BAB III

ANALISIS DAN PERANCANGAN

Dalam bab ini berisi beberapa hal diantaranya seperti data yang digunakan, penerapan algoritma dan analisis perancangan sistem dalam mengimplementasikan algoritma *Serpent* untuk permasalahan keamanan data dalam *folder*.

III.1. Analisis Sistem

Hasil pengamatan pada sistem yang sedang berjalan, proses pengamanan data dalam *folder* terbagi menjadi dua bagian yaitu bagian enkripsi *folder* yaitu proses untuk melakukan enkripsi terhadap suatu *folder* yang didalamnya berisi data penting agar kerahasiaanya dapat terjamin dan tidak dapat dibaca oleh orang yang tidak berkepentingan. Bagian kedua yaitu proses dekripsi, dimana dalam proses dekripsi ini diperlukan suatu *folder* yang berisi datayang sudah terenkripsi dan kunci yang digunakan sewaktu melakukan enkripsi pada *folder* tersebut.

Masukan pada sistem yang berjalan, yaitu masukan berupa *e-document* atau data *file* teks yang berada didalam *folder*. Jenis data *file* yang dapat ditampung pada sistem yang berjalan berbeda-berbeda, tergantung aplikasi yang digunakan. Pengguna sistem akan memilih *folder* yang akan dienkripsi maupun didekripsi, kemudian akan memasukkan kunci yang telah ditentukan sebelumnya. Dari proses itu kemudian sistem melakukan tugasnya, yaitu memproses data *file*

teks masukan dan menghasilkan suatu data *file* teks sesuai target dan tujuan dari sistem itu sendiri.

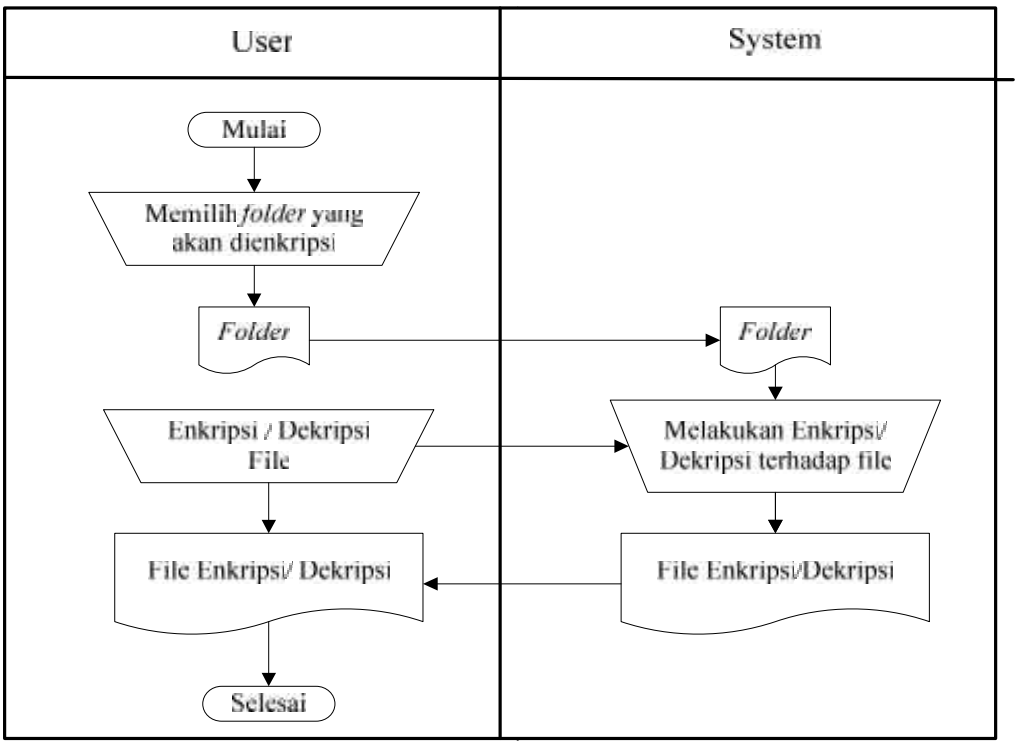
Pada sistem yang berjalan, bentuk keluaran atau hasil yang akan dihasilkan oleh sistem juga ada dua bagian. Bagian pertama adalah jika masukan pada sistem berupa *folder* yang sudah dienkripsi maka sistem akan menghasilkan data berektensi *.Encypted. Bagian kedua jika masukan pada sistem berupa *folder* yang masih terenkripsi, *folder* tersebut bisa dikembalikan kebentuk aslinya, maka sistem akan melakukan dekripsi pada *folder* tersebut.

III.1.1. Analisis *Input*

Analisis masukan atau bentuk inputan pada sistem yang sedang berjalan adalah berupa *folder* yang berisi data-data belum dienkripsi (data *file* teks asli) yang akan dimasukkan pada proses enkripsi yang sudah dikunci dan *file*yang sudah di enkripsi yang akan dimasukkan pada proses dekripsi. Ukuran dan jenis data *file* teks yang dapat ditampung didalam *folder* oleh sistem yang antara lain: *txt, *.doc, *.docx. dan lain sebagainya.

