BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis dilakukan dengan tujuan untuk mengetahui dan mengevaluasi seluruh komponen yang berhubungan dengan sistem yang akan dibangun. Tahap analisis merupakan tahap sangat penting karena kesalahan pada tahap ini akan menyebabkan kesalahan pada tahap selanjutnya. Tahap analisis sistem dilakukan dengan cara menguraikan suatu sistem yang utuh kedalam bagian - bagian komponennya dengan tujuan untuk mengetahui dan mengevaluasi permasalahan - permasalahan yang ada sehingga ditemukan kelemahan dan keuntungan pada sistem tersebut.

3.1.1 Analisis Masalah

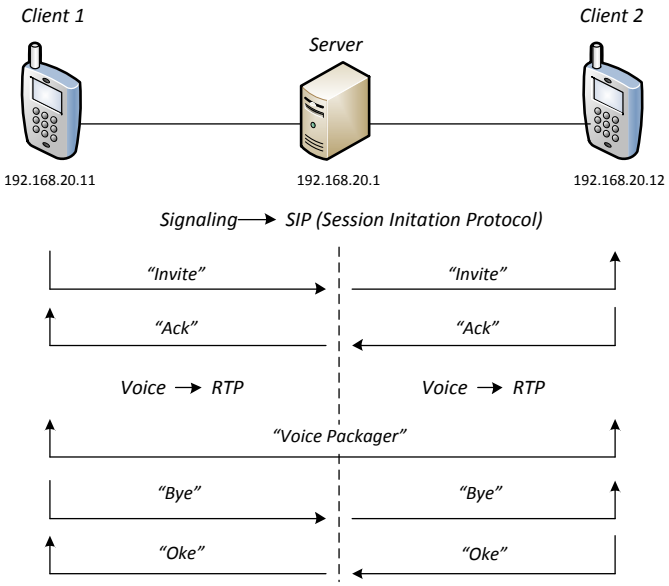
Voice Over Internet Protocol atau disingkat VoIP, dikenal juga dengan sebutan IP *Telephony*. VoIP didefinisikan sebagai suatu sistem yang menggunakan jaringan *internet* untuk mengirimkan data paket suara dari suatu tempat ke tempat yang lain menggunakan perantara protokol IP. Karena VoIP bekerja dengan memanfaatkan jaringan *internet*, maka tidak menutup kemungkinan bahwa VoIP memiliki kelemahan yaitu masih rendahnya tingkat keamanannya, jika dibandingkan dengan berkomunikasi secara langsung. Salah satu ancaman dan kerawanan pada VoIP yaitu terjadinya penyadapan oleh pihak – pihak yang tidak berwenang.

Berdasarkan masalah tersebut, diperlukannya algoritma enkripsi dengan menggunakan algoritma kriptografi AES dengan mode CTR dan metode Diffie-Hellman untuk proses pertukaran kuncinya. Agar sistem yang akan dibangun memberikan keamanan pada saat kedua *end user* melakukan komunikasi.

3.1.2 Analisis SIP Dan RTP

Komunikasi suara berlangsung pada *media transfer protocol*, tetapi sebelum komunikasi suara berlangsung, diperlukan sebuah *protocol* yang dinamakan *signaling protocol. signaling protocol* difungsikan untuk menangkap jaringan yang dituju sehingga dapat melakukan inisialisasi (penyampaian) pesan atau percakapan antar *end user.*

SIP merupakan *signaling protocol* yang tidak membawa paket data *audio* maupun *video,* sehingga dibutuhkan sebuah *media transfer protocol* untuk membawa dan mengirimkan paket data audio tersebut. Media transfer protocol yang digunakan adalah RTP *(Realtime Transfer Protocol).*



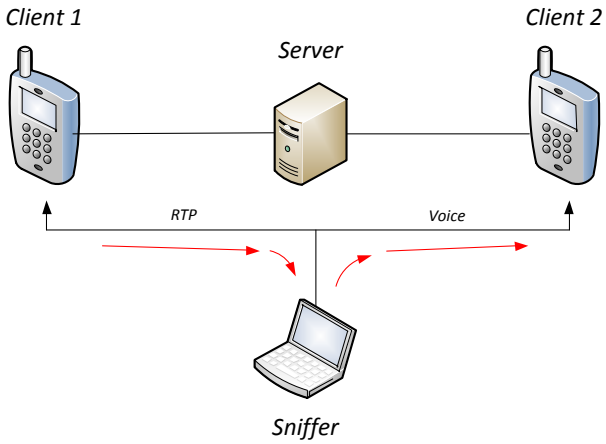
Gambar 3.1 *Callflow* SIP Dan RTP

Gambar 3.1 diatas menjelaskan bagaimana proses komunikasi suara dapat berlangsung antara *Client1* dan *Client2*, dimana *Client1* menggunakan IP Address 192.168.20.11 dan *Client2* menggunakan IP Address 192.168.20.12 sedangkan *server* menggunakan IP Address 192.168.20.1. *Client1* melakukan proses *invite* kepada *Client2*, proses *invite* tersebut tidak langsung masuk ke *Client2* melainkan diterima terlebih dahulu oleh *server*. Setelah itu baru dikirimkan ke *Client2* untuk bergabung dalam sesi komunikasi. Setelah *Client2* menerima pesan *invite*, *Client2* melakukan proses Ack yang artinya *Client2* mengkonfirmasi bahwa telah

menerima pesan *invite* yang dikirimkan oleh *Client1*. Setelah *Client2* melakukan proses Ack maka komunikasi dapat berlangsung melalui media *transfer protocol* yaitu RTP. Tetapi pada dasarnya komunikasi suara yang memanfaatkan IP memiliki kelemahan pada paket RTPnya yang dapat dimanipulasi oleh pihak ke tiga.

3.1.3 Analisis Metode Serangan

Salah satu metode penyerangan pada paket RTP adalah *Sniffing*. Penyerangan ini bertujuan untuk mengambil paket RTP dimana komunikasi antara kedua *end user* sedang berlangsung. Sehingga *sniffer* (orang yang melakukan proses *Sniffing* / penyadapan) akan mendapatkan hasil suara yang dilakukan oleh kedua *end user* tersebut.



Gambar 3.2 Proses *Sniffing*

Pada gambar 3.2 menjelaskan bahwa *Sniffer* sedang melakukan proses *sniffing* untuk mengetahui percakapan antara *client1* dan *client*2.

3.1.4 Analisis Algoritma Kriptografi

Algoritma kriptografi merupakan metode yang digunakan dalam kriptografi untuk mengubah data berupa *plaintext* menjadi *ciphertext* sehingga membuat data yang ingin dikirimkan menjadi rahasia dan hanya dapat diakses oleh pengguna – pengguna yang mengetahui kunci untuk mendekripsikan *ciphertext* tersebut. Contoh algoritma kriptografi yang dapat mengenkripsi pesan suara adalah

Algoritma AES, Algoritma DES dan Algoritma TDES. Berikut ini adalah penjabaran serta perbandingan dari masing-masing algoritma dapat dilihat pada tabel 3.1

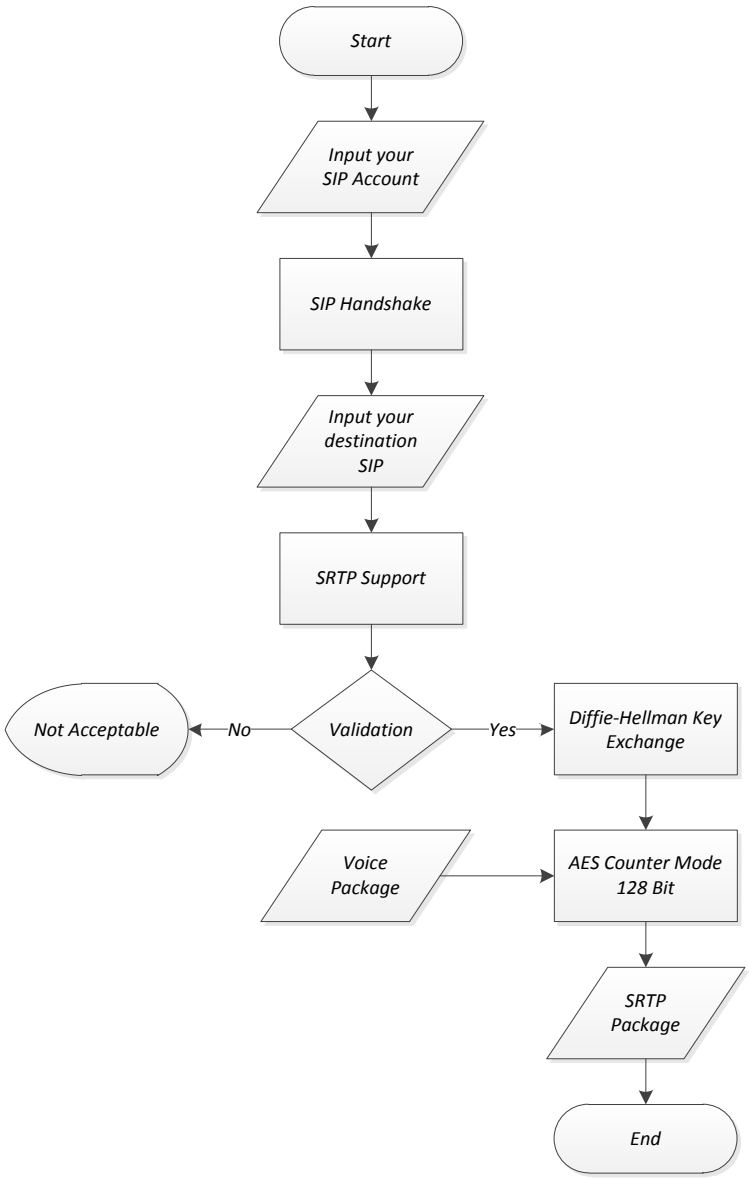
Tabel 3.1 Perbandingan Algoritma Kriptografi

Faktor	DES	TDES	AES
Dibuat oleh	IBM tahun 1975	IBM tahun 1978	Vincent Rijmen, Joan Daemen tahun 2001
Panjang kunci	56 bits	168 bits (untuk kunci ke 1, 2 dan 3) 112 bits (untuk kunci ke 1 dan 2)	128 bits, 192 bits, 256 its
Putaran	16	48	10 – 128 *bit key*, 12 – 192 *bit key*, 14 – 256 *bit key*
Ukuran *block*	64 bits	64 bits	128 bits
Tipe algoritma	Simetrik	Simetrik	Simetrik
Kecepatan	Lambat	Sangat lambat	Cepat
Keamanan	Tidak cukup aman	Cukup aman	Sangat aman

Berdasarkan tabel 3.1 hasil dari penelitian Gurpreet Singh dan Supriya [19] maka dapat ditarik kesimpulan bahwa algoritma AES ini baik untuk mengenkripsi dan mendekripsi pesan suara dikarenakan AES memiliki panjang kunci sampai dengan 256 bits dengan hanya 14 kali putaran sehingga proses enkripsi dan dekripsi menjadi lebih cepat dan sangat aman.

3.1.5 Analisis SRTP

SRTP *(Secure Realtime Tranport Protocol)* merupakan metode pengaman pada RTP. SRTP memiliki sistem keamanan *default* berupa AES *integer counter mode* dengan panjang bit 128bit. Disamping itu AES juga memiliki *mode null cipher* yang artinya SRTP berjalan tanpa adanya keamanan. Sedangkan untuk pertukaran kuncinya menggunakan Diffie-Hellman *Key Exchange.* Berikut ini adalah alur kerja SRTP dengan AES *integer counter mode.*



Gambar 3.3 Alur SRTP

Berdasarkan Gambar 3.3 Pengguna memasukan akun SIP, kemudian *server* melakukan proses SIP *handshake* yang berfungsi untuk melakukan autentikasi antara *client* dengan *server*. Setelah autentikasi sukses pengguna memasukan alamat SIP yang dituju. Kemudian *server* melakukan pengecekan apakah *server* memiliki SRTP dan kedua pengguna mengaktifkan SRTP. Kemudian *server* melakukan proses pertukaran kunci Diffie–Hellman untuk mendapatkan *master key*. Setelah *master key* didapatkan, *master key* akan digunakan sebagai kunci untuk

mengenkripsi suara yang masuk pada AES-CTR sehingga menghasilkan paket suara yang terenkripsi.

3.1.5.1 Analisis Algoritma AES-CTR

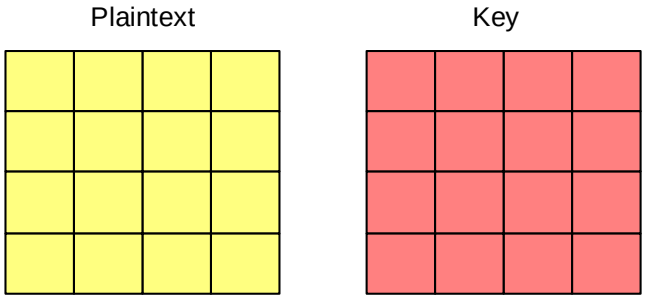
AES ini merupakan algoritma *block cipher* dengan menggunakan sistem permutasi dan substitusi. Jenis AES terbagi 3, yaitu : AES-128, AES-192, dan AES-256. Pengelompokkan jenis AES ini adalah berdasarkan panjang kunci yang digunakan. Selain itu, hal yang membedakan dari masing – masing AES ini adalah banyaknya *round* yang dipakai. AES-128 menggunakan 10 *round*, AES-192 sebanyak 12 *round*, dan AES-256 sebanyak 14 *round*.

