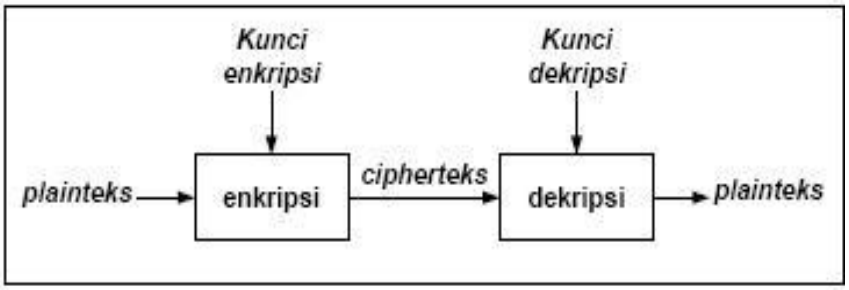
BAB 2
TINJAUAN PUSTAKA

2.1 Kriptografi

2.1.1 Pengertian Kriptografi

Kriptografi (cryptography) berasal dari bahasa yunani yaitu *cryptos* artinya rahasia (secret) dan *graphein* artinya tulisan (writing). Jadi kriptografi berarti tulisan rahasia (secret writing). Secara istilah kriptografi didefinisikan sebagai ilmu sekaligus seni untuk menjaga kerahasian pesan (data atau informasi) yang mempunyai arti atau nilai, dengan cara menyamarkannya (mengacak) menjadi bentuk yang tidak dapat dimengerti menggunakan suatu algoritma tertentu.

Suatu pesan yang akan dirahasisakan atau disandikan dalam ilmu kriptografi disebut *plaintext*, sedangkan pesan yang telah disandikan sehingga tidak memiliki nilai dan arti lagi dengan tujuan agar pesan tidak dapat dibaca oleh pihak yang tidak berhak disebut *chipertext*. Dalam ilmu kriptografi juga terdapat istilah enkripsi dan dekripsi. Enkripsi merupakan proses menyandikan *plaintext* menjadi *chipertext* dengan menggunakan algoritma tertentu. Sedangkan proses mengembalikan *chipertext* menjadi *plaintext* disebut dekripsi. Berikut ini skema umum sebuah penyandian dalam kriptografi [1] :



Gambar 2.1. Sistem Kriptografi

2.1.2 Algoritma Kriptografi

Perkembangan algoritma kriptografi dapat kita bagi menjadi dua, yaitu :

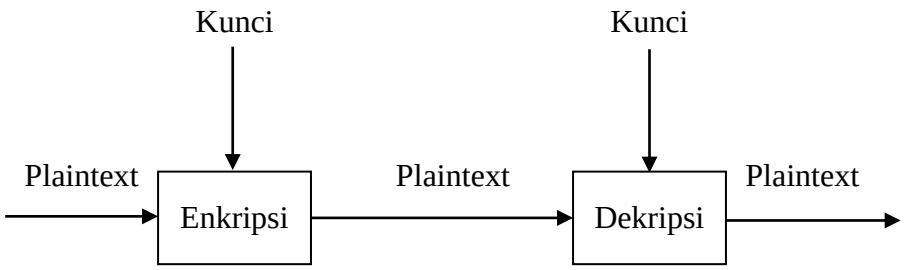
1. Kriptografi klasik

   Pada algoritma klasik, diterapkan teknik enkripsi konvesional (simetris). Algoritma ini merupakan algoritma kriptografi yang biasa digunakn orang sejak berabad-abad yang lalu.
2. Kriptografi Modern

   Kriptografi modern lebih menitikberatkan pada kerahasian kunci yang digunakan pada algoritma tersebut (oleh pemakainya) sehingga algoritma tersebut dapat saja disebarkan tanpa takut kehilangan kerahasian bagi para pemakainya.

