BAB II

TINJAUAN PUSTAKA

II.1. Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi sutau paket disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan OpenOffice.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki atarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah. Jenis-jenis *Software* Aplikasi :

a. *Software* aplikasi hiburan, contohnya yaitu winamp untuk mendengarkan musik, games dan sebagainya untuk hiburan.
b. *Softwar*e aplikasi pendidikan yaitu *software* digunakan untuk mempelajari atau mereferensikan tentang pendidikan atau pengetahuan.
c. *Software* aplikasi bisnis yaitu *software* yang digunakan untuk aplikasi bisnis.

d. *Software* aplikasi khusus yaitu *Software* aplikasi untuk produktivitas kerja.(Abdullah dan Erlina. 2012 : 38-45) [3]

II.2. Chat/*Chatting*

Sistem *texting/chatting* merupakan sebuah sistem untuk berkomunikasi melalui *text* antar *device* dalam sebuah jaringan. Dalam implementasinya, aplikasi ini akan sangat efisien dan sangat ekonomis. Aplikasi *chatting* terdiri dari dua sisi yaitu *Server* dan *Client*.

a. *Server* berjalan pada perangkat tertentu dan memiliki sebuah *socket* dan terikat pada port tertentu. *Server* hanya menunggu permintaan *client* untuk melakukan sambungan.
b. *Client* berjalan pada perangkat tertentu juga dimana aplikasi *client* telah mengetahui nama *host* dari *server* yang sedang berjalan dan pada nomor *port server* yang sedang menunggu.(Apriani ; 2014) [4]

II.3. Kriptografi

Kriptografi pada awalnya dijabarkan sebagai ilmu yang mempelajari bagaimana menyembunyikan pesan. Namun pada pengertian modern kriptografi adalah ilmu yang berdasarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas. Jadi pengertian kriptografi modern adalah tidak saja berurusan hanya dengan menyembunyikan namun lebih pada sekumpulan teknik yang menyediakan keamanan informasi.

Berikut ini adalah rangkuman beberapa mekanisme yang berkembang peda kriptografi modern:

1. Fungsi *Hash*

   Fungsi *hash* adalah fungsi yang melakukan pemetaan pesan dengan panjang sembarang ke sebuah teks khusus yang disebut *massage digest* dengan panjang tetap. Fungsi *hash* umumnya dipakai sebagai nilai uji (*check value)* pada mekanisme keutuhan data.

2. Penyadian dengan kunci simetrik (*symmetric key encipherment)*

   Penyandian dengan kunci simetrik adalah penyandian yang kunci enkripsi dan kunci deskripsi bernilai sama. Kunci pada penyandian simetrik yang di asumsikan bersifat rahasia hanya pihak yang melalukan enkripsi dan dekripsi yang mengetahui nilainya. Oleh karena itu penyandian dengan kunci simetrik disebut juga penyandin dengan kunci rahasia *secret key encipherment.*

3. Penyandian dengan kunci asimetrik *(Asymmetric key encipherment)*

   Penyandian dengan kunci asimetrik atau sering juga disebut dengan penyandian kunci publik *(publik key)* adalah penyandian dengan kunci enkripsi dan dekripsi berbeda nilai. Kunci enkripsi yang juga disebut dengan kunci publik *(publik key)* bersifat terbuka. Sedangkan, kunci dekripsi yang juga disebut kunci privat *(private key)* bersifat tertutup atau rahasia.

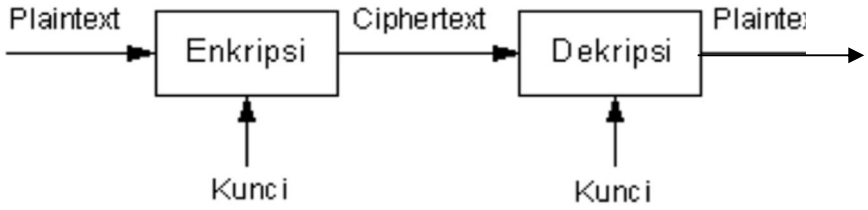
4. Kriptografi adalah ilmu yang berguna untuk mengacak data sedemikian rupa, sehingga tidak bisa di baca oleh pihak ketiga. Tentu saja data yang diacak harus bisa dikembalikan kebentuk oleh pihak yang berwenang. Data diacak biasanya disebut teks asli *(Plain teks)*. Data diacak menggunakan kunci

