan aplikasi ini. Dengan adanya aplikasi pengamanan dokumen menggunakan metode kriptografi (*enkripsi* dan *dekripsi*) yang menerapkan algoritma simetris *Blowfish* diharapkan dokumen rahasia akan aman dan tidak bocor kepada penyadap atau pihak ketiga yang tidak bertanggungjawab.

Kata Kunci : Aplikasi, Blowfish , Kriptografi

1. Pendahuluan

1.1 Latar Belakang

Pertukaran data melalui jaringan komputer terutama *internet*, sangat mungkin dilakukan karena tentunya akan mempercepat dan memudahkan proses pertukaran data terutama untuk pertukaran data dengan jarak yang jauh. Sebagai contoh adalah pertukaran data yang dilakukan oleh sebuah kantor pusat yang ditujukan kepada kantor cabang. Akan terasa lamban dan tidak efisien jika pertukaran data tersebut dilakukan dengan pendistribusian secara manual dibandingkan melalui media *internet*.

Banyak keuntungan yang dapat diperoleh dengan melakukan pertukaran data melalui media jaringan komputer khususnya *internet* salah satunya yaitu untuk mengefisienkan waktu mengingat pentingnya data yang didistribusikan tersebut sampai pada penerimanya tepat waktu. Banyak jenis data yang sering dipergunakan untuk pertukaran data melalui media *internet* diantaranya seperti gambar, video, audio, dokumen *Office* dan lain sebagainya.

Disamping banyaknya keuntungan yang diperoleh, ada juga bahaya yang muncul dalam proses pertukaran data melalui media *internet*, salah satunya adalah pencurian data yang dilakukan pihak ketiga yang tidak bertanggungjawab yang bertujuan untuk kepentingan pribadi. Pada

dasarnya, pengiriman data melalui media *internet* tersebut tidak ada pengamanan terhadap isi dari data itu sendiri, sehingga pada saat proses pengiriman data, seseorang dengan mudah dapat mencuri data dan langsung dapat mengetahui isi dari data tersebut.

Semua hal diatas merupakan gambaran bahwa media *internet* sekarang menjadi jalur pertukaran yang sangat vital untuk data yang bersifat rahasia. Oleh karena itu, dibutuhkan teknik pengamanan data untuk menghindari penyadapan terhadap konten data yang digunakan sebagai objek pertukaran.

Berdasarkan paparan dan analisa dari masalah diatas, maka penulis mengambil judul "Aplikasi Pengamanan Dokumen *Office* dengan Algoritma Kriptografi *Blowfish*." sebagai salah satu alternatif mengatasi masalah keamanan data dari pencurian data baik yang tidak penting maupun yang penting dan rahasia.

1.2 Rumusan Masalah

Berdasarkan penjelasan dari latar belakang yang telah tersebut di atas, maka permasalahan yang akan dianalisa oleh penulis dalam pembuatan laporan Tugas Akhir ini dapat dirumuskan :

"Bagaimana mengamankan dokumen yang digunakan sebagai objek pertukaran data agar terhindar dari pencurian dan penyadapan oleh pihak yang tidak bertanggungjawab".

1.3 Tujuan Penelitian

Tujuan dari penelitian Tugas Akhir ini adalah membangun aplikasi pengamanan dokumen dengan teknik kriptografi.

1.4 Batasan Masalah

Untuk menghindari perluasan masalah pada pembuatan aplikasi pengamanan dokumen *Office*, maka dalam hal ini akan dibatasi pemasalahan pada :

a. Enkripsi dan dekripsi *text* pada *file Microsoft Offtice 2003* menggunakan Algoritma Kriptografi *Blowfish.*
b. Enkripsi dan dekripsi *text* pada *file Microsoft Office 2003* diantaranya *Microsoft* Word dan *Microsoft Excel*,
c. Aplikasi pengamanan dokumen *Office* ini dibuat dengan menggunakan bahasa pemrograman Java.
d. Aplikasi pengamanan dokumen *Office* ini dibuat dengan memanfaatkan *library* yang ada pada Java.

2. Tinjauan Pustaka

2.1 Algoritma *Blowfish*

Blowfish atau disebut juga *OpenPGP.Chiper.4* adalah enkripsi yang termasuk dalam golongan *Symmetric Cryptosystem*. Algoritma kunci simetrik *cipher blok* yang dirancang pada tahun 1993 oleh Bruce Schneider untuk menggantikan DES (*Data Encryption Standard*).

Algoritma *Blowfish* dibuat untuk digunakan pada komputer yang mempunyai mikroprosesor besar (32 bit ke atas dengan *cache* data yang besar). Pada saat itu banyak sekali rancangan algoritma yang ditawarkan,

namun hampir semua terhalang oleh paten atau kerahasiaan pemerintah Amerika. Schneier menyatakan bahwa *blowfish* bebas paten dan akan berada pada domain publik. Dengan pernyataan Schneier tersebut *blowfish* telah mendapatkan tempat di dunia kriptografi, khususnya bagi masyarakat yang membutuhkan algoritma kriptografi yang cepat, kuat, dan tidak terhalang oleh lisensi.