III.1.2. Analisis Proses

Analisis proses pada sistem yang sedang berjalan dapat digambarkan dalam sebuah *flow of document* berikut ini:



Gambar III.1. *Flow Of Document* Sistem Lama

Gambar *flow of document* sistem yang lama tersebut di atas dapat dijelaskan sebagai berikut :

1. *User* akan memilih *folder* yang akan dienkripsi atau didekripsi, kemudian mengirimkannya pada sistem.
2. Langkah selanjutnya adalah *user* akan melakukan enkripsi atau dekripsi dan mengirimkannya pada sistem.
3. Sistem akan melakukan pengenkripsian/dekripsi terhadap *folder*yang akan dikirimkan oleh *user*.
4. Hasil enkripsi maupun dekripsi yang dilakukan oleh sistem akan diterima oleh *user*.

III.1.3. Analisis *Output*

Analisa output atau bentuk keluaran pada sistem yang sedang berjalan adalah berupa *folder* berisi data penting yang sudah dienkripsi jika proses yang dilakukan adalah enkripsi dan *folder* yang asli jika proses yang dilakukan adalah dekripsi. Bentuk keluaran pada proses dekripsi akan sama dengan bentuk masukan pada waktu proses enkripsi, baik dari segi ukuran maupun ekstensi *folder* tersebut.

III.2. Evaluasi Sistem yang Berjalan

Evaluasi sistem yang berjalan berdasarkan apa yang telah penulis jabarkan tersebut di atas, adalah sebagai berikut:

1. Kerahasiaan data dalam *folder* pada media penyimpanan komputer tidak terjamin karena sistem pengamanan *folder* berisi data penting sangat minim.
2. Masih kurangnya pemanfaatan kriptografi dalam pengiriman pesan dalam *folder* berisi data penting dengan menggunakan algoritma *serpent* dalam mengamankan data dalam *folder* yang berbasis komputer.

III.3. Perancangan Sistem

Perancangan sistem yang diusulkan meliputi perancangan diagram dan perancangan antarmuka program. Perancangan diagram terdiri dari perancangan *use case diagram*, perancangan *class diagram*, dan perancangan *activity diagram*. Perancangan antarmuka program terdiri dari perancangan *form* utama, perancangan *form* enkripsi dan perancangan *form* dekripsi. Perancangan diagram yang diusulkan dimaksudkan untuk menjelaskan alur kegiatan atau proses yang

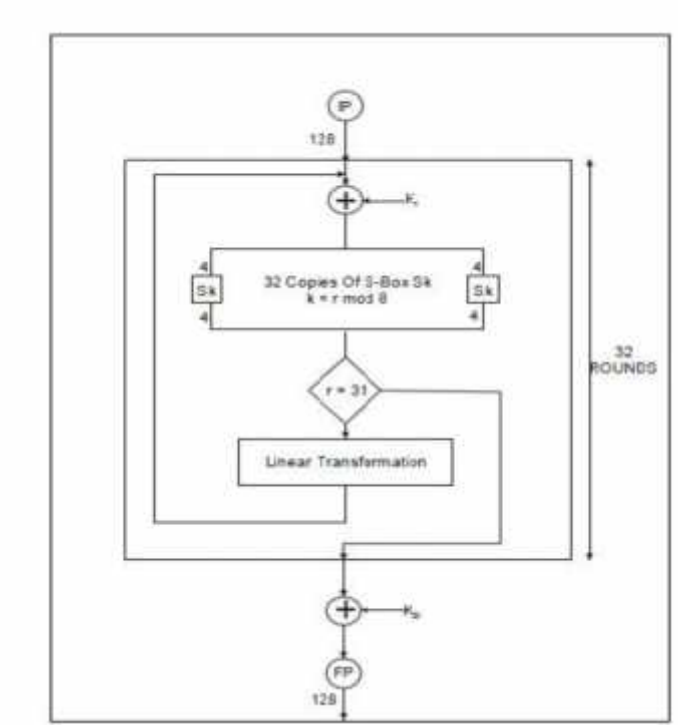
terjadi pada sistem yang diusulkan, sedangkan perancangan antarmuka program dimaksudkan untuk menampung masukan dari *user* dan sebagai tempat terjadinya interaksi antara sistem dengan *user*.

III.3.1. Perancangan Algoritma *Serpent*

III.3.1.1. *Block Chiper Serpent*

Algoritma *serpent* merupakan algoritma block chipper yang didesain oleh Ros Anderson, Eli Biham, dan Lars Knudsen. Algoritma ini merupakan *runner-up* dalam kompetisi Advanced Encryption Standart (AES) yang dimenangkan oleh algoritma *Rijandael*. Para perancangnya mengklaim meskipun *Rijandael* lebih cepat karena memiliki putaran lebih sedikit, *serpent* lebih aman. Meskipun lebih lambat daripada algoritma AES (Rijandael), algoritma ini masih lebih cepat daripada algoritma DES (Data Encryption Algorithm) yang telah banyak digunakan sebelumnya. Pada mesin Pentium 200MHz, algoritma *serpent* dapat berjalan dengan kecepatan 45 Mbit/s. Ini lebih cepat daripada DES yang memliki kecepatan 15 Mbit/s pada mesin yang sama.

