KRIPTOGRAFI MENGGUNAKAN METODE RIJNDAEL UNTUK PENGIRIMAN MMS PADA APLIKASI MOBILE WIRELESS TECHNOLOGY (J2ME)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Disusun Oleh:

ALAN NUR

NIM. 0610633003 - 63

KEMENTERIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2011

LEMBAR PERSETUJUAN

KRIPTOGRAFI MENGGUNAKAN METODE RIJNDAEL UNTUK PENGIRIMAN MMS PADA APLIKASI MOBILE WIRELESS TECHNOLOGY (J2ME)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Teknik

Disusun oleh:

ALAN NUR
NIM. 0610633003 - 63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Muhammad Aswin, MT.	**Adharul Muttaqin, ST., MT.**
NIP. 19640626 199002 1 001	**NIP. 19760121 200501 1 001**

PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi Allah SWT karena atas Taufik, Hidayah, Ridho, Inayah, Mahabbah serta KemanjaanNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "**Kriptografi Menggunakan Metode Rijndael Untuk Pengiriman MMS Pada Aplikasi *Mobile Wireless Technology***". Shalawat dan salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan baik bantuan moril maupun materiil selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Kedua Orang Tua penulis (Imam Safii dan Muslikah Azzahra) dan seluruh keluarga yang senantiasa tiada henti-hentinya memberikan do'a dan restu demi terselesaikannya skripsi ini.
2. Bapak Sholeh Hadi Pramono, DR., Ir., MS. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak M. Azis Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
5. Bapak Waru Djuriatno, ST. MT. sebagai Ketua Kelompok Dosen Keahlian Teknik Informatika dan Komputer Jurusan Teknik Elektro Universitas Brawijaya.
6. Bapak Muhammad Aswin, Ir., MT. selaku dosen pembimbing I yang telah bersedia memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.

7. Bapak Adharul Muttaqin ST., MT. selaku dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.
8. Ibu Ir. Retnowati selaku dosen pendamping akademik.
9. Bapak dan Ibu Dosen, Staff Administrasi Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
10. Semua Asisten, Ka. Lab serta Laboran dari Laboratorium Teknik Informatika dan Komputer yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan tugas akhir ini.
11. Saudara Anggakara Yudha Pradana, Andik Nur Achmad, Qidam Lubis ST., Andhika Herdiawan, Aflahlana Septian Habibina atas bantuan, dukungan serta penciptaan suasana persaingan yang kompetitif selama studi di kampus ini.
12. Teman-teman angkatan 2006 (Ge-Force), teman anggota aktif RisTIE dan Workshop dan EME TEUB 2008/2009 dan 2009/2010 atas segala bantuan serta motivasinya selama menjadi mahasiswa.
13. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Hanya doa yang bisa penulis berikan, semoga Allah SWT tidak pernah melepaskan kemanjaan segala RahmatNya yang tak terhingga buat kita selama-lamanya. Amin

Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 25 November 2011

Penulis

DAFTAR ISI

ABSTRAK

Alan Nur. 2011. : Kriptografi Menggunakan Metode Rijndael Untuk Pengiriman MMS Pada Aplikasi Mobile Wireless Teknologi Teknik Elektro Universitas Brawijaya. Dosen Pembimbing : Ir. M. Aswin, MT dan Adharul M, ST., MT.

MMS atau *Multimedia Messaging Service* adalah jasa layanan pesan yang memfasilitasi para pengguna telepon selular untuk melalukan pertukaran pesan multimedia. Kebebasan teknologi sekarang memaksa kita selalu terbuka dan setiap saat bisa merugikan kita karena sebuah informasi rahasia yang seharusnya bersifat pribadi bisa di dibaca dan disalahgunakan oleh orang lain oleh karena itu program ini mengangkat bagaimana informasi rahasia melalui MMS hanya bisa dibaca oleh pengirim dan penerima saja.

Pengiriman MMS yang telah dienkripsi dengan menggunakan metode Rijndael pada *handheld* merupakan terobosan baru untuk menjamin kerahasiaan MMS yang dikirimkan. Isi asli dari MMS tersebut hanya dapat dibaca oleh pengirim dan penerima MMS tersebut

Hasil pengujian yang telah dilakukan menunjukan algoritma kriptografi kunci privat untuk enkripsi MMS telah berhasil dibangun. Perangkat lunak yang dibangun tersebut dapat melakukan pengiriman MMS dan penerimaan MMS terenkripsi tersebut dengan baik. Perangkat unak tersebut menggunakan algoritma Rijndael untuk enkripi MMS. Perangkat lunak tersebut dapat ditanamkan pada telepon selular samsung monte dengan nomor XL berkode depan 0817 dan dibangun dengan menggunakan bahasa pemrograman J2ME.

Kata Kunci : MMS, kriptografi, algoritma rijndael, J2ME.

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi/data secara jarak jauh. Antar kota antar wilayah antar negara bahkan antar benua bukan merupakan suatu kendala lagi dalam melakukan komunikasi dan pertukaran data. Seiring dengan itu tuntutan akan sekuritas (keamanan) terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu-individu tidak ingin informasi yang disampaikannya diketahui oleh orang lain atau kompetitornya atau negara lain. Oleh karena itu dikembangkanlah cabang ilmu yang mempelajari tentang cara-cara pengamanan data atau dikenal dengan istilah Kriptografi.

Kriptografi adalah ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu pesan. Dalam kriptografi terdapat dua konsep utama yakni enkripsi dan dekripsi. Enkripsi adalah suatu proses yang melakukan perubahan suatu kode dari yang bisa dimengerti menjadi tidak bisa dimengerti (tidak terbaca). Dekripsi adalah suatu proses dengan algoritma yang sama untuk mengembalikan informasi teracak tadi menjadi bentuk aslinya

Multimedia messaging didefinisikan oleh 3GPP dan WAP sebagai badan standarisasi. *Multimedia Messaging Service (*MMS*)* menggunakan WAP sebagai sarana transportasi dan *independent* sebagai *bearer*nya sehingga membuatnya bisa berjalan melalui jaringan GPRS. Layanan MMS yang diluncurkan menggunakan jaringan GPRS akan menawarkan fasilitas yang lebih bagi para pengguna. Jaringan GPRS menyediakan peningkatan yang penting dalam hal *bandwidth* dan bantuan peningkatan kerja layanan MMS dan penggunaannya.

Bahkan sekarang ini untuk meningkatkan jumlah pelanggan, biaya yang dikenakan untuk pengiriman sebuah SMS dan MMS dibuat murah oleh penyedia layanan komunikasi bergerak. Oleh karena itu, SMS dan MMS sebagai media alternatif penyampaian pesan akhir-akhir ini cukup diminati oleh masyarakat, dan karenanya membuat para penyedia layanan tersebut berlomba-lomba meningkatkan pelayanannya dengan cara meningkatkan kecepatan pengiriman pesan.

1.2 Rumusan Masalah

Berdasarkan alasan di atas, maka terbentuk suatu masalah yaitu :

1. Bagaimana merancang sistem yang merahasiakan isi dari mms agar hanya dapat dibaca oleh pengirim dan penerima mms saja?
2. Bagaimana implementasi kriptografi rijndael yang digunakan untuk memproteksi isi dari mms?
3. Bagaimana pengujian dan analisis kerja sistem pengiriman mms dengan proteksi kriptografi rijndael pada handheld yang berbasis java?

1.3 Tujuan Penulisan

Tujuan dari tugas akhir ini adalah menghasilkan suatu aplikasi yang dapat mengirim dan membaca MMS yang hanya bisa terbaca oleh orang yang mempunyai account (pengirim dan penerima) melalui proses penyandian kriptografi rijndael menggunakan program Java.