enkripsi *(Encription key)*. Proses pengacakan di sebut enkripsi *(Encryption)*. Plainteks yang diacak disebut cipher teks. Kemudian proses untuk mengembalikan chiper teks ke plain teks disebut dekripsi *(Decryption)*. Kunci yang digunakan pada deskripsi disebut kunci dekripsi *(Decryption key)*. Pada praktiknya, selain pihak yang berwenang ada pihak ketiga yang selalu berusaha unuk mengembalikan *cipher* text ke *plain* text atau memecahkan kunci deskripsi. Usaha dari pihak ketiga ini disebut kriptonalisis *(cryptanalysis)*. *Cryptography* adalah suatu ilmu atau pun seni mengamankan pesan, dan dilakukan oleh c*ryptographer*. Sedangkan *Cryptanalysis* adalah suatu ilmu dan seni membuka (breaking) *chiperteks* dan orang yang melakukan disebut *Cryptanalyst*.(Kurniadi ; 2015 : 1-13) [1]

II.4. Enkripsi dan Dekripsi

Salah satu hal yang sangat penting dalam komunikasi menggunakan komputer untuk menjamin kerahasiaan data adalah dengan enkripsi. Enkripsi adalah sebuah proses yang melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti (tidak terbaca). Enkripsi dapat diartikan sebagai kode atau *cipher*. Sebuah sistem pengkodean menggunakan suatu tabel atau kamus yang telah didefinisikan untuk mengganti kata dari informasi atau yang merupakan bagian dari informasi yang dikirim. Sebuah *cipher* menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (*stream*) *bit* dari sebuah pesan menjadi *cryptogram* yang tidak dimengerti (*unnitelligible)*.

Dekripsi merupakan algoritma atau cara yang dapat digunakan untuk membaca informasi yang telah dienkripsi untuk kembali dapa Plaintext gan kata lain dekripsi merupakan proses membalikkan hasil yang diberikan dari proses enkripsi ke dalam bentuk awal sebelum dienkrip.



Gambar 2. Proses Enkripsi / Deskripsi sederhana
(Sumber : Kurniadi ; 2015 : 5)

Ada beberapa elemen dari enkripsi yang akan dijabarkan di bawah ini:

a. Algoritma dari Enkripsi dan Dekripsi

Algoritma dari enkripsi adalah fungsi-fungsi yang digunakan untuk melakukan fungsi enkripsi dan dekripsi. Algoritma yang digunakan menentukan keakuratan dari enkripsi, dan ini biasanya dibuktikan dengan basis matematika. Berdasarkan cara memproses teks (*plaintext*), *cipher* dapat dikategorikan menjadi dua jenis : *block cipher* dan *stream cipher*. *Block cipher* bekerja dengan memproses data secara blok, dimana beberapa karakter digabungkan menjadi satu blok. Setiap proses satu blok menghasilkan keluaran satu blok juga. Sementara itu *stream cipher* bekerja memproses masukan (karakter atau data) secara terus menerus dan menghasilkan data pada saat yang bersamaan.

b. Kunci yang digunakan dan panjang kunci

Kekuatan dari penyandian bergantung kepada kunci yang digunakan. Untuk itu, kunci yang lemah tersebut tidak boleh digunakan. Selain itu, panjangnya kunci yang biasanya dalam ukuran *bit*, juga menentukan kekuatan dari enkripsi. Kunci yang lebih panjang biasanya lebih aman dari kunci yang pendek. Jadi enkripsi dengan menggunakan kunci 128-*bit* lebih sukar dipecahkan dengan algoritma enkripsi yang sama tetapi dengan kunci 56-*bit*. Semakin panjang sebuah kunci, semakin besar *keyspace* yang harus dijalani untuk mencari kunci dengan cara *brute force attack* karena *keyspace* yang harus dilihat merupakan pangkat dari bilangan 2. Jadi kunci 128 *bit* memiliki *keyspace* 2128, sedangkan kunci 56-*bit* memiliki *keyspace* 256, artinya semakin lama kunci baru bisa diketahui.

