BAB II
LANDASAN TEORI

2.1. KRIPTOGRAFI

Mekanisme keamanan jaringan pada implementasi menggunankan teknik-teknik penyandian yaitu kriptografi.

2.1.1. Pengertian Kriptografi

Kriptografi adalah bidang ilmu pengetahuan yang mempelajari pemakaian persamaan matematika untuk melakukan proses penyandian data (Onno, 200). Kriptografi mempunyai tujuan yaitu mengamankan isi data atau menjaga kerahasian informasi dari orang yang tidak berhak untuk mengetahui isi data tersebut. Agar isi data aman maka diperlukan teknik atau algoritma untuk mengamankannya seperti proses enkripsi dan Dekripsi , untuk dapat melakukan proses tersebut maka pengirim dan penerima harus mengetahui algoritma yang digunakan dan memiliki kunci yang sesuai.

Tingkat keamanan dari data sandi terhadap upaya proses deskirpsi secara paksa oleh orang yang tidak berhak ditentukan oleh kekuatan algoritma yang digunakan dan kerahasian kunci. Kekuatan algoritma yang digunakan untuk proses enkripsi dan deskipsi berhubungan erat dengan penggunaan persamaan matematika . semakin banyak dan rumit perhitungan dari persamaan matematika yang digunakan maka data sandi semakin aman (Alfred, 1997). Kerahasian kunci adalah bagaimana cara kunci tersebut disimpan dan didistribusikan kepada pihak yang berhak menerima data, karena kunci ini akan digunakan untuk melakukan dekripsi , semakin rapi kunci disimpan dan didistribusikan maka kunci sandi semakin aman, Berikut ini istilah yang berhubungan erat dengan kriptografi (Onno, 200).

1. *Plaintext* adalah data asli atau informasi bersifat terbuka yangisinya dapat dibaca dan dipahami secara langsung.
2. *Chiphertext* adalah data sandi hasil dari proses dekripsi .

3. *Chipher* adalah algoritma mengubah plaintext menjadi ciphertext menggunakan persamaan matematika.
4. *Substitution Chipher* adalah algoritma mengubah *plaintext* menjadi *ciphertext* dengan cara mengganti menggunakan persamaan matematika tertentu.
5. *Transporsition Cipher* adalah algoritma mengubah *plaintext* menjadi *ciphertext* dengan cara menggeser menggunakan persamaan matematika tertentu.
6. *Block Cipher* adalah algoritma mengubah *plaintext* menjadi *ciphertext* untuk setiap blok data , jumlah data atau besarnya block adalah tertentu.
7. Kunci (*key*) adalah data atau nilai yang sangat spesifik yang diketahui oleh pengirim dan penerima yang berhak.
8. Enkripsi adalah proses yang digunakan untuk menyembunyikan *plaintext.*
9. Dekripsi adalah proses mengembalikan *ciphertext* menjadi *plaintext.*
10. Kriptosistem adalah system kriptografi yang didalamnya terdiri dari algoritma kriptografi, plaintext, *cipheryrxt, key*, dan unsur lain yang berpengaruh dalam system kriptografi.
11. *Code Breaking* adalah kegiatan untuk mengubah *ciphertext* menjadi pesan asli tanpa mengetahui kunci yang sesuai dengan cara mencoba-coba secara sistimatis.
12. Cryptology adalah ilmu matematika yang mendasari *crypthography* dan *code breaking*.

Keamanan informasi setelah dilakukan proses pengiriman dan penerimaan informasi maka dapat dilakukan tindakan – tindakan berikut ini :

1. Membuktikan keaslian adalah proses yang memungkinkan penerima informasi untuk mengetahui asal atau pengirim informasi yang sebenarnya.
2. Menjaga integritas data yaitu proses yang menjamin penerima informasi dapat memeriksa, apakah infomasi telah berubah sebelum diterima.
3. Pembuktian seseorang telah mengirim pesan adalah proses untuk menjamin pengirim informasi tidak dapat menyangkal bahwa dia telah mengirim informasi tersebut.

4. Menjaga kerahasian yaitu proses untuk menjamin informasi yang dikirim tidak dapat dipahami isinya oleh orang yang tidak berhak.

Pada Gambar 2.1. dapat dilihat alur proses enkripsi dan dekripsi yang menyertakan kunci

Gambar 2.1. Alur proses enkripsi dan dekripsi yang menyertakan kunci.

Secara persamaan matematika dapat dituliskan:

1. Proses enkripsi $ek_1(M) = C$
2. Proses dekripsi $dk_2(C) = M$
3. Proses pengujian dapat dilakukan dengan cara $dk_2(ek_1(M)) = M$

Dalam perkembangannya algoritma kriptografi yang menggunakan kunci dibedakan menjadi dua, berdasarkan nilai kunci yang digunakan yaitu:

1. Algoritma Simetri

   Kunci k_1 untuk proses enkripsi sama dengan kunci k_2 untuk proses dekripsi, sehingga $k_1 = k_2 = k$. Kunci k hanya boleh diketahui oleh pengirim dan penerima saja.

