BAB III

ANALISIS DAN PERANCANGAN

III.1. Analisa Masalah

Dalam era teknologi yang semakin pesat saat ini, keamanan merupakan suatu prioritas utama. Banyak tindakan-tindakan kejahatan yang sudah marak dilakukan dengan menggunakan teknologi masa kini khususnya dalam hal pencurian data yang ada pada sms. Tentunya hal ini bisa terjadi karena tingkat keamanan yang dimiliki perangkat tersebut masih tergolong rendah sehingga memungkinkan bagi orang yang tidak bertanggung jawab untuk melihat, membaca bahkan mencuri data sms tersebut. Jika hal ini sampai terjadi, maka kemungkinan besar akan merugikan bahkan membahayakan orang yang mengirim pesan atau menerima pesan. Informasi yang terkandung di dalamnya pun bisa saja berubah sehingga menyebabkan salah penafsiran oleh penerima. Untuk mencegah hal-hal yang tidak diinginkan perlu dilakukan pengamanan salah satunya dengan menyandikan data sms menjadi kode-kode yang hanya dapat dibuka oleh orang yang memiliki kunci. Sistem keamanan pengiriman data dipasang untuk mencegah pencurian, kerusakan dan penyalagunaan data melalui pesan. Algoritma yang digunakan penulis untuk mengamankan data sms pada permasalahan ini adalah algoritma kriptografi DES atau AES.

III.2. Strategi Pemecahan Masalah

Setelah melihat permasalah diatas maka penulis mencoba untuk merancang suatu aplikasi keamanan sms berbasis android yang lebih baik sehingga dapat menjamin kerahasiaan pengguna sms. Dengan menggunakan algoritma AES atau DES, masalah keamanan data sms dapat teratasi karena tingkat kesulitan dalam pemecahan kode yang ada pada kedua algoritma ini cukup tinggi.

III.3. Analisa Metode yag Digunakan

III.3.1. Penerapan Algoritma DES

DES adalah blok cipher. Ini beroperasi pada blok dari 64-bit dalam ukuran. Sebuah masukan blok 64-bit dari plaintext akan dienkripsi menjadi 64-bit output blok teks cipher. Ini adalah sebuah Algoritma simentris, yang berarti algoritma dan kunci yang sama digunakan untuk enkripsi dan dekripsi.

Keamanan DES terletak di kunci 56-bit. blok plaintext diambil dan dimasukan melalui permutasi awal. Kuncinya adalah juga diambil pada waktu yang sama. Kuncinya disajikan dalam blok 64-bit dengan setiap bit 8 menjadi bit paritas. Kunci 56-bit kemudian diekstraksi siap digunakan 64-bit blok plaintext dibagi menjadi dua bagian 32 bit, bernama kanan setengah setengah kiri. Itu dua bagian plaintext kemudian digabungkan dengan data dari kunci dalam operasi yang disebut Fungsi F. Ada 16 putaran Fungsi f, setelah itu dua bagian yang digabungkan menjadi satu 64-bit blok, yang kemudian dimasukkan melalui final
permutasi untuk menyelesaikan operasi algoritma dan 64-bit teks cipher blok dikeluarkan.

III.3.1.1. Pembangkitan Kunci Internal DES

Pembangkitan Kunci Internal DES Pada algoritma DES, dibutuhkan kunci internal sebanyak 16 buah, yaitu K1, K2,…,K16. 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 pada DES panjangnya 64-bit atau 8 karakter seperti pada Tabel III.1 : (c) Kunci Eksternal 64-bit dibawah.

Tabel III.1 : (c) Kunci Eksternal 64-bit.

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

Misalkan kunci eksternal yang tersusun atas 64-bit adalah K. Kunci eksternal ini menjadi masukan untuk permutasi dengan menggunakan matriks kompresi PC-1 seperti pada Tabel (d). Dalam permutasi ini, tiap-bit kedelapan dari delapan byte kunci diabaikan (Tabel (c). dengan kolom yang berwarna gelap). Hasil permutasinya adalah sepanjang 56-bit, sehingga dapat dikatakan panjang kunci DES adalah 56-bit.

Tabel III.2 : (d) Matriks Permutasi Kompresi PC-1

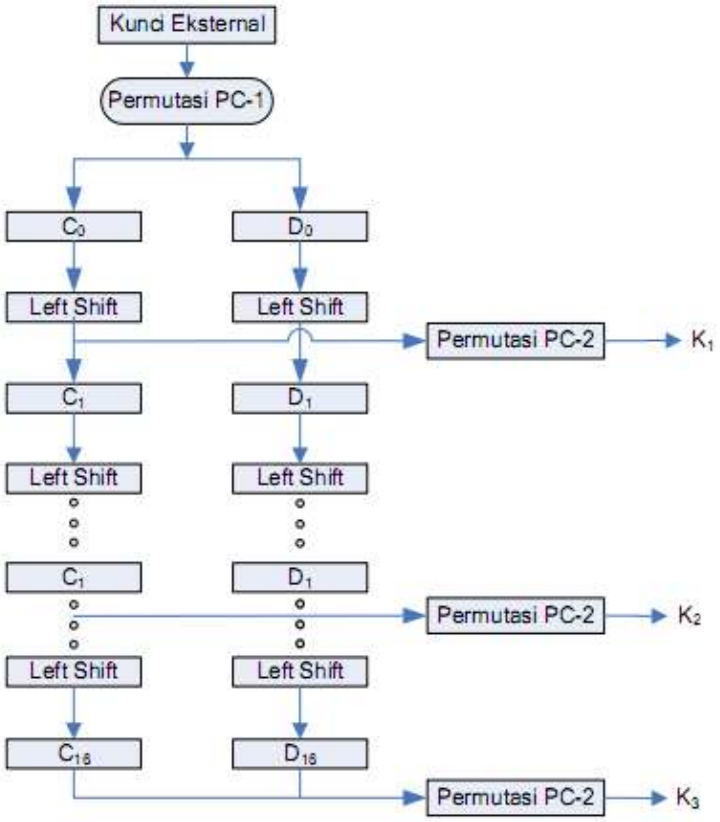
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Misalkan (Ci, Di) menyatakan penggabungan Ci dan Di. (Ci-1, Di-1) diperoleh dengan menggeser Ci dan Di satu atau dua-bit. Setelah Pergeseran-bit, (Ci, Di) mengalami permutasi kompresi dengan menggunakan matriks PC-2 seperti pada Tabel III.3 : (d) Matriks Permutasi Kompresi-2 (PC-2)

Tabel III.3 : (d) Matriks Permutasi Kompresi-2 (PC-2)

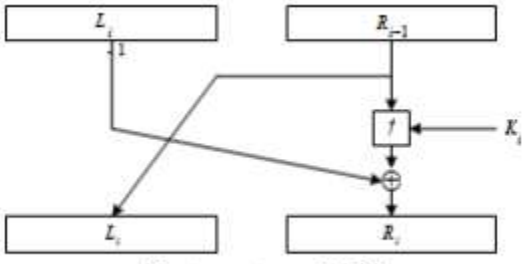
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Dengan permutasi ini, kunci internal Ki diturunkan dari (Ci, Di) yang dalam hal ini Ki merupakan panggabungan-bit-bit Ci pada sisi gelap tabel (f), dengan bit-bit Di pada sisi putih tabel (f). Setiap kunci internal Ki mempunyai panjang 48-bit. Proses Pembangkitan kunci-kunci internal dapat dilihat pada gambar berikut ini :



Gambar III.1. Skema Proses Pembangkitan Kunci-kunci Internal

Proses Enkripsi DES Proses enkripsi terhadap blok plainteks dilakukan setelah permutasi awal. Setiap blok plainteks mengalami 16 kali putaran enkripsi. Untuk setiap putaran, digambarkan seperti gambar berikut:



(Satu putaran DES)

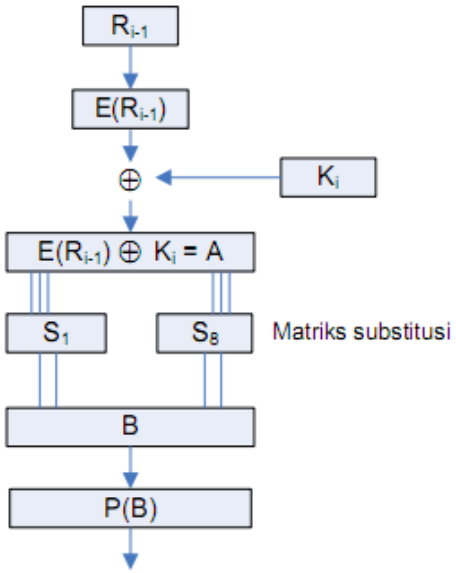
Gambar III.2. Skema 1 Putaran DES

Setiap putaran enkripsi DES secara matematis dinyatakan sebagai :

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

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \quad \text{……..} \quad [1]$$

Dengan f adalah suatu fungsi yang ditunjukkan pada Gambar berikut :



Gambar III.3. Skema Komputasi Fungsi

E adalah fungsi ekspansi yang memperluas blok Ri – 1 32-bit menjadi blok 48-bit. Fungsi ekspansi direalisasikan dengan matriks permutasi ekspansi :

Tabel III.4: (f) Matriks Permutasi 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

Hasil ekpansi, yaitu E(Ri – 1) di-XOR-kan dengan Ki menghasilkan vektor A 48-bit: Matriks A dikelompokkan menjadi 8 kelompok, masing-masing 6-bit, dan menjadi masukan bagi proses substitusi. Proses substitusi dilakukan dengan menggunakan delapan buah

kotak-S (S-box), S1 sampai S8. Setiap kotak-S menerima masukan 6-bit dan menghasilkan keluaran 4-bit. Kelompok 6-bit pertama menggunakan S1, kelompok 6-bit kedua menggunakan S2, dan seterusnya. Kedelapan kotak-S tersebut ditunjukkan pada gambar di bawah ini (klik untuk memperbesar).

