BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis dan perancangan sistem memerlukan tahapan yang sistematis untuk mendapatkan aplikasi yang baik dan bersesuaian dengan kegunaan dan tujuannya. Tahap awal dari analisis adalah menganalisis kebutuhan-kebutuhan sistem mulai dari kebutuhan pengguna, kebutuhan non fungsional, dan kebutuha fungsional. Sedangkan untuk tahapan perancangan aplikasi yaitu, perancangan sistem kerja aplikasi dan perancangan antarmuka aplikasi.

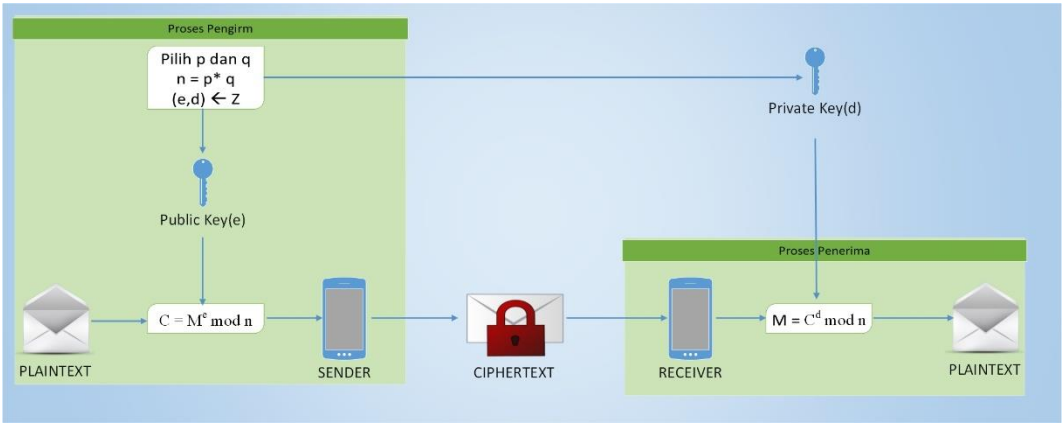
3.1.1 Analisis Masalah

Analisis masalah merupakan bentuk penjabaran mengenai masalah yang ada sebelum dibangunnya aplikasi. Adapun analisis masalah dari aplikasi keamanan SMS menggunakan algoritma RSA berbasis *mobile* pada *platform* android adalah sebagai berikut :

1. Layanan SMS yang menggunakan aplikasi bawaan ponsel masih banyak digunakan oleh setiap orang, dan bukan merupakan jalur yang aman dalam pertukaran informasi, apalagi jika informasi yang sifatnya penting atau rahasia perusahaan yang tidak boleh diketahui oleh orang yang tidak berhak.
2. SMS dirancang untuk komunikasi tidak singkron dimana pesan yang dikirimkan adalah *plaintext*, data *plaintext* seperti ini dapat dicegat di jalan oleh siapa saja yang memiliki akses ke sistem SMS. *Short Message Service Center* (SMSC) milik operator merupakan salah satu pihak yang dapat mengambil data ini.
3. SMS dibangun dengan sistem bahasa program yang sejenis dengan bahasa program *hardware* seperti komputer dan telepon selular dapat menerjemahkan semua data dalam frekuensi tertentu yang terbuka (di udara) yang sangat rentan akan ancaman seperti penyadapan pihak yang tak bertanggung jawab.

3.1.2 Analisis Proses Kerja Sistem

Analisis proses kerja sistem digunakan untuk mengetahui alur proses dari aplikasi yang akan dibangun. Pembangunan aplikasi ini menggunakan algoritma RSA untuk proses enkripsi dan dekripsi dengan sebagai keamanan terhadap SMS. Tahapan-tahapan pada aplikasi Keamanan SMS secara umum terlihat pada gambar 3.1 :

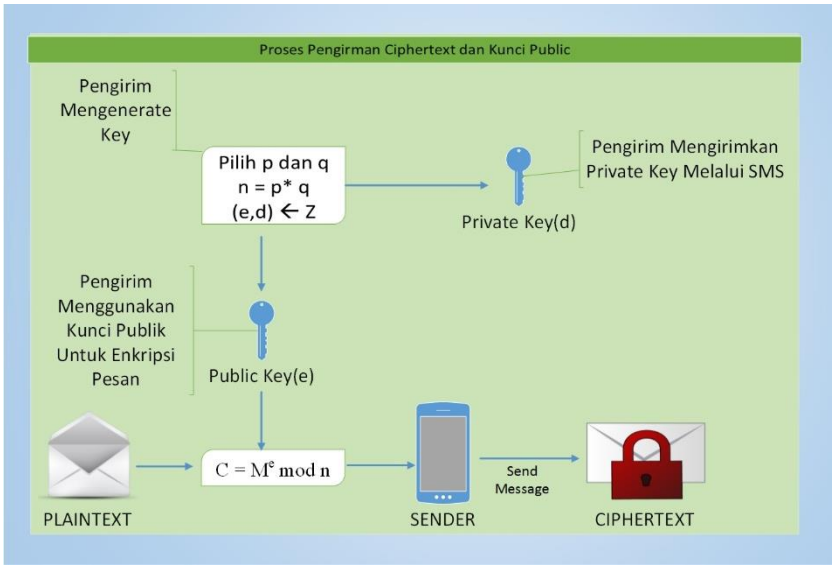


Gambar 3.1 Gambaran Umum Aplikasi Keamanan SMS

Tahapan-tahapan yang terdapat pada gambaran umum aplikasi terbagi menjadi dua proses utama, yaitu proses pengiriman pesan dan proses penerimaan pesan. Tahapan- tahapan setiap prosesnya yaitu sebagai berikut :

1. Proses Pengiriman Pesan
    a. Pengirim pesan membangkitkan sepasang kunci, lalu menyimpan dan mengirimkan *private key* kepada peneriman pesan.
    b. Pengirim memasukan nomor tujuan, jika nomor tujuan sudah ada di dalam *database*, dalam arti sudah memiliki pasangan kunci maka secara otomastis kunci untuk melakukan enkripsi terhadap pesan menggunakan pasangan kunci yang dikhususkan untuk nomor tersebut.
    c. Pengirim pesan mengetikan *plaintext* yang akan di enkripsi.
    d. Pengirim melakukan enkripsi terhadap *plaintext* menggunakan kunci publik yang telah dibangkitkan untuk menghasilkan pesan *ciphertext.*
    e. Pengirim mengirimkan *ciphertext* melalui SMS kepada penerima pesan.

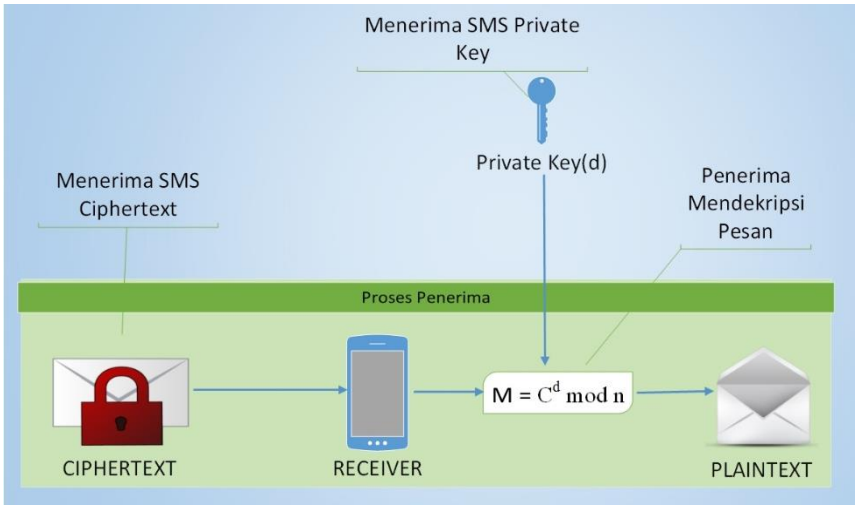
Tahapan-tahapan pengiriman pesan dapat digambarkan lebih rinci terlihat pada gambar 3.2 :



Gambar 3.2 Proses disisi Pengirim

2. Proses Penerimaan Pesan
   a. Penerima menyimpan *private key* yang telah dikirimkan oleh pengirim.
   b. Penerima memilih pesan *ciphertext* yang dikirimkan pengirim.
   c. Penerima menekan tombol *decrypt* untuk mendeikipsi pesan.
   d. Penerima pesan mendapatkan pesan *plaintext* yang telah di dekripsi.

   Tahapan-tahapan penerimaan pesan dapat digambarkan lebih rinci pada gambar berikut :



Gambar 3.3 Proses disisi Penerima

3.1.3 Analisis Algoritma RSA

Algoritma kriptografi RSA adalah algoritma untuk keamanan data yang ditemukan oleh Rivest, Shamir dan Adleman (RSA) pada tahun 1977-1978. Penerapan Algoritma kriptografi pada tabel di suatu basis data merupakan salah satu cara untuk mengamankan *record* yang tersimpan di dalam basis data.

Menurut M. Zaki Riyanto dan Ardhi Ardhian (*Kriptografi Kunci Publik : Sandi RSA*, 2008) : Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar. Selain sulitnya memfaktorkan bilangan yang besar pada pembentukan kunci pada RSA, RSA menawarkan kemampuan untuk mengendalikan panjang kunci yang digunakan oleh penggunanya sehingga dapat menyesuaikan dengan kebutuhan pengguna [9].

3.1.3.1 Langkah Kerja Algoritma RSA

Skema algoritma kriptografi kunci publik (asimetris) RSA terdiri dari tiga proses yaitu pembentukan kunci, proses enkripsi, dan proses dekripsi.

3.1.3.2 Algoritma Pembangkit Kunci

Berikut langkah-langkah membangkitkan kunci algoritma RSA :

1. Menentukan 2 buah bilangan prima acak dan dirahasiakan, dimisalkan kedua bilangan prima tersebut diwakilkan oleh variabel *p* dan *q*. Dengan syarat nilai *p* ≠ nilai *q*.

   Semakin besar bilangan prima *p* dan *q*, maka kemungkinan untuk dilakukannya faktorisasi dengan metode *brute force* guna mencari nilai *p* dan nilai *q* nantinya akan semakin kecil.
2. Menentukan (RSA) modulus yang berfungsi sebagai salah satu pasangan kunci publik maupun kunci rahasia. (RSA) modulus di wakilkan dengan variabel *n*. Nilai *n* didapatkan dari perkalian dua buah bilangan prima yang ditentukan pada langkah sebelumnya, yang diwakilkan oleh variabel p dan variabel *q*. Secara sistematis, formula mencari nilai (RSA) modulus adalah sebagai berikut :

$$\boldsymbol{n = p * q}$$

3. Langkah selanjutnya adalah menentukan (RSA) *enciphering exponent* yang nantinya digunakan sebagai pasangan kunci publik bersama (RSA) modulus. (RSA) enciphering exponent diwakilkan oleh variabel *e*. Formula untuk menentukan nilai *e* sebagai berikut :

$$\boldsymbol{gcd(e, m) = 1}$$

dengan syarat 1 < *e* < *m* dan *e* adalah bilangan prima. *gcd* adalah akronim dari *greatest common divisor* atau faktor persekutuan terbesar (FPB). Bila gcd atau FPB dari dua buah bilangan prima terhadap nilai *m*, sehingga nilai *e* bisa digunakan untuk menjadi pasangan kunci publik bersama (RSA) modulus.

Formula penentuan nilai (RSA) *enciphering exponent* yang disebutkan sebelumnya hanya bisa dijalankan bila nilai *m* telah ditentukan.

Formula untuk mencari nilai *m* sendiri adalah : bernilai = 1, misalnya *gcd(e,m)= 1*, artinya nilai *e* merupakan bilangan yang relatif prima terhadap nilai *m*, sehingga nilai *e* bisa digunakan untuk menjadi pasangan kunci publik bersama (RSA) modulus.

Formula penentuan nilai (RSA) *enciphering exponent* yan disebutkan sebelumnya hanya bisa dijalankan bila nilai m telah ditentukan.

Formula untuk mencari nilai m sendiri adalah :

$$\boldsymbol{m = (p - 1) * (q - 1)}$$

4. Langkah selanjutnya adalah menentukan nilai (RSA) *deciphering exponent* yang diwakilkan oleh variabel *(d)*, (RSA) *deciphering exponent* berfungsi sebagai pasangan (RSA) modulus dalam pembentukan kunci rahasia *(n,d)*. Formula mencari nilai *d* adalah sebagai berikut :

$$\boldsymbol{d = e^{-1} mod\, m}$$

Nilai dari vaiabel *e* dan variabel *m* adalah nilai yang didapat dari langkah sebelumnya dalam pembentukan (RSA) *enciphering exponent.*

Dengan syarat *1 < d < m.*

Dalam praktiknya, rumus baku untuk mencari nilai *d* diatas dijabarkan lebih lanjut sebagai berikut :

$$\boldsymbol{d = e^{-1} \; mod \; m}$$

$$\boldsymbol{e * d = 1 * (mod \; m)}$$

$$\boldsymbol{e * d = 1 + (k * m)}$$

$$\boxed{\boldsymbol{e * d = 1 + (k * m)}}$$

Variabel *k* adalah bilangan integer dengan nilai sembarang yang diharapkan menghasilkan *d* yang bernilai integer juga karena nilai d yang akan diambil jika d = integer.

5. Nilai variabel ***d, p, q*** dan ***m*** adalah nilai yang harus dirahasiakan. Nilai variabel ***n, e*** adalah nilai yang boleh dipublikasikan.

   Pasangan ***(n,e)*** merupakan kunci publik yang boleh dipublikasikan Pasangan ***(n,d)*** merupakan kunci rahasia yang harus dirahasiakan.

3.1.3.3 Algoritma Enkripsi

Untuk melakukan Enkripsi menggunakan kunci publik *(n,e)* digunakan formula enkripsi sebagai berikut [9] :

$$\boxed{\boldsymbol{C = M^e \; mod \; n}}$$

Dimana :

c = *Ciphertext* / text yang telah dienkripsi

m = *Plaintext* / Pesan asli

e = *Enciphering Exponent*

n = (RSA) modulus

3.1.3.4 Algoritma Deskripsi

Untuk melakukan dekripsi suatu *chiphertext* yang telah dienkripsi sebelumnya, digunakan pasangan kunci rahasia *(n,d)*, formula deskripsi yang digunakan sebagai berikut [9] :

$$M = C^d \bmod n$$

Dimana :

C = *Ciphertext* / text yang telah dienkripsi

M = *Plaintext* / Pesan asli

e = *Enciphering Exponent*

n = (RSA) modulus

3.1.4 Analisis *Unicode Transformasi Format* (UTF) 16

Orang-orang di negara-negara yang berbeda menggunakan karakter berbeda untuk menuliskan kata-kata dalam bahasa pribumi mereka. Sekarang ini kebanyakan aplikasi, mencakup sistem *email* dan *web browsers*, menggunakan sistem 8-bit yaitu mereka dapat beroperasi dengan teks yang tepat sesuai dengan ketentuan, seperti ISO-8859-1. ASCII dan EBCDIC sudah mampu mengkodekan 128 karakter, tetapi masih dianggap perlu untuk menciptakan set karakter baru. Lebar set karakter yang cuma 128 karakter tidak memungkinkan penulisan karakter-karakter di luar huruf Latin (*basic Latin)*, seperti misalnya huruf *ü* atau simbol-simbol matematika dalam huruf Yunani.

Set karakter *Unicode* mampu menampung lebih dari satu juta karakter (220 = 1.048.576). Akan tetapi, saat ini hanya 65.535 karakter yang pertama yang mampu direpresentasikan pada komputer (65.535 karakter pertama dari *Unicode* sering disebut dengan istilah UCS-2 atau Universal Character Set-2). Karakter 0 sampai dengan 65.535 menampung karakter-karakter dari alfabet-alfabet yang belum punah (Latin, Kanji, Devanagari, dan lain sebagainya) sedangkan karakter 65.536 sampai dengan 1.048.575 menampung karakter-karakter dari alfabet-alfabet yang sudah punah (misalnya *hieroglyph* dan beberapa karakter Cina yang sangat jarang digunakan).

Terdapat lebih dari 256 karakter di dunia sepert kode *cyrillic, hebrew, arabic, chinese, japanese, korean dan thailand*, dan karakter baru kadang-kadang ditemukan. Dengan menggunakan kode 16-bit (216 = 65.535) diharapkan karakter-karakter tersebut dapat terwakili. Set karakter *Unicode* dialokasikan untuk lebih dari satu alfabet. Bahkan, *Unicode Consortium* menargetkan untuk mengkodekan seluruh alfabet yang ada di dunia. Set karakter *Unicode* ini diharapkan dapat menjadi standar set karakter pada semua komputer di masa depan. Karena *Unicode* mampu merepresentasikan semua alfabet yang ada di dunia ini, maka secara teori seluruh set karakter lainnya tidak diperlukan lagi. *Unicode* mampu mengkodekan berbagai karakteristik alfabet. Mulai dari alfabet Latin yang sederhana, alfabet Arab yang ditulis sambung-menyambung (*cursive*) dari kanan ke kiri, alfabet Cina yang ditulis dari atas ke bawah, dan alfabet India yang memiliki huruf vokal yang letaknya di atas-bawah-depan-belakang dari konsonan.

