BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Bedasarkan penelitian yang dilakukan oleh Arisantoso, dkk (2017) mengenai Penerapan Aplikasi Pengamanan Data Dengan Metode *Enkripsi* Dan *Dekripsi* Algoritma AES Dalam Jaringan Lokal *Area*, Arisantoso, dkk menyimpulkan bahwa menerapkan aplikasi pengamanan data menggunakan bahasa pemrograman *Microsoft Visual Studio.Net*. Cara kerja pengenkripsian pada kirim dan terima data yaitu pengguna mengirimkan data melalui jaringan lokal *area* lalu penerima data menginstal, membuka *file* dengan hak aksesnya (*password*) dari pengirim. Cara mengembalikan data yang orisinal tanpa mengalami cacat yaitu dengan cara *dekripsi* artinya kapasitas *file* yang telah di*enkripsi* dan kapasitas *file* hasil *dekripsi* sama dengan *file* asli sebelum di*enkripsi*.

Bedasarkan penelitian yang dilakukan oleh Susanto, dkk (2016) mengenai Aplikasi *Enkripsi* Dan *Dekripsi* Untuk Keamanan Dokumen Menggunakan AES Dengan Memanfaatkan *field*, Susanto, dkk menyimpulkan bahwa untuk memudahkan penggunaan algoritma AES, maka dibuat suatu program algoritma AES dengan alat bantu *software* komputer yang dapat mengenkripsi dan mendekripsi data dan memanfaatkan *field* sebagai salah satu kunci dalam penggunaan aplikasi AES.

Bedasarkan penelitian yang dilakukan oleh Aulia, dkk (2016) mengenai Aplikasi *Enkripsi* Dan *Dekripsi* Menggunakan *Visual Basic* 2012 Dengan Algoritma AES, Aulia, dkk menyimpulkan bahwa untuk memudahkan penggunaan algoritma AES, maka dibuat suatu program algoritma AES dengan alat bantu *software* komputer, yaitu *android* yang dapat mengenkripsi data.

II.2. Landasan Teori

II.2.1. Perancangan

Untuk membuat tampilan yang menarik memang tidak mudah dilakukan. Seorang perancang tampilan selain harus mempunyai jiwa seni yang memadai, ia juga harus mengerti selera pengguna secara umum. Hal lain yang pelu disadari oleh seorang perancang tampilan adalah bahwa ia harus bisa meyakinkan pemrogramnya bahwa apa yang ia bayangkan dapat diwujudkan dengan peranti bantu yang tersedia (Insap Santoso ; 2014 : 185).

Perancangan merupakan proses pengolahan hasil analisis perangkat lunak menjadi rencana pengembangan perangkat lunak dan batasan-batasan perangkat lunak atau masalah yang mungkin dihadapi dalam pengembangan perangkat lunak. Perancangan yang dilakukan meliputi perancangan arsitektur, perancangan modul, dan perancangan antarmuka.

Bagi perancang antarmuka, hal yang sangat penting untuk ia perhatikan adalah mendokumentasikan semua pekerjaan yang dilakukan. Dokumentasi rancangan dapat dikerjakan atau dilakukan dengan beberapa cara :

1. Membuat sketsa pada kertas
2. Menggunakan peranti purwarupa GUI
3. Menuliskan keterangan yang menjelaskan tentang kaitan antara jendela.
4. Menggunakan peranti bantu CASE (*Computer Aided Software Engineering*).

Cara kedua dan keempat tidak selalu dapat diterapkan, karena peranti tersebut biasanya harus dibeli dan seringkali cukup mahal. Cara ini kebanyakan diterapkan pada pembuatan antarmuka grafis untuk suatu jenis pekerjaan berskala besar.

II.2.2. Cara Pendekatan

Sebuah program aplikasi pastilah ditujukan kepada pengguna, yang utama, bukan perancangan program aplikasi tersebut. Program aplikasi pada dasarnya dapat dikelompokkan dalam dua kategori besar, yakni program aplikasi untuk keperluan khusus dengan pengguna yang khusus pula dan program aplikasi yang akan digunakan oleh pengguna umum, yang juga sering dikenal dengan sebutan public software. Karena perbedaan pada calon pengguna, maka perancang program antarmuka perlu memperhatikan hal ini (Insap Santoso ; 2014 : 186).

Pada kelompok pertama, yakni pada program aplikasi untuk keperluan khusus, misalnya program aplikasi untuk inventori gudang, pengelolaan data akademis mahasiswa, pelayanan reservasi hotel, dan program-program aplikasi yang serupa, kelompok calon pengguna yang akan memanfaatkan program aplikasi tersebut dapat dengan mudah diperkirakan, baik dalam hal keahlian pengguna maupun ragam antarmuka yang akan digunakan. Untuk kelompok ini ada satu pendekatan yang dapat dilakukan, yakni pendekatan yang disebut dengan

pendekatan perancangan berpusat ke pengguna (*user centered design approach*). Cara pendekatan ini berbeda pendekatan perancangan oleh pengguna (*user design approach*).

Pendekatan perancangan berpusat ke pengguna adalah perancangan antarmuka yang melibatkan pengguna. Pelibatan pengguna di sini tidak diartikan bahwa pengguna harus ikut memikirkan bagaimana implementasinya nanti, tetapi pengguna diajak untuk aktif berpendapat ketika perancangan antarmuka sedang menggambar wajah antarmuka yang mereka inginkan. Dengan kata lain, perancangan dan pengguna duduk bersama-sama untuk merancang wajah antarmuka yang diinginkan pengguna. Pengguna menyampaikan keinginannya. Sementara perancangan menggambar keinginan pengguna tersebut sambil menjelaskan keuntungan dan kerugian wajah antarmuka yang diinginkan oleh pengguna, seolah-olah sudah mempunyai gambaran nyata tentang antarmuka yang nanti akan mereka gunakan (Insap Santoso ; 2014 : 187).