Serpent merupakan operasi SP-*network* (*substitution permutation network*) 32 putaran pada 4 word berukuran 32 bit (blok berukuran 128 bit). Pada komputasi internal, semua nilai direpresentasikan *little-endian* dengan word pertama adalah least significant word dan word terakhir adalah most significant word dengan bit 0 merupakan least significant bit dari word pertama. Panjang kunci yang dapat digunakan berukuran 128, 192, atau 256 bit. Jika panjang kurang dari 256 bit, kunci tersebut ditambahkan satu bit ’1’ yang diikuti ‘0’ hingga panjangnya 256 bit.



Gambar III.2. Skema Algoritma *Serpent*

Serpent menggunakan S-Box dari DES yang telah banyak dipelajari selama bertahun-tahun dengan properti-properti yang dapat dipahami dengan baik, dengan struktur baru yang telah dioptimasi untuk implementasi yang lebih efisien pada *Processor* modern. Rancangan *serpent* tahan terhadap semua jenis serangan-serangan yang menggunakan teknik diferensial dan linear.

Ada beberapa varian *Cipher serpent*. Varian utama adalah *Cipher* 32 putaran yang diyakini seaman *Cipher* tiga kunci *triple* DES. *Cipher serpent* 32 putaran ini sedikit lebih lambat daripada DES. *Cipher* ini beroperasi pada empat *word* berukuran masing-masing 32 bit sehingga ukuran blok menjadi 128 bit.

Varian lainnya dibuat dengan meningkatkan ukuran blok. Ukuran blok dapat ditingkatkan menjadi dua kali lipat menjadi 256 bit baik dengan meningkatkan ukuran *word* dari 32 bit menjadi 64 bit, maupun dengan menggunakan fungsi putaran pada jaringan *Feistal*. Dua varian *Cipher serpent* ini dapat dikombinasikan untuk menghasilkan *Cipher* dengan panjang blok 512 bit.

Semua nilai yang digunakan pada *Cipher* direpresentasikan dalam *little-endian*, termasuk urutan 32 bit (0-31 dalam *word* berukuran 32 bit, atau 0-127 dalam blok 128 bit), dan urutan *word* dalam blok. Bit 0 adalah bit paling tidak berarti (*Least Significant Bit*), dan *word* 0 adalah *word* paling tidak berarti (*LeastSignificant Word*). Notasi penulisan sangat penting karena ada dia representasi *serpent* yang ekuivalen yaitu representasi standard dan representasi *bitslice.*

III.3.1.2. Deskripsi *Block Chiper Serpent*

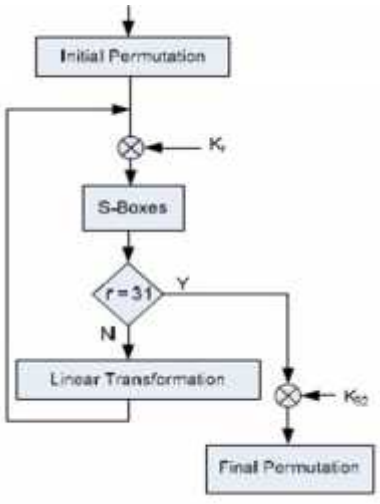
Varian utama dari *Cipher serpent* mengenkripsi 128 bit *plain*teks P menjadi 128 bit *Cipher* teks C dalam r putaran menggunakan r_{+1} kunci internal (K_0,K_1, …, K_r) yang masing-masing panjangnya 128 bit. Secara *default*, jumlah putaran r adalah sebanyak 32.

Cipher nya sendiri adalah sebuah jaringan SP yang terdiri atas:

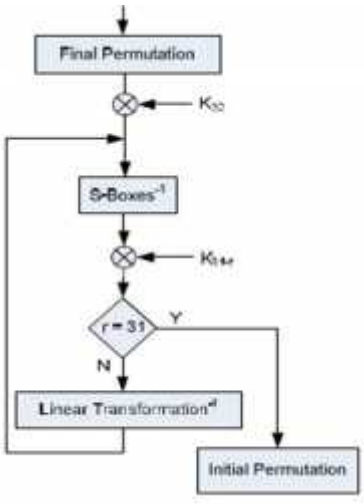
a. Sebuah *initial permutation* IP.
b. 32 putaran, dimana tiap putaran mengandung operasi *key-mixing*, dan operasi terhadap S-Box, dan (untuk semua putaran kecuali putaran terakhir) sebuah transformasi linear. Pada putaran terakhir, transformasi linear ini diganti dengan operasi *key-mixing* tambahan.

c. *Final Permutation* FP.

Initial permutation dan *final permutation* tidak memilik nilai. Kriptografik yang signifikan, melainkan hanya digunakan untuk mengoptimasi dan menyederhanakan implementasi *Cipher*, dan untuk meningkatkan efisiensi komputasional. Notasi yang digunakan adalah sebagai berikut. *Initial permutation* IP dioperasikan terhadap *plain*teks P menghasilkan B_0 yang merupakan input putaran pertama. Putaran diberi nomor 0 sampai 31, dimana putaran pertama adalah putaran 0, dan putaran terakhir adalah putaran 31. Output putaran pertama (putaran 0) adalah B_1, output putaran kedua (putaran 1) adalah B_2, output putaran I adalah B_{i+1} dan seterusnya, sampai output putaran terakhir (dimana transformasi linier digantikan dengan operasi *key-mixing* tambahan) adalah B_{32}. *Final permutation* FP dioperasikan terhadap B_{32} untuk mendapatkan *Cipher*teks C.