Secara umum berdasarkan kesamaan kuncinya, algoritma sandi dibedakan menjadi:

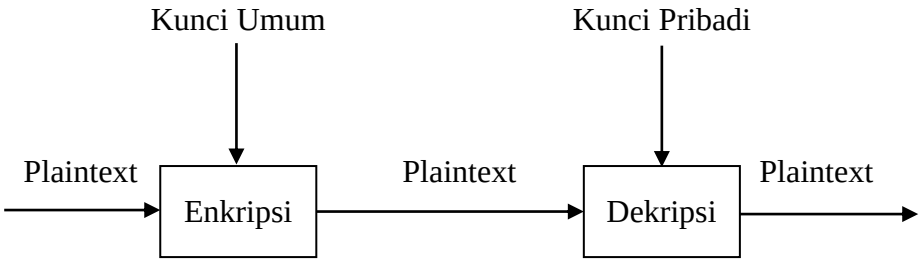
1. Algoritma Simetrik (*symmetric algorithm*) adalah suatu algoritmayang menggunakan kunci enkripsi sama dengan kunci dekripsi sehingga algoritma ini disebut juga sebagai *single-key algorithm*. Beberapa algoritma kriptografi simetrik antara lain DES, Blowfish, IDEA, RC4, RC5, RC6,AES atau Rijandel, Serpent dan Twofish. Proses enkripsi – dekripsi algoritma kriptografi simetris dapat dilihat pada gambar dibawah ini :



Gambar 2.2. Algoritma Simetris

Algoritma kriptografi simetris dibagi menjadi 2 kategori yaitu algoritma algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi satu bit atau satu byte data. Sedang pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit atau byte data (per blok).

2. Algoritma Asimetrik (*asymmetric algorithm*) adalah suatu algoritma yang menggunakan kunci enkripsi tidak sama dengan kunci dekripsi. Algoritma ini menggunakan dua kunci yakni kunci public (*public key*) dan kunci privat (*private key*). Beberapa algoritma kunci public antara lain adalah RSA, Rabin dan ElGamal. Proses enkripsi-dekripsi algoritma asimetris dapat dilihat pada gambar dibawah ini :



Gambar 2.3. Algoritma Asimetris

Pada algoritma public key ini, semua orang dapat mengenkripsi data dengan memakai public key penerima yang telah diketahui secara umum. Akan tetapi data yang telah terenkripsi tersebut hanya dapat didekripsi dengan menggunakan private key yang hanya diketahui oleh penerima[2].

2.2 Tinjauan matematis

2.2.1 Konsep Dasar

Dasar matematis yang mendasri proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu himpunan berisi elemen plaintext dan himpunan berisi elemen ciphertext. Enkripsi dan dekripsi merupakan fungsi tranformasi antara dua himpunan tersebut. Bila himpunan plaintext dinotasikan dengan P dan himpunan ciphertext dinotasikan dengan C, sedang fungsi enkripsi dinotasikan dengan E dan fungsi dekripsi dengan D maka proses enkripsi-dekripsi dapat dinyatakan dalam notasi matematis denan :

$$E(P) = C \text{ dan}$$

$$D(C) = P$$

Karena proses enkripsi-dekripsi bertujuan memperoleh kembali data asal, maka :

$$D(E(P)) = P$$

Relasi antara himpunan plaintext dengan himpunan cihpertext harus merupakan fungsi korespondensi satu-satu (*one to one relation*). Hal ini merupakan keharusan untuk mencegah terjadinya ambigu dalam dekripsi yaitu satu elemen ciphertext menyatakan lebih dari satu elemen plaintext.

Pada metode kriptografi simetris atau konvensional digunakan satu buah kunci. Bila kunci dinotasikan denan ‘K’ maka proses enkripsi-dekripsi metode kriptografi simeteris dapat dinotasikan dengan :

$$E_k(P) = C \text{ dan}$$

$$D_k (C) = P$$

Dan keseluruhan sistem dinyatakan sebagai :

$$D_k(E_k(P))=P$$

Pada metode kriptografi nirsimeteris digunakan kunci umum untuk enkripsi dan kunci pribadi untuk dekripsi. Bila kunci umum dinotasikan dengan ‘PK’ dan kunci pribadi dinotasikan dengan ‘SK’ maka

proses enkripsi-dekripsi metode kriptografi nirsimeteris dapat dinotasikan dengan :