1.4 Batasan Masalah penulisan

Pembahasan skripsi ini dibatasi oleh hal-hal sebagai berikut:

1. Isi pesan berupa MMS yang diambil pada ponsel berupa pesan *Text* dan *Image* (format JPEG)
2. Pembuatan program menggunakan software Netbeans IDE 7.0, dua buah handphone GSM yaitu sebagai *Request* (permintaan) dan satu sebagai *reply* (memberikan jawaban), dan menggunakan Java 2 Micro Edition (J2ME) serta wireless toolkit (WTK 2.5.2)
3. Membahas tentang algoritma Rijndael yang meliputi sistem enkripsi dan dekripsi

BAB II
TINJAUAN PUSTAKA

2.1 *Multimedia Messaging Service* (MMS)

MMS atau *Multimedia Messaging Service* adalah jasa layanan pesan yang memfasilitasi para pengguna telepon selular untuk melalukan pertukaran pesan multimedia. MMS dapat dikatakan sebagai bentuk evolusi dari SMS atau *Short Messaging Service*, dimana pada layanan pesan tersebut terdapat transmisi jenis media tambahan yang meliputi teks, image, audio, animasi, video clip atau kombinasi antar media-media tersebut.

Multimedia Messaging Service (MMS) adalah sebuah standar layanan pesan telepon yang memungkinkan untuk mengirim pesan yang mengandung objek multimedia, seperti gambar, audio, video, dan *rich text*. Layanan ini berbeda dengan SMS (*Short Messaging Service*) yang hanya dapat mengirim pesan teks saja. MMS digunakan bersama-sama dalam sebuah jaringan selular dengan sistem perpesanan lainnya, seperti SMS, Mobile Instant Messaging, dan Mobile E-mail. Standarisasi MMS dilakukan oleh 3GPP (*3rd Generation Partnership Project*), 3GPP2, dan OMA (*Open Mobile Alliance*).

MMS merupakan layanan pesan yang bersifat *non-real time*. Proses pengiriman MMS sama seperti SMS yaitu dalam mode *store and forward* menggunakan *kanal traffic*, bukan menggunakan *kanal signaling* seperti pada SMS. MMS disimpan dalam MMSC (MMS *Centre*) dan diforward seperti pada SMS. Dengan MMS ini kita dapat menikmati suatu pesan gambar berwarna, diiringi dengan suara dan penjelasan berupa teks, sehingga pesan dinamis.

Multimedia messaging didefinisikan oleh 3GPP dan WAP sebagai badan standardisasi. *Multimedia Messaging Service (*MMS*)* menggunakan WAP sebagai sarana transportasi dan *independent* sebagai *bearer*nya sehingga membuatnya bisa berjalan melalui jaringan GPRS. Layanan MMS yang diluncurkan menggunakan jaringan GPRS akan menawarkan fasilitas yang lebih bagi para pengguna. Jaringan GPRS menyediakan peningkatan yang penting dalam hal *bandwidth* dan bantuan peningkatan kerja layanan MMS dan penggunaannya.

Telepon selular *(handphone)* yang telah memiliki fitur MMS memungkinkan pengguna untuk membuat dan mengirim pesan dengan satu atau beberapa konten multimedia. Konten ini dapat berupa *text*, gambar, audio, dan video. Tipe konten ini harus memnuhi standar MMS. Sebagai contoh, suatu tipe handphone dapat mengirim sebuah video MPEG-4 dalam format AVI, tapi tipe telepon yang menerima pesan tersebut tidak dapat memainkan konten tersebut. Untuk menghindari hal ini, semua handphone harus memenuhi standar yang telah ditetapkan oleh OMA.

2.1.1 Sejarah MMS

MMS pertama kali dikembangkan oleh 3GPP, organisasi yang bekerja untuk pembuatan standarisasi jaringan UMTS/GSM. Sejak itu, MMS dipakai di seluruh penjuru dunia dan digunakan antar jaringan GSM/GPRS dan CDMA. MMS kemudian juga telah mengalami standarisasi oleh 3GPP2, sebuah organisasi yang berfokus untuk pembuatan spesifikasi dari jaringan CDMA2000. Seperti halnya kebanyakan standar 3GPP lainnya, standar MMS memiliki 3 *stage*/ tahapan :

- Tahapan 1 - *Requirements* / Persyaratan (3GPP TS 22.140)
- Tahapan 2 - *System Functions* / Fungsi Sistem (3GPP TS 23.140)
- Tahapan 3 - *Technical Realizations* / Realisasi Teknik

Baik 3GPP dan 3GPP2 telah menyerahkan pengembangan dari tahapan 3 (*Technical Realizations*) ke OMA, sebuah organisasi yang berfokus dalam spesifikasi jaringan nirkabel. Selain itu, asosiasi GSM juga telah membuat dokumen panduan interkoneksi MMS (MMS *Interworking Guidelines*} IR.52 yang dipergunakan sebagai standar inter-koneksi antar operator GSM.

2.1.2 Format pada MMS

Adapun format-format yang dapat disimpan melalui MMS yaitu:

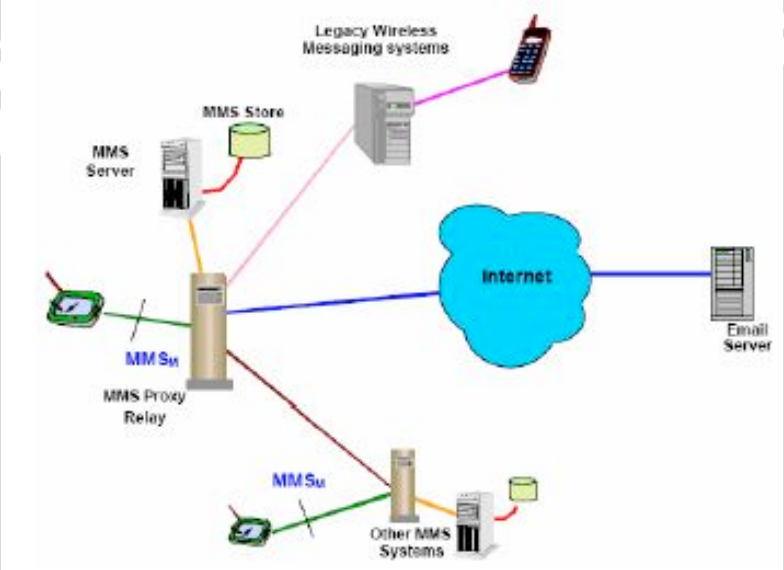
- *Text* (dengan format *fonts*, warna, dan lain-lain)
- *Image* (format JPEG, PNG, WBMP)
- Audio (format WAV, AMR)
- Video (format MPEG)

Data-data yang digunakan tidak dapat lebih dari 30KB dan untuk image digunakan dari dimensi kecil sampai dimensi besar namun ukurannya masih harus dibawah ukuran display 320x208 pixel.

2.1.3 Struktur Pesan MMS

Sampai sekarang ada dua standarisasi internasional yang mengatur tentang layanan MMS ini, yakni 3GPP dan WAP Forum. Perbedaan yang paling mendasar dari kedua standarisasi ini adalah standarisasi oleh 3GPP bersifat global dan standar dari WAP Forum lebih bersifat spesifik, yaitu langsung dengan implementasi menggunakan protocol WAP. Dan standarisasi oleh WAP Forum ini tetap mengacu pada standar 3GPP.