Gambar III.3. Proses Enkripsi



Gambar III.4 Proses Dekripsi

Algoritma Enkripsi Metode *Serpent* :

1. Membagi kunci masukan K menjadi delapan bagian, masing-masing 32 bit yang dinotasikan dengan w-8, … , w-1

2. Membentuk 132 kunci antara (prekey) yang dinotasikan dengan w0, … , w131 melalui persamaan:

wi = (wi-8 wi-5 wi-3 wi-1 Ø i)<<< 11

Notasi Ø merupakan bagian kecil dari golden ratio $(\sqrt{5} + 1) / 2$ atau 0x9e3779b9 dalam heksadesimal.

3. Membentuk 132 kunci putaran (round key) k0 sampai k131 yang dibentuk dari kunci antara yang dihasilkan dari proses sebelumnya dengan menggunakan S-boxes. S-boxes digunakan untuk mengubah kunci antara wi menjadi ki dengan ketentuan berikut ini :

{k0, k1, k2, k3} = S3 (w0, w1, w2, w3)

{k4, k5, k6, k7} = S2 (w4, w5, w6, w7)

…

{k8, k9, k10, k11} = S1 (w8, w8, w10, w11)

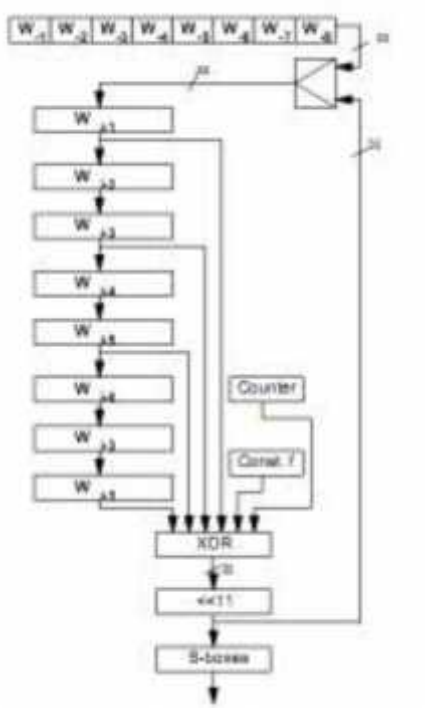
{k12, k13, k14, k15} = S0 (w12, w13, w14, w15)

. . .

{k124, k125, k126, k127} = S4 (w124, w125, w126, w127)

{k128, k129, k130, k131} = S3 (w128, w129, w130, w131)

Pembentukan kunci putaran untuk tahap (1) sampai tahap (3) dapat digambarkan dalam gambar 3.



Gambar III.5 Pembentukan kunci putaran

4. Membentuk upakunci 128 bit Ki (untuk I € {0, …, 32}) dari 32 bit nilai kj dengan cara:

Ki = {k4i, k4i+1, k4i+2, k4i+3}

5. Menerapkan IP pada upakunci yang dihasilkan untuk menempatkan bit-bitnya ke dalam urutan yang sesuai.

i = IP (Ki)

Algoritma Metode *Serpent* Dalam Pemrograman :

Plainteks : A (01000001)

Key : B (01000010)

Teta = (sqrt(5) + 1) / 2

For u = 1 to 32

For i = 1 to 132

Wi = Wi8 xor Wi5 xor Wi1 xor Teta xor i

Next i

For t = 1 to 132

k0, k1, k2, k3 = S3 (w0, w1, w2, w3)

k4, k5, k6, k7 = S2 (w4, w5, w6, w7)

…

k12, k13, k14, k15 = S0 (w12, w13, w14, w15)

. . .

k128, k129, k130, k131 = S3 (w128, w129, w130, w131)

Next t

Next u

III.3.1.3. *Key Schedule Serpent*

Sebagaimana dekripsi pada Cipher, kita dapat mendeskripsikan key schedule dalam mode standard atau mode *bitslice*. *Cipher* serpent memerlukan 132 buah *word* yang masing-masing berukuran 32 bit. Pertama, kita ekspansi kunci masukan user sepanjang 256 bit menjadi 33 buah sub kunci (K_0, ..., K_{32}) dengan panjang masing-masing 128 bit. Kita tulis K sebagai delapan *word* (w_{-8}, …, w_{-1}) yang masing-masing berukuran 32 bit lalu ekspansi menjadi *intermediate key* (disebut juga pre*key*) w_0,...,w_{131} dengan cara rekursif sebagai berikut :

$$w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \Phi \oplus i) <<< 11$$

Dimana Φ adalah bagian fraksional dari golden ratio (5 +1)/2 atau 0x9e3779b9 dalam heksadesimal. Polinomial $x^8+x^7+x^5+x^3+1$ adalah primitif

dimana bersama dengan penjumlaha index putaran dipilih untuk memastikan distribusi bit kunci merata pada setiap putaran, dan untuk menghilangkan kunci lemah dan keterhubungan antar kunci.