Oleh karena ke-superior-an dari *Unicode*, maka *Unicode* menjadi set karakter standar yang digunakan pada komunikasi data melalui Internet. *Unicode* memiliki lebar per karakter sebesar 20 bit. Akan menjadi sangat boros jika kita mengirim data *Unicode* yang berisikan teks huruf Latin saja menggunakan 20-bit per karakter. Oleh karena itu, maka *Unicode* perlu ditransformasikan terlebih dahulu menjadi UTF-8 atau UTF-16 (*Unicode* Transformation Format). Dengan UTF-8, maka karakter-karakter pada U+0000 (Notasi U+abcd digunakan untuk mengacu pada karakter bernomor *abcd* pada tabel *Unicode*. Sebagai contoh, U+0053 adalah Latin *Capital Letter S* dan U+0584 adalah *Armenian Small Letter KEH*) sampai dengan U+007F (128 karakter pertama) dapat dikodekan menjadi satu byte saja; sedangkan karakter-karakter lainnya dikodekan dengan menggunakan antara 2 sampai 4 byte per karakter. UTF-8 dan UTF-16 digunakan hanya pada saat komunikasi data saja sedangkan *Unicode* dalam bentuk normal (20 bit) tetap digunakan pada representasi karakter di memori komputer.

Pada dasarnya ada empat cara untuk mengkode karakter *Unicode* yaitu:

1. UTF-8

   128 karakter digunakan mengkode 1 *byte* (karakter ASCII). 1.920 karakter digunakan mengkode 2 *byte* (untuk karakter-karakter Roma, Yunani, Cyrillic, Coptic, Armenian, Ibrani, dan Arab). 63.488 karakter digunakan mengkode 3 byte (Cina dan Jepang). 2.47.418.112 karakter yanglainnya, yang tidak belum digunakan dapat digunakan mengkode 4, 5 atau 6 karakter.
2. UCS-2

   Tiap-tiap karakter direpresentasikan oleh dua byte. Pengkodean ini digunaka untuk merepresentasikan 65.536 karakter *Unicode* yang pertama.
3. UTF-16

   Ini adalah perluasan UCS-2 dimana dapat direpresentasikan 1.112.06 karakter. *Unicode*. 65.536 karakter *Unicode* yang pertama diwakili dua *byte*, yang yang lainnya empat *byte*.
4. UCS-4

   Tiap-tiap karakter direpresentasikan oleh empat *byte*.

Dalam pembangunan aplikasi Kemanan SMS ini, UTF-16 digunakan sebagai pengkodean karakter dengan maksud menyebunyikan hasil enkripsi yang berupa angka-angka ke dalam kode-kode yang masih jarang digunakan agar hasil enkripsi lebih tidak bisa di mengerti dan agar lebih menpersingkat *ciphertext* yang akan dikirimkan.

3.1.5 Contoh Kasus

3.1.5.1 Enkripsi (RSA) Modulus 3 Digit

Contoh kasus kali ini menggunakan (RSA) modulus yang berjumlah 3 digit. Modulus 3 digit yang dimaksud adalah jumlah digit modulus *(n)* yang akan digunakan dalam contoh kasus. *Plaintext* yang akan di enkripsi adalah “RAHMAT”, berikut contoh pengerjaannya:

1. Pembentukan Kunci
   a. Menentukan 2 bilangan prima sembarang dan acak yang diwakilkan oleh variable *p* dan variabel *q*.
      Dimisalkan *p* = 13 dan *q* = 17
   b. Menghitung (RSA) modulus *(n)* dengan formula *n = p * q*, dimana nilai p =13 dan nilai q = 17
      *n = p * q*
      *n = 13 * 17*
      n = 221
   c. Menghitung nilai *m* yang akan digunakan untuk mencari (RSA) *enciphering exponet (e).*
      *m = (p – 1) * (q – 1)*
      *m = (13 – 1) * (17 – 1)*
      *m = (12) * (16)*
      m = 192
   d. Menghitung nilai *e* dengan formula : *gcd(e,m) = 1*, dengan syarat
      e = bilangan prima dan *1 < e < m*.
      dimisalkan e = 5, maka formula *gcd*(5,192) = 1 bernilai *true*.
   e. Menghitung nilai (RSA) *deciphering exponent* (d) dengan menggunakan formula e * d = 1 mod (m), dimana nilai *e* dan *m* didapatkan dari langkah sebelumnya *e* = 5 dan *m* = 192.
      *e * d = 1 mod (m)*
      *d = 1 + (k * m) / e*
      *d = 1 (k * 192) /5*
      dengan nilai k = integer sembarang, maka dimisalkan nilai *d* yang akan diambil adalah d yang bernilai integer. Nilai *d* yang diambil kali ini adalah *d* = 77.
   f. Dari langkah diatas, nilai *n, e,* dan *d* telah ditemukan yang berarti juga pasangan kunci telah terbentuk.
      Pasangan kunci publik *(n,e)* = (221, 5)
      Pasangan kunci rahasia *(n,d)* = (221, 77)

2. Dimisalkan terdapat himpunan karakter : “RAHMAT” yang akan disandikan menggunakan kunci yang telah dibentuk pada langkah sebelumnya, maka terlebih dahulu karakter tersebut dikonversi kedalam bentuk numerik, proses konversi bisa menggunakan teknik tersendiri dari penggunaan atau menggunakan tabel ASCII *decimal* seperti contoh berikut :

Tabel 3.1 Konversi *char* RAHMAT ke ASCII *Decimal*

Char	ASCI Dec
R	82
A	65
H	72
M	77
A	65
T	84

Dari tabel 3.1 hasil konversi setiap baris dijadikan satu deret, akan menghasilkan *plaintext* ASCII *decimal* ***(M)*** dari karakter yang akan dienkrip sebagai berikut : **M = 826572776584**

3. Untuk menjalankan proses enkripsi, digunakan kunci publik yang telah dibentuk sebelumnya yaitu kunci publik (221,5), dengan formula $C = M^e \, mod \, n$. Namun sebelum melakukan perhitungan, terlebih dahulu dilakukan pemecahan deret *plaintext* ASCII menjadi blok yang panjang digit setiap bloknya kurang dari panjang digit *n*. Pada contoh kali ini, (RSA) modulus yang digunakan adalah = 221 (2 digit), jadi untuk setiap blok dibatasi maksimal 2 digit per blok. Dilakukan pemenggalan 3 digit per blok karena karakter yang digunakan dalam proses enkripsi adalah karakter ASCII yang setiap karakternya 8 bit, agar maksimal bit 8 digit (2^3).

   Hasil perhitungan enkripsi dengan formula $C = M^e \, mod \, n$ terlihat pada tabel 3.2 :

Tabel 3.2 Hasil Enkripsi

Blok ke	M	e	n	C
1	82	5	221	114
2	65	5	221	182
3	72	5	221	89
4	77	5	221	25
5	65	5	221	182
6	84	5	221	67

Dari hasil nilai C pada tabel 3.2, dilakukan penyamaan panjang digit setiap blok dengan menambahkan 0 sebanyak kekurangan panjang digit blok tersebut, kemudian jadikan suatu deret, terlihat pada tabel 3.3

Tabel 3.3 Penyamaan Panjang Digit

Blok ke	C
1	114
2	182
3	089
4	025
5	182
6	067

maka akan dihasilkan deret *ciphertext* :

C =114182089025182067

4. Hasil enkripsi telah didapatkan yaitu **C = 114182089025182067**, deret ini akan akan di konversikan ke UTF-16 menggunakan 16-bit untuk menentukan karakter yang berupa simbol-simbol agar lebih tidak bisa dimengerti dan untuk mempersingkat *ciphertext* yang akan dikirim, dengan cara memenggal 3 karakter menjadi satu blok dan dikonversikan ke simbol UTF-16. Dilakukan pemenggalan 3 digit per blok karena karakter yang digunakan dalam proses enkripsi adalah karakter ASCII yang setiap karakternya 8 *bit*, agar maksimal

bit 8 digit (2^3). Setelah dipenggal per 3 karakter, lalu ada nilai yang kurang dari 3 digit maka akan ditambahkan nilai nol (0) di depannya agar menjaga konsistensi panjang setiap blok sehingga tidak terjadi kesalahan pada proses dekripsi. Pemenggalan 3 digit ini dilakukan karena UTF-16 yang digunakan memiliki nilai *decimal* di bawah 1000 dikarenakan UTF-16 diatas decimal 4 karakter tidak dapat terdefinisikan di dalam *smartphone* android. Konversi *ciphertext*(C) ke UTF-16 dengan memenggal setiap 3 digit dari belakang terlihat pada tabel 3.4.

C =114 182 089 025 182 067

Tabel 3.4 Konversi Deret C Ke UTF-16

Block Ke	C	UTF-16
1	114	r
2	182	¶
3	089	Y
4	025	space
5	182	¶
6	067	C

Ketika ada blok yang tidak memiliki karaktek Simbol UTF-16, maka akan digantikan oleh spasi (space) tetapi tetap tidak mempengaruhi ketika akan melakukan proses dekripsi. Merujuk pada tabel 3.4, maka dihasilkan deret ciphertext **C =r¶Y ¶C.** Deret inilah yang nantinya dikirim sebagai *ciphertext.*

Sebetulnya proses enkripsi sudah bisa dikirmkan kepada penerima ketika proses pada nomor 3 selesai, akan tetapi *ciphertext* tersebut terlalu panjang untuk di kirimkan melalui sms kepada penerima, selain itu *ciphertext* yang masih berupa angka-angka mudah dibaca oleh siapapun. Oleh karena itu sebagai sulusinya ciphertext pada no 3 yang masih berupa angka-angka dikonversikan terlebih dahulu ke UTF-16 menggunakan 16 bit untuk menentukan karakter.

3.1.5.2 Dekripsi (RSA) Modulus 3 Digit

Berdasarkan hasil enkripsi maka proses dekripsi dapat dilakukan dengan melakukan koversi UTF-16 terlebih dahulu ke C dalam bentuk decimal.

Tabel 3.5 Konversi Dari UTF-16 Ke C

Block Ke	UTF-16	C
1	r	114
2	¶	182
3	Y	089
4	*(space)	025
5	¶	182
6	C	067

Dari tabel 3.5 dihasilkan deret **C =114182089025182067.** Untuk membuktikan pasangan kunci publik dan kunci rahasia berjalan sempurna, dicoba untuk mendekrip *cipher text* yang telah dienkrip sebelumnya. Proses dekripsi menggunakan kunci rahasia (221, 77), dengan formula $M = C^d \ mod \ n$. Hasil perhitungan untuk proses dekripsi ditunjukkan pada tabel 3.6.

Tabel 3.6 Perhitungan Dekripsi

Blok ke	C	d	n	M
1	114	77	221	82
2	182	77	221	65
3	89	77	221	72
4	25	77	221	77
5	182	77	221	65
6	67	77	221	84

Jika hasil perhitungan dekripsi (nilai M) dijadikan deret akan terbentuk kembali deret *plaintext* ASCII yang sama antara deret sebelum dienkrip dan setelah didekrip yaitu :

M = 826572776584

Langkah terakhir adalah mengubah deret *plaintext* ASCII menjadi *plaintext* menggunakan tabel ASCII sebagai pedoman konversi.

Tabel 3.7 Konversi ASCII ke Char

ASCII Dec	Char
82	R
65	A
72	H
77	M
65	A
84	T

Pembentukan nilai *e* dengan formula *gcd(e,m) = 1* , yang berarti nilai FPB antara kedua variabel *e* dan *m* harus bernilai 1 atau kedua bilangan tersebut harus relatif prima. Syarat *e* adalah *1 < e < m*, sehingga untuk mendapatkan nilai *e* dimiliki kemungkinan lebih dari 1, karena kemungkinan bilangan prima dari 1 hingga m bisa lebih dari 1.

Sedangkan dalam pembentukan nilai d dengan formula akhir *"**d = 1 + (k * m) / e**"* dimungkinkan ditambahkan aturan lain untuk mendapatkan nilai *d*. Variabel *k* adalah nilai integer sembarang, yang jika dimasukkan kedalam formula akan menghasilkan nilai *d*, namun nilai *d* yang memenuhi syarat adalah nilai *d* yang integer. Sehingga dalam proses penentuan nilai *d* dengan menggunakan nilai *k* sebagai integer sembarang, dapat ditambahkan aturan misalnya dalam penentuan *range* percobaan nilai *k*, atau nilai *d* yang akan diambil adalah *d* yang bernilai integer yang setelah berapa kali ditemukan dan lain sebagainya.

Salah satu syarat wajib yang menjadi perhatian dalam proses enkripsi dan dekripsi adalah nilai *M* atau *Message* atau *Number* harus lebih kecil jika dibandingkan nilai Modulus, yang berarti untuk membuatnya berada dibatas aman (selalu lebih kecil) ditetapkan jumlah digit maksimal nilai *M* atau *Message* atau *Number* adalah jumlah digit nilai modulus dikurang 1.

Untuk mengubah *string* menjadi bilangan tidak diharuskan menggunakan tabel konversi tertentu sehingga pengguna dapat menerapkan tabel konversi sendiri sesuai dengan kebutuhan pengguna.

Pada akhirnya, untuk mengimplementasi algoritma kriptografi asimetris RSA dimungkinkan untuk diterapkan banyak kondisi dan batasan yang menyesuaikan dengan kebutuhan pengguna, sehingga memungkinkan dalam penerapan RSA terbentuk algoritma yang berbeda dari tiap penyusun sintak diluar algoritma RSA yang utama. Hal ini memungkinkan untuk menjadi alasan tidak dapat dibentuk *function* khusus untuk algoritma kriptografi asimetris RSA karena banyaknya sisi algoritma yang tidak mutlak.

3.1.6 Analisis SMS

Short Message Service (SMS) merupakan pesan singkat yang memungkinkan pengiriman dan penerimaan pesan singkat antara *mobile station*. Dalam satu SMS terdiri dari 140 octets yang sama dengan 1120 bits data sehingga dalam 1 SMS berisikan :

1. 160 karakter apabila menggunakan pengkodean 7 *bits* per karakter, pengkodean ini sesuai untuk pengkodean karakter Latin seperti huruf bahasa Inggris.
2. 70 karakter apabila menggunakan pengkodean 16 bits *Unicode* UCS2, pengkodean ini digunakan untuk karakter selain Latin seperti huruf Cina, Arab, Jepang.

3.1.6.1 Analisis Pengiriman Paket Data SMS

Analisis pengiriman paket data sms adalah analisis paket data yang akan dikirimkan kepada penerima pesan. Paket data yang dikirim ini merupakan *ciphertext*. Untuk contoh, pada tabel 3.4, telah didapatkan *ciphertext* yang akan dikimkan kepada penerima yaitu : **r¶Y ¶C** yaitu hasil enkripsi dari kata “RAHMAT”**.** Satu paket pengiriman *ciphertext* adalah 70 karakter dikarenakan *ciphertext* menggunakan pengkodean 16 bits *Unicode*. Berikut tabel paket yang dikirimkan :

Tabel 3.8 Paket Data Yang Kirimkan

Data yang dikirm	**r¶Y ¶C**
Panjang Karakter	6
Jumlah Bit	96

Paket data yang akan dikirimkan adalah **r¶Y ¶C**, kaerena pesan yang dikirmkan berupa pengkodean 16 bits *Unicode* maka panjang karakter maksimal dalm 1 sms adalah 70 karakter. Berikut tabel konversi yang menggambarkan pesan *ciphertext* 16 bit *Unicode* "**r¶Y ¶C"**, ASCII Decimal dan Biner dari *plaintext* "RAHMAT" yang sudah disandikan sebelumnya :

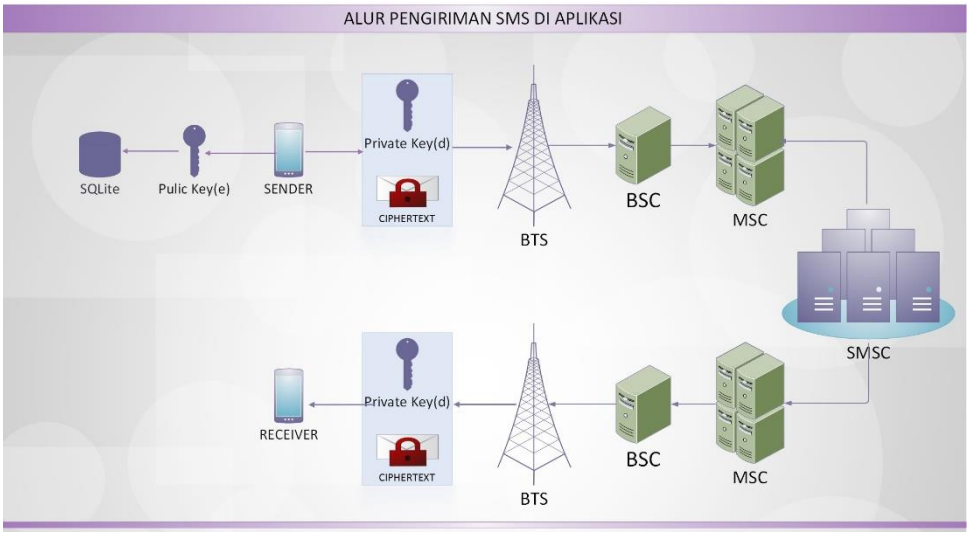
Tabel 3.9 Konversi *Plaintext* RAHMAT

Plaintext	**ASCII Dec**	**Biner**	***Ciphertext* 16 bit *Unicode***
R	82	0000000001110010	**r**
A	65	0000000010110110	**¶**
H	72	0000000001011001	**Y**
M	77	0000000000011001	space
A	65	0000000010110110	**¶**
T	84	0000000001000011	**C**

Jumlah bits SMS yang dikirimkan berdasarkan contoh *ciphertext* **r¶Y ¶C** adalah 96 bits, karena karakter UTF-16 jumlah *bits* perkarakternya adalah 16 bits, maka 6 karakter dikalikan 16 bits perkaraternya = 96 bits, (6 x 16 = 96). Dilihat dari jumlah bits tersebut jumlah SMS yang digunakan untuk mengirimkan pesan *ciphertext* **r¶Y ¶C** adalah sebanyak 2 SMS, karena panjang SMS dalam karakter *unicode* adalah 70 karakter.