Tabel III.5 : *Substitution Box* 1 (S[1])

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

Tabel III.6 : *Substitution Box* 2 (S[2])

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

Tabel III.7: *Substitution Box* 3 (S[3])

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

Tabel III.8: *Substitution Box* 4 (S[4])

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

Tabel III.9 : *Substitution Box* 5 (S[5])

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

Tabel III.10 : *Substitution Box* 6 (S[6])

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

Tabel III.11: *Substitution Box* 7 (S[7])

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

Tabel III.12: *Substitution Box* 8 (S[8])

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

Keluaran proses substitusi adalah vektor B yang panjangnya 32-bit. Vektor B menjadi masukan untuk proses permutasi. Tujuan permutasi adalah untuk mengacak hasil proses substitusi kotak-S. Permutasi dilakukan dengan menggunakan matriks permutasi P (P-box) sbb :

Tabel III.13 : Kotak P

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

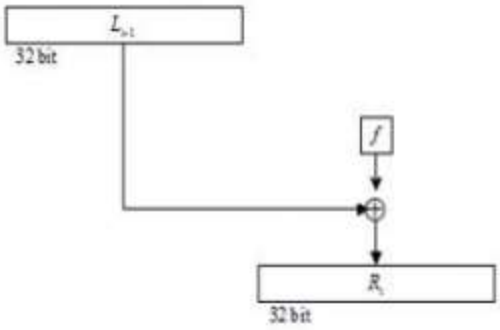
Bit-bit P(B) merupakan keluaran dari fungsi f. Akhirnya, bit-bit P(B) di-XOR-kan dengan Li1 untuk mendapatkan Ri

$$R_i = L_{i-1} (+) P(B) \ldots\ldots\ldots\ldots \quad [2]$$

Jadi, keluaran dari putaran ke-i adalah

$$(L_i, R_i) = (R_{i-1}, L_{i-1} (+) P(B)) \ldots\ldots\ldots\ldots \quad [3]$$

Skema perolehan Ri



Gambar III.4. Skema Perolehan Ri

III.3.1.2. Proses Dekripsi DES

Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah K1, K2, …, K16, maka pada proses dekripsi urutan kunci yang digunakan adalah K16, K15, …, K1. Setiap putaran 16, 15, …, 1, keluaran pada proses dekripsi adalah

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \quad \ldots\ldots\ldots\ldots \quad [4]$$

dalam hal ini, (R16, L16) adalah blok masukan awal untuk proses dekripsi. Blok (R16, L16) diperoleh dengan mempermutasikan cipherteks dengan matriks permutasi IP. Pra-keluaran dari proses dekripsi adalah adalah (L0, R0). Dengan permutasi awal IP-1 akan didapatkan kembali blok plainteks semula. Kunci-kunci dekripsi diperoleh dengan menggeser Ci dan Di dengan cara yang sama seperti pada proses enkripsi, tetapi pergeseran kiri (left shift) diganti menjadi pergeseran kanan (right shift).

III.3.2. Penerapan Algoritma AES

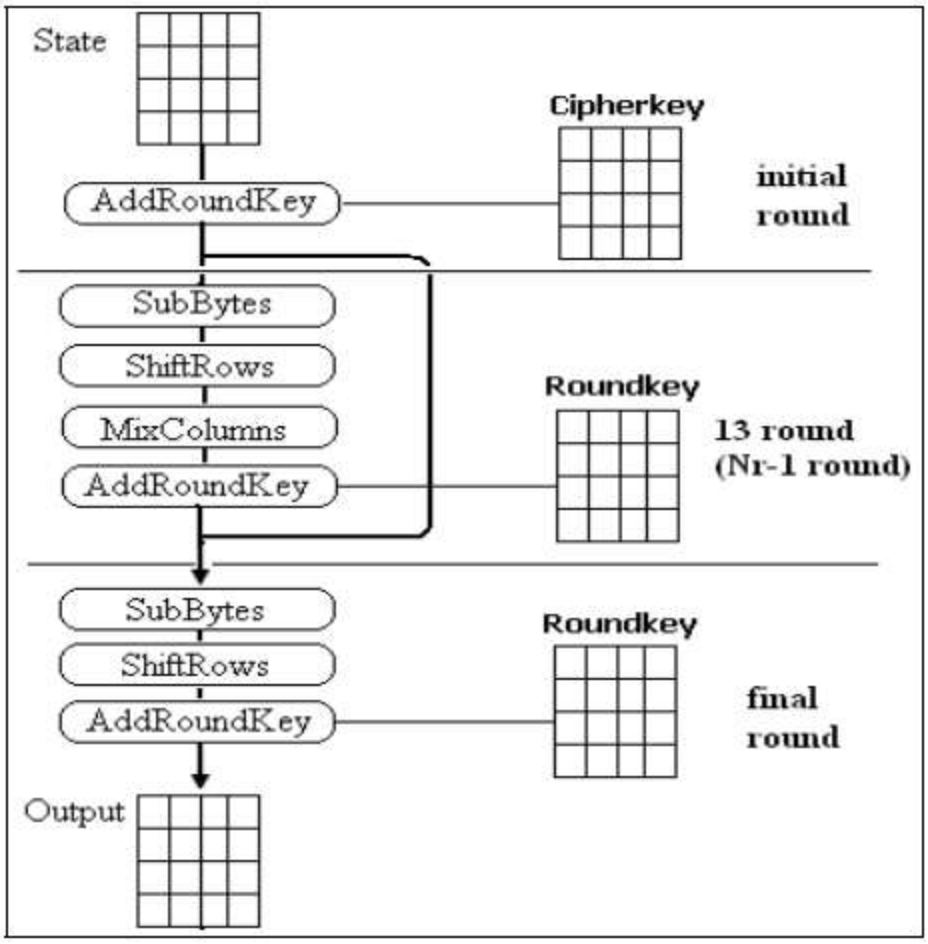
AES adalah salah satu algoritma popular yang digunakan dalam kriptografi kunci simetris. Algoritma AES menggunakan subtitusi, permutasi dan sejumlah putaran yang

dikenakan pada tiap blok yang akan dienkripsi/ dekripsi. Untuk setiap putarannya, menggunakan kunci yang berbeda *Rijndael* beroperasi dalam orientasi *byte* sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam *software* dan *hardware.*

Rijndael termasuk dalam jenis algoritma kriptografi yang sifatnya simetri dan cipher block. Dengan demikian algoritma ini mempergunakan kunci yang sama saat enkripsi dan dekripsi serta masukan dan keluarannya berupa blok dengan jumlah bit tertentu. Rijndael mendukung berbagai variasi ukuran blok dan kunci yang akan digunakan. Namun Rijndael mempunyai ukuran blok dan kunci yang tetap sebesar 128, 192, 256 bit. Pemilihan ukuran blok data dan kunci akan menentukan jumlah proses yang harus dilalui untuk proses enkripsi dan dekripsi.

III.3.3.Proses Enkripsi Advanced Enryption Standard

Proses enkripsi algoritma AES terdiri dari 4 jenis *transformasi bytes,* yaitu *SubBytes,ShiftRows, Mixcolumns,* dan *AddRoundKey*. Pada awal proses enkripsi, input yang telah dicopykan ke dalam state akan *mengalami transformasi byte AddRoundKey*. Setelah itu, stateakan mengalami *transformasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey* secara berulang-ulang sebanyak Nr. Proses ini dalam algoritma AES disebut sebagai *round function. Round* yang terakhir agak berbeda dengan round-round sebelumnya dimana pada round terakhir, *state* tidak mengalami *transformasi MixColumns*. Ilustrasi proses enkripsi AES dapat digambarkan seperti pada Gambar 2 di bawah ini :



Gambar III.5. Diagram Proses Enkripsi AES
(Sumber : Kriptografi & Implementasi Menggunakan Matlab : 2015:169)

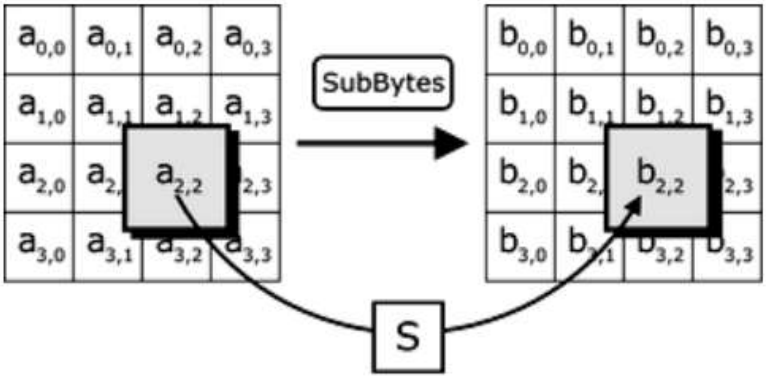
III.3.3.1. SubBytes

SubBytes merupakan *transformasi byte* dimana setiap elemen pada state akan dipetakandengan menggunakan sebuah tabel substitusi *(S-Box*). Tabel substitusi *S-Box* akan dipaparkandalam Tabel III.14.

Tabel III.14 : *S-Box SubBytes*
(Sumber : Kriptografi & Implementasi Menggunakan Matlab:2015:170)

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

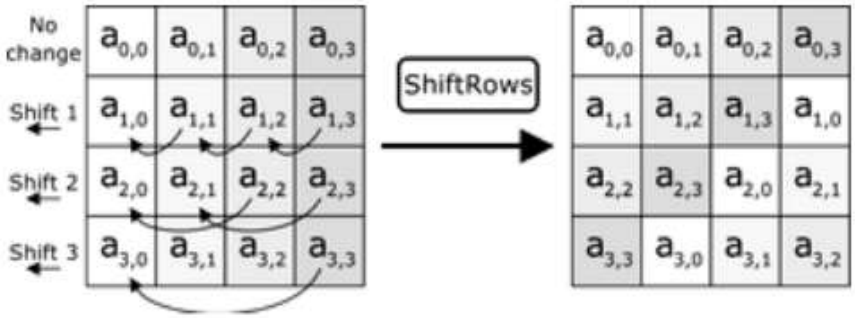
Untuk setiap byte pada array state, misalkan S[r, c] = xy, yang dalam hal ini xy adalahdigit heksadesimal dari nilai S[r, c], maka nilai substitusinya, dinyatakan dengan S'[r, c], adalahelemen di dalam tabel subtitusi yang merupakan perpotongan baris x dengan kolom y. Gambar 3mengilustrasikan pengaruh pemetaan byte pada setiap byte dalam state.



Gambar III.6. *Transformasi SubBytes* Gambar
(Sumber : Kriptografi & Implementasi Menggunakan Matlab:2015:171)

III.3.3.2. *Shiftrows*

Transformasi Shiftrows pada dasarnya adalah proses pergeseran bit dimana bit paling kiri akan dipindahkan menjadi bit paling kanan (rotasi bit). Proses pergeseran Shiftrowditunjukkan dalam Gambar 4 berikut:



Gambar III.7. *Transformasi ShiftRows*
(Sumber : Kriptografi & Implementasi Menggunakan Matlab:2015:171)

III.3.3.3. *MixColumns*

MixColumns mengoperasikan setiap elemen yang berada dalam satu kolom pada state. Secara lebih jelas, transformasi mixcolumns dapat dilihat pada perkalian matriks berikut ini:

$$\begin{bmatrix} S'_{0,C} \\ S'_{1,C} \\ S'_{2,C} \\ S'_{3,C} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,C} \\ S_{1,C} \\ S_{2,C} \\ S_{3,C} \end{bmatrix} \quad \text{…….. [5]}$$

Hasil dari perkalian matriks diatas dapat dianggap seperti perkalian yang ada di bawah ini :

