BAB III

ANALISIS DAN DESAIN SISTEM

III.1. Analisis Masalah

Analisis masalah bertujuan untuk mengidentifikasi permasalahan-permasalahan yang ada pada sistem dimana aplikasi dibangun, meliputi perangkat keras (*hardware*), perangkat lunak (*software*) dan pengguna (*user*). Analisis ini diperlukan sebagai dasar bagi tahapan perancangan sistem.

Keamanan informasi adalah suatu keharusan yang perlu diperhatikan apalagi jika informasi itu bersifat rahasia. Pada saat ini segala macam informasi disimpan kedalam database dikarenakan untuk memudahkan *user* mengakses informasi tersebut. Akan tetapi apabila *content* dari *database* tidak diamankan, tentu ada kemungkinan informasi tersebut jatuh ketangan pihak-pihak yang tidak berhak tanpa ada kesulitan sekalipun untuk membaca informasi tersebut.

Untuk mengatasi hal ini, penulis akan mencoba menerapkan salah satu Algoritma kriptografi yaitu Algoritma *Knapsack* sebagai sistem pengamannya. Hal ini diharapkan mampu untuk melindungi seluruh informasi yang ada di dalam *database* dengan cara mengenkripsi informasi yang sebelumnya berupa *plaintext* menjadi *ciphertext* sebelum disimpan ke dalam *database*.

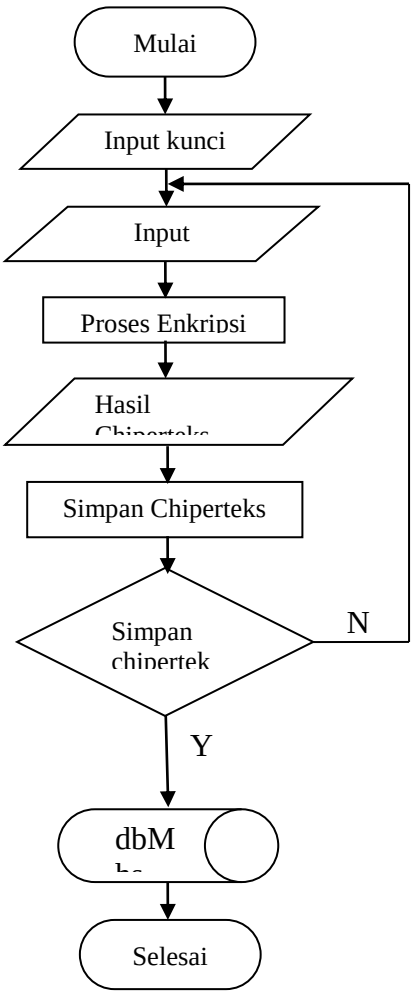
III.2. Penerapan Metode

Penerapan Metode/Algoritma akan membahas tentang dimana akan diterapkan metode yang digunakan di dalam penelitian penulis, dalam hal ini penulis akan menerapkan Algoritma *Knapsack* dalam pengolahan data yang nantinya akan dienkripsi maupun didekripsi dengan menggunakan Algoritma tersebut.

Pada aplikasi yang akan dirancang penulis, Algoritma ini akan diterapkan untuk mengenkripsi *field* nama, nim, matakuliah sebelum disimpan kedalam database. Selanjutnya hasil dari enkripsi akan disimpan ke dalam database agar dapat diakses oleh *user*. Hal ini di harapkan mampu untuk melindungi seluruh informasi yang ada di dalam *database*.

III.2.1 Penerapan Enkripsi dan Dekripsi Algoritma Knapsack

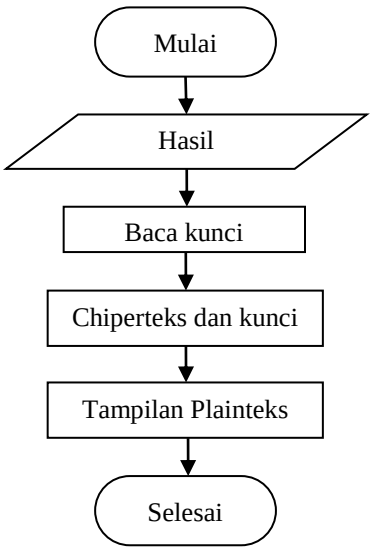
Untuk membuat sebuah aplikasi yang dapat mengamankan data. Tentu perlu adanya rancangan dalam tahapan-tahapan proses. Tahapan-tahapan tersebut akan direpresentasikan dalam *flowchart*. Pada *flowchart* menjelaskan proses pada tahapan enkripsi Algoritma Knapsack. Algoritma ini akan diterapkan untuk mengenkripsi *database* dan selanjutnya hasil dari enkripsi akan disimpan ke dalam *database*.