E_{PK} (P) = C dan

D_{SK} (C) =P

Dan keseluruhan sistem dinyakatan sebagai

$D_{SK}(E_{PK}(P)) = P$

2.3 RC6 Block Chiper

RC6 merupakan algoritma yang merupakan keturunan dari RC5 yang juga merupakan kandidat AES (Advanced Encryption Standard). Pada mulanya, perancangan RC6 diawali ketika RC5 dianggap dapat dijadikan kandidat untuk mengikuti kompetisi pemilihan AES. Modifikasi kemudian dibuat untuk meningkatkan keamnan dan performa dan juga untuk dapat memenuhi persyaratan AES

RC6 dirancang untuk menghilangkan segala ketidakamanan yang ditemukan pada RC5, karena analisis pada RC5 menunjukan bahwa ternyata jumlah rotasi yang terjadi pada RC5 tidak sepenuhnya bergantung pada data yang terdapat dalam blok. Selain itu serangan kritanalisis diferensial juga ternyata dapat menembus keamanan yang ditawarkan RC5.

RC6 juga dirancang untuk memenuhi persyaratan AES yang diantaranya adalah kemampuan untuk beroperasi pada mode blok 128 bit. Jika besar blok 128 bit langsung dipaksakan untuk diimplemetasikan dengan algoritma RC5, maka akan dibutuhkan register kerja 64 bit. Spesifikasi arsitektur dan bahasa yang menjadi tempat implementasi algoritma yang ditentukan oleh AES belum mendukung pengoperasian 64 bit yang efisien. Oleh karena itu, daripada menggunakan 2 register 64 bit seperti pada RC5, RC6 menggunakan 4 register 32 bit. Karena menggunakan 4 register makan akan terdapat 2 operasi rotasi pada

setiap *half-round yang ada*, dan juga akan lebih banyak bit-bit yang akan digunakan untuk mempengaruhi banyaknya bit yang dirotasi.

RC6 seperti juga RC5 mengeksploitasi penggunaan operasi-operasi primitive yang diimplementasikan secara efisien dalam prosesor-prosesor modern. RC6 juga selain menggunakan ketiga operasi primitive yang digunakan dalam RC5, juga menggunakan operasi perkalian 32 bit yang telah diimplementasikan secara efisien dalam prosesor modern saat ini. Primitive operasi perkalian ini sangat efektif dalam menghasilkan efek *"diffusion"* atau penyebaran yang tentu saja mengakibatkan RC6 lebih aman daripada RC5. Operasi perkalian ini digunakan untuk menghitung jumlah bit yang dirotasi sehingga konsep *data-dependent rotations* dapat dengan lebih sempurna diimplementasikan [3].

2.3.1 Operasi-operasi Primitif

Selain ketiga operasi primitif yang digunakan pada RC5, RC6 juga menggunakan operasi perkalian modulu-2w. Berikut adalah daftar operator-operator primitive (termasuk invers) yang digunakan dalam RC6:

a+b : penjumlahan integer modulu 2w
a-b : pengurangan integer modulu 2w
a⊕b : operasi eksklusif –OR (xor) dari w-bit *word*
axb : perkalian integer modulu 2W
a<<<b : rotasi a ke kiri sebanyak nilai yang diperoleh dari bit-bit orde bawah sejumlah 2log(w) dari b
a>>>b : rotasi terhadap a ke kanan sebanyak nilai yabg diperoleh dari bit-bit orde bawah sejumlah 2log(w) dari b

2.3.2 Algoritma RC6

Algoritma RC6 seperti juga RC5 merupakan algoritma *chipper* yang terparameterisasi. RC6 secara tepat ditulis sebagai :

RC6 – w /r/b

Nilai parameter w, r,dan b menyatakan hal yang sama seperti yang ditunjukan dalam algoritma RC5. Algoritma RC6 yang dipakai sebagai kandidat AES adalah RC6-32/20/b, yang berarti ukuran *word* 32 bit, jumlah ronde 20 kali, dengan panjang kunci b ditentukan pengguna.