Blowfish dirancang dan diharapkan mempunyai kriteria perancangan yang diiginkan sebagai berikut :

1. Cepat, *Blowfish* melakukan enkripsi data pada microprocessor 32-bit dengan rate 26 *clock cycles per byte*.
2. *Compact*, *Blowfish* dapat dijalankan pada memory kurang dari 5K.
3. Sederhana, *Blowfish* hanya menggunakan operasi – operasi sederhana, *Blowfish* hanya menggunakan operasi – operasi sederhana, seperti penambahan, XOR, dan *lookup* tabel pada operan 32-bit.
4. Memiliki tingkat keamanan yang bervariasi, panjang kunci yang digunakan oleh *Blowfish* dapat bervariasi dan bisa sampai sepanjang minimal 32-bit, maksimal 448 -bit, *Multiple* 8 bit, default 128 bit.

Namun, dalam penerapannya sering kali algortima ini menjadi tidak optimal. Karena strategi implementasi yang tidak tepat. Algoritma *Blowfish* akan lebih optimal jika digunakan untuk aplikasi yang tidak sering berganti kunci, seperti jaringan komunikasi atau enkripsi *file* otomatis. [1]

2.2 Struktur Algoritma *Blowfish*

Blowfish merupakan *cipher blok* yang berarti selama proses enkripsi dan dekripsi, *blowfish* bekerja dengan membagi pesan menjadi blok-blok bit dengan ukuran sama panjang, yaitu 64-bit dengan panjang kunci bervariasi yang mengenkripsi data dalam 8 byte blok. Pesan yang bukan merupakan kelipatan 8 byte akan ditambahkan bit-bit tambahan (*padding*) sehingga ukuran untuk tiap blok sama. Algoritma *Blowfish* terdiri dari dua bagian yaitu *key expansion* dan enkripsi data.

2.2.1 Key Expansion

Key expansion berfungsi untuk mengkonversikan sebuah kunci sampai 56 byte (448 bit) menjadi beberapa array subkey dengan total 4168 byte.

2.2.2 Enkripsi Data

Enkripsi data, proses ini terjadi di dalam jaringan *feistel* dan terdiri dari iterasi fungsi sederhana sebanyak 16 kali putaran. Setiap putaran terdiri dari permutasi *key-dependent* serta substitusi kunci dan *data-dependent*. Semua operasi merupakan XOR dan penjumlahan (*addition*) pada variable 32 bit. Operasi penambahan yang terjadi hanya merupakan empat *index array data lookup* pada setiap iterasi.

Blowfish menggunakan subkunci besar yang harus dihitung sebelum enkripsi dan dekripsi data. Algoritma *Blowfish* menerapkan jaringan *Feistel* yang terdiri dari 16 putaran.*Input* adalah elemen 64-

bit, X untuk alur algoritma enkripsi dengan metode *Blowfish* dijelaskan sebagai berikut: [2]

1. Bentuk inisial P-array sebanyak 18 buah (P1, P2, ..., P18) masing-masing bernilai 32-bit. Array P terdiri dari delapan belas kunci 32-bit subkunci P1,P2, ..., P18
2. Bentuk S-box sebanyak 4 buah masing-masing bernilai 32-bit yang memiliki masukan 256. Empat 32-bit S-box masing-masing mempunyai 256 entri :

   $S_{1,0}, S_{1,1}, \ldots\ldots\ldots\ldots, S_{1,255}$

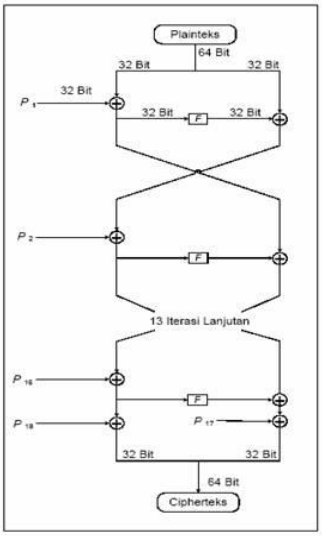
   $S_{2,0}, S_{2,1}, \ldots\ldots\ldots\ldots, S_{2,255}$