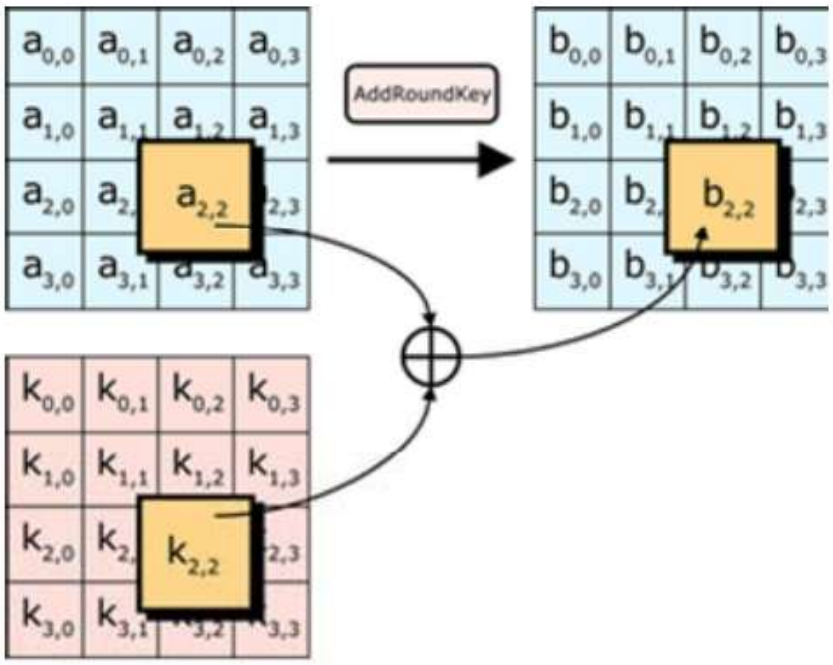
$$\begin{aligned} s'_{0,c} &= (\{02\} \bullet s_{o,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}) \end{aligned} \quad \text{……. [6]}$$

III.3.2.4. *AddRoundKey*

Pada proses enkripsi dan dekripsi AES proses *AddRoundKey* sama, sebuah round keyditambahkan pada state dengan operasi XOR. Setiap round key terdiri dari Nb word dimana tiap word tersebut akan dijumlahkan dengan word atau kolom yang bersesuaian dari state sehingga :

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*Nb+c}] \text{ untuk } 0 \le c \le Nb$$

[wi] adalah word dari key yang bersesuaian dimana i = round*Nb+c.Transformasi *AddRoundKey* pada proses enkripsi pertama kali pada round = 0 untuk round selanjutnya round = round + 1, pada proses dekripsi pertama kali pada round = 14 untuk round selanjutnya round =round - 1.

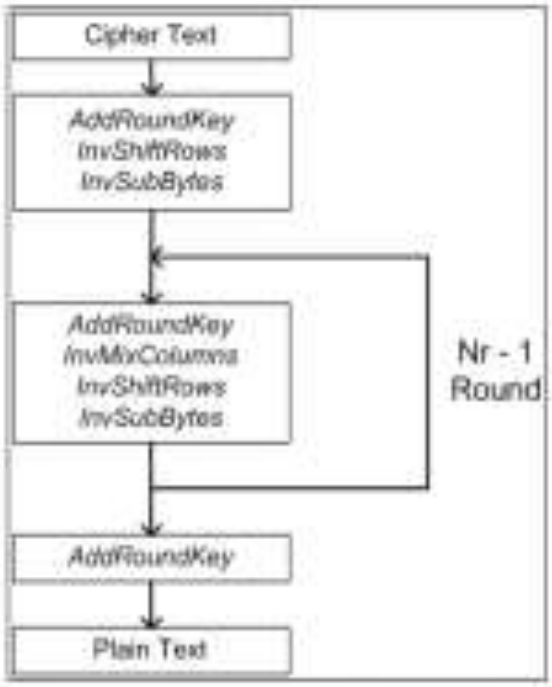


Gambar III.8. ***Transformasi AddRoundKey***
(Jurnal : Didi Surian : 2006:1)

III.3.4. Proses Dekripsi *Advanced Encryption Standard*

III.3.4.1. Proses Dekripsi AES

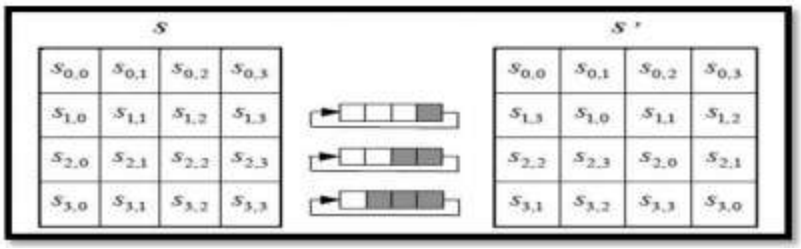
Transformasi cipher dapat dibalikka dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan inverse cipher yang mudah dipahami untuk algoritma AES. Transformasi byte yang digunakan pada *invers cipher* adalah *InvShiftRows, InvSubBytes, InvMixColumns, dan AddRoundKey*. Algoritma dekripsi dapat dilihat pada skema berikut ini :



Gambar III.9. Diagram Proses Dekripsi AES
(Sumber : Kriptografi & Implementasi Menggunakan Matlab: 2015:173)

III.3.4.2. *InvShiftRows*

InvShiftRows adalah transformasi byte yang berkebalikan dengan *transformasi ShiftRows*. Pada transformasi InvShiftRows, dilakukan pergeseran bit ke kanan sedangkan pada ShiftRows dilakukan pergeseran bit ke kiri. Ilustrasi transformasi InvShiftRows terdapat pada Gambar 6:



Gambar III.10. ***Transformasi InvShiftRows***
(Sumber : Kriptografi & Implementasi Menggunakan Matlab: 2015:174)

III.3.4.3. *InvSubBytes*

InvSubBytes juga merupakan transformasi bytes yang berkebalikan dengan transformasi SubBytes. Pada InvSubBytes, tiap elemen pada state dipetakan dengan menggunakan table Inverse S-Box. Tabel Inverse S-Box akan ditunjukkan dalam Tabel III.15 berikut:

Tabel III.15: ***InvSubBytes***
(Sumber : Kriptografi & Implementasi Menggunakan Matlab: 2015:174)

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

III.3.4.4. *InvMixColumns*

Setiap kolom dalam state dikalikan dengan matrik perkalian dalam AES. Perkalian dalam matrik dapat dituliskan :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{…….} \quad [7]$$

Hasil dari perkalian dalam matrik adalah :

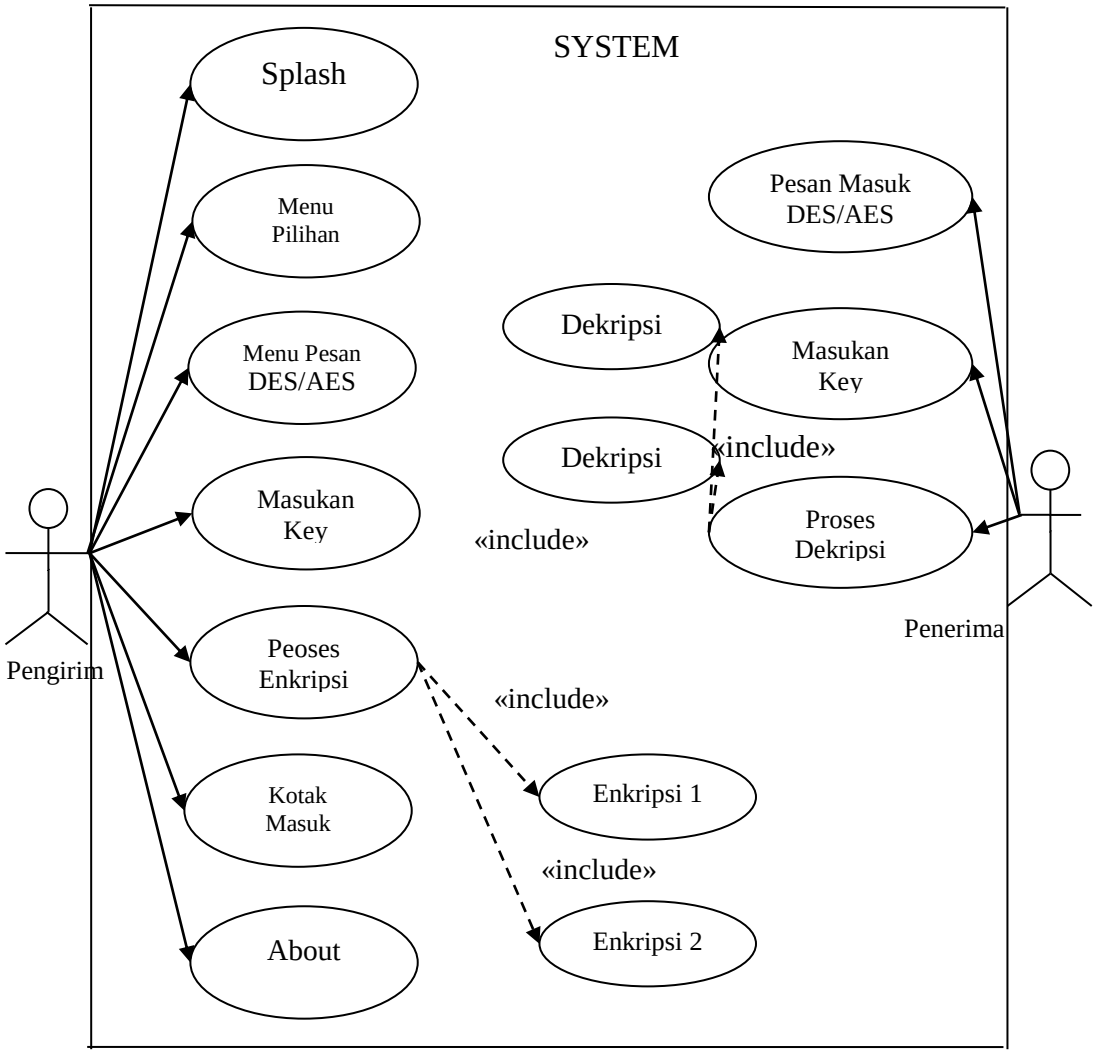
$$\begin{aligned} s'_{0,c} &= (\{0E\} \bullet s_{o,c}) \oplus (\{0B\} \bullet s_{1,c}) \oplus (\{0D\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c}) \\ s'_{1,c} &= (\{09\} \bullet s_{o,c}) \oplus (\{0E\} \bullet s_{1,c}) \oplus (\{0B\} \bullet s_{2,c}) \oplus (\{0D\} \bullet s_{3,c}) \\ s'_{2,c} &= (\{0D\} \bullet s_{o,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0E\} \bullet s_{2,c}) \oplus (\{0B\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{0B\} \bullet s_{o,c}) \oplus (\{0D\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0E\} \bullet s_{3,c}) \end{aligned} \quad \text{….} \quad [8]$$

III.4. *UML*

Penggambaran *UML* menggunakan diagram *use-case* yang selanjutnya setiap proses bisnis yang terjadi akan diperjelas dengan diagram *activity* lalu diilustrasikan secara detail menggunakan diagram *sequence*. Aktor dan pelaku yang terlibat dalam sistem adalah sebagai berikut :

III.4.1. *Use Case* Diagram

Adapun *use-case* diagram dapat dilihat pada gambar III.11 berikut.



