BAB III

ANALISA DAN DESAIN SISTEM

III.1 Analisa Masalah

Dalam melakukan pengamanan data SMS kita harus mengerti tentang masalah keamanan dan kerahasiaan data merupakan hal yang sangat penting dalam suatu organisasi maupun pribadi. Sistem keamanan pengiriman data dipasang untuk mencegah pencurian, kerusakan dan penyalahgunaan data melelui pesan. Jika hal ini sampai terjadi, maka kemungkinan besar akan merugikan bahkan membahayakan orang yang mengirim pesan atau menerima pesan, maupun organisasinya. Informasi yang terkandung di dalamnyapun bisa saja berubah sehingga menyebabkan salah penafsiran oleh penerima.

Dalam pembahasan kriptografi yang sedang dibahas yaitu mengenai mengamankan isi dari sebuah pesan singkat pada *platform* android dengan menggunakan algoritma kriptografi DES. Berikut dibawah ini analisa rancangan dari permasalahan yang sedang dibahas :

1. Kurangnya keamanan pada saat pengiriman pesan singkat melalui SMS.
2. Memanfaatkan layanan SMS untuk mengirim pesan atau informasi yang

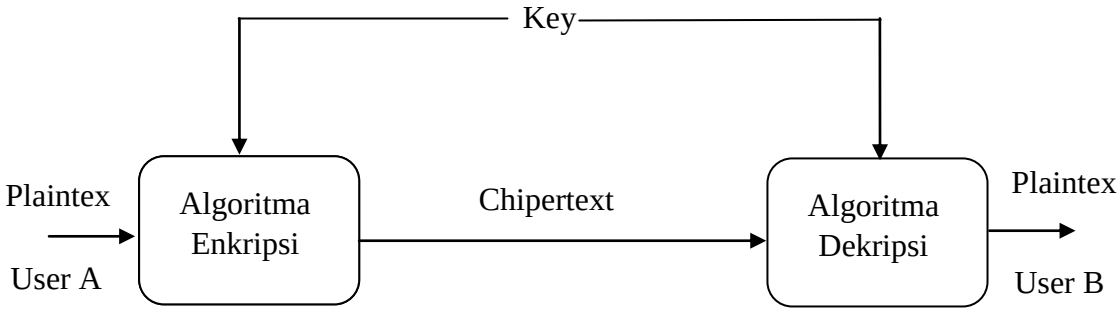
III.2 Penerapan Algoritma DES

Algoritma kriptografi disebut juga *chiper* yaitu fungsi matematika yang digunakan untuk melakukan enkripsi dan dekripsi suatu data atau pesan. Berdasarka jenis kuncinya jenis kuncinya algoritma kriptografi dibagi menjadi dua bagian yaitu :

1. Skema Enkripsi Kunci Simetrik

Dalam sistem kriptografi skema enkripsi kunci simetrik, kunci yang digunakan untuk proses enkripsi dan proses dekripsi adalah sama atau simetrik. Suatu *plaintext* dienkripsi dengan menggunakan suatu algoritma dan kunci sehingga dihasilkan suatu *chipertext*. Kemudian *cipertext* tersebut disimpan atau dikirim ke pihak lain. Selanjutnya untuk mendapatkan kembali *plaintext* dari *chipertext* dengan menggunakan algoritma dan kunci yang sama. Karena kuncinya sama, ketika data atau informasi digunakan untuk dua atau lebih pengguna, maka masing-masing pengguna yang terlibat harus saling menaruh kepercayaan untuk tidak membocorkan kunci yang telah disepakati.

Keuntungan dari skema enkripsi kunci simetrik ini adalah lebih cepat dalam melakukan prosesnya dibandingkan terhadap skema kunci asimetrik. Keuntungan inilah yang menjadi alasan mengapa skema ini masih banyak digunakan sekalipun diakui kurang aman dibandingkan dengan kunci asimetrik. Contoh skema enkripsi kunci simetrik antara lain : DES (*Data Encryption Standart*), IDEA (*International Data Encryption Algorithm*), FEAL, RC5.

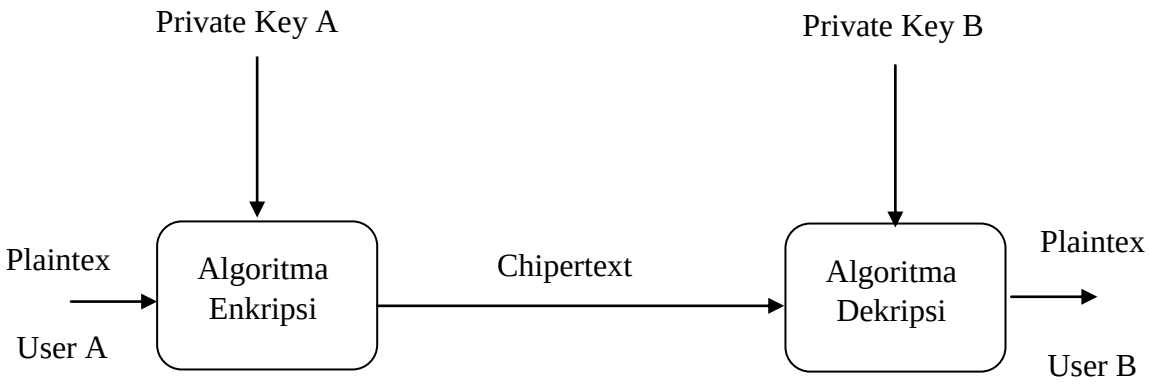


Gambar III.1 Model Kriptografi Simetrik

2. Skema Enkripsi Kunci Asimetrik

Dalam kriptografi skema kunci simetrik, semua pihak pengguna mengetahui dan menggunakan kunci yang sama, yang digunakan untuk keperluan enkripsi dan dekripsi. Sebagaimana telah disebutkan pada bagian sebelumnya skema ini memiliki persoalan mendasar dalam manajemen kuncinya untuk memecahkan masalah ini.

Hubungan antara kunci publik dan kunci pribadi merupakan fungsi satu arah (*one way function*). Seseorang dapat mengirim pesan rahasia dengan hanya menggunakan kunci publik tetapi pesan *chipertext* hanya dapat didekripsi dengan kunci pribadi oleh penerima pesan.