Pada perancangan oleh pengguna, pengguna sendirilah yang merancang wajah antarmuka yang diinginkan. Di satu sisi, cara ini akan mempercepat proses pengimplementasian modul antarmuka. Tetapi di sisi yang lain, hal ini justru sangat memberatkan pemrogram karena apa yang diinginkan pengguna belum tentu dapa diimplementasikan dengan mudah, atau bahkan tidak dapat dikerjakan dengan menggunakan peranti bantu yang ada.

Perancang program aplikasi yang dimasukkan dalam kelompok kedua, aau public *software*, perlua menganggap bahwa program aplikasi tersebut akan digunakan oleh pengguna dengan berbagai tingkat kepandaian dan karakteristik

yang sangat beragam. Di satu sisi keadaan ini dapat ia gunakan untuk memaksa pengguna menggunakan antarmuka yang ia buat, tetapi pada sisi lain pemaksaan itu akan berakibat bahwa program aplikasinya menjadi tidak banyak penggunanya. Satu kunci penting dalam pembuatan modul antarmuka untuk program-program aplikasi pada kelompok ini adalah dengan melakukan customization. Dengan customization pengguna dapat menggunakan program aplikasi dengan wajah antarmuka yang sesuai dengan selera masing-masing pengguna.

Salah satu contoh dari adanya kemampuan yang dimiliki oleh sebuah program aplikasi atau sistem operasi yang dapat disesuaikan dengan karakteristik pengguna adalah pengaturan desktop pada OS X versi 10.5 favoritnya, sehingga pengguna dapat mengubahnya sesuai keinginan justru akan membuat mata pengguna itu sakit, dikarenakan mata harus melakukan akomomasi maksimum terus menerus untuk menyesuaikan dengan warna tampilan yang ada.

Selain cara pendekatan yang dijelaskan di atas, yang terbiasa menulis program-program aplikasi mungkin mempunyai cara khusus untuk berhadapan dengan pengguna. Tetapi perlu diingat bahwa apapun cara yang Anda gunakan, tetap harus mempunyai pedoman bahwa pada akhirnya program itu bukan untuk Anda sendiri, tetapi akan digunakan oleh orang lain. Dengan kata lain, jangan pernah mengabaikan pendapat (calon) pengguna program aplikasi Anda. (Insap Santoso ; 2014 : 188).

II.2.3. Prinsip Dan Petunjuk Perancangan

Antarmuka pengguna secara alamiah terbagi menjadi empat komponen model pengguna, bahasa perintah, umpan balik, dan penampilan informasi. Model pengguna merupakan dasar dari tiga komponen yang lain (Insap Santoso ; 2014 : 188).

Model mental pengguna merupakan model konseptual yang dimiliki oleh pengguna ketika ia menggunakan sebuah sistem atau program aplikasi. Model ini memungkinkan seorang pengguna untuk mengembangkan pemahaman mendasar tentang bagian yang yang dikerjakan oleh program, bahkan oleh pengguna yang sama sekali tidak mengetahui teknologi komputer. Dengan pertolongan model itu pengguna dapat mengantisipasi pengaruh suatu tindakan yang dilakukan dan dapat memilih strategi yang cocok untuk mengoperasikan program tersebut. Model pengguna dapat berupa suatu simulasi tentang keadaan yang sebenarnya dalam dunia nyata, sehingga ia tidak perlu mengembangkannya sendiri dari awal.

Setelah pengguna mengetahui dan memahami model yang di inginkan, dia memerlukan peranti untuk memanipulasi model itu. Peranti pemanipulasian model ini sering disebut dengan bahasa perintah (command language), yang sekaligus merupakan komponen kedua dari antarmuka pengguna. Idealnya program komputer kita mempunyai bahasa perintah yang alami, sehingga model pengguna dengan cepat dapat dioperasionalkan. (Insap Santoso ; 2014 : 189).

Komponen ketiga adalah umpan balik. Umpan balik di sini diartikan sebagai kemampuan sebuah program yang membantu pengguna untuk mengoperasikan program itu sendiri. Umpan balik dapat berbentuk pesan

penjelasan, pesan penerimaan perintah, indikasi adanya obyek terpilih, dan penampilan karakter yang diketikkan lewat papan ketik. Beberapa bentuk umpan balik terutama ditujukan kepada pengguna pengguna yang belum berpengalaman dalam menjalankan program sebuah aplikasi. Umpan balik dapat digunakan untuk member keyakinan bahwa program telah menerima perintah pengguna dan dapat memahami maksud perintah tersebut.

Komponen keempa adalah tampilan informasi. Komponen ini digunakan untuk menunjukkan status informasi atau program ketika pengguna melakukan suatu tindakan. Pada bagian ini perancang harus menampilkan pesan-pesan tersebut seefektif mungkin sehingga mudah dipahami oleh pengguna. Setelah memahami beberapa prinsip dalam perancangan antarmuka pengguna. Pada bagian berikut ini akan diberikan petunjuk singkat tentang perancangan anarmuka yang akan Anda lakukan sebagai seorang perancang tampilan.

II.3. Aplikasi

Aplikasi merupakan suatu subteks perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan satu tugas yang diinginkan pengguna. Pengertian dari *mobile applications* adalah aplikasi perangkat lunak yang dibuat khusus untuk dijalankan di dalam tablet dan juga *smartphone*. Umumnya, *developer mobile apps* memerlukan IDE atau *Integrated Development Environments* dan juga SDK untuk pengembangan dari *mobile apps* itu sendiri. Pada saat ini, pada *smartphone* dan juga *tablet*, ada satu aplikasi yang berguna untuk menyediakan berbagai macam aplikasi yang dapat dijalankan di

device tersebut. Aplikasi ini sering disebut *store*. Contoh *store* yaitu *apple apps store*, *Samsung apps*, *amazon kindlefire*, *windows store* dan *google playstore*. http://ausdroid.net/).(Edy Irwansyah; 2014:61)