2.3.3 Key Expansion Algoritma

Algoritma untuk membangkitkan kunci internal sama seperti pada RC5. Nilai konstanta Pw dan Qw yang digunakan juga sama, tetapi ukuran array S tidak sama dengan yang seperti RC5. Ukuran t dari array S dalam RC6 adalah t = 2(r+2), yang berarti terdapat lebih banyak kunci internal yang dibangkitkan daripada jumlah kunci internal RC5. Berikut algoritmanya[3].

S[0] = Pw
fori = 1 **to** (2r + 3) **do**
S[i] = S[i – 1] + Qw
i = 0
j = 0
A = 0
B = 0
for3 × max(c, (2r + 4)) **times do**
S[i] = (S[i] + A + B) <<< 3
A = S[i]
L[i] = (L[j] + A + B) <<< 3
B = L[i]

$$i = (i + 1) \bmod (2r+4)$$

$$j = (j + 1) \bmod c$$

2.3.4 Algoritma Enkripsi

Fungsi enkripsi menerima input 1 blok *plaintext* yang terbagi dalam 4 register yang masing-masing berupa w-bit *word*, yaitu A, B, C dan D. *Chipertext* hasil proses terbagi dan disimpen dalam A, B, C dan D. Dalam proses enkripsi diperlukan table kunci S yang dianggap telah didapat dari proses sebelumnya.

Secara lebih detil, proses enkripsi dengan RC6 dapat dibagi dalam beberapa langkah. Dalam penjelasan berikut, notasi (A,B,C,D) = (B,C,D,A) berarti adalah operasi assignment yang dilakukan parallel (bersamaan) untuk setiap elemen di ruas kanan ke ruas kiri yang berkorespondensi. Langkah-langkahnya adalah sebagai berikut :

1. Mula-mula lakukan *half-round loop* yang seperti pada RC5 :

   Fori = 1 to r do

   A = ((AB)<<<B)+S[i] $\oplus$

   (A,B) = (B,A)
2. Lakukan dua proses RC5 secara parallel, yang satu untuk register A, B dan yang lain untuk register C, D.

   fori = 1 **to** r **do**

   A = ((AB) <<< B) + S[2i] $\oplus$

   C = ((CD) <<< D) + S[2i+1] $\oplus$

   (A,B) = (B,A)

   (C,D) = (D,C)
3. Pada tahap pertukaran,daripada menukar A dengan B dan C dengan D, lakukan permutasi antar keempat register (A,B,C,D) = (B,C,D,A), sehingga komputasi AB bercampur dengan komputasi CD.

fori = 1 **to** r **do**

A = ((AB) <<< B) + S[2i] ⊕

C = ((CD) <<< D) + S[2i+1] ⊕

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

4. Campurkan komputasi AB dengan CD lebih jauh, yaitu dengan mempertukarkan kedua nilai yang menyatakan jumlah rotasi pada masing-masing komputasi.

fori = 1 **to** r **do**

A = ((A⊕B) <<< D) + S[2i]

C = ((C⊕D) <<< B) + S[2i+1]

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

5. Daripada mengunakan nilai B dan D secara langsung, RC6 menggunakan hasil transformasi kedua register ini. Hal ini dilakukan untuk tidak mengulangi maslah rotasi seperti pada RC5 dimana tidak seluruh bit dalam data yang berpengaruh dalam rotasi. Oleh karena itu, fungsi transformasi yang dipilih harus dapat memanfaatkan seluruh bit di dalam data untuk mengatur jumlah bit yang dirotasikan. Fungsi yang dipilih adalah $f(x) = x(2x +1) \ (mod \ 2w)$ yang kemudian diikitu dengan rotasi ke kiri sebanyak 5 bit. Transformasi ini terpilih karena fungsi f(x) yang merupakan fungsi satu ke satu memiliki bit-bit orde atas yang menentukan jumlah rotasi yang akan digunakan yang sangat bergantung pada x.