AES memiliki ukuran *block* yang tetap sepanjang 128 bit dan ukuran kunci sepanjang 128, 192, atau 256 bit. Berdasarkan ukuran *block* yang tetap, AES bekerja pada matriks berukuran 4x4 dimana tiap – tiap sel matriks terdiri atas 1 *byte* (8 bit). Berikut adalah penjabaran cara proses enkripsi dan dekrisi:

3.1.5.1.1 Proses Enkripsi

Langkah – langkah yang dilakukan pada saat proses enkripsi pada Algoritma AES-128 adalah sebagai berikut:

1. Siapkan 2 buah *array* berukuran 4x4 bernama *Plaintext* dan *Key*



Gambar 3.4 *State* dan *Subkey*

2. Pilih *Plaintext* yang akan di enkripsi beserta kuncinya, contoh *plaintext* : "**helloapakabarnya**" dan *key* : "**kabarakubaiksaja**" dari *Plaintext* dan *key* tersebut Konversikan ke dalam bentuk bit menggunakan kode ASCII. Lalu Konversikan kode ASCII tersebut ke dalam heksadesimal. Kemudian

Masukkan tiap – tiap bagian teks tersebut ke dalam tiap – tiap sel pada matriks berukuran 4x4. Sehingga hasilnya sebagai berikut:

Plaintext

68	6f	6b	72
65	61	61	6e
6c	70	62	79
6c	61	61	61

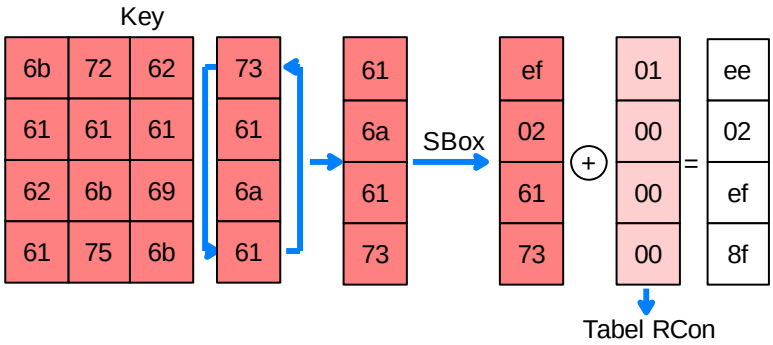
Key

6b	72	62	73
61	61	61	61
62	6b	69	6a
61	75	6b	61

Gambar 3.5 *State* dan *Subkey*

3. Sebelum melakukan langkah selanjutnya terlebih dahulu membuat *key schedule* yang di XOR kan dengan nilai yang ada pada table RCon, yang hasilnya disebut *Roundkey* pada ditahapan enkripsi. Untuk menghasilkan *Roundkey* 1 sampai *Roundkey* 10 tahapannya adalah sebagai berikut:

   a. Tahap1

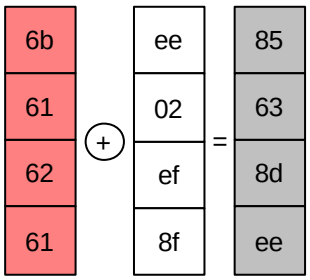


Gambar 3.6 Tahap *Roundkey*

Pada Gambar 3.7 Pada *key* di kolom terakhir, *byte* yang diatas tukarkan dengan *byte* yang dibawah, setelah ditukar lakukan langkah *SubByte* dengan melihat tabel SBox. Hasil dari SBox di XOR kan dengan kolom1 yang ada di tabel Rcon. Tabel SBox dan tabel RCon dapat dilihat di bab sebelumnya.

b. Tahap2

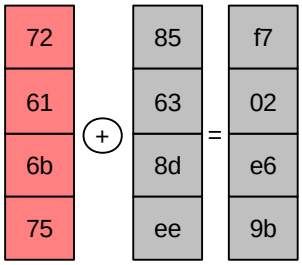
Ambil kolom pertama pada *key* lalu XOR kan dengan hasil dari tahap1.



Gambar 3.7 Tahap *Roundkey* Kolom1

c. Tahap3

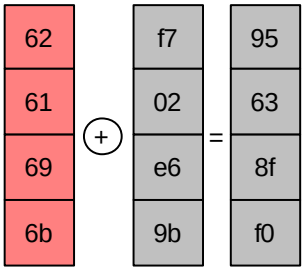
Ambil kolom kedua pada *key* lalu XOR kan dengan hasil dari tahap2.



Gambar 3.8 Tahap *Roundkey* Kolom2

d. Tahap4

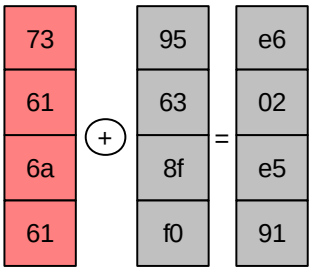
Ambil kolom ketiga pada *key* lalu XOR kan dengan hasil dari tahap3.



Gambar 3.9 Tahap *Roundkey* Kolom3

e. Tahap5

Ambil kolom keempat pada *key* lalu XOR kan dengan hasil dari tahap4.



Gambar 3.10 Tahap *Roundkey* Kolom 4

f. Tahap 6

Setelah melakukan tahap 1 sampai tahap 5, didapatkan *roundkey* 1. Hasilnya dapat dilihat pada gambar 3.11

85	f7	95	e6
63	02	63	02
8d	e6	8f	e5
ee	9b	f0	91

Gambar 3.11 Hasil *Roundkey* 1

g. Tahap 7

Lakukan tahap 1 – tahap 6, untuk menghasilkan *Roundkey* selanjutnya. Untuk *roundkey* selanjutnya dapat dilihat pada gambar 3.12

Roundkey 2

f0	07	92	74
ba	b6	db	d9
0c	ea	65	80
60	fb	0b	9a

Roundkey 3

c1	c6	54	29
77	cf	14	cd
b4	5e	3b	bb
f2	09	02	98

Roundkey 4

74	b2	e6	c6
9d	52	46	8b
f2	ac	97	2c
45	4c	4e	d6

Roundkey 5

59	eb	0d	cb
ec	be	f8	73
04	a8	3f	13
f1	bd	f3	25

Roundkey 6

f6	1d	10	db
91	2f	d7	a4
3b	93	ac	bf
ee	53	a0	85

Roundkey 7

ff	e2	f2	29
99	b6	61	c5
ac	3f	93	2c
57	04	a4	21

Roundkey 8

d9	3b	c9	e0
e8	5e	3f	fa
51	6e	fd	d1
f2	f6	52	73

Roundkey 9

ef	d4	1d	fd
d6	88	b7	4d
de	b0	4d	9c
13	e5	b7	c4

Roundkey 10

3a	08	c2	47
ee	80	72	a3
f3	37	3f	15
0e	7a	a3	d1

Gambar 3.12 Hasil *Roundkey* Selanjutnya

4. Lakukan langkah *AddRoundKey* (*Initial State*)

   Pada proses ini *key* digabungkan dengan *Plaintext*. Proses penggabungan ini menggunakan operasi XOR untuk setiap *byte* dari *key* dengan setiap *byte* dari *plaintext*. Sehingga hasilnya sebagai berikut:

Plaintext

68	6f	6b	72
65	61	61	6e
6c	70	62	79
6c	61	61	61

+

Key

6b	72	62	73
61	61	61	61
62	6b	69	6a
61	75	6b	61

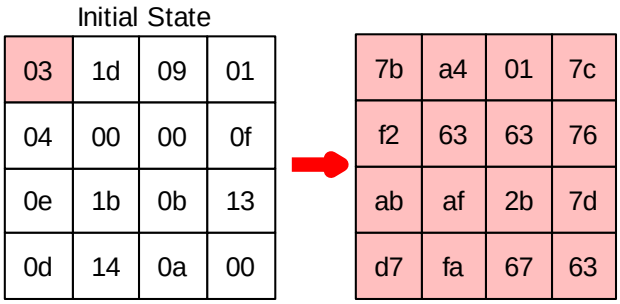
=

03	1d	09	01
04	00	00	0f
0e	1b	0b	13
0d	14	0a	00

Gambar 3.13 Proses *Addroundkey* (*Initial State*)

5. Lakukan langkah *SubBytes*

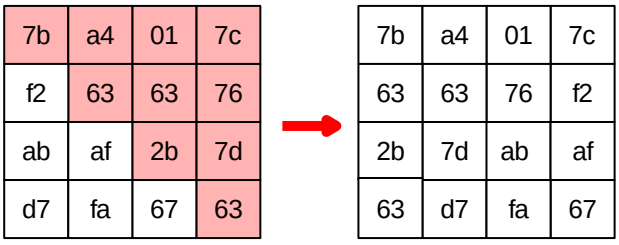
   Proses *SubBytes* adalah operasi yang akan melakukan substitusi tidak linear dengan cara mengganti setiap *byte* dari *initial state* hasil langkah *addroundkey* (*initial state*) dengan *byte* pada sebuah tabel yang SBox. Misal, kita akan menggantikan elemen 03 pada *initial state*. Maka elemen yang bersesuaian pada SBox terletak pada baris ke-0 dan kolom ke-3 yaitu hasilnya 7b. kemudian lakukan pada *byte – byte* selanjutnya. Hasilnya dapat dilihat pada gambar 3.14



Gambar 3.14 Hasil Sub*byte*

6. Lakukan langkah *ShiftRows*

   Proses *ShiftRows* akan beroperasi pada tiap baris dari tabel *plaintext*. Proses ini akan bekerja dengan cara memutar *byte – byte* pada 3 baris terakhir (baris 1, 2, dan 3) dengan jumlah perputaran yang berbeda – beda. Baris 1 akan diputar sebanyak 1 kali, baris 2 akan diputar sebanyak 2 kali, dan baris 3 akan diputar sebanyak 3 kali. Sedangkan baris 0 tidak akan diputar. Hasiilnya dapat dilihat pada gambar 3.15



Gambar 3.15 Hasil ShiftRows

7. Lakukan langkah *MixColumns*

Proses *MixColumns* dapat dilakukan dengan mengalikan koordinat vector dari empat nomor pada Rijndael's Galois field menggunakan operasi perkalian matriks dengan rumus dibawah ini:

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Dimana r0, r1, r2 dan r3 adalah hasil setelah transformasi. a0 - a3 dapat diperoleh dari matriks setelah data mengalami proses substitusi dalam Sbox. Untuk memperoleh nilai r0, rumusnya seperti dibawah ini:

$$r0 = \{02.a0\}xor\{03.a1\}xor\{01.a2\}xor\{01.a3\}$$
$$r1 = \{01.a0\}xor\{02.a1\}xor\{03.a2\}xor\{01.a3\}$$
$$r2 = \{01.a0\}xor\{01.a1\}xor\{02.a2\}xor\{03.a3\}$$
$$r3 = \{03.a0\}xor\{01.a1\}xor\{01.a2\}xor\{03.a3\}$$

Untuk melakukan proses *MixColumns* ambil state dari hasil langkah sebelumnya, langkah – langkahnya adalah sebagai berikut:

a. Kolom 1

1. Langkah 1

02.7b = 0111 1011

03.63 = 0110 0011 = 1100 0110 XOR 0110 0011 = 1010 0101

01.2b = 0010 1011

01.63 = 0110 0011

Hitung:

$r0 = \{02.a0\}xor\{03.a1\}xor\{01.a2\}xor\{01.a3\}$

$r0 = \{0111\ 1011\}xor\{1010\ 0101\}xor\{0010\ 1011\}xor\{01100011\}$

$r0 = 0001\ 1011$

$r0 = 1b$

2. Langkah 2

01.7b = 0111 1011

02.63 = 0110 0011 = 1100 0110

03.2b = 0010 1011 = 0101 0110 XOR 0010 1011 = 0111 1101

01.63 = 0110 0011

Hitung:

$r1 = \{01.a0\} xor \{02.a1\} xor \{03.a2\} xor \{01.a3\}$

$r1 = \{0111\ 1011\} xor \{1100\ 0110\} xor \{0111\ 1101\} xor \{01100011\}$

$r1 = 1010\ 0011$

$r1 = a3$

3. Langkah 3

01.7b = 0111 1011

01.63 = 0110 0011

02.2b = 0010 1011 = 0101 0110

03.63 = 0110 0011 = 1100 0110 XOR 0110 0011 = 1010 0101

Hitung:

$r2 = \{01.a0\} xor \{01.a1\} xor \{02.a2\} xor \{03.a3\}$

$r2 = \{0111\ 1011\} xor \{01100011\} xor \{0101\ 0110\} xor \{1010\ 0101\}$

$r2 = 1110\ 1011$

$r2 = eb$

4. Langkah 3

03.7b = 0111 1011 = 1111 0110 x0r 0111 1011 = 1000 1101

01.63 = 0110 0011

01.2b = 0010 1011

02.63 = 0110 0011 = 1100 0110

Hitung:

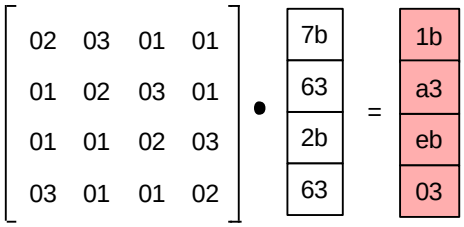
$r3 = \{03.a0\} xor \{01.a1\} xor \{01.a2\} xor \{03.a3\}$

$r3 = \{1000\ 1101\} xor \{0110\ 0011\} xor \{0010\ 1011\} xor \{1100\ 0110\}$

$r3 = 0000\ 0011$

$r3 = 03$

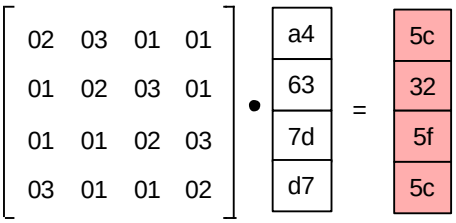
Hasilnya dapat dilihat pada gambar dibawah ini:



Gambar 3.16 Kolom1

b. Kolom2

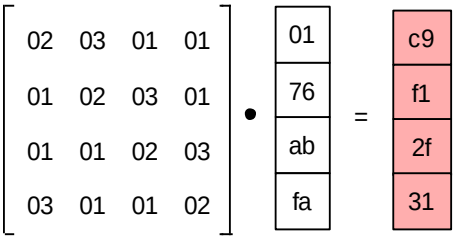
Lakukan seperti langkah pada kolom 1, sehingga hasilnya dapat dilihat pada gambar dibawah ini:



Gambar 3.17 Kolom2

c. Kolom 3 dari table state

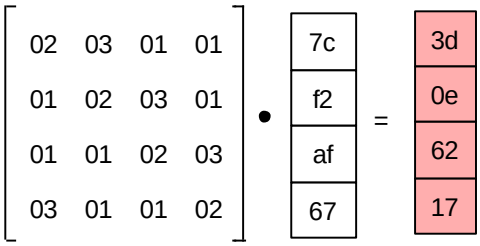
Lakukan seperti langkah pada kolom 1, sehingga hasilnya dapat dilihat pada gambar dibawah ini:



Gambar 3.18 Kolom3

d. Kolom 4 dari table state

Lakukan seperti langkah pada kolom 1, sehingga hasilnya dapat dilihat pada gambar dibawah ini:



Gambar 3.19 Kolom4

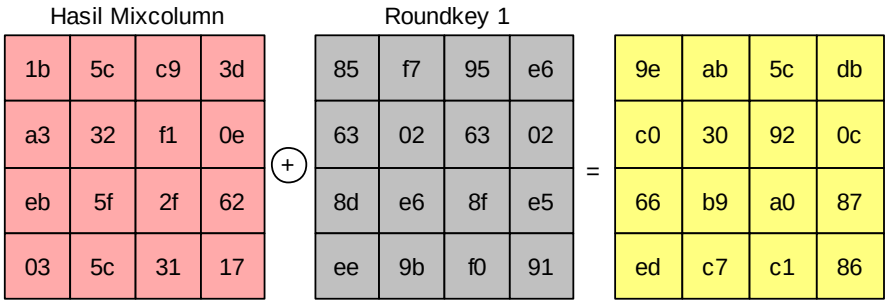
Sehingga hasil akhir yang di peroleh dari perhitungan *Mixcolumns* dapat dilihat pada gambar dibawah ini:

1b	5c	c9	3d
a3	32	f1	0e
ab	5f	2f	62
03	5c	31	17

Gambar 3.20 Hasil Akhir Dari Perhitungan MixColumn

8. Lakukan langkah *AddRoundKey* (*Round* 1)

   Pada proses ini *key* digabungkan dengan *plaintext* (hasil dari perhitungan *MixColumns*). *Key* pada proses *addroundkey* ini berbeda dengan *addroundkey* pada langkah 4 yaitu menggunakan *key schedule* (*round* 1) yang telah dibuat sebelumnya yaitu pada langkah 3. untuk proses penggabungannya sama seperti pada langkah 4. Sehingga hasilnya adalah sebagai berikut:



Gambar 3.21 Hasil *Addroundkey*

9. Ulangi langkah *SubBytes*, *ShiftRows*, *MixColumns*, dan *Addrounkey* sebanyak 9 kali untuk mendapatkan hasil *Round* 2 sampai *Round* 9. sedangkan untuk mendapatkan hasil untuk *Round* 10 ulangi langkah *SubBytes*, ShiftRows, dan

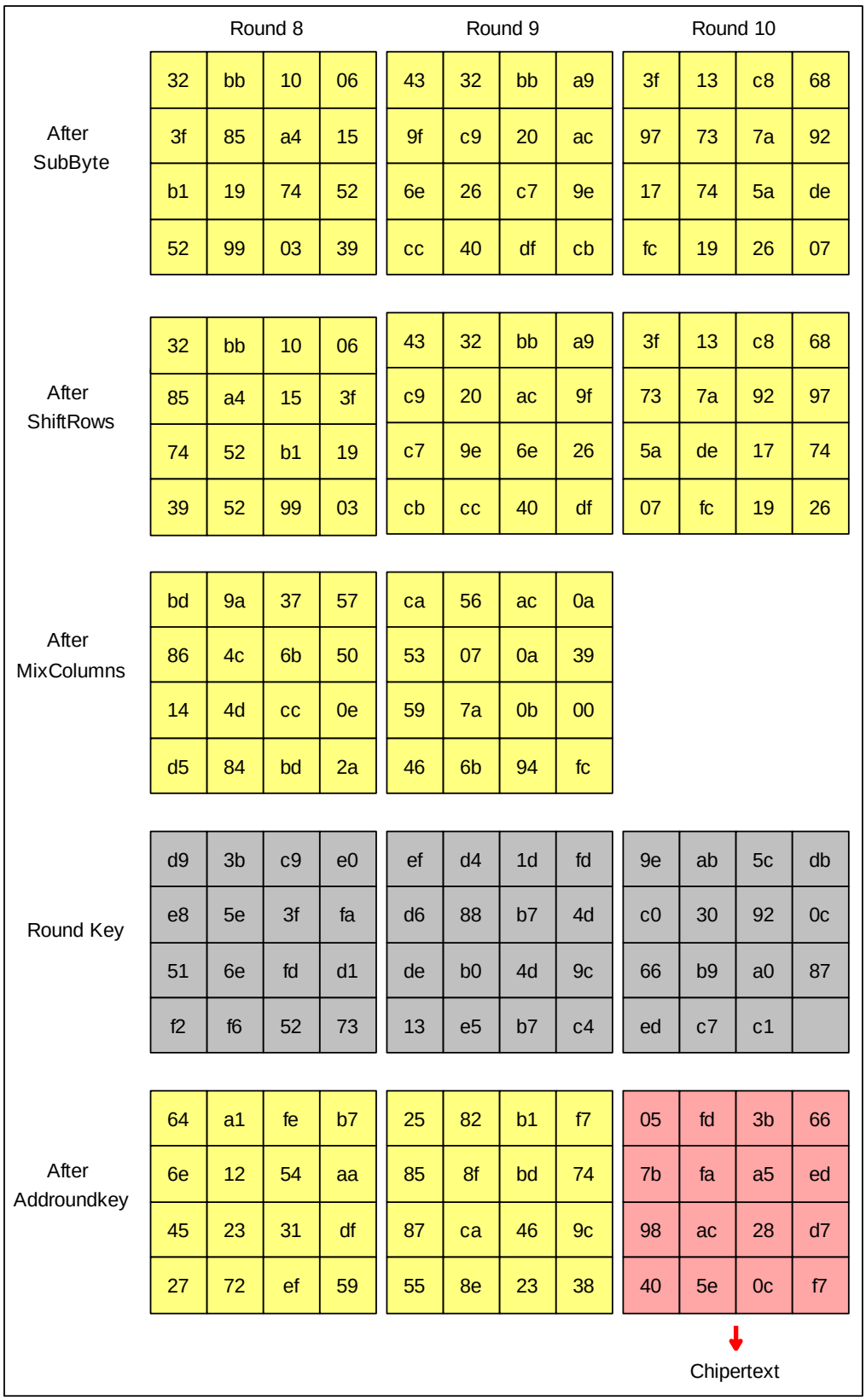
Addrounkey (*final state*). Hasilnya dapat dilihat pada gambar 3.22 sampai gambar 3.24

	Round 2				Round 3				Round 4			
After SubByte	0b	62	4a	b9	2f	53	87	8e	b0	8b	3f	48
	ba	04	4f	fe	b4	34	d9	5d	30	ed	6e	7e
	33	56	e0	17	fa	47	8e	5c	d0	a1	97	b8
	55	c6	78	44	82	79	73	b1	09	33	a6	f2
After ShiftRows	0b	62	4a	b9	2f	53	87	8e	b0	8b	3f	48
	04	4f	fe	ba	34	d9	5d	b4	ed	6e	7e	30
	e0	17	33	56	8e	5c	fa	47	97	b8	d0	a1
	44	55	c6	78	b1	82	79	73	f2	09	33	a6
After MixColumns	be	57	78	92	3d	08	71	f4	32	0e	1f	c7
	7c	90	3e	54	7f	9c	51	47	21	8d	9b	76
	18	fc	83	27	d4	af	be	21	65	95	af	d0
	71	54	84	cc	b2	6f	c7	9c	4e	42	89	1e
Round Key	f0	07	92	74	c1	c6	54	20	74	b2	e6	c6
	ba	b8	db	d9	77	cf	14	cd	9d	52	46	8b
	0c	ea	65	80	b4	5e	3b	bb	f2	ac	97	2c
	60	fb	0b	9a	f2	09	02	98	45	4c	4e	d6
After Addroundkey	4e	50	ea	e6	fc	ce	25	d4	46	bc	f9	01
	c6	28	e5	8d	08	53	45	8a	bc	df	dd	fd
	14	16	e6	a7	60	f1	85	9a	97	39	38	fc
	11	af	8f	56	40	66	c5	04	0b	0e	c7	c8

Gambar 3.22 Hasil Proses Enkripsi

	Round 5				Round 6				Round 7			
After SubByte	5a	65	99	7c	ea	98	0f	52	18	fa	50	58
	65	9e	c1	54	51	3a	f8	96	0e	b5	44	52
	88	12	07	b0	55	67	c0	d2	28	40	b2	4e
	2b	ab	c6	e8	e3	96	b5	83	71	f0	c1	18
After ShiftRows	5a	65	99	7c	ea	98	0f	52	18	fa	50	58
	9e	c1	54	65	3a	f8	96	51	b5	44	52	0e
	07	b0	88	12	c0	d2	55	67	b2	4e	28	40
	e8	2b	ab	c6	83	e3	96	b5	18	71	f0	6e
After MixColumns	e2	09	f6	83	c2	09	7c	85	5e	1c	8e	8c
	9c	1c	19	46	46	fd	51	ec	bc	d1	7c	ea
	e9	a2	20	6c	d5	e1	92	09	fa	b1	59	64
	bc	88	21	64	c2	44	e5	bf	1f	fd	71	7a
Round Key	59	eb	0d	cb	f6	1d	10	db	ff	e2	f2	29
	ec	be	f8	73	91	2f	d7	a4	99	b6	61	c5
	04	a8	3f	13	3b	93	ac	bf	ac	3f	93	2c
	f1	bd	f3	25	ee	53	a0	85	57	04	a4	21
After Addroundkey	bb	e2	fb	48	34	14	6c	5e	a1	fe	7c	a5
	70	a2	e1	35	d7	d2	86	48	25	67	1d	2f
	ed	0a	1f	7f	ee	72	3e	b6	56	8e	ca	48
	4d	35	d2	41	2c	17	45	34	48	f9	d5	5b

Gambar 3.23 Hasil Proses Enkripsi



Gambar 3.24 Hasil Proses enkripsi

10. *Ciphertext* yang dihasilkan dari langkah – langkah sebelumnya dapat dilihat pada gambar 3.25

05	fd	3b	66
7b	fa	a5	ed
98	ac	28	d7
40	5e	0c	f7

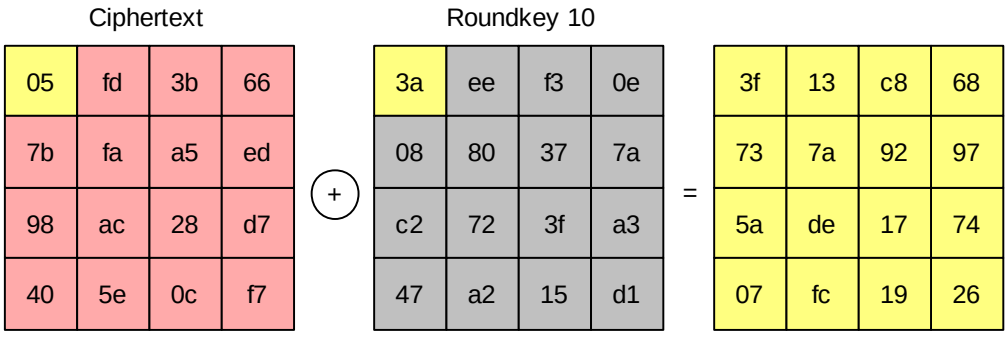
Gambar 3.25 *Ciphertext*

3.1.5.1.2 Proses Dekripsi

Proses dekripsi pada Algoritma AES-128 langkah – langkahnya adalah sebagai berikut:

1. Lakukan langkah *AddRoundKey*

   Langkah *AddRoundKey* pada proses dekripsi ini *ciphertext* digabungkan dengan *Roundkey* 10 (dari *key schedule* di proses enkripsi). Proses penggabungan ini menggunakan operasi XOR caranya sama seperti pada saat proses *Addroundkey* diproses enkripsi. Sehingga hasilnya sebagai berikut:

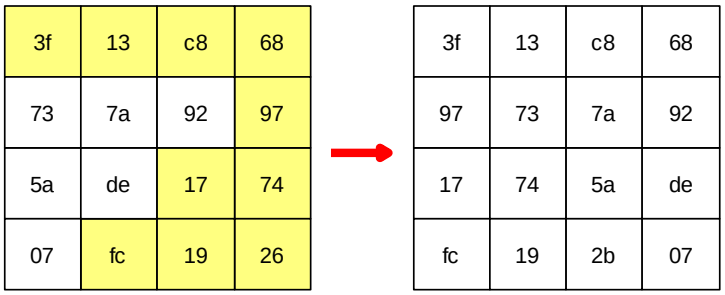


Gambar 3.26 Hasil *Addroundkey*

2. Lakukan langkah *InvShiftRows*

   Proses *InShiftRows* pada proses dekripsi caranya sama seperti pada proses enkripsi. Proses ini akan bekerja dengan cara memutar *byte-byte* pada 3 baris terakhir (baris 1, 2, dan 3) dengan jumlah perputaran yang berbeda – beda. Baris

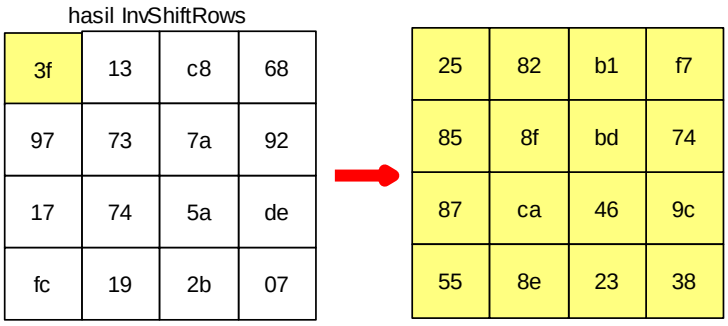
1 akan diputar sebanyak 1 kali, baris 2 akan diputar sebanyak 2 kali, dan baris 3 akan diputar sebanyak 3 kali. Sedangkan baris 0 tidak akan diputar.