Jika membahas tentang *Mobile Apps*, umumnya tidak bias tidak menyinggung soal *Developer*. *Developer* ialah badan usaha yang membuat aplikasi yang nantinya akan dijalankan dalam *device*. Pada dasarnya, aplikasi akan berjalan menggunakan tenaga baterai dan juga mendapat dukungan dari prosesor yang ukurannya lebih kecil disbanding dengan prosesor komputer. Sebelum dilempar ke pasar, umunya *mobile apps* akan dites terlebih dahulu menggunakan *emulator*. *Emulator* merupakan salah satu cara untuk menghemat biaya yang dikeluarkan oleh *developer* unuk membuat *mobile apps*.(Edy Irwansyah; 2014:62)

II.4. *Encryption*

Untuk dapat membahas arti penting dari jumlah karakter pada sebuah *password*, harus pertama-tama mengenal apa yang namanya sistem enkripsi. Singkatnya enkripsi itu sendiri adalah sebuah teknik yang melakukan pengubahan informasi (berupa tulisan) menjadi sesuatu yang tidak bisa dimengerti orang lain, dengan menggunakan sebuah kunci matematis. Orang lain tidak akan pernah mengerti apa arti dari kata yang sudah diubah (diacak) oleh proses enkripsi apabila mereka tidak memiliki kuncinya. Kunci daripada proses enkripsi bisa berupa aturan, angka, arahan, kata maupun kalimat. (ThOR ; 2014 : 27)

Analogi mudah, misalnya saya ingin melakukan enkripsi kalimat ”saya ingin makan” dan kuncinya adalah sebuah perhitungan matematis Matris 8x8,

mungkin kalimat tersebut akan menjadi seperti 9xKKL^%ASNK 40lv. Beberapa contoh enkripsi yang paling banyak digunakan di dunia keamanan informasi adalah RSA, MD5 dan SHA1. (ThOR ; 2014 : 28)

Banyak orang salah mengartikan *Encoding* dan Enkripsi (*Encryption*) sebagai suatu hal yang sama. Kenyataannya kedua hal ini berbeda jauh. Walaupun memiliki fungsi yang sama, yakni melakukan pengacakan informasi dan membuat sebuah informasi menjadi tidak bisa dimengerti oleh orang yang tidak memiliki hak untuk memilikinya, namun ada sebuah perbedaan besar antara *Encoding* dan Enkripsi.

II.5. Pengenalan *Database* dan PHP

Database merupakan kumpulan *file* yang saling berhubungan. Akan tetapi *database* tidak hanya kumpulan *file*. *Record* di dalam tiap *file* harus dapat dihubungkan dengan *record* di dalam *file* lain.

Dalam manajemen *database* relational terdapat komponen utama dalam konsep *database* :

1. *Field* adalah unit terkecil data yang disimpan dalam *database*. Unit terkecil data yang disimpan dalam *database*.
   a. *Primary key* yaitu *field* yang unik dan mengidentifikasi satu *record*. Contoh *customer number* dan *order number*.
   b. *Secondary key* yaitu *field* yang mengidentifikasi sebuah *record* atau bagian dari beberapa *record* yang terkait.

c. *Foreign key* yaitu field yang menunjuk beberapa *record* pada *file* lain. Contoh *Order Record* berisi *foreign key customer number*.

d. *Descriptive field* yaitu non *key field*.

2. *Record* adalah kumpulan *field* yang diatur dalam format yang predermined (telah ditentukan).

a. Fixed length record structures

Sebagian besar teknologi *database* memaksakan struktur *record fixed length*, dalam artian setiap *instance record* mempunyai *field* yang sama, jumlah *field* yang sama, dan ukuran logika yang sama. Akan tetapi beberapa sistem *database* akan mengkompresi *field-field* dan nilai-nilai yang tidak digunakan untuk menghemat ruang penyimpanan disk.

b. Variable length record structured

Memperoleh *record-record* pada *file* yang sama memiliki *length* yang berbeda.

3. *Field* dan Tabel

File adalah kumpulan semua kejadian dari struktur record yang ditentukan. Tipe-tipe dari *file* yaitu :

a. File induk / master adalah *file* penting dalam sistem dan akan tetap ada selama siklus hidup sistem informasi berputar.

b. *File* transaksi adalah *file* yang digunakan untuk merekam data dari suatu transaksi yang terjadi.

c. *File* laporan adalah *file* yang berisi sistem informasi yang akan ditampilkan.

d. *File* sejarah adalah *file* yang berisi data masa lalu yang sudah tidak aktif lagi.

e. *File pelindung adalah salinan dari file*-file yang masih aktif di *database* pada saat tertentu yang digunakan bila *file database* rusak.

f. *File* kerja adalah suatu proses program secara sementara karena memori komputer tidak mencukupi.

II.5.1. Mengenal MySQL

MySQL merupakan *database server open source* yang cukup populer keberadaannya. Dengan berbagai keunggulan yang dimiliki, membuat software database ini banyak digunakan oleh para praktisi untuk membangun suatu project. Adanya fasilitas API (*Application Programming Interface*) yang dimiliki oleh MySQL, memungkinkan bermacam-macam aplikasi komputer yang ditulis dengan berbagai bahasa pemrogramman dapat mengakses basis data MySQL (Wahana Komputer ; 2010 : 2).

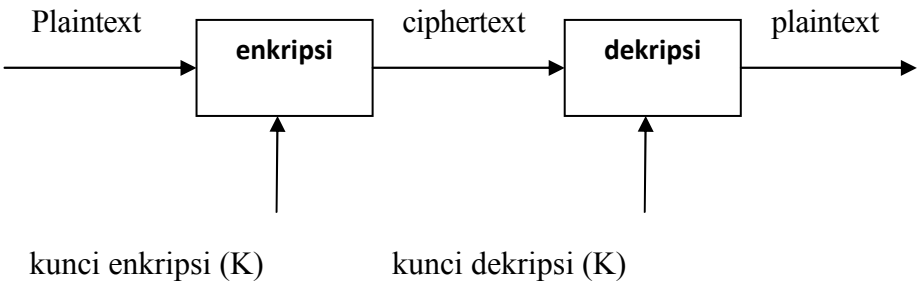
II.5.2. Mengenal PHP

Kelebihan PHP yang paling terasa adalah tersedianya PHP parser di banyak platform. Anda bisa menjalankan skrip PHP di banyak *server*, seperti Apache dan IIS dan di banyak sistem operasi. Untuk melihat halaman download PHP dapat dilihat pada Gambar II.1.