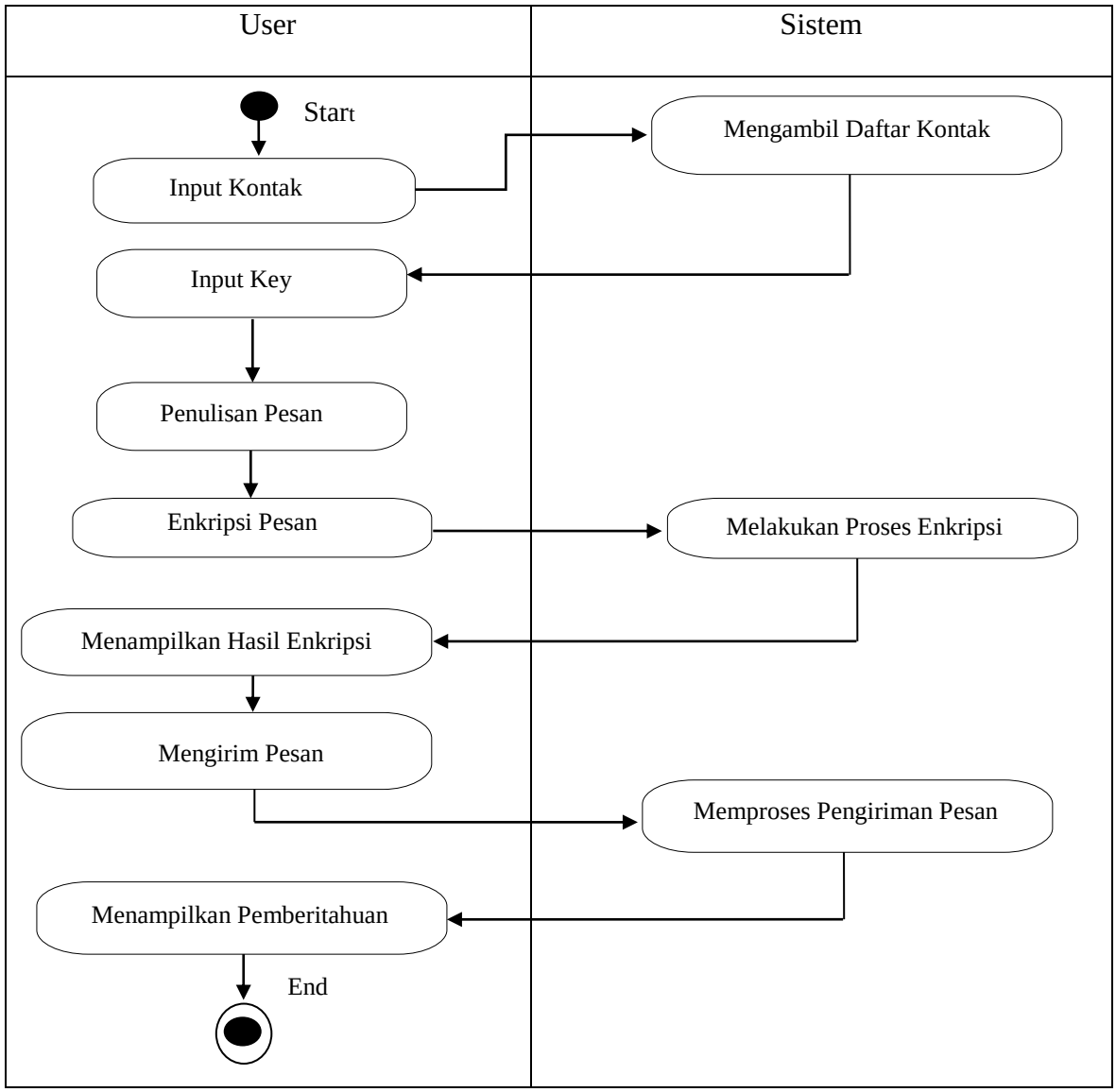
Gambar III.11. *Use Case Diagram*

III.4.2. *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 sitem yang dirancang yaitu :

1. *Activity Diagram Pengiriman Pesan*

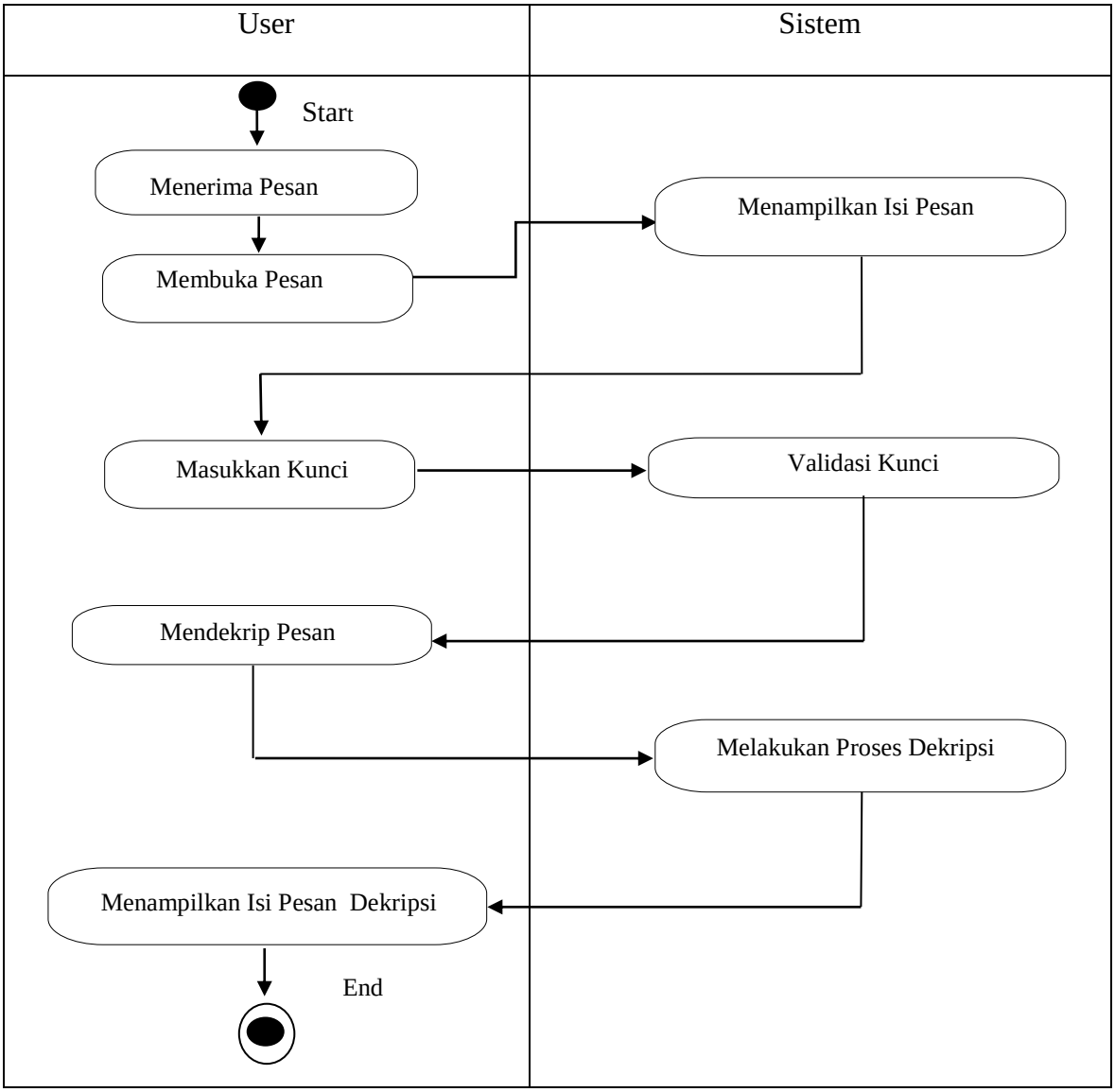
Activity diagram Pengiriman Pesan, dapat dilihat pada gambar berikut ini :



Gambar III.12. *Activity Diagram* Pengiriman Pesan

2. Activity Diagram Membaca Pesan

Activity diagram Membaca Pesan, dapat dilihat pada gambar berikut ini :



Gambar III.13. *Activity Diagram* Membaca Pesan

III.4

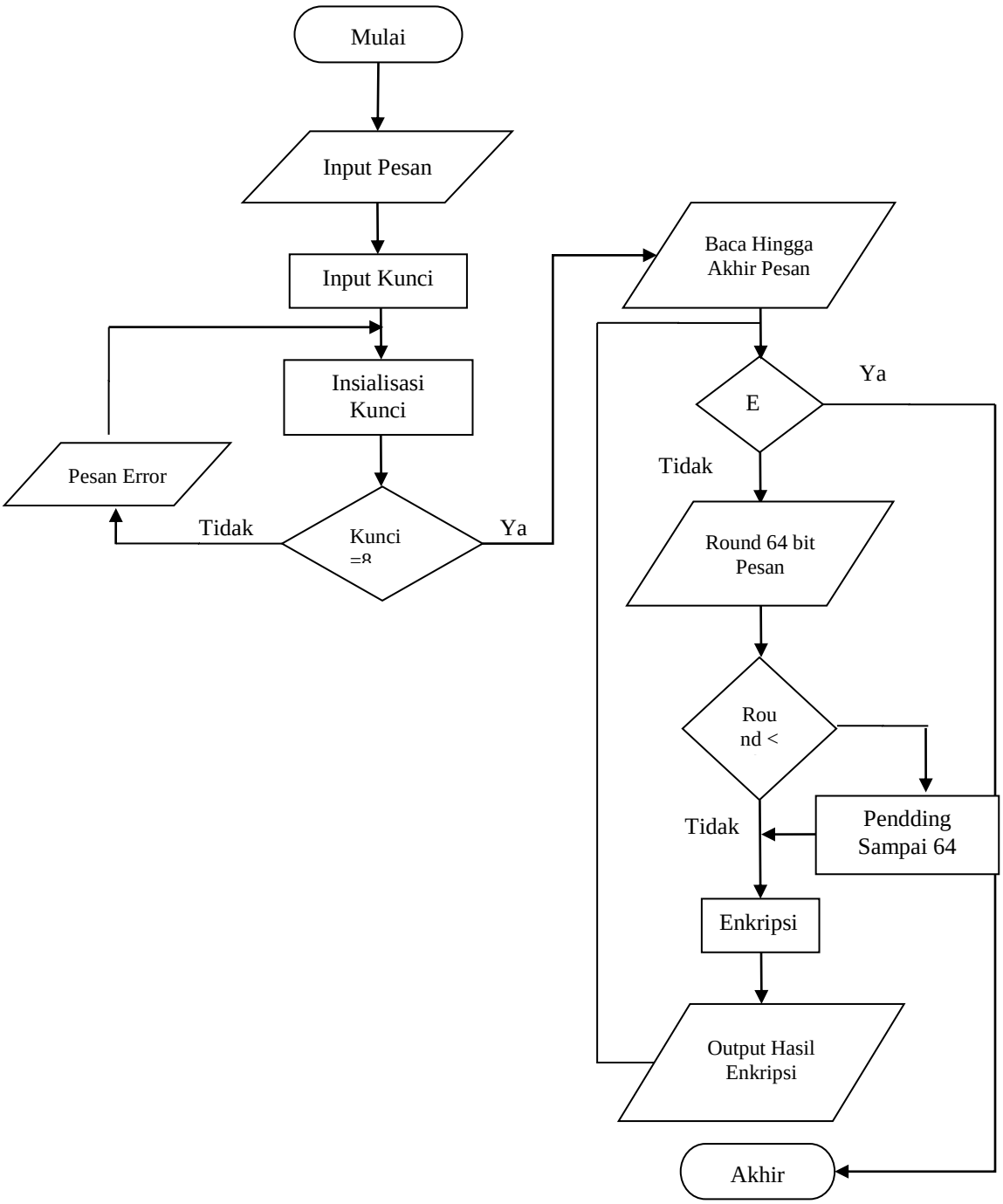
.3. *Flowchart*

Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap symbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. Flowchart ini merupakan langkah awal pembuatan program. Dengan