Gambar 3.27 Hasil *InvShiftRows*

3. Lakukan langkah *InvSubByte*

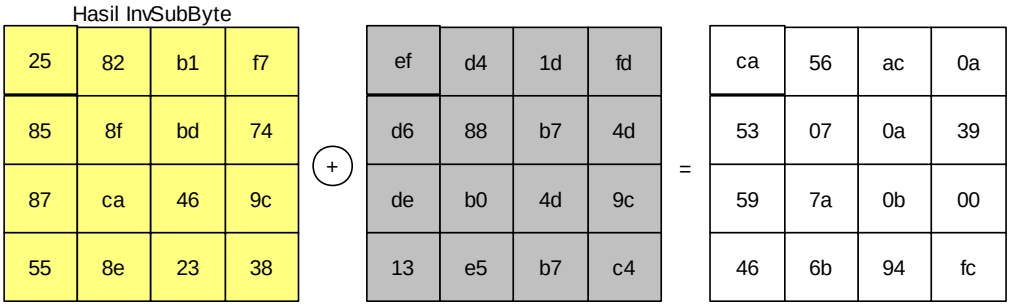
   Proses *InvSubBytes* adalah operasi yang akan melakukan substitusi tidak linear dengan cara mengganti setiap *byte* dari hasil *InvShiftRows* pada sebuah tabel InvSbox yang dapat dilihat dibab sebelumnya. Misal, kita akan menggantikan *byte* 31 pada hasil *InvShiftRows* yang diperoleh dari hasil sebelumnya. Maka elemen yang bersesuaian pada InvSbox terletak pada baris ke-3 dan kolom ke-1 yaitu hasilnya 25. kemudian lakukan pada *byte* – *byte* selanjutnya. Hasilnya dapat dilihat pada gambar 3.29



Gambar 3.28 Hasil *InvSubByte*

4. Lakukan langkah *Addrounkey*

   Langkah *AddRoundKey* pada proses dekripsi ini *ciphertext* digabungkan dengan *Roundkey* 9. Proses penggabungan ini menggunakan operasi XOR caranya sama seperti pada saat proses *Addroundkey* diproses enkripsi. Sehingga hasilnya sebagai berikut:



Gambar 3.29 Hasil *Addroundkey*

5. Lakukan langkah *InvMixColumns*

   Proses *InvMixColumns* akan beroperasi pada tiap kolom dari tabel state. Operasi ini menggabungkan 4 *byte*s dari setiap kolom tabel *state* dan menggunakan transformasi linier. Teori perhitungan *InvMixColumns* menggunakan operasi perkalian matriks dengan rumus dibawah ini:

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 14 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Atau

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

   Dimana r0, r1, r2 dan r3 adalah hasil setelah transformasi. a0 - a3 dapat diperoleh dari matriks setelah data mengalami proses substitusi dalam InvSbox. Untuk memperoleh nilai r0, rumusnya seperti dibawah ini:

$$r0 = \{0e.a0\}xor\{0b.a1\}xor\{0d.a2\}xor\{09.a3\}$$
$$r1 = \{09.a0\}xor\{0e.a1\}xor\{0b.a2\}xor\{0d.a3\}$$
$$r2 = \{0d.a0\}xor\{09.a1\}xor\{0e.a2\}xor\{0b.a3\}$$
$$r3 = \{0b.a0\}xor\{0d.a1\}xor\{09.a2\}xor\{0e.a3\}$$

   Untuk langkah *InvMixColumns* ini state yang diambil adalah state setelah langkah *Addroundkey*, proses perhitungannya adalah sebagai berikut:

a. Kolom 1

0e.ca = (0000 1110).(1100 1010)

= $\langle 3x + 2x + x \rangle . \langle 7x + 6x + 3x + x \rangle$

= $\langle 10x + 9x + 6x + 4x + 9x + 8x + 5x + 3x + 8x + 7x + 4x + 2x + \rangle$

= $\langle 10x + 7x + 6x + 5x + 3x + 2x \rangle \; mod \; \langle 8x + 4x + 3x + x + 1 \rangle$

= 10011101100 mod 100011011

= 00010000000

= 10000000

= 80

0b.53 = (0000 1011).(0101 0011)

= $\langle 3x + x + 1 \rangle . \langle 6x + 4x + x + 1 \rangle$

= $\langle 9x + 7x + 4x + 3x + 7x + 5x + 2x + x + 6x + 4x + x + 1 \rangle$

= $\langle 9x + 6x + 5x + 3x + 2x + 1 \rangle \; mod \; \langle 8x + 4x + 3x + x + 1 \rangle$

= 1001101101 mod 100011011

= 0001011011

= 01011011

= 5b

0d.59 = (0000 1101).(0101 1001)

= $\langle 3x + 2x + 1 \rangle . \langle 6x + 4x + 3x + 1 \rangle$

= $\langle 9x + 7x + 6x + 3x + 8x + 6x + 5x + 2x + 6x + 4x + 3x + 1 \rangle$

= $\langle 9x + 8x + 7x + 6x + 5x + 4x + 2x + 1 \rangle \; mod \; \langle 8x + 4x + 3x + x + 1 \rangle$

= 1111110101 mod 100011011

= 0111000011 mod 100011011

= 011011000

= 11011000

= d8

09.46 = (0000 1001).(0100 0110)

= $\langle 3x + 1 \rangle . \langle 6x + 2x + x \rangle$

= $\langle 9x + 5x + 4x + 3x + 6x + 2x + x \rangle$

= $\langle 9x + 6x + 5x + 4x + 2x + x \rangle \; mod \; \langle 8x + 4x + 3x + x + 1 \rangle$

= 10001110110 mod 100011011

= 0001000000

= 01000000

= 40

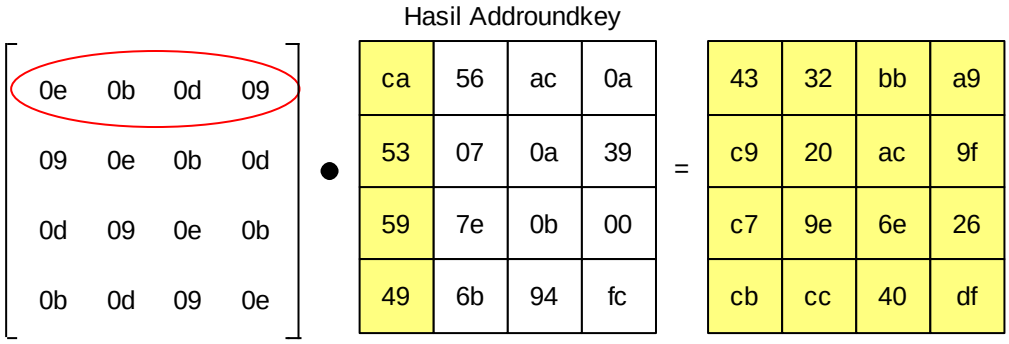
Hitung:

$r0 = \{0e.a0\}xor\{0b.a1\}xor\{0d.a2\}xor\{09.a3\}$

$r0 = \{80\}xor\{5b\}xor\{d8\}xor\{40\}$

$r0 = 43$

Ulangi proses diatas untuk menghasilkan nilai – nilai pada *byte* dan kolom selanjutnya, sehingga hasilnya dapat dilihat pada gambar 3.32

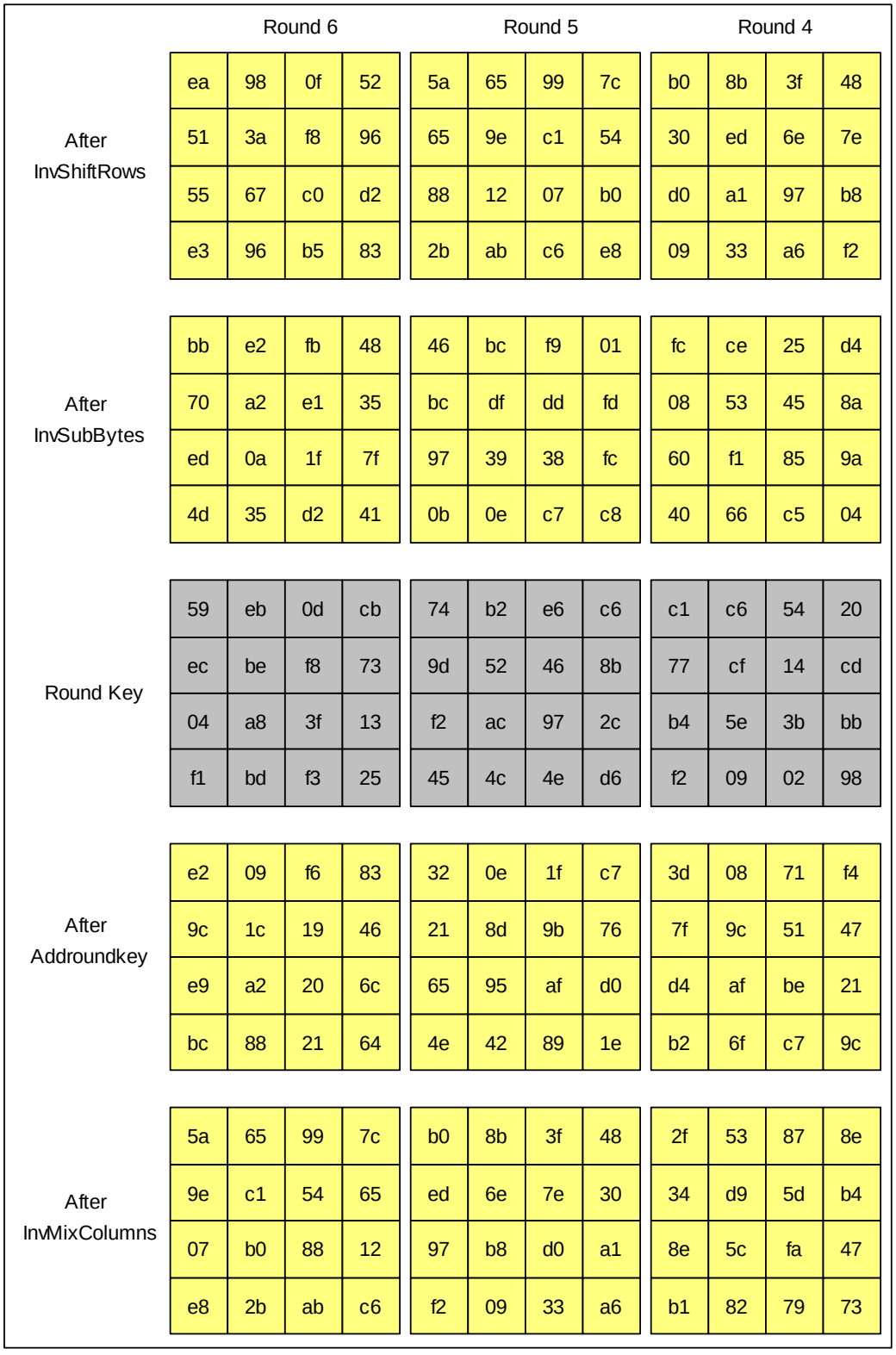


Gambar 3.30 Hasil *InvMixColumns* (*round* 10)

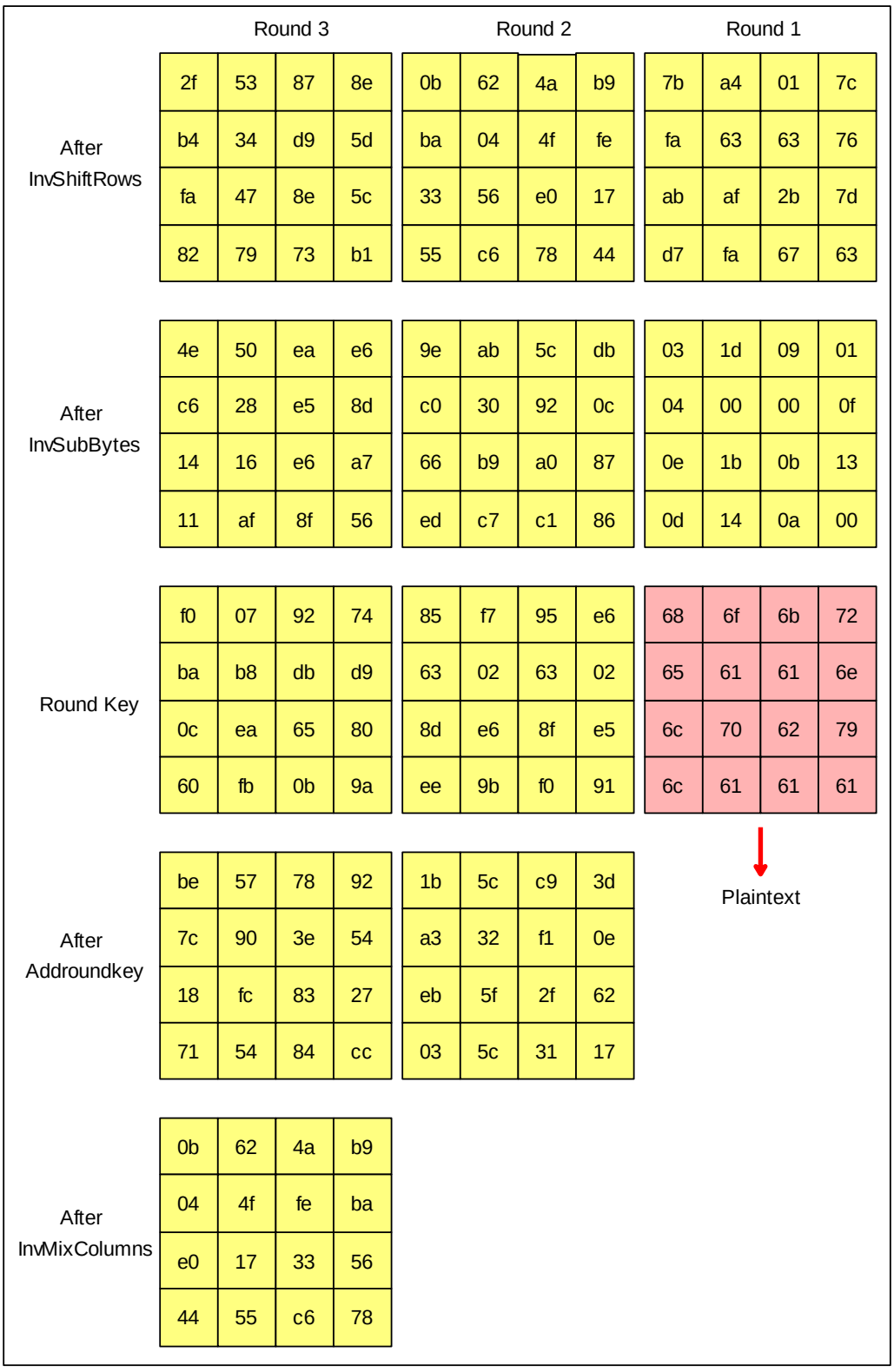
6. Ulangi langkah 1 sampai 5 untuk menghasilkan *Round* – *Round* selanjutanya untuk mendapatkan *plaintext*, sehingga hasil proses dekripsi dapat dilihat pada gambar dibawah ini.