Gambar III.1. *Flowchart* Enkripsi Algoritma Knapsack

Penjelasan :

1. Mulai
2. Masukan Private Key, hasil generate key yaitu Public Key
3. Inputkan Plainteks yang ingin dienkripsikan
4. Progam akan melakukan operasi knapsack dengan kunci public dan menghasilkan chiperteks
5. Kemudian hasil chiperteks di simpan ke dalam *database*
6. Selesai



Gambar III.2. *Flowchart* Dekripsi Algoritma Knapsack

Penjelasan :

1. Mulai
2. Mengambil chiperteks dari *database*
3. Program akan membaca kunci yang dihasilkan key private
4. Progam akan melakukan operasi knapsack antara chiperteks dengan kunci dan menghasilkan cplainteks
5. Progaram akan menampilkan hasil dekripsi berbentuk plainteks
6. Selesai

III.2.2 Studi Kasus Penelitian

Key private : 10, 23,55,100

Key public : 50 19 35 20

Enkripsi = Plainteks : B I M A

Ascii : 66 73 77 65

Key public : 5 0 1 9

Pembulatan bilangan : 16 16 16 16

Chiperteks =(66+5+16) = 87, (73+0+16) =89, (77+1+16) = 94 (65+9+19) = 90

Dekripsi : Chiperteks : 87 , 89, 94, 90

Key private : 1 , 0, 2, 3

Public-Privatte : 4 , 0, -1, 6

Pembulatan bilangan : 16 16 16 16

Mod : 120

Plainteks : (87-1-4-16 mod 120) = 66, (89-0-0-16 mod 120) = 73 ,(94-2-(-1) -16 mod 120) = 77, (90-3-6-16 mod 120) = 65

Hasil plainteks : 66,73,77,65

III.3. Analisis Kebutuhan Perancangan

III.3.1. Analisis Kebutuhan

Untuk mencapai penyelesaian dalam merancang aplikasi ini adapun kebutuhan pokok yang diperlukan adalah :

1. Pengumpulan berbagai data yang akan dijadikan input data dalam pengolahan dengan kriptografi.
2. Pengumpulan data – data mengenai pengamanan database dengan teknik enkripsi.

III.3.2. Spesifikasi dan Desain

1. Perangkat Keras (*Hardware*)

   Perangkat keras yang dapat digunakan untuk penelitian ini, yaitu :

   a) *Processor CoreDuo 1.80 GHz*

   b) *Hard disk :* 320 *GB*

c) *RAM* 2 *GB*

2. Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan untuk penelitian ini, yaitu :

a) Sistem Operasi *Windows 7.*

b) *Java* untuk bahasa pemrogramannya.

c) *MySql*

III.4. Desain Sistem

Sebagai solusi dari permasalahan yang telah teridentifikasi dan untuk membuat sebuah sistem yang dapat berjalan dengan baik serta sesuai harapan yang diinginkan maka tentunya terlebih dahulu haruslah membuat tahapan perencanaan sistem berupa *use case diagram, activity diagram* dan *flowchart.*

III.4.1. Algoritma Knapsack

Algoritma Knapsack juga adalah algoritma kriptografi kunci-publik. Keamanan algoritma ini terletak pada sulitnya memecahkan persoalan knapsack (*Knapsack Problem.*). Knapsack artiya karung/kantung. Karung mempunyai kapasitas muat terbatas. Barang-barang dimasukkan ke dalam karung hanya sampai batas kapasitas karung saja.

Knapsack problem:

Diberikan bobot knapsack adalah *M*. diketahui *n* buah objek yang masing-masing bobotnya adalah $w_1, w_2, \ldots, w_n$. Tentukan nilai *bi* sedemikian sehigga

$M = b_1w_1 + b_2w_2 + \ldots + b_nw_n$

Yang dalam hal ini, bi bernilai 0 atau 1. Jika $b_i = 1$, berarti objek *i* dimasukkan ke dalam knapsack, sebaliknya jika $bi = 0$, objek *i* tidak dimasukkan.

1. Algoritma knapsack sederhana

Ide dasar dari algoritma kriptografi knapsack adalah mengkodekan pesan sebagai rangkaian solusi dari persoalan knapsack. Setiap bobot w_i didalam persoalan knapsack merupakan kunci privat, sedangkan bit-bit plainteks menyatakan b_i.

Misalkan $n = 6$, $w_1 = 1$, $w_2 = 5$, $w_3 = 6$, $w_4 = 11$, $w_5 = 14$, dan $w_6 = 20$.