Secara umum arsitektur MMS seperti pada Gambar dibawah:

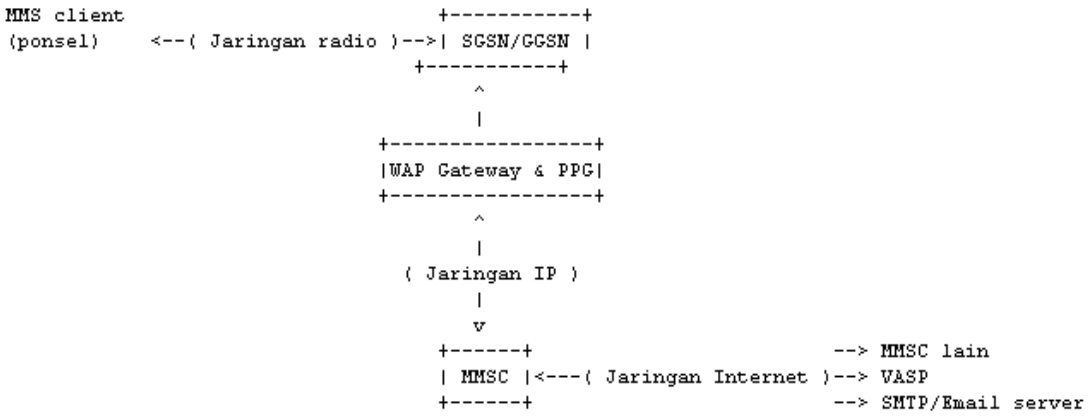


Gambar2.1 Elemen-Elemen MMS

MMS *Environment* (MMSE) merupakan kumpulan elemen-elemen tertentu dalam satu administrasi pengaturan. MMSE meliputi semua layanan dari elemen-elemen untuk proses penyampaian (*delivery*), penyimpanan (*storage*) dan pemberitahuan (*notification*) suatu message. MMS atau M*ultimedia Messaging Service* sebenarnya adalah hasil pengembangan dari SMS (*Short Message Service*). Pada teknologi J2ME pengiriman multimedia dengan MMS, ukuran file masih dibatasi maksimum 30 KB karena jika ukuran terlalu besar maka MIDlet akan menjadi *error*.

Struktur pesan MMS sesuai dengan format yang biasa digunakan dalam sistem internet pada MMS, setiap pesan terdiri dari bagian *message envelope* dan *message content. Message envelope* memiliki *field-field* yang diperlukan untuk proses penyampaian dan *interpretasi* isi pesan. Jika isi pesan terdiri dari *non-tekstual* seperti *image, audio*, dan *video* dalam pesan *multimedia*, maka format isi pesan tersebut berupa *multipart message* berdasarkan *Multipurpose Internet Mail Extensions* atau MIME. Sebuah pesan MMS dapat dibentuk seperti *slide show presentation*, yaitu pesan MMS terdiri dari beberapa media tertentu yang memiliki durasi, layout dan bagaimana ditampilkannya.

MMS dan fungsionalitas MMSC dispesifikasikan oleh 3GPP dan OMA. Definisi dan format MMS serta proses pengiriman/penerimaan MMS dapat dibaca pada spesifikasi tersebut. Pada dasarnya MMSC merupakan penggabungan dua fungsi yaitu MMS Server dan MMS *proxy/relay*. MMS Server berfungsi sebagai tempat antrian atau penyimpanan MM (MMBox). Sedangkan MMS *proxy/relay* berfungsi sebagai elemen yang menghubungkan MMS Server dengan ponsel pengguna, melakukan inisialisasi koneksi, mengirimkan notifikasi, routing dan lain-lain. Arsitektur yang umum adalah sebagai berikut



Gambar2.2 Arsitektur MMS

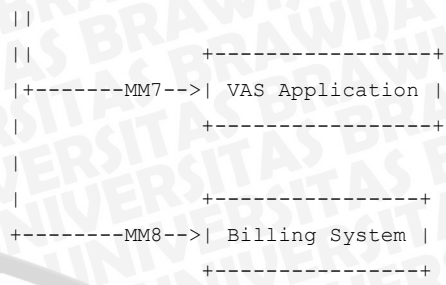
Titik-titik integrasi (*reference point*) yang menunjukan hubungan antara MMSC dengan elemen lain diberi nama dan dispesifikasikan dalam dokumen 2GPP TS 20.140 (*Multimedia Messaging Service Functional Description*) sebagai berikut:

- **MM1**: *Reference point* antara MMS *User Agent* dengan MMS Relay/Server. Biasanya menggunakan WAP/WSP, walaupun dalam

spesifikasi dimungkinkan untuk menggunakan protokol lain berbasis TCP/IP misalnya HTTP

- **MM2**: *Reference point* antara MMS Relay dengan MMS Server. Belum dispesifikasikan.
- **MM3**: *Reference point* antara MMS Relay/Server dengan *external* (*legacy*) *messaging systems*, misalnya MMSC lain atau Mail Server. Pada *reference point* ini biasanya digunakan protokol SMTP/IMAP.
- **MM4**: *Reference point* antara the MMS Relay/Server dengan MMS Relay/Server yang lain yang berada di lain MMSE (*Multimedia Message Service Environment*). Protocol yang digunakan adalah SMTP (RFC 821). STD 11 (RFC 2822), MIME (RFC 2046)
- **MM5**: *Reference point* antara the MMS Relay/Server dengan Home Location Register (HLR). Menggunakan MAP.
- **MM6**: *Reference point* antara the MMS Relay/Server dengan MMS User Databases. Belum dispesifikasikan.
- **MM7**: *Reference point* antara the MMS Relay/Server dengan MMS VAS Applications. Berbasis SOAP dan SOAP *message with attachment* [http://www.w3.org/TR/SOAP-attachments] dengan HTTP sebagai transport layer.
- **MM8**: *Reference point* antara the MMS Relay/Server dengan *billing system*. Belum dispesifikasikan.

Titik-titik integrasi (*reference point*) yang umum adalah sebagai berikut



Gambar 2.3 Titik-titik Integrasi (*Reference Point*)

2.1.4 Penanganan Berkas MMS di perangkat mobile

2.1.4.1 Pengiriman MMS

Jaringan *packet switched* pada *core network* digunakan sebagai lalu-lintas sebuah *multimedia message* (MM). MM akan dikirimkan oleh ponsel ke sebuah elemen yang berfungsi sebagai pengatur lalu-lintas dan penyimpan MM yang disebut MMSC (*Multimedia Messaging Service Center*). Proses peringiriman MMS dapat dijelaskan secara garis besar sebagai berikut:

1. Sebuah ponsel harus mengetahui alamat dari MMSC operator agar bisa mengirimkan sebuah MMS. Ponsel akan melakukan koneksi GPRS dan session GPRS dibuat pada SGSN.
2. Lewat WAP gateway, MMS dikirimkan ke MMSC menggunakan protokol HTTP atau WSP/WAP.
3. MMSC akan mengirimkan MMS ke tujuan. Jika tujuan adalah pelanggan pada operator yang sama maka MMSC akan mengirimkan WAP Push indikator-MMS ke ponsel tujuan. Jika tujuan adalah alamat email maka SMSC akan mengirimkannya ke Email Server tujuan.
4. Jika Wap Push indikator-MMS tidak dapat dikirimkan maka MMS akan disimpan ke dalam tempat penyimpanan atau *Multimedia Message Box* (MMBox).
5. MMSC kemudian akan melakukan pengiriman ulang (retry) ke tujuan beberapa kali. Jika hingga batas retry MMS tidak dapat dikirimkan maka biasanya MMSC akan mengirimkan pesan SMS kepada nomor tujuan memberitahukan bahwa sebuah MMS diterima dan dapat diambil melalui alamat web site (URL) tertentu dalam batas waktu tertentu pula. Jika penerima MMS tidak mendownload pesan yang dikirimkan kepadanya

hingga melewati batas waktu *expiring date* dari pesan, maka pesan tersebut akan dihapus oleh server.

2.1.4.2 Penerimaan MMS

MMSC sebagai elemen yang menerima MMS dari subscriber akan mengirimkan notifikasi (MMS notification) kepada ponsel tujuan. Notifikasi MMS yang memberitahukan ponsel bahwa subscriber mendapat kiriman MMS diterima ponsel dalam bentuk WAP-push yang dikirim lewat SMS bearer. WAP-push indikator tersebut memiliki header content type yang berisi "application/vnd.wap.mms-message" dan header X-Wap-Application-Id yang berisi "x-wap-application:mms.ua". Dari header tersebut ponsel tau bahwa WAP push tersebut adalah MMS notification indicator dan harus diproses oleh MMS agent. MMS agent kemudian memproses notification tersebut yaitu meresponse untuk memberitahu MMSC bahwa notification indicator telah diterima kemudian melakukan request (WSP/HTPP GET) ke MMSC untuk mendownload content dari MMS. Sebuah MMS notification indicator membawa informasi diantaranya:

a. Nomor versi MMS
b. Alamat pengirim MMS
c. Subject dari MMS
d. Deliveri report status yang menunjukan pengirim membutuhkan staus delivery report atau tidak
e. MMS message class, yang menunjukan prioritas pengiriman
f. Ukuran MMS
g. Waktu kadaluarsa (expiry) dari MMS
h. Lokasi MMS content yaitu alamat untuk mengambil isi dari MMS

Informasi tersebut dispesifikasikan pada dokumen spesifikasi OMA yang berjudul "Multimedia Messaging Service Encapsulation Protocol" dan juga spesifikasi 3GPP TS 23.140 (Multimedia Messaging Service (MMS); Functional description) pada bagian yang membahas tentang MM1 interface. MM1 interface adalah interface yang menhubungkan antara MMSC dengan ponsel.