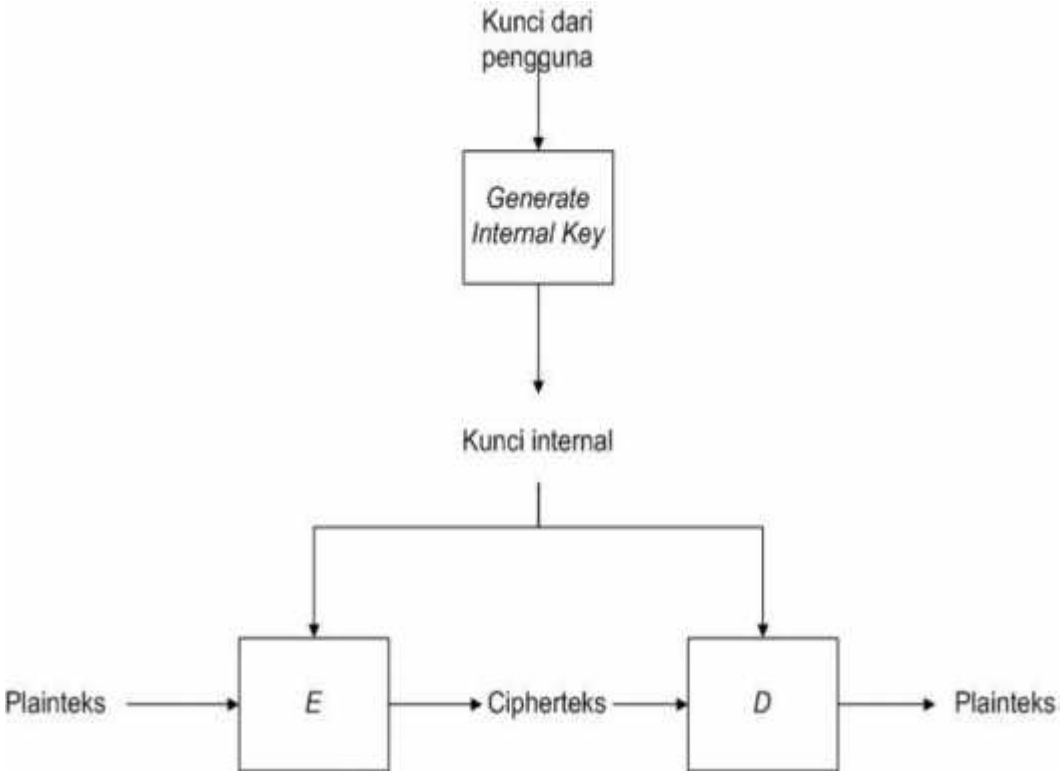
2. Algoritma Kunci Public

   Kunci k_1 untuk proses enkripsi tidak sama dengan kunci k_2 untuk proses dekripsi, sehingga $k_1 \neq k_2$. Kunci k_1 boleh diketahui oleh umum tetapi kunci k_2 hanya diketahui oleh pihak penerima saja yang akan melakukan proses dekripsi. Kunci k_1 dan k_2 memiliki sifat berpasangan.

2.2.Block Cipher

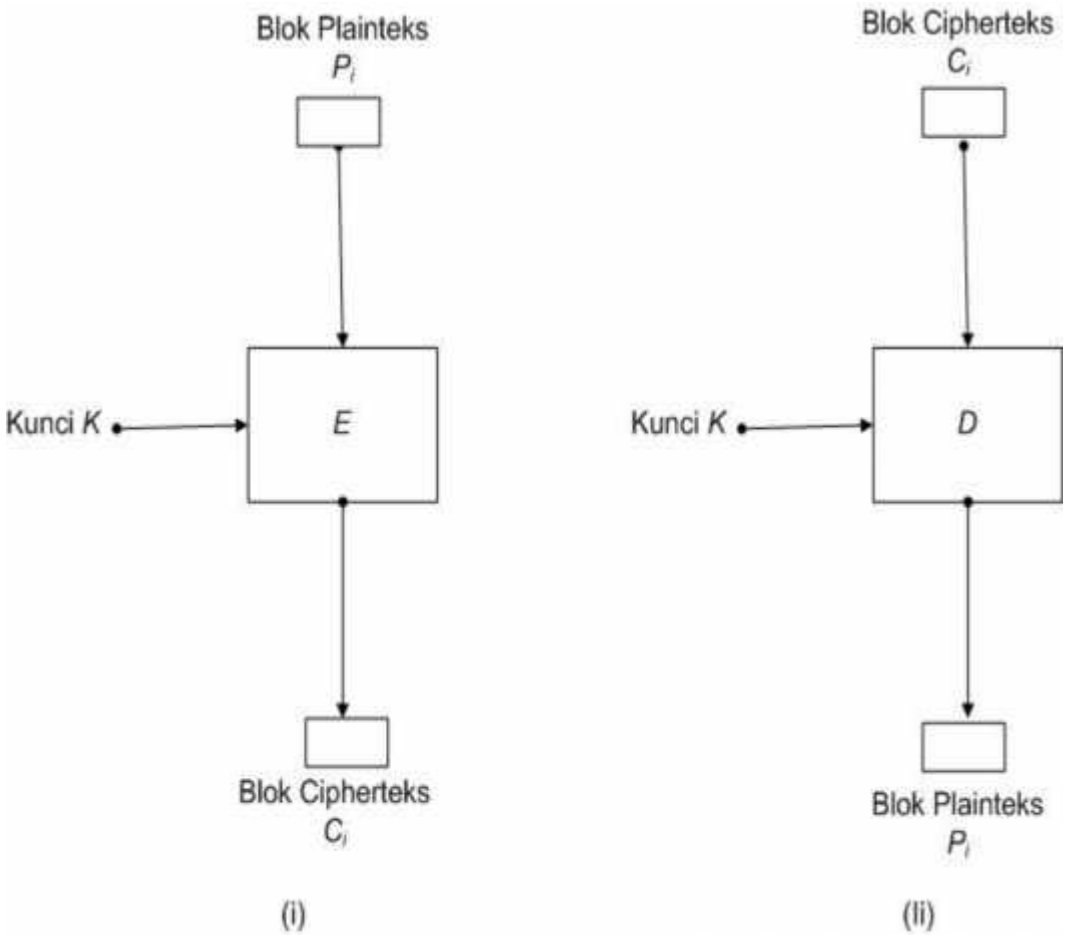
Block cipher adalah suatu tipe algoritma kriptografi kunci simetris yang mengubah *plainteks* yang dibagi dalam blok-blok dengan panjang yang sama menjadi *cipherteks* yang memiliki panjang blok yang sama. Ukuran panjang blok dapat beragam bergantung kepada algoritma yang digunakan, ukuran yang sering digunakan adalah 64 bit dan menuju 128 bit. Seperti semua algoritma kunci

simetri, proses enkripsi yang dilakukan akan menggunakan suatu input dari user yang disebut sebagai kunci rahasia. Kunci rahasia ini juga akan dipakai ketika melakukan proses dekripsi. Cara kerja secara umum dari *block cipher* dapat dilihat pada Gambar 2.2.