Gambar III.2 Model Kriptografi Asimetrik

III.2.1 Proses Perputaran DES

Didalam proses *enchipering*, blok plainteks terbagi menjadi dua bagian, kiri (L) dan kanan (R) yang masing-masing panjangnya 32 bit. Kedua bagian ini masuk kedalam putaran DES. Pada setiap putaran i, blok R merupakan masukan

untuk fungsi transformasi yang disebut F. Pada fungsi F, blok R dikombinasikan dengan kunci internal Ki. Keluaran dari fungsi F di-XOR-kan dengan blok L untuk mendapatkan blok R yang baru. Sedangkan blok L yang baru langsung diambil dari blok R sebelumnya. Ini adalah satu putaranDES secara matematis, satu putaran DES dinyatakan sebagai berikut:

$$L_i = R_{i-1}$$

$$R_i = L_i - 1 \oplus f(R_i\text{-}1, K_i \quad \text{.....(1)}$$

Sebelum putaran pertama, terhadap blok plainteks dilakukan permutasi awal (*initial permutation* atau IP). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan bit-bit dalamnya berubah. Karena ada 16 putaran, maka dibutuhkan kunci internal sebanyak 16 buah. Kunci-kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi. Kunci internal dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Kunci eksternal panjangnya 64 bit atau 8 karakter.

Dalam permutasi ini, tiap bit kedelapan (*parity bit*) dari delapan *byte* kunci diabaikan. Hasil permutasinya adalah sepanjang 56 bit, sehingga dapat dikatakan panjang kunci des adalah 56 bit. Selanjutnya 56 bit ini dibagi menjadi 2 bagian, kiri dan kanan yang masing-masing panjangnya 28 bit, yang masing-masing disimpan di dalam C0 dan D0. Selanjutnya, kedua bagian digeser ke kiri (*left shift*) sepanjang satu atau dua bit bergantung pada tiap putaran. Oprasi pergeseran bersifat *weapping* atau *round-shift*. Misalkan (C_i , D_i) meyatakan penggabungan Ci dan Di. (Ci+1, Di+1) diperoleh dengan menggeser Ci dan Di satu atau dua bit.

Jadi setiap kunci internal Ki mempunyai panjang 48 bit. Bila jumlah pergeseran bit-bit pada tabel 1 dijumlahkan senuanya, maka jumlah seluruhnya

sama dengan 28, yang sama dengan jumlah bit pada Ci dan Di. Karena itu, setelah putaran ke-16 akan didapatkan kembali C16 = C0 dan D16 = D0. E adalah fungsiekspansi yang memperluas blok Ri – 1 yang panjangnya 32 bit menjadi blok 48 bit. Selanjutnya hasil ekspansi yaitu E(Ri – 1), yang panjangnya 48 bit di-XOR-kan dengan Ki yang panjangnya 48 bit menghasilkan vektor A yang panjangnya 48 bit.

$$E(R_{i-1}) \oplus Ki = A\,i \text{ .....(2)}$$

Vektor A dikelompokan masing-masing 6 bit, dan menjadi masukan bagi proses substitusi. Proses subsutusi dilakukakn dengan menggunakan delapan bua kotak – S (S-box), S1 sampai S8. Setiap kotak S menerima masukan 6 bit dan menghasilkan 4 bit. Kelompok 6 bit pertama menggunakan S1, kelompok 6 bit kedua menggunakan S2 dan seterusnya. Keluaran proses substitusi adalah vektor B yang panjang yang panjangnya 48 bit. Vektor B menjadi masukan untuk proses permutasi. Tujuan permutasi adalah untuk mengacak hasil proses substitusi kotak-S. Bit-bit P(B) merupakan keluaran dari fungsi F, akhirnya bit-bit P(B) di-XOR-kan dengan Li – 1 untuk mendapatkan Ri.

$$Ri = Li\text{-}_1 \oplus P(B) \text{ .....(3)}$$

Jadi keluaran dari putaran ke-i adalah $(L_i\ ,\ R_i) = (R_{i-1}\ ,\ L_{i-1} \oplus P(B))$

Proses dekripsi terhadap chiperteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Keluaran pada setiap putran *dechipering* adalah

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f\,(R_{i-1}\ ,\ Ki) \text{ .....(4)}$$

Dalam hal ini, (R16, L16) adalah blok masukan awal untuk *deciphering.* Blok (R16, L16) diperoleh dengan mempermutasikan *chipertext* dengan matriks permutasi IP^{-1}. Pra-keluaran dari *deciphering* adalah (L0, R0). Dengan permutasi awal IP akan didapatkan kembali blok plainteks semula. Selama *dechipering*, K16 dihasilkan dari (C16, D16) dengan permutasi PC-2. Tentu saja (C16, D16) tidak diperoleh langsung pada permulaan *deciphering*. Tetapi karena (C16,D16) = (C0, D0), maka K16 dapat dihasilkan dari (C0, D0) tanpa perlu lagi melakukan pergeseran bit.

III.2.2 Perhitungan Algoritma DES

Adapun langkah-langkah data menggunakan algoritma DES (*Data Encryption Standart*) yaitu :

Diberikan contoh :

- ➔ Plainteks (X) : COMPUTER
- ➔ Key (k) : 13 34 57 79 9B BC DF F1

1. Langkah Pertama

   Ubah plainteks kedalam bentuk biner

   C : 01000011
   O : 01001111
   M : 01001101
   P : 01010000
   U : 01010101
   T : 01010100
   E : 01000101
   R : 01010010

   Ubahlah key kedalam bentuk biner

13 : 00010011 9B : 10011011

34 : 00110100 BC : 10111100

57 : 01010111 DF : 11011111

79 : 01111001 F1 : 11110001

2. Langkah kedua

   Lakukan initial permutaran (IP) pada vit plainteks menggunakan tabel IP berikut :

Tabel III.1 Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	37	39	31	23	7

   Urutan bit plaintext urutan ke 58 ditaruh diposisi 1

   Urutan bit plaintext urutan ke 50 ditaruh diposisi 2

   Urutan bit plaintext urutan ke 42 ditaruh diposisi 3, dan seterusnya

   Sehingga hasil outputnya adalah :

   IP(x) : 11111111 10111000 01110110 01010111 00000000 00000000 00000110 10000011

   Pecah bit pada IP(x) menjadi 2 bagian yaitu :

   L_0 : 11111111 10111000 01110110 01010111

   R_0 : 00000000 00000000 00000110 10000011

3. Langkah ketiga

   Generate kunci yang digunakan untuk mengenkripsi plainteks dengan menggunakan tabel permutasi kompresi PC-1, pada langkah ini terjadi

kompresi dengan membuang 1 bit masing-masing blok kunci dari 64 bit menjadi 56 bit.