fori = 1 **to** r **do**

p = (B × (2B + 1)) <<< 5

q = (D × (2D + 1)) <<< 5

A = ((A⊕p) <<< q) + S[2i]

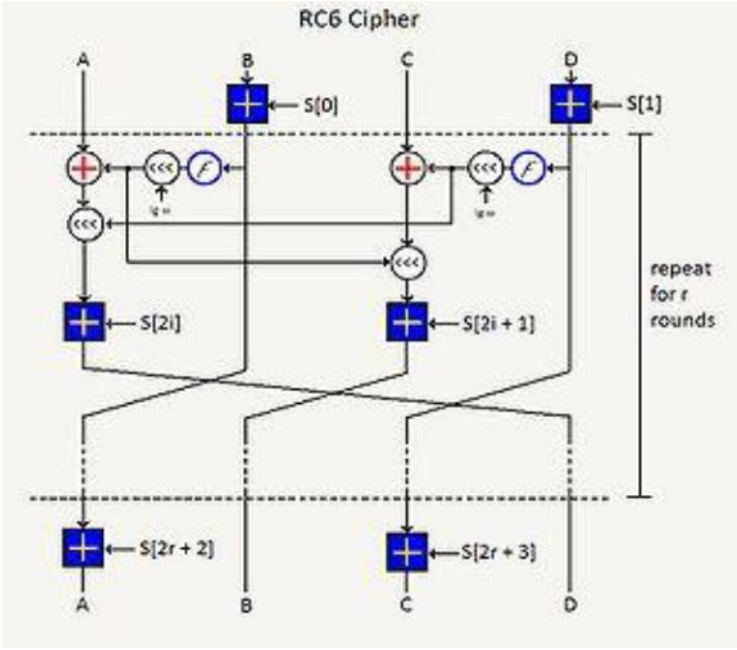
C = ((C⊕q) <<< p) + S[2i+1]

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

6. Setelah *loop* di atas selesai, akan terdapat hasil dimana *plaintext* bias menunjukan bagian input ronde pertama dalam enkripsi dan *chipertext* bias menunjukan bagian input ronde terakhir dalam enkripsi. Oleh karena itu perlu ditambahkan langkah-langkah di awal dan di akhir *loop* untuk menyamarkan hubungan ini. Sehingga, terbentuk algoritma enkripsi RC6 yang sebagai berikut :

   B = B + S[0]
   D = D + S[1]
   fori = 1 **to** r **do**
   p = (B × (2B + 1)) <<< 5
   q = (D × (2D + 1)) <<< 5
   A = ((A⊕p) <<< q) + S[2i]
   C = ((C⊕q) <<< p) + S[2i+1]
   (A,B,C,D) = (B,C,D,A)
   A = A + S[2r + 2]
   C = C + S[2r + 3]

Perlu diketahui juga, dalam varian baru RC6 jumlah rotasi ke kiri yang mengikuti fungsi kuadrat bukan 5 bit tetapi adalah 2log(w) bit[3].



Gambar 2.4. RC6 Block Chiper

2.3.5 Algoritma Dekripsi

Proses dekripsi ciphertext pada algoritma RC6 merupakan pembalikan dari proses enkripsi. Pada proses whitening, bila proses enkripsi menggunakan operasi penjumlahan, maka pada proses dekripsi menggunakan operasi pengurangan. Sub kunci yang digunakan pada proses whiteneing setelah iterasi terakhir diterapkan sebelum iteasi pertama, begitu juga sebaliknya sub kunci yang diterapkan pada proses whitening sebelum iterasi pertama digunakan pada whitening setelah iterasi terkahir. Akibatnya, untuk melakukan dekripsi, hal yang harus dilakukan semata-mata hanyalah menerapkan algoritma yang sama dengan enkripsi, dengan tiap iterasi menggunakan sub kunci yang sama dengan yang digunakan pada saat enkripsi, hanya saja urutan sub kunci yang digunakan terbalik[3].