Gambar II.1 Halaman PHP
(Sumber : Ali Zaki ; 2013 : 32)

II.6. Algoritma

Algoritma simetris *(symmetric algorithm)* adalah suatu algoritma dimana kunci enkripsi yang digunakan sama dengan kunci dekripsi sehingga algoritma ini disebut juga sebagai *single-key algorithm.*



Gambar II.2 ***Algoritma Simetris***

Sebelum melakukan pengiriman pesan, pengirim dan penerima harus memilih suatu suatu kunci tertentu yang sama untuk dipakai bersama, dan kunci ini haruslah rahasia bagi pihak yang tidak berkepentingan sehingga algoritma ini disebut juga algoritma kunci rahasia *(secret-key algorithm).*

Kelebihan :

1. Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetrik.
2. Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem

real-time

Kelemahan :

1. Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.

2. Permasalahan dalam pengiriman kunci itu sendiri yang disebut *key distribution problem*. Contoh algoritma : *TwoFish*, *Rijndael*, *Camellia*.

II.6.1. Sejarah AES

AES merupakan nama untuk *Federal Information Processing Standards Publication* 197. AES menjelaskan mengenai algoritma kriptografi yang digunakan untuk melindungi data sebagai pengganti DES. Algoritma AES adalah sebuah *symmetric block cipher* yang dapat melakukan enkripsi dan dekripsi pada data. Algoritma AES dapat menggunakan kunci 128, 192 dan 256 *bit* untuk melakukan enkripsi dan dekripsi terhadap data dengan ukuran blok 128 *bit*.

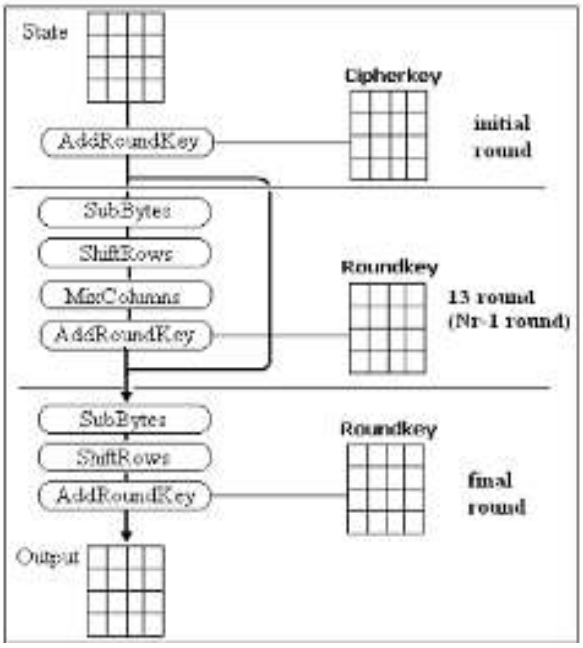
Metode AES256 merupakan salah satu metode kriptografi simetris yang mampu untuk mengenkripsi pesan sepanjang 16 karakter dengan menggunakan kunci sepanjang 32 karakter. Untuk meningkatkan keamanannya, maka kunci yang digunakan dapat dimasukkan ke dalam fungsi hash SHA1 sehingga nilai hash yang diperoleh yang digunakan sebagai kunci.

The Advanced Encryption Standard (AES) dengan metode Rijndael ini diperkenalkan oleh 2 orang kriptografer asal Belgia yaitu Joan Daemen dan Vincent Rijmen. Karena menggunakan kunci dengan ukuran 128, 192 atau 256 *bit*

maka algoritma ini sering disebut dengan “AES-128”, “AES-192”, atau”AES-256” sesuai dengan kunci yang dipakai, sedangkan ukuran blok yang digunakan adalah 128 *bit*.

II.6.2. Proses *Enkripsi* AES (*Edvanced Enryption Standard*)

Proses enkripsi algoritma AES terdiri dari 4 jenis transformasi *bytes*, yaitu *SubBytes*, *ShiftRows*, *Mixcolumns*, dan *AddRoundKey*. Pada awal proses enkripsi, *input* yang telah di *copy*kan ke dalam *State* akan mengalami transformasi *byte AddRoundKey*. Setelah itu, *State* akan mengalami transformasi, *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* secara berulang-ulang sebanyak Nr. Proses ini dalam algoritma AES sebagai *round function*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, state tidak mengalami transformasi *MixColumns*. Ilustrasi proses enkripsi AES dapat digambarkan seperti pada Gambar II.3 di bawah ini (Voni Yuniati,dkk ;2014:24).



Gambar II.3 Ilustrasi Proses Enkripsi AES
(Sumber : Voni Yuniati,dkk ;2014:24)

Pada proses enkripsi, algoritma AES menggunakan empat transformasi yang berbeda yaitu:

1. SubBytes()

Transformasi *SubBytes()* merupakan substitusi *byte* yang beroperasi terhadap setiap *byte* pada *State* dengan menggunakan tabel substitusi (*S-box*). Sebagai contoh, jika s 1,1 = {53} maka nilai substitusi diperoleh dari perpotongan antara baris "5" dengan kolom "3" pada *S-box* sehingga didapat hasilnya {ed}.