Gambar 2.2. Skema cara kerja block cipher

Dalam penggunaannya *block cipher* dikombinasikan dengan suatu teknik yang dinamakan mode operasi dari *block cipher*. Mode operasi yang sederhana dan sering digunakan adalah mode *Electronic Code Book* (ECB). Pada mode ECB setiap blok pada *plainteks* dienkripsi satu persatu secara independen. Hasil enkripsi masing - masing blok tidak mempengaruhi blok yang lain. Proses enkripsi pada mode ini sangat sederhana, setiap blok *plainteks* dienkripsi dengan fungsi enkrispsi secara terpisah. Seperti halnya dalam proses enkripsi, dalam proses dekripsi, masing-masing blok-blok *cipherteks* dikenakan dengan fungsi dekripsi secara independen. Proses enkripsi dan dekripsi dari mode ECB dapat dilihat pada Gambar 2.3.



Gambar 2.3.

(i)Proses enkripsi ECB dan (ii)Proses dekripsi ECB pada sebuah blok

Dalam melakukan perancangan *block cipher*, beberapa prinsip harus dipertimbangkan. Prinsip-prinsip tersebut yaitu:

1. Prinsip *Confusion* dan *Diffusion.*

   Tujuan dari prinsip *confusion* adalah untuk menyembunyikan hubungan apapun yang ada antara *plainteks, cipherteks*, dan kunci, sehingga dapat membuat kripanalisis kesulitan dalam menemukan pola-pola pada *cipherteks*. Tujuan dari prinsip *diffusion* adalah menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin *cipherteks,* sehingga dengan berubahnya satu bit *plainteks* dapat mengubah *cipherteks* yang sulit untuk diprediksi (Munir, 2006)

2. *Iterated Cipher*

Untuk menambah keamanan, pada algoritma-algoritma *block cipher* dilakukan iterasi pada pemrosesan setiap blok, pada setiap rotasi dari iterasi tersebut digunakan fungsi transformasi yang sama namun memakai kunci yang berbeda yang disebut dengan kunci internal. Kunci internal pada umumnya merupakan hasil dari kunci yang dimasukan oleh pengguna yang dikomputasi menggunakan suatu fungsi tertentu. Dengan adanya iterasi tersebut keamanan akan semakin terjamin, namun performansi akan berkurang karena adanya waktu lebih yang dibutuhkan untuk melakukan iterasi. *Block cipher* yang menerapkan konsep iterasi ini disebut juga dengan *iterated block cipher*.

3. Kunci Lemah

Suatu hal yang perlu dihindari dalam melakukan perancangan algoritma kriptografi adalah kunci yang dapat menghasilkan *cipherteks* yang mirip atau serupa dengan *plainteks.*

2.3.Citra Digital

Citra digital (*image*) adalah istilah lain untuk gambar yang merupakan salah satu komponen multimedia dan memegang peranan penting sebagai informasi *visual*. *Citra* mempunyai karekteristik yang tidak memiliki oleh data bentuk teks yakni kaya dengan informasi. Ada sebuah pribahasa mengatakan, *a picture is more then a thousand word*, artinya sebuah gambar dapat memberikan informasi lebih banyak daripada informasi tersebut disajikan dalam bentuk ribuan kata. (Munir, 2004).

Citra ada dua macam yakni *citra kontiniu* dan *citra diskrit*. *Citra kontiniu* dihasilkan dari *system optic* yang menerima *sinyal analog* sedangkan *citra diskrit* dihasilkan memalui proses *digitalisasi* terhadap *citra kontiniu*.

Citra diskrit disebut juga *citra digital* yang dinyatakan sebagai suatu fungsi dua dimensi f (X, Y) dengan X maupun Y merupakan posisi koordinat. Sedangkan F merupakan intensitas cahaya (*brightness*) pada (X, Y). Suatu *citra digital* dapat dinyatakan dengan persaman 2.1.

$$f(x,y)\begin{pmatrix} f(1,1) & f(1,2) & f(1,3) & \dots & f(1,n) \\ f(2,1) & f(2,2) & f(2,3) & \dots & f(2,n) \\ f(3,1) & f(3,2) & f(3,3) & \dots & f(3,n) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ f(m-1,1) & f(m-1,2) & f(m-1,3) & \dots & f(m-1,n) \\ f(m,1) & f(m,2) & f(m,3) & \dots & f(m,n) \end{pmatrix} \quad (2.1)$$

Dari persamaan 2.1 *citra digital* dapat dinyatakan sebagai matrik dengan banyak baris matrik / tinggi *citra* = m dan bnyak kolom matrik / lebar citra = n. indeks x dan *indeks* y menyatakan suatu koordinat titik pada *citra*, sedangkan f (x , y) merupakan *intensitas* pada titik (x , y). Masing –masing elemen matrik disebut *image element* / *picture elementl pixel*. Jadi *citra* yang berukuran m x n mempunyai mn buah *pixel*.

Suatu citra disimpan dalam bentuk *file* dengan format tertentu. Salah satu format *citra* yang dipakai *JPEG* (*JPG*) *Joint Photographic Experts Group JPEG* atau *lossy kompresi* adalah ukuran digunakan oleh gambar di Web. JPEG format umumnya Digunakan untuk foto.