Pengiriman SMS di Aplikasi Keamanan SMS ini tidak seperti pengiriman SMS biasa dikarenakan pesan yang dikirimkan bukan text seperti biasa melainkan pesan yang terenkripsi dalam bentuk pengkodean 16 bits *Unicode*. Dalam proses pengiriman pesan terenkripsi sebenarnya hampir mirip dengan pengiriman pesan biasa, hanya bedanya di aplikasi keamanan SMS ini ketika menekan tombol send akan mengirimkan 2 pesan sekaligus yaitu pesan terenkripsi dan *modulus*. Dalam

proses penerimaan terdapat 2 pesan masuk yang masuk ke dalam ponsel yaitu di pesan masuk di aplikasi bawaan ponsel dan pesan masuk di aplikasi Keamanan SMS. Pesan masuk masuk di aplikasi bawaan ponsel yaitu pesan yang masih terenkripsi, sedangkan pesan masuk di aplikasi kemanan SMS adalah pesan yang sudak terdekripsi secara otomatis ketika kita memilih pesan masuk tersebut. Berikut gambaran proses pengiriman dan penerimaan SMS :



Gambar 3.4 Proses Pengiriman SMS

Proses pengiriman SMS yang di gambarkan oleh gambar 3.4 terbagi menjadi beberapa tahapan sebagai berikut :

1. Pengirim mengirimkan SMS yang berisi *ciphertext* atau *private key*.
2. Pesan dikirim ke MSC melalui jaringan sellular yang tersedia meliputi tower BTS yang meng-*handle* komunikasi pengguna, lalu ke BSC.
3. MSC kemudian mem-*forward* lagi SMS ke SMSC untuk disimpan, lalu kemudian mengecek (lewat HLR – *Home Location Register*) untuk mengetahui apakah *handphone* tujuan sedang aktif dan dimanakah *handphone* tujuan tersebut. Jika *handphone* sedang tidak aktif maka pesan tetap disimpan di SMSC itu sendiri
4. MSC memberitahukan bahwa handphone sudah aktif kembali untuk kemudian SMS dikirim dengan batas maksimum waktu tunggu yaitu *validity period* dari pesan SMS itu sendiri. Jika *handphone* tujuan aktif

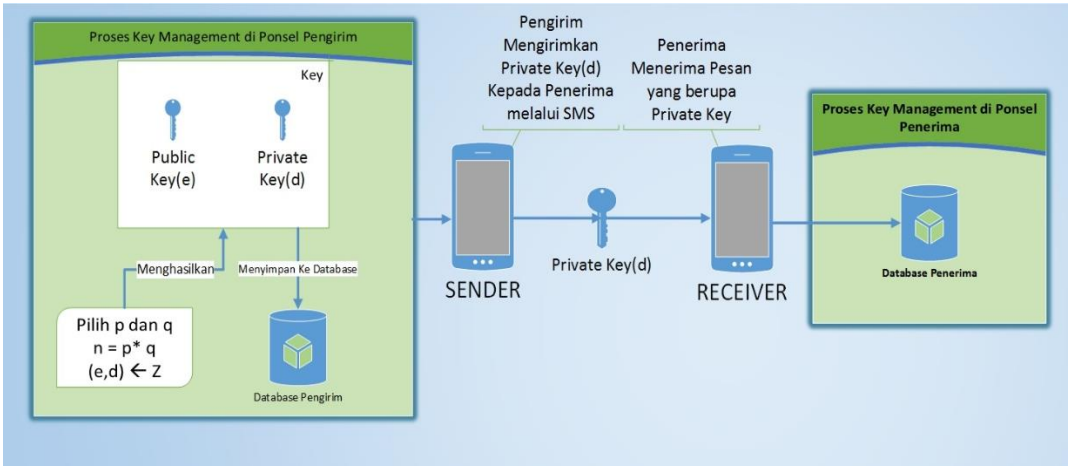
maka pesan disampaikan MSC lewat jaringan yang sedang meng-*handle* penerima (BSC dan BTS).

5. Penerima menerima pesan yang dikirimkan oleh pengirim berupa *ciphetext* atau *private key*.

3.1.7 Analisis *Key Management*

Key Management adalah sebuah manajemen kunci di dalam sebuah sistem kriptografi. *Key management* ini berurusan dengan *generat*e, pertukaran, penyimpanan, dan penggunaan kunci. *Key management* ini berkaitan dengan pengguna, baik antara pengguna dengan pengguna atau pengguna dengan sistem. *Key management* ini berbeda dengan penjadwalan kunci, karena penjadwalan kunci biasanya mengacu pada penanganan material kunci dalam operasi ciphernya. *Management key* penting dalam sebuah sistem kriptografi karena kesuksesan sebuah sistem kriptografi tergantung pada pengelolaan *key management*.

Pada aplikasi kemanan SMS ini juga terdapat *key management*, dimana *key management* ini dibuat untuk mengamankan kunci yang akan dikirimkan pengirim pesan kepada penerima pesan. *Key management* di aplikasi kemanan sms ini dilakukan dengan cara autentifikasi nomor telepon, maksudnya setiap penerima pesan mendapatkan *private key* yang berbeda-beda. Berikut gambaran *key management* di apliasi keamanan SMS :



Gambar 3.5 *Management Key*

Proses dari gambar 3.5 dijelaskan sebagai berikut :

1. Sistem menentukan 2 bilangan prima sembarang dan acak yang diwakilkan oleh variable *p* dan variabel *q*.
2. Sistem menghitung (RSA) modulus *(n)* dengan formula $n = p * q$.
3. Sistem menghitung nilai *m* yang akan digunakan untuk mencari (RSA) *enciphering exponet (e).*
4. Sistem menghitung nilai *e* dengan formula $gcd(e,m) = 1$, dengan syarat *e* = bilangan prima dan $1 < e < m$.
5. Sistem menghitung nilai (RSA) deciphering exponent (d) dengan menggunakan formula e * d = 1 mod (m).
6. Maka menghasilkan nilai n, e, dan d telah ditemukan yang berarti juga pasangan kunci telah terbentuk yaitu *private key* dan *public key*.
7. Pengirim menyimpan kunci kedalam *database*.
8. *Private key* dikirimkan kepada penerima pesan melalui sms.
9. Penerima menerima pesan sms yang berupa kunci lalu menyimpannya ke dalam database berdasarkan nomor ponsel pengirim.

3.1.8 Analisis Kebutuhan non-Fungsional

Analisis kebutuhan non-fungsional adalah sebuah langkah dimana seorang pembangun aplikasi menganalisis sumber daya yang dibutuhkan untuk menggunakan aplikasi yang akan dibangun. Analisis kebutuhan nonfungsional yang dilakukan dibagi dalam tiga tahap, yaitu analisis pengguna (*user*), analisis kebutuhan perangkat keras, dan analisis perangkat lunak.

3.1.8.1 Analisis Pengguna

Pengguna yang akan menggunakan aplikasi keamanan SMS ini dibagi menjadi dua bagian yaitu bagian pengirim dan penerima pesan dengan kebutuhan spesifikasi seperti pada tabel 3.10.

Tabel 3.10 Kebutuhan Pengguna Aplikasi

Tipe Pengguna	Hak Akses	Tingkat Keterampilan	Jenis Pelatihan
Pengirim	Dapat melakukan operasi enkripsi pesan.	Dapat mengoprasikan ponsel bersistem, operasi android dan memahami konsep kriptografi.	Tidak diperlukan pelatihan khusus.
Penerima	Dapat melakukan operasi deskripsi pesan.	Dapat mengoprasikan ponsel bersistem, operasi android dan memahami konsep kriptografi.	Tidak diperlukan pelatihan khusus.

3.1.8.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras merupakan salah satu kebutuhan yang sangat penting bagi pembuatan aplikasi Keamanan SMS ini. Perangkat keras akan mempengaruhi kinerja dari pembuatan Aplikasi Keamanan SMS ini, semakin tinggi spesifikasi dari perangkat keras yang digunakan maka akan semakin cepat pula pembuatan aplikasinya. Perangkat keras yang digunakan pada pembangunan aplikasi Keamanan SMS ini yaitu seperti pada tabel 3.11.

Tabel 3.11 Spesifikasi Perangkat Keras

No.	Nama Perangkat	Spesifikasi
1	Processor	Intel Core i3 1.40 GHz
2	Monitor	Monitor 14 inch (1366 x 768)
3	Memori	RAM 2 GB DDR3
4	Hardisk	320 GB SATA 7200 RPM

Sedangkan kebutuhan perangkat keras untuk menjalankan aplikasi yang dibangun harus dipebuhi yaitu seperti pada tabel 3.12.

Tabel 3.12 Spesifikasi Perangkat Keras Minimal

No.	Nama Perangkat	Spesifikasi
1	Processor	Intel Core i3 1.40 GHz
2	Monitor	Monitor 14 inch (1366 x 768)
3	Memori	RAM 1 GB DDR3
4	Hardisk	250 GB SATA 7200 RPM

3.1.8.3 Analisis Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun aplikasi Keamanan SMS ini yaitu seperti pada tabel 3.13.

Tabel 3.13 Spesifikasi Perangkat Lunak

No.	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Windows 8 Profesional 32 bit
2	JDK (Java Development Kit)	JDK 1.7.0
3	JRE (Java Runtime Environment)	JRE 7
4	Eclipse	Juno
5	ADT	22.2.1

Sedangkan kebutuhan perangkat lunak minimal untuk menggunakan aplikasi Keamana SMS yaitu seperti pada tabel 3.14.

Tabel 3.14 Spesifikasi Perangkat Lunak Minimal

No.	Nama Perangkat	Spesifikasi
1	Sistem Operasi	Android 2.2 Froyo (Frozen Yoghurt)
2	Memori	384 MB
3	Monitor	240 x 320 pixels
4	Hardisk	160 MB

3.1.9 Analisis Spesifikasi dan Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak adalah analisis spesifikasi prangkat lunak yang akan dibangun. Spesifikasi perangkat lunak dibutuhkan untuk menetapkan layanan apa yang akan dituntut pada oprasi dan pengembangan perangkat lunak. Berikut spesifikasi perangkat lunak yang akan dibangun:

1. Ketika pertama kali menginstal aplikasi pengguna diminta membuat *password* untuk masuk aplikasi.
2. Sebelum masuk ke menu utama aplikasi, pengguna harus memasukan *password* terlebih dahulu.
3. Aplikasi dapat melakukan proses enkripsi dan dekripsi pesan.
4. Aplikasi dapat menggenerate *private key* dan *public key* untuk enkripsi dan dekripsi pesan.
5. Aplikasi dapat melakukan pengiriman pesan.

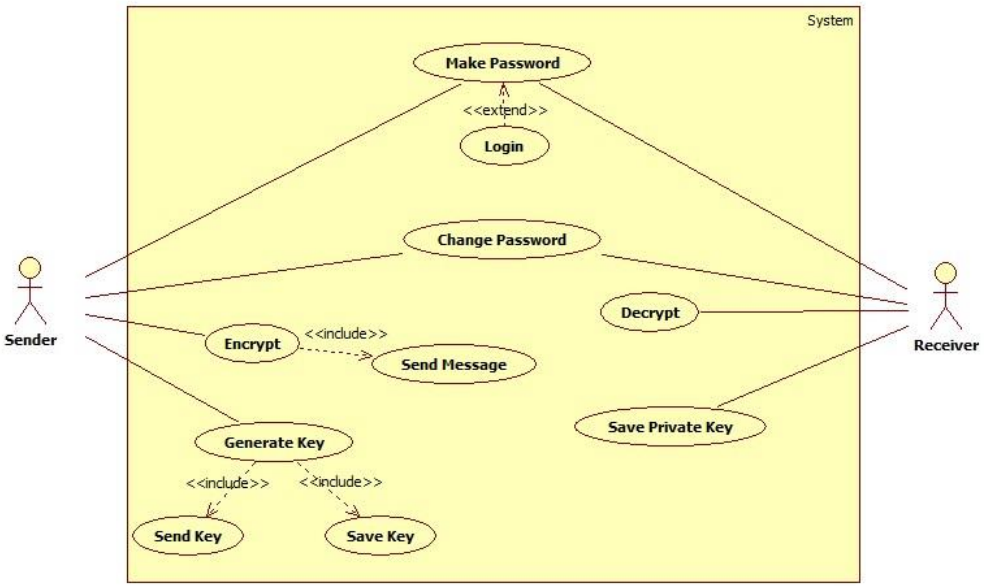
6. Pesan yang dikirmkan berupa *ciphertext.*
7. Dalam aplikasi terdapat *inbox* pesan.
8. Ketika mendekripsi pesan, penerima pesan tidak harus memasukan *private key* karena *private ke*y sudah otomatis masuk menurut nomor telepon pengirim.

3.1.10 Analisis Kebutuhan Fungsional

Analisi kebutuhan fungsional adalah segala bentuk data yang dibutuhkan oleh sistem agar sistem dapat berjalan sesuai dengan prosedur yang dibangun. Aplikasi yang dibangun akan dimodelkan menggunakan *Unified Modeling Language* (UML), dan *tools* yang akan digunakan yaitu *use case* diagram, *activity* diagram, *sequence* diagram, dan *class* diagram.

3.1.10.1 Use Case Diagram

Use case diagram digunakan untuk mengetahui apa saja yang dapat dilakukan oleh pengguna/aktor terhadap fungsionalitas yang terdapat pada aplikasi yang dibangun. *Use case* diagram pada aplikasi kemanan SMS terlihat pada gambar 3.6.



Gambar 3.6 *Use Case Diagram* Aplikasi Keamanan SMS

3.1.10.2 *Skenario* Use Case

Skenario *use case* merupakan penjelasan secara rinci setiap *use case* yang terdapat pada diagram *use case*. Skenario *use case* pada aplikasi ini yaitu sebagai berikut:

1. **Skenario *Make Password***

Skenario *make password* dapat dilihat pada tabel 3.15.

Tabel 3.15 Skenario Buat *Password*

Use case name	*Make Password*
Actor	*Sender* atau *Receiver (User)*
Use case description	Digunakan untuk membuat *password* yang digunakan untuk masuk kembali ke dalam aplikasi sebelum masuk menu utama aplikasi
Precondition	Sistem menampilkan *make password*
Trigger	*User* menekan *button make password*
Scenario :	
1. *User* memasukan *password*. 2. *User memasukan* konfirmasi *password.* 3. *User* menekan button buat *password*. 4. Sistem menyimpan *password* ke dalam *database*.	
Exception :	
1. *User* menekan tombol *back*, buat password dibatalkan. 2. *User* mengetikan password kurang dari 6 karakter. 3. *User* mengetikan konfirmasi password yang salah.	

2. Skenario *Login*

Skenario *login* dapat dilihat pda tabel 3.16.

Tabel 3.16 Skenario *Login*

Use case name	*Login*
Actor	*Sender* atau *Receiver* (*User*)
Use case description	Digunakan untuk memasukan *password* sebelum masuk menu utama aplikasi
Precondition	Sistem menampilkan *form login.*
Trigger	*Sender* menekan button *login*.
Scenario :	
1. *User* memasukan *password*. 2. *User* menekan *button login*. 3. Sistem melakukan proses verifikasi *password.*	
Exception :	
1. *User* menekan tombol *back, login* dibatalkan. 2. *User* salah memasukan *password.*	

3. Skenario *Generate Key*

Skenario *generate key* dapat dilihat pada tabel 3.17.