Dalam spesifikasi OMA, notifikasi MMS disebut M-Notification.ind dan serponsenya disebut M-NotifyResp.ind. Sedangkan pada MM1 interface,

notifikasi MMS disebut MM1_notification.REQ dan responnya disebut MM1_notification.RES

Sebuah ponsel biasanya memiliki setting untuk penerimaan MMS yaitu

- Cara penerimaan yang menentukan apakah MMS client/agent akan otomatis mengambil MMS content setelah mendapat MMS notifikasi
- APN (Access Point Name) yang merupakan setting GPRS bearer agar MMS content dapat dikirim dari ponsel ke MMSC atau diambil dari MMSC

2.2 *Global System for Mobile Communication* (GSM)

Global System for Mobile Communication disingkat GSM adalah sebuah teknologi komunikasi selular yang bersifat digital. Teknologi GSM banyak diterapkan pada komunikasi bergerak, khususnya telepon genggam. Teknologi ini memanfaatkan gelombang mikro dan pengiriman sinyal yang dibagi berdasarkan waktu, sehingga sinyal informasi yang dikirim akan sampai pada tujuan. GSM dijadikan standar global untuk komunikasi selular sekaligus sebagai teknologi selular yang paling banyak digunakan orang di seluruh dunia.

2.2.1 Sejarah dan Perkembangan GSM

Teknologi komunikasi selular sebenarnya sudah berkembang dan banyak digunakan pada awal tahun 1980-an, diantaranya sistem C-NET yang dikembangkan di Jerman dan Portugal oleh Siemens, sistem RC-2000 yang dikembangkan di Prancis, sistem NMT yang dikembangkan di Belanda dan Skandinavia oleh Ericsson, serta sistem TACS yang beroperasi di Inggris. Namun teknologinya yang masih analog membuat sistem yang digunakan bersifat regional sehingga sistem antara negara satu dengan yang lain tidak saling kompatibel dan menyebabkan mobilitas pengguna terbatas pada suatu area sistem teknologi tertentu saja (tidak bisa melakukan roaming antar negara).

Teknologi analog yang berkembang, semakin tidak sesuai dengan perkembangan masyarakat Eropa yang semakin dinamis, maka untuk mengatasi keterbatasannya, negara-negara Eropa membentuk sebuah organisasi pada tahun 1982 yang bertujuan untuk menentukan standar-standar komunikasi selular yang dapat digunakan di semua negara Eropa. Organisasi ini dinamakan Group Special Mobile (GSM). Organisasi ini memelopori munculnya teknologi digital selular

yang kemudian dikenal dengan nama *Global System for Mobile Communication* atau GSM.

GSM muncul pada pertengahan 1991 dan akhirnya dijadikan standar telekomunikasi selular untuk seluruh Eropa oleh ETSI (*European Telecomunication Standard Institute*). Pengoperasian GSM secara komersil baru dapat dimulai pada awal kuartal terakhir 1992 karena GSM merupakan teknologi yang kompleks dan butuh pengkajian yang mendalam untuk bisa dijadikan standar. Pada September 1992, standar type approval untuk *handphone* disepakati dengan mempertimbangkan dan memasukkan puluhan *item* pengujian dalam memproduksi GSM.

Pada awal pengoperasiannya, GSM telah mengantisipasi perkembangan jumlah penggunanya yang sangat pesat dan arah pelayanan per area yang tinggi, sehingga arah perkembangan teknologi GSM adalah DCS **(***Digital Cellular System***)** pada alokasi frekuensi 1800 Mhz. Dengan frekuensi tersebut, akan dicapai kapasitas pelanggan yang semakin besar per satuan sel. Selain itu, dengan luas sel yang semakin kecil akan dapat menurunkan kekuatan daya pancar *handphone*, sehingga bahaya radiasi yang timbul terhadap organ kepala akan dapat di kurangi. Pemakaian GSM kemudian meluas ke Asia dan Amerika, termasuk Indonesia. Indonesia awalnya menggunakan sistem telepon selular *analog* yang bernama AMPS (Advances Mobile Phone System) dan NMT (Nordic *Mobile* Telephone). Namun dengan hadir dan dijadikannnya standar sistem komunikasi selular membuat sistem *analog* perlahan menghilang, tidak hanya di Indonesia, tapi juga di Eropa. Pengguna GSM pun semakin lama semakin bertambah.

Pada akhir tahun 2005, pelanggan GSM di dunia telah mencapai 1,5 triliun pelanggan. Akhirnya GSM tumbuh dan berkembang sebagai sistem telekomunikasi seluler yang paling banyak digunakan di seluruh dunia.

2.2.2 Spesifikasi Teknis GSM

Di Eropa, pada awalnya GSM didesain untuk beroperasi pada frekuensi 900 Mhz. Pada frekuensi ini, frekuensi uplinks-nya digunakan frekuensi 890–915 MHz, sedangkan frekuensi downlinks nya menggunakan frekuensi 935–960 MHz. Bandwith yang digunakan adalah 25 Mhz (915–890 = 960–935 = 25 Mhz), dan

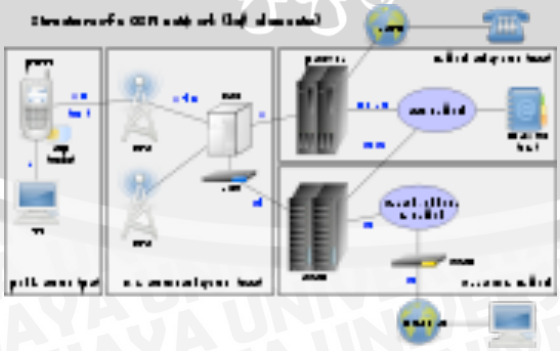
lebar kanal sebesar 200 Khz. Dari keduanya, maka didapatkan 125 kanal, dimana 124 kanal digunakan untuk suara dan satu kanal untuk sinyal. Pada perkembangannya, jumlah kanal 124 semakin tidak mencukupi dalam pemenuhan kebutuhan yang disebabkan pesatnya pertambahan jumlah pengguna. Untuk memenuhi kebutuhan kanal yang lebih banyak, maka regulator GSM di Eropa mencoba menggunakan tambahan *frekuensi* untuk GSM pada band *frekuensi* di range 1800 Mhz dengan frekuensi 1710-1785 Mhz sebagai frekuensi *uplinks* dan frekuensi 1805-1880 Mhz sebagai frekuensi *downlinks*. GSM dengan frekuensinya yang baru ini kemudian dikenal dengan sebutan GSM 1800, yang menyediakan *bandwidth* sebesar 75 Mhz (1880-1805 = 1785–1710 = 75 Mhz). Dengan lebar kanal yang tetap sama yaitu 200 Khz sama, pada saat GSM pada frekuensi 900 Mhz, maka pada GSM 1800 ini akan tersedia sebanyak 375 kanal. Di Eropa, standar-standar GSM kemudian juga digunakan untuk komunikasi railway, yang kemudian dikenal dengan nama GSM-R.

2.2.3 Arsitektur Jaringan GSM

Secara umum, network element dalam arsitektur jaringan GSM dapat dibagi menjadi:

1. *Mobile Station* (MS)
2. *Base Station* Sub-system (BSS)
3. *Network Sub-system* (NSS),
4. *Operation and Support System* (OSS)

Secara bersama-sama, keseluruhan *network element* di atas akan membentuk sebuah PLMN *(Public Land Mobile Network)*.