Tabel III.2 PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	45	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Dapat kita lihat pada tabel diatas, tidak terdapat urutan bit 8, 16, 24, 32, 40, 48, 56, 64 karena telah dikompress. Berikut hasil outputnya :

CD(k) : 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Pecah CD(k) menjadi 2 bagian kiri dan kanan, sehingga menjadi

C_0 : 1111000 0110011 0010101 0101111

D_0 : 0101010 1011001 1001111 0001111

4. Langkah keempat

Langkah pergeseran kiri (*Left Shift*) pada C_0 dan D_0 , sebanyak 1 dan 2 kali berdasarkan kali putaran yang ada pada tabel putaran sebagai berikut :

Tabel III.3 *Left Shift*

Putaran ke - i	Jumlah Pergeseran (*Left Shit*)
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Untuk putaran ke 1, dilakukan pergeseran 1 bit ke kiri

Untuk putaran ke 2, dilakukan pergeseran 1 bit ke kiri

Untuk putaran ke 3, dilakukan pergeseran 2 bit ke kiri, dst

Berikut hasil outputnya :

C_0 : 1111000 0110011 0010101 0101111

D_0 : 0101010 1011001 1001111 0001111

Digeser 1 bit ke kiri

C_1 : 1110000 1100110 0101010 1011111

D_1 : 1010101 0110011 0011110 0011110

Digeser 2 bit ke kiri

C_2 : 1100001 1001100 1010101 0111111

D_2 : 0101010 1100110 0111100 0111101

Digeser 2 bit ke kiri

C_3 : 0000110 0110010 1010101 1111111

D_3 : 0101011 0011001 1110001 1110101

Digeser 2 bit ke kiri

C_4 : 0011001 1001010 1010111 1111100

D_4 : 0101100 1100111 1000111 1010101

Digeser 2 bit ke kiri

C_5 : 1100110 0101010 1011111 1110000

D_5 : 0110011 0011110 0011110 1010101

Digeser 2 bit ke kiri

C_6 : 0011001 0101010 1111111 1000011

D_6 : 1001100 1111000 1111010 1010101

Digeser 2 bit ke kiri

C_7 : 1100101 0101011 1111110 0001100

D_7 : 0110011 1100011 1101010 1010110

Digeser 2 bit ke kiri

C_8 : 0010101 0101111 1111000 0110011

D_8 : 1001111 0001111 0101010 1011001

Digeser 1 bit ke kiri

C_9 : 0101010 1011111 1110000 1100110

D_9 : 0011110 0011110 1010101 0110011

Digeser 2 bit ke kiri

C_{10} : 0101010 1111111 1000011 0011001

D_{10} : 1111000 1111010 1010101 1001100

Digeser 2 bit ke kiri

C_{11} : 0101011 1111110 0001100 1100101

D_{11} : 1100011 1101010 1010110 0110011

Digeser 2 bit ke kiri

C_{12} : 0101111 1111000 0110011 0010101

D_{12} : 0001111 0101010 1011001 1001111

Digeser 2 bit ke kiri

C_{13} : 0111111 1100001 1001100 1010101

D_{13} : 0111101 0101010 1100110 0111100

Digeser 2 bit ke kiri

C_{14} : 1111111 0000110 0110010 1010101

D_{14} : 1110101 0101011 0011001 1110001

Digeser 2 bit ke kiri

C_{15} : 1111100 0011001 1001010 1010111

D_{15} : 1010101 0101100 1100111 1000111

Digeser 1 bit ke kiri

C_{16} : 1111000 0110011 0010101 0101111

D_{16} : 0101010 1011001 1001111 0001111

Setiap hasil putaran digabungkan kembali menjadi CiDi dan diinput kedalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data CiDi 56 bit menjadi CiDi 48 bit.

Tabel III.4 PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Berikut hasil outputnya:

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

C_2D_2 = 1100001 1001100 1010101 0111111 0101010 1100110 0111100 0111101

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101

C_3D_3 = 0000110 0110010 1010101 1111111 0101011 0011001 1110001 1110101

K_3 = 010101 011111 110010 001010 010000 101100 111110 011001

C_4D_4 = 0011001 1001010 1010111 1111100 0101100 1100111 1000111 1010101

K_4 = 011100 101010 110111 010110 110110 110011 010100 011101

C_5D_5 = 1100110 0101010 1011111 1110000 0110011 0011110 0011110 1010101

K_5 = 011111 001110 110000 000111 111010 110101 001110 101000

C_6D_6 = 0011001 0101010 1111111 1000011 1001100 1111000 1111010 1010101

K_6 = 011000 111010 010100 111110 010100 000111 101100 101111

C_7D_7 = 1100101 0101011 1111110 0001100 0110011 1100011 1101010 1010110

K_7 = 111011 001000 010010 110111 111101 100001 100010 111100

C_8D_8 = 0010101 0101111 1111000 0110011 1001111 0001111 0101010 1011001

K_8 = 111101 111000 101000 111010 110000 010011 101111 111011

C_9D_9 = 0101010 1011111 1110000 1100110 0011110 0011110 1010101 0110011

K_9 = 111000 001101 101111 101011 111011 011110 011110 000001

$C_{10}D_{10}$ = 0101010 1111111 1000011 0011001 1111000 1111010 1010101 1001100

K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111

$C_{11}D_{11}$ = 0101011 1111110 0001100 1100101 1100011 1101010 1010110 0110011

K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110

$C_{12}D_{12}$ = 0101111 1111000 0110011 0010101 0001111 0101010 1011001 1001111

K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001

$C_{13}D_{13}$ = 0111111 1100001 1001100 1010101 0111101 0101010 1100110 0111100

K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001

$C_{14}D_{14}$ = 1111111 0000110 0110010 1010101 1110101 0101011 0011001 1110001

K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010

$C_{15}D_{15}$ = 1111100 0011001 1001010 1010111 1010101 0101100 1100111 1000111

K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010

$C_{16}D_{16}$ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

5. Langkah kelima

   Pada langkah ini, kita akan meng-ekspansi data R_i-1 32 bit menjadi R_i 48 bit sebanyak 16 kali putaran dengan nilai perputaran 1<= i <=16 menggunakan Tabel Ekspansi (E).