Tabel 3.17 Skenario *Generate Key*

Use case name	*Get Key*
Actor	*Sender*
Use case description	Digunakan untuk membangkitkan pasangan kunci.
Precondition	Sistem menampilkan *form Key Management.*
Trigger	*Sender* menekan *button* Bangkitkan Kunci.
Scenario :	
1. Sender menekan button Manajemen Kunci di menu utama. 2. Sistem menampilkan *form* Manajemen Kunci. 3. *Sender* memasukan nomor telepon penerima. 4. *Sender* menekan *button* Bangkitkan Kunci	
Exception :	

1. Pengirim menekan tombol *back, Generate Key* dibatalkan.

4. Skenario *Save Key*

Skenario save key dapat dilihat pada tabel 3.18.

Tabel 3.18 Skenario *Save Key*

Use case name	*Save Key*
Actor	*Sender*
Use case description	Digunakan untuk menyimpan kunci yang telah dibangkitkan ke dalam *database*.
Precondition	Sistem menampilkan *form Key Management.*
Trigger	*Sender* menekan *button* Simpan Kunci.
Scenario :	
1. Sender menekan button Manajemen Kunci di menu utama. 2. Sistem menampilkan *form* Manajemen Kunci. 3. *Sender* memasukan nomor telepon penerima. 4. *Sender* menekan *button* Simpan Kunci	
Exception :	
1. Pengirim menekan tombol *back, Save Key* dibatalkan.	

5. Skenario *Send Key*

Skenario *send key* dapat dilihat pada tabel 3.19.

Tabel 3.19 Skenario *Send Key*

Use case name	*Send Key*
Actor	*Sender*
Use case description	Digunakan untuk mengirimkan kunci yang sudah dibangkitkan di *database*.
Precondition	Sistem menampilkan *form Key Management.*
Trigger	*Sender* menekan *button* Ambil Kunci.
Scenario :	
1. Sender menekan button Manajemen Kunci di menu utama. 2. Sistem menampilkan *form* Manajemen Kunci. 3. *Sender* memasukan nomor telepon penerima. 4. *Sender* menekan *button* Kirim Kunci.	

Exception :
1. Pengirim menekan tombol *back, Send Key* dibatalkan.

6. Skenario *Encrypt*

Skenario *encrypt* dapat dilihat pada tabel 3.20.

Tabel 3.20 Skenario *Encrypt*

Use case name	*Encrypt*
Actor	*Sender*
Use case description	Digunakan untuk menyadikan pesan.
Precondition	Sistem menampilkan form *Create New Message.*
Trigger	*Sender* menekan tombol *encrypt.*
Scenario :	
1. *Sender* menekan tombol Pesan Baru. 2. Sistem menampilkan form *Cretae New Message.* 3. Sender mengetikan pesan. 4. *Sender* memilih menekan *button encrypt.*	
Exception :	
1. Pengirim menekan tombol *back, encrypt* dibatalkan.	

7. Skenario *Send Message*

Skenario kirim pesan dapat dilihat pada tabel 3.21.

Tabel 3.21 Skenario *Send Ciphertext*

Use case name	*Send Message*
Actor	*Sender*
Use case description	Digunakan untuk mengirim pesan yang telah dienkripsi.
Precondition	Sistem menampilkan *form Create New Message.*
Trigger	*Sender* menekan tombol Kirim.
Scenario :	
1. *Sender* menekan *button* Pesan Baru. 2. Sistem menampilkan *form Create New Message.* 3. *Sender* mengetikan pesan. 4. *Sender* menekan *button encrypt*.	

5. *Sender* menekan *button* Kirim.
Exception :
1. *Sender* menekan tombol *back, Send Message* dibatalkan.

8. Skenario *Save Private Key*

Skenario *save private key* dapat dilihat pada tabel 3.22.

Tabel 3.22 Skenario *Save Private Key*

Use case name	*Save Private Key*
Actor	*Receiver*
Use case description	Digunakan untuk menyimpan kunci privat yang dikirimkan oleh *sender*.
Precondition	Sistem menampilkan *list inbox.*
Trigger	*Sender* memilih pesan masuk.
Scenario :	
1. *Receiver* menekan *button* Pesan Masuk. 2. Sistem menampilkan *list Inbox.* 3. *Receiver* memilih kunci yang akan disimpan. 4. Sistem menampilkan menu *detail inbox*. 5. *Receiver* menekan tombol *Save Key.*	
Exception :	
1. *Receiver* menekan tombol *back, Save Key* dibatalkan.	

9. Skenario *Decrypt*

Skenario *Decrypt* dapat dilihat pda tabel 3.23.

Tabel 3.23 Skenario *Decrypt*

Use case name	*Decrypt*
Actor	*Receiver*
Use case description	Digunakan mendekripsi pesan
Precondition	Sistem menampilkan pesan yang akan didekripsi.
Trigger	Pengirim menekan tombol *decrypt*
Scenario :	

1. *Receiver* memilih pesan yang akan didekripsi. 2. Sistem menampilkan pesan yang akan di dekripsi. 3. *Receiver* menekan *button decrypt*.
Exception :
1. *Receiver* menekan tombol *back, decrypt* dibatalkan.

10. Skenario *Change Password*

Skenario *change password dapat dilihat pada tabel 3.24.*

Tabel 3.24 Skenario *Change Password*

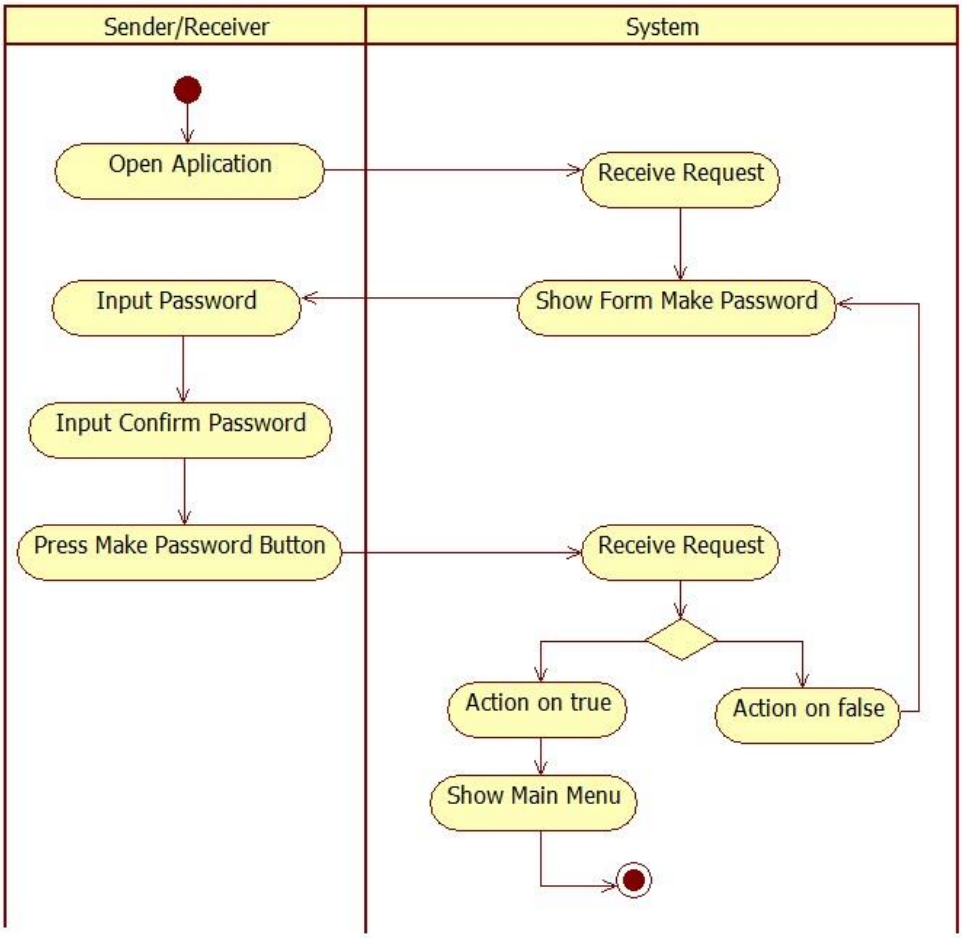
Use case name	*Change Password*
Actor	*Receiver* atau *Receiver (User)*
Use case description	Digunakan untuk mengganti password.
Precondition	Sistem menampilkan *form change password.*
Trigger	Pengirim menekan tombol *change password.*
Scenario :	
1. *User* menekan button *change password*. 2. Sistem menampilkan form change password. 3. *User* memasukan *password* lama, *password* baru, dan *confirm password*.	
Exception :	
1. *User* menekan tombol *back, change password* dibatalkan.	

3.1.10.3 Activity Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram yang terdapat pada aplikasi yang dibangun yaitu sebagai berikut :

1. Activity Diagram Make Password

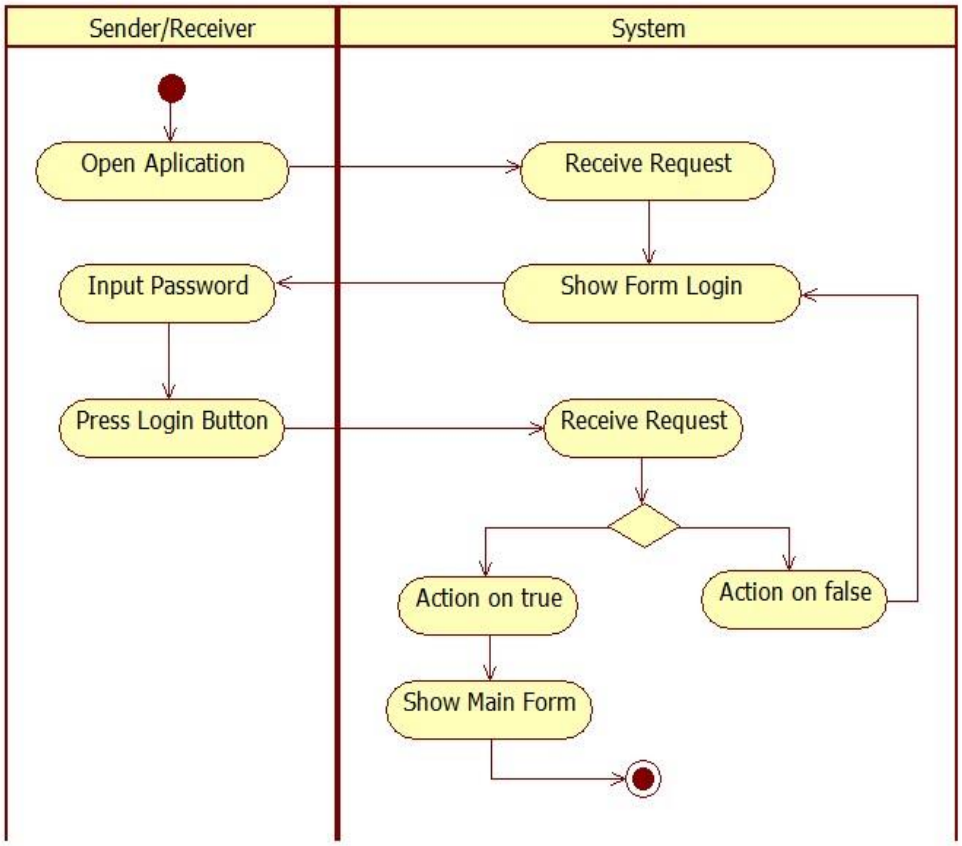
Activity diagram make password menggambarkan alur aktivitas pembuatan *password* antara pengirim/penerima (*user*) dengan sistem seperti terlihat pada gambar 3.7.



Gambar 3.7 ***Activity Diagram Make Password***

2. *Activity Diagram Login*

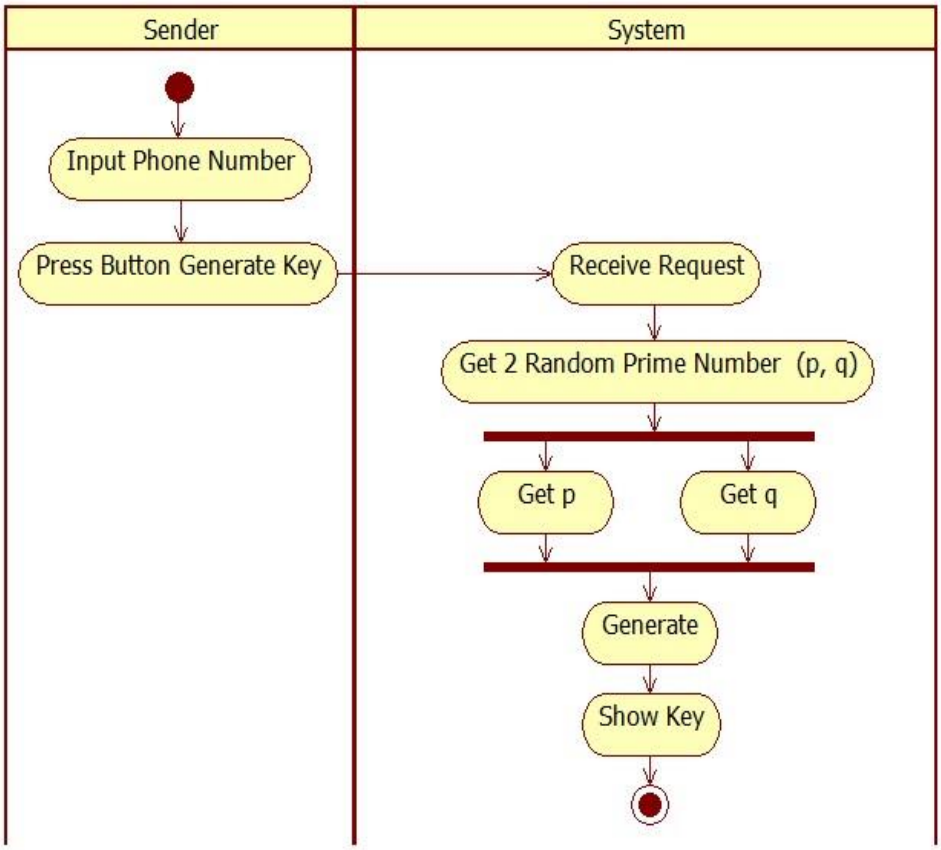
Activity login menggambarkan alur aktivitas input password antara pengirim/penerima pesan (*user*) dengan sistem seperti terlihat pada gambar 3.8.



Gambar 3.8 ***Activity Diagram Login***

3. *Activity Diagram Generate Key*

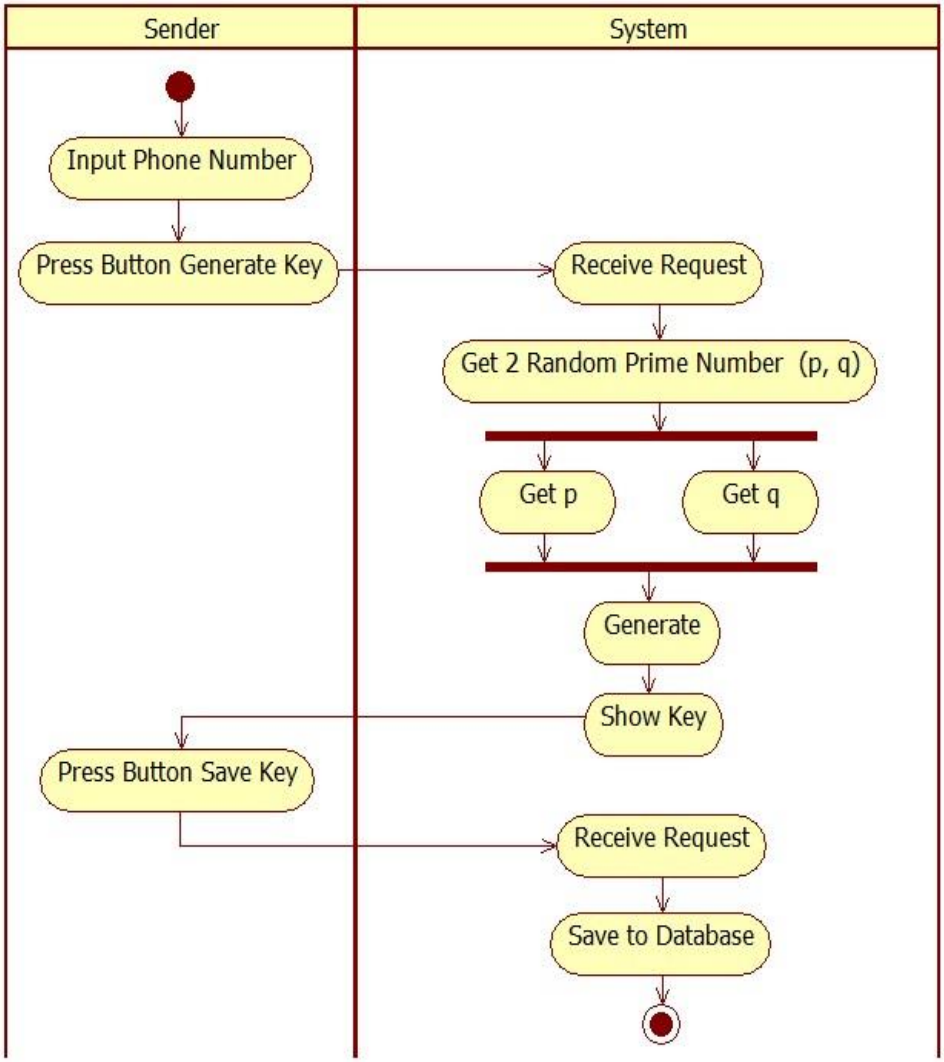
Activity diagram *generate key* mengambarkan alur aktivitas membangkitkan sepasang kunci dengan mangambil 2 bilangan prima besar secara acak seperti terlihat pada gambar 3.9.