2.4. Algoritma RC6

Algoritma RC6 adalah suatu algoritma kriptografi *block cipher* yang dirancang oleh Ronald L. Rivest, Matt J.B. Robshaw, Ray Sidney, dan Yuqin Lisa Yin dari RSA Laboratories. Algoritma ini pada mulanya dirancang untuk menjadi AES (Advance Encryption Standard). Algoritma RC6 ini berhasil menjadi finalis dan menjadi kandidat kuat untuk menjadi AES walaupun pada akhirnya algoritma ini tidak terpilih menjadi AES melainkan algoritma rinjdael. Versi 1.1 dari RC6 mulai dipublikasikan pada tahun 1998. Dasar desain dari algoritma RC6 ini didasarkan pada pendahulunya yaitu algoritma RC5.

Algoritma RC6 dispesifikasikan dengan notasi RC6-w/r/b. Dimana w adalah ukuran dari *word* dalam *bit*, karena pada RC6 menggunakan 4 buah *register* maka word adalah ukuran blok dibagi 4. R adalah jumlah iterasi, dimana R tidak boleh negatif, dan B adalah panjang kunci dalam bytes.

Dalam rancangan untuk menjadi kandidat AES algoritma RC6 yang digunakan menggunakan ukuran w sebesar 32 bit dan jumlah iterasi r sebesar 20 kali putaran. Cara kerja dari algoritma RC6 adalah menggunakan 4 buah *register* dan menggunakan prinsip *Iterated Block Cipher* yang mengunakan iterasi.

2.4.1. Pembentukan Kunci Internal

Untuk membangkitkan urutan kunci internal yang akan digunakan selama proses enkripsi, algoritma RC6 melakukan proses pembangunan kunci yang identik dengan algoritma RC5, yang membedakan hanyalah pada algoritma RC6, jumlah word yang diambil dari kunci yang dimasukan oleh pengguna ketika melakukan enkripsi ataupun dekripsi lebih banyak. Tujuan dari proses pembangunan kunci tersebut adalah untuk membangun suatu array S yang berukuran 2r+4 dari kunci masukan pengguna sepanjang b bytes ($0 \leq b \leq 255$), array tersebut akan digunakan baik dalam proses enkripsi maupun dekripsi.

Proses untuk membangun kunci-kunci internal menggunakan dua buah konstanta yang disebut dengan “magic constant”. Dua buah *magic constant* Pw dan Qw tersebut didefinisikan sebagai berikut:

Pw = Odd((e-2)2w)…….(2.2)

Qw = Odd((-1)2w)…….(2.3)

Dimana :

e = 2.7182818284859…(basis dari logaritma natural)

Φ = 1.618022988749…(golden ratio)

Odd (x) adalah *integer* ganjil terdekat dari x, jika x genap maka diambil integer ganjil setelah x. Berikut adalah daftar *magic constant* pada beberapa panjang blok dalam heksadesimal:

P_{16} = b7e1

Q_{16} = 9e37

P_{32} = b7e15163

Q_{32} = 9e3779b9

P_{64} = b7e151628aed2a6b

Q_{64} = 9e3779b97f4a7c15

Dengan menggunakan dua buah *magic constant* tersebut, pembangunan kunci terdiri dari tiga tahap :

2.4.1.1 Konversi Kunci Rahasia Dari *Bytes* ke *Words*

Langkah pertama adalah menyalin kunci rahasia K[0..b-1] kedalam sebuah

array L[0..c-1], dimana c = pembulatan keatas(b/u) dan u = w/8, penyalinan tersebut dilakukan secara *little endian*. Untuk semua posisi byte pada L yang kosong diberi nilai nol. Untuk kasus dimana b = 0, maka c = 1 dan L[0] = 0.

Langkah ini dapat dilakukan dengan cara berikut :

```
if c=0 then
c←1
endif
for i←b-1 downto 0 do
L[i/u] ←  (L[i/u]<<<8) + K[i]
Endfor
```

2.4.1.2 Inisialisasi Array S

Langkah kedua adalah melakukan inisialisasi array S agar memiliki pola *pseudo-random bit* tertentu menggunakan progresi aritmatika modulo 2w yang ditentukan dengan Pw dan Qw. Berikut langkah kedua dalam *pseudo code* :

```
S[0] ← Pw
for i←0 to 2r+3 do
S[i] ←S[i-1]+ Qw
Endfor
```

2.4.1.3 Mencampurkan L dan S

Langkah terakhir adalah mencampurkan kunci rahasia dari pengguna yang sudah tersimpan dalam L dengan S sebanyak 3 kali iterasi. Berikut adalah langkah pencampuran tersebut:

```
i←0
j←0
A←0
B←0
V←3*max(c,2r+4)
for index←1 to v do
S[i]_(S[i]+A+B) <<< 3
A←S[i]
L[j]←(L[j]+A+B) <<< (A+B)
B←L[j]
i←(i+1)mod(2r+4)
```

j←(j+1)mod c
endfor

Pembentukan kunci yang dilakukan, mengubah kunci dari user yang panjangnya beragam (0-255) menjadi suatu rangkaian kunci dengan sepanjang word sebanyak 2r +3 buah. Hal ini menjadikan RC6 dapat bekerja dengan kunci masukkan pengguna yang beragam.

Kunci yang dihasilkan oleh proses pembentukan kunci ini memiliki sifat satu arah, sehingga proses pembentukan kunci ini dapat digunakan sebagai fungsi *hash* satu arah. Dengan sifat satu arah tersebut, maka kunci internal akan sangat berbeda dengan kunci yang dimasukkan oleh pengguna, hal ini akan membuat hubungan statistik antara kunci yang dimasukan oleh pengguna dengan plainteks dan cipherteks menjadi lebih rumit karena dalam melakukan enkripsi, kunci yang dipakai adalah kunci internal.