Tabel III.5 Ekspansi (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

   Hasil $E(R_{i-1})$ kemudian di XOR dengan Ki dan menghasilkan Vektor Matriks Ai.

   Berikut hasil outputnya:

   Iterasi 1

   E(R(1)-1) = 100000 000000 000000 000000 000000 001101 010000 000110

   K1 = 000110 110000 001011 101111 111111 000111 000001 110010

---------- XOR

A1 = 100110 110000 001011 101111 111111 001010 010001 110100

Iterasi – 2

E(R(2)-1) = 011010 101110 100001 010110 100110 100101 010000 001101

K2 = 011110 011010 111011 011001 110110 111100 100111 100101

---------- XOR

A2 = 000100 110100 011010 001111 010000 011001 110111 101000

Iterasi – 3

E(R(3)-1) = 010001 010111 111011 110011 110001 010101 010010 100001

K3 = 010101 011111 110010 001010 010000 101100 111110 011001

---------- XOR

A3 = 000100 001000 001001 111001 100001 111001 101100 111000

Iterasi – 4

E(R(4)-1) = 010111 110001 010111 110011 110101 011100 001111 110001

K4 = 011100 101010 110111 010110 110110 110011 010100 011101

---------- XOR

A4 = 001011 011011 100000 100101 000011 101111 011011 101100

Iterasi – 5

E(R(5)-1) = 110110 101001 011100 000101 011001 011010 100110 100011

K5 = 011111 001110 110000 000111 111010 110101 001110 101000

---------- XOR

A5 = 101001 100111 101100 000010 100011 101111 101000 001011

Iterasi – 6

E(R(6)-1) = 100101 011011 110001 010110 101110 101100 000111 111010

K6 = 011000 111010 010100 111110 010100 000111 101100 101111

---------- XOR

A6 = 111101 100001 100101 101000 111010 101011 101011 010101

Iterasi – 7

E(R(7)-1) = 110010 100001 011111 110010 100111 111101 011001 010011

K7 = 111011 001000 010010 110111 111101 100001 100010 111100

---------- XOR

A7 = 001001 101001 001101 000101 011010 011100 111011 101111

Iterasi – 8

E(R(8)-1) = 111100 001010 101001 010101 010011 110000 001010 100011

K8 = 111101 111000 101000 111010 110000 010011 101111 111011

---------- XOR

A8 = 000001 110010 000001 101111 100011 100011 100101 011000

Iterasi – 9

E(R(9)-1) = 010010 101111 111000 000000 000010 101111 110101 010001

K9 = 111000 001101 101111 101011 111011 011110 011110 000001

---------- XOR

A9 = 101010 100010 010111 101011 111001 110001 101011 010000

Iterasi – 10

E(R(10)-1)= 100111 111000 001110 100010 100111 110111 111000 001010

K10 = 101100 011111 001101 000111 101110 100100 011001 001111

---------- XOR

A10 = 001011 100111 000011 100101 001001 010011 100001 000101

Iterasi – 11

E(R(11)-1)= 010011 110111 111010 101010 101111 110011 110001 011001

K11 = 001000 010101 111111 010011 110111 101101 001110 000110

---------- XOR

A11 = 011011 100010 000101 111001 011000 011110 111111 011111

Iterasi – 12

E(R(12)-1)= 001001 011010 101001 011111 110001 010111 110010 101100

K12 = 011101 010111 000111 110101 100101 000110 011111 101001

---------- XOR

A12 = 010100 001101 101110 101010 010100 010001 101101 000101

Iterasi – 13

E(R(13)-1)= 101000 110100 011100 111011 111110 101110 101100 001010

K13 = 100101 111100 010111 010001 111110 101011 101001 000001

---------- XOR

A13 = 000011 011011 100000 101010 000000 000101 000101 001011

Iterasi – 14

E(R(14)-1)= 111001 010111 110000 001000 001000 001000 001011 111011

K14 = 010111 110100 001110 110111 111100 101110 011100 111010

---------- XOR

A14 = 101110 100011 111110 111111 110100 100110 010111 000001

Iterasi – 15

E(R(15)-1)= 000110 101100 001100 000001 011001 011010 100101 010100

K15 = 101111 111001 000110 001101 001111 010011 111100 001010

---------- XOR

A15 = 101001 010101 001010 001100 010110 001001 011001 011110

Iterasi – 16

E(R(16)-1)= 101101 011101 010100 000101 010101 010001 010110 100010

K16 = 110010 110011 110110 001011 000011 100001 011111 110101

---------- XOR

A16 = 011111 101110 100010 001110 010110 110000 001001 010111

Menghasilkan Output:

Cipher(dalam biner) = 01010110 11110001 11010101 11001000 01010010 10101111 10000001 00111111

Atau Cipher(dalam hexa) = 56 f1 d5 c8 52 af 81 3f.

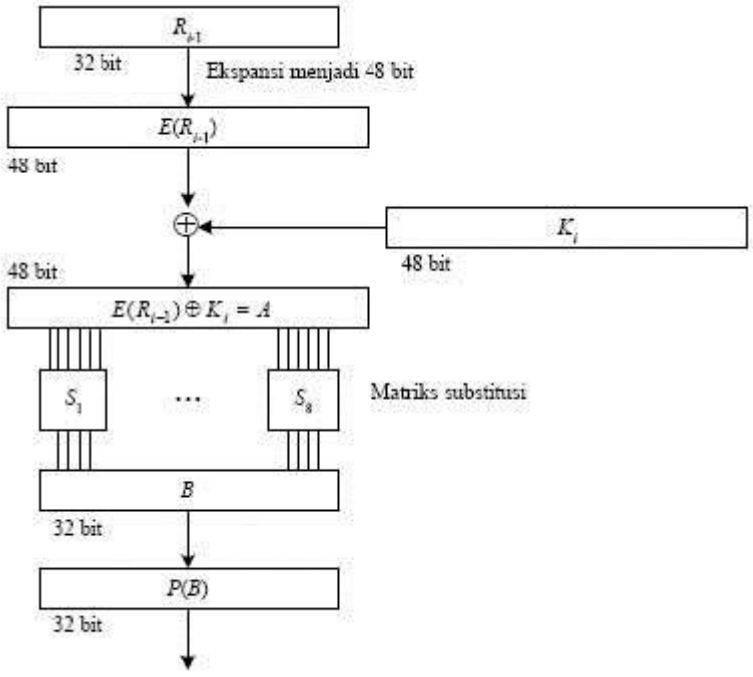
III.2.2.1 Enchipering Algoritma DES

Seperti sudah dijelaskan sebelumnya, setiap blok plaintext mengalami 16 kali putaran enchipering. Secara matematis, satu putaran DES dinyatakan sebagai berikut :