Gambar 3.9 *Activity Diagram Generate Key*

4. *Activity Diagram Save Key*

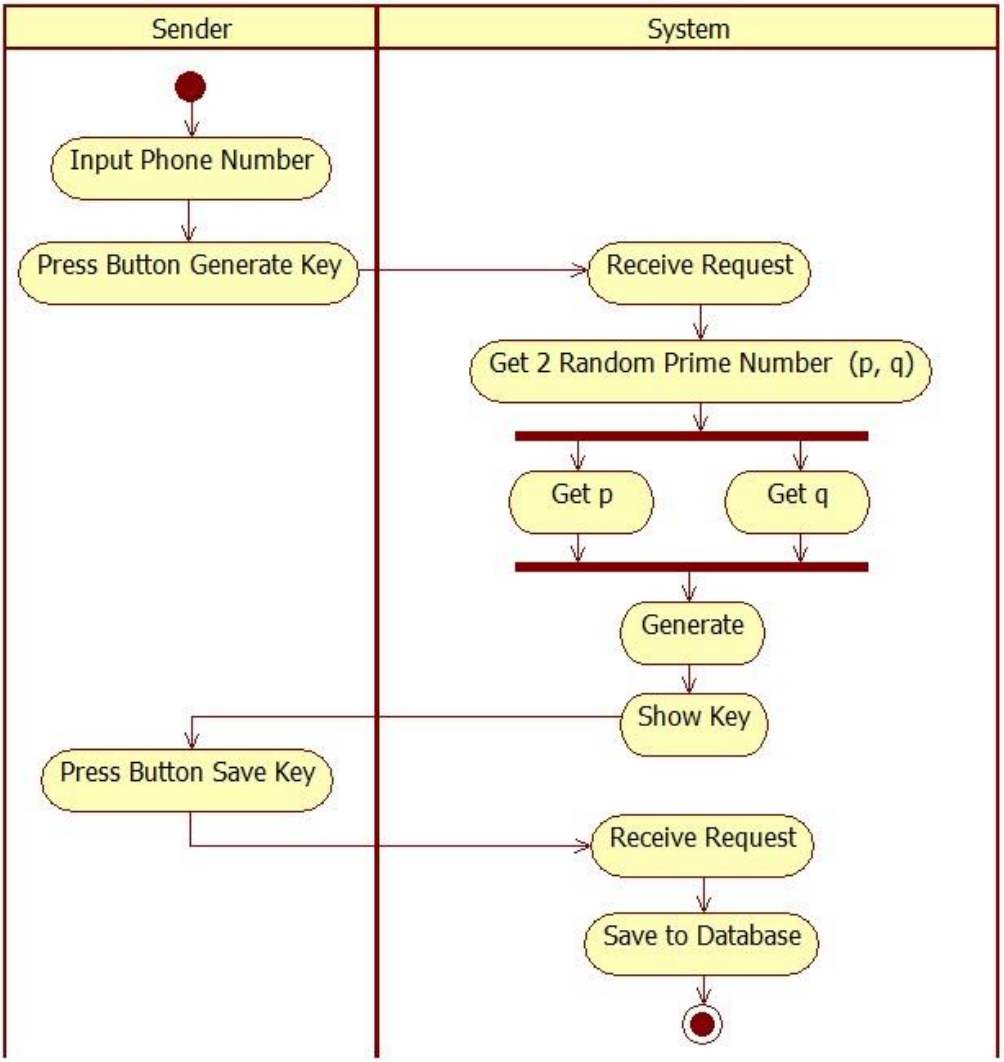
Activity diagram save key menggambarkan alur aktivitas menyimpan kunci yang telah dibangkitkan ke dalam database seperti terlihat pada gambar 3.10.



Gambar 3.10 *Activity Diagram Save Key*

5. *Activity Diagram Send Key*

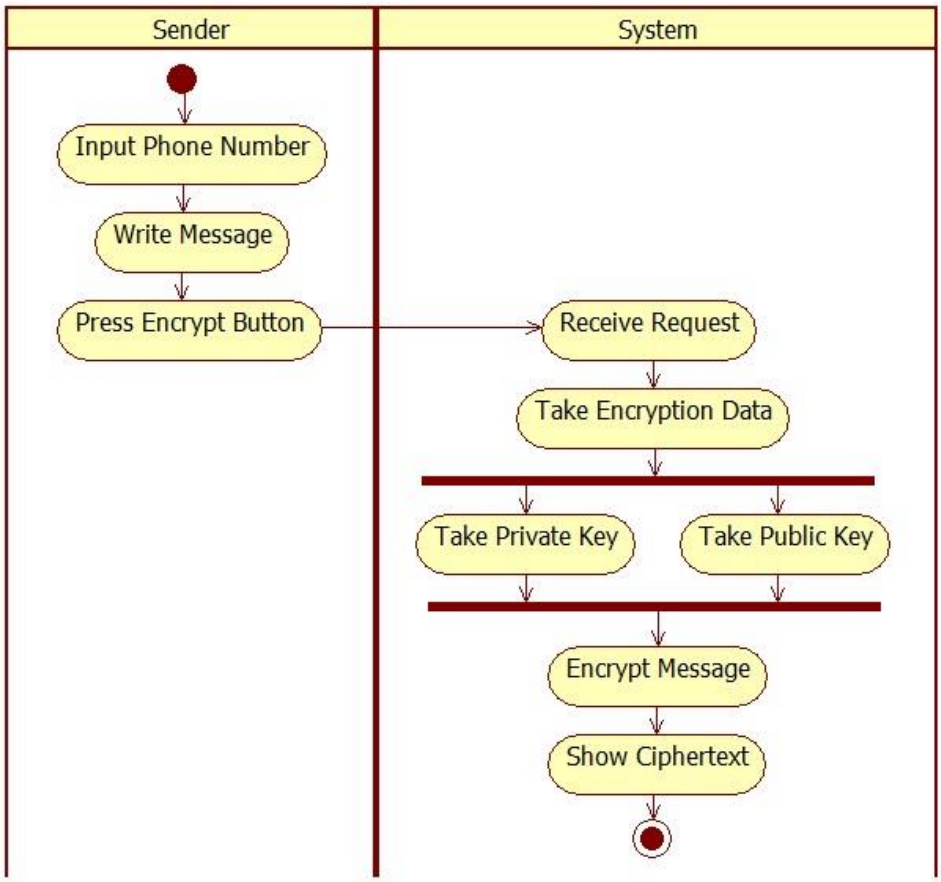
Activity diagram send key menggambarkan alur aktivitas mengirim pesan yang telah dienkripsi kepada penerima seperti terlihat pada gambar 3.11.



Gambar 3.11 *Activity Diagram Send Key*

6. *Activity Diagram Encrypt*

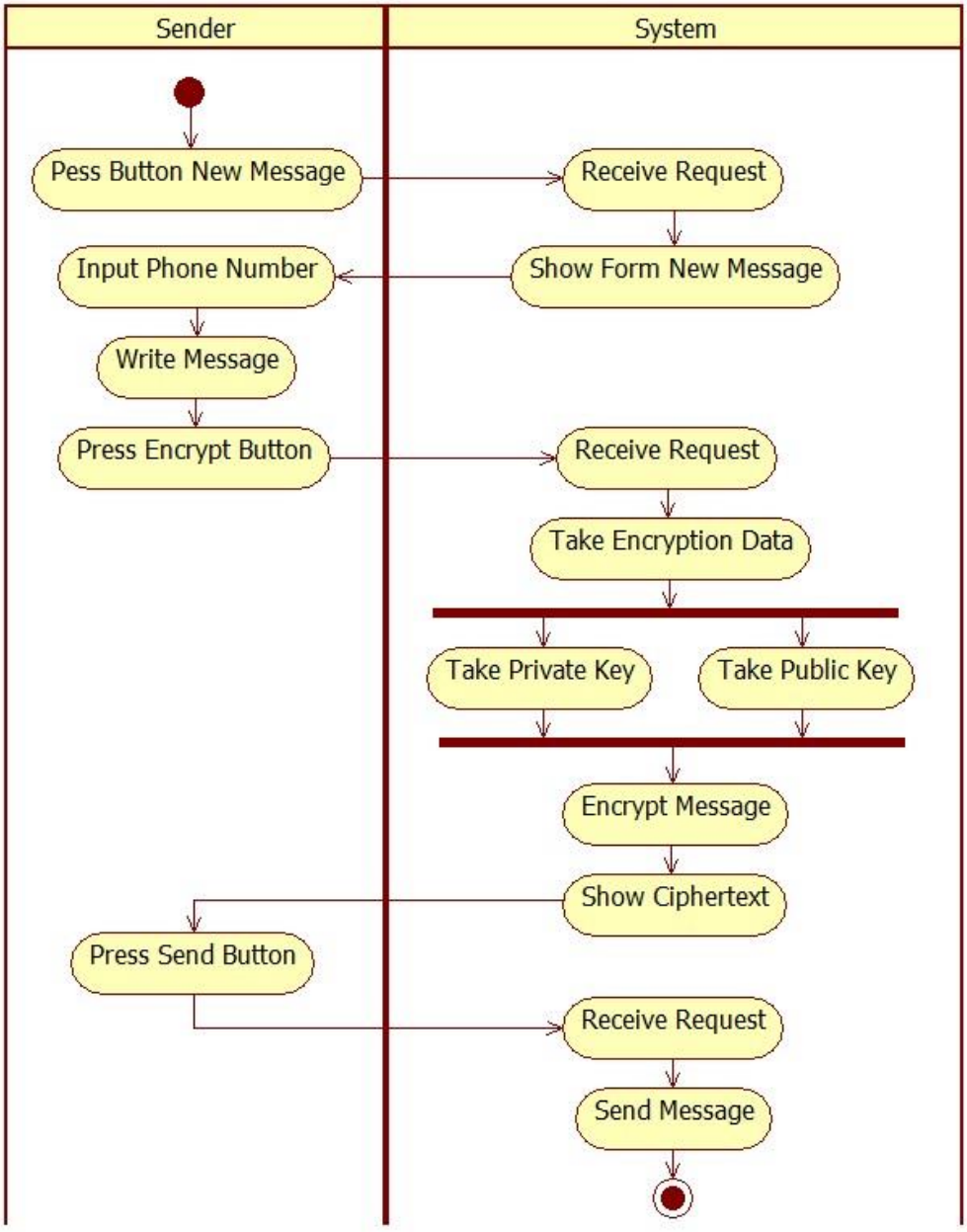
Activity diagram encrypt menggambarkan alur aktivitas enkripsi pesan antara pengirim dengan sistem seperti terlihat pada gambar 3.12.



Gambar 3.12 *Activity Diagram Encrypt*

7. *Activity Diagram Send Message*

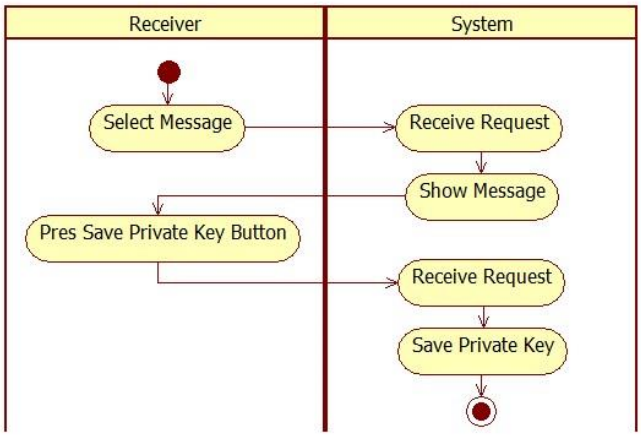
Activity diagram encrypt menggambarkan alur aktivitas enkripsi pesan antara pengirim dengan sistem seperti terlihat pada gambar 3.13.



Gambar 3.13 *Activity Diagram Send Message*

8. *Activity Diagram Save Private Key*

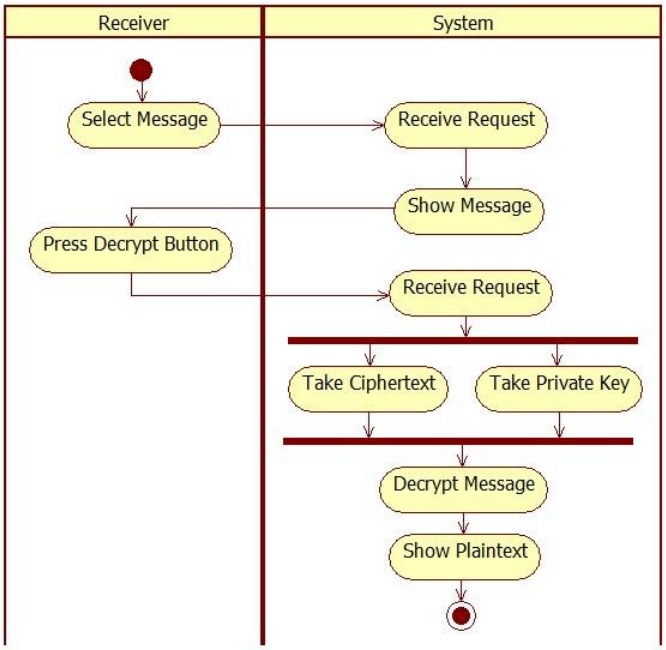
Activity diagram save private key menggambarkan alur aktivitas menyimpan kunci privat yang dikirimkan antara pengirim dengan sistem seperti terlihat pada gambar 3.14.



Gambar 3.14 *Activity Diagram Save Private Key*

9. *Activity Diagram Decrypt*

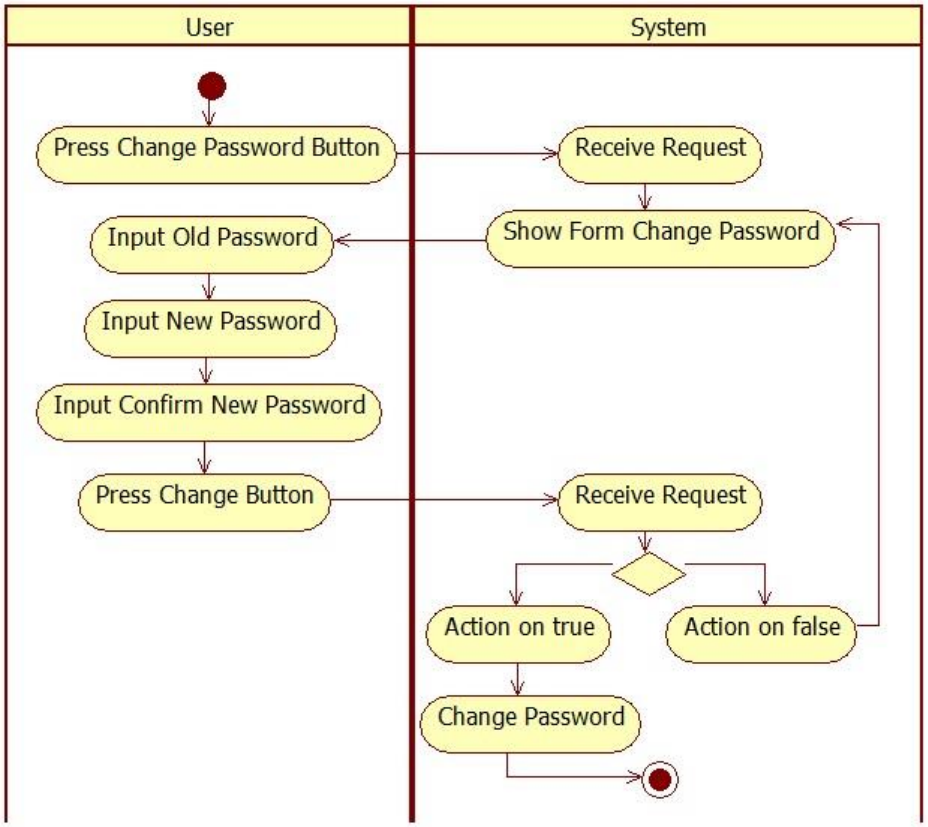
Activity diagram decrypt menggambarkan alur aktivitas dekripsi pesan antara penerima dengan sistem seperti terlihat pada gambar 3.15.



Gambar 3.15 *Activity Diagram Decrypt*

10. *Activity Diagram Change Password*

Activity diagram *Decrypt* menggambarkan alur aktivitas merubah password antara penerima/pengirim dengan sistem seperti terlihat pada gambar 3.16.