Plainteks: 111001010110000000011000

Plainteks dibagi menjadi blok yang panjangnya n, kemudian setiap bit didalam blok dikalikan dengan w_i yang berkoresponden sesuai dengan persamaan:

Blok plainteks ke-1 : 111001

Knapsack : 1, 5, 6, 11, 14, 20

Kriptogram : (1 x 1) + (1 x 5) + (1 x 6) + (1 x 20) = 32

Blok plainteks ke-2 : 010110

Knapsack : 1, 5, 6, 11, 14, 20

Kriptogram : (1 x 5) + (1 x 11) + (1 x 14) = 30

Blok plainteks ke-3 : 000000

Knapsack : 1, 5, 6, 11, 14, 20

Kriptogram : 0

Blok plainteks ke-4 : 011000

Knapsack : 1, 5, 6, 11, 14, 20

Kriptogram : (1 x 5) + (1 x 6) = 11

Jadi, cipherteks yang dihasilkan: 32 30 0 11

Algoritma knapsack sederhana diatas hanya dapat digunakan untuk enkripsi, tetapi tidak dirancang untuk dekripsi. Misalnya, jika diberikan kriptogram = 32, maka tentukan $b_1, b_2, \ldots, b_6$ sedemikian sehingga

$$32 = b_1 + 5b_2 + 6b_3 + 11b_4 + 14b_5 + 20b_6$$

Solusi persamaan ini tidak dapat dipecahkan dalam orde waktu polynomial dengan semakin besarnya *n* (dengan catatan barisan bobot tidak dalam urutan menaik). Namun, hal inilah yang dijadikan sebagai kekuatan algoritma knapsack.

2. *Superincreasing* knapsack

Superincreasing knapsack adalah persoalan knapsack yang dapat dipecahkan dalam orde *O(n)(*jadi,polinomial). Ini adalah persoalan knapsack yang mudah sehingga tidak disukai untuk dijadikan sebagai algoritma kriptografi yang kuat.

Jika senarai bobot disebut barisan superincreasing, maka kita dapat membentuk superincreasing knapsack. Barisan superincreasing adalah suatu barisan dimana setiap nilai didalam barisan lebih besar daripada jumlah semua nilai sebelumnya.

Misalnya {1,3,6,13,27,52} adalah barisan *superincreasing*, tetapi {1,3,4,9,15,25} bukan.

Solusi dari *superincreasing* (yaitu $b_1, b_2, \ldots, b_n$) mudah dicari sebagai berikut (berarti sama dengan mendekripsikan cipherteks menjadi plainteks semula):

1. Jumlahkan semua bobot didalam barisan.
2. Bandingkan bobot total dengan bobot terbesar didalam barisan. Jika bobot terbesar lebih kecil atau sama dengan bobot total, maka ia dimasukkan kedalam knapsack, jika tidak, maka ia tidak dimasukkan.

3. Kurangi bobot total dengan bobot yang telah dimasukkan, kemudian bandingkan bobot total sekarang dengan bobot terbesar selanjutnya. Demikian seterusnya sampai seluruh bobot didalam barisan selesai dibandingkan.
4. Jika bobot total menjadi nol, maka terdapat solusi persoalan superincreasing knapsack, tetapi jika tidak nol, maka tidak ada solusinya.

Contoh. Misalkan bobot-bobot yang membentuk barisan *superincreasing* adalah {2,3,6,13,27,52}, dan diketahui bobot knapsack M=70. Kita akan mencari b_1, b_2, ...,b_6 sedemikian sehingga

$$70=2b_1 + 3b_2 + 6b_3 + 13b_4 + 27b_5 + 52b_6$$

Caranya sebagai berikut:

a. Bandingkan 70 dengan bobot terbesar, yaitu 52. Karena $52 \leq 70$, maka 52 dimasukkan ke dalam knapsack.
b. Bobot total sekarang menjadi 70 – 52 = 18 dengan bobot terbesar kedua, yaitu 27. Karena 27 > 18, maka 27 tidak dimasukkan ke dalam knapsack.
c. Bandingka 18 dengan bobot terbesar berikutnya, yaitu 13. Karena $13 \leq 18$, maka 13 dimasukkan ke dalam knapsack.
d. Bobot total sekarang menjadi 18 – 13 = 5.
e. Bandingkan 5 dengan bobot terbesar kedua, yaitu 6. Karena 6 > 5, maka 6 tidak dimasukkan ke dalam knapsack.
f. Bandingkan 5 dengan bobot terbesar berikutnya, yaitu 3. Karena $3 \leq 5$, maka 3 dimasukkan ke dalam knapsack.
g. Bobot total sekarang 5 – 3 = 2.

h. Bandingkan 2 dengan bobot terbesar berikutnya, yaitu 2. Karena $2 \leq 2$, maka 2 dimasukkan kedalam knapsack.

i. Bobot total sekarang menjadi $2 - 2 = 0$.

Karena bobot total tersisa = 0, maka solusi persoalan *superincreasing* knapsack ditemukan.