Pada pembentukan kunci internal digunakan iterasi yang cukup banyak baik pada tahap satu, dimana untuk melakukan ekspansi kunci dibutuhkan iterasi, dan pada tahap dua, dimana dibutuhkan iterasi untuk melakukan inisialisai *array* serta pada tahap terakhir yang dibutuhkan untuk menggabungkan dua buah *array*, yang bahkan dilakukan selama tiga kali. Iterasi-iterasi ini membutuhkan waktu yang cukup besar untuk dilakukan.

2.4.2. Proses Enkripsi dan Dekripsi

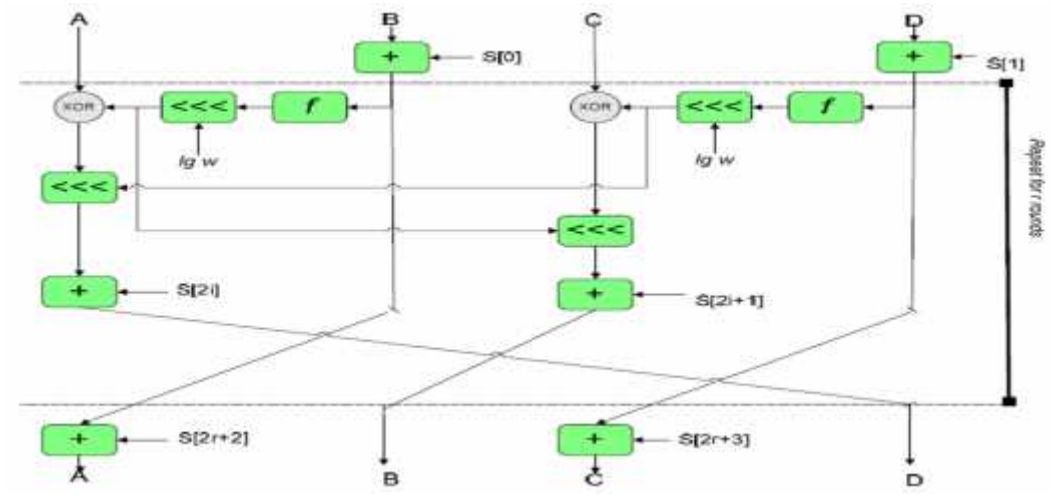
Algoritma RC6 bekerja dengan empat buah register A,B,C,D yang masing-masing berukuran w-bit, register-register tersebut akan diisi oleh plainteks yang kemudian akan digunakan selama proses enkripsi dan setelah proses enkripsi berakhir isi dari register-register tersebut merupakan *cipherteks*. Byte pertama dari *plainteks* atau *cipherteks* akan disimpan pada least significant byte dari A dan byte terakhir dari *plainteks* atau *cipherteks* disimpan pada *most significant byte* dari D.

Proses enkripsi dan dekripsi algoritma RC6 menggunakan enam buah operasi dasar:

1. a + b = penjumlahan *integer* modulo 2w
2. a - b = pengurangan *integer* modulo 2w
3. a⊕b = operasi bitwise *exclusive-or* sebesar w-*bit words*
4. a * b = perkalian *integer* modulo 2w

5. a<<<b = rotasi sejumlah w-*bit word* ke kiri sebanyak jumlah yang diberikan oleh least *sifnificant* lg w *bit* dari b
6. a>>>b = rotasi sejumlah w-bit word ke kanan sebanyak jumlah yang diberikan oleh *least sifnificant* lg w *bit* dari b

Dimana lg w adalah logaritma basis dua dari w. Proses enkripsi dapat dilihat pada Gambar 2.4. dan dekripsi dapat dilihat pada Gambar 2.5. dengan f(x)=x * (2x+1).



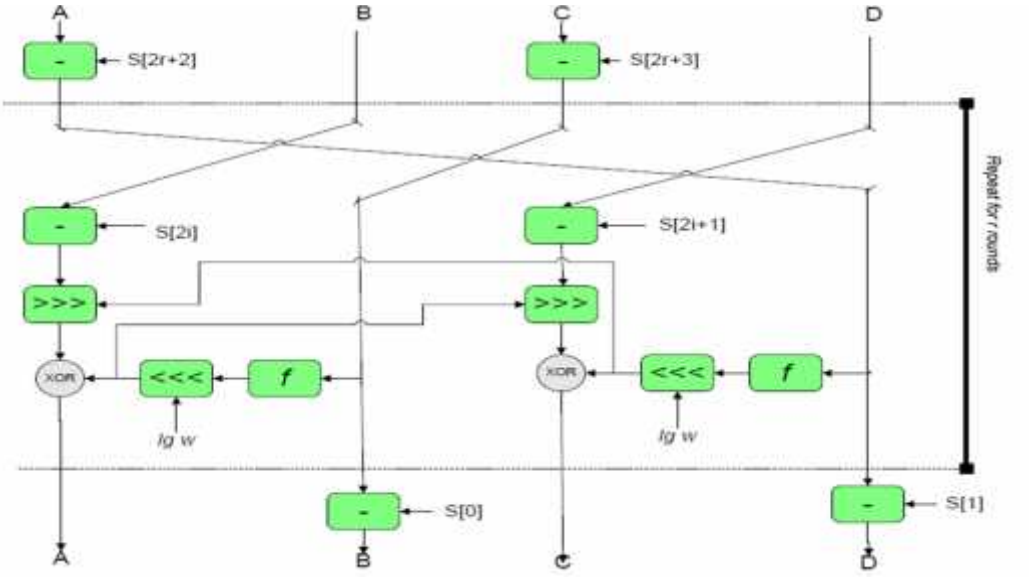
Gambar 2.4. Diagram Enkripsi RC6

Prosedure Enkripsi (Input : *Plainteks* dalam A,B,C,D
r : integer (jumlah rotasi)
S[0..2r+3] : kunci internal
Output : *Ciphertek*s dalam A,B,C,D)
Kamus
u : integer
t : integer