Gambar 3.31 Hasil Proses Dekripsi



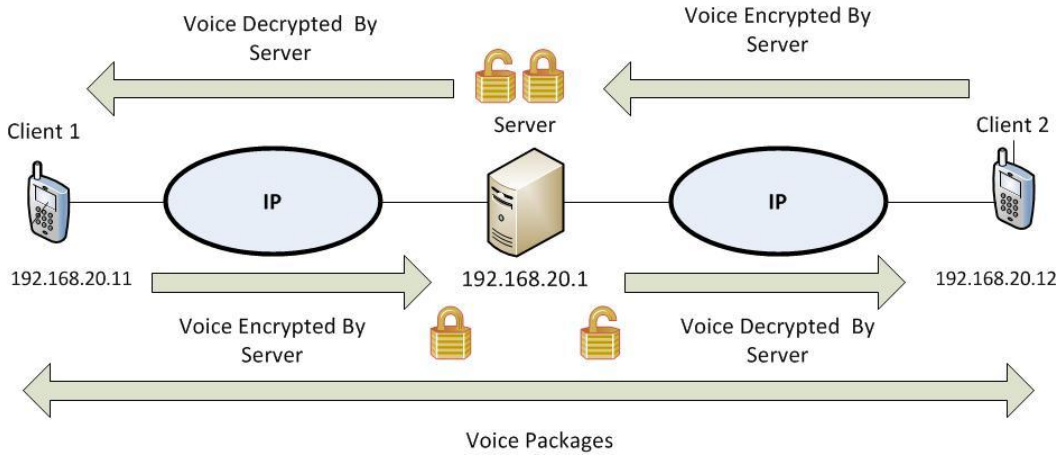
Gambar 3.32 Hasil Proses Dekripsi



Gambar 3.33 Hasil Proses Dekripsi

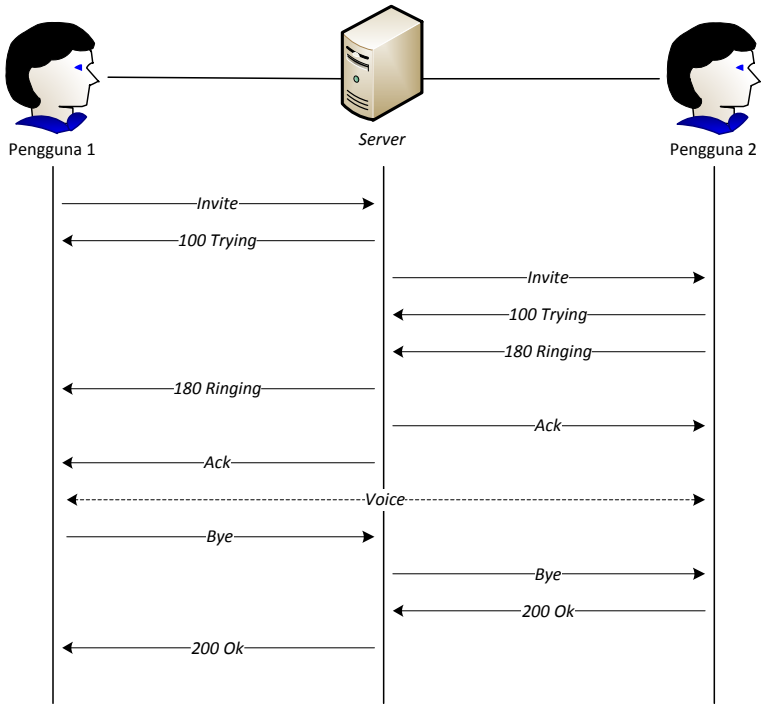
3.1.6 Analisis Arsitektur Sistem Yang Akan Dibangun

Analisis arsitektur sistem yang akan dibangun pada SSipCall ini merupakan aplikasi android yang akan dibangun dengan menggunakan SIP sebagai *signaling protocol*nya dan RTP sebagai *media transfer protocol*nya. SSipCall menggunakan UDP *(User Datagram protocol)* sebagai protocol TCP/IP nya, ini dikarenakan UDP memiliki kecepatan dalam transfer data tanpa harus menghabiskan sumber daya memori dan prosesor yang besar. Pada SSipCall akan terdapat dua fitur, sehingga pengguna dapat memilih melakukan panggilan tanpa keamanan atau melakukan panggilan dengan keamanan menggunakan metode SRTP. Arsitektur sistem pada aplikasi SSipCall dapat dilihat pada gambar 3.34



Gambar 3.34 Arsitektur Aplikasi SSipCall

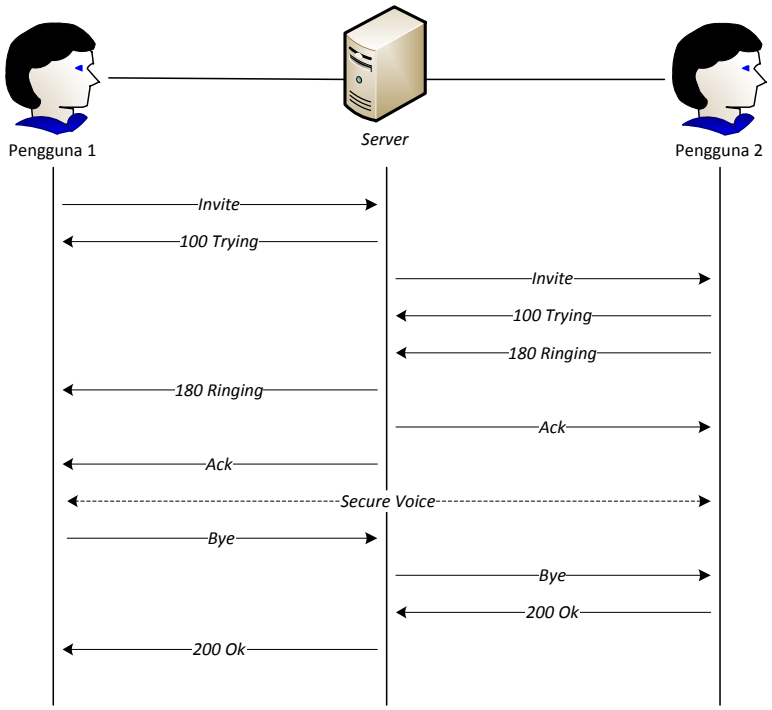
Berikut adalah *Callflow* yang melakukan panggilan tanpa tanpa keamanan dapat dilihat pada Gambar 3.35 sedangkan *Callflow* yang melakukan panggilan dengan keamanan dapat dilihat pada Gambar 3.36



Gambar 3.35 Alur *User* Yang Melakukan Panggilan Tanpa Keamanan

Berdasarkan Gambar 3.35 pengguna 1 melakukan *request Invite* ke pengguna 2, pesan tersebut tidak langsung disampaikan ke pengguna 2 melainkan ke *server* terlebih dahulu untuk melakukan proses autentikasi. Setelah *server* selesai melakukan proses autentikasi maka *server* membalas pesan *request* pengguna 1 dengan memberikan pesan *response* 100 *Trying* yang artinya *server* telah menerima *request Invite* pengguna 1 dan akan meneruskan *request Invite* ke pengguna 2. Kemudian pengguna 2 mengirim *response* 100 *Trying* dan 180 *Ringing* secara berurutan ke *server*. Dimana 100 *Trying* berarti pengguna 2 telah berhasil menerima *Invite* dari pengguna 1 yang disampaikan oleh *server* dan mencoba memberi tanda dengan mengirim *response* 100 *Ringing*. Setelah itu *server* meneruskan *request ringing* ke pengguna 1. Kemudian pengguna 2 melakukan proses *Ack* yang artinya pengguna 2 menerima pesan *Invite* dari pengguna 1 dan penguna 1 juga melakukan proses *Ack*. Setelah pengguna 1 dan pengguna 2 melakukan proses *Ack* maka keduanya telah bisa saling melakukan percakapan tanpa keamanan. Setelah percakapan selesai pengguna 1 mengirim *request Bye* ke *server* untuk menghentikan sesi komunikasi yang kemudian diteruskan ke pengguna 2. Dan

pengguna 2 menjawab dengan pesan *Response* 200 *Ok* ke *server* dan *server* melanjutkan *response* tersebut ke pengguna 1.



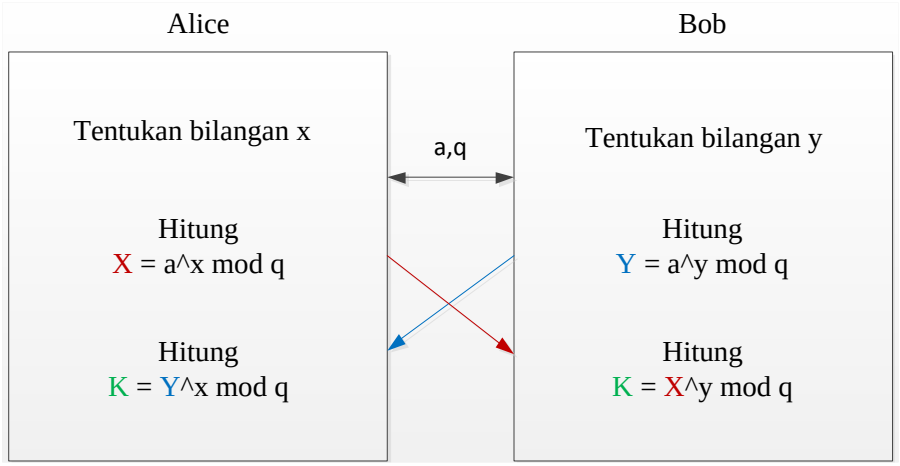
Gambar 3.36 Alur *User* Yang Melakukan Panggilan Dengan Keamanan

Berdasarkan Gambar 3.36 pengguna 1 melakukan *request Invite* ke pengguna 2, pesan tersebut tidak langsung disampaikan ke pengguna 2 melainkan ke *server* terlebih dahulu untuk melakukan proses autentikasi. Setelah *server* selesai melakukan proses autentikasi maka *server* membalas pesan *request* pengguna 1 dengan memberikan pesan *response* 100 *Trying* yang artinya *server* telah menerima *request Invite* pengguna 1 dan akan meneruskan *request Invite* ke pengguna 2. Kemudian pengguna 2 mengirim *response* 100 *Trying* dan 180 *Ringing* secara berurutan ke *server*. Dimana 100 *Trying* berarti pengguna 2 telah berhasil menerima *Invite* dari pengguna 1 yang disampaikan oleh *server* dan mencoba memberi tanda dengan mengirim *response* 100 *Ringing*. Setelah itu *server* meneruskan *request ringing* ke pengguna 1. Kemudian pengguna 2 melakukan proses *Ack* yang artinya pengguna 2 menerima pesan *Invite* dari pengguna 1 dan penguna 1 juga melakukan proses *Ack*. Setelah pengguna 1 dan pengguna 2 melakukan proses *Ack* maka keduanya telah bisa saling melakukan percakapan dengan keamanan. Setelah

percakapan selesai pengguna 1 mengirim *request Bye* ke *server* untuk menghentikan sesi komunikasi yang kemudian diteruskan ke pengguna 2. Dan pengguna 2 menjawab dengan pesan *Response* 200 *Ok* ke *server* dan *server* melanjutkan *response* tersebut ke pengguna 1.

3.1.7 Analisis Diffie-Hellman *Key Exchange*

Diffie-Hellman merupakan *key exchange* yang digunakan untuk pertukaran *public key*, pada Gambar 3.7 diperlihatkan cara kerja DH *key exchange*.



Gambar 3.37 Alur DH *Key Exchange*

Pada Gambar 3.37 menjelaskan bahwa Alice dan Bob sedang bertukar kunci. Hasil dari pertukaran *private key* tersebut akan menghasilkan *key* yang akan digunakan sebagai kunci untuk membuka pesan yang dienkripsi. Untuk membuktikan bahwa Alice dan Bob memiliki *key* yang sama dapat dilihat pada contoh kasus berikut ini:

1. Alice menentukan nilai x dengan 5
2. Bob menentukan nilai y dengan 6
3. Alice dan Bob bersepakat menentukan nilai a = 10 dan q = 2
4. Alice menghitung nilai X
5. Bob menghitung nilai Y
6. Alice dan Bob saling bertukar kunci
7. Alice dan Bob menghitung K dan akan menghasilkan nilai K yang sama.

Untuk membuktikan bahwa Alice dan Bob menghasilkan nilai K yang sama, dapat dilihat pada Tabel 3.2

Tabel 3.2 Perhitungan Diffie-Hellman

Alice	Bob
Private key x=5 Menghitung nilai X $a^x mod\ q = X$ $2^5 mod\ 10 = 2$ Hitung nilai K $Y^x mod\ q = K$ $4^5 mod\ 10 = 4$	*Private key* x=6 Menghitung nilai Y $a^y mod\ q = Y$ $2^6 mod\ 10 = 4$ Hitung nilai K $X^y mod\ q = K$ $2^6 mod\ 10 = 4$

Pada Tabel 3.2 dapat disimpulkan bahwa Alice dan Bob memiliki nilai *key* yang sama yaitu 4. Nilai tersebut didapat dari pertukaran *public key* Alice dan Bob yang berasal dari *private key* masing – masing.

3.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dibangun dibagi menjadi dua kebutuhan, yaitu kebutuhan nonfungsional dan kebutuhan fungsional. Spesifikasi kebutuhan perangkat lunak dapat dilihat pada Tabel 3.3 untuk SKPL fungsional dan Tabel 3.4 untuk SKPL nonfungsional.

Tabel 3.3 Spesifikasi Kebutuhan Fungsional

Nomor	Spesifikasi Kebutuhan Fungsional
SKPL-F-001	Sistem dapat melakukan proses enkripsi suara
SKPL-F-002	Sistem dapat melakukan pertukaran kunci
SKPL-F-003	Sistem dapat menginputkan nomor tujuan
SKPL-F-004	Sistem dapat melakukan panggilan suara dengan enkripsi
SKPL-F-005	Sistem dapat melakukan panggilan suara tanpa enkripsi
SKPL-F-006	Sistem dapat menerima panggilan suara
SKPL-F-007	Sistem dapat mengakhiri panggilan suara

Tabel 3.4 Spesifikasi Kebutuhan NonFungsional

Nomor	Spesifikasi Kebutuhan NonFungsional
SKPL-NF-001	Sistem yang dibangun menggunakan bahasa pemrograman *Java.*
SKPL-NF-002	Sistem dapat dijalankan pada android versi 2.3 – android versi 4.4
SKPL-NF-003	Sistem ini membutuhkan *Linux* sebagai sistem operasi *Asterisk* sebagai *server*.
SKPL-NF-004	Android Studio sebagai aplikasi untuk pembuatan aplikasinya.
SKPL-NF-005	Tidak ada pengguna khusus untuk menggunakan sistem ini.

3.2.1 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional menggambarkan kebutuhan luar sistem yang diperlukan untuk menjalankan aplikasi yang dibangun. Adapun analisis kebutuhan non fungsional untuk menjalankan aplikasi *SSipCall* ini meliputi, analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis kebutuhan pengguna. Analisis kebutuhan non fungsional bertujuan agar aplikasi yang akan dibangun dapat digunakan sesuai dengan kebutuhan.