Barisan bobot yang dimasukkan ke dalam knapsack adalah

{2, 3, - , 13, - , 52}

sehingga

70 = (1 x 2) + (1 x 3) + (0 x 6) + (1 x 13) + (0 x 27) + (1 x 52)

Dengan kata lain, plainteks dari kriptogram 70 adalah 110101

3. Algoritma knapsack Kunci-Publik

Algoritma *superincreasing* knapsack adalah algoritma yang lemah, karena *cipherteks* dapat didekripsi menjadi *plainteks*nya secara mudah dalam waktu lanjar ($O(n)$). Algoritma *non-superincreasing* knapsack atau normal knapsack adalah kelompok algoritma knapsack yang sulit (dari segi komputasi) karena membutuhkan waktu lama orde eksponensial untuk memecahkannya. Namun, *superincreasing* knapsack dapat dimodifikasi menjadi *non-superincreasing* knapsack dengan menggunakan kunci publik (untuk enkripsi) dan kunci privat (untuk dekripsi). Kunci publik merupakan barisan *non-superincreasing* sedangkan kunci privat tetap merupakan barisan *superincreasing*. Modifikasi ini ditemukan oleh Martin Hellman dan Ralph Merkle.

Cara membuat kunci publik dan kunci privat:

1. Tentukan barisan *superincreasing*.

2. Kalikan setiap elemen di dalam barisan tersebut dengan n modulo m. Modulus m seharusnya angka yang lebih besar daripada jumlah semua elemen di dalam barisan, sedangkan pengali n seharusnya tidak mempunyai faktor persekutuan dengan m.
3. Hasil perkalian akan menjadi kunci publik sedangkan barisan *superincreasing* semula menjadi kunci privat.

Contoh: Misalkan barisan *superincreasing* adalah {2, 3, 6, 13, 27, 52}, *m*=105, dan *n*=31. Barisan *non-superincreasing* dihitung sebagai berikut:

2 x 31 mod 105 = 62

3 x 31 mod 105 = 93

6 x 31 mod 105 = 81

13 x 31 mod 105 = 88

27 x 31 mod 105 = 102

52 x 31 mod 105 = 37

Jadi, kunci publik adalah {62, 93, 81, 88, 102, 37}, sedangkan kunci privat adalah {2, 3, 6, 13, 27, 52}.

1. Enkripsi

Enkripsi dilakukan dengan cara yang sama seperti algoritma knapsack sebelumnya. Mula-mula *plainteks* dipecah menjadi blok bit yang panjangnya sama dengan kardinalitas barisan kunci publik. Kalikan setiap bit di dalam blok dengan elemen yang berkoresponden di dalam kunci publik.

Contoh: Misalkan plainteks 011000110101101110 dan kunci publik {62, 93, 81, 88, 102, 37}. Plainteks dibagi menjadi blok yang panjangnya 6, kemudian setiap bit di dalam blok dikalikan dengan elemen yang berkoresponden di dalam kunci publik:

Blok plainteks ke-1 : 011000

Kunci publik : 62, 93, 81, 88, 102, 37

Kriptogram : (1 x 93) + (1 x 81) = 174

Blok plainteks ke-2 : 110101

Kunci publik : 62, 93, 81, 8 8, 102, 37

Kriptogram : (1 x 62) + (1 x 93) + (1 x 88) + (1 x 37) = 280

Blok plainteks ke-3 : 101110

Kunci publik : 62, 93, 81, 88, 102, 37

Kriptogram : (1 x 62) + (1 x 81) + (1 x 88) + (1 x 102) = 333

Jadi, cipherteks yang dihasilkan: 174, 280, 333.

2. Dekripsi

Dekripsi dilakukan dengan menggunakan kunci privat. Mula-mula penerima pesan menghitung n^{-1}, yaitu balikan *n* modulo *m*, sedemikian sehingga $n \times n^{-1} = 1 \pmod m$.

kekongruenan ini dapat dihitung dengan cara yang sederhana sebagai berikut:

$n \cdot n^{-1} = 1 \pmod m$
$\Leftrightarrow n \cdot n^{-1} = 1 + km$
$\Leftrightarrow n^{-1} = (1 + km)/n$, *k* sembarang bilangan bulat

Kalikan setiap kriptogram dengan $n^{-1} \bmod m$, lalu nyatakan hasil kalinya sebagai penjumlahan elemen-elemen kunci privat untuk memperoleh plainteks dengan menggunakan algoritma pencarian solusi *superincreasing* knapsack.