Model-model Enkripsi beserta algoritma yang akan dipakai untuk setiap enkripsi ada 2 hal yang penting yang akan dijabarkan, yaitu enkripsi dengan kunci pribadi dan enkripsi dengan kunci publik.

a. Enkripsi dengan kunci pribadi

Enkripsi dapat dilalukan jika si pengirim dan si penerima telah sepakat untuk menggunakan metode enkripsi atau kunci enkripsi tertentu. Metode enkripsi kunci yang harus dijaga ketat supaya tidak ada pihak luar yang mengetahuinya.

b. Enkripsi dengan kunci publik

Enkripsi ini mempunyai banyak kelebihan, salah satunya adalah tiap orang hanya perlu memiliki satu set kunci, tanpa peduli berapa banyak orang yang akan diajak berkomunikasi.(Kurniadi ; 2015 : 1-13) [1]

II.5. Algoritma Enkripsi RC5

Algoritma enkripsi RC5 didesain oleh Profesor Ronald Rivest dan pertama kali dipublikasikan pada Desember 1994. Sejak publikasinya, RC5 telah menarik perhatian banyak peneliti dalam bidang kriptografi dalam rangka menguji tingkat keamanan yang ditawarkan oleh algoritma RC5 (RSA Laboratory Technical Report TR-602). Pada dasarnya RC5 di desain dengan sedemikian rupa untuk memenuhi syarat-syarat sebagai :

a. RC5 harus merupakan blok *chiper* simetris. Kunci rahasia yang sama dalam kriptografi digunakan dalam enkripsi dan dekripsi. Teks awal dan teks terenkripsi memiliki panjang yang sudah ditentukan dalam blok.
b. RC5 harus cocok untuk *hardware* ataupun *software*. Hal tersebut berarti RC5 hanya akan menggunakan operasi perhitungan primitif yang sering kali ditemukan pada mikroprosessor.
c. RC5 harus cepat, hal tersebut lebih kurang dikarenakan RC5 merupakan algoritma *word-oriented* dengan kata lain pada operasi komputasi dasar yang digunakan harus dapat

   memproses data *word* secara penuh dalam satu waktu.
d. RC5 harus dapat beradaptasi pada berbagai panjang data *word*. Semisal pada prosessor 64 bit, panjang data *word* yang digunakan lebih panjang daripada

prosesor 32 bit. RC5 harus dapat memanfaatkan hal tersebut, oleh karenanya RC5 memiliki parameter w yang yang menandakan panjang *word*.

e. RC5 harus dapat beroperasi dalam berbagai jumlah *round*. Jumlah *round* yang bervariasi memungkinkan pengguna untuk memanipulasi RC5 untuk menjadi lebih cepat dan aman.
f. RC5 harus dapat beroperasi dalam berbagai panjang kunci. Hal tersebut mengakibatkan panjang kunci b menjadi parameter dalam algoritma RC5.
g. RC5 haruslah berstruktur sederhana. Struktur yang sederhana tersebut belum tentu menghasilkan kemanan yang rendah. Namun, struktur sederhana akan memugkinkan analisis dan eveluasi yang cepat untuk menentukan kekuatan algoritma.
h. RC5 harus hemat dalam penggunaan memori. Hal tersebut memungkinkan implemantasi RC5 kedalam *smart-card* atau perangkat lain yang memiliki keterbatasan memori.
i. RC5 harus mengimplementasikan metode *data-dependent rotations*. Metode RC5 merupakan kriptografi primitif yang merupakan sasaran pengkajian RC5. *Data-dependent rotations* merupakan suatu teknik yang merotasi *data* secara sirkuler sebanyak N rotasi.

Seperti yang dijelaskan sebelumnya, algoritma RC5 merupakan metode enkripsi menggunakan metode simetrik dan pengolahan dalam bentuk blok *chiper*, jadi kata kunci yang sama digunakan untuk proses enkripsi dan dekripsi. Parameter-parameter yang digunakan dalam RC-5 adalah sebagai berikut:

a. Round atau jumlah putaran disimbolkan dengan r yang memiki nilai antara 1,

2, 3, 4, ..., 225.

b. Jumlah *word* dalam *bit* disimbolkan dengan w. Jumlah yang disupport adalah 16 *bit*, 32 *bit*, dan 64 *bit*.

c. Kata kunci (*key word*) disimbolkan dengan b dengan range 1, 2, 3, 4,..., 225.