Sama seperti pada RC5, algoritma dekripsi RC6 jga merupakan penurunan dari algoritma enkripsi. Algoritmanya sebagai berikut :

$$C = C - S[2r + 3]$$
$$A = A - S[2r + 2]$$

fori = r **downto** 1 **do**

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

p = (D × (2D + 1)) <<< 5

q = (B × (2B + 1)) <<< 5

C = ((C – S[2i + 1]) >>> q)p ⊕

A = ((A – S[2i]) >>> p)q ⊕

D = D – S[1]

B = B – S[0]

2.4 Sejarah Android

Menurut Nazaruddin Safaat H, Android adalah sebuah system operasi untuk perangkat mobile berbasis linux yang mencakup system operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel *smartphone*. 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[4].

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler.

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

Sekitar September 2007 Google mengenalkan *Nexus One*, salah satu jenis *smartphone* yang menggunakan android sebagai system operasinya. Telepon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 januari 2010. Pada 9 desember 2008, diumumkan anggota baru yang bergabung dalam program kerja android ARM Holdings. Atheros Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka, Android, perangkat *mobile* yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan *bug* dan penambahan fitur baru.

Pada masa saat ini sebagian besar *vendor-vendor smartphone* sudah memproduksi *smartphone* berbasis android, *vendor-vendor* itu antara lain HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, WayteQ, Sony Ericsson, Acer, Philips, T-Mobile, Nexian, IMO, Asus dan masih banyak lagi *vendor smartphone* di dunia yang memproduksi android. Hal ini, karena android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikan dan dipakai oleh *vendor* manapun.

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi Table PC. Pesatnya pertumbuhan Android selain faktor yang disebutkan di atas adalah karena Android itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, Apalikasi dan *Tool* Pengembangan, Market aplikasi android serta dukungan yang sangat tinggi dari komunitas *Open Source* di dunia, sehingga android terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia [4].

2.4.1 Android SDK (Software Development Kit)

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi-netral, Android memberi Anda kesempatan untuk membuat Aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone/Smartphone*. Beberapa fitur-fitur Android yang paling penting adalah

1. *Framework* aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin *Vistual Dalvik* dioptimalkan untuk perangkat *mobile*.
3. *Integrated* browser berdasarkan *engine open source webkit.*
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasin openGL ES 1.0 (opsional akselerasi *hardware*).
5. SQLite untuk penyimpanan data.
6. *Media Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM *telephony* (tergantung *hardware*).
7. *Bluetooth,* EDGE, 3G, dan WiFi (tergantung *hardware*).
8. Kamera, GPS, kompas, dan *accelerometer* (tergantung *hardware*).
9. Linkungan *Development* yang lengkap dan kaya termasuk perangkat *emulator, tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*

2.4.2 ADT (Android Development Tools)

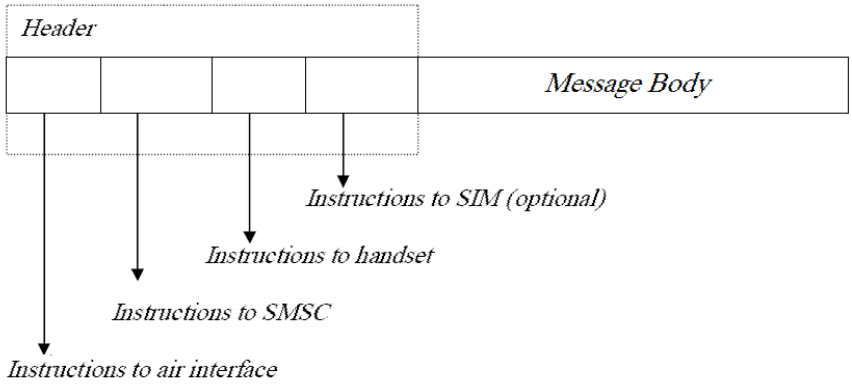
Android Development Tools (ADT) adalah plugin yang didesain untuk IDE *Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE *Eclipse*. Dengan menggunakan ADT untuk *Eclipse* akan memudahkan kita dalam membuat aplikasi *project* anroid, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan Android SDK melalui *eclipse*. Dengan ADT juga kita dapat melakukan pembuatan *package* android (.apk) yang digunakan untuk distribusi aplikasi android yang kita rancang.