Contoh: kita akan mendekripsikan *cipherteks* dari contoh sebelumnya dengan menggunakan kunci privat {2, 3, 6, 13, 27 , 52}. Disini, n = 31 dan m = 105. Nilai n^{-1} diperoleh sebagai berikut:

$$n^{-1} = (1 + 105k)/31$$

dengan mencoba k = 0, 1, 2, …,maka untuk = 18 diperoleh n^{-1} bilangan bulat, yaitu

$$n^{-1} = (1 + 105 \times 18)/31 = 61$$

cipherteks dari contoh adalah 174, 280, 222. *Plainteks* yang berkoresponden diperoleh kembali sebagai berikut:

174 × 61 mod 105 = 9 = 3 + 6, berkoresponden dengan 011000

280 × 61 mod 105 = 70 = 2 + 3 + 13 + 52, berkoresponden dengan 110101

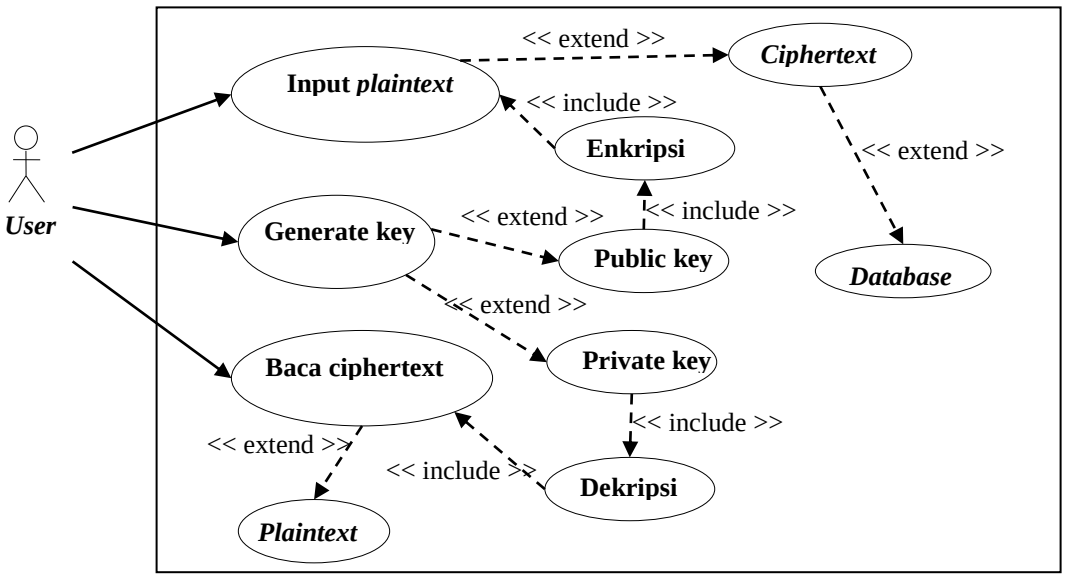
333 ×61 mod 105 = 48 = 2 + 6 + 13 + 27, berkoresponden dengan 101110

Jadi plainteks yang dihasilkan kembali adalah: 011000110101101110.

(Rinaldi Munir, 2004)

III.4.2. *Use Case Diagram*

Use case diagram dari aplikasi *database Knapsack* yang akan dirancang oleh penulis adalah seperti gambar III.3 sebagai berikut :

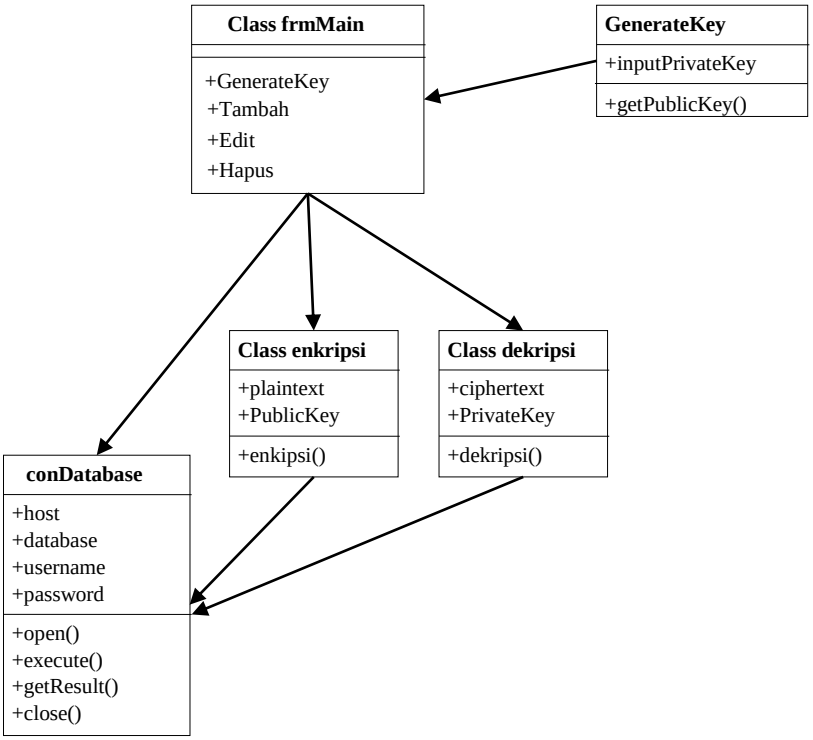


Gambar III.3. *Use Case Diagram* Aplikasi *database Knapsack*

Pada gambar di atas, *user* berfungsi sebagai *actor*. *User* dapat menginput *plaintext* kemudian *plaintext* akan di enkripsi terlebih dahlu menggunakan algoritma *knapsack* lalu disimpan kedalam *database*. *User* juga dapat membaca *ciphertext* dari *database* kemudian *ciphertext* harus terlebih dahulu di dekripsi agar menghasilkan *plaintext* kembali sehingga dapat di baca oleh *user*.

