BAB II

TINJAUAN PUSTAKA

II.1. 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)*

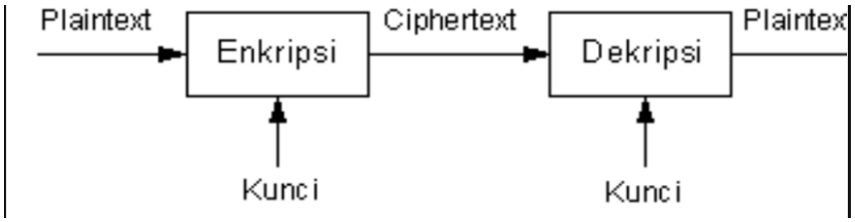
   Penyandian dengan kunci asimetrik atau sering juga disebut dengan penyandian kunci publik *(publik key)* adalah penyandian dengan kunci enkripsi dan dekripsi berbeda nilai. Kunci enkripsi yang juga disebut dengan kunci publik *(publik key)* bersifat terbuka. Sedangkan, kunci dekripsi yang juga disebut 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 tekt. Kemudian proses untuk mengembalikan chiper tekt ke plain tekt deisebut 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*. Sedagkan *Cryptanalysis* adalah suatu ilmu dan seni membuka (breaking) *chipertex* dan orang yang melakukan disebut *Cryptanalyst*.(Kurniadi ; 2015 : 1-13)

II.2. 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 dapat dibaca. Dengan 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)

II.3. Algoritma RC4

Algoritma RC4 ini dikembangkan oleh Ronald Rivest untuk RSA *data security* pada tahun 1987 dan baru dipublikasikan untuk umum pada tahun 1994. RC4 merupakan salah satu algoritma kunci simetris yang berbentuk *stream cipher* dimana algoritma ini melakukan proses enkripsi/dekripsi dalam satu byte dan menggunakan kunci yang sama. RC4 menggunakan variabel yang panjang kuncinya dari 1 sampai 256 bit yang digunakan untuk menginisialisasikan tabel sepanjang 256 bit. Tabel ini digunakan untuk generasi yang berikut dari *pseudo random bit* dan kemudian untuk menggenerasikan aliran *pseudo random* digunakan operasi XOR dengan *plaintext* untuk menghasilkan *ciphertext*. Masing-masing elemen dalam tabel ditukarkan minimal sekali. Berikut ini adalah penggunaan sistem sandi *stream* dengan kunci simertis:

a. Sistem Sandi RC4

Sistem sandi RC4 dikembangkan oleh Ronald Rivest pada tahun 1984 merupakan sistem sandi stream yang paling banyak digunakan misalnya pada protokol SSL/TLS. RC4 merupakan sistem sandi stream berorientasi byte. Masukkan algoritma enkripsi RC4 merupaka sebuah byte, kemudian dilakukan operasi XOR dengan sebuah byte kunci, dan menghasilkan sebuah byte sandi.

b. Penjadwalan Kunci RC4

Sistem sandi RC4 menggunakan state, yaitu larik byte berukuran 256 yang terpermutasi, dan tercampur oleh kunci. Kunci enkripsi juga merupkan larik byte berukuran 256. Sebelum melakukan enkripsi, dan dekripsi, sistem sandi RC4 melakukan inisialisasi terhadap state dengan Algoritma, algoritma ini disebut dengan penjadwalan kunci *(key scheduling)*.

Input: kunci

Output: {S[1],...,S[N]}

For i=0 255 do

S[i]=i

End for

J=0

For i=0 255 do

J=(j+S[i]+Kunci[imod [kumci]]) mod

256

swap(S[i],S[j])

end

c. Enkripsi RC4

Setelah state sandi RC4 merupakan state, yaitu larik byte pada teks asli dikenakan operasi XOR dengan kunci byte untuk menghasilkan byte pada teks sandi, Kunci byte yang digunakan pada enkripsi dibangitkan denagn memanfaatkan state S. Algoritma enkripsi RC4:

Input: P {Stream teks asli}

Output: C{Stream teks sandi}

i=0, j=0 {bisa diisi nilai lain}

While P masih memiliki byte do i=(i+1)

mod 256

j=(j+S[i]) mod 256

swap (S[i], S[j])

k= S[S[i]+S[j]] mod 256

C=Pok

End while

d. Dekripsi RC4

Algoritma dekripsi sistem sandi RC4 serupa dengan algoritma enkripsi sistem RC4.

e. Implementasi Sistem Sandi RC4

f. Ukuran kunci sandi RC4 sangat berpengaruh terhadap keamanan sistem sandi RC4. RC4 telah dibuktikan tidak aman untuk ukuran kunci yang kecil, yaitu dibawah 5 byte. Rekomendasi penggunaan sistem sandi RC4 agar memiliki

keamanan yang kuat adalah:

1. Ukuran kunci sama atau lebih besar daripada 256 bit(16byte)
2. Setiap sesi baru membangkitkan kunci yang baru (dengan pembangkitan kunci yang baru menghindari penyerang untuk melakukan analisis sandi diferensial pada sistem sandi).

g. Implementasi sistem Sandi RC4

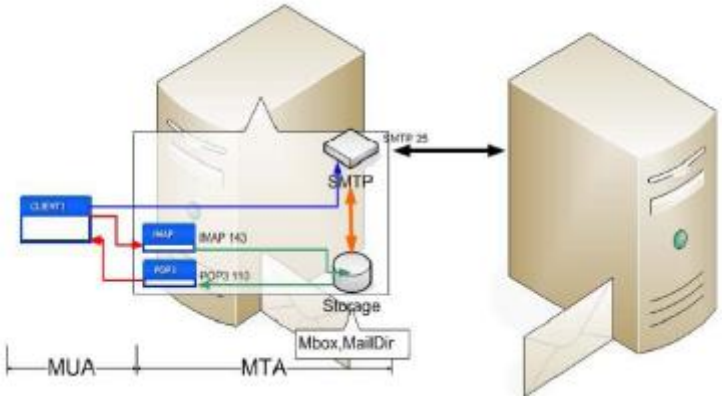
Sistem sandi RC4 mudah diimplementasikan dalam perangkat lunak karena beroperasi pada tipe data *byte*. (Kurniadi ; 2015 : 1-13)

II.4. *Email*

Surat elektronik atau pos elektronik (bahasa Inggris: *email*) adalah sarana mengirim surat melalui jalur jaringan komputer (misalnya *internet*). Dengan surat biasa pada umumnya pengirim perlu membayar per pengiriman (dengan membeli perangko), tetapi surat elektronik umumnya biaya yang dikerluarkan adalah biaya untuk membayar sambungan *internet*. *Email* merupakan aplikasi TCP/IP yang paling banyak digunakan. *Email* adalah pesan yang terdiri atas kumpulan string ASCII dalam format RFC 822. Sistem *email* yang beroperasi di atas jaringan berbasis pada model *store and forward*. Sistem ini mengaplikasikan sebuah sistem server *email* yang menerima, meneruskan, dan mengirimkan, serta menyimpan pesan-pesan *user*, dimana *user* hanya perlu untuk menghubungkan komputer mereka kedalam jaringan.

Email yang dikirim belum tentu akan diteruskan komputer penerima (*end user*), tapi disimpan atau dikumpulkan dahulu dalam sebuah komputer server (*host*) yang akan *online* secara terus menerus (*continue*) dengan media

penyimpanan (*storage*) yang relatif besar dibanding komputer biasa. Hal ini bisa diibaratkan dengan sebuah kantor pos, jika seseorang mempunyai alamat (*mailbox*), maka dia dapat memeriksa secara berkala jika dia mendapatkan surat. Komputer yang melayani penerimaan *email* secara terus-menerus tersebut biasa disebut dengan *mailserver* atau *mailhost*. Secara umum, sistem kerja dari *email* adalah seperti gambar II.1.