Round key sekarang dihitung dari pre*key* menggunakan S-box pada mode *bitslice*. Input dan output S-box diambil pada jarak 33 *word*s, untuk meminimalkan potensi serangan dengan teknik diferensial pada beberapa putaran akhir. Kita menggunakan S-box untuk mentransformasi pre*key* w_i menjadi *word* k_i yang merupakan *round key* dengan membagi vektor pre*key* menjadi empat bagian dan mentransformasi *word* ke_{-i} pada tiaptiap bagian menggunakan $S_{(r+3-i)}$mod r. Dapat dilihat pada contoh untuk r = 32 sebagai berikut :

$\{k_0, k_{33}, k_{66}, k_{99}\} = S_3(w_0, w_{33}, w_{66}, w_{99})$

$\{k_1, k_{34}, k_{67}, k_{100}\} = S_3(w_1, w_{34}, w_{67}, w_{100})$ …

$\{k_{31}, k_{64}, k_{97}, k_{130}\} = S_3(w3_1, w_{64}, w_{97}, w_{130})$

$\{k_{32}, k_{65}, k_{98}, k_{131}\} = S_3(w_{32}, w_{65}, w_{98}, w_{131})$

Lalu kita lakukan penomoran ulang terhadap 32 bit nilai k_j sebagai128 bit kunci internal K_i (untuk i $\in\{0,...,r\}$) sebagai berikut :

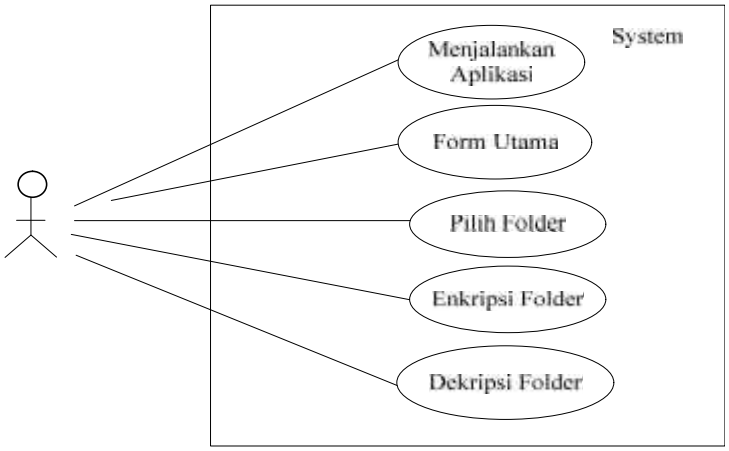
$K_i = \{k_{4i}, k_{4i}+1, k_{4i}+2, k4_{i+3}\}$

Kemudian operasikan IP ke *round key* untuk menempatkan bit kunci ke dalam kolom yang benar.

III.3.2. Perancangan *Use Case Diagram*

Perancangan *use case diagram* akan menjelaskan keterkaitan *antara use case*, baik yang mempunyai hubungan antara di dalam (*include*) lingkup *use case*

itu sendiri maupun di luar (*extend*) lingkup *use case*. *Use case diagram* pada sistem yang diusulkan terdiri dari tiga buah *use case* dan satu aktor. *Ketiga use case* tersebut adalah *use case* enkripsi yang mempunyai hubungan asosiasi dengan user. Rancangan *use case diagram* pada sisitem yang diusulkan dapat dilihat pada Gambar III.5.



Gambar III.6. *Use Case Diagram* Aplikasi Keamanan Data pada *Folder*

Gambar *use case diagram* tersebut di atas, dapat di jelaskan secara lebih detail dalam narasi *use case*, seperti berikut ini:

1. Narasi *Use Case* Enkripsi Teks/ *File* Teks

Tabel III.1. Narasi *Use Case* Enkripsi *Folder*

Use case name	Enkripsi	
Use case type	*Essential*	
Priority	*High*	
Actor	*End User*	
Description	*Use Case* ini digunakan oleh aktor untuk membuka dan menampilkan serta memulai aktifitas enkripsi data pada sistem.	
Basic Flow	**Aktor**	**Sistem**
	1.Masuk dari *form* utama Memilih *folder* dan	2.Memeriksa dan melaksanakan enkripsi terhadap data dalam

	memasukkan kunci yang kemudian mengklik tombol Enkripsi.	*folder* yang dikirimkan oleh aktor.
	3.Menerima Informasi dari sistem	4. Menampilkan hasil enkripsi
Past condotion	Aktor dapat memilih *folder* yang akan di enkripsi	
Extend	-	
Include	-	

2. Narasi *Use Case* Dekripsi *Folder*

Tabel III.2 . Narasi *Use Case* Dekripsi *Folder*

Use case name	Dekripsi	
Use case type	*Essential*	
Priority	*High*	
Actor	*End User*	
Description	*Use Case* ini digunakan oleh aktor untuk membuka dan menampilkan serta memulai aktifitas dekripsi data pada sistem.	
Basic Flow	**Aktor**	**Sistem**
	1.Masuk dari *form* utama Memilih *folder* yang telah dienkripsi dan memasukkan kunci yang kemudian mengklik tombol dekripsi.	2.Memeriksa dan melaksanakan dekripsi terhadap *folder* yang telah terenkripsi dikirimkan oleh aktor.
	3.Menerima Informasi dari sistem	4. Menampilkan hasil dekripsi
Past condotion	Aktor dapat memilih *folder* yang terenkripsiuntuk di dekripsi	
Extend	-	
Include	-	