Gambar2.4 Struktur jaringan GSM

2.2.3.1 *Mobile Station*

Mobile Station atau MS merupakan perangkat yang digunakan oleh pelanggan untuk melakukan pembicaraan. Terdiri atas:

- *Mobile Equipment* (ME) atau handset, merupakan perangkat GSM yang berada di sisi pengguna atau pelanggan yang berfungsi sebagai terminal transceiver (pengirim dan penerima sinyal) untuk berkomunikasi dengan perangkat GSM lainnya.
- *Subscriber Identity Module* (SIM) atau SIM *Card*, merupakan kartu yang berisi seluruh informasi pelanggan dan beberapa informasi pelayanan. ME tidak akan dapat digunakan tanpa SIM didalamnya, kecuali untuk panggilan darurat. Data yang disimpan dalam SIM secara umum, adalah:
    1. IMMSI *(International Mobile Subscriber Identity)*, merupakan penomoran pelanggan.
    2. MSISDN *(Mobile Subscriber ISDN)*, nomor yang merupakan nomor panggil pelanggan.

2.2.3.2 *Base Station System*

Base Station System atau BSS, terdiri atas:

- BTS *Base Transceiver Station*, perangkat GSM yang berhubungan langsung dengan MS dan berfungsi sebagai pengirim dan penerima sinyal.
- BSC *Base Station Controller*, perangkat yang mengontrol kerja BTS-BTS yang berada di bawahnya dan sebagai penghubung BTS dan MSC

2.2.3.3 *Network Sub System*

Network Sub System atau NSS, terdiri atas:

- *Mobile Switching Center* atau MSC, merupakan sebuah network element central dalam sebuah jaringan GSM. MSC sebagai inti dari jaringan seluler, dimana MSC berperan untuk interkoneksi hubungan pembicaraan, baik antar selular maupun dengan jaringan kabel PSTN, ataupun dengan jaringan data.
- *Home Location Register* atau HLR, yang berfungsi sebagai sebuah database untuk menyimpan semua data dan informasi mengenai pelanggan agar tersimpan secara permanen.
- *Visitor Location Register* atau VLR, yang berfungsi untuk menyimpan data dan informasi pelanggan.

- *Authentication Center* atau AuC, yang diperlukan untuk menyimpan semua data yang dibutuhkan untuk memeriksa keabsahaan pelanggan. Sehingga pembicaraan pelanggan yang tidak sah dapat dihindarkan.
- *Equipment Identity Registration* atau EIR, yang memuat data-data pelanggan.

2.2.3.4 *Operation and Support System*

Operation and Support System atau OSS, merupakan sub sistem jaringan GSM yang berfungsi sebagai pusat pengendalian, diantaranya *fault management, configuration management, performance management, dan inventory management.*

Terdapat Frekuensi pada 3 Operator Terbesar di Indonesia, yaitu:

1. Indosat : 890 – 900 Mhz (10 Mhz)
2. Telkomsel : 900 – 907,5 Mhz (7,5 Mhz)
3. Excelcomindo : 907,5 – 915 Mhz (7,5 Mhz)

Tabel 2.1 Operator dan layanan telekomunikasi seluler di Indonesia

Operator dan Layanan Telekomunikasi Seluler Indonesia		
CDMA	Bakrie Telecom	Esia
	Indosat	StarOne
	Mobile-8	Fren · Hepi
	Sampoerna	Ceria
	Smart Telecom	Smart
	Telkom	TelkomFlexi
GSM	Hutchison	3
	Indosat	IM3 · Mentari · Matrix
	Natrindo	AXIS
	PSN	ByRU · PASTI
	Telkomsel	simPATI · kartuAs · kartuHALO
	XL Axiata	XL Prabayar · XL Pascabayar

2.2.4 Keunggulan GSM sebagai Teknologi Generasi Kedua (2G)

GSM, sebagai sistem telekomunikasi selular digital memiliki keunggulan yang jauh lebih banyak dibanding sistem analog, di antaranya:

- Kapasitas sistem lebih besar, karena menggunakan teknologi digital dimana penggunaan sebuah kanal tidak hanya diperuntukkan bagi satu pengguna saja. Sehingga saat pengguna tidak mengirimkan informasi, kanal dapat digunakan oleh pengguna lain.
- Sifatnya yang sebagai standar internasional memungkinkan *international roaming.*
- Dengan teknologi digital, tidak hanya mengantarkan suara, tapi memungkinkan servis lain seperti teks, gambar, dan video.
- Keamanan sistem yang lebih baik
- Kualitas suara lebih jernih dan peka.
- Mobile (dapat dibawa kemana-mana)

Bagaimanapun, keunggulan GSM yang beragam pantas saja membuatnya menjadi sistem telekomunikasi selular terbesar penggunanya di seluruh dunia.

2.3 Kriptografi

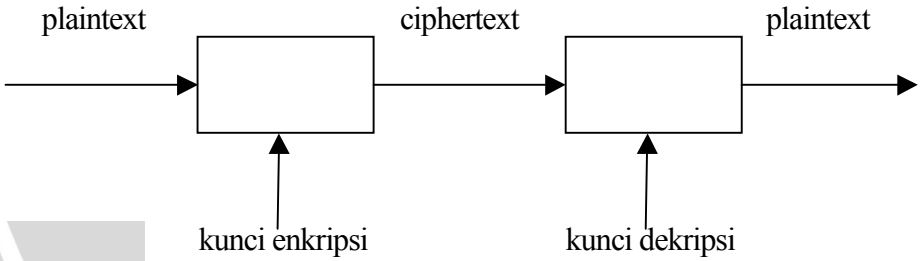
Berkat perkembangan teknologi yang begitu pesat memungkinkan manusia dapat berkomunikasi dan saling bertukar informasi/data secara jarak jauh. Antar kota antar wilayah antar negara bahkan antar benua bukan merupakan suatu kendala lagi dalam melakukan komunikasi dan pertukaran data. Seiring dengan itu tuntutan akan sekuritas (keamanan) terhadap kerahasiaan informasi yang saling dipertukarkan tersebut semakin meningkat. Begitu banyak pengguna seperti departemen pertahanan, suatu perusahaan atau bahkan individu-individu tidak ingin informasi yang disampaikannya diketahui oleh orang lain atau kompetitornya atau negara lain. Oleh karena itu dikembangkanlah cabang ilmu yang mempelajari tentang cara-cara pengamanan data atau dikenal dengan istilah Kriptografi.

Kriptografi adalah ilmu dan seni penyandian yang bertujuan untuk menjaga keamanan dan kerahasiaan suatu pesan. Dalam kriptografi terdapat dua konsep utama yakni enkripsi dan dekripsi. Enkripsi adalah suatu proses yang melakukan perubahan suatu kode dari yang bisa dimengerti menjadi tidak bisa dimengerti (tidak terbaca). Dekripsi adalah suatu proses dengan algoritma yang sama untuk mengembalikan informasi teracak tadi menjadi bentuk aslinya.

Enkripsi **Dekripsi**

Konsep kriptografi sendiri telah lama digunakan oleh manusia misalnya pada peradaban Mesir dan Romawi walau masih sangat sederhana. Prinsip-prinsip yang mendasari kriptografi yakni:

- *Confidelity* (kerahasiaan) yaitu layanan agar isi pesan yang dikirimkan tetap rahasia dan tidak diketahui oleh pihak lain (kecuali pihak pengirim, pihak penerima / pihak-pihak memiliki ijin). Umumnya hal ini dilakukan dengan cara membuat suatu algoritma matematis yang mampu mengubah data hingga menjadi sulit untuk dibaca dan dipahami.
- *Data integrity* (keutuhan data) yaitu layanan yang mampu mengenali/ mendeteksi adanya manipulasi (penghapusan, pengubahan atau penambahan) data yang tidak sah (oleh pihak lain).
- *Authentication* (keotentikan) yaitu layanan yang berhubungan dengan identifikasi. Baik otentikasi pihak-pihak yang terlibat dalam pengiriman data maupun otentikasi keaslian data/informasi.
- *Non-repudiation* (anti-penyangkalan) yaitu layanan yang dapat mencegah suatu pihak untuk menyangkal aksi yang dilakukan sebelumnya (menyangkal bahwa pesan tersebut berasal dirinya).