2. ShiftRows()

Transformasi *ShiftRows()* dilakukan pada 3 baris terakhir dengan melakukan geser (*shift*) memutar dengan nilai shift yang berbeda–beda tergantung kepada barisnya. Baris pertama (r = 0) tidak dilakukan operasi *ShiftRows()*. Secara spesifik, proses transformasi *ShiftRows()* adalah sebagai berikut:

s‟r,c= sr, (c + shift(r,Nb)) mod Nb ; untuk 0 < r < 4 dan 0 ≤ c < Nb

dimana nilai shift(r,Nb) tergantung pada nomor baris, untuk Nb=4 maka : shift (1,4) = 1; shift(2,4) = 2; shift(3,4) = 3

3. MixColumns()

Transformasi *MixColumns()* dioperasikan pada *State* secara kolom per kolom, masing–masing kolom dikalikan dengan matrik yang sudah ditentukan

4. AddRoundKey()

Pada transformasi *AddRoundKey()*, sebuah subkunci ditambahkan pada *State* dengan operasi XOR. Setiap subkunci terdiri dari Nb *word* dari himpunan subkunci. Subkunci ditambahkan dengan *State* dengan cara sebagai berikut:

[s‟0,c, s‟1,c, s‟2,c, s‟3,c] = [s0,c, s1,c, s2,c, s3,c] ⊕ [w_{round}*Nb+c] ;untuk 0 c<Nb

[wi] adalah *word* dari *key* yang bersesuaian dimana i = round*Nb+c. Transformasi *AddRoundKey* pada proses enkripsi pertama kali pada *round* = 0 untuk *round* selanjutnya *round* = *round* + 1, pada proses dekripsi pertama kali pada *round* = 14 untuk *round* selanjutnya *round* = *round* – 1 (Voni Yuniati,dkk ;2014:24).

Pada permulaan enkripsi, *input* yang berupa *plaintext* dimasukkan ke dalam *State*, pada initial *round* dilakukan transformasi AddRoundKey(State, SubKunci(0)), setelah initial *round* proses menuju pada *round function* sebanyak Nr-1 putaran ($1 \leq round < Nr$), dimana di dalam *round function* ini dilakukan transformasi berturut – turut yaitu *SubBytes()*, *ShiftRows()*, *MixColumns()*, dan *AddRoundKey()*. Setelah itu proses akan menuju pada putaran terakhir (*final round*) dimana pada putaran terakhir ini dilakukan transformasi *SubBytes()*, *ShiftRows()* dan *AddRoundKey()*, pada putaran terakhir ini setelah transformasi *AddRoundKey()* maka akan menghasilkan *final State* yang merupakan *output* yang disebut *ciphertext.* (Voni Yuniati,dkk ;2014:25).

II.7. Android

Perangkat berbasis android hanya mempunyai satu layar *foreground*. Normalnya saat menghidupkan android, yang pertama dilihat adalah *home*. Kemudian bila menjalankan sebuah aplikasi catur, *User Interface*nya (UI) akan menumpuk diatas layar sebelumnya (*home*). Kemudian bila melihat *help*-nya catur, maka UI *help* akan menimpa UI sebelumnya (catur), begitu seterusnya. Semua proses diatas direkam di *application stack* oleh sistem *activity manager*.

Menekan tombol *back* hanya kembali ke halaman sebelumnya, analoginya mirip dengan *browser* dimana ketika Kamu meng-klik tombol *back browser* akan kembali menampilkan halaman sebelumnya. (Arief Akbarul Huda; 2014:9).

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, *Google Inc*. Membeli *Android Inc*., Pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google*, HTC, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile*, dan *Nvidia*. (master.com; 2014:5)

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, *Google* merilis kode-kode Android di bawah *lisensi Apache*, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung *Google* atau dikenal sebagai *Open Handset Distribution* (OHD).

Pada Juli 2000, *Google* bekerjasama dengan *Android Inc*, perusahaan yang berada di *Palo Alto California* Amerika Serikat. Para pendiri *Android Inc*, bekerja pada *Google*, di antaranya Andy Rubin Rich Miner, Nick Sears, dan Chris White.

Saat itu banyak yang menganggap fungsi *Android Inc*, hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa *Google* hendak memasuki pasar telepon seluler. Di perusahaan *Google*, tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh *kernel* Linux. Hal ini menunjukkan indikasi bahwa *Google* sedang bersiap menghadapi persaingan dalam pasar telepon seluler.

Telepon pertama yang memakai sistem operasi Android adalah HTC *Dream*, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis telepon seluler yang menggunakan Android.

1. Android versi 1.1

   Pada 9 Maret 2009, *Google* merilis Android versi 1.1 Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, search (pencarian suara), pengiriman pesan dengan *gmail*, dan pemberitahuan *email*.

2. Android versi 1.5 (*Cupcake*)

   Pada pertengahan Mei 2009, *Google* kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke *Youtube* dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP, kemampuan terhubung secara otomatis ke *headset Bluetooth*, animasi layar dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

3. Android versi 1.6 (*Donut*)

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, *camcorder*, dan galeri yang di integrasikan CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text to speech engine*, kemampuan dial kontak, teknologi *text to change speech* (tidak tersedia pada semua ponsel, pengadaan resolusi VWGA).

4. Android versi 2.0/2.1 (*Eclair*)

   Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan *Google Maps* 3.12, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, digital *zoom*, dan *Bluetooth* 2.1.

   Untuk bergerak cepat dalam persaingan perangkat generasi berikut, *Google* melakukan investasi dengan mengadakan kompetisi aplikasi *mobile* terbaik (*killer apps* – aplikasi unggulan). Kompetisi ini berhadiah $25.000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik.

   Dengan semakin berkembangnya dan semakin bertambahnya jumlah *headset* Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah

ke dalam sistem operasi Android adalah *Shazam*, *Backgrounds*, dan *WeatherBug*. Sistem operasi Android dalam situs *Internet* juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh *MySpace* dan *Facebook*.

5. Andorid versi 2.2 (*Froyo: Frozen Yoghurt*)

   Pada 20 Mei 2010, Android versi 2.2 (*Froyo*) dilucurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungn *Adobe Flash* 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 *Javascript engine* yang dipakai *Google Chrome* yang mempercepat kemampuan rendering pada *browser*, pemasangan aplikasi dalam *SD Card*, kemampuan WiFi/*Hotspot portabel*, dan kemampuan *auto update* dalam aplikasi Android Market.