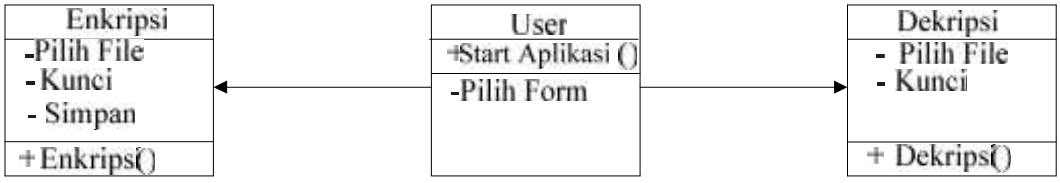
3. Narasi *Use Case* Masuk

Tabel III.3. Narasi *Use Case* Masuk

Use case name	Jalankan aplikasi	
Use case type	*Essential*	
Priority	*High*	
Actor	*End User*	
Description	*Use Case* ini digunakan oleh aktor untuk membuka dan menampilkan serta memulai aktifitas dan membuka sistem.	
Basic Flow	**Aktor**	**Sistem**
	1.Aktor menjalankan aplikasi	2.Membuka sistem aplikasi
Past condotion	Aktor mengklik tombol *Encrypt* ataupun *Decrypt*	
Extend	-	
Include	-	

III.3.3. Perancangan *Class Diagram*

Class diagram pada sistem yang di usulkan terdiri dari kelas *Form* Utama, kelas *Form* Enkripsi dan kelas *Form* Dekripsi. Kelas *Form* Enkripsi dan kelas *Form* Dekripsi mempunyai hubungan asosiasi dengan kelas *Form* Utama. Hubungan asosiasi antara kelas *Form* Utama dengan kelas *Form* Enkripsi dan kelas *Form* Utama dengan kelas *Form* Dekripsi merupakan hubungan satu ke satuan.



Gambar III.7 : *Class* Diagram Enkripsi dan Dekripsi *Folder*

Pada gambar *class* diagram diatas, menunjukkan bahwa hal pertama yang harus dilakukan *user* adalah memulai menjalankan aplikasi, kemudian memilih *form* aplikasi yang tersedia. *User* dapat menjalankan *form* aplikasi enkripsi maupun dekripsi. Dalam *form* aplikasi enkripsi, user memilih yang hendak di enkripsi dan memasukkan kunci enkripsi. *Folder* yang selesai dieksekusi dapat disimpan di directory. Kemudian, *user* mampu melakukan dekripsi *folder* yang sudah di enkripsi. Pilih *folder* yang sudah di enkripsi dan memasukkan kode dekripsi, dimana kunci yang sama dengan kunci enkripsi. Kemudian jalankan program dekripsi.

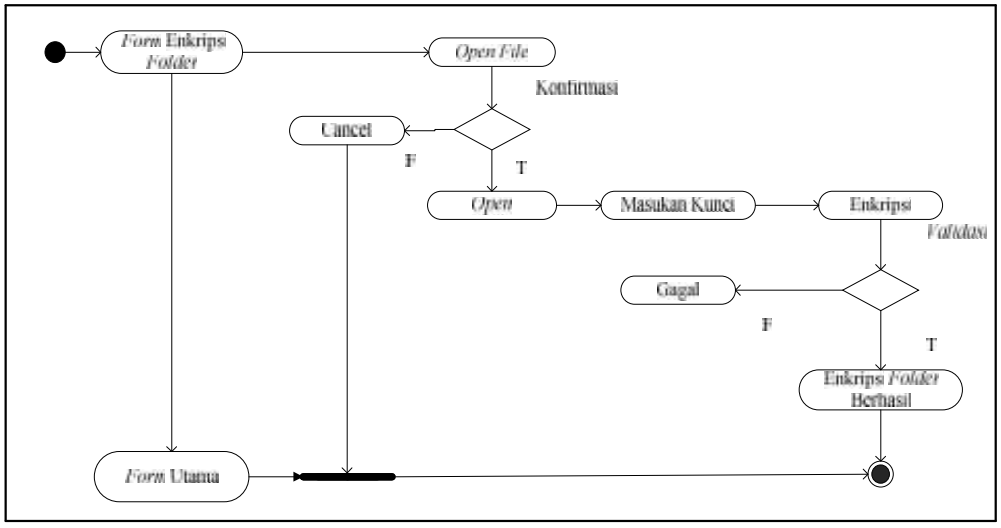
III.3.4. Perancangan *Activity Diagram*

Perancangan *activity* diagram atau diagram aktivitas ini bertujuan untuk menjelaskan kegiatan yang dapat dilakukan aktor (*user*) pada sebuah *use case*. *Activity* diagram pada sistem yang disusulkan terdiri dari tiga buah *activity* yang masing-masing mewakili *activity* dari *use case*, yaitu *activity* diagram enkripsi, *activity* diagram dekripsi dan *activity* diagram masuk.