3.2.1.1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan dalam membangun dan menjalankan aplikasi dapat dilihat pada Tabel 3.5

Tabel 3.5 Spesfikasi Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	Ubuntu 12.04 LTS
2	*Server*	Asterisk
3	Editor JAVA	Android Studio
4	*Sotware* Penguji	Wireshark

3.2.1.2 Analisis Kebutuhan Perangkat Keras

Untuk menjalankan aplikasi *SSipCall* ini maka dibutuhkan perangkat keras yang dapat mendukung proses kerja dari aplikasi *SSipCall*. Adapun spesifikasinya dapat dilihat pada Table 3.6 untuk spesifikasi perangkat keras kompter dan Tabel 3.7 untuk spesifikasi perangkat keras minimum *smartphone* dibawah ini:

Tabel 3.6 Spesfikasi Perangkat Keras Komputer Sebagai *Server*

No	Perangkat Keras	Spesifikasi
1	Prosesor	Kecepatan 2.30GHz
2	Memori	512MB
3	Harddisk	10GB
4	Sistem Operasi	Ubuntu

Tabel 3.7 Spesfikasi Perangkat Keras Minimun *Smartphone* Sebagai *Client*

No	Perangkat Keras	Spesifikasi
1	Dimensi layar	3,5”
2	RAM	512MB
3	Memory Internal	1GB
4	Prosesor	800Mhz
5	Resolusi	480 x 320 pixel
6	Sistem Operasi	GingerBread 2.3

3.2.1.3 Analisis Kebutuhan Pengguna

Analisis pengguna sistem dimaksudkan untuk mengetahui siapa saja aktor yang terlibat dalam menjalankan sistem. Aplikasi *SSipCall* ini sendiri menggunakan *platform* android. Berikut ini karakteristik pengguna sistem yang ditujukan pada tabel berikut:

Tabel 3.8 Karakteristik Pengguna Sistem

NO	Pengguna	Hak Akses	Tingkat Keterampilan
1.	*Client*	Menggunakan aplikasi untuk melakukan panggilan (dengan keamanan / tanpa kemanan) dan menerima panggilan telepon (dengan keamanan / tanpa kemanan).	Mengerti dan menguasai penggunaan *Smartphone* Android.
2.	Admin	Mengontrol dan mengelola *Server*.	1. Memahami Sistem Operasi Linux Ubuntu. 2. Memahami Konsep VoIP. 3. Memahami konsep jaringan.

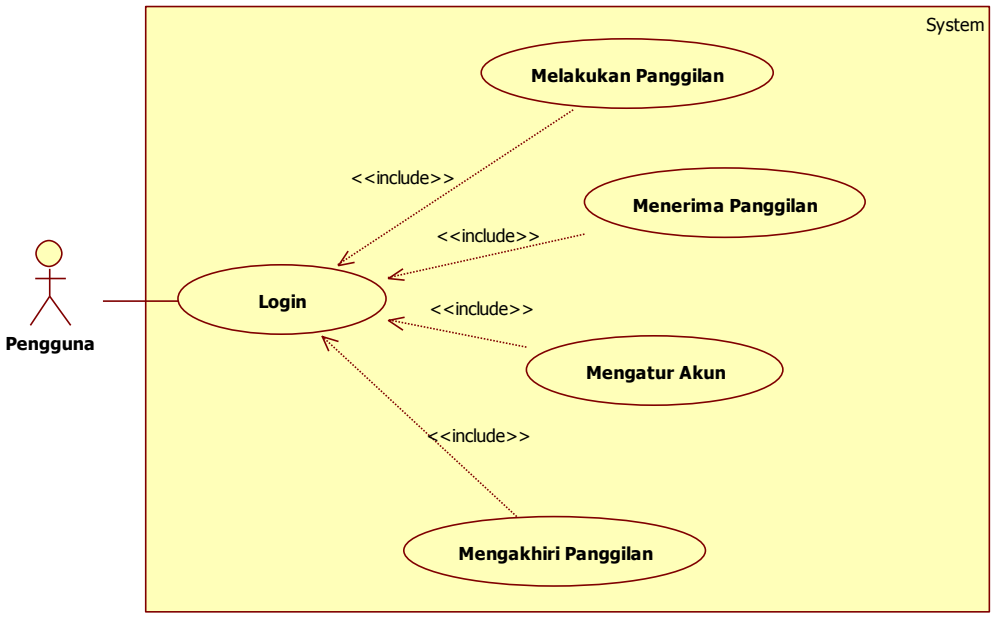
3.2.2 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional ini merupakan analisis tentang hal – hal yang dilakukan sistem ketika diimplemntasikan. Analisis kebutuhan ini untuk

menentukan masukan yang dibutuhkan sistem, proses yang digunakan untuk mengolah masukan menjadi keluaran, dan keluaran yang akan dihasilkan oleh sistem. Analisis sistem yang dilakukan menggunakan *tools* UML yang memiliki tahapan analisis sistem yang terdiri dari *use case diagram*, *use case scenario*, *activity diagram*, *class diagram*, dan *sequence diagram*.

3.2.2.1 Use Case Diagram

Sasaran pemodelan *use case* diantaranya adalah mendefinisikan gambaran umum dari sebuah sistem sehingga pengguna mengetahui mengenai sistem yang akan dibangun. *Use case diagram* untuk proses keseluruhan aplikasi *SSipCall* ini dapat dilihat pada Gambar 3.38 dan untuk definisi *use case* dapat dilihat pada Tabel 3.9



Gambar 3.38 *Use Case Diagram SSipCall*

Tabel 3.9 Definisi *Use Case*

No	*Use Case*	Deskripsi
1.	Login	Proses untuk mengiputkan Nama Pengguna, *Server*, Kata Sandi dan memilih keamanan
2.	Melakukan Panggilan	Proses untuk menginputkan alamat SIP yang ingin dituju untuk ditelepon

No	*Use Case*	Deskripsi
3.	Menerima Panggilan	Proses untuk menerima panggilan masuk
4.	Mengatur akun	Proses untuk mengedit Nama Pengguna, *Server*, Kata Sandi dan memilih keamanan
5.	Mengakhiri Panggilan	Proses untuk mengakhiri panggilan

3.2.2.2 Use case scenario

Use Case Scenario bertujuan untuk mendeskripsikan urutan langkah - langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor. Berdasarkan Gambar 3.38 maka *use case scenario* untuk sistem yang dibangun akan dijelaskan sebagai berikut:

Tabel 3.10 *Use Case Scenario* Login

Use Case Name	Login	
Goal In Context	Sistem berhasil Login	
Precondition	-	
Description	Proses ini digunakan untuk registrasi terhadap *server*	
Successful End Condition	Sistem berhasil Login dan registrasi sukses	
Failed End Condition	Sistem tidak berhasil Login dan registrasi gagal	
Actors	Pengguna	
Trigger	Pengguna meminta sistem untuk mengautentikasi kepada *server*	
Main Flow	Step	Action
	1.	Login
	2.	Inputkan Nama Pengguna
	3.	Inputkan *Server*
	4.	Inputkan Kata Sandi
	5.	Pilih keamanan
Extension	5.1	Jika Keamanan diaktifkan maka panggilan akan terenkripsi
	5.2	Jika Keamanan tidak diaktifkan panggilan suara tidak akan terenkripsi.

Tabel 3.11 *Use Case Scenario* Melakukan Panggilan

Use Case Name	Melakukan Panggilan	
Goal In Context	Berhasil melakukan Panggilan suara	
Precondition	Login	
Description	Proses ini digunakan untuk melakukan panggilan.	
Successful End Condition	Sistem berhasil melakukan panggilan suara	
Failed End Condition	Sistem tidak melakukan panggilan suara	
Actors	Pengguna	
Trigger	Pengguna meminta sistem untuk memproses panggilan suara	
Main Flow	Step	Action
	1.	Masukan alamat SIP yang ingin dituju
	2.	Tekan Oke

Tabel 3.12 *Use Case Scenario* Menerima Panggilan

Use Case Name	Menerima Panggilan	
Goal In Context	Berhasil menerima Panggilan masuk	
Precondition	Login	
Description	Proses ini digunakan untuk menerima panggilan masuk	
Successful End Condition	Sistem berhasil Berhasil menerima Panggilan masuk	
Failed End Condition	Sistem tidak Berhasil menerima Panggilan masuk	
Actors	Pengguna	
Trigger	-	
Main Flow	Step	Action
	1.	Pilih Terima

Tabel 3.13 *Use Case Scenario* Mengatur Akun

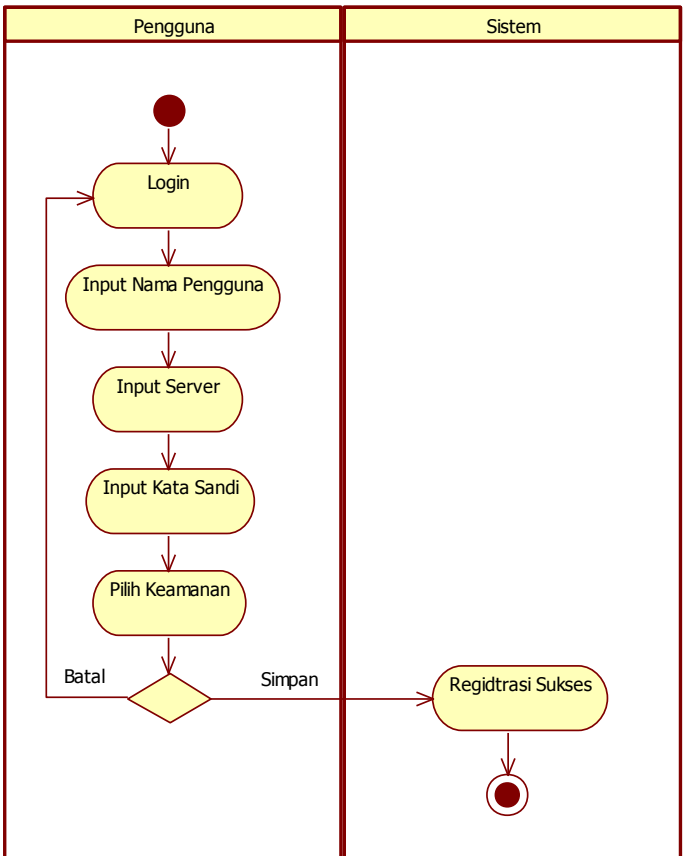
Use Case Name	Mengatur Akun	
Goal In Context	Berhasil Mengatur Akun	
Precondition	Login	
Description	Proses ini digunakan untuk mengedit Nama Pengguna, *Server*, Kata Sandi dan memilih keamanan	
Successful End Condition	Sistem berhasil mengedit akun dan regisrasi sukses	
Failed End Condition	Sistem tidak berhasil mengedit akun dan regisrasi gagal	
Actors	Pengguna	
Trigger	Pengguna meminta sistem untuk memproses akun yang telah diedit	
Main Flow	Step	Action
	1.	Pilih Mengatur Akun
	2.	Edit Nama Pengguna
	3.	Edit *Server*
	4.	Edit Kata Sandi
	5.	Pilih keamanan
Extension	5.1	Jika Keamanan diaktifkan maka panggilan akan terenkripsi
	5.2	Jika Keamanan tidak diaktifkan panggilan suara tidak akan terenkripsi.

Tabel 3.14 *Use Case Scenario* Mengakhiri Panggilan

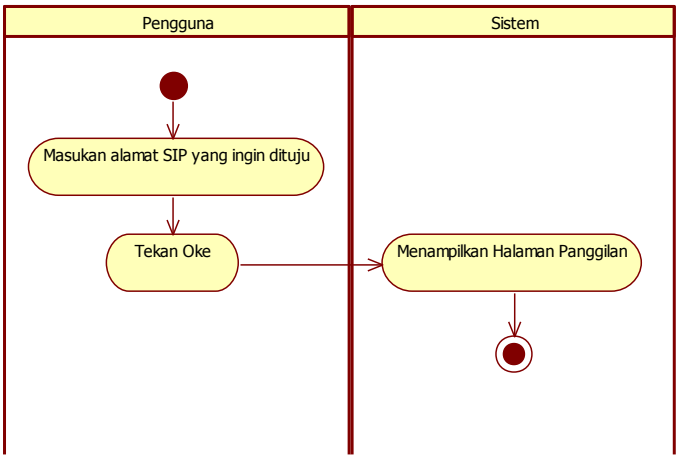
Use Case Name	Mengakhiri panggilan	
Goal In Context	Berhasil mengakhiri panggilan suara	
Precondition	-	
Description	Proses ini digunakan untuk mengakhiri panggilan suara	
Successful End Condition	Sistem berhasil mengakhiri panggilan suara	
Failed End Condition	Sistem tidak berhasil mengakhiri panggilan suara	
Actors	Pengguna	
Trigger	-	
Main Flow	Step	Action
	1.	Tekan Akhiri Panggilan

3.2.2.3 Activity Diagram

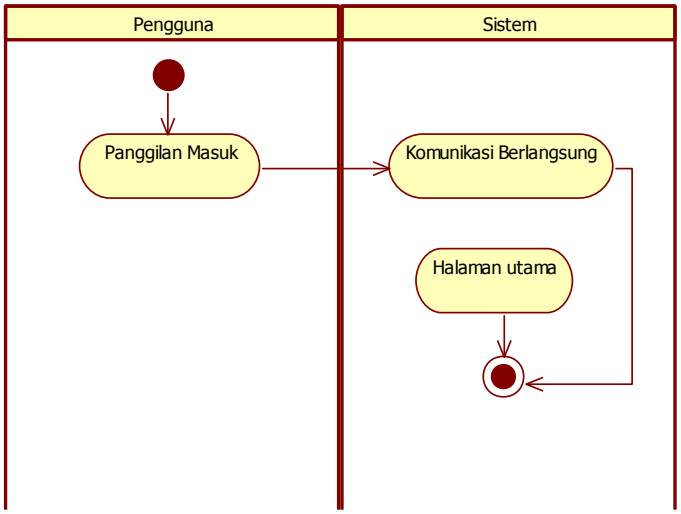
Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Berikut merupakan *activity diagram* dari pembangunan sistem ini berdasarkan dari *use case scenario* :