6. Android versi 2.3 (*Gingerbread*)

   Pada 6 Desember 2010, Android versi 2.3 (*Gingerbread*) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antar muka (*user interface*) didesain ulang, dukugan *format video* VP8 dan WebM, efek *audio* baru (*reverb*, *equalization*, *headphone*, *virtualization*, dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC) dan dukungan jumlah kamera yang lebih dari satu.

7. Android versi 3.0/3.1 (*Honeycomb*)

   Android *honeycomb* dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb*

juga berbeda karena sudah didesain untuk *tablet*. *Honeycomb* juga mendukung multi prosesor dan juga ekselerasi perangkat keras (*hardware*) untu grafis. *Tablet* pertama yang dibuat dengan menjalan *Honeycomb* adalah *MotorolaXoom*. Perangkat tablet dengan platform Android 3.0 akan segera hadir di Indonesia. Perangkat tersebut bernama *Eee Pad Transformer* produksi dari Asus. Rencana masuk pasar Indonesia pada Mei 2011.

8. Android versi 4.0 ICS (*Ice Cream Sandwich*)

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kotak jaringan sosial, perangkat tambahan fotografi mencari email secara *offline*dengan menggunakan NFC.

9. Android 5.0 Lollipop (2014)

   Pembaruan yang mencolok pada Lollipop tampak dari sisi desainnya yang diperhalus dan disesuaikan dengan zaman. Selain itu, fitur-fitur yang sudah hadir pada Android sebelumnya ditingkatkan. Inovasi kurang terasa pada versi ini. Satu-satunya yang lumayan baru adalah dukungan untuk gambar berformat RAW. Format itu memungkinkan para ilustrator, fotografer, atau *graphic designer* menyimpan *file* dengan ukuran besar agar bisa diedit tanpa mengurangi kualitas.

10. Android 6.0 Marshmallow (2015)

    Menu aplikasi pada Android Marshmallow benar-benar dibuat baru. Desainnya membuat pengguna merasa naik kelas dari versi sebelumnya karena lebih

dinamis. Selain itu, ada juga fitur *memory manager* yang memungkinkan pengguna mengecek penggunaan memori pada tiap aplikasi. Rentan waktu pengecekannya bisa disetel dari tiga jam yang lalu hingga 24 jam sebelumnya. Pembaruan kedua ditilik dari pengaturan *volume*. Pada Marshmallow, pengguna bisa mengontrol *volume* yang berbeda-beda pada panggilan, media, dan alarm.

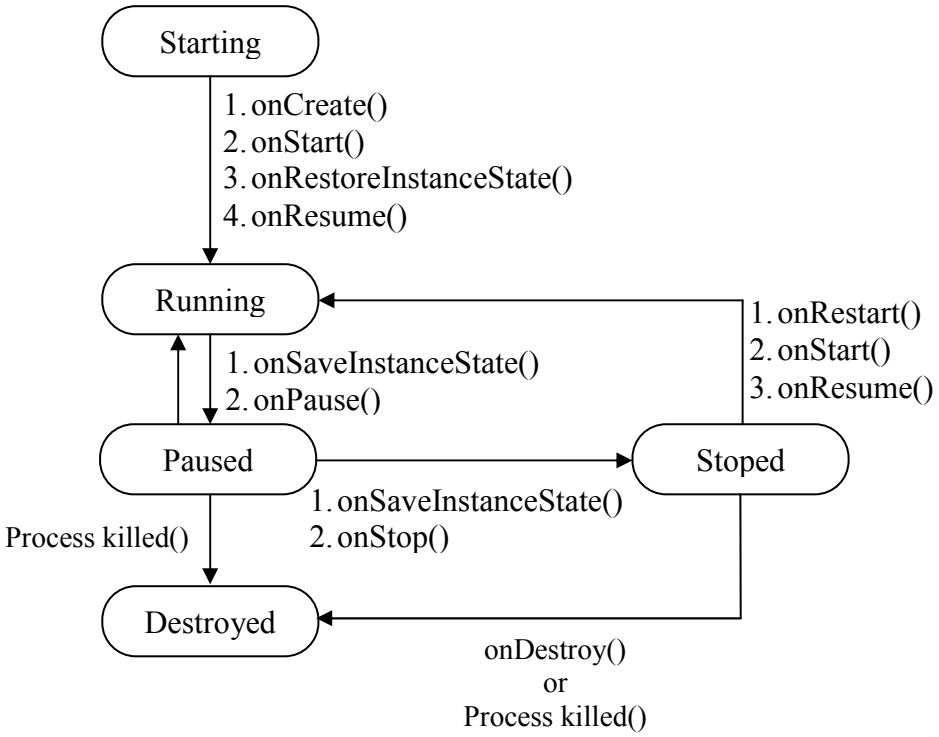
11. Android 7.0 Nougat (2016)

Nougat adalah versi Android termutakhir yang baru diperkenalkan pada ajang kumpul *developer Google I/O*, pertengahan 2016 ini. Beberapa lama setelahnya, *Google* menghadirkan Nougat secara resmi untuk publik. Pembaruan paling mendasar pada versi Nougat adalah kehadiran *Google Assistant* yang menggantikan *Google Now*. Asisten digital tersebut lebih bisa diandalkan untuk menjalankan pelbagai fungsi. Fitur-fitur baru lainnya mencakup layar *split-screen* saat dipakai *multitasking*, serta fitur Doze yang telah dikenalkan di versi *Android Marshmallow* namun telah ditingkatkan. *Android Nougat* juga memiliki dukungan terhadap platform *virtual reality* terbaru *Google*.