Gambar 2.5 Proses Enkripsi dan Dekripsi

Berbeda dengan kriptografi klasik yang menitikberatkan kekuatan pada kerahasiaan algoritma yang digunakan (yang artinya apabila algoritma yang digunakan telah diketahui maka pesan sudah jelas "bocor" dan dapat diketahui isinya oleh siapa saja yang mengetahui algoritma tersebut), kriptografi modern lebih menitikberatkan pada kerahasiaan kunci yang digunakan pada algoritma tersebut (oleh pemakainya) sehingga algoritma tersebut dapat saja disebarkan ke kalangan masyarakat tanpa takut kehilangan kerahasiaan bagi para pemakainya.

Berikut adalah istilah-istilah yang digunakan dalam bidang kriptografi :

- **Plaintext** (M) adalah pesan yang hendak dikirimkan (berisi data asli).

- **Ciphertext** (C) adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
- **Enkripsi** (fungsi E) adalah proses pengubahan *plaintext* menjadi *ciphertext*.
- **Dekripsi** (fungsi D) adalah kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga berupa data awal/asli.
- **Kunci** adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi.

Dalam sistem komputer, pesan terbuka (*plaintext*) diberi lambang M, yang merupakan singkatan dari *Message*. *Plaintext* ini dapat berupa tulisan, foto, atau video yang berbentuk data biner. *Plaintext* inilah yang nantinya akan dienkripsi menjadi pesan rahasia atau *ciphertext* yang dilambangkan dengan C (*ciphertext*). Secara matematis, fungsi enkripsi ini dinotasikan dengan :

$$E(M) = C$$

Sedangkan fungsi dekripsi adalah proses pembalikan dari *ciphertext* menjadi *plaintext* kembali. Secara matematis dinotasikan sebagai berikut :

$$D(C) = M$$

$$D(E(M)) = M$$

Dalam *cryptography* modern algoritma yang digunakan tidak lagi rahasia, yang dirahasiakan hanyalah *key* untuk melakukan enkripsi pesan.

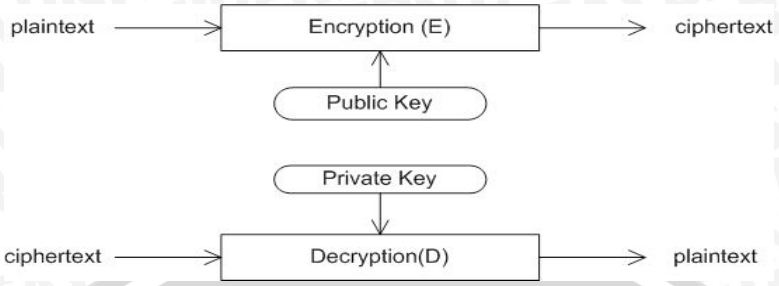
2.3.1 Pembagian Algoritma Kriptografi Berdasarkan Jenis *Key* nya

2.3.1.1 Kriptografi Asimetris

Cryptography public key sering disebut dengan kriptografi asimetris. *key* yang digunakan pada proses enkripsi dan proses dekripsi pada *cryptography public key* ini berbeda satu sama lain. Jadi dalam *cryptography public key*, suatu *key generator* akan menghasilkan dua *key* berbeda dimana satu *key* digunakan untuk melakukan proses enkripsi dan *key* yang lain digunakan untuk melakukan proses dekripsi.

Key yang digunakan untuk melakukan enkripsi akan dipublikasikan kepada umum untuk dipergunakan secara bebas. Oleh sebab itu, *key* yang digunakan untuk melakukan enkripsi disebut juga sebagai *public key*. Sedangkan *key* yang digunakan untuk melakukan dekripsi akan disimpan oleh pembuat *key*

dan tidak akan dipublikasikan kepada umum. *Key* untuk melakukan dekripsi ini disebut *private key*.



Gambar2.6 *Cryptography* Asimetris

Dengan cara demikian, semua orang yang akan mengirimkan pesan kepada pembuat *key* dapat melakukan proses enkripsi terhadap pesan tersebut, sedangkan proses dekripsi hanya dapat dilakukan oleh pembuat atau pemilik *key* dekripsi. Dalam kenyataannya kriptografi asimetris ini dipakai dalam *ssh*, suatu layanan untuk mengakses suatu *server*.

Pada umumnya kunci publik *(public key)* digunakan sebagai kunci enkripsi sementara kunci privat *(private key)* digunakan sebagai kunci dekripsi.

Kelebihan :

- Masalah keamanan pada distribusi kunci dapat lebih baik.
- Masalah manajemen kunci yang lebih baik karena jumlah kunci yang lebih sedikit

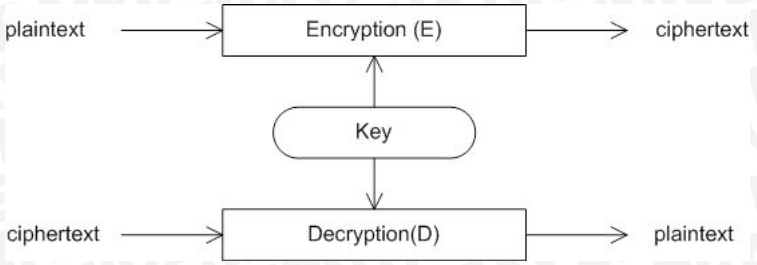
Kelemahan :

- Kecepatan yang lebih rendah bila dibandingkan dengan algoritma simetris.
- Untuk tingkat keamanan sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritma simetris.

Algoritma-algoritma yang termasuk kedalaman *Cryptography* asimetris diantaranya adalah : *Elliptic Curve Cryptography* (ECC), *Digital Signature Algorithm* (DSA), *LUC, RSA, ELGAMAL, Diffie-Hellman* (DH).

2.3.1.2 Kriptografi Simetris

Kriptografi Simetris atau *secret key* adalah kriptografi yang hanya melibatkan satu *key* dalam proses enkripsi dan dekripsi. Proses dekripsi dalam *cryptography secret key* ini adalah kebalikan dari proses enkripsi.



Gambar2.7 *Cryptography* Simetris

Kriptografi simetris dapat dibagi menjadi dua, yaitu penyandian blok (*Block Cipher*) dan penyandian alir (*Stream Cipher*). Penyandian blok bekerja pada suatu data yang terkelompok menjadi blok-blok data atau kelompok data dengan panjang data yang telah ditentukan. Pada penyandian blok, data yang masuk akan dipecah-pecah menjadi blok data yang telah ditentukan ukurannya. Penyandian alir bekerja pada suatu data bit tunggal atau terkadang dalam satu *byte*. Jadi format data yang mengalami proses enkripsi dan dekripsi adalah berupa aliran *bit-bit* data.

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 :

- Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetrik.
- Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem *real-time*

Kelemahan :

- Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
- Permasalahan dalam pengiriman kunci itu sendiri yang disebut *"key distribution problem"*

Algoritma-algoritma yang termasuk ke dalam *cryptography* simetris diantaranya adalah : *Data Encryption Standard* (DES), *International Data Encryption Algorithm* (IDEA), *Rijndael* (AES), *A5*, *RC2*, *RC4*, *RC5*, *RC6* dll.

2.3.1.2.1 *Stream Cipher*

Stream Cipher (Aliran *Cipher*) merupakan suatu *Cipher* yang berasal dari hasil XOR. Setiap *bit plaintext* dengan setiap *bit key*. *Key* merupakan *key*

Key

Pseudo-Random Sequence Generator (PRSG)

Plaintext Bitstream

Plaintext Bitstream

Plaintext Bitstream

Pseudo-Random Stream

Ciphertext Stream

utama (kunci induk) yang digunakan untuk membangkitkan *key* acak semu yang dibangkitkan dengan *Pseudo Random Sequence Generator* yang merupakan suatu nilai yang nampak seperti diacak, tetapi sesungguhnya nilai tersebut merupakan suatu urutan. Secara khusus urutan dari nilai yang dihasilkan oleh RNG (*Random Number Generator*), *computational* mekanisme *deterministic* atau FSM (*Finite State Machine*) merupakan kebalikan dari *Really Random*.