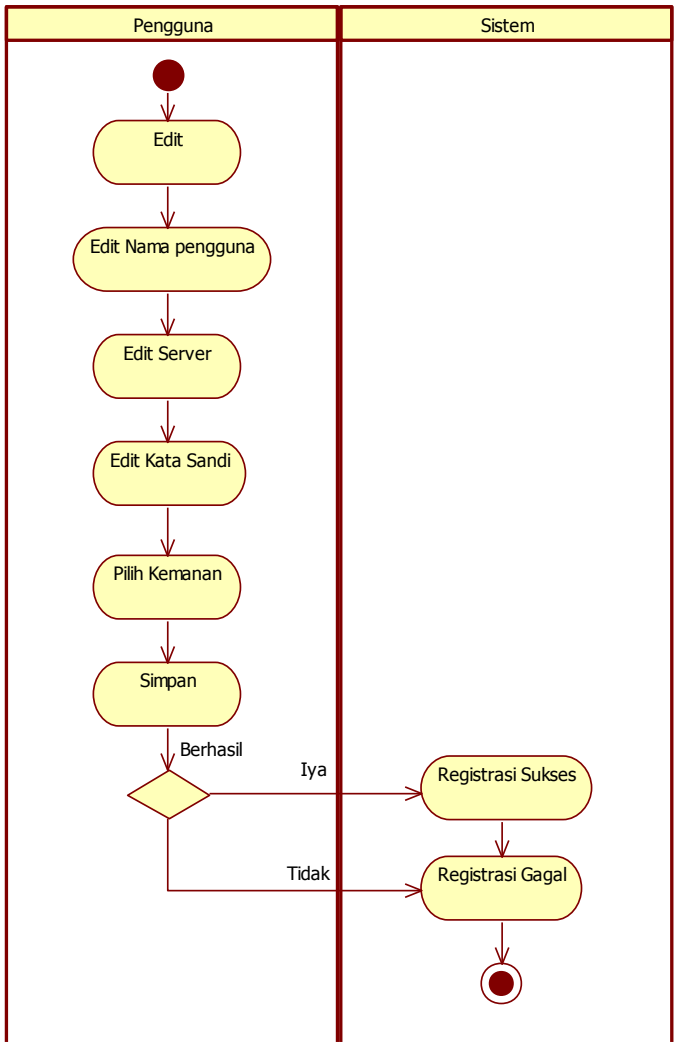
Gambar 3.39 *Activity Diagram* Login



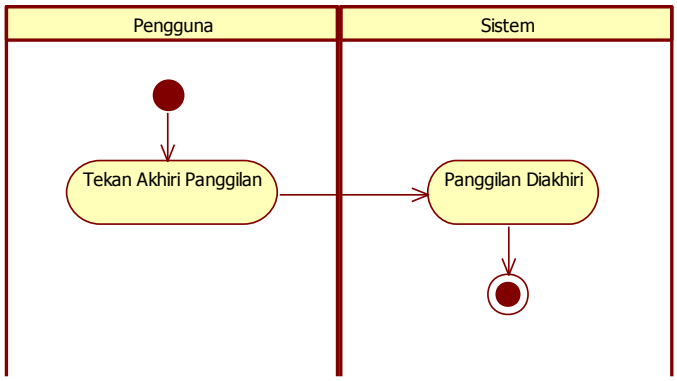
Gambar 3.40 *Activity Diagram* Melakukan Panggilan



Gambar 3.41 *Activity Diagram* Menerima Panggilan



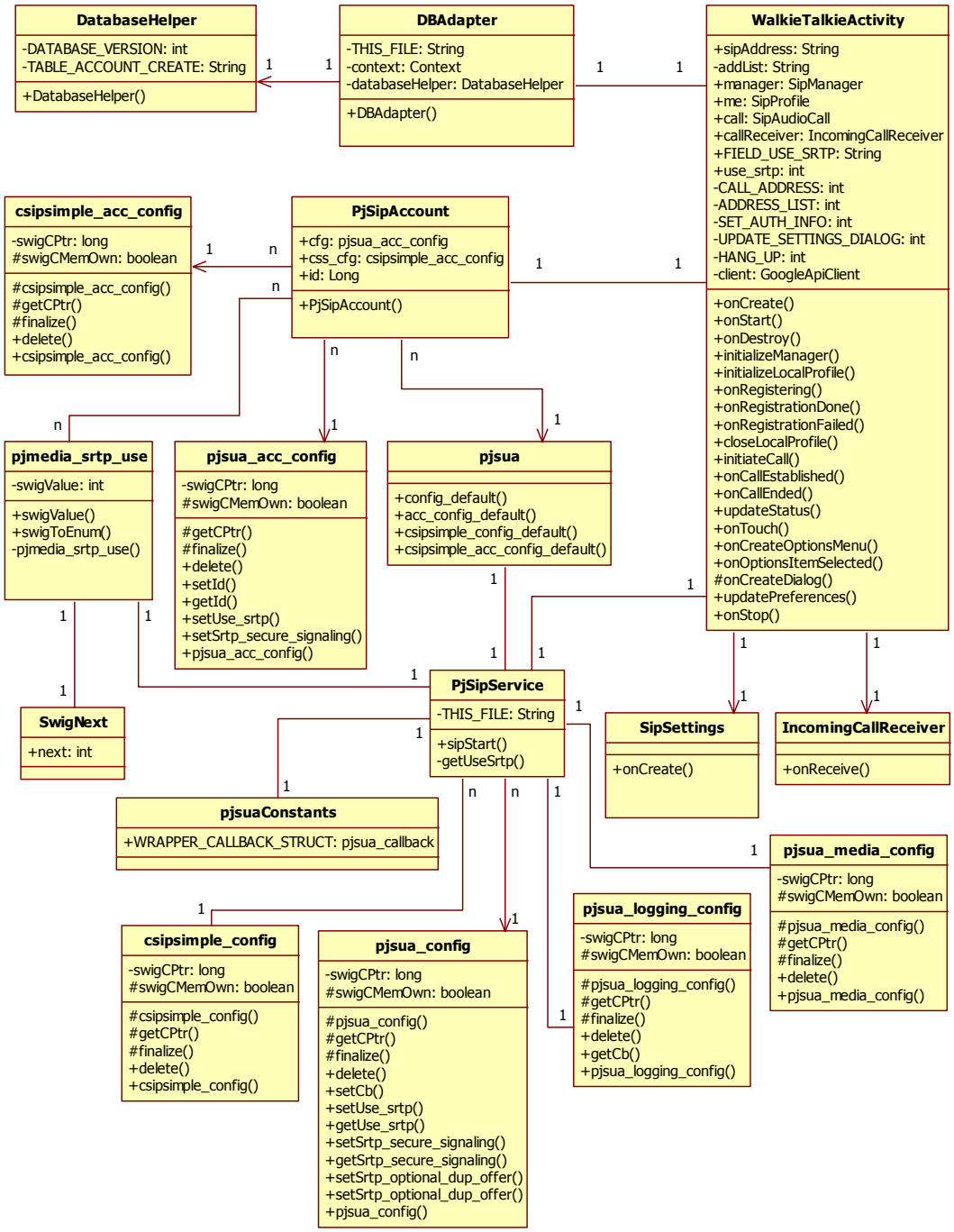
Gambar 3.42 *Activity Diagram* Pengaturan Akun



Gambar 3.43 *Activity Diagram* Mengakhiri Panggilan

3.2.2.4 *Class Diagram*

Pada gambar dibawah ini merupakan *class diagram* dari pembangunan Aplikasi SSipCall ini.

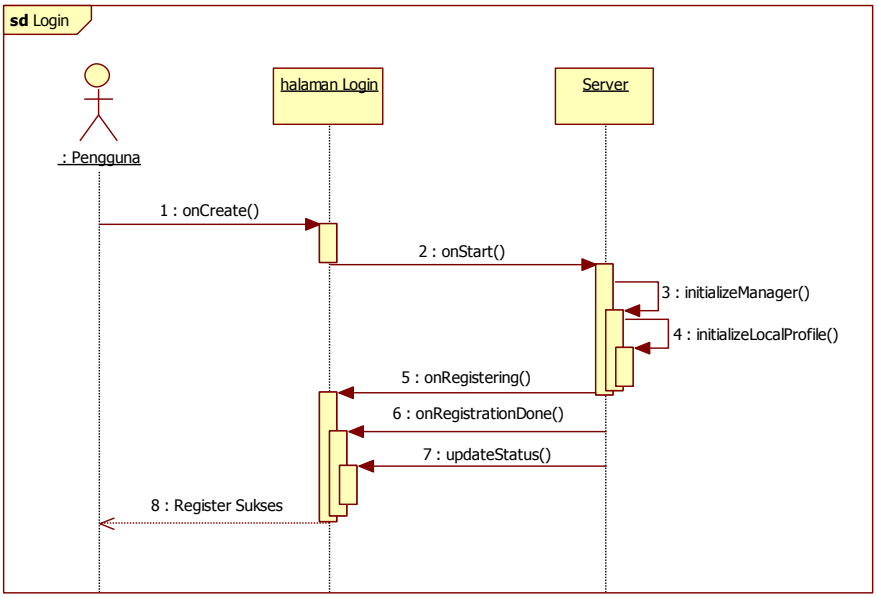


Gambar 3.44 *Class Diagram Aplikasi SSipCall*

3.2.2.5 Sequence Diagram

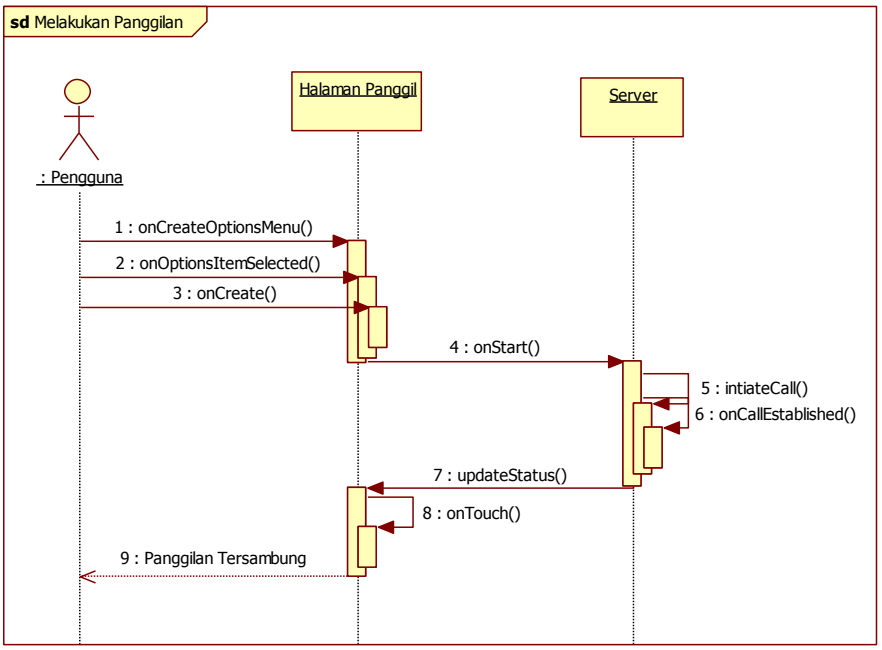
Pada bagian ini akan dipaparkan *sequence diagram* dari *usecase* yang berkaitan.

1. ***Sequence Diagram* Login**



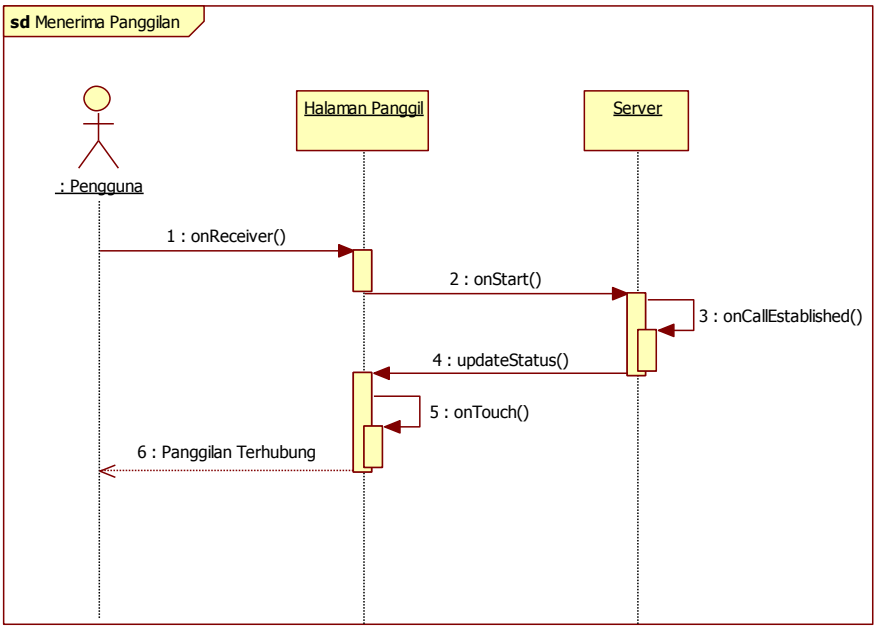
Gambar 3.45 *Sequence Diagram* Login

2. ***Sequence Diagram* Melakukan Panggilan**



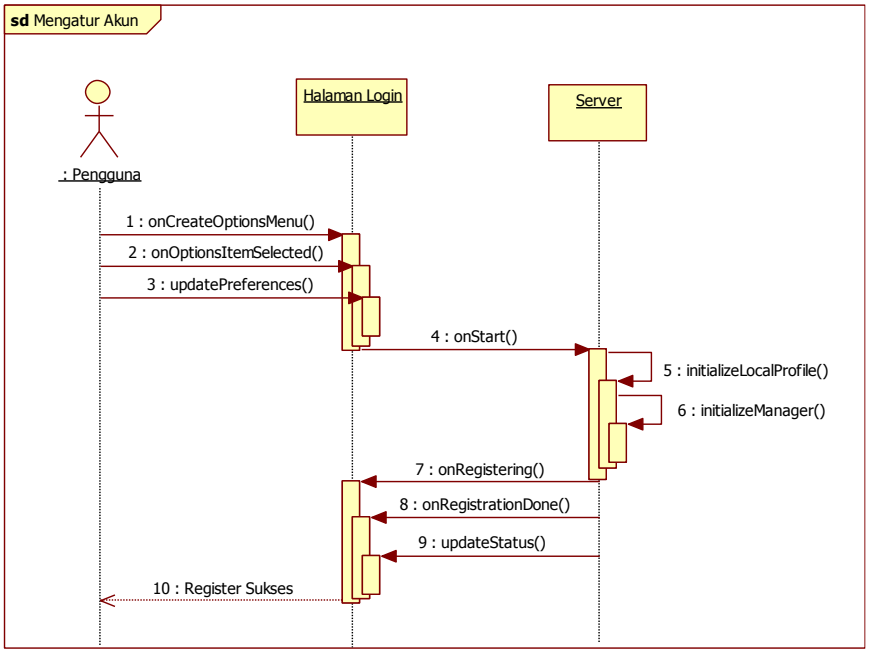
Gambar 3.46 *Sequence Diagram* Melakukan Panggilan