II.7.1. Konsep *Android*

Perangkat berbasis *android* hanya mempunyai satu layar *foreground*. Normalnya saat menghidupkan *android*, yang pertama dilihat adalah *home*. Kemudian bila menjalankan sebuah aplikasi catur, *User* Interfacenya (UI) akan menumpuk diatas layar sebelumnya (*home*). Kemudian bila melihat *help* catur, maka *UI help* akan menimpa UI sebelumnya (catur), begitu seterusnya

Semua proses diatas direkam di *application stack* oleh sistem *Activity manager*. Menekan tombol *back* hanya kembali ke halaman sebelumnya, analoginya mirip dengan *browser* dimana ketika Kamu meng-klik tombol *back browser* akan kembali menampilkan halaman sebelumnya. Setiap *User Interface* diwakili oleh kelas *Activity* (*Activity class*).



Gambar II.4. Siklus Activity
(*Sumber: Arif Akbarul Huda, 2013: 10*)

Selama siklus ini berjalan, *activity* bisa mempunyai lebih dari 2 status seperti yang terlihat pada gambar II.4. Yang tidak bisa mengontrol setiap status karena semuanya sudah ditangani oleh sistem. Namun akan mendapat pesan saat terjadi perubahan status melalui method onXX(). Berikut penjelasan setiap status.

Tabel II.1. Siklus *Activity*

onCreate (*Bundle*)	Dipanggil saat pertama kali aplikasi dijalankan. Agar dapat menggunakan ini untuk deklarasi variable atau membuat *userinterface*.
onStart()	Mengindikasikan *activity* yang ditampilkan ke pengguna (*user*).
onResume()	Dipanggil saat aplikasi dimulai berinteraksi dengan pengguna. Disini sangat cocok untuk meletakkan animasi ataupun musik.
onPause()	Dipanggil saat applikasi yang dijalankan kembali ke halaman sebelumnya atau biasanya karena ada *activity* baru yang dijalankan. Disini cocok untuk meletakkan algoritma penyimpanan (*save*).
onStop()	Dipanggil saat aplikasi berjalan di belakang layar dalam waktu cukup lama.
onRestart()	Activity kembali menampilkan *userinterface* setelah status *stop*.
onDestroy()	Dipanggil saat aplikasi benar-benar berhenti.
onSaveInstanceState (*Bundle*)	*Method* ini mengijinkan *activity* untuk menyimpan setiap status *instance*. Misalnya dalam mengedit teks, kursor bergerak dari kiri ke kanan.
onRestoreInstanceState (*Bundle*)	Dipanggil saat *activity* kembali menginstalisasi dari status sebelumnya yang disimpan oleh onSaveInstanceState (*Bundle*).

Kelebihan *Android*:

1. *Open Source* Bagi para programming mania bisa dengan mudah membuat aplikasi berbasis *Android*. FYI Aplikasi *Android* bisa dibuat dengan *Framework* yang *Free* juga, selain itu kalau yang mengerti OS dapat membuat aplikasi baru atau memperbaikinya.

2. *Multitasking*, ponsel *Android* bisa menjalankan berbagai aplikasi, itu artinya bisa *browsing* dan mendengarkan lagu dalam waktu yang bersamaan.
3. Notifikasi Baik itu dari SMS, *Twitter*, *Facebook*, *Email*, semuanya ernotifikasi di layar utama, jadi semua notifikasi mirip ada pesan masuk. Semua pemberitahuan akan kamu ketahui dalam 1 layar.
4. Pada layar utama *Android* terdapat *widget* yang memudahkan untuk mengakses aplikasi *Android*, bukan cuma berupa *shortcut* tetapi beberapa *widget* bisa melakukan sesuatu yang keren seperti menampilkan status terbaru di FB dan *Twitter*, atau melihat cuaca.
5. Android *Market* Terdapat puluhan ribu aplikasi yang bisa di *dowload* baik itu bebas biaya atau berbayar.

Kelemahan *Android*

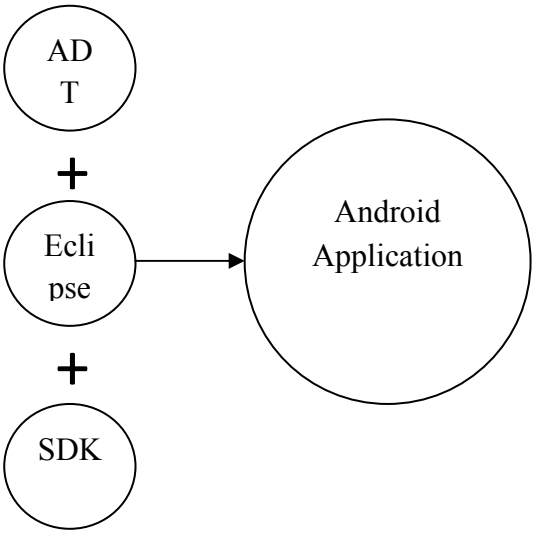
1. Koneksi *Internet* Sebenarnya koneksi *internet* bisa dimatikan namun jika dimatikan itu artinya kita tidak bisa menikmati fasilitas *Android* seperti notifikasi sosial media maupun *email*.
2. Di setiap aplikasi hampir selalu muncul iklan, kecuali aplikasi bawaan Androidnya sendiri seperti *facebook* dan *twitter*.
3. Mahal Untuk *Android* yang berseri *Froyo*, harganya memang mahal. Jadi OS *Android* mahal? bukan itu penyebabnya namun OS *Android* membutuhkan perangkat yang mampu untuk menjalankannya sehingga mau tidak mau Hardwarenya harus yang bagus.

II.7.2. Instalasi *Eclipse*

Untuk memiliki aplikasi *eclipse* harus memiliki 3 buah *file* dibawah ini, bisa mengunduhnya langsung dari *web* resminya di www.eclipse.org (*download* saja versi terbarunya)

1. *Eclipse*
2. ADT 16.0.1 *Plugin*
3. Android-SDK_r16

Eclipse merupakan sebuah editor, secara default editor ini belum bisa dipakai untuk mendevelop *android*. Agar bisa digunakan untuk membuat aplikasi android maka harus diinstall *plugin* dulu namanya ADT (*Android Development Tools*). Setelah *terinstal*, maka *eclipse* sudah siap digunakan, hanya saja belum dilengkapi *library*dan emulator.. Untuk memenuhinya, maka perlu diinstall SDK Android (*StKamurt Development Kit*).