III.3.4.1. *Activity Diagram Encrypt*

Activity diagram encrypt ini akan menjelaskan alur kegiatan yang dapat dilakukan oleh aktor dalam kegiataanya melakukan enkripsi suatu *folder*. Pertama kali yang harus dilakukan oleh aktor adalah menjalankan aplikasi, kemudian actor masuk pada *form* utama, kemudian pilih masuk, kemudian memilih *folder*yang akan dienkripsi. Jika pada opsi *encrypt* dan *decrypt*, aktor memilh *encrypt* maka langkah selanjutnya yang harus dilakukan adalah memasukkan kunci. Setelah

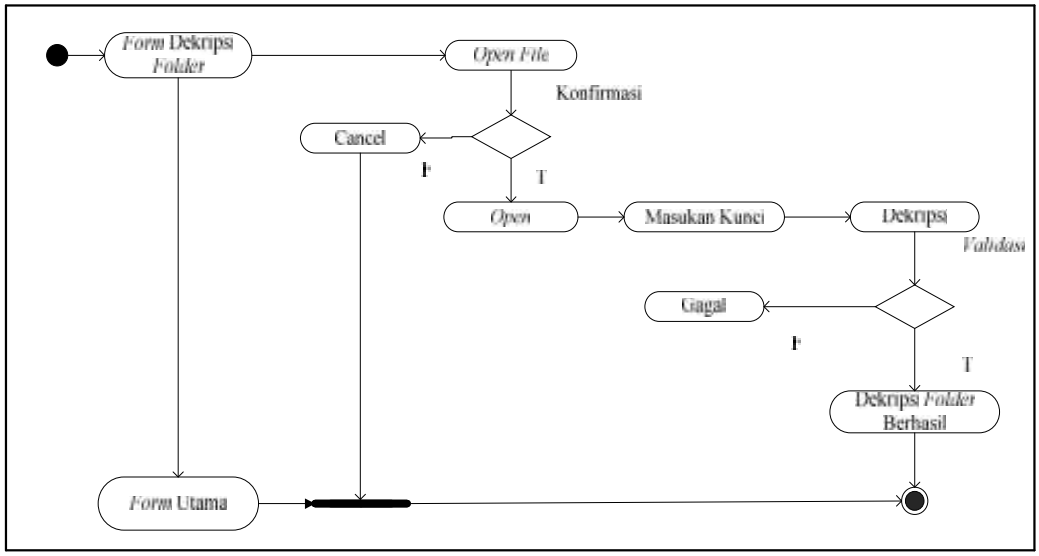
memasukkan kunci, langkah yang harus dilakukan oleh aktor adalah mengklik tombol *encrypt* untuk melakukan enkripsi terhadap *folder* yang diunggah.



Gambar III.8. ***Activity Diagram Encrypt Folder***

III.3.4.2. *Activity Diagram Decrypt*

Adapun rancangan *activity diagram decrypt* pada sistem yang diusulkan dapat dilihat pada Gambar III.7.



Gambar III.9. ***Activity Diagram Decrypt Folder***

Activity diagram decrypt tersebut di atas akan menjelaskan alur kegiatan yang dapat dilakukan oleh aktor dalam kegiataanya melakukan enkripsi *folder*. Pertama kali harus dilakukan oleh aktor adalah memilih *folder* yang akan di dekripsi. Jika pada opsi *encrypt* dan *decrypt*, aktor memilih *decrypt* maka langkah selanjutnya yang harus dilakukan adalah memasukkan kunci. Setelah memasukkan kunci, langkah yang harus dilakukan oleh aktor adalah mengklik tombol *decrypt* untuk melakukan dekripsi terhadap *folder* yang diunggah.

III.3.5. Perancangan *Form* Aplikasi

Dalam perancangan *form* aplikasi keamanan data dalam *folder* harus melakukan penyesuaian terhadap perancangan sistem yang dibuat sebelumnya. *Form* aplikasi yang di rancang menggunakan algoritma kriptografi dengan metode *Serpent Cipher* sehingga dapat melakukan enkripsi maupun dekripsi terhadap suatu *folder*.

III.3.5.1. Perancangan *User Interface Form* Utama

Perancangan *interface* untuk halaman utama perancangan aplikasi keamanan data pada *folder* sebagai berikut:

Gambar III.10. Rancangan *Form* Utama

III.3.5.2. Perancangan *User Interface Form* Enkripsi

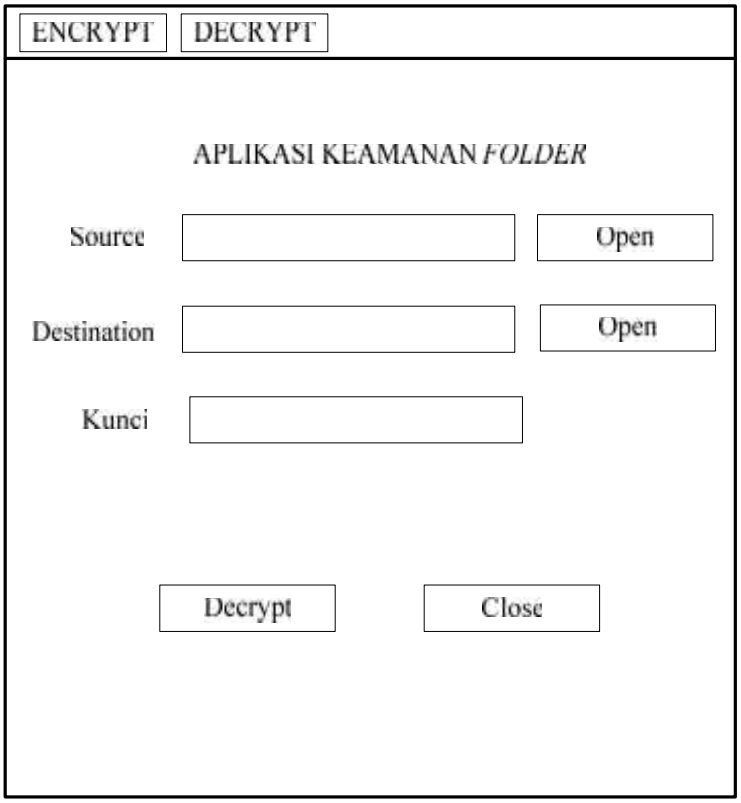
Form enkripsi *folder* merupakan *form* untuk melakukan proses enkripsi *folder*. Proses enkripsi menggunakan algoritma *serpent* dan kemudian disimpan ke dalam *directori* kita pilih. Berikut tampilan *form* enkripsi *folder*:

Gambar III.10. Rancangan *Form* Enkripsi

Pertama, pilih *folder* yang mau dienkripsi pada direktori dengan tekan tombol open, dan simpan hasil enkripsi pada direktori dengan tekan tombol destination, kemudian masukkan kunci, kemudian tekan tombol kunci *encrypt*

III.3.5.3. Perancangan *User Interface Form* Dekripsi

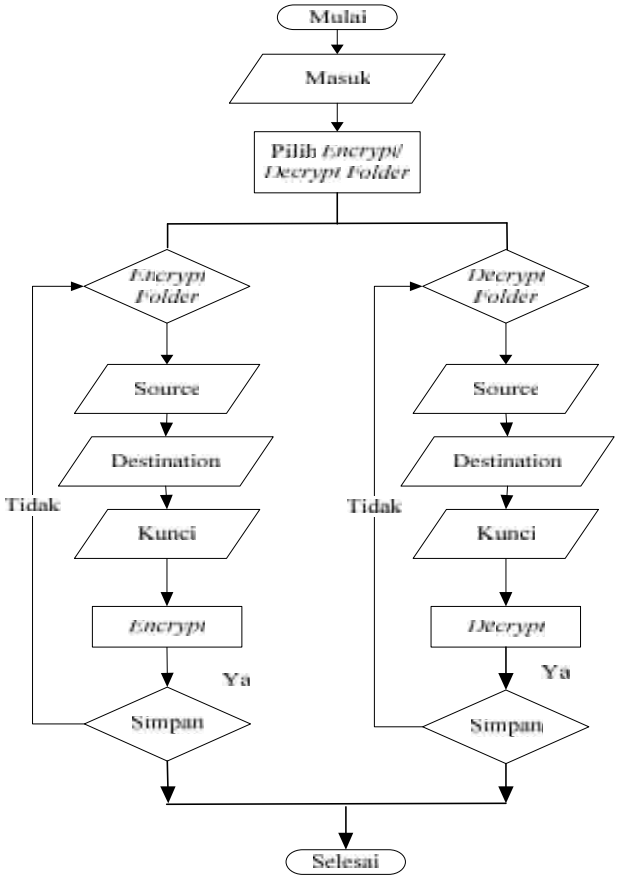
Form dekripsi *folder* merupakan *form* untuk melakukan proses dekripsi*folder*. Proses dekripsi menggunakan algoritma *serpent* dan kemudian disimpan ke dalam *directori* kita pilih. Berikut tampilan *form* dekripsi*folder*:



Gambar III.12. Rancangan *Form* Dekripsi

Lakukan hal yang sama dengan enkripsi yaitu memilih folder yang terenkripsi pada direktori, kemudian lakukan proses dekripsi menggunakan algoritma *serpent* dengan cara klik tombol dekripsi. Klik tombol *close* untuk keluar dari aplikasi.

III.3.6.*Flowchart* Aplikasi Enkripsi Dan Dekripsi Algoritma *Serpent*



Gambar III.13. *Flowchart* Aplikasi Enkripsi Dan Dekripsi *Serpent*

Program ini dimulai dengan masuk ke *form*utama, kemudian memilih tombol masuk pada form utama,kemudian pilihan apakah *user* akan menggunakan Enkripsi maupun Deskripsi, kemudian dalam enkripsi atau deskripsi, *user* akan memasukkan sumber *folder* yang akan diproses, lalu memasukkan kunci yang telah kita tentukan dan kemudian memasukkan tujuan *folder* yang sudah di enkripsi atau yang sudah di deskripsi. Dan jika ketiga hal itu telah terpenuhi dan

sudah dimasukkan, maka selanjutnya klik tombol *encrypt* untuk enkripsi dan tombol *decrypt* untuk dekripsi untuk melakukan proses. Setelah proses telah terjadi maka selanjutnya adalah keluar, setelah itu selesai.