Gambar II.1. Sistem Kerja *Email*

(Sumber : Wibowo dan Suprayogi ; 2014 : 77)

Simple Mail Transfer Protocol (SMTP) adalah suatu protokol yang digunakan untuk mengirimkan pesan *e-mail* antar *server*, yang bisa dianalogikan sebagai kantor pos. Ketika kita mengirim sebuah *e-mail*, komputer kita akan mengarahkan *e-mail* tersebut ke sebuah SMTP *server*, untuk diteruskan ke *mailserver* tujuan. *Mail-server* tujuan ini bisa dianalogikan sebagai kotak pos dipagar depan rumah kita, atau kotak PO BOX di kantor pos. *Email-email* yang terkirim akan "bertengger" di tempat tersebut hingga Si pemiliknya mengambilnya. Urusan pengambilan *email* tersebut tergantung kapan dipenerima memeriksa *account email*-nya.

Post Office Protocol version 3 adalah suatu protokol yang berfungsi untuk menarik atau mengambil *email* dari *server email* yang digunakan. Untuk menggunakan POP3 bisa dari Microsoft Outlook. biasanya untuk menggunakan POP3 di perlukan settingan:

a. *Email Address* : anda@domainanda.com
b. *Incoming Mail* (POP3, IMAP or HTTP) *server* : mail.doaminanda.com
c. *Outgoing* (SMTP) *server* : mail.domainanda.com
d. *Account Name* : anda@domainanda.com
e. *Password* : *password* yang telah anda buat sebelumnya
f. Protokol POP3 dibuat karena desain dari sistem surat elektronik yang mengharuskan adanya i surat elektronik yang menampung surat elektronik untuk sementara sampai surat elektronik tersebut diambil oleh penerima yang berhak. Kehadiran *server* surat elektronik ini disebabkan kenyataan hanya sebagian kecil dari komputer penerima surat elektronik yang terus-menerus melakukan koneksi ke jaringan *internet*. Protokol ini di spesifikan pada RFC 1939. (Wibowo dan Suprayogi ; 2014 : 75-83)

II.5. 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.

a. 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)

II.6. 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)

II.7. 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)

II.8. 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)

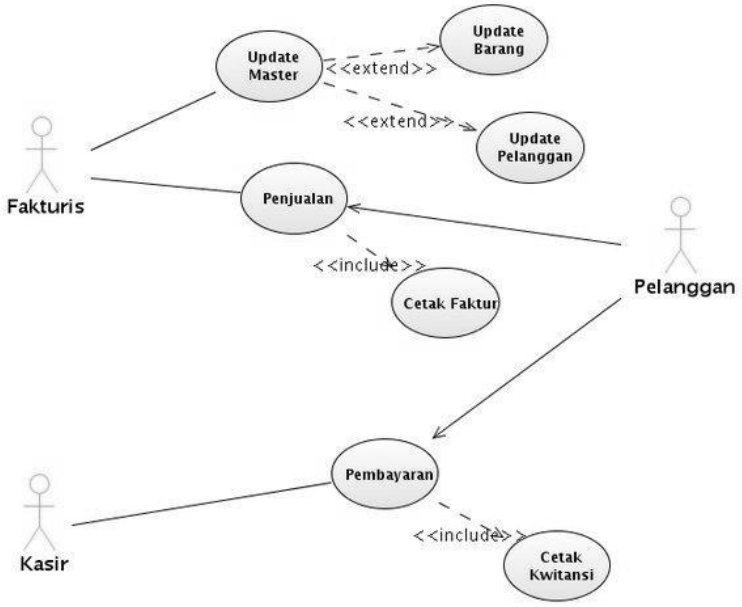
II.9. Pengertian UML

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.9.1. *Use Case Diagram*

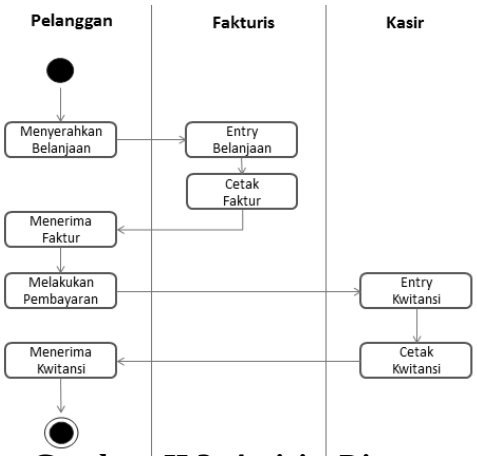
Use case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam 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.2.