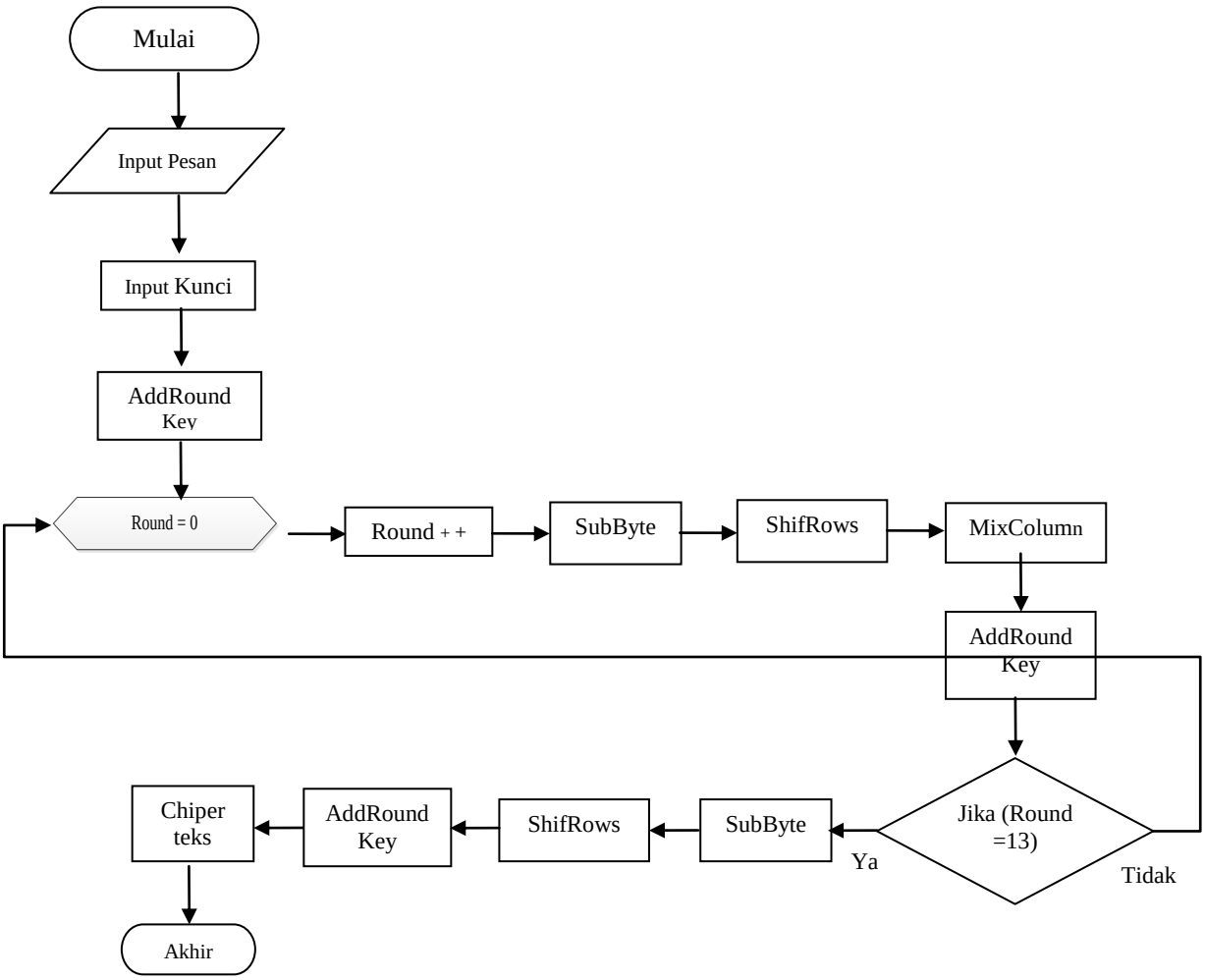
adanya flowchart urutan proses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah.

III.4.3.1. *Flowchart* Enkripsi

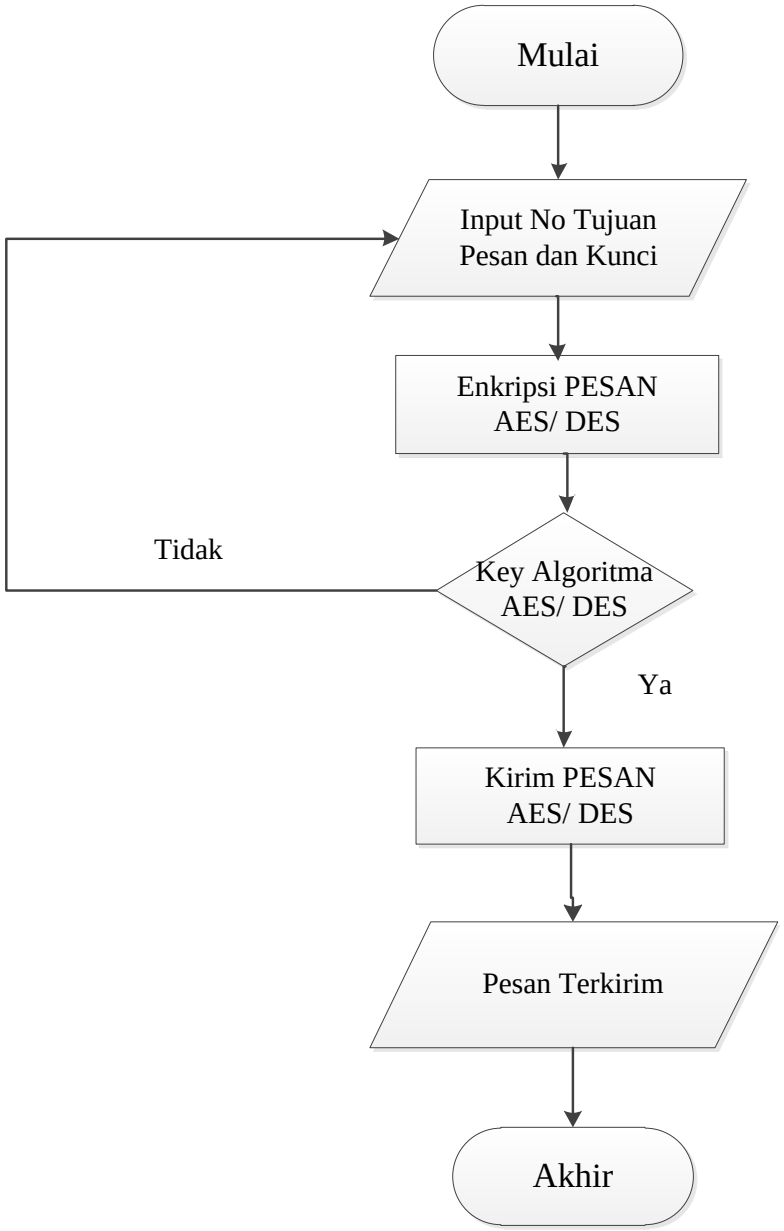
Flowchart ini dibuat untuk menjelaskan proses jalannya enkripsi pada program, seperti pada gambar III.14 berikut ini.



Gambar III.14. Flowchart Metode DES Enkripsi



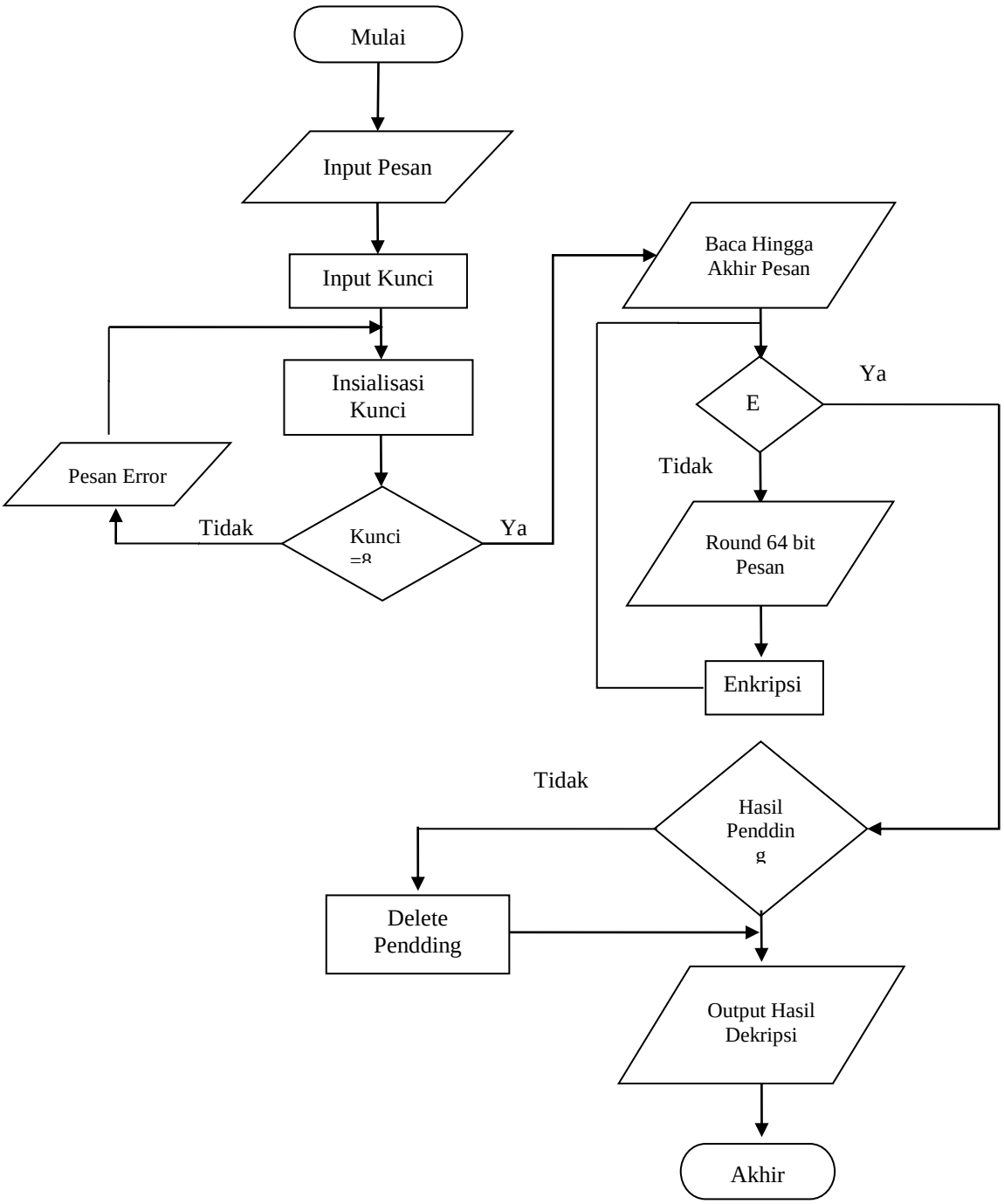
Gambar III.15. Flowchart Metode AES Enkripsi