Algoritma
B← B + S[0]
D← D + S[1]
for i← 1 to r do
t← (B * (2B + 1))<<<lg w
u← (D * (2D + 1))<<<lg w
A← ((A⊕t)<<<u) + S[2i]
C← ((C⊕u)<<<t) + S[2i+ 1]
(A,B,C,D)← (B,C,D,A)
endfor

A← A + S[2r + 2]

C← C + S[2r + 3]



Gambar 2.5. Proses Dekripsi RC6

Prosedure Dekripsi (Input : Cipherteks dalam A,B,C,D

r : integer (jumlah rotasi)

S[0..2r+3] : kunci internal

Output : Plainteks dalam A,B,C,D)

Kamus

u : integer

t : integer

Algoritma

C← C - S[2r + 3]

A← A - S[2r + 2]

for i← r downto 1 do

(A,B,C,D)← (D,A,B,C)

u← (D * (2D + 1))<<<lg w

t← (B * (2B + 1))<<<lg w

C← ((C - S[2i+ 1])>>>t) ⊕ u

A← ((A - S[2i])>>>u) ⊕ t

endfor

D← D - S[1]

B← B - S[0]

Langkah-langkah enkripsi algoritma RC6 secara detil adalah sebagai berikut :

1. *Blok Plainteks* dibagi menjadi 4 bagian, A, B, C dan D yang masing-masing memiliki panjang w bit atau panjang blok dibagi 4. Kemudian B dan D dijumlahkan (dalam modulo 2w) dengan kunci *internal* S[0] dan S[1].

   B← B + S[0]

   D← D + S[1]

2. Selanjutnya pada setiap putaran dari 1 sampai r, lakukan XOR dan pergeseran kekiri terhadap A dengan f(x) yang di geser ke kiri sebanyak lg w, dimana f(x) = x* (2x+1) dan x = B. Setelah itu melakukan penjumlahan (dalam modulo 2w) dengan kunci internal. Hal serupa dilakukan pula terhadap C dengan x = D. Kemudian melakukan *swapping*

   A←B, B←C, C←D dan D←A.

   for I ← 1 to r do

   t← (B * (2B + 1))<<<lg w

   u← (D * (2D + 1))<<<lg w

   A← ((A ⊕ t)<<<u) + S[2i]

   C← ((C ⊕ u)<<<t) + S[2i+ 1]

   (A,B,C,D) = (B,C,D,A)

   Endfor

   Fungsi f(x) = x * (2x+1) memiliki keistimewaan dalam diterapkan pada *iterated cipher*, kesitimewaannya adalah fungsi ini memiliki sifat satu ke satu pada aritmatik modulo 2w dan cenderung merubah bit yang *high-order* (dekat MSB). Sifat satu arah tersebut dapat terlihat sebagai berikut :

   Misalkan A dan B adalah bilangan bulat positif dan A B, jika

   $A * (2A + 1) = B * (2B + 1) \pmod{2^w}$, maka:

   $\Leftrightarrow 2A2 + A = 2B2 + B \pmod{2^w}$

   $\Leftrightarrow 2A2 - 2B2 + A - B = 0 \pmod{2^w}$

   $\Leftrightarrow (A - B)(2A + 2B + 1) = 0 \pmod{2^w}$

   Namun, A ≠ B, jadi (A - B) ≠ 0 kemudian, 2A dan 2B merupakan genap sehingga (2A + 2B + 1) merupakan bilangan ganjil dan tidak mungkin nol, maka tidak ada A dan B yang memenuhi A* (2A + 1) = B * (2B + 1) (mod 2 w) atau f(x) bersifat satu ke satu pada modulo 2w .

Sifat satu ke satu ini cenderung berbeda pada bit yang *high-order* atau menuju MSB, hal ini dikarenakan fungsi f(x) = x * (2x + 1) merupakan fungsi kuadratik dimana pada perkalian dua buah bilangan akan cenderung menambah digit didepan. Apabila x pada f(x) yang terdiri dari i bit mengalami perubahan bit pada posisi ke j maka f(x) akan berubah pada bit posisi ke j dan cenderung pada posisi > j.

Dengan sifat satu ke satu pada fungsi f(x) tersebut, maka kemungkinan hasil f(x) yang berulang dalam iterasi-iterasi yang terjadi akan sangat kecil, sehingga semakin banyak jumlah iterasi, maka keamanan akan semakin terjaga, hal ini diperkuat dengan kemungkinan perubahan yang terjadi pada high-order bit sehingga pengaruh perbuatan lebih besar.

Jika tidak terdapat sifat satu ke satu tersebut, algoritma enkripsi akan menjadi tidak baik, karena pada algoritma ini terdapat XOR dan apabila suatu bilangan di XOR-kan 2 kali maka bilangan tersebut akan muncul kembali.

3. Setelah iterasi selesai langkah terakhir adalah melakukan penjumlahan (dalam modulo 2w) terhadap A dan C dengan dua kunci internal terakhir. Setelah semua selesai blok yang terbagi menjadi 4 bagian disatukan kembali.

   A← A + S[2r + 2]

   C← C + S[2r + 3]

   Algoritma RC6 termasuk kedalam *iterated cipher*, kekuatan utama algoritma ini terletak pada iterasi yang dilakukannya. Dengan dilakukannya iterasi yang berulang-ulang dengan menggunakan kunci yang berbeda-beda, maka prinsip *confusion* dan *diffusion* dilakukan secara berulang-ulang pula, sehingga keamanan akan semakin baik.

2.5. Pengertian Android

Android adalah system operasi bergerak (mobile operating system) yang mengadopsi system operasi linux. Android diambil alih oleh google pada tahun 2005 dari android inc, pihak google ingin android ini bersifat terbuka dan gratis

oleh sebab itu hampir semua kode program android berlisensi *open-source apache* yang bearti semua orang yang menggunakanan android dapat *mendownload* penuh *source code* nya.