Gambar II.5. Penggunaan Eclipse
(*Sumber: Arif Akbarul Huda, 2013: 15*)

Jadi intinya, *eclipse* bisa digunakan setelah selesai mengkonfigurasi antara editor *Eclipse*, ADT dan SDK. Selanjutnya ikuti langkah demi langkah berikut.

Menginstal Plugin ADT sebagai berikut :

1. Jalankan *eclipse*, kemudian pilih *help > install new software*
2. Klik Add di sebelah kanan atas.
3. Pada kotak dialog *add repository*, isikan nama ADT dan lokasi http://dl-ssl.google.com/android/eclipse/
4. Lihat kotak *available software*. Centang pada item *Developer Tools* kemudian pilih *next*.
5. Selanjutnya pilih *next*.
6. Kemudian pilih *license agreement*, pilih *accept* kemudian klik *finish*
7. Setelah proses instalasi selesai, *restart eclipse*

Konfigurasi SDK sebagai berikut :

1. Jalankan *eclipse*, pilih *window > preference*
2. Pilih Android
3. Pada SDK lokasi, klik *browse* cari kemudian SDK *File* yang sudah ada.
4. Klik *Apply*
5. Klik *ok*.

II.7.3. Sejarah *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*:

1. Multi platform

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

2. *Mulit language*

   Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lain seperti C/C++, *CobolPython*, *Perl*, PHP, dan lain sebagainya.

3. *Multi role*

   Selain sebagai IDE untuk pengembangan aplikasi. *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak seperti dokumentasi, pengujian perangkat lunak, pengembangan web, dan lain sebagainya.

Pada saatini, *Eclipse* merupakan salah satu IDE favorit karena gratis dan *open source*. *Open source* berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan membuat komponen yang disebut *plug-in*.

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak pengembangan IBM *Visual Age for Java* 4.0. Produk *Eclipse* ini diluncurkan oleh IBM pada tanggal 5 November 2001. IBM menginvestasikan US$ 40 juta untuk pengembangannya. Sejak 5 November 2001, konsorsium *Eclipse Foundation* mengambil alih pengembangan *Eclipse* lebih lanjut. (Wina Noviani Fatimah; 2011: 2)

II.8. *Unified Modeling Languange* (UML)

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara *visual.* Juga merupakansatu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek.

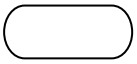
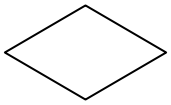
Sejarah UML terbagi dalam dua fase sebelum dan sesudah munculnya UML. Dalam *fase* sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan desain berbeda-beda, sehingga dapat dikatakan belum memiliki standarisasi.

Sebagian besar para perancang sistem informasi dalam menggambarkan informasi dengan memanfaatkan UML diagram dengan tujuan utama untuk membantu tim proyek berkomunikasi, mengeksplorasipotensi desain,dan memvalidasi desain arsitektur perangkat lunak atau pembuat program. Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik.

II.8.1. *Activity* diagram

Menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas. Peralatan yang digunakan untuk membuat struktur diagram *activity* dapat dilihat pada Tabel II.4.

Tabel II.2. Activity Diagram

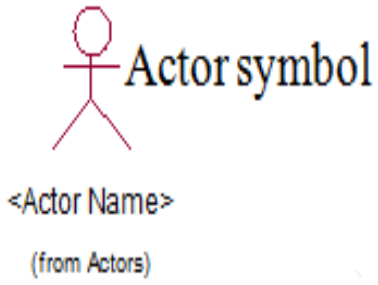
	*Activity*
	*Transition*
	*Decison*
	*SynchronizationBars*

(Sumber : Haviluddin; 2013 : 4)

UML memiliki seperangkat notasi yang akan digunakan kedalam tiga kategori diatas yaitu struktur diagram, *behavior* diagram dan *interaction* diagram. Berikut beberapa notasi dalam UML diantaranya:

1. *Actor*

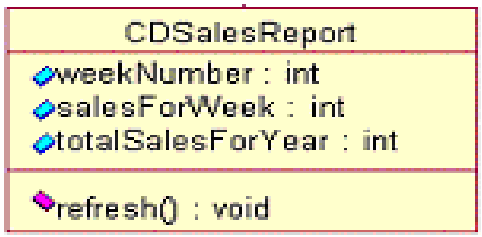
Actor menentukan peran yang dimainkan oleh *user* atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas *actor* adalah memberikan informasi kepada system dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.



Gambar II.8. Notasi *actor*
(*Sumber : Haviluddin; 2013 : 6*)

2. *Class Diagram*

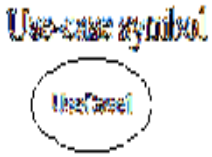
Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari system berorientasi objek



Gambar II.9. Notasi *class*
(*Sumber : Haviluddin; 2013 : 6*)

3. *UseCase* dan *UseCase Specification*

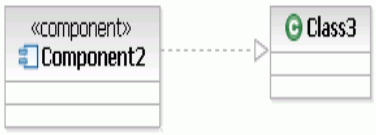
Usecase adalah deskripsi fungsi dari sebuah sistem perspektif pengguna.*Usecase* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan system disebut skenario. *Usecase* merupakan awal yang sangat baik untuk setiap *fase* pengembangan berbasis objek, *design, testing*, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang diluar sistem. Perlu diingat bahwa *usecase* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan non-fungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya.



Gambar II.10. Notasi*use case*
(Sumber : Haviluddin; 2013 : 6)

4. *Realization*

Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.



Gambar II.11. Notasi *realization*
(*Sumber : Haviluddin; 2013 : 6*)

5. *Interaction*

*Interaction*d igunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.

Gambar II.12. Notasi *Interaction*
(*Sumber : Haviluddin; 2013 : 6*)