   $S_{3,0}, S_{3,1}, \ldots\ldots\ldots\ldots, S_{3,255}$

   $S_{4,0}, S_{4,1}, \ldots\ldots\ldots\ldots, S_{4,255}$

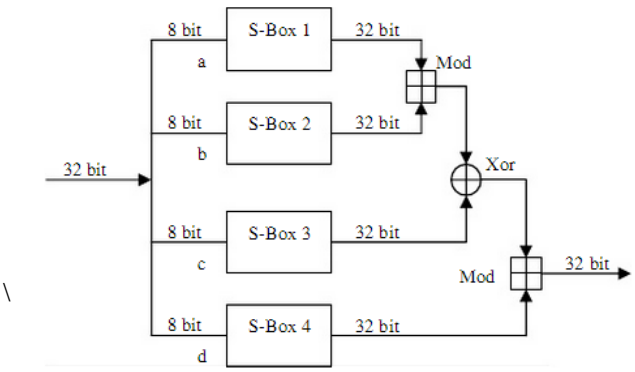
3. Plaintext yang akan dienkripsi diasumsikan sebagai masukan, Plaintext tersebut diambil sebanyak 64-bit, dan apabila kurang dari 64-bit maka kita tambahkan bitnya, supaya dalam operasi nanti sesuai dengan datanya.
4. Hasil pengambilan tadi dibagi 2, 32-bit pertama disebut XL, 32-bit yang kedua disebut XR.
5. Selanjutnya lakukan operasi XL = XL xor Pi dan XR = F(XL) xor XR
6. Hasil dari operrasi diatas ditukar XL menjadi XR dan XR menjadi XL.
7. Lakukan sebanyak 16 kali, perulangan yang ke-16 lakukan lagi proses penukaran XL dan XR.
8. Pada proses ke-17 lakukan operasi untuk XR = XR xor P17 dan XL = XL xor P18.
9. Proses terakhir satukan kembali XL dan XR sehingga menjadi 64-bit kembali.

Blowfish menggunakan jaringan Feistel yang terdiri dari 16 buah putaran. Skema jaringan Feistel dapat dilihat pada Gambar 2.1.



Gambar 2.1 : Jaringan *Feistel* untuk Algoritma *Blowfish*

Algoritma *Blowfish* memiliki keunikan dalam hal proses dekripsi, yaitu proses dekripsi dilakukan dengan urutan yang sama persis dengan proses enkripsi, hanya saja pada proses dekripsi P1, P2, …, P18 digunakan dalam urutan yang terbalik. Dalam algoritma *Blowfish* juga terdapat fungsi f. Berikut ini gambar mengenai fungsi f tersebut pada Gambar 2.2 [22]



Gambar 2.6 : Fungsi F dalam *Blowfish*

Fungsi F adalah bagi XL, menjadi empat bagian 8-bit : a,b,c dan d.
$F(XL)=((S1,a + S2,b \bmod 2^{32}) xor\ S3,c) + S4,c \bmod 2^{32}$
Subkunci dihitung menggunakan algoritma *Blowfish*, metodanya adalah sebagai berikut :

1. Pertama-tama inilialisasi P-array dan kemudian empat S-box secara berurutan dengan string yang tetap. String ini terdiri atas digit hexadesimal dari Pi.
2. XOR P1 dengan 32-bit pertama kunci, XOR P2 dengan 32-bit kedua dari kunci dan seterusnya untuk setiap bit dari kunci

(sampai P18).Ulangi terhadap bit kunci sampai seluruh P-array di XOR dengan bit kunci.

3. Enkrip semua string nol dengan algoritma *Blowfish* dengan menggunakan subkunci seperti dijelaskan pada langkah (1) dan (2).
4. Ganti P1 dan P2 dengan keluaran dari langkah (3).
5. Enkrip keluaran dari langkah (3) dengan algoritma *Blowfish* dengan subkunci yang sudah dimodifikasi.
6. Ganti P3 dan P4 dengan keluaran dari langkah (5).
7. Lanjutkan proses tersebut, ganti seluruh elemen dari P-array, kemudian seluruh keempat S-box berurutan, dengan keluaran yang berubah secara kontiyu dari algoritma *Blowfish*.

Total yang diperlukan adalah 521 iterasi untuk menghasilkan semua subkunci yang dibutuhkan. Aplikasi kemudian dapat menyimpan subkunci ini dan tidak membutuhkan langkah-langkah proses penurunan berulang kali, kecuali kunci yang digunakan berubah.

Untuk deskripsi sama persis dengan enkripsi, kecuali pada P-array (P1,P2,.....,P18) digunakan dengan urutan terbalik atau di inverskan. [3]

3. Metodologi Penelitian

3.1 Pengumpulan Kebutuhan

Kegiatan yang dilakukan pada tahapan ini yaitu dengan mengenali dan mendefinisikan masalah pengamanan dokumen *Office* dan mencari alternatif pemecahannya.

3.2 Perancangan

Pada tahap ini, penulis merancang struktur data, arsitektur perangkat lunak, detail prosedur, karakteristik tampilan yang akan disajikan serta memenuhi kebutuhan pemakai (*user*) dari program yang akan dibuat. Penulis menggunakan UML (*Unified Modelling Language*) untuk perancangan dan mendokumentasikan serta sebagai bahasa pemodelan sistem piranti lunak

3.3 Implementasi

Pada tahap ini dilakukan proses transformasi dokumen desain kebentuk kode yang dapat dimplementasikan oleh mesin. Tahap implementasi ini akan menggunaan beberapa *tool* pengembangan yang meliputi : konversi dokumen *Office* menjadi rangaian bit, mengenkripsi *plaintext* dan mendeskripsikan *ciphertext* dengan menggunakan kriptografi *Blowfish*.