III.4.3 *Class Diagram*

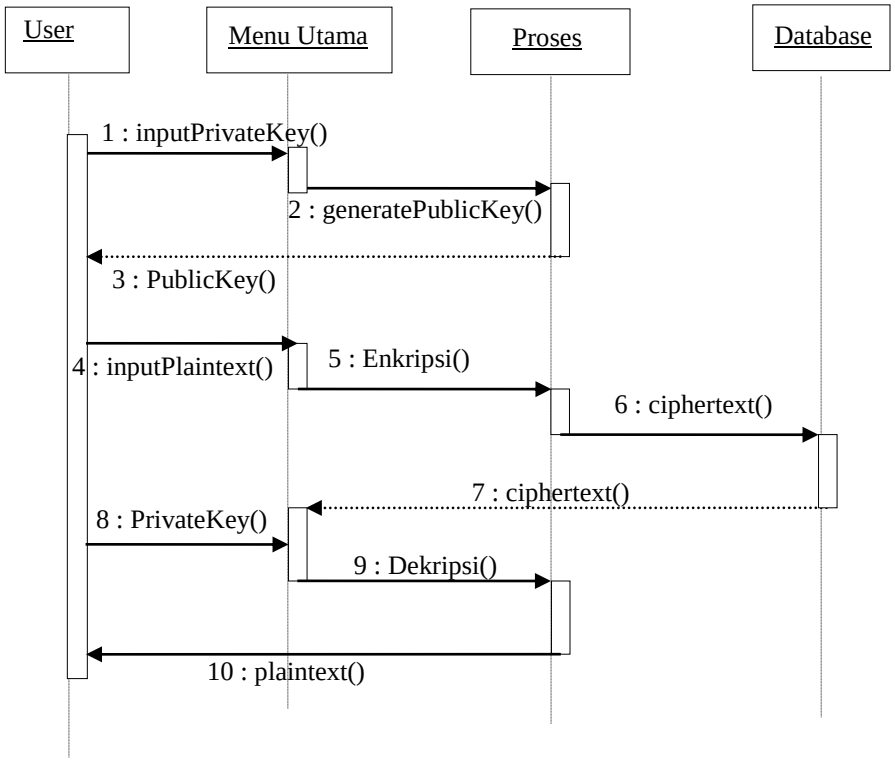
Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut. Adapun c*lass diagram* dari aplikasi *database knapsack* yang akan dirancang oleh penulis adalah sebagai berikut:



Gambar III.4. *Class Diagram* Aplikasi *Database Knapsack*

III.4.4. *Sequence Diagram*

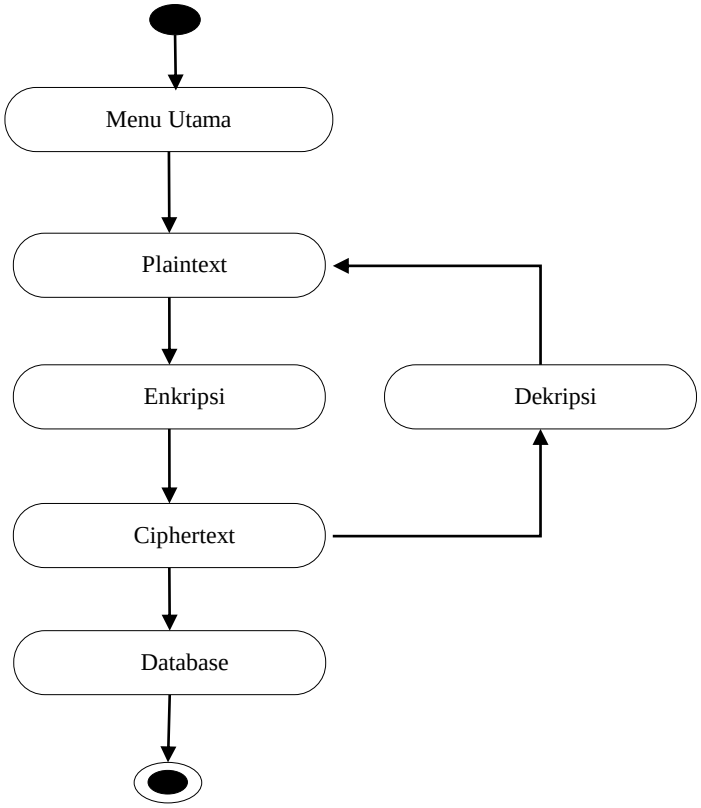
Sequence Diagram dari aplikasi *database knapsack* yang akan dirancang oleh penulis yang merupakan gambaran proses dari suatu sistem adalah sebagai berikut :