Ada 3 proses utama dalam RC5, yaitu perluasan kunci, enkripsi dan dekripsi. Perluasan kunci merupakan proses membangkitkan kunci internal dengan memanfaatkan komputasi rotasi *left regular shift* (<<<) dan *right regular shift* (>>>), dengan panjang kunci tergantung dari jumlah putaran. Kunci internal kemudian digunakan dalam proses enkripsi dan dekripsi.

Proses enkripsi dibagi menjadi tiga, yaitu: penjumlahan integer, XOR dan rotasi. Untuk lebih jelasnya, adalah sebagai berikut:

1. Enkripsi

Diasumsikan terdapat dua buah blok *input* sebesar w *bit*, A dan B. Dan diasumsikan juga bahwa pembentukan kunci internal telah dilakukan, sehingga array S[0...t-1] telah dihitung. Sehingga *pseudecode* untuk proses enkripsi seperti di bawah:

A← A+ KI[0]

B← B + KI[1]

for i ←1 to r do

A← ((A⊕B) <<< B) + KI[2i]

B← ((B⊕A) <<< A) + KI[2i+1]

Endfor

2. Dekripsi

Algoritma pada proses dekripsi merupakan kebalikan dari proses enkripsi. Jika tadinya digeser ke kiri, maka pada proses dekripsi dilakukan pergeseran ke kanan (*right regular shift*).

for i ←r downto 1 do

B← ((B – KI[2i+1])>>>A) ⊕A

A← ((A – KI[2i])>>>B) ⊕B

endfor

B ←B - KI[1]

A ←A - KI[0]

Untuk mendekripsi cipherteks, diperlukan KI yang sama dengan KI saat mengenkripsi. Proses pembangkitan KI pada kedua proses tersebut juga sama.

3. Pembentukan kunci internal

K[0-1]...K[b] disalin ke tabel L[0-1]...L[b] dengan aturan di-padding dengan karakter 0 hingga ukuran L[i]menjadi w/2 bit. Sebagai contoh:

K[0] = k L[0] = k000

K[1] = r L[1] = r000

K[2] = i L[2] = i000

K[3] = p L[3] = p000

K[4] = t L[4] = t000

K[5] = o L[5] = o000

Kemudian, inisialisasi tabel kunci internal KI dengan ukuran t = 2r + 2 seperti berikut:

KI[0] ←P

for i ←1 to t – 1 do

KI[i] ←KI[i – 1]

Endfor

Algoritma pembentukan kunci internal menggunakan konstanta P dan Q yang didapatkan dari fungsi yang melibatkan bilangan irasional sebagai berikut:

$P = Odd[(e - 2)2^w]$

$Q = Odd[(f – 1)2^w]$

Keterangan:

e = 2.718281828459.....

F = 1.618033988749.....

Kemudian L dan S digabungkan dengan algoritma berikut:

i ←0

j ←0

X ←0

Y ←0

n ←3*max(r,c)

for k ←1 to n do

KI[i] ← (KI[i] + X + Y)<<< 3

X ←KI[i]

i ← (i + 1) mod t

L[j] ← (L[j] + X + Y)<<< 3

Y ←L[j]

j← (j + 1) mod c

endfor

Keterangan:

Max(r,c) adalah fungsi menentukan bilangan terbesar antara r dan c. c adalah nilai maksimal dari panjang kunci b dibagi 4.(Suryawan dan Hamdani ; 2013 : 44-49) [2]

II.6. Android

Menurut Arifianto Teguh, android adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat *mobile,* semua perangkat lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Dalam definisi lain, Android merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware,* dan aplikasi inti yang dirilis oleh Google. Sedangkan Android *SDK (Software Development Kit)* menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform* Android dengan menggunakan bahasa pemrograman Java.