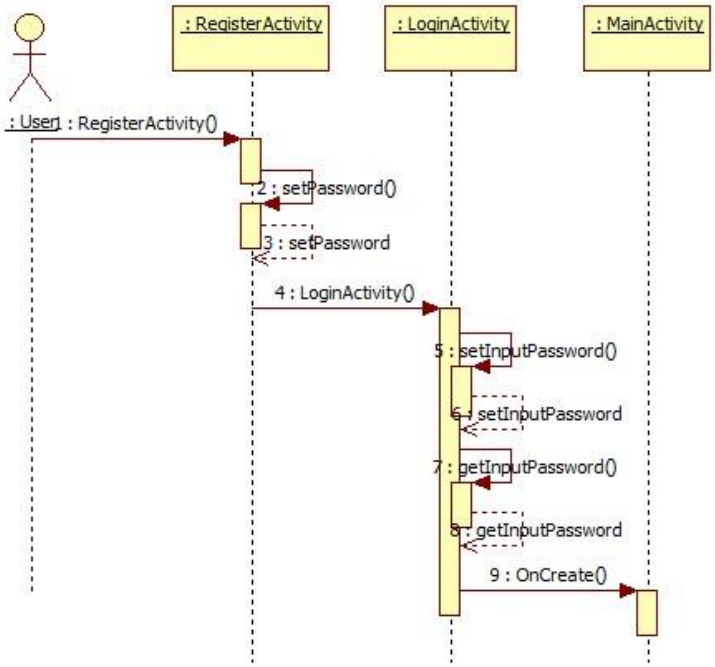
Gambar 3.16 ***Activity Diagram Change Password***

3.1.10.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) yang digambarkan terhadap waktu. *Sequence diagram* yang terdapat pada aplikasi keamanan sms yaitu sebagai berikut:

1. ***Sequence Diagram Make Password***

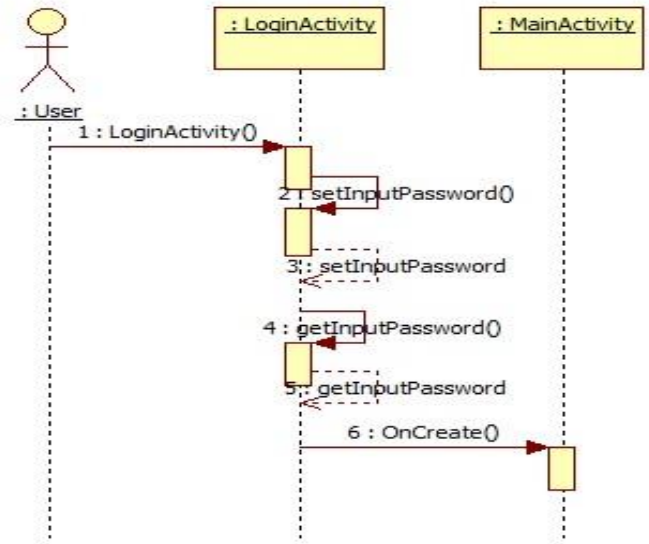
   Sequence diagram make password merupakan diagram yang menggambarkan interaksi yang terjadi dalam sistem antara pengirim/penerima dengan sistem dalam membuat *password*. *Sequence diagram make password* aplikasi kemanan SMS terlihat pada gambar 3.17.



Gambar 3.17 ***Sequence Diagram Make Password***

2. ***Sequence Login***

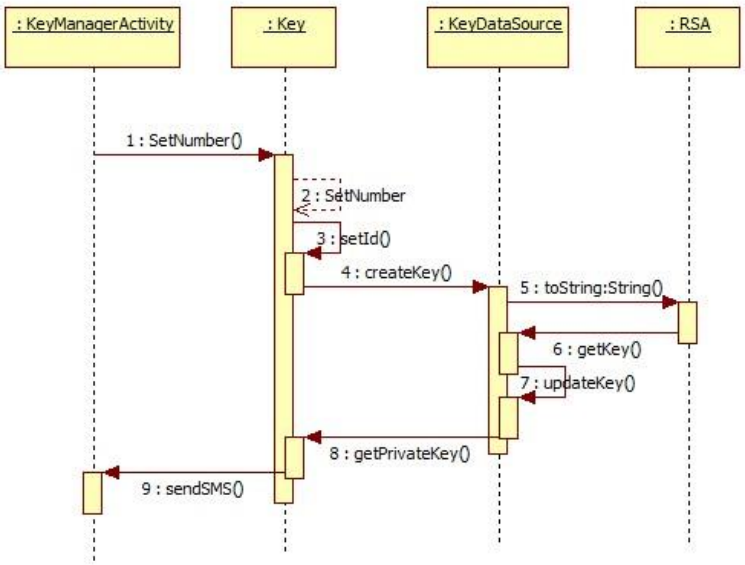
   Sequence diagram login merupakan diagram yang menggambarkan interaksi yang terjadi dalam sistem antara pengirim/penerima dengan sistem dalam memasukan *password*. *Sequence diagram login* aplikasi kemanan SMS terlihat pada gambar 3.18.



Gambar 3.18 ***Sequence Diagram Login***

3. *Sequrnce Diagram Generate Key*

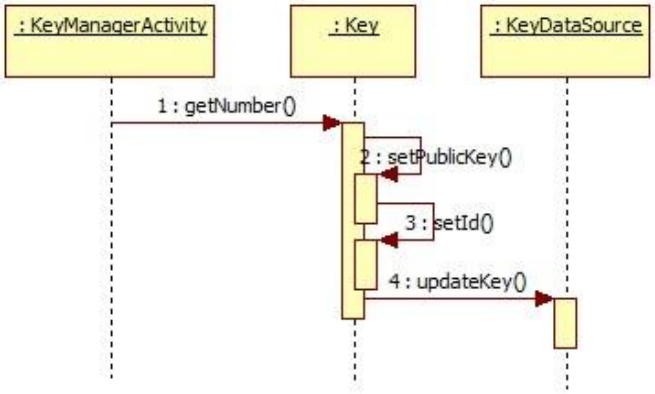
Sequence diagram generate key merupakan diagram yang menggambarkan interaksi yang terjadi dalam sistem antara pengirim dengan sistem untuk membangkitkan kunci. *Sequence diagram get key* aplikasi kemanan SMS terlihat pada gambar 3.19.



Gambar 3.19 ***Sequence Diagram Generate Key***

4. *Sequence Diagram Save Key*

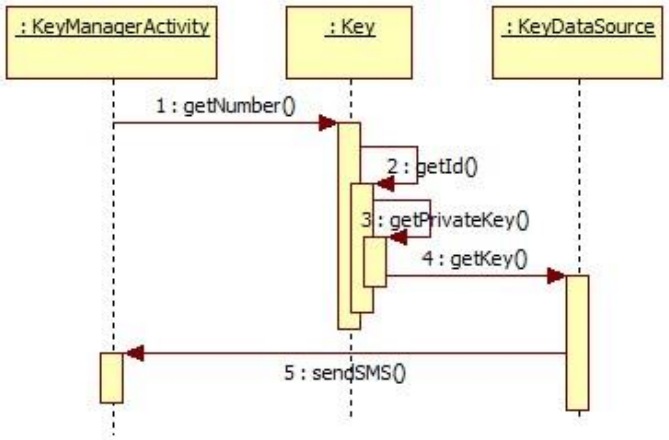
Sequence diagram save key merupakan diagram yang menggambarkan interaksi yang terjadi dalam sistem antara pengirim dengan sistem untuk menyimpan kunci yang telah dibangkitkan. *Sequence diagram get key* aplikasi kemanan SMS terlihat pada gambar 3.20.



Gambar 3.20 *Sequence Diagram Save Key*

5. *Sequence Diagram Send Key*

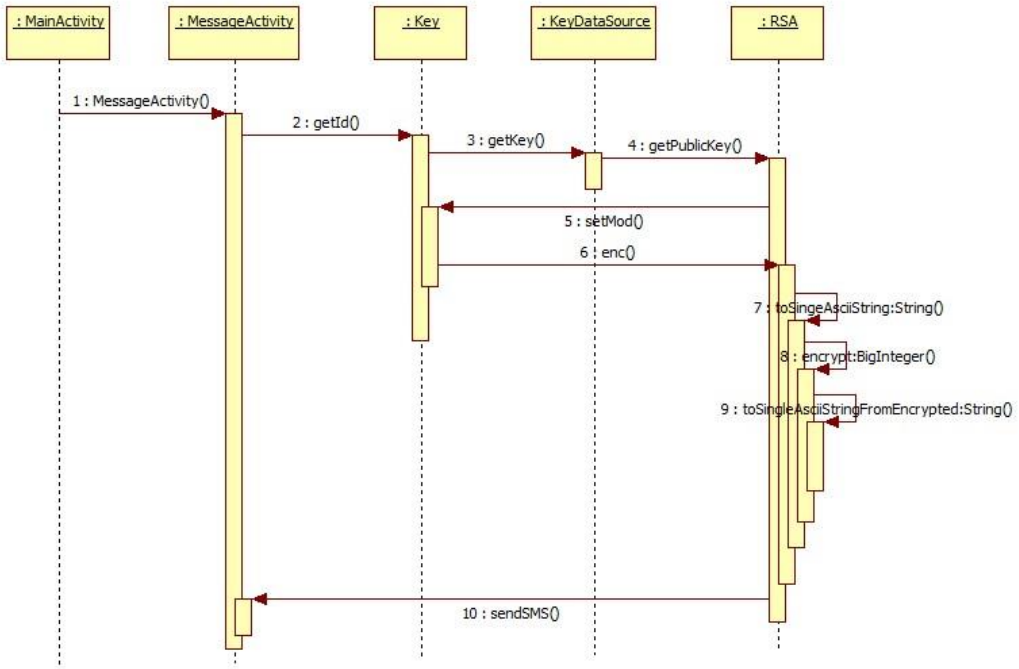
Sequence diagram send key merupakan diagram yang menggambarkan interaksi yang terjadi dalam sistem antara pengirim dengan sistem untuk mengirimkan kunci. *Sequence diagram send key* aplikasi kemanan SMS terlihat pada gambar 3.21.



Gambar 3.21 *Sequence Diagram Send Key*

6. *Sequence Diagram Encrypt*

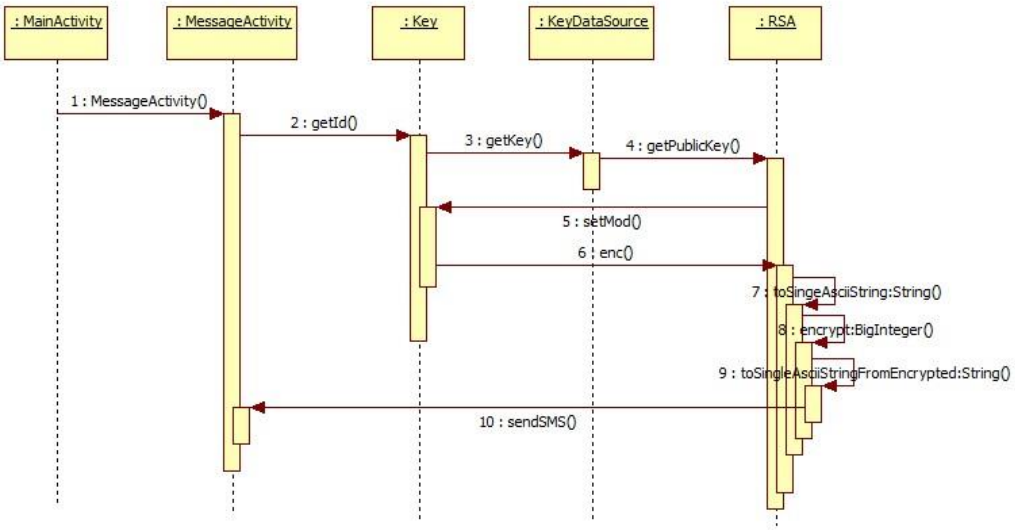
Sequence diagram encrypt merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam mengenkripsi pesan. *Sequence diagram encrypt* aplikasi keamanan SMS terlihat pada gambar 3.22.



Gambar 3.22 *Squence Diagram Encrypt*

7. *Sequence Diagram Send Message*

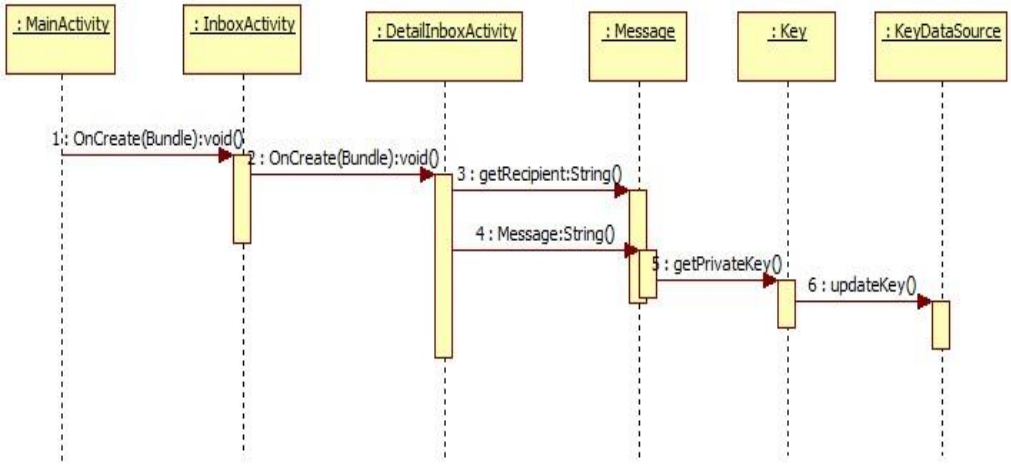
Sequence diagram send message merupakan diagram yang mengambarkan interaksi yang terjadi didalam sistem antara pengirim dengan sistem dalam mengirimkan pesan yang telah dienkripsi. *Sequence diagram send message* aplikasi keamanan SMS terlihat pada gambar 3.23.



Gambar 3.23 ***Sequence Diagram Send Message***

8. ***Sequence Diagram Save Private Key***

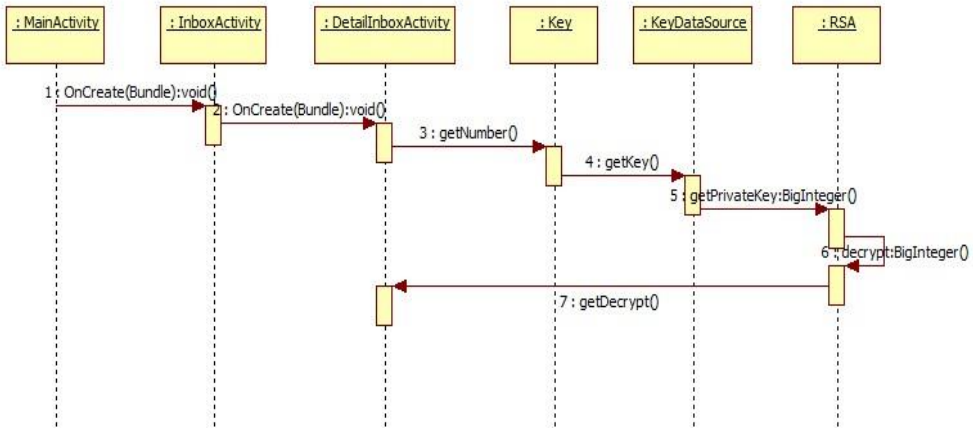
Sequence diagram save private key merupakan diagram yang mengambarkan interaksi yang terjadi didalam *system* antara *receive*r dengan sistem dalam menyimpan kunci privat yang dikirimkan ke dalam *database*. *Sequence diagram save private key* aplikasi keamanan SMS terlihat seperti pada gambar 3.24.



Gambar 3.24 ***Sequence Diagram Save Private Key***

9. *Sequence Diagram Decrypt*

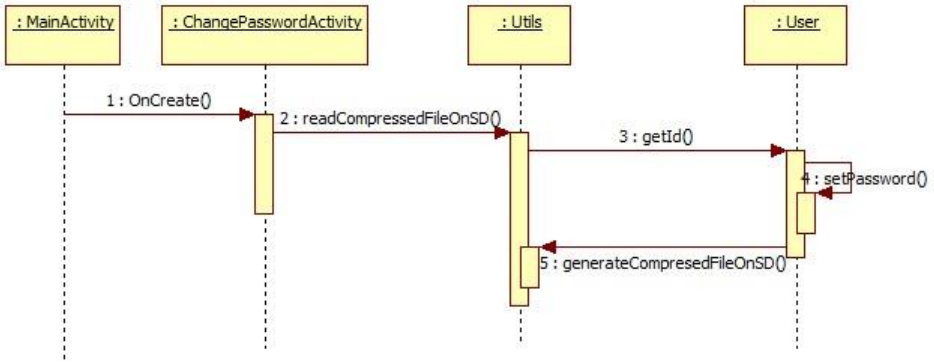
Sequence diagram decrypt merupakan diagram yang mengambarkan interaksi yang terjadi didalam *system* antara *receive*r dengan sistem dalam dendekripsi pesan. *Sequence diagram decrypt* aplikasi keamanan SMS terlihat seperti pada gambar 3.25.