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

II.9.2. *Activity Diagram*

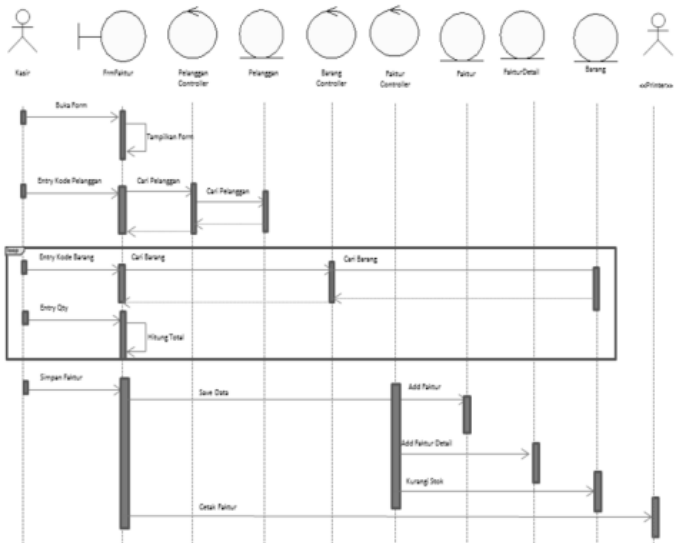
Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Windu dan Grace ; 2013 : 81). Contoh pembuatan *activity diagram* dapat dilihat pada gambar II.3. berikut:



Gambar. II.3. *Activity Diagram*
(Sumber : Windu dan Grace ; 2013 : 83)

II.9.3. *Sequence Diagram*

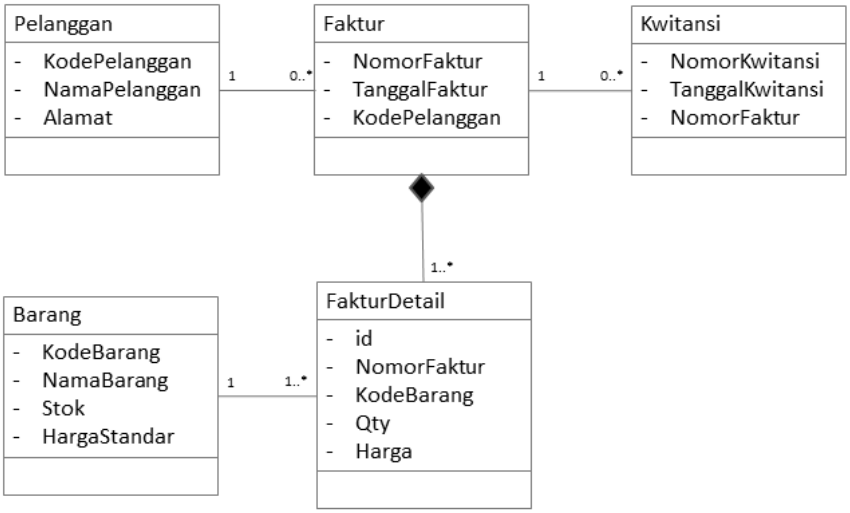
Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Windu dan Grace ; 2013 : 81). Contoh pembuatan *sequence diagram* dapat dilihat pada gambar II.4. :



Gambar. II.4. *Sequence Diagram*
(Sumber : Windu dan Grace ; 2013 : 84)

II.9.4. *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.5. berikut :



Gambar. II.5. *Class Diagram*
(Sumber : Windu dan Grace ; 2013 : 83)