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi Android, sebuah file yang ditandai dengan akhiran *.apk.* file inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. File ini diunduh oleh pengguna ke perangkat *mobile* mereka. Semua kode dijadikan satu

file *.apk,* dan kemudian kita sebut sebagai sebuah aplikasi. (Ahmad ; 2015 : 190-200) [5]

II.7. Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform_independent*). Berikut ini adalah sifat dari Eclipse:

a. *Multi platform* :

   Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.

b. *Multi language* :

   Eclipse dikembangkan dengan bahasa pemrograman Java. Akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.

c. *Multi role* :

   Selain sebagai IDE untuk pengembangan aplikasi, Eclipse bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan *web* dan lain sebagainya. (Murtiwiyati dan Lauren ; 2013 : 1-10) [6]

II.8. Android SDK (*Software Development Kit*)

Android SDK adalah tools API (*Aplication Programming Interface*) yang diperlukan untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi netral, Android memberi kesempatan bagi semua orang untuk membuat aplikasi yang dibutuhkan, yang

bukan merupakan aplikasi bawaan *Handphone/Smartphone*. Beberapa fitur-fitur Android yang paling penting adalah:

a. Mesin *Virtual* Dalvik yang dioptimalkan untuk perangkat *mobile*.
b. *Integrated browser* berdasarkan *engine open source WebKit*.
c. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3Dberdasarkan spesifikasi opengl ES 1.0 (Opsional akselerasi perangkat keras).
d. SQLite untuk penyimpanan data (*database*).
e. Media yang mendukung *audio*, *video*, dan gambar.
f. *Bluetooth*, EDGE, 3G dan WiFi.
g. Kamera, GPS, dan kompas.
h. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse. (Lindung, Yunus Dwi ; 2012) [7]

II.9. AVD (*Android Virtual Device*)

Android *Virtual Device* merupakan *emulator* untuk menjalankan aplikasi android. Setiap AVD terdiri dari:

a. Sebuah profil perangkat keras yang dapat mengatur pilihan untuk menentukan fitur *hardware emulator*. Misalnya, menentukan apakah menggunakan perangkat kamera, apakah menggunakan keyboard QWERTY fisik atau tidak, berapa banyak memori internal, dan lain-lain.
b. Sebuah pemetaan versi Android, maksudnya kita menentukan versi dari platform Android akan berjalan pada *emulator*.

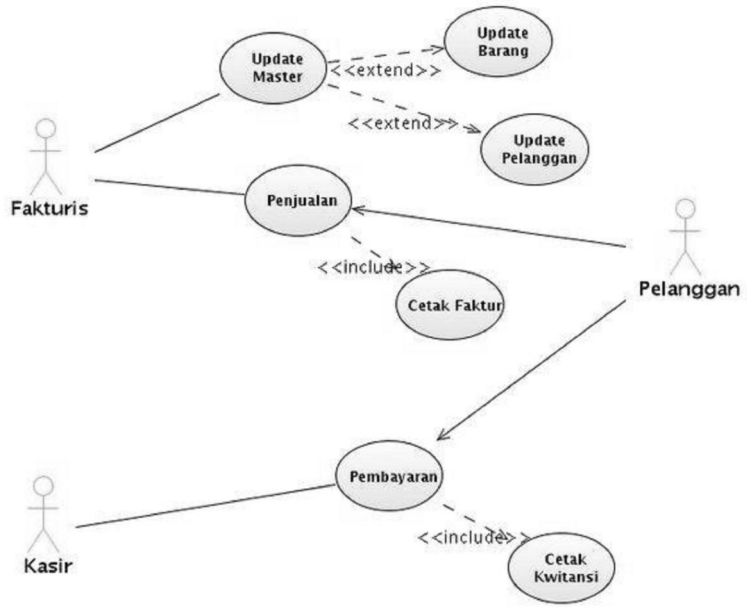
c. Pilihan lainnya, misalnya menentukan *skin* yang kita ingin gunakan pada *emulator*, yang memungkinkan untuk menentukan dimensi layar, tampilan, dan sebagainya. Kita juga dapat menentukan SD *Card virtual* untuk digunakan dengan di *emulator*. (Lindung, Yunus Dwi ; 2012) [7]