2.4.3 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: Multi-platform Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X. Mulit-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. Multi-role: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya. Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan open source, yang 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 komponen yang dinamakan plug-in.

2.5 Struktur Pesan SMS

Sturuktur pesan pada sebuah paket SMS dapat dilihat pada gambar berkut :



Gambar 2.5. Struktur Pesan SMS

Pada Gambar dapat terlihat bahwa pada sebuah paket pesan SMS terdiri dari *header* dan *body*. Header pesan terdiri dari instruksi-instruksi kepada komponen-komponen yang bekerja dalam jaringan SMS. Pada instruksi-instruksi tersebut, terdapat informasi yang diperlukan selama pengiriman pesan seperti informasi validitas pesan, dan informasi-informasi lainnya. Pada bagian *message body*, terdapat isi dari pengirim pesan yang akan dikirim.

Panjang isi pesan pada sebuah paket SMS berukuran maksimal 160 karakter, diman setiap karakter memiliki panjang 7 bit. Beberapa aplikasi standar telepon selular dapat mendukung panjang pesan dengan karakter sepanjang 8 bit (panjang pesan maksimum 140 karakter) dan karakter yang lebih panjang lainnya seperti 16 bit, namun karakter sepanjang 8 bit dan 16 bit ini tidak didukung oleh semua aplikasi standard telepon selular. Pada umumnya karakter sepanjang 8 bit dan 7 bit digunakan untuk menampilkan data seperti gambar dan symbol[6].

2.6 UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP.UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem[7].

2.6.1 Sejarah UML

Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan kelompok/perusahaan lain yang menggunakan metodologi yang berlainan.

2.6.2 Diagram UML

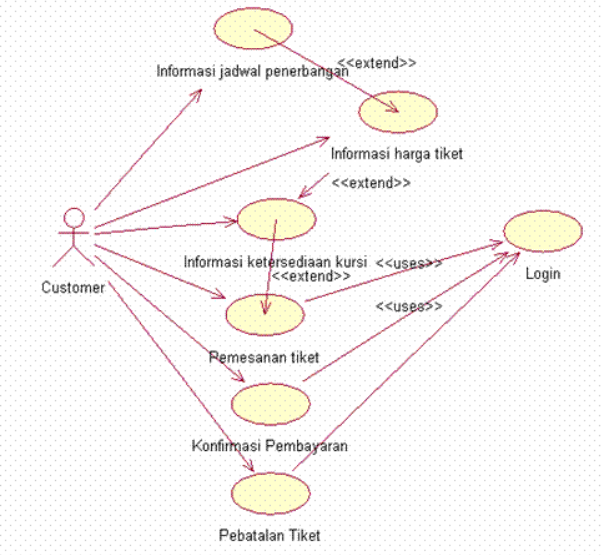
UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

1. Use Case Diagram untuk memodelkan proses bisnis.
2. Conceptual Diagram untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. Secuence Diagram untuk memodelkan pengiriman pesan (message) antar objects.
4. Collaboration Diagram untuk memodelkan interaksi antar objects.
5. State Diagram untuk memodelkan perilaku objects di dalam sistem.
6. Activity Diagram untuk memodelkan perilaku Use Cases dan objects di dalam system.
7. Class Diagram untuk memodelkan struktur kelas.
8. Objects Diagram untuk memodelkan struktur object.
9. Component Diagram untuk memodelkan komponen object.
10. Deployment Diagram untuk memodelkan distribusi aplikasi.

2.6.2.1 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* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Sebuah *use case* dapat meng-*include*

fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dan yang lain. Penggambaran Use case diagram dapat di lihat dalam gambar 2.6 dibawah ini.