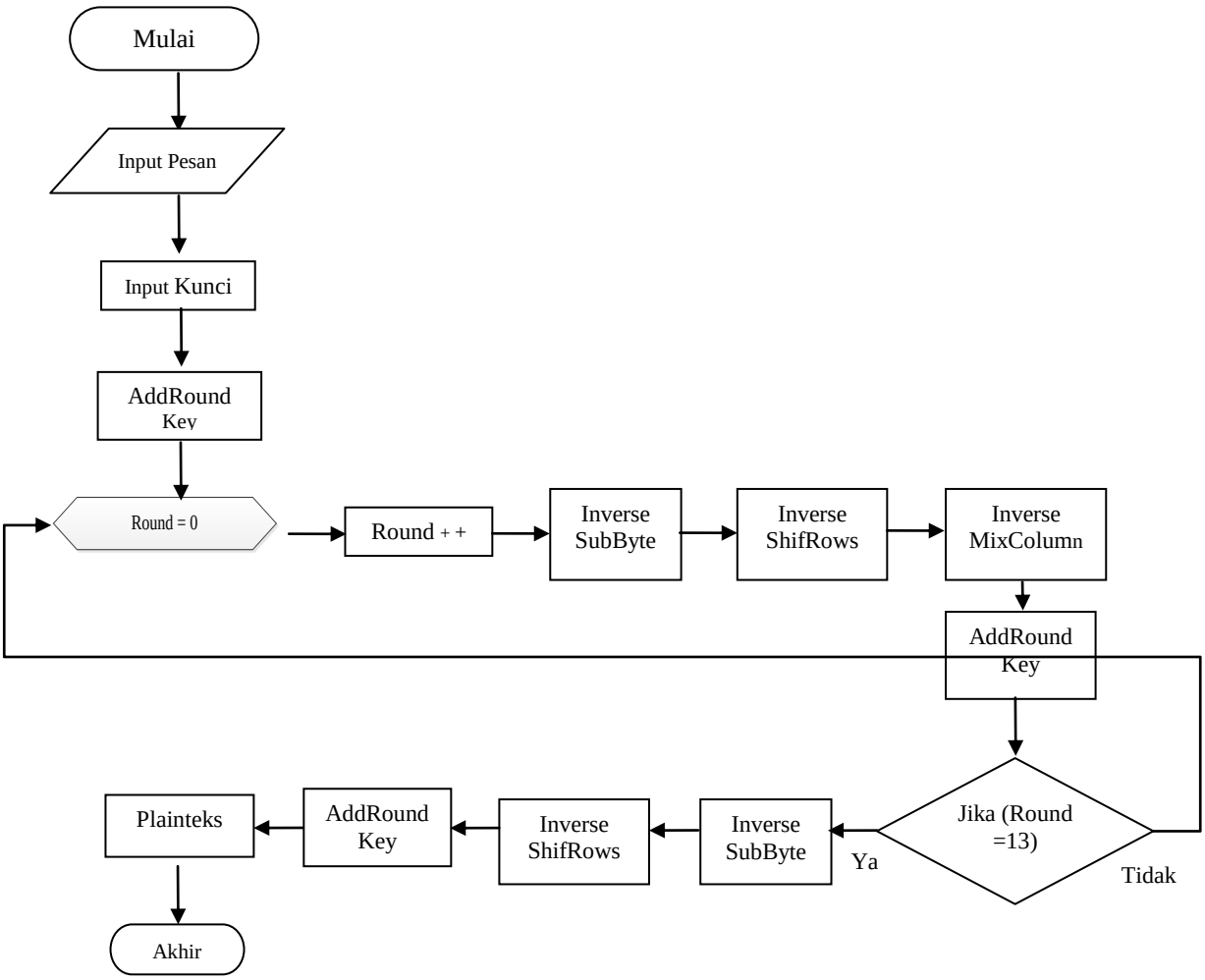
Gambar III.16. Flowchart Aplikasi SMS Enkripsi

III.4.3.2. *Flowchart* Dekripsi

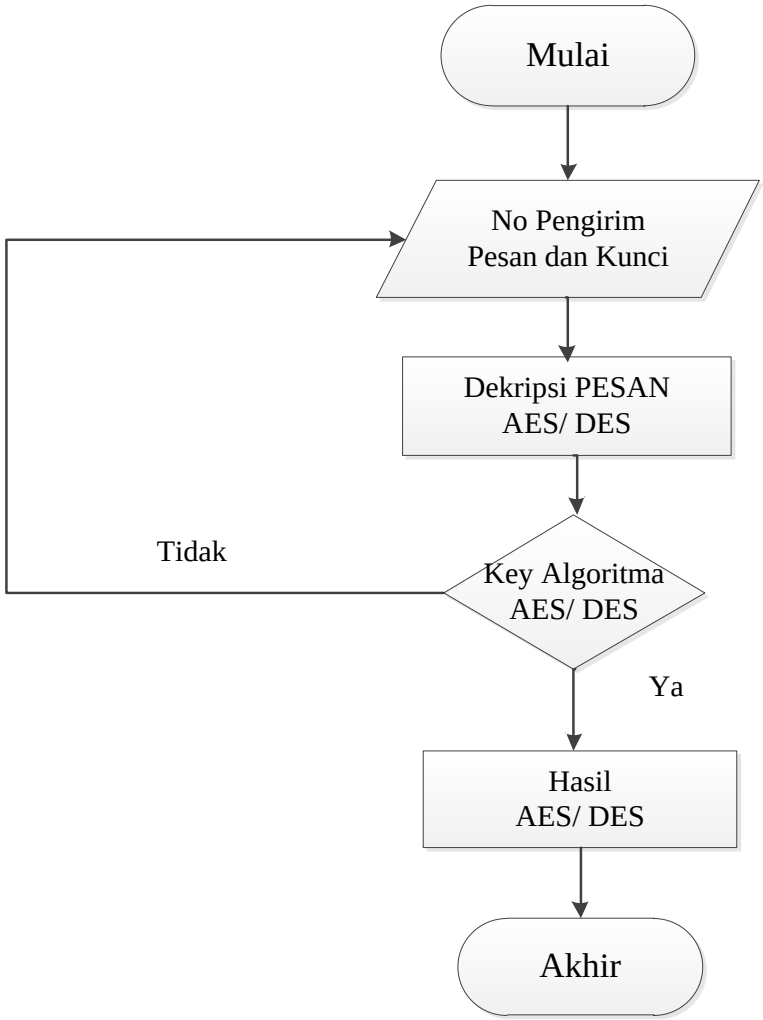
Flowchart ini dibuat untuk menjelaskan proses jalannya dekripsi pada program, seperti pada gambar III.17 berikut ini.



Gambar III.17. Flowchart Metode DES Dekripsi



Gambar III.18. Flowchart Metode AES Dekripsi



Gambar III.19. Flowchart Aplikasi SMS Dekripsi

Contoh :

1. Mengubah Plaintext dan key Menjadi bilangan biner.

Contoh:

Plaintext(x) = **activity**

Key(k) = **13 34 57 79 9B BC DF F1**

Ubahlah plaintext dalam bentuk biner

a = 01000001

c = 01000011

t = 01010100

i = 01001001

v = 01010110

i = 01001001

t = 01010100

y = 01011001

Ubah key dalam bentuk biner

13 = 00010011

34 = 00110100

57 = 01010111

79 = 01111001

9B= 10011011

BC= 10111101

DF= 11011111

F1 = 11110001

2. Setelah itu lakukan Initial Permutation (IP) pada bit plaintext menggunakan tabel IP berikut:

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	39	31	23	15	7

Hasil outputnya adalah sebagai berikut :

IP(x) : 11111111 11010100 01010100 10101011 00000000 00000000 10101000 10010010

Kemudian pecah bit pada IP(x) menjadi 2 bagian seperti berikut:

L_0 = 11111111 11010100 01010100 10101011

R_0 = 00000000 00000000 10101000 10010010

3. Langkah selanjutnya generate kunci yangdigunakan untuk mengenkripsi plaintext dengan menggunakan tabel Permutasi Kompresi (PC-1) dengan membuang 1 bit masing-masing block kunci dan 64 bit menjadi 58 bit.

Tabel PC-1

57	49	41	33	25	17	19
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	48	30	2
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Hasil outputnya adalah sebagai berikut :

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

Pecah CD(k) menjadi dua bagian sebagai berikut :

C_0 =1111000 0110011 0010101 0101111

D_0 =0101010 1011001 1001111 0001111

4. Lakukan pergeseran ke kiri (Left Shift pada) C_0, D_0 sebanyak 1 atau 2 kali berdasarkan urutan putaran sebagai berikut:

   Geser 1 bit ke kiri

   C_1 =1110000 1100110 0101010 1011111
   D_1 =1010101 0110011 0011110 0011110
   Geser 2 bit ke kiri

   C_2 =1100001 1001100 1010101 0111111
   D_2 =0101010 1100110 0111100 0111101
   Geser 2 bit ke kiri

   C_3 =1000011 0011001 0101010 1111111
   D_3 =0101011 0011001 1110001 1110101
   Geser 2 bit ke kiri

   C_4 =0001100 1100101 0101011 1111100
   D_4 =0101100 1100111 1000111 1010101
   Sampai pengeseran ke-16

   C_{15} = 1111100 0011001 1001010 1010111
   D_{15} = 1010101 0101100 1100111 1000111
   Geser 1 bit ke kiri

   C_{16} = 1111000 01100110 010101 0101111
   D_{16} = 0101010 1011001 1001111 0001111

5. Setiap hasil putaran digabungkan kembali menjadi C_i D_i dan di input kedalam tabel Permutation Compression 2(PC-2) untuk mengubah 56 bit menjadi 48 bit.

Tabel 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 sebagai berikut:

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

C_2 D_2 = 1100001 1001100 1010101 0111111 0101010 1100110 0111100 0111101

Sampai ke-16

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

C_{16} D_{16} = 1111000 01100110 010101 0101111 0101010 1011001 1001111 0001111

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

6. Setelah itu kita akan meng-Espansi data R_{i-1} 32 bit menjadi R_i 48 bit sebanyak 16 kali putaran dengan nilai perputaran 1<=i<16 dengan mengunakan tabel Ekspansi(E)

Tabel Espansi (E)

32	**1**	**2**	**3**	**4**	**5**
4	**5**	**6**	**7**	**8**	**9**
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**

7. Kemudian hasil E (R_{i-1}) kemudian di XOR kan dengan K dan menghasilkan vektor Metriks A. Berikut hasil Outputnya:

   Iterasi 1

   E (R_{i-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

   Sampai iterasi ke 16

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

   K16 = 110010 110011 110110 001011 000011 100001 011111 110101

   ----------XOR

   A16 = 011111 110011 110110 001011 000011 100001 011111 110101

8. Kemudian vektor A disubsitusikan kedelapan buah S-Box (Substitution S-Box), dimana blok pertama disubsitusikan dengan S_1 blok kedua dengan S_2 dan seterus dan seterusnya sehingga menghasilkan output vektor B_i 32 bit.