3.4 Pengujian (*Testing*)

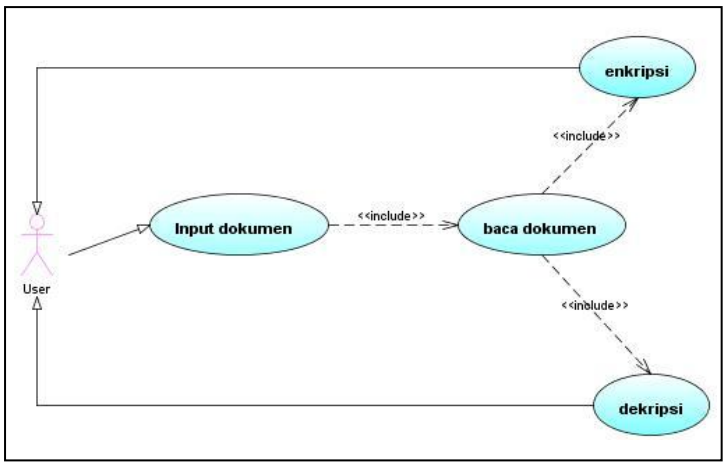
Tahap pengujian ini dilakukan dengan *Black box testing* untuk menjamin aplikasi enkripsi dokumen *Office* yang dikembangkan dapat benar-benar bebas dari kesalahan-kesalahan pada *interface*, kesalahan

pada performansi dan fungsi yang salah atau hilang. Tahap pengujian ini juga bertujuan untuk menunjukan tentang cara beroprasinya, apakah masukan data dan keluaran data telah berjalan sebagaimana yang diharapkan. Evaluasi dari aplikasi enkripsi dokumen *Office* ini berbasis kuesioner.

4. Perancangan dan Hasil Implementasi

4.1 Perancangan *Use Case Diagram*

Berikut merupakan *use case diagram* yang penulis gunakan dalam membuat aplikasi pengamanan dokumen *Office* ini.



Gambar 4.1 : *Use case diagram* aplikasi pengamanan dokumen *Office*

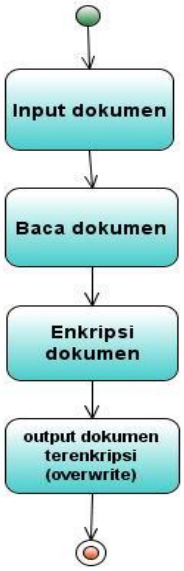
Dalam *Use Case* ini, untuk melakukan enkripsi dan dekripsi dokumen, pertama *user* melakukan penginputan dokumen yang akan dienkripsi atau di dekripsi. Setelah *user* menginputkan dokumen, aplikasi pengamanan dokumen *Office* tersebut akan membaca dokumen tersebut.

Setelah dokumen berhasil dibaca, *user* melakukan perintah untuk selanjutnya dokumen akan dienkripsi atau didekripsi. Jika *user* melakukan perintah enkripsi, maka setelah dokumen terbaca, proses enkripsi dilakukan. Hasil dari enkripsi itu akan kembali lagi pada *user* sebagai dokumen *cipher*. Begitu pula dengan perintah dekripsi, setelah pembacaan dokumen *cipher*, maka aplikasi akan memproses dokumen tersebut untuk di dekripsi, sehingga menghasilkan dokumen asli untuk *user*.

4.2 Perancangan *Activity Diagram*

Activity diagram adalah representasi grafis dari alur kerja tahapan aktifitas. Diagram yang penulis buat ssaat ini menggunakan 2 macam model diagram yaitu diagram pada saat enkripsi dokumen dan dekripsi dokumen.

a. *Activity Diagram* Enkripsi



Gambar 4.2 : *Activity diagram* enkripsi dokumen

Pertama adalah *user* melakukan input dokumen/memilih dokumen yang akan di enkripsi. Kemudian sistem akan menbaca dokumen yang telah *user* inputkan tadi. Setelah itu, dokumen yang telah dibaca oleh sistem, digunakan sebagai *plaintext* untuk di enkripsi. *Output* yang diperoleh dari hasil enkripsi adalah berupa dokumen *cipher*.

b. *Activity Diagram* Dekripsi

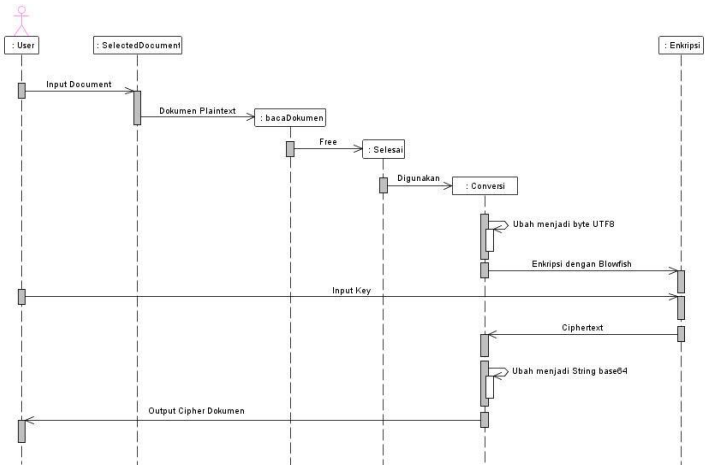
Gambar 4.3 : *Activity diagram* dekripsi dokumen