Disamping itu produsen perangkat keras juga dapat menambahkan extensionnya sendiri kedalam Android sesuai kebutuhan produk mereka. Model pemgembangan sederhana membuat Android menarik vendor-vendor perangkat keras salah satu nya adalah Samsung

Keuntugan utama dari Android ini adalah pendekatan aplikasi secara terpadu, pengembangan hanya berkosentrasi pada aplikasi saja, aplikasi bisa berjalan pada beberapa perangkat yang berbeda selama masih ditenagai oleh android.

2.5.1. Versi Android

Adroid telah mengalami update sejak diluncurkan pertama kali. Dapat dilihat pada Tabel 2.1. dibawah ini.

Versi Android	Dilincurkan	Nama Kode
Beta	5 November 2007	
1.0	23 September 2008	
1.1	9 Febuari 2009	
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 Oktober 2009	Éclair
2.2	20 Mei 2010	Froyo
2.3	6 Desember 2010	Gingerbread
3.0	19 Oktober 2011	Ice cream sandwich
40.1	Sekitar pertengahan 2012	Jelly bean
	Sekitar 2013	Key lime pie

Tabel 2.1. versi Android

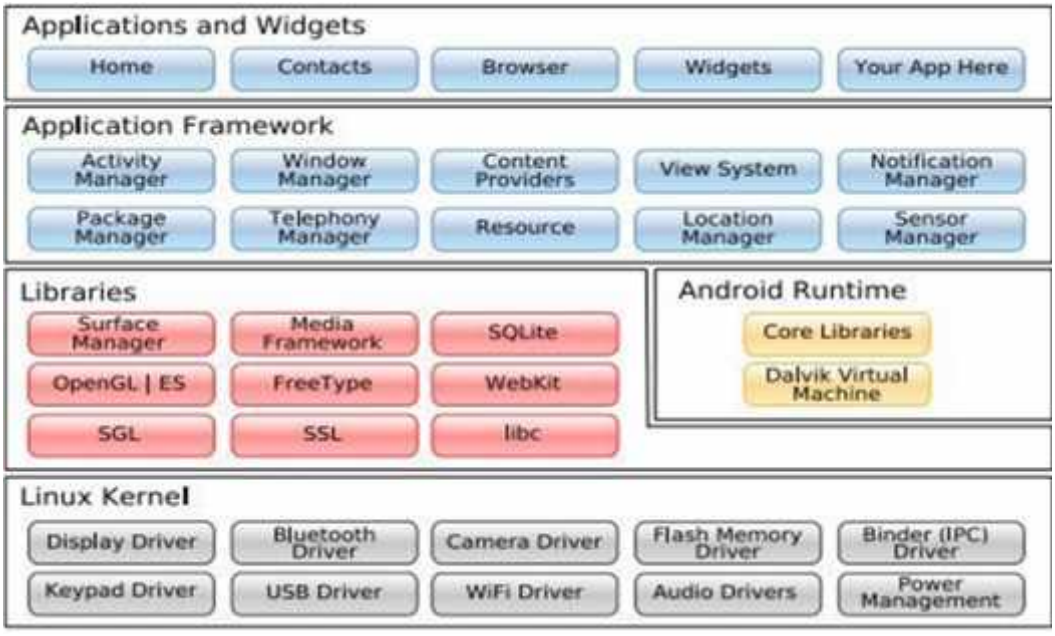
2.5.2. Fitur - Fitur Android

Android tersedia secara *open source* bagi perangkat keras untuk memodifikasi sesuai dengan kebutuhan meskipun konfigurasi perangkat Android tidak sama antara satu dengan yang lain namun android sendiri mendukung fitur-fitur sebagai berikut:

1. Penyimpanan: menggunakan *sqlite* yang merupakan data base relational yang ringan untuk menyimpan data.
2. Koneksi: mendukung GSM/EDGE,IDEN,CDMA,EVDO,UMTS.
3. Pesan: mendukung SMS dan MMS.
4. Web browser: chrome.
5. Media-media yang mendukung MP4, WAV,dan gambar.
6. Hardwere yang terdapat accelerometer sensor, camera digital, digital commpas dan GPS.
7. Multi-touch mendukung layar multi-touch.
8. Multi-tasking mendukung aplikasi multitasking.
9. Dukungan flash terdapat pada Android 2.3 mendukung flash 10.1.

2.5.3. Arsitektur Android

Berikut ini bagan tingkatan-tingkatan system operasi Android



Gambar 2.6. Arsitektur System Operasi Android

Secara garis besar system operasi Android terbagi menjadi lima tingkatan.

1. *Linux Kernel* adalah kernel dasar Android, pada tingkat ini berisi driver perangkat tingkat rendah untuk komponen *hardware* perangkat Android.
2. Libraries berisi semua kode program yang menyediakan layanan-layanan utama system operasi Android, contohnya *library sqlite* yang menyediakan dukungan database sehingga aplikasi android dapat menggunakan menyimpan data *library webkit* yang menyediakan fungsi-fungsi browsing web dan lain-lain.
3. *Android Runtime* kedudukannya setingkat dengan *libraries*, Android runtime menyediakan kumpulan pustaka inti yang dapat diaktifkan oleh pengembang untuk menulis kode aplikasi Android dengan bahasa pemograman java. Dalvik virtual machineakif setiap kali aplikasi Android berproses (Aplikasi Android dikompilasi menjadi Dalvik Executable). Davik adalah mesin semu yang dirancang khusus untuk android yang dapat dioptimalkan daya *battery* perangkat bergerak dengan *memory* dan CPU terbatas.
4. *Application Framework* adalah semacam kumpulan *class built-in* yang tertanam dalam system operasi Android sehingga pengembang dapat memanfaakannya untuk aplikasi yang sedang dibangun.
5. *Apllicattions* pada tingkat ini kita akan bekerja contoh nya aplikasi yang ditemui seperti *phone*, *contact*, *browse* dan lain-lain seperti aplikasi Android pada umumnya yang dapat di*downlod* dan di install dari market Android. Semua aplikasi yang dibuat terletak pada tingkat *applications.*