   B_1 = 1000 0101 0100 1000 0011 0010 1110 1010

   B_2 = 1101 1100 0100 0011 1000 0000 1111 1001

   Sampai ke-16

   B_{15} = 0100 0001 0011 1001 1111 0111 0010 0111

   B_{16} = 1000 0001 0110 1010 1111 0111 0100 1011

9. Setelah didapatkan nilai dari vektor B, langkah selanjutnya adalah memutasikan bit vektor B menggunakan tabel P-Box kemudian dikelompokkan menjadi 4 blok dimana tiap-tiap blok memiliki 32 bit data.

Tabel P-Box

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

Sehingga hasil yang didapat adalah sebagai berikut:

????

10. Kemudian P(B) di XOR kan dengan L_{i-1} untuk mendapatkan nilai R_i

L_0 = 11111111 11010100 01010100 10101011

R_0 = 00000000 00000000 10101000 10010010

P(B1) = 00101000 10110011 01000100 11010001
L(1)-1 = 11111111 11010100 01010100 10101011
----------XOR
R1 = 11010111 01100111 00010000 01111010

Sampai dengan 16

P(B15)= 10100101 00100110 11101100 11101100
L(15)-1= 11001011 11101000 01100110 10100001
----------XOR
R15 = 01101110 11001110 10001010 01001101
P(B16)= 00101001 11110111 01101000 11001100
L(16)-1= 10011100 00001000 11101101 11010010

----------XOR

R16 = 10110101 11111111 10000101 00011110

11. L16 dan R16 digabungkan kemudian dipermutasikan untuk terakhir kali dengan tabel Invers Initial Permutasi (IP^{-1}) seperti berikut:

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	50	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

Input : 10110101 11111111 10000101 00011110 01101110 11001110 10001010 01001101

Output : 01010110 10111001 11110111 10111011 01010001 11010100 10110110 01111000

Chiper (dalam Hex) : **56B9F7BB51D4B678**

proses enkripsi AES dengan Algoritma AES

Plaintext = 0 1 2 3 4 5 6 7 8 9 A B C D E F

Dalam HEX = 30 31 32 33 34 35 36 37 38 39 40 41 42 43 45 46

Kunci = A B C D E F G H I J K L M N O P

Dalam HEX = 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50

Sediakan 2 buah matrix 4 x 4 yang setiap kolom metriksnya mewakili 8 bit data input yang berurutan adalah sebagai berikut :

Plaintext HEX :

$$\begin{pmatrix} 30 & 34 & 38 & 43 \\ 31 & 35 & 39 & 44 \\ 32 & 36 & 41 & 45 \\ 33 & 37 & 42 & 46 \end{pmatrix}$$

Kunci HEX :

$$\begin{pmatrix} 41 & 45 & 49 & 4D \\ 42 & 46 & 4A & 4E \\ 43 & 47 & 4B & 4F \\ 44 & 48 & 4C & 50 \end{pmatrix}$$

1. Dilakukan proses XOR pada plaintext/state dengan roundkey. Proses XOR antar kolom kedua matriks yang bersesuaian setiap kolom menjadi biner. Contoh biner HEX 30 adalah 0011 0000, dan HEX 41 adalah 0100 0001. Jadi, 0011 0000 XOR 0100 0001 = 01110001 HEX 71. Demikian seterusnya proses XOR untuk setiap kolom yang bersesuaian. Langkah ini disebut AddRoundKey, yang menghasilkan matriks baru berikut ini :

$$\begin{pmatrix} 71 & 71 & 71 & 0E \\ 73 & 73 & 73 & 0A \\ 71 & 71 & 0A & 0A \\ 77 & 1F & 0E & 16 \end{pmatrix}$$

$$\begin{pmatrix} A3 & A3 & A3 & AB \\ 8F & 8F & 8F & 67 \end{pmatrix}$$

2. Pada matriks hasil XOR antara plainttext dengan roundkey dilakukan proses subsitusi dengan S-Box. Hasil subsitusinya sebagai berikut ini :

3. Pada hasil subsitusi dengan S-Box di lakukan proses ShiftRows. Prosesnya sangat sederhana dengan melakukan penggeseran secara wrapping (siklik) pada 3 baris terakhir array state. Junmlah penggeseran bergantung pada nilai baris (r). Baris r=1 digeser ejauh 1 byte hasilnya, baris r=2 digeser 2 byte, dan baris r=3 digeser sejauh 3 byte. Baris r=0 tidak digeser. Hasilnya beruopa matriks seperti berikut ini :

$$\begin{pmatrix} A3 & A3 & A3 & AB \\ 8F & 8F & 67 & 8F \\ 67 & 67 & A3 & A3 \\ 47 & F5 & C0 & AB \end{pmatrix}$$

4. Langkah selanjutnya setelah hasil ShiftRows didapat adalah melakukan MixCoulumns dengan cara mengalikan matriks hasil ShiftRows dengan matriks Rijndael. Transformasi ini dinyatakan sebagai perkalian matriks :

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} A3 \\ 8F \\ 67 \\ 47 \end{pmatrix} = \begin{pmatrix} 07 \\ 06 \\ 2B \\ FB \end{pmatrix}$$

semua state dilakukan dengan cara yang sama, sehingga didapatkan metriks hasil proses MixColumns sebagai berikut :

5. Langkah 1-4 dilakukan sampai putaran ke 9 namun untuk putaran 10 tidak dilakukan MixColumns tapi langsung dilakukan proses XOR hasil state ShiftRows dengan kunci terakhir.
6. Proses ekspansi kunci 128 bit . ambil 4 byte terakhir dari kunci yaitu 4D 4E AF 50, lalu geser byte terakhir, 4E 4F 50 4D. Subsitusikan dengan S-Box, hasilnya adalah 2F 84 53 E3. Dan langkah selanjutnya, XOR-kan dengan konstanta nilai tertentu dari pengguna.

   2F XOR 01 = 00101111 XOR 00000001 = 00101110=2E
   84 XOR 00 = 00000100 XOR 00000000 = 10000100=84
   53 XOR 00 = 01010011 XOR 00000000 = 01010011=53
   E3 XOR 00 = 11100011 XOR 00000000 = 11100011=E3
   Kemudian XOR kan 2E 84 53 E3 dengan 4 byte baris pertama kunci awal yaitu 41 42 43 44.
   2E XOR 41= 00101110 XOR 01000001 = 01101111 =6F
   84 XOR 42 = 10000100 XOR 01000010 = 11000110 =C6
   53 XOR 43 = 01010011 XOR 01000011 = 00010000 =10
   E3 XOR 44 = 11100011 XOR 01000100 = 10100111 =A7
   Hasil proses XOR diatas adalah 6F C6 10 A7 yang merupakan 4 byte pertama dari kunci yang baru untuk byte.

45 XOR 6F= 01000101 XOR 01101111 = 00101010 =6F

46 XOR C6 = 01000110 XOR 11000110 = 10000000=C6

47 XOR 10 = 01000111 XOR 00010000 = 01010111 =10

48 XOR A7 = 01001000 XOR 10100111 = 11101111 =A7

Demikian seterusnya hingga didapatkan 16 byte set kunci yang baru.

7. Tabel matriks enkripsi menggunakan metode AES dengan kunci 128 bit berikut ini :

Round	Mulai	Setelah	Setelah	Setelah	Nilai
Input	30 34 38 43 31 35 39 44 32 36 41 45 33 37 42 46				41 45 49 4D 42 46 4A 4E 43 47 4B 4F 44 48 4C 50
1	71 71 71 0E 73 73 73 0A 71 71 0A 0A 71 1F 0E 16	A3 A3 A3 AB 8F 8F 8F 67 A3 A3 67 67 F5 C0 AB 47	A3 A3 A3 AB 8F 8F 67 8F 67 67 A3 A3 47 F5 C0 AB	07 B5 C7 96 06 41 CC 2A 2B B1 91 69 FB E3 9A A8	6F 2A 63 2E C6 80 CA 8D 10 57 1C 53 A7 EF A3 F3
Sampai ke-10					
10	A0 19 DB 75 54 14 CF 99 D4 15 D8 80 58 E9 41 B4	E0 04 B9 9D 20 FA 8A EE 48 59 61 E7 6E 1E 83 8D	0E D4 B9 9D FA 8A EE 20 61 E7 48 59 8D 6A 1E 83		BA D3 43 85 2E CC F2 74 4E 56 34 75 ED 6B 3D 55

Untuk mendapatkan hasil chiper text dari keseluruhan round adalah sebagai berikut :

EO XOR BA = 11100000 10111010 = 01011010 = 5A
FA XOR 2E = 11111010 00101110 = 11010100 = D4
61 XOR 4E = 01100001 01001110 = 00101111 = 2F
8D XOR ED = 10001101 11101101 = 01100000 = 60

D4 XOR D3 = 11010100 11010011 = 00000111 = 07
8A XOR CC = 10001010 11001100 = 01000110 = 46
E7 XOR 56 = 11100111 01010110 = 10110001 = B1
6A XOR 6B = 01101010 01101011 = 00000001 = 01

B9 XOR 43 = 10111001 01000011= 11111010 = FA
EE XOR F2 = 11101110 11110010 = 00011100 = 1C
48 XOR 34 = 01001000 00110100 = 00101111 = 7C
1E XOR 3D = 00011110 00111101 = 01100000 = 2E

9D XOR 85 = 10011101 10000101 = 00010001 = 18
20 XOR 74 = 00100000 01110100 = 01010100 = 54
59 XOR 75 = 01011001 01110101 = 00101100 = 2C
83 XOR 55 = 10000011 01010101 = 11010110 = D6

Maka di dapatkan hasil chiphertext dari keseluruhan round adalah sebagai berikut:

5A D4 2F 60 07 46 B1 01 FA 1C 7C 2E 18 54 2C D6

III.5. Spesifikasi Perangkat

Dalam perancangan aplikasi untuk perangkat *Android Smartphone* ini penulis menggunakan beberapa perangkat agar aplikasi ini dapat berjalan lancar dan sesuai dengan yang diharapkan, yaitu sebagai berikut ini :