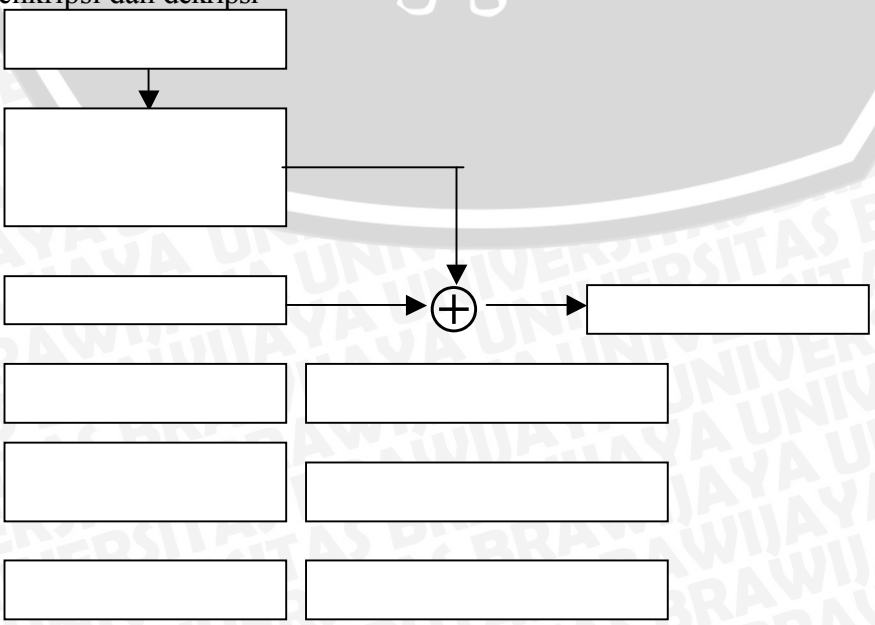
Salah satu pembangkit bilangan acak yang cocok untuk kriptografi dinamakan *cryptographically secure pseudorandom generator* (CSPRNG)

Persyaratan *CSPRNG* adalah:

1. Secara statistik ia mempunyai sifat-sifat yang bagus (yaitu lolos uji keacakan statistik).
2. Tahan terhadap serangan (*attack*) yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan.

Random Number Generator secara umum adalah *Pseudo Random.* Yang memberikan *intial state* atau *seed* (nilai yang di-*input* ke dalam *state*), seluruh urutan tersebut ditentukan secara keseluruhan, tetapi meskipun demikian banyaknya karakteristik yang ditampilkan dari suatu urutan yang diacak tersebut. *Pseudorandomness* menghasilkan urutan yang sama secara berulang-ulang pada penempatan yang berbeda. Kemudian kunci acak semu tersebut diberikan operasi *XOR* dengan *Plaintext* untuk mendapatkan *ciphertext.*

Secara garis besar *Stream Cipher* merupakan algoritma yang dalam operasinya bekerja dalam suatu pesan berupa bit tunggal atau terkadang dalam suatu byte, jadi format data berupa aliran dari bit untuk kemudian mengalami proses enkripsi dan dekripsi



Gambar2.8 Proses *Stream Cipher*

2.3.1.2.2 *Block Cipher*

Block Cipher merupakan suatu logaritma yang mana *input* dan *output* nya berupa satu *block*, dan setiap *block* terdiri dari beberapa *bit* (1 *block* = 64 *bit* atau 128 *bit*). *Block cipher* mempunyai banyak aplikasi, aplikasi tersebut digunakan untuk memberikan layanan *confidentiality* (kerahasiaan), integritas data atau *authentication* (pengesahan pemakai), dan juga bisa memberikan layanan *keystream Generator* untuk *StreamCpiher*.

Pada algoritma penyandian blok *(block cipher),* plainteks yang masuk akan diproses dengan panjang blok yang tetap yaitu *n*, namun terkadang jika ukuran data ini terlalu panjang maka dilakukan pemecahan dalam bentuk blok yang lebih kecil. Jika dalam pemecahan dihasilkan blok data yang kurang dari jumlah data dalam blok maka akan dilakukan proses *pading* (penambahan beberapa bit).

Secara garis besar algoritma kriptografi ini bekerja pada suatu data yang berbentuk blok/kelompok data dengan panjang data tertentu (dalam beberapa byte), jadi dalam sekali proses enkripsi atau dekripsi data yang masuk mempunyai ukuran yang sama.

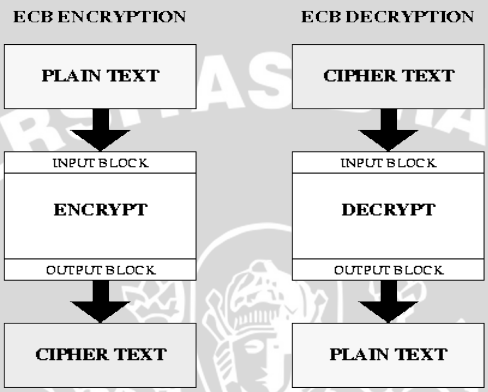
Mode operasi dari *block cipher* terbagi menjadi empat bagian diantaranya adalah :

2.3.1.2.2.1 Mode *Electric Code Book* (ECB)

Menggunakan mode ini suatu *block cipher* yang panjang dibagi dalam bentuk *sequence biner* menjadi satu *block* tanpa mempengaruhi *block* yang lain, satu *block* terdiri dari 64 *bit* atau 128 *bit*, setiap *block* merupakan bagian dari pesan yang di enkripsi.

Mode ini merupakan suatu enkripsi yang sederhana, kerusakan satu *block* data tidak mempengaruhi *block* lainnya, sehingga jika penerima mendapatkan satu *block* yang rusak, maka penerima hanya minta dikirimkan kembali hanya *block* yang rusak, tanpa harus meminta semua *block*, hal ini membantu pengiriman *block* yang rusak dengan cepat. Pada dasarnya sifat yang

paling mendasar dari mode ECB ini adalah *block plaintext* yang sama akan di kodekan menjadi *cipher* yang sama.



Gambar2.9 Mode Operasi ECB

Sifat- sifat dari mode operasi ECB :

- Sederhana dan efisien
- Memungkinkan implementasi parallel
- Tidak menyembunyikan pola plaintext
- Dimungkinkan terjadi adanya active attack.

Keuntungan dari mode OBC ini adalah kemudahan dalam implementasi dan pengurangan resiko salahnya semua *plaintext* akibat kesalahan pada satu *plaintext*. Namun mode ini memiliki kelemahan pada aspek keamanannya. Dengan mengetahui pasangan *plaintext* dan *ciphertext*, seorang *cryptanalist* dapat menyusun suatu *code book* tanpa perlu mengetahui kuncinya.

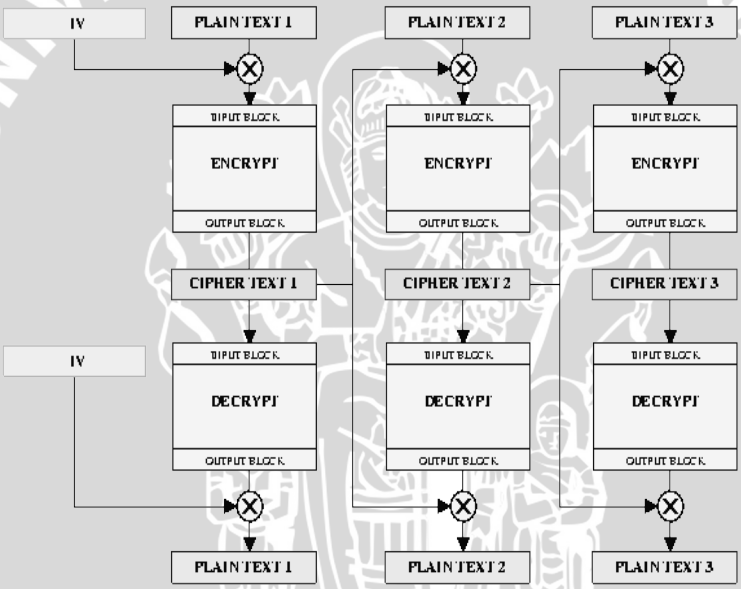
2.3.1.2.2.2 Mode *Cipher Block Chainning* (CBC)

Sistem dari mode *cipher block chainning* (CBC) adalah *plaintext* yang sama akan di enkripsi ke dalam bentuk *cipher* yang berbeda, disebabkan *block cipher* yang satu tidak berhubungan dengan *block cipher* yang lain. Melainkan tergantung pada *cipher* yang sebelumnya.