Tidak jauh berbeda dengan *activity* enkripsi, pertama *user* menginputkan dokumen yang sudah terenkripsi. Kemudian sistem akan membaca dokumen yang kemudian akan di didekripsi oleh sistem. *Output* yang diperoleh dari hasil dekripsi adalah dokumen asli / *plaintext.*

4.3 Perancangan *Sequence Diagram*

Sequence diagram adalah suatu diagram yang menggambarkan interaksiantar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Pada *sequence diagram* ini terdapat 2 model yang penulis buat yaitu pada saat enkripsi dokumen dan dekripsi dokumen.

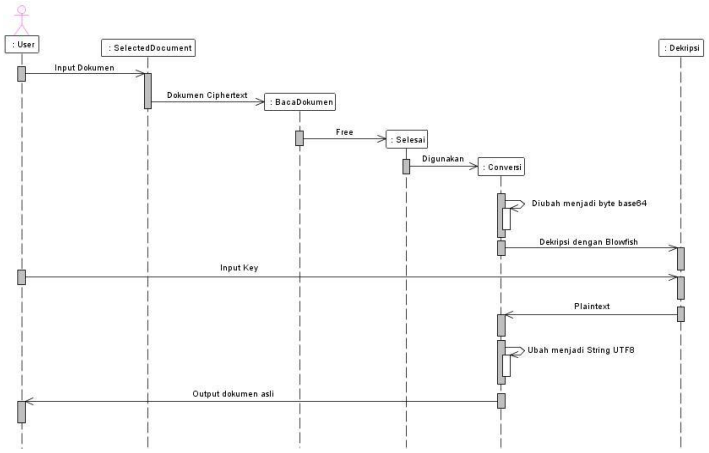
a. *Sequence Diagram* Enkripsi



Gambar 4.4 : *Sequence diagram* enkripsi

Pada proses enkripsi ini, *user* menginputkan dokumen untuk di enkripsi. Setelah itu, aplikasi ini akan membaca dokumen yang kemudian diubah tipe datanya menjadi *byte* dengan format UTF8. Kemudian dokumen tersebut digunakan sebagai *plaintext* yang siap untuk di enkripsi. Setelah itu, user memasukan *key* sebagai kunci untuk mengenkripsi dokumen. Setelah dokumen terenkripsi, type datanya akan diubah menjadi *Base64* berupa tipe data *string.* Hasilnya akan dikembalikan lagi kepada *user* berupa *ciphertext* atau dokumen yang telah terenkripsi.

b. *Sequence Diagram* Dekripsi



Gambar 4.5 : *Sequence diagram* dekripsi

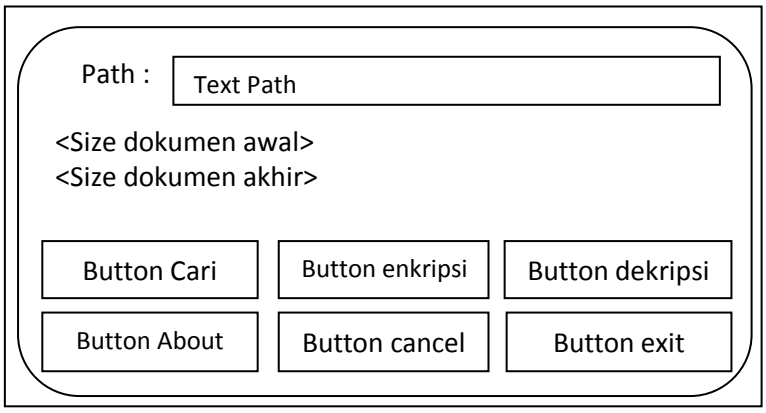
Pada proses dekripsi, *user* menginputkan dokumen yang telah terenkripsi (*ciphertext*). Aplikasi akan membaca dokumen yang telah diinputkan oleh *user*. Kemudian dokumen akan diubah menjadi *byte* dengan *Base64* dan siap untuk di dekripsi.

Setelah itu, *user* menginputkan *key* sebagai kunci untuk mendekripsikan dokumen. *Key* yang diinputkan adalah *key* yang sama, yang digunakan dalam proses enkripsi tadi. Setelah berhasil di dekripsi, dokumen akan diubah menjadi s*tring* dengan format UTF8 kemudian menjadi sebuat *output* yang dikembalikan lagi kepada *user* berupa teks asli.