$$L^i = R^{i-1}$$

$$R^i = L^{i-1} \oplus f(R^{i-1}, K^i$$

Adapun langkah-langkah enchipering dapat dilihat pada skema berikut :



Gambar III.3 Langkah-langkah Enchiphering

Adapun penjelasan dari skema diatas adalah :

1. E merupakan fungsi ekspansi yang memperluas blok Ri-1 yang panjangnya 32-bit menjadi 48-bit. Fungsi ekspansi ini direalisasikan melalui tabel berikut :

Tabel III.6 Fungsi Ekspansi

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

2. Selanjutnya hasil ekspansi yaitu E(Ri-1), yang panjangnya 48-bit di XOR kan dengan Ki yang panjangnya juga 48-bit dan menghasilkan vektor A yang panjangnya 48-bit.
3. Vektor A kemudia dikelompokkan menjadi 8 kelompok yang masing-masing panjangnya 6 bit dan menjadi masukkan bagi proses substitusi. Proses substitusi dilakukan dengan menggunakan 8 buah kotak-s (s-box). Setiap kotak-s menerima 6 bit masukkan dan menghasilkan keluaran 4 bit. Kelompok 6 bit pertama akan menggunakan s1, 6 bit selanjutnya akan menggunakan s2, dan seterusnya. Bit awal dan akhir menentukan baris dan 4 bit ditengah akan menentukan kolom yang akan dipilih.

Kedelapan kotak S (s-box) adalah :

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	4	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Contoh pencarian output dari kotak s adalah :

Bit 100110 = keluaran dari kotak-s adalah kotak-s pertama, baris ke 2 dan kolom ke3 yaitu 8 (1000).

Keluaran proses substitusi adalah vector B yang panjangnya 48 bit. Vector B menjadi masukan untuk proses permutasi. Adapun tujuan dari proses permutasi adalah untuk mengacak hasil proses substitusi kotak-S. Hasil permutasi dinyatakan dalam fungsi f(Ri-1,Ki). Permutasi ini dilakukan dengan menggunakan matriks permutasi sebagai berikut :

16	7	20	21	29	12	28	17
1	15	23	26	5	8	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

DES memiliki proses yang sama untuk algoritma enkripsi dan dekripisinya. Proses pendekripsian juga dilakukan dengan menggunakan cipher Feistel sebanyak 16 round, dengan pada masing-masing round mengerjakan proses yang sama. Yang membedakan hanya urutan kunci dan juga input masukannya yang berupa ciphertext.

Pada round pertama, yang digunakan adalah K16, round kedua menggunakan K15, dan seterusnya hingga round terakhir akan menggunakan K1.

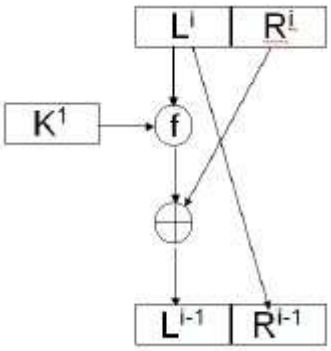
Ciphertext yang digunakan yaitu berupa L16R16 yang diperoleh dari hasil permutasi invers IP-1 terhadap ciphertext sebenarnya (y) kemudian menukar 32 bit pertama dengan 32 bit terakhir dari hasil tersebut.

$$y = IP^{-1}(R^{16}L^{16})$$
$$IP(y) = IP(IP^{-1}(R^{16}L^{16}))$$
$$IP(y) = (R^{16}L^{16})$$
$$R^{16}L^{16} = IP(y)$$

Proses dekripsi dilakukan dengan cara berkebalikan dari proses enkripsi, yaitu dengan menggunakan L16 R16 untuk menemukan L0 R0 atau plaintext.

$$L^{i-1} = R^i \oplus f(L^i, K^i)$$
$$R^{i-1} = L^i$$

Atau dapat ditunjukkan dengan gambar berikut untuk proses setiap round-nya.



Maka dari itu, untuk mendapatkan L^0 R^0 bisa digunakan langkah berikut :

$$L^{15} = R^{16} \oplus f(L^{16}, K^{16})$$
$$R^{15} = L^{16}$$
$$L^{14} = R^{15} \oplus f(L^{15}, K^{15})$$
$$R^{14} = L^{15}$$
...
...
...
$$L^{1} = R^{2} \oplus f(L^{2}, K^{2})$$
$$R^{1} = L^{2}$$

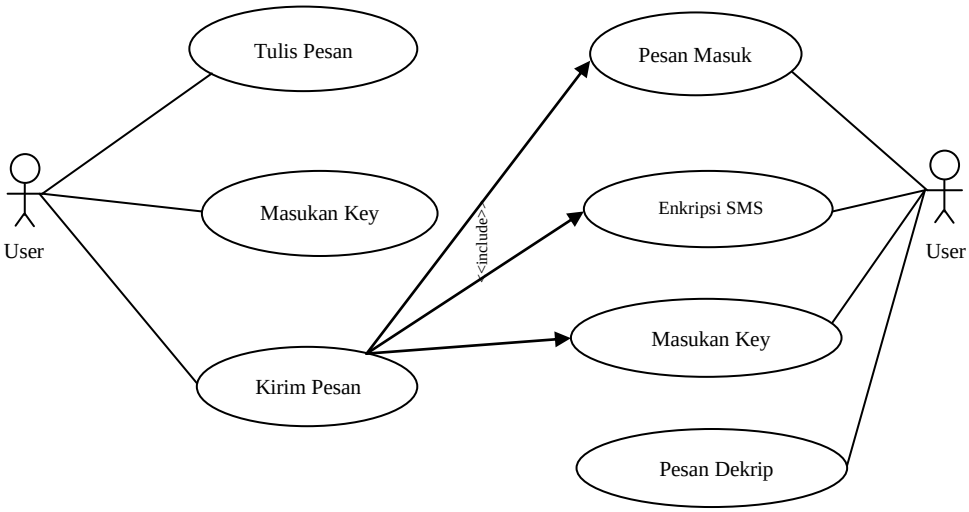
$$L^{0} = R^{1} \oplus f(L^{1}, K^{1})$$
$$R^{0} = L^{1}$$