Gambar 3.25 *Sequence Diagram Decrypt*

10. *Sequence Diagram Change Password*

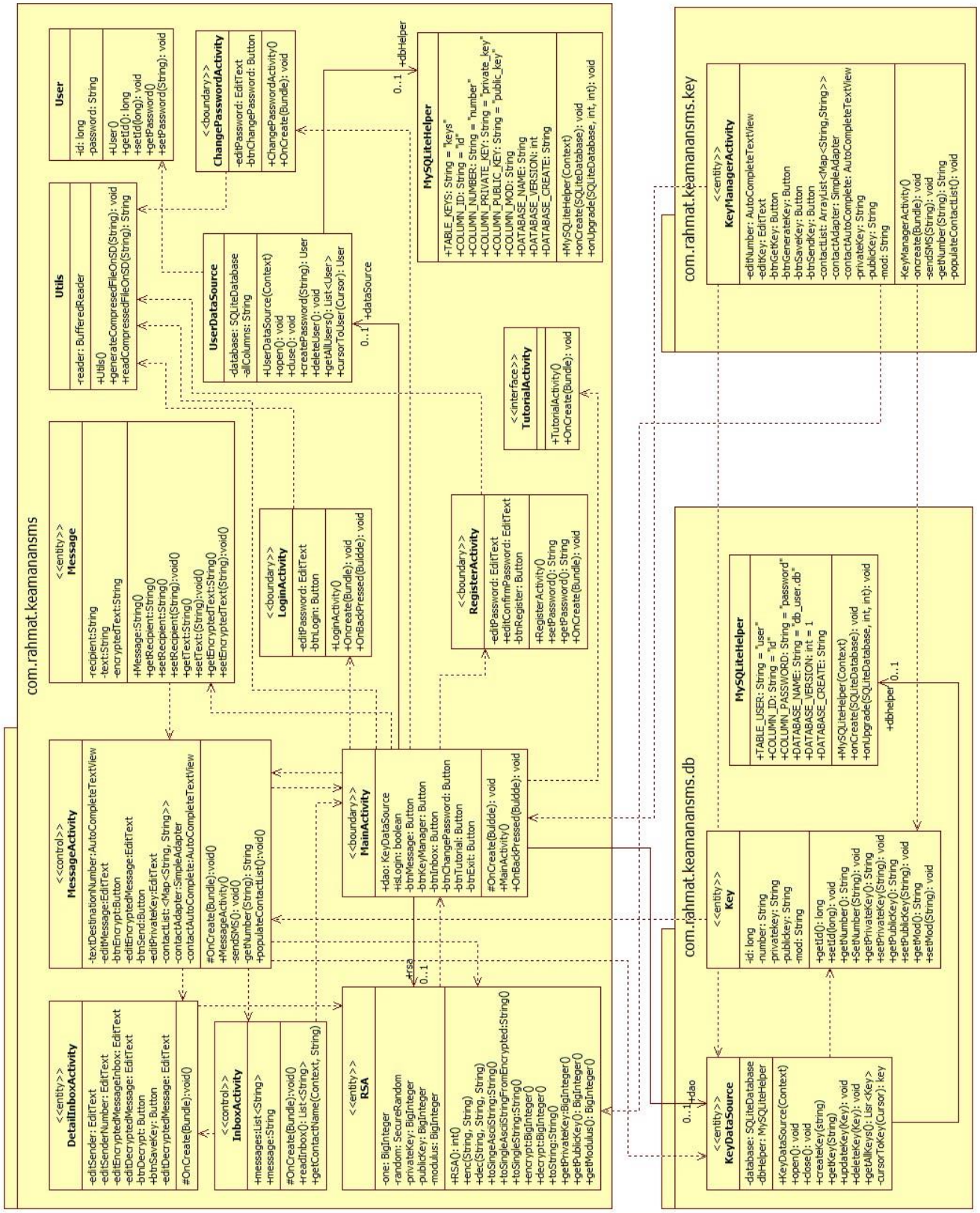
Sequence diagram *change password* merupakan diagram yang mengambarkan interaksi yang terjadi didalam *system* antara *user* dengan sistem dalam merubah password. *Sequence diagram change passwword* aplikasi keamanan SMS terlihat seperti pada gambar 3.15.



Gambar 3.26 *Sequence Diagram Change Password*

3.1.10.5 Class Diagram

Class diagram digunakan untuk menggambarkan keadan suatu sistem dengan menjelaskan keterhubungan antara suatu *class* dengan *class* yang lain yang terdapat pada sistem. *Class diagram* yang digunakan pada pembangunan aplikasi kriptografi ini yaitu terlihat pada gambar 3.27.



Gambar 3.27 ***Class Diagram***

Berikut jenis-jenis *clas diagram* terlihat pada tabel II.25.

Tabel 3.25 Jenis *Class Diagram*

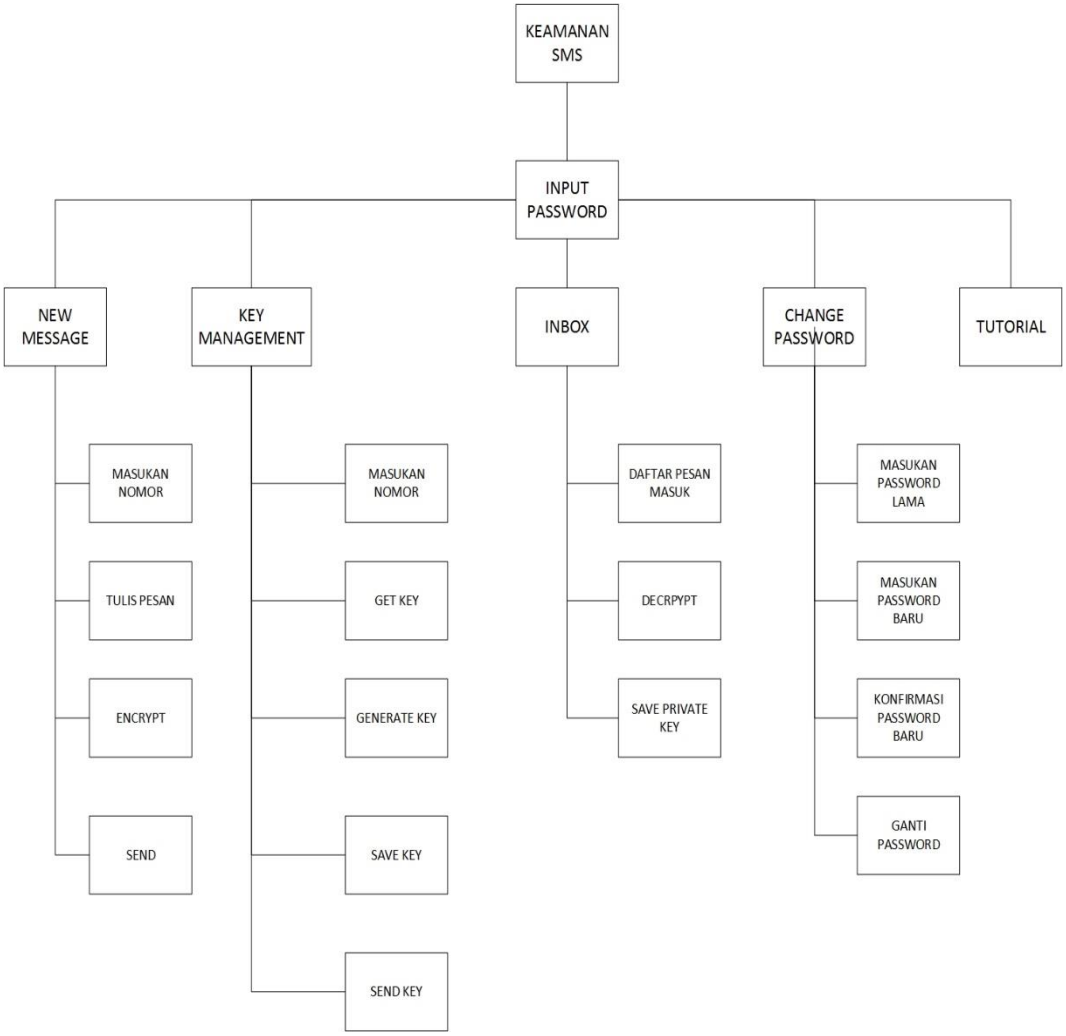
Nama Class	Jenis Class
RegisterActivity	Boundary
LoginActivity	Boundary
MainActivity	Boundary
MassageActivity	Control
Message	Entity
RSA	Entity
InboxActivity	Control
DetailInboxActivity	Entity
ChangePasswordActivity	Boundary
Key	Entity
KeyManagerActivity	Entity
KeyDataSource	Entity
ChangePasswordActivity	Boundary

3.2 Perancangan Sistem

Perancangan sistem merupakan suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis.

3.2.1 Perancangan Struktur Menu

Perancangan menu dilakukan untuk mempermudah interaksi antara sistem dengan pengguna. Gambar 3.28. merupakan struktur menu dari aplikasi keamanan sms yang dibangun.



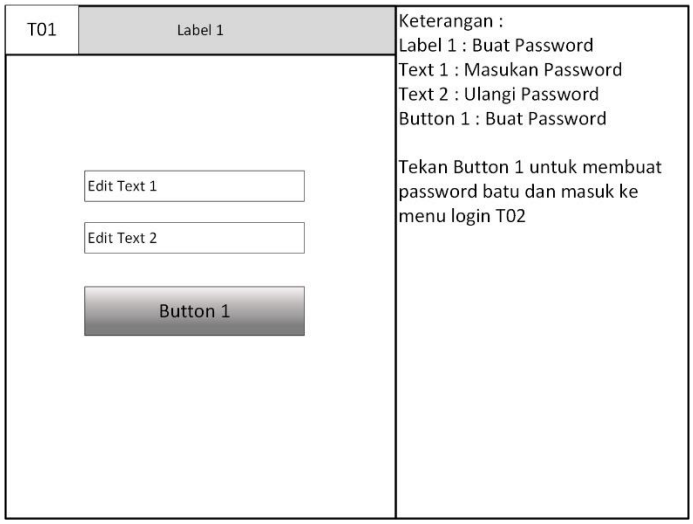
Gambar 3.28 Struktur Menu Aplikasi Keamanan SMS

3.2.2 Perancangan Antarmuka

Perancangan antarmuka adalah tahapan pembuatan rancangan antarmuka untuk digunakan pada pembangunan aplikasi kemanan sms yang dibagi menjadi sembilan bagian yaitu perancangan antarmuka *make password*, perancangan antarmuka *login*, parancangan antarmuka menu utama, perancangan antarmuka *new message*, perancangan antarmuka *inbox*, perancangan antarmuka detail pesan, perancangan antarmuka dekripsi pesan dan perancangan antarmuka tutorial.

1. Perancangan Antarmuka Buat *Password* (T01)

Perancangan antarmuka buat *password* merupakan tahapan rancangan pertama ketika pengguna telah menginstal aplikasi sebelum memasuki menu utama, jika sebelumnya telah membuat *password* maka akan langsung ke antarmuka *Login*. Perancangan antarmuka buat *password* dari aplikasi keamanan SMS ini terlihat pada gambar 3.29.



Gambar 3.29 Perancangan Antarmuka Buat *Password*

2. Perancangan Antarmuka *Login* (T02)

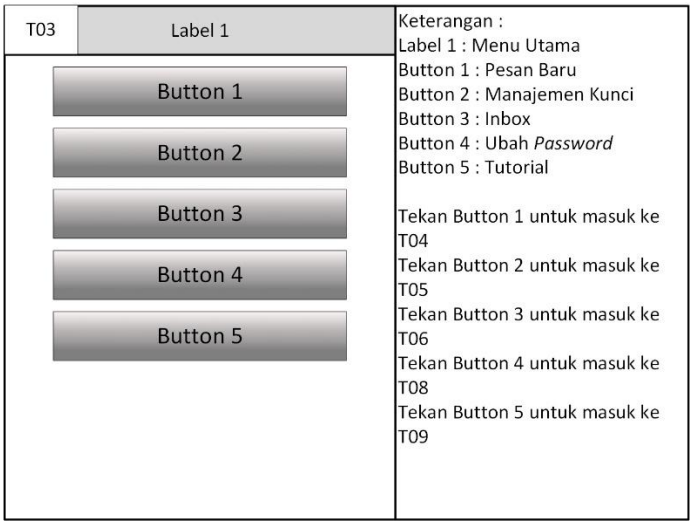
Perancangan antarmuka *login* merupakan tahapan rancangan antarmuka sebelum memasuki menu utama. Perancangan antarmuka *login* dari aplikasi Keamanan SMS ini terlihat pada gambar 3.30.



Gambar 3.30 Perancangan Antarmuka *Login*

3. Perancangan Antarmuka Menu Utama (T03)

Perancangan antarmuka menu utama merupakan tahapan rancangan antarmuka utama setelah sukses menginputkan *password*. Perancangan antarmuka menu utama dari aplikasi Keamanan SMS ini terlihat pada gambar 3.31.



Gambar 3.31 Perancangan Antarmuka Menu Utama

4. Perancangan Antarmuka Pesan Baru (T04)

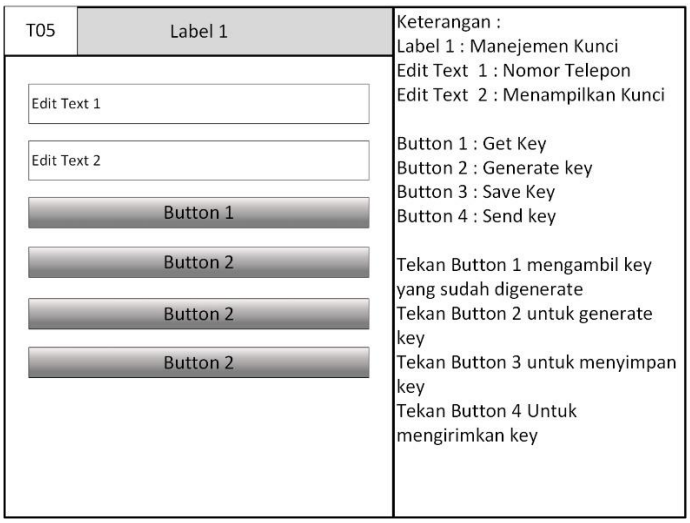
Perancangan antarmuka pesan baru merupakan tahapan rancangan antarmuka setelah menakan *button* kirim pesan pada menu utama. Perancangan antarmuka pesan baru dari aplikasi Keamanan SMS ini terlihat pada gambar 3.32.



Gambar 3.32 Perancangan Antarmuka Pesan Baru

5. Perancangan Menejemen Kunci (T05)

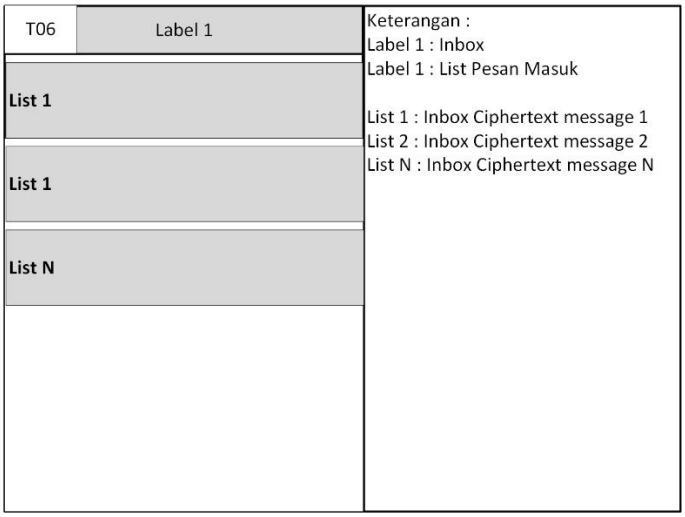
Perancangan antarmuka manejemen kunci merupakan tahapan perancangan antarmuka setelah menekan *button management key* pada menu utama. Perancangan antarmuka manejemen kunci dari aplikasi Keamanan SMS ini terlihat pada gambar 3.33.



Gambar 3.33 Perancangan Antarmuka Manajemen Kunci

6. Perancangan Antarmuka Kotak Masuk (T06)

Perancangan antarmuka *inbox* merupakan tahapan rancangan antarmuka setelah menakan button *inbox* pada menu utama. Perancangan antarmuka *inbox* dari aplikasi Keamanan SMS ini terlihat pada gambar 3.34.



Gambar 3.34 Perancangan Antarmuka Kotak Masuk

7. Perancangan Antarmuka Detail Kotak Masuk (T07)

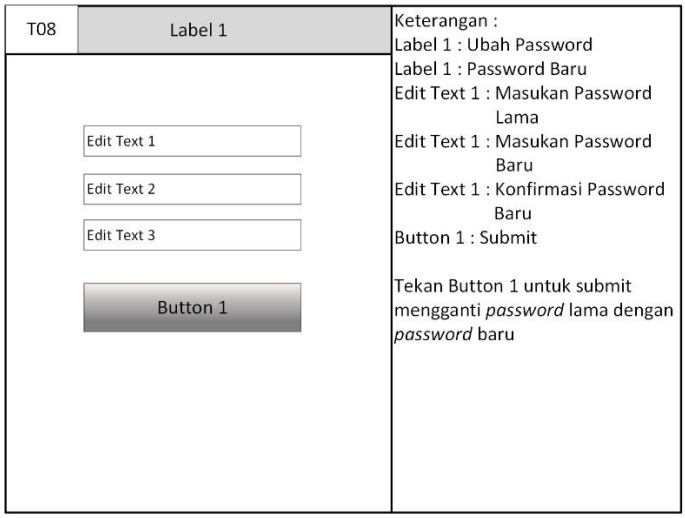
Perancangan antarmuka detail kotak masuk merupakan tahapan rancangan antarmuka setelah memilih pesan di dalam kotak masuk. Perancangan antarmuka detail inbox dari aplikasi Keamanan SMS ini terlihat pada gambar 3.35.