Gambar III.5. *Sequence Diagram* Aplikasi *Database Knapsack*

III.4.5. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity Diagram* dari aplikasi *database Knapsack* yang akan dirancang oleh penulis adalah seperti gambar III.6 sebagai berikut :



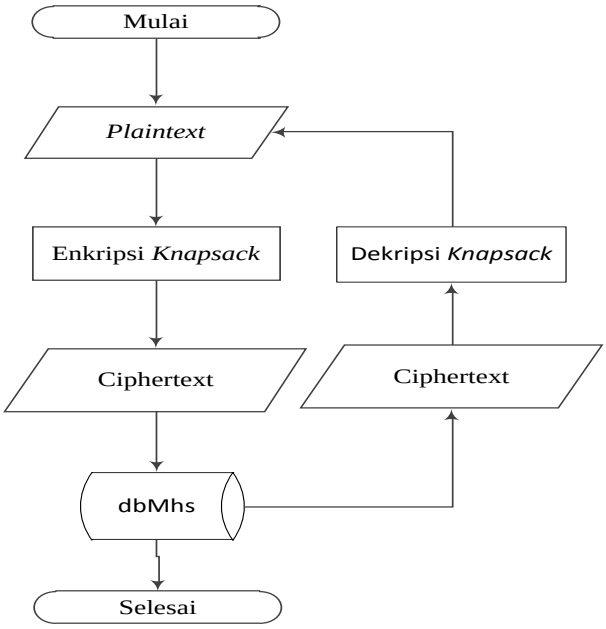
Gambar III.6. *Activity Diagram* Aplikasi *database Knapsack*

Pada gambar activity diagram di atas memiliki struktur sebagai berikut:

1. *User* menginputkan *plaintext* kemudian dienkripsi lalu akan menghasilkan *ciphertext* kemudian disimpan ke dalam *database*.
2. *Ciphertext* harus didekripsi terlebih dahulu untuk menghasilkan *plaintext*.

III.4.6. *Flowchart*

Berikut ini flowchart dari aplikasi *database Knapsack* yang akan dirancang oleh penulis adalah sebagai berikut :



Gambar III.7. *Flowchart* Aplikasi *Database Knapsack*

Dari gambar III.7. *Flowchart* aplikasi *database knapsack* dapat diperoleh keterangan sebagai berikut :

1. Memulai *input plaintext*.
2. Enkripsikan *plaintext* menggunakan algoritma *Knapsack*.
3. Setelah dienkripsi kemudian akan menghasilkan *ciphertext.*
4. Kemudian *ciphertext* disimpan ke dalam *database* dbMhs yang kemudian dapat digunakan untuk proses dekripsi.
5. Selesai.

III.5. Desain *Database*

Dalam perancangan aplikasi ini, penulis menggunakan *database* sebagai tempat penyimpanan seluruh data. Adapun struktur *database* yang didesain penulis dalam perancangan aplikasi ini adalah sebagai berikut.

Tabel III.1. Struktur *Database* Aplikasi *database Knapsack*

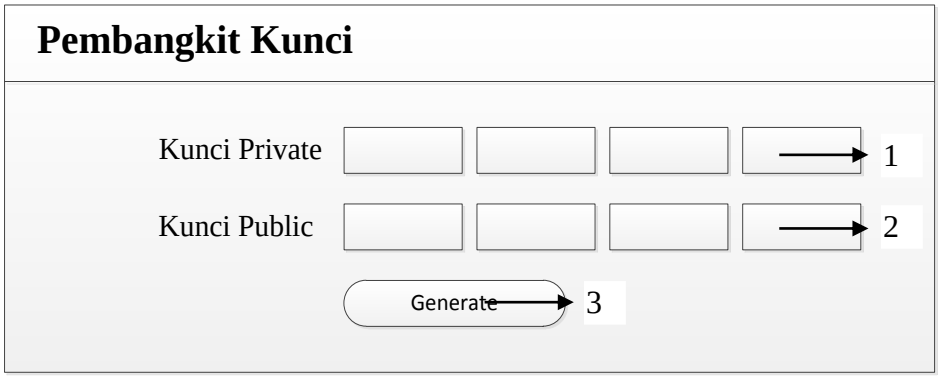
***Database* dbMhs**			
Nama Tabel	**No**	**Tipe**	**Atribut**
mahasiswa	1 2 3	*Varchar* *Varchar* *Varchar*	- nama (50) - *nim (15) - matakuliah (25)

III.6. Desain *User Interface*

Setelah perancangan diagram telah dibuat maka selanjutnya adalah perancangan *user interface* sebagai berikut :

1. *User Interface* Pembangkit Kunci

Pada gambar III.8 adalah tampilan *user interface* pembangkit kunci tampilan ini digunakan untuk proses pembangkit kunci private dan kunci public.