4.4 Perancangan Antar Muka (*Interface*)

Perancangan antarmuka aplikasi pengamanan dokumen *office* ini dibuat dengan menggunakan *NetBean IDE* 6.5 karena pada *Netbean* tersedia *tools* yang digunakan dalam implementasi program. Sehingga akan memudahkan dalam mendisain antarmuka maupun dalam implementasi sistem.

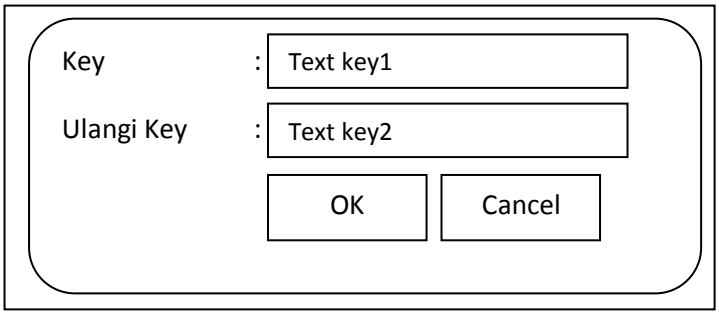
a. Menu Utama



Gambar 4.6 : *Storyboard* menu utama

Menu utama ini adalah tampilan yang akan muncul pada saat pertama *user* menjalankan apikasi pengamanan dokumen *office*. Pada menu utama ini, *user* dapat melakukan enkripsi dokumen dan dekripsi dokumen.

b. Menu *Input Key*



Gambar 4.7 : *Storyboard* menu *input key*

Pada jendela ini, *user* dapat memasukan key untuk mengenkripsi dokumen atau mendekripsi dokumen. *Text key1* adalah untuk masukan *key* yang minimal panjang adalah tiga karakter. Jika *user* memasukan key kurang dari tiga karakter, maka *text key2* tidak akan aktif. Untuk mengkonfirmasi *key*, user dapat menginputkan lagi key yang sama pada *text key2*. Jika *user* memasukan key yang berbeda pada *text key2*, maka *button OK* tidak akan aktif.

c. Menu *About Us*

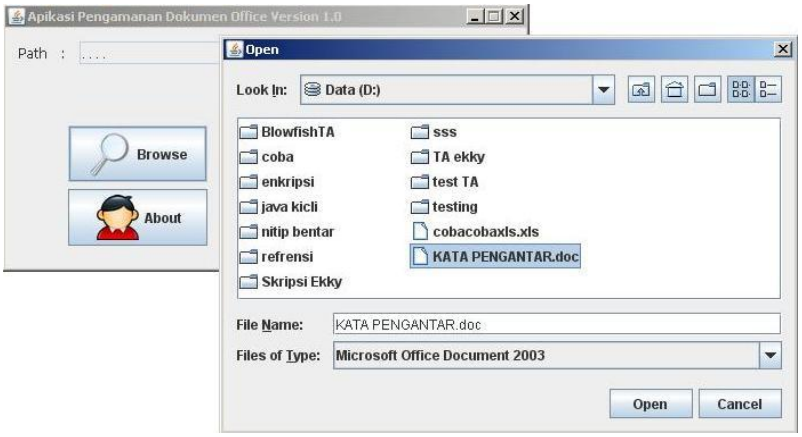
Gambar 4.8 : *Storyboard* menu *about us*

Menu *About Us* adalah menu yang memberikan informasi tentang pembuat aplikasi tersebut. Selain itu ada button *close* yang berfungsi untuk menutup jendela *about us* dan kembali pada jendela menu utama.

4.5 Hasil Implementasi

Gambar 4.9 : Menu utama

Pada tampilan menu utama, terdapat beberapa *button* antralain *button browse, button* enkripsi, *button* dekripsi, *button about, button cancel* dan *button exit. Button Browse* digunakan untuk memilih dokumen yang akan di inputkan untuk dienkripsi atai didekripsi. Berikut tampilan menu *button browse* :



Gambar 4.10 : Menu pilih dokumen

Pada tampilan pemilihan dokumen ini, penulis melakukan filter dokumen sehingga yang muncul dalam jendela pemilihan dokumen adalah dokumen bertipe *Microsoft Office Document 2003* dengan ekstensi *doc, xls, ppt* dan *mdb.* Setelah *user* memilih dokumen yang akan di enkripsi, maka dokumen tersebut dibaca oleh aplikasi pengamanan dokumen serta alamat dokumen tersebut ditampilkan dalam menu utama seperti gambar 3.11.

Gambar 4.11 : Tampilah dokumen telah dipilih

Setelah dokumen terbaca, user dapat melakukan enkripsi dengan memilih *button* enkripsi, kemudian akan muncul jendela untuk memasukan *key* seperti gambar dibawah ini :

Gambar 4.12 : Tampilah penginputan *key*

User memasukan *key* untuk mengenkripsi dokumen tersebut, kemudian mengulangi *key* yang sama untuk konfirmasi ulang *key*. Jika *key* yang dimasukan kurang dari tiga karakter maka label ulangi *key* tidak akan aktif, begitu juga dengan *button OK*, tidak akan aktif selama *user* memasukan kunci yang berbeda dengan label *key*. Hal ini bertujuan agar *user* tidak lupa akan *key* yang dibuat guna mengenkripsi dokumennya.