II.10. Pengertian UML

UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem (Windu dan Grace ; 2013 : 81). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.10.1. *Use Case Diagram*

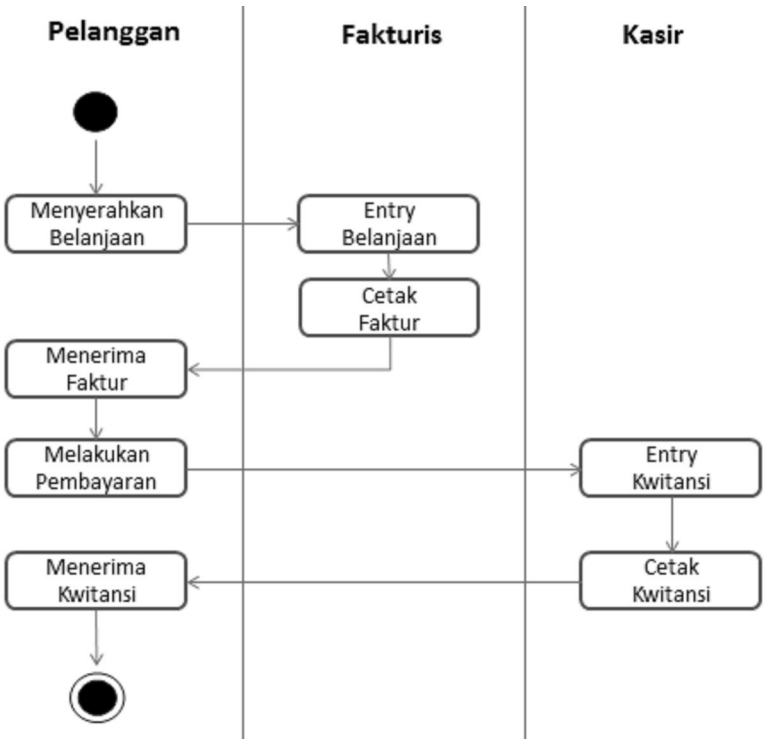
Use case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Windu dan Grace ; 2013 : 81). Contoh pembuatan *use case* diagram dapat dilihat pada gambar II.1. berikut :



Gambar. II.1. *Use Case Diagram*
(Sumber : Windu dan Grace ; 2013 : 83)

II.10.2. *Activity Diagram*

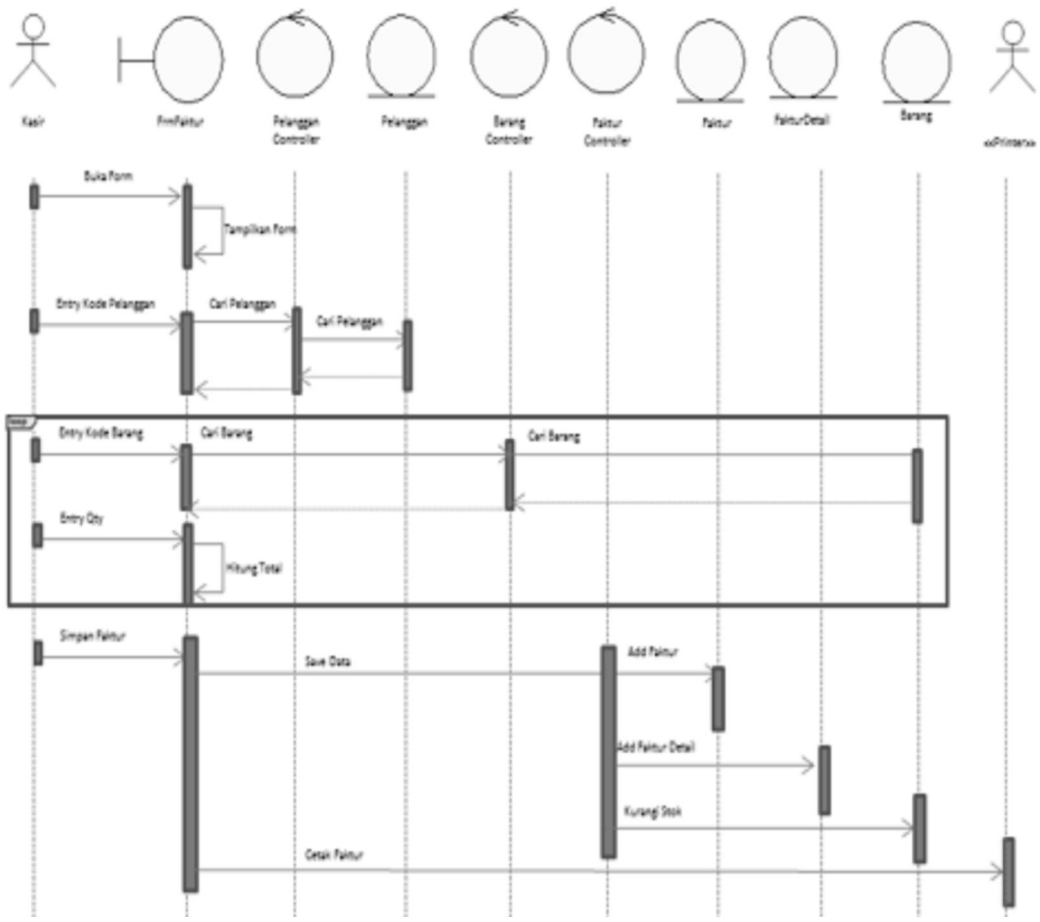
Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Windu dan Grace ; 2013 : 81). Contoh pembuatan *activity diagram* dapat dilihat pada gambar II.2. berikut :