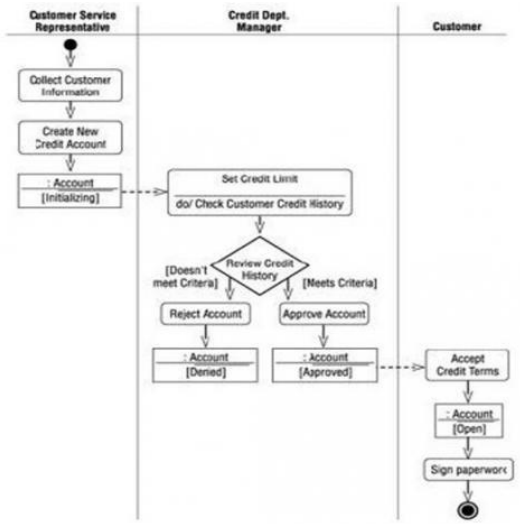
Gambar 2.6. Contoh Use Case Diagram

2.6.2.2 Activity Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak

menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih.

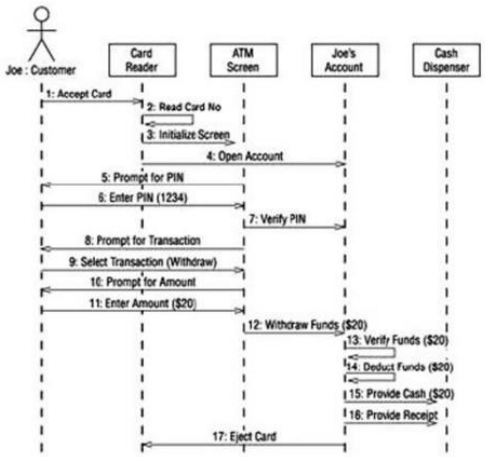
Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu. Contoh Activity diagram dapat di lihat pada gambar 2.7 berikut ini.



Gambar 2.7. Contoh Activity Diagram

2.6.2.3 Sequence Diagram

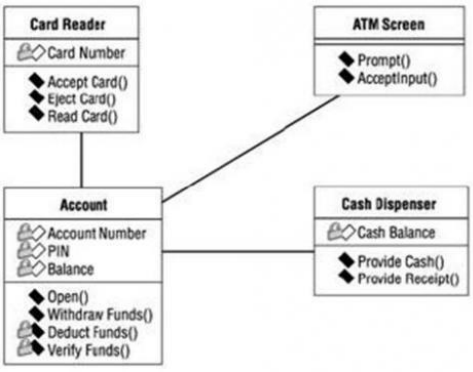
Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case: interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi. Contoh sequence diagram dapat di lihat pada gambar 2.8 berikut ini.



Gambar 2.8. Contoh Sequence Diagram

2.6.2.4 Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. Class diagram menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Penggambaran class diagram dapat di lihat dalam gambar 2.9 dibawah ini.



Gambar 2.9. Contoh Class Diagram

2.7 MySql

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi *user* serta menggunakan perintah standar SQL (*Structured Query Language*) . MySQL merupakan sebuah *database server* yang *free*, artinya kita bebas menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensinya.

MySQL pertama kali dirintis oleh seorang programer database bernama Michael Widenius. Selain MySQL ada beberapa program *database server* lain yang menggunakan standar *query* berupa SQL, antara lain adalah :

1. Oracle
2. PostgreSQL
3. MySQL Front
4. MsQL
5. SQL Server
6. DLL

MySQL adalah sebuah *database server*, dapat juga berperan sebagai *client* sehingga sering disebit *database client/server*, yang pen *source* dengan kemampuan dapat berjalan baik di OS (Operating Sistem) manapun, dengan *Platform* Windows maupun LINUX.

2.8 PhpMyAdmin

PhpMyAdmin merupakan *tools open source* yang dibuat menggunakan program PHP untuk mengakses *database* MySQL via web. Dengan menggunakan program ini kita dapat mengelolah *database* MySQL dengan sangat mudah.

Hampir semua operasi terhadap *database* dan perintah yang ada dapat dijalankan pada program tersebut, termasuk diantaranya menggunakan perintah *Database Definition Language* (DDL), *Database Manipulation Language* (DML) serta*DataControlLanguage*(DCL).