Cara untuk mendapatkan plainteks kembali yaitu:

$$x = IP^{-1} (RD^0 LD^0)$$

III.3 Desain Sistem

Proses desain sistem merupakan dekripsi dari kebutuhan yang direprentasikan ke dalam perangkat lunak sehingga dapat diperkirakan kualitasnya sebelum dimulai pembuatan *code* atau *coding*. Dibawah ini merupakan pemodelan use case yang penulis pakai pada pembuatan aplikasi ini.



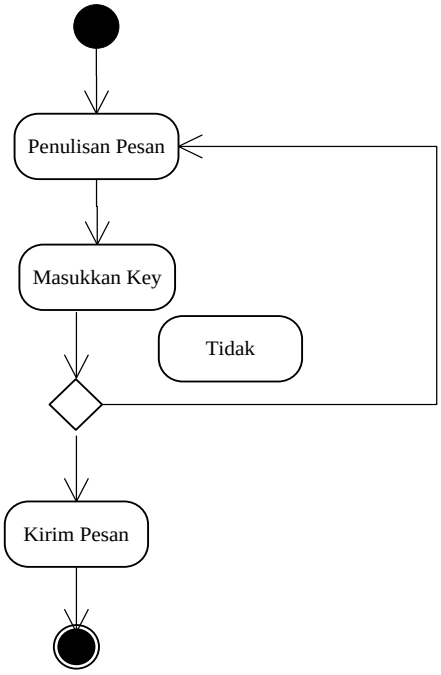
Gambar III.4 Use Case Diagram

III.3.1 *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* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Berikut adalah gambar *activity diagram* dari sistem yang dirancang yaitu :

1. ***Activity Diagram Penulisan Pesan***

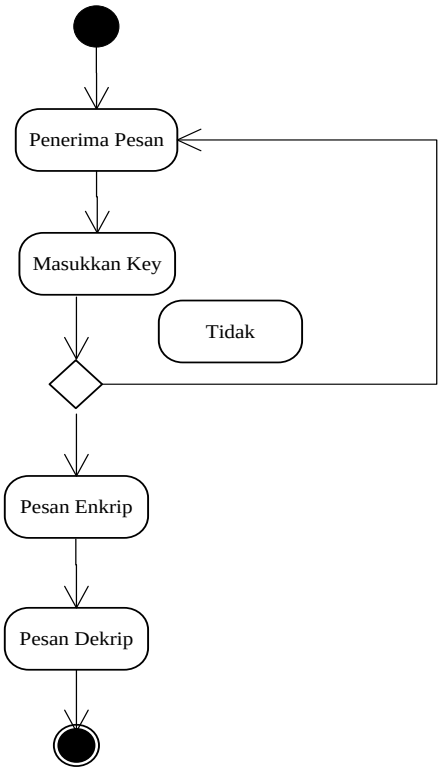
   Activity diagram Penulisan Pesan, dapat dilihat pada gambar dibawah ini :



Gambar III.5 *Activity Diagram* Penulisan Pesan

2. *Activity Diagram Penerima Pesan*

Activity diagram Penerima Pesan, dapat dilihat pada gambar dibawah ini :



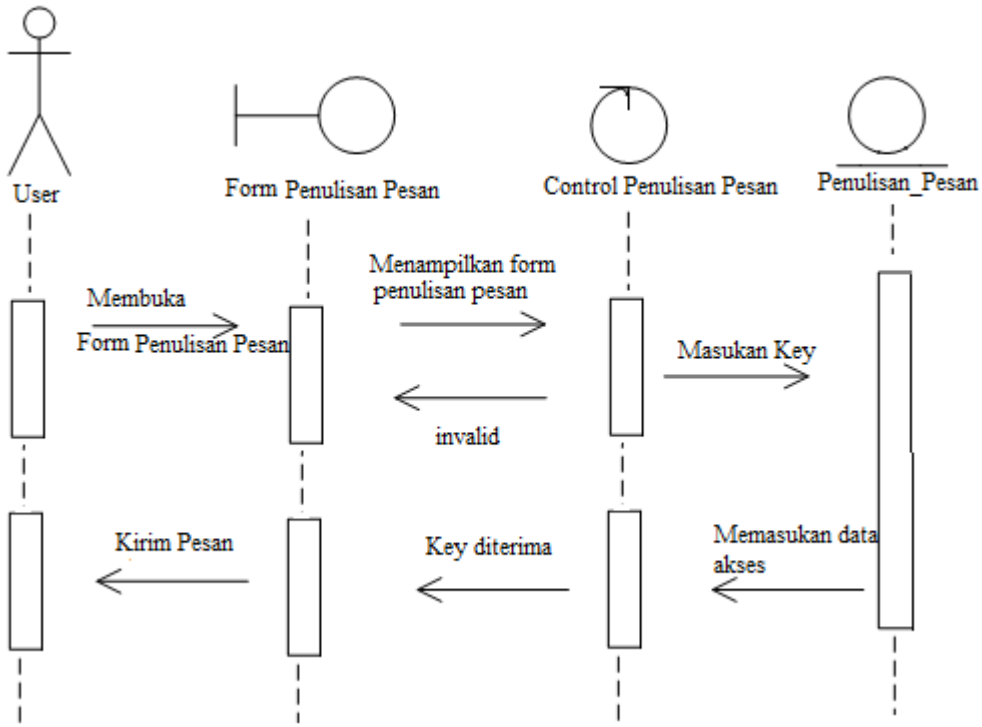
Gambar III.6 *Activity Diagram* Penerima Pesan

III.3.2 *Sequence Diagram*

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*, berikut beberapa gambar *sequence diagram* **:**