Gambar 3.35 Perancangan Antarmuka Detail Kotak Masuk

8. Perancangan Antarmuka Ubah *Password* (T08)

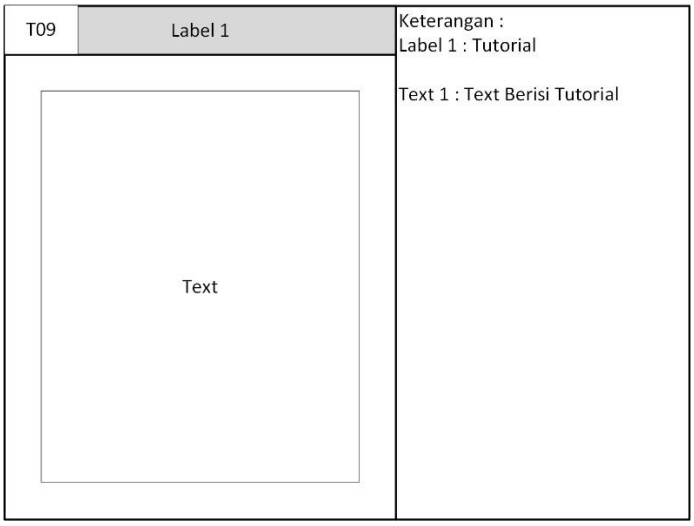
Perancangan antarmuka ubah *password* merupakan tahapan rancangan antarmuka setelah menekan *button* ubah *password* pada menu utama. Perancangan antarmuka ubah *password* dari aplikasi Keamanan SMS ini terlihat pada gambar 3.36.



Gambar 3.36 Perancangan Antarmuka Ubah *Password*

9. Perancangan Antarmuka Tutorial (T09)

Perancangan antarmuka tutorial merupakan tahapan rancangan antarmuka setelah menekan tombol tutorial pada menu utama. Perancangan antarmuka tutorial dari aplikasi Keamanan SMS terlihat pada gambar 3.37.



Gambar 3.37 Perancangan Antarmuka Tutorial

3.2.3 Perancangan Pesan

Pesan merupakan tampilan dari suatu perangkat lunak yang berfungsi untuk menyampaikan notifikasi dan informasi kepada pengguna agar perangkat lunak lebih interaktif. Perancangan pesan pada aplikasi keamanan SMS dibagi menjadi empat bagian yaitu perancangan pesan *New Password*, perancangan pesan *Input Password*, perancangan pesan *New Mesasage*, perancangan pesan *Detail* Pesan, dan perancangan pesan *Change Password*.

3.2.3.1 Perancangan Pesan Buat Password

Perancangan pesan pada *form* Buat *Password* ini terdapat tiga perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika *Password* Kurang dari 6 Karakter (M01)**
   Perancangan pesan ini muncul ketika pengirim atau penerima menekan tombol Buat *Password* ketika *password* yang dimasukan kurang dari 6 karakter. Perancangan pesan ini terlihat pada gambar 3.38.



Gambar 3.38 Perancangan Pesan Jika Password Kurang dari 6 Karakter

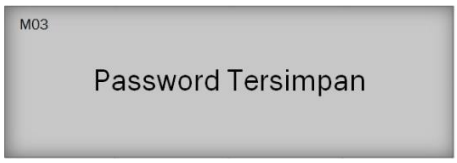
2. **Perancangan Pesan Jika Konfirmasi *Password* Tidak Sesuai (M02)**
   Perancangan pesan ini muncul ketika pengirim atau penerima menekan tombol Buat *Password* dan *password* kedua yang dimasukan tidak sama dengan *password* pertama yang dimasukan. Perancangan pesan ini terlihat pada gambar 3.39.



Gambar 3.39 Perancangan Pesan Jika Konfirmasi *Password* Tidak Sesuai

3. **Perancangan Pesan Jika *Password* Berhasil Disimpan (M03)**

   Perancangan pesan ini muncul ketika pengirim atau penerima berhasil membuat *password* dengan menekan tombol Buat *Password*. Perancangan pesan ini terlihat pada gambar 3.40.



Gambar 3.40 Perancangan Pesan *Password* Berhasil Disimpan

3.2.3.2 Perancangan Pesan Login

Perancangan pesan pada menu *Input Password* ini terdapat dua perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Berhasil *Login* (M04)**

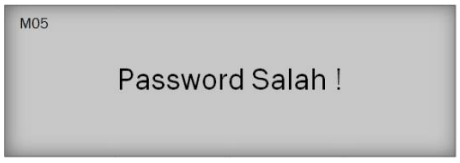
   Perancangan pesan ini muncul ketika menekan tombol Masuk dan *password* yang dimasukan benar. Perancangan pesan ini terlihat pada gambar 3.41.



Gambar 3.41 Perancangan Pesan Jika Berhasil *Login*

2. **Perancangan Pesan Jika Gagal *Login* (M05)**

   Perancangan pesan ini muncul ketika menekan tombol Masuk dan *password* yang dimasukan salah. Perancangan pesan ini terlihat pada gambar 3.42.



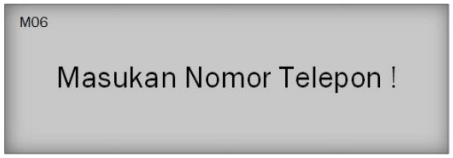
Gambar 3.42 Perancangan Pesan Jika Gagal *Login*

3.2.3.3 Perancangan Pesan Pesan Baru

Perancangan pesan pada menu Pesan Baru ini terdapat lima perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Nomor Tujuan Belum Dimasukan (M06)**

   Perancangan pesan ini muncul ketika pengirim menekan tombol *encrypt* sebelum memasukan nomor tujuan. Perancangan pesan ini terlihat pada gambar 3.43.



Gambar 3.43 Perancangan Pesan Jika Nomor Tujuan Belum Dimasukan

2. **Perancangan Pesan Jika Pesan Belum di Isi (M07)**

   Perancangan pesan ini muncul ketika pengirim menekan tombol *encrypt* sebelum mengisikan pesan. Perancangan pesan ini terlihat pada gambar 3.44.



Gambar 3.44 Perancangan Pesan Jika Pesan Belum di Isi

3. **Perancangan Pesan Jika Pesan Berhasil Dienkripsi (M08)**

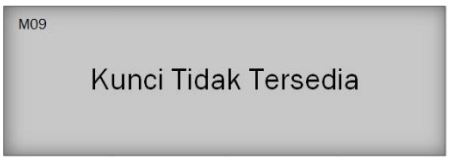
   Perancangan pesan ini muncul ketika pengirim berhasil mengenkripsim pesan dengan menekan tombol *encrypt*. Perancangan pesan ini terlihat pada gambar 3.45.



Gambar 3.45 Perancangan Pesan Jika Pesan Berhasil Dienkripsi

4. **Perancangan Pesan Jika Kunci Tidak Tersedia (M09)**

   Perancangan pesan ini muncul jika kunci tidak tersedia ketika pengirim menekan tombol *encrypt.* Perancangan pesan ini terlihat pada gambar 3.46.



Gambar 3.46 Perancangan Pesan Jika Kunci Tidak Tersedia

5. **Perancangan Pesan Jika Pesan Terenkripsi Berhasil Dikirim (M10)**

   Perancangan pesan ini muncul ketika pengirim berhasil mengirimkan pesan dengan menekan tombol kirim Perancangan pesan ini terlihat pada gambar 3.47.



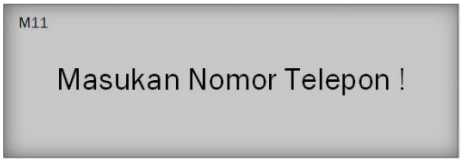
Gambar 3.47 Perancangan Pesan Jika Pesan Terenkripsi Berhasil Dikirim

3.2.3.4 Perancangan Pesan Manejemen Kunci

Perancangan pesan pada menu Menejemen Kunci ini terdapat lima perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Belum Masukan Nomor Telepon (M11)**

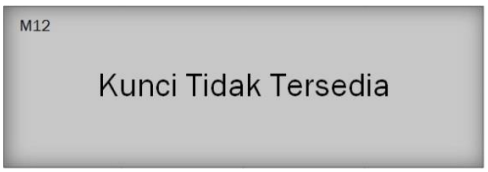
   Perancangan ini muncul jika pengirim belum memasukan nomor telepon ketika menekan tombol Ambil Kunci, Bangkitkan Kunci, Simpan Kunci, atau Kirim Kunci. Perancangan pesan ini terlihat pada gambar 3.48.



Gambar 3.48 Perancangan Pesan Jika Belum Memasukan Nomor Telepon

2. Perancangan Pesan Jika Kunci Tidak Tersedia (M12)

Perancangan ini muncul jika nomor tersebut belum ada di dalam *database* dan belum memiliki kunci ketika pengirim menekan tombol Ambil Kunci. Perancangan pesan ini terlihat pada gambar 3.49.



Gambar 3.49 Perancangan Pesan Jika Kunci Tidak Tersedia

3. Perancangan Pesan Jika Berhasil Membangkitkan Kunci (M13)

Perancangan ini muncul jika sistem berhasil membangkitkan kunci ketika pengirim menekan tombol Bangkitkan Kunci. Perancangan pesan ini terlihat pada gambar 3.50.



Gambar 3.50 Perancangan Pesan Jika Berhasil Membangkitkan Kunci

4. Perancangan Pesan Jika Berhasil Menyimpan Kunci (M14)

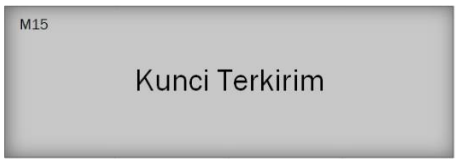
Perancangan pesan ini muncul jika sistem berhasil menyimpan kunci yang telah dibangkitkan ke dalam *database* ketika pengirim menkan tombol Simpan Kunci. Perancangan pesan ini terlihat pada gambar 3.51.



Gambar 3.51 Perancangan Pesan Jika Berhasil Menyimpan Kunci

5. **Perancangan Pesan Jika Berhasil Mengirimkan Kunci (M15)**

   Perancangan pesan ini muncul ketika pengirim menekan tombol send dan berhasil mengirimkan kunci melalui sms. Perancangan pesan ini terlihat pada gambar 3.52.



Gambar 3.52 Perancangan Pesan Jika Kunci Berhasil Dikirimkan

3.2.3.5 Perancangan Detail Pesan

Perancangan pesan pada Detail Pesan ini terdapat dua perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Berhasil Mendekripsi Pesan (M16)**

   Perancangan pesan ini muncul ketika penerima berhasil mendekripsi pesan dengan menekan tombol *decrypt*. Perancangan pesan ini terlihat pada gambar 3.53.



Gambar 3.53 Perancangan Pesan Jika Berhasil Mendekripsi Pesan

2. **Perancangan Pesan Kunci Privat Berhasil Disimpan (M17)**

   Perancangan pesan ini muncul jika penerima berhasil menyimpan *private key* yang dikirmkan pengirim ketika menekan Tombol Simpan Kunci. Perancangan pesan ini terlihat pada gambar 3.54.



Gambar 3.54 Perancangan Pesan Jika Berhasil Menyimpan Private Key

3.2.3.6 Perancangan Ubah Password

Perancangan pesan pada ubah *password* terdapat empat perancangan pesan yaitu sebagai berikut :

1. **Perancangan Pesan Jika Salah Memasukan *Password* Lama (M18)**

   Perancangan pesan ini muncul jika *password* lama yang dimasukan salah ketika menekan tombol Ubah *Password*. Perancangan pesan ini terlihat pada gambar 3.55.



Gambar 3.55 Perancangan Pesan Jika Salah Memasukan *Password* Lama

2. **Perancangan Pesan Jika *Password* Baru Kurang dari 6 karakter (M19)**

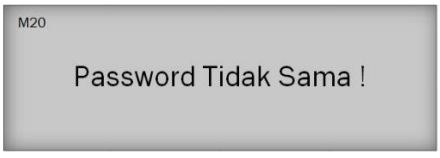
   Perancangan pesan ini muncul ketika pengirim atau penerima menekan tombol Ubah *Password* ketika *password* yang dimasukan kurang dari 6 karakter. Perancangan pesan ini terlihat pada gambar 3.56.



Gambar 3.56 Perancangan Pesan Jika *Password* Kurang dari 6 Karakter

3. **Perancangan Pesan Jika Konfirmasi *Password* Tidak Sesuai (M20)**

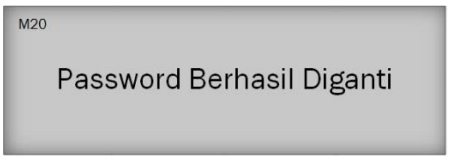
   Perancangan pesan ini muncul ketika pengirim atau penerima menekan tombol Ubah *Password* dan *password* kedua yang dimasukan tidak sama dengan *password* pertama yang dimasukan. Perancangan pesan ini terlihat pada gambar 3.57.



Gambar 3.57 Perancangan Pesan Jika Konfirmasi *Password* Tidak Sama

4. Perancangan Pesan Jika Berhasil Mengganti *Password* (M21)

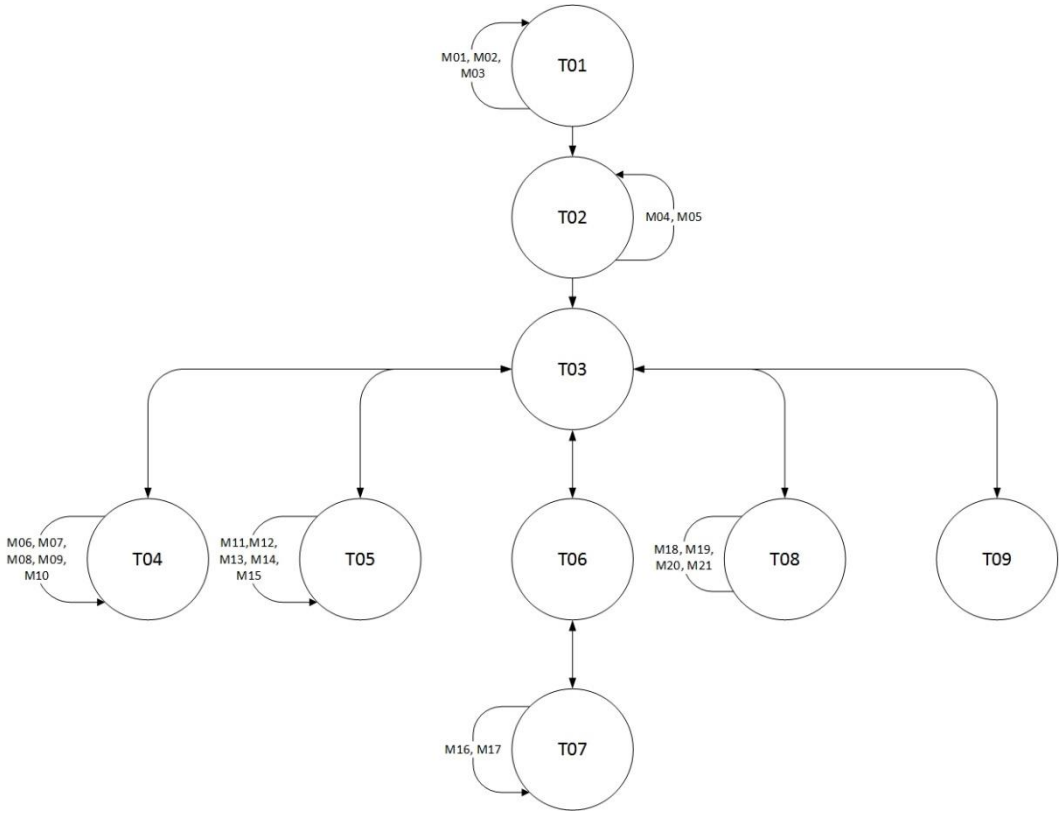
Perancangan pesan ini muncul ketika pengguna berhasi mengganti *password* dengan menekan tombol Ubah *Password*. Perancangan pesan ini terlihat pada gambar 3.58.



Gambar 3.58 Perancangan Pesan Jika Berhasil Mengganti *Password*

3.2.3.7 Perancangan Jaringan Semantik

Jaringan semantik adalah diagram yang menggambarkan aliran-aliran menu dan pesan dalam sebuah program. Jaringan semantik dari aplikasi keamanan SMS yang dibangun yaitu seperti pada gambar 3.59.



Gambar 3.59 Jaringan Semantik Aplikasi Keamanan SMS