Gambar. II.2. ***Activity Diagram***
(Sumber : Windu dan Grace ; 2013 : 83)

II.10.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Windu dan Grace ; 2013 : 81). Contoh pembuatan *sequence diagram* dapat dilihat pada gambar II.3. :

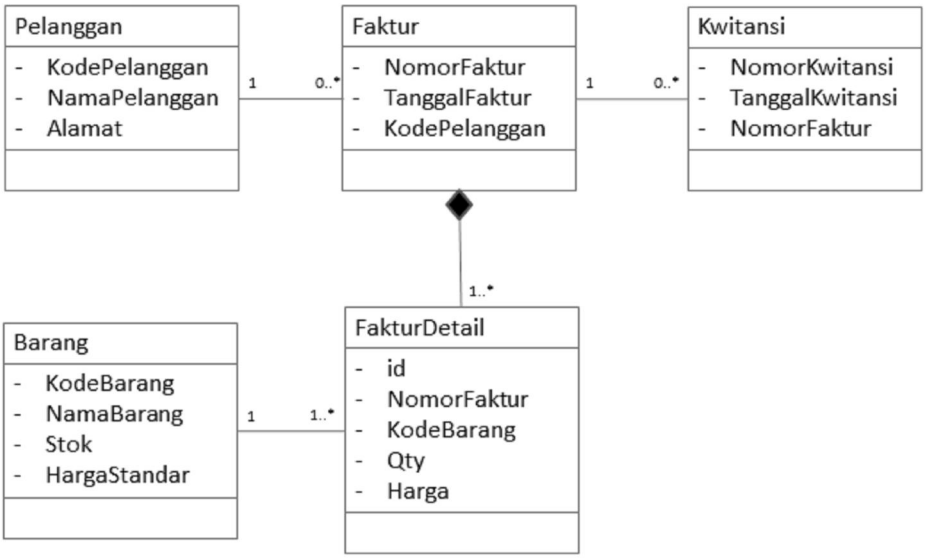


Gambar. II.3. *Sequence Diagram*
(Sumber : Windu dan Grace ; 2013 : 84)

II.10.4. *Class Diagram*

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan obyek yang dikoneksikan. *Class diagram* secara khas meliputi:Kelas (*Class*), *Relasi, Associations, Generalization* dan *Aggregation,* Atribut *(Attributes),* Operasi *(Operations/Method),* dan*Visibility*, tingkat akses

objek eksternal kepada suatu operasi atau atribut. Hubungan antar Kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti (Windu dan Grace ; 2013 : 81). Contoh pembuatan *class diagram* dapat dilihat pada gambar II.4. berikut :



Gambar. II.4. *Class Diagram*
(Sumber : Windu dan Grace ; 2013 : 83)