1. *Sequence Diagram Penulisan Pesan*

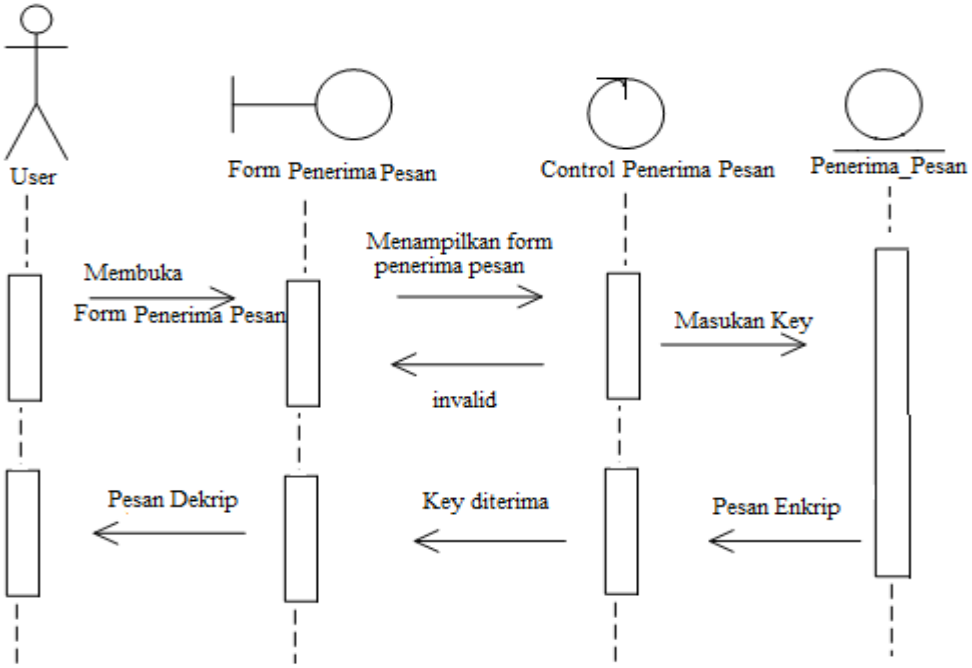
Berikut ini gambaran skenario *sequence diagram penulisan pesan*.



Gambar III.7 *Sequence Diagram* Penulisan Pesan

1. *Sequence Diagram Penerimaan Pesan*

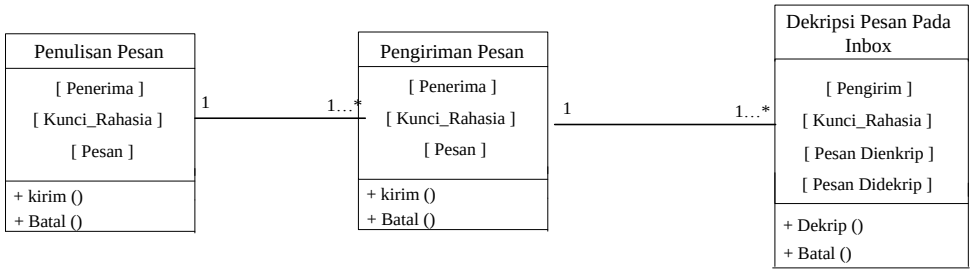
Berikut ini gambaran skenario *sequence diagram penerimaan pesan*.



Gambar III.8 *Sequence Diagram* Penerima Pesan

III.3.3 *Class Diagram*

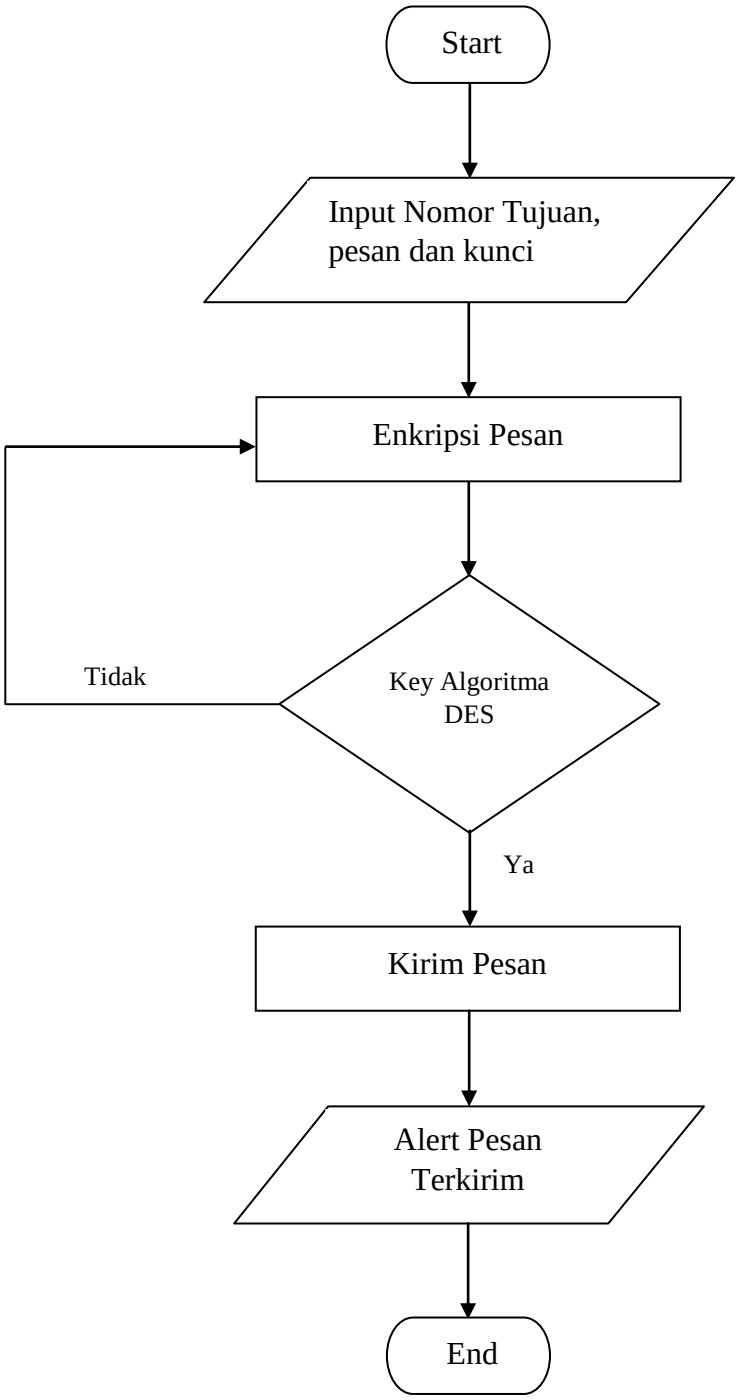
Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi), berikut gambar *Class Diagram* :



Gambar III.9 *Class Diagram* Perancangan Aplikasi Pengamanan Data SMS Dengan Algoritma DES Pada Android

III.3.4 *Flowchart*

Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung.Flowchart ini merupakan langkah awal pembuatan program. Dengan adanya flowchart urutan poses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah.