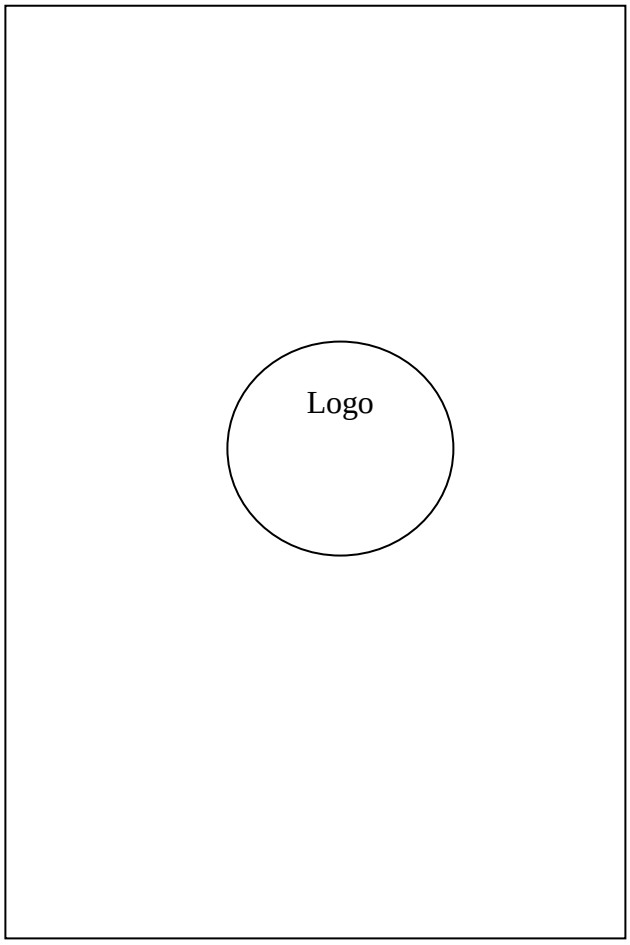
1. Perangkat Keras (*Hardware*)
   a. Prosesor AMD
   b. *Smartphone Android* dengan OS 4.2.1 atau diatasnya
   c. *Mouse, Keyboard* dan Monitor
2. Perangkat Lunak (*Software)*
   a. *Operating System,* OS yang dipergunakan dalam perancangan adalah *Windows* 7 dan untuk pengujian adalah OS Android pada perangkat *mobile.*
   b. *Eclipse ADT (Android Development Tools),* sebagai editor *source code Java.*
   c. JDK *Java 7.0,* sebagai bahasa program.

III.6. Desain Sistem

Dalam proses perancangan ini akan dijelaskan beberapa rancangan aplikasi yang akan dibangun yaitu sebagai berikut :

III.6.1. Rancangan Awal Pembukaan Program

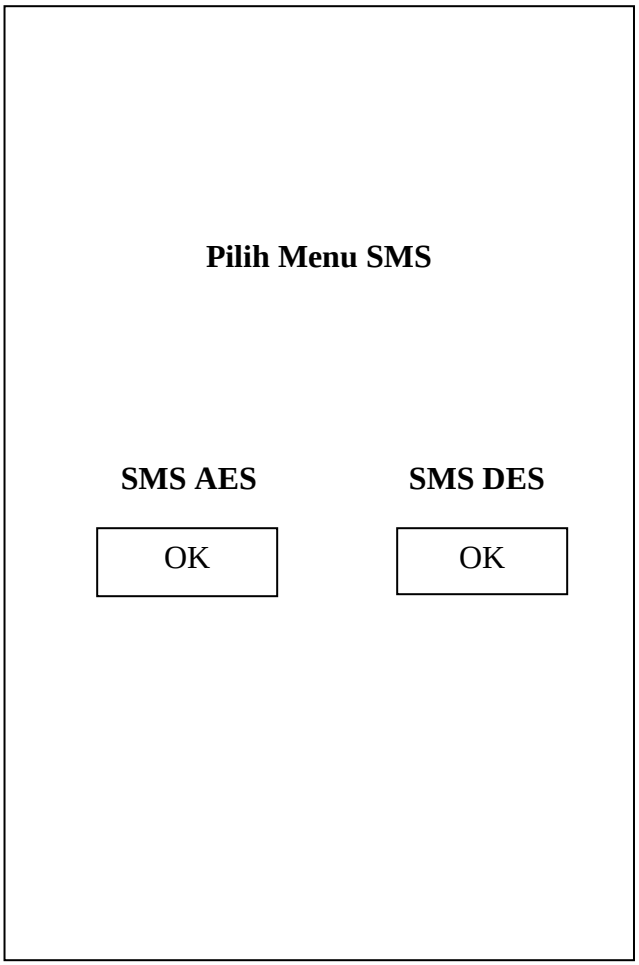
Gambar III.20 ini dibuat untuk menampilkan rancangan awal ketika program pertama kali dibuka.



Gambar III.20. Halaman Pembuka

III.6.2. Rancangan Menu Pilihan

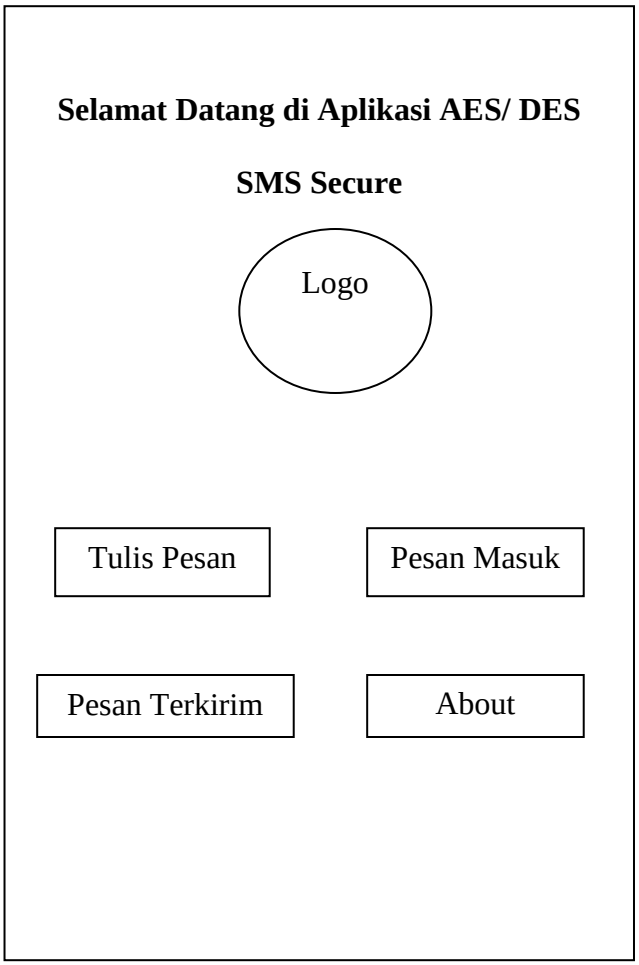
Berikut ini adalah rancangan form menu utama yang dapat lihat pada gambar III.21 berikut.



Gambar III.21 Form Pilihan

III.6.3. Rancangan Form Pesan AES/ DES

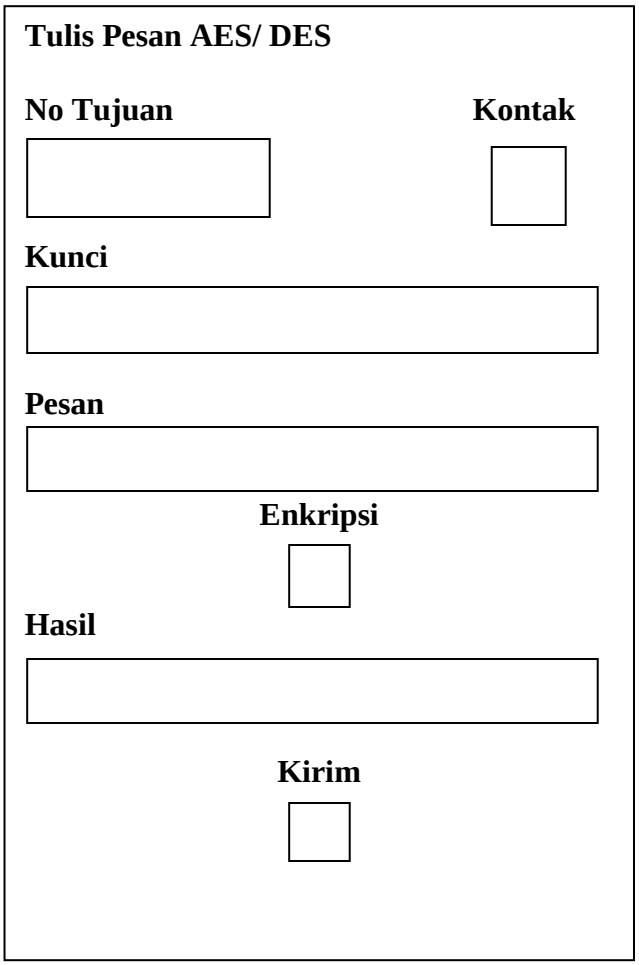
Berikut ini adalah rancangan form pesan aes yang dapat lihat pada gambar III.22 berikut.



Gambar III.22. Form Pesan AES/ DES

III.6.4. Rancangan Form Tulis Pesan

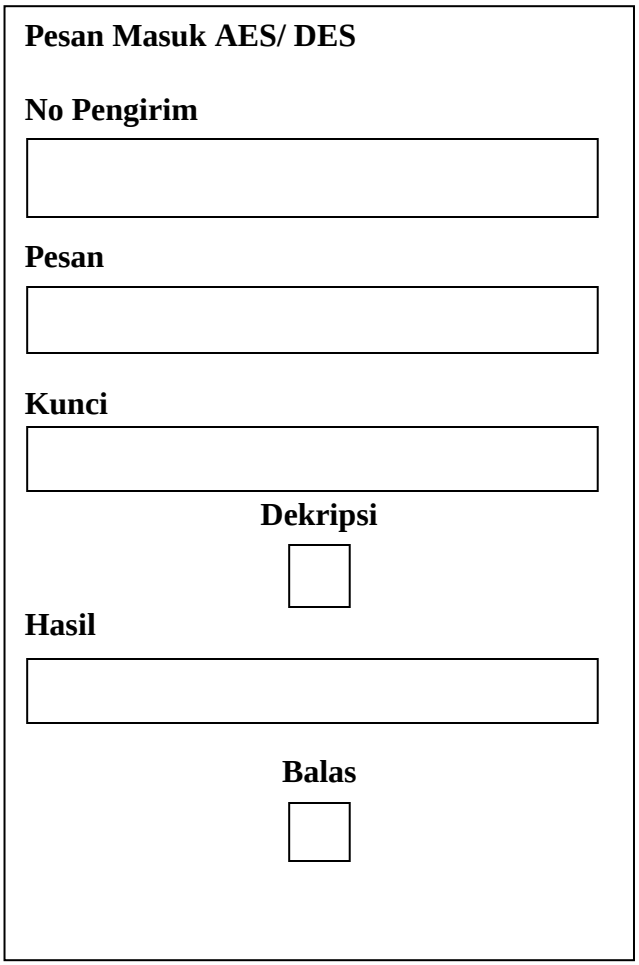
Berikut ini adalah rancangan form kotak masuk yang dapat lihat pada gambar III.23 berikut.



Gambar III.23. Form Tulis Pesan

III.6.5. Rancangan Form Kotak Masuk

Berikut ini adalah rancangan form kotak masuk yang dapat lihat pada gambar III.24 berikut.



Gambar III.24. Form Kotak Masuk