Gambar III.8. Desain *User Interface* Pembangkit Kunci

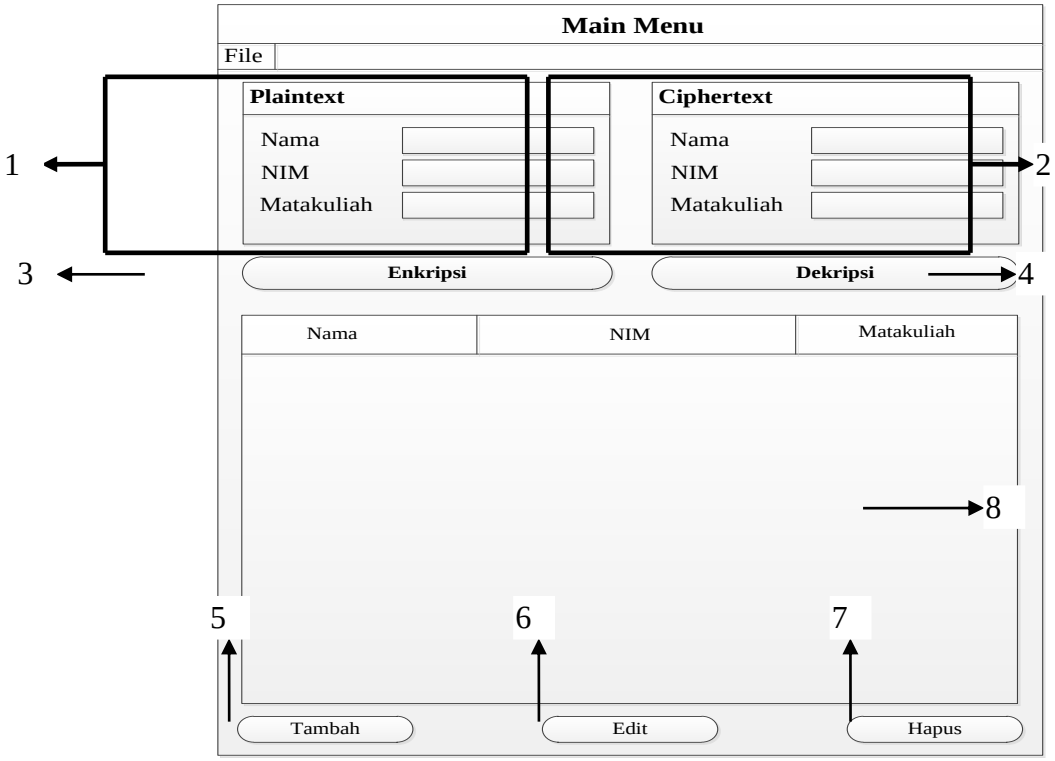
Keterangan dari desain *user interface* pembangkit kunci adalah sebagai berikut:

1. *Textbox* kunci *private* untuk menginputkan kunci *private*

2. *Textbox* kunci *public* untuk menampilkan kunci *public*
3. *Button generate* untuk memproses kunci *private* dan menampilkan hasil kunci *public*

4. *User Interface* Menu Utama

Pada gambar III.9 adalah tampilan *user* menu utama, tampilan ini akan muncul saat pertama kali aplikasi dijalankan. Tampilan ini digunakan untuk menginputkan data, mengedit, maupun menghapus. Tampilan ini juga digunakan untuk proses enkripsi dan dekripsi.



Gambar III.9. Desain *User Interface* Menu Utama

Keterangan dari Desain *User Interface* Menu Utama adalah sebagai berikut:

1. Panel *Plaintext* sebagai *input*an *plainteks* data mahasiswa, pada panel ini terdapat tiga *textbox* yaitu:

a. *Textbox* nama untuk inputan nama mahasiswa.

b. *Textbox* NIM untuk inputan NIM mahasiswa

c. *Textbox* Matakuliah untuk inputan Matakuliah mahasiswa

2. Panel *ciphertext* untuk menampilkan hasil *cipherteks* dari panel *plaintext*.
3. Button Enkripsi untuk melakukan proses enkripsi dari panel *plainteks* dan menampilkan hasil *cipherteks* pada panel *cipherteks*.
4. *Button* dekripsi untuk melakukan proses dekripsi dari panel *cipherteks* dan menampilkan hasil *plainteks* pada panel *plainteks*
5. *Button* tambah untuk melakukan proses penambahan pada tabel mahasiswa
6. *Button* edit untuk melakukan proses edit pada tabel mahasiswa
7. *Button* hapus untuk melakukan proses menghapus isi dari tabel mahasiswa