Gambar 4.13: Tampilah sukses terenkripsi

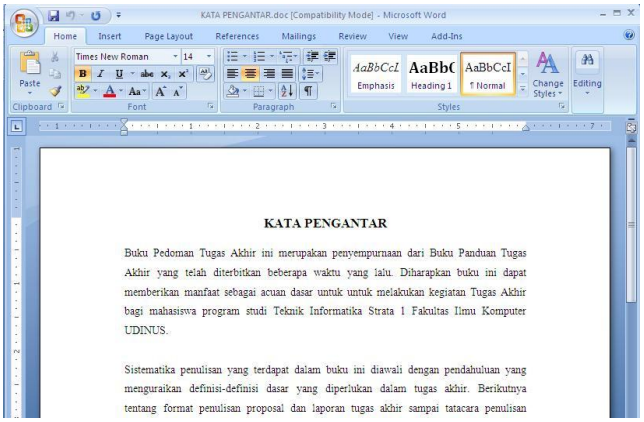
Setelah sukses terenkripsi, pada tampilan *form* akan muncul pemberitahuan bahwa dokumen telah sukses dienkripsi, dan akan muncul ukuran dokumen yang sudah di enkripsi seperti gambar diatas. Tidak jauh berbeda dengan proses enkripsi, proses dekripsi awalnya menginputkan dokumen yang telah di enkripsi melalui *button browse*. Untuk pembacaan dokumen terenkripsi sama dengan saat proses pembacaan dokumen sebelum terenkripsi.

4.6 Testing

Penulis melakukan beberapa percobaan untuk membuktikan kinerja aplikasi yang telah dibuat. Berikut adalah percobaan yang penulis lakukan untuk mengetahui kinerja aplikasi.

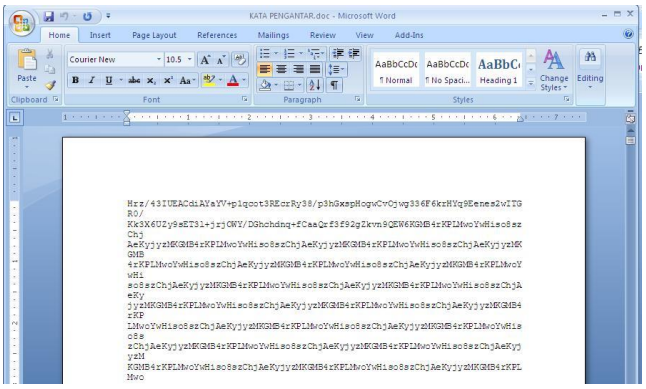
4.6.1 Testing pada Dokumen *Microsoft Word*

Penulis melakukan percobaan terhadap dokumen *Microsoft Office 2003* dengan ukuran dokumen yaitu 27 Kb. Isi dokumen bisa dilihat seperti gambar dibawah ini. Penulis akan mencoba mengenkripsi dokumen tersebut.



Gambar 4.14 : Dokumen *Microsoft Word 2003*

Setelah dilakukan enkripsi pada dokumen tersebut, maka isi dokumen *Microsoft Word* akan berubah menjadi tulisan yang tidak dapat terbaca seperti pada gambar 3.15

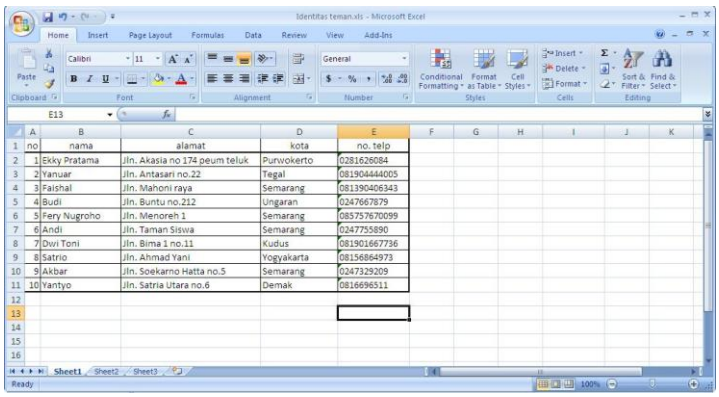


Gambar 4.15 : Dokumen *Microsoft Word* terenkripsi

Setelah dokumen berhasil di enkripsi, ukuran dokumen terenkripsi berubah menjadi 43 Kb. Kemudian dokumen *cipher* ini didekripsikan kembali dengan key yang sama, maka akan kembali menjadi dokumen asli, seperti sebelum dienkripsi dan *size*-nya juga akan kembali seperti semula yaitu 27 Kb dan dapat terbaca lagi.