3. *Sequence Diagram* Menenerima Panggilan



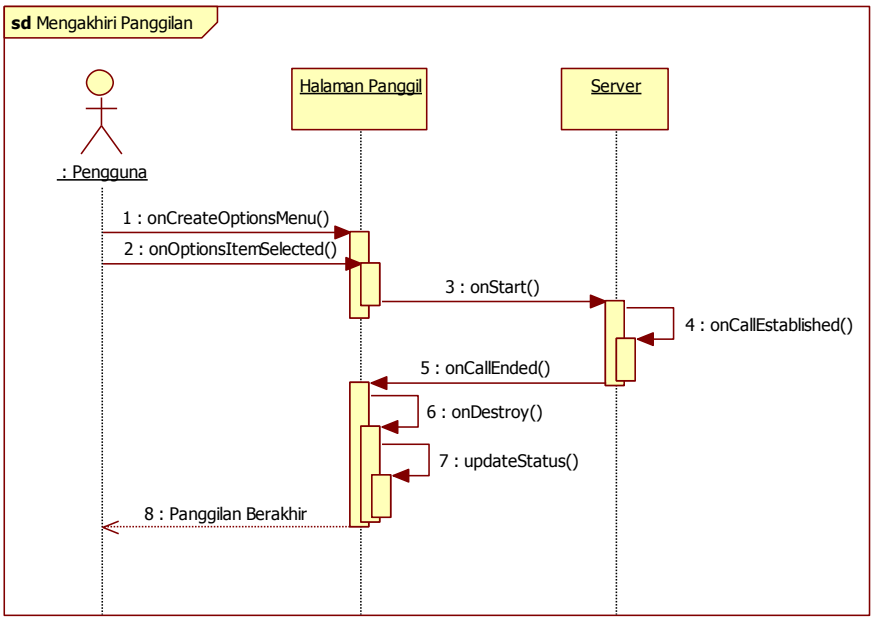
Gambar 3.47 *Sequence Diagram* Menenerima Panggilan

4. *Sequence Diagram* Mengatur Akun



Gambar 3.48 *Sequence Diagram* Mengatur Akun

5. ***Sequence Diagram* Mengakhiri Panggilan**



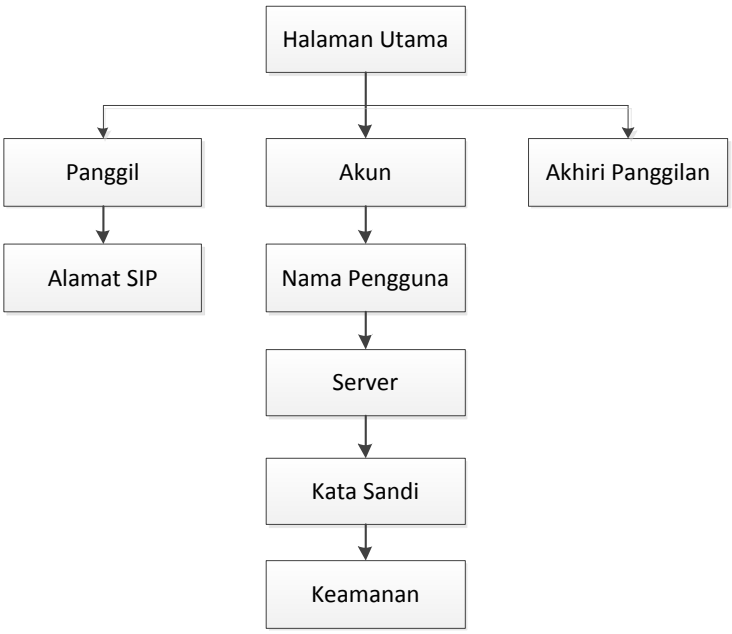
Gambar 3.49 *Sequence Diagram* Mengakhiri Panggilan

3.3 Perancangan Sistem

Perancangan sistem dari pembangunan Aplikasi SSipCall ini adalah sebagai berikut:

3.3.1 Perancangan Struktur Menu

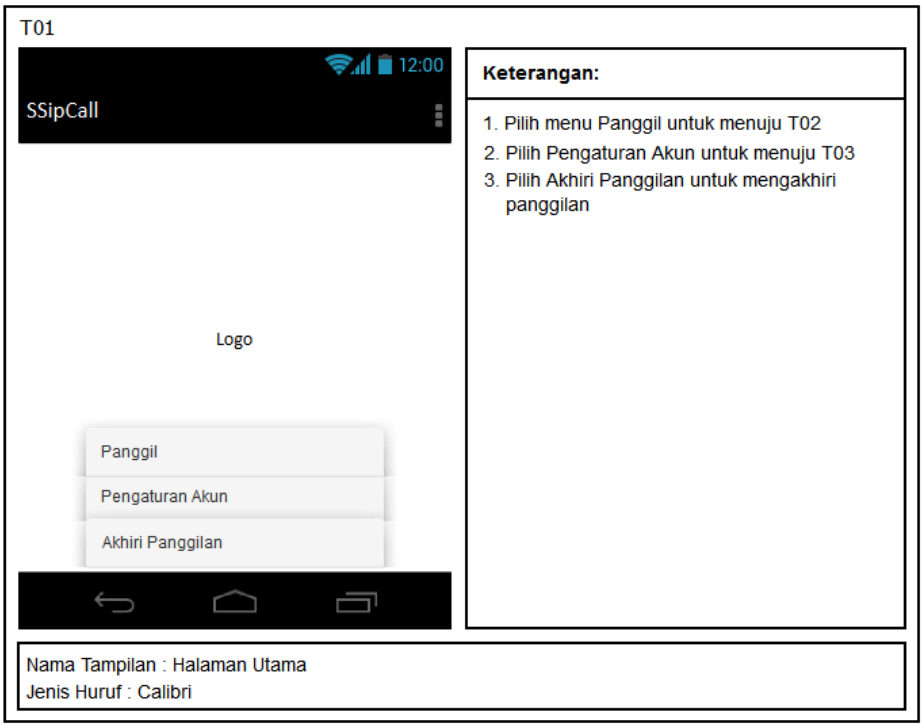
Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi sehingga aplikasi yang dibangun mudah dipahami dan mudah digunakan. Perancangan struktur menu dari sistem ini dapat dilihat pada Gambar 3.50



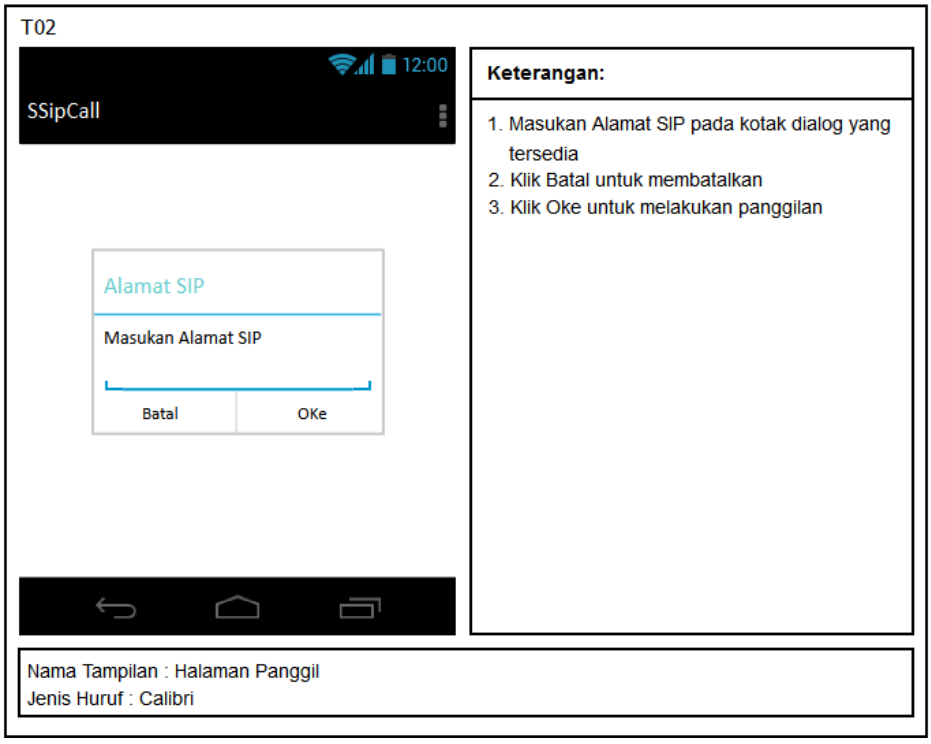
Gambar 3.50 Struktur Menu

3.3.2 Perancangan Antarmuka Sistem

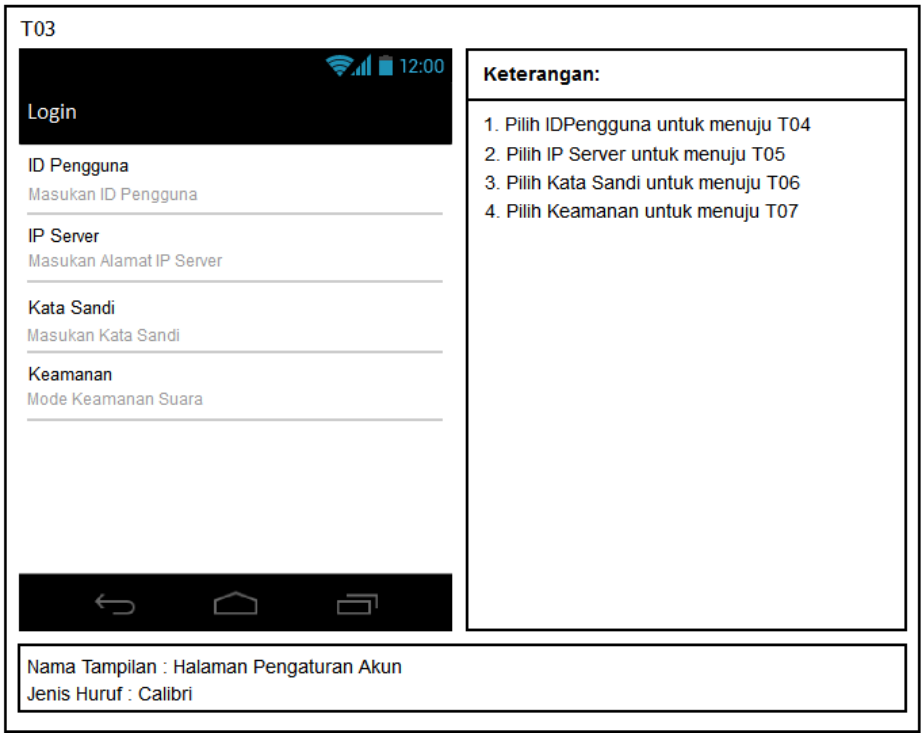
Perancangan antarmuka mendeskripsikan rencana tampilan dari setiap *form* yang akan diimplementasikan pada aplikasi SSipCall. Berikut adalah perancangan antarmuka aplikasi SSipCall.



Gambar 3.51 Perancangan Antarmuka T01 – Halaman Utama



Gambar 3.52 Perancangan Antarmuka T02 – Halaman Panggil



Gambar 3.53 Perancangan Antarmuka T03 – Halaman Pengaturan Akun



Gambar 3.54 Perancangan Antarmuka T04 – Halaman Nama Pengguna



Gambar 3.55 Perancangan Antarmuka T05 – Halaman *Server*



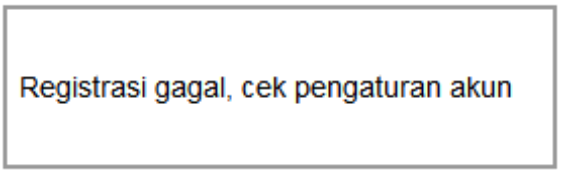
Gambar 3.56 Perancangan Antarmuka T06 – Halaman Kata Sandi



Gambar 3.57 Perancangan Antarmuka T07 – Halaman Keamanan

3.3.3 Perancangan Pesan

Perancangan pesan mendeskripsikan rencana tampilan dari setiap pesan yang muncul dalam aplikasi *SSipCall*. Berikut adalah perancangan pesan sistem *SSipCall*.



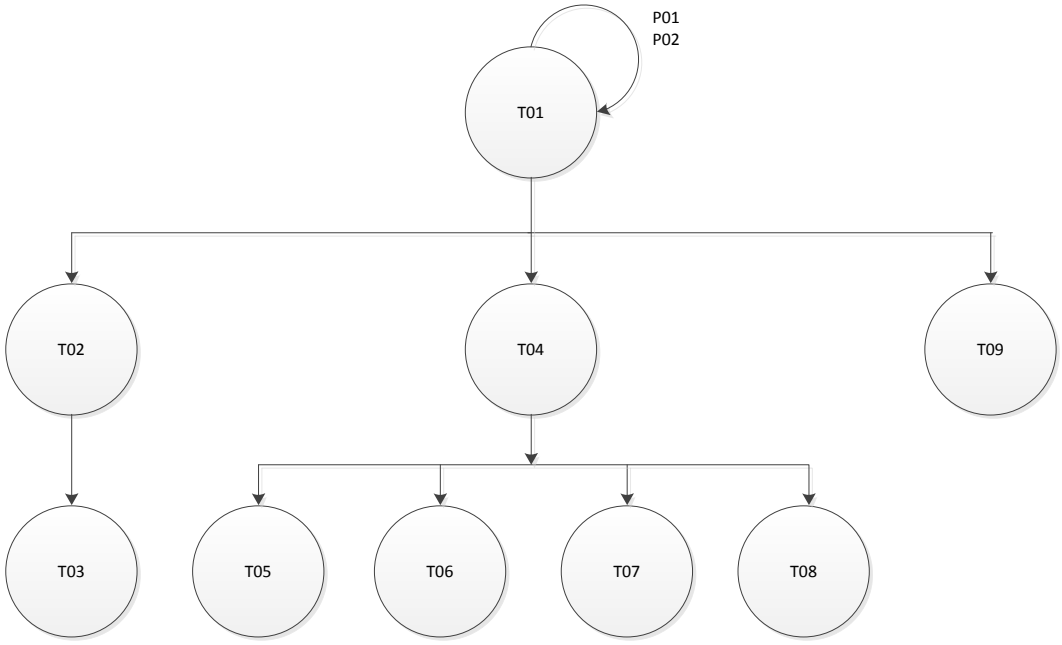
Gambar 3.58 Perancangan Pesan P01 – Pesan Saat Registrasi Gagal



Gambar 3.59 Perancangan Pesan P02 – Pesan Saat Registrasi Sukses

3.3.4 Jaringan Semantik

Jaringan semantik menggambarkan keterhubungan dari setiap menu dari tampilan satu ke tampilan lain. Berikut adalah jaringan semantik pada aplikasi *SSipCall* :



Gambar 3.60 Jaringan Semantik