2.5.4. Perangkat Android

Perangkat Android datang dengan berbagai model dan ukuran, type perangkat yang menggunakan system operasi Android yaitu smartphone, tablet, perangkat E-reader, netbook, MP4 player, internet TV. Sedangkan contoh produk Android misalnya saja Samsung Galaxy, HTC Desere HD dan LG Optimus Ome Smartphones, dan lain-lain

Dalam katagory lain yaitu tablet, tablet memiliki ukuran berbeda, namun umum nya berukuran mulai dari tujuh inchi diagonal contoh nya antara lain

Samsung Tab dan dell steak disamping smartphone dan tablet ada pula Ebook seperti Noble NOOKClour.

Produk lain yang sedang tren adalah TV Smart Android diawali oleh People of Iava sebuah perusahaan swedia mengembangkan TV berbasis Android yang disebut Scandinavia TV Android. Google sendiri juga sedang mengembangkan platform TV cerdas berbasis Android dengan menggandeng beberapa perusahaan Intel, Sony, dan Logitech.

2.5.5. Market Android

Salah satu faktor penentu suksesnya Platform Smartphone adalah karena banyaknya aplikasi yang tersedia oleh karenanya pada agustus 2008 Google mengenalkan Android market yaitu salah satu toko online untuk perangkat Android dan mulai siap menyediakan aplikasi bagi pengguna pada oktober 2008. Dengan aplikasi ini pengguna dengan mudah mendownload aplikasi dari pihak ketiga secara langsung melalui perangkat mereka. Market menyedikan aplikasi berbayar maupun gratis.

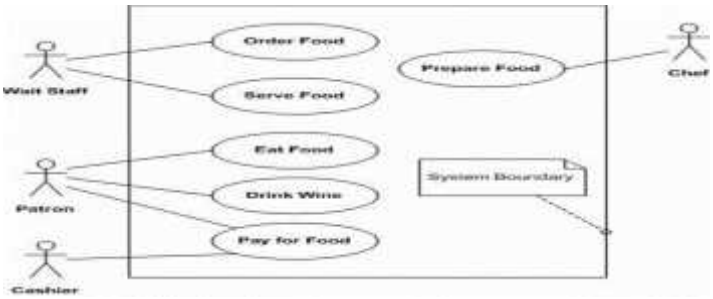
2.6. *Unified Modeling Language* (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yangg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C (Dharwiyanti, 2003).

2.6.1. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan

"bagaimana". Sebuah *use case* mempresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Contoh *use case diagram* terlihat pada Gambar 2.7 dibawah ini.



Gambar 2.7. Contoh *Use Case Diagram*

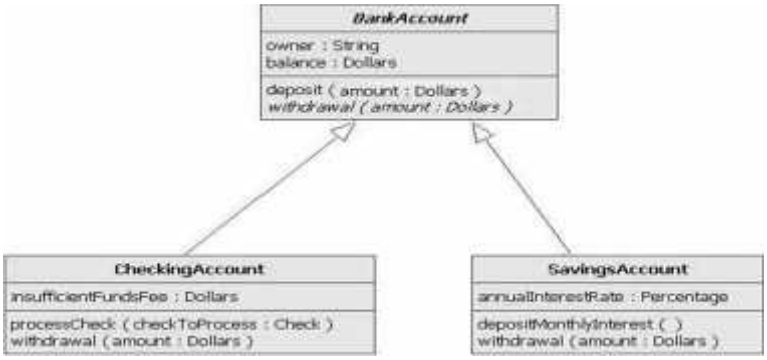
2.6.2. *Class Diagram*

Class didefinisikan sebagai kumpulan/ himpunan objek yang memiliki kesamaan dalam atribut/ properti, perilaku (operasi), serta cara berhubungan dengan objek lain (Nugroho, 2009). *Class* menggambarkan keadaan (atribut/ properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. Terlihat pada Gambar 2.8. dibawah ini.*Class* memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

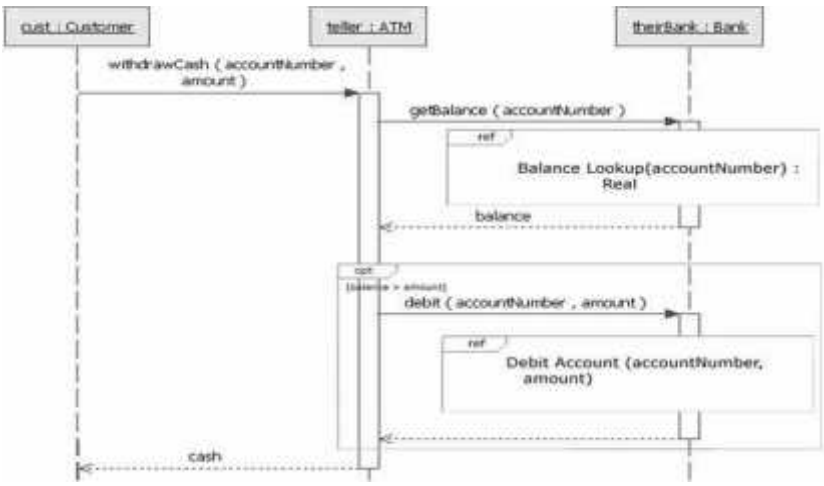


Gambar 2.8 Contoh *Class Diagram*

2.6.3. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Terlihat pada Gambar 2.8. dibawah ini.



Gambar 2.9 Contoh *Sequence Diagram*