4.6.2 Testing pada Dokumen *Microsoft Excel*

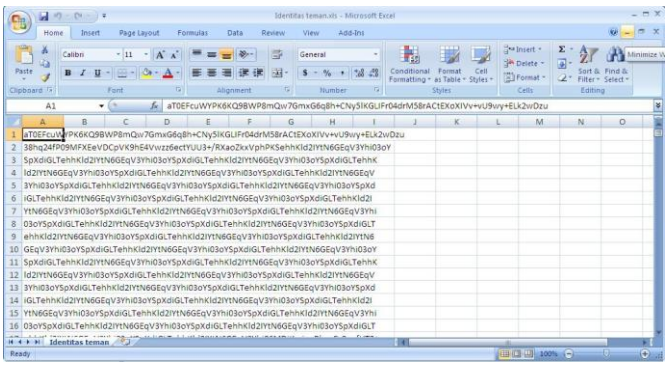
Penulis melakukan percobaan kedua yaitu enkripsi pada dokumen *Microsoft Excel* dengan ukuran 19 Kb.



no	nama	alamat	kota	no. telp
1	Ekky Pratama	Jln. Akasia no 174 peum teluk	Purwokerto	0281626084
2	Yanuar	Jln. Antasari no.22	Tegal	081904444005
3	Faishal	Jln. Mahoni raya	Semarang	081390406343
4	Budi	Jln. Buntu no.212	Ungaran	0247667879
5	Fery Nugroho	Jln. Menoreh 1	Semarang	085757670099
6	Andi	Jln. Taman Siswa	Semarang	0247755890
7	Dwi Toni	Jln. Bima 1 no.11	Kudus	081901667736
8	Satrio	Jln. Ahmad Yani	Yogyakarta	08156864973
9	Akbar	Jln. Soekarno Hatta no.5	Semarang	0247329209
10	Yantyo	Jln. Satria Utara no.6	Demak	0816696511

Gambar 3.16: Dokumen *excell* sebelum dienkripsi

Penulis melakukan enkripsi pada dokumen *Microsoft excel* seperti tampilan di atas. Awal mula dokumen asli memiliki *size* 18,5 Kb, dan setelah terenkripsi, isi dokumen tadi akan berubah menjadi karakter yang tidak dapat terbaca seperti gambar dibawah ini. Selain itu juga size dokumen *excel* tersebut berubah menjadi 29,1 Kb. Berikut gambar dokumen setelah di enkripsi :



Gambar 4.23 : Dokumen *excell* setelah dienkripsi

Kemudian penulis mencoba mendekripsikan dokumen tersebut, dan hasilnya *size* dokumen *excel* berubah menjadi seperti semula yaitu 18,5Kb dan dari dokumen *excell* tersebut, kembali seperti semula.

5. Penutup

5.1 Kesimpulan

Berikut adalah kesimpulan yang dapat ditarik dari pembahasan masalah ini:

1. Aplikasi ini dapat mengenkripsi dan mendekripsi dokumen *Microsoft Office 2003* yang berekstensi doc dan xls dengan algoritma kriptografi *Blowfish*.
2. Kelebihan aplikasi pengamanan dokumen *Office* ini yaitu memiliki ukuran aplikasi yang kecil, karena memanfaatkan kamus fungsi dari *Java* dan *sun microsystem* untuk meminimalkan *codding*.
3. Kekurangan dari aplikasi pengamanan dokumen *Office* ini yaitu hanya bisa mengenkripsi dan mendekripsi dokumen *Microsoft Office 2003* saja yang berekstensi doc dan xls, masih terlalu besarnya ukuran dokumen yang terenkripsi.

5.2 Saran

Saran – saran yang berguna untuk pengembangan aplikasi ini adalah sebagai berikut :

1. Penyempurnaan untuk pendekripsian dokumen *Microsoft Office 2003* yang berekstensi ppt dan mdb. Aplikasi ini telah sukses mengenkripsi dan mendekripsi ke dua jenis dokumen tersebut, tetapi saat hasil dekripsi dibuka, *Microsoft Office* tidak dapat membacanya.
2. Penambahan fungsi enkripsi agar pada aplikasi ini mampu mengenkripsi dokumen yang berisi gambar atau audio.
3. Meminimalkan *size* dokumen yang telah terenkripsi agar tidak terlalu besar dengan *size* dokumen aslinya.
4. Untuk memaksimalkan aplikasi ini, pada penelitian berikutnya diharapkan tidak hanya mengenkripsi dokumen *Microsoft Office 2003* saja, tetapi dapat digunakan juga untuk *Microsoft Office 2007* .

6. Referensi

[1] Schneier. (1993). *Applied Cryptography :Protocols, Algorithms, and Source Code in C (Paperback).* USA: Wiley.

[2] Schneier. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition (Paperback).* USA: Wiley.

[3] Schneier. (1993). *Applied Cryptography :Protocols, Algorithms, and Source Code in C (Paperback).* USA: Wiley.