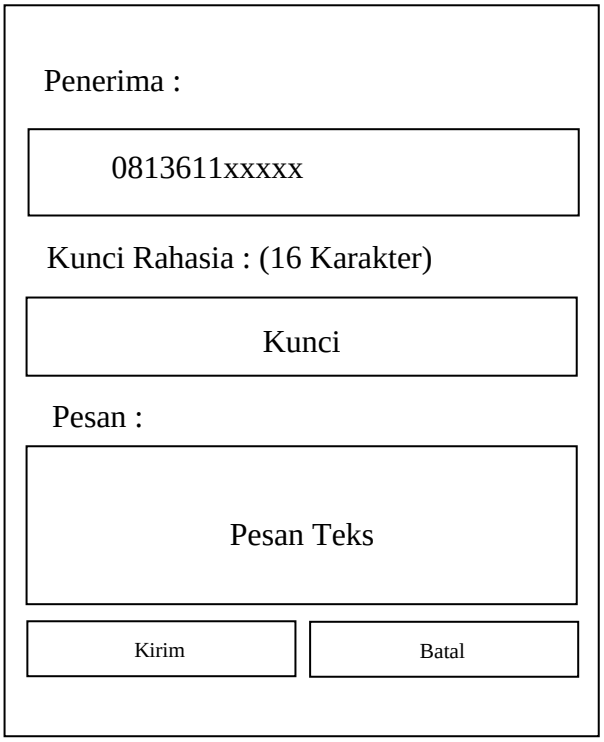


Gambar III.10 *Flowchart* Aplikasi SMS Enkripsi

Pada gambar III.5 di atas dapa dilihat cara kerja sistem ini di bagi kedalam beberapa tahapan proses supaya dapat dilihat dengan jelas tahapan proses dibagi menjadi empat sebelum sebuah sistem yang bisa digunakanuntuk bisa mengirim pesan via SMS dan pesan tersebut terenkripsi.

III.4 Desain User Interface

Desain user interface merupakan proses yang dilkaukan oleh penulis untuk merancang aplikasi tersebut. Perancangan sistem yang penulis buat secara umum adalah perancangan aplikasi pengamanan data SMS dengan algoritma DES pada android. Tahap awal dalam perancangan adalah halaman utama aplikasi, seperti yang terlihat pada gambar III.6 dibawah ini

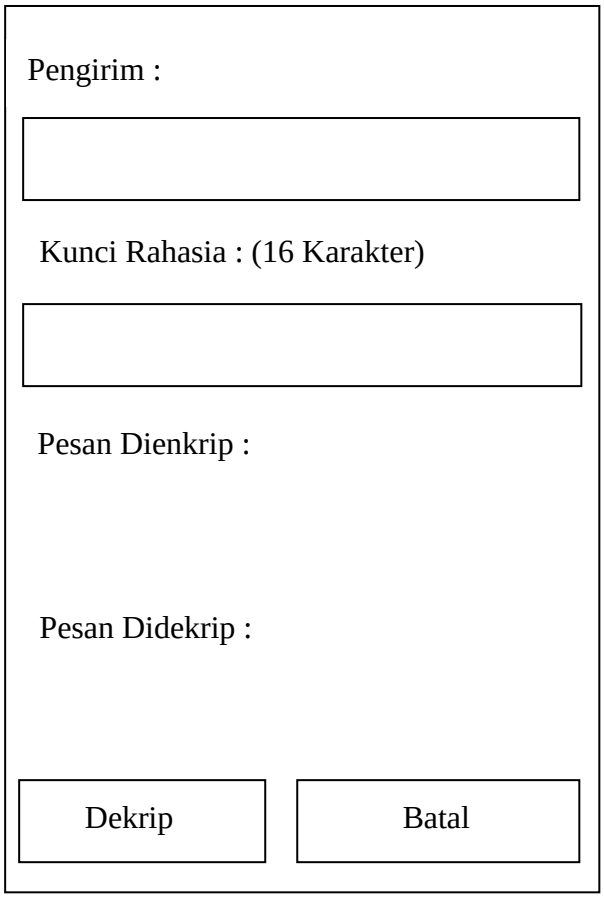


Gambar III.11*Form* Aplikasi SMS Enkripsi

Tampilan gambar III.6 merupakan tampilan untuk aplikasi enkripsi dan deskripsi DES. Tahap yang dilakukan adalah meng*input*kan nomor tujuan dan *plaintext* pada kolom *edittext* tersebut, sehingga akan melakukan proses untuk enkrip pesan tersebut kedalam metode DES, dimana dalam *form* aplikasi diatas tampak beberapa bagian yang memiliki fungsi sebagai berikut :

1. *Edittext* pada nomor tujuan berfungsi untuk menginputkan nomor tujuanyang akan dituju.
2. *Edittext* pada pesan merupakan penyandian pesan yang akan dikirim.
3. *Edittext* pada *key* merupakan data kunci dari pesan yang akan dienkripsi untuk dikirim ketujuan.
4. *Button Send SMS* adalah proses untuk mengubah menjadi suatu kode yang telah dienkrip oleh program, dimana kode tersebut akan dikirim ke nomor tujuan yang telah di inputkan.
5. *Button Cancel* adalah proses untuk membatalkan pengiriman pesan.

Adapun tampilan gambar inbox pada aplikasi ini adalah sebagai berikut :



Gambar III.12 *Form* Aplikasi Penerima SMS