Pada CBC digunakan operasi umpan balik atau dikenal dengan operasi berantai (*chaining*). Pada CBC, hasil enkripsi dari blok sebelumnya adalah *feedback* untuk enkripsi dan dekripsi pada *block* berikutnya. Dengan kata lain, setiap *block ciphertext* dipakai untuk memodifikasi proses enkripsi dan dekripsi pada *block* berikutnya.

Sifat-sifat dari mode operasi CBC :

- Lebih aman dari active attacks dibandingkan mode operasi ECB
- Error pada satu ciphertext dapat berakibat parah
- Menutupi pola plaintext
- Implementasi parallel belum diketahui



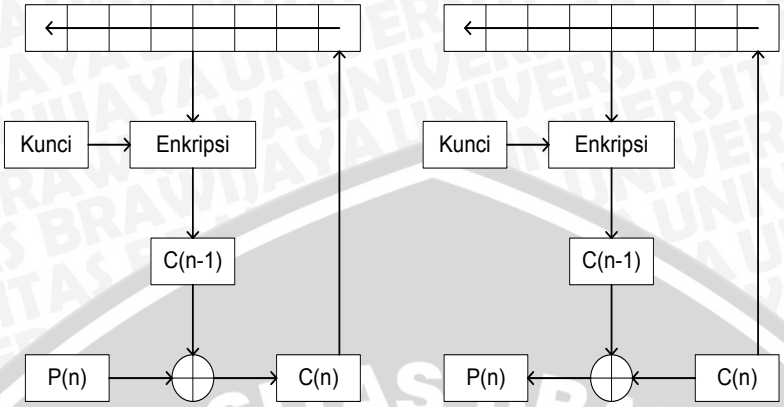
Gambar2.10 Mode Operasi CBC

Pada CBC diperlukan data acak sebagai *block* pertama. Blok data acak ini sering disebut *initialization vector* atau IV. IV digunakan hanya untuk membuat suatu pesan menjadi unik dan IV tidak mempunyai arti yang penting sehingga IV tidak perlu dirahasiakan.

2.3.1.2.2.2 *Mode Cipher Feed Back* (CFB)

Pada mode CBC, proses enkripsi atau dekripsi tidak dapat dilakukan sebelum blok data yang diterima lengkap terlebih dahulu. Masalah ini diatasi pada mode *Cipher Feedback* (CFB). Pada mode CFB, data dapat dienkripsi pada unit-

unit yang lebih kecil atau sama dengan ukuran satu *block*. Misalkan pada CFB 8 *bit*, maka data akan diproses tiap 8 *bit*.

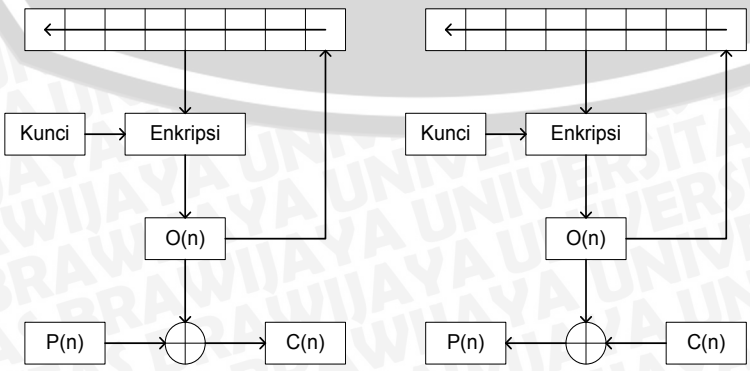


Gambar2.11 Mode Operasi CFB

Pada permulaan proses enkripsi, IV akan dimasukkan dalam suatu *register* geser. IV ini akan dienkripsi dengan menggunakan kunci yang sudah ada. Dari hasil enkripsi tersebut, akan diambil 8 bit paling kiri atau *Most Significant Bit* untuk di-XOR dengan 8 *bit* dari *plaintext*. Hasil operasi XOR inilah yang akan menjadi *ciphertext* dimana *ciphertext* ini tidak hanya dikirim untuk ditransmisikan tetapi juga dikirim sebagai *feedback* ke dalam *register* geser untuk dilakukan proses enkripsi untuk 8 *bit* berikutnya.

2.3.1.2.2.3 *Mode Output Feedback* (OFB)

Sama pada mode CFB, mode OFB juga memerlukan suatu *register* geser dalam pengoperasiannya. Pertama kali, IV akan masuk ke dalam *register* geser dan dilakukan enkripsi terhadap IV tersebut. Dari hasil proses enkripsi tersebut akan diambil 8 *bit* paling kiri untuk dilakukan XOR dengan *plaintext* yang nantinya akan menghasilkan *ciphertext*. *Ciphertext* tidak akan diumpan balik ke dalam *register* geser, tetapi yang akan diumpan balik adalah hasil dari enkripsi IV.



Gambar2.12 Mode Operasi OFB

2.4 Pengantar Matematis

Seluruh *byte* dalam algoritma *Rijndael* diinterpretasikan sebagai elemen *finite field* GF(2^8). Elemen *finite field* ini dapat dikalikan dan dijumlahkan, tetapi hasil dari penjumlahan dan perkalian elemen *finite field* sangat berbeda dengan hasil dari penjumlahan dan perkalian bilangan biasa.

2.4.1 *Finite Field* GF(2^8)

Pada *finite field* GF(2^8) 1 *Byte* terdiri dari 8 *bit* yang dapat dituliskan dengan $b_7, b_6, b_5,...,b_0$. dianggap *polynomials* dengan koefisien antara {0,1}.

Contoh : $63 = 01100011 = x^6 + x^5 + x + 1$.

2.4.1.1 ***Addition*** (Penjumlahan)

Penjumlahan dari dua elemen dalam suatu *finite field* dilakukan dengan menjumlahkan koefisien dari pangkat *polynomials* yang bersesuaian dari dua elemen tersebut. Penjumlahan dilakukan dengan operasi *XOR* dan dinotasikan dengan $\oplus$. Dengan operasi ini, maka $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, dan $0 \oplus 0 = 0$. Pengurangan dari *polynomials* identik atau sama dengan penjumlahan *polynomials*.

Sebagai alternatif, penjumlahan elemen-elemen pada *finite field* dapat dijelaskan sebagai penjumlahan *modulo* 2 dari *bit* yang bersesuaian dalam *byte*. Untuk 2 *byte* $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ dan $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, hasil penjumlahannya adalah $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$ dimana setiap $c_i = a_i \oplus b_i$. Contoh dari operasi penjumlahan adalah sebagai berikut :

$(x^6+x^4+x^2+x+1) \oplus (x^7+x+1) = x^7+x^6+x^4+x^2$ (notasi *polynomials*)

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (notasi *biner*)

$\{57\} \oplus \{83\} = \{d4\}$ (notasi *exadecimal*)

2.4.1.2 ***Multiplication*** (Perkalian)

Dalam representasi *polynomials*, perkalian dalam GF(2^8) yang dinotasikan dengan • mengacu pada perkalian modulo *polynomials* suatu *irreducible polynomials* yang berderajat 8. Suatu *polynomials* bersifat *irreducible* jika satu-satunya pembagi adalah dirinya sendiri dan 1. Untuk algoritma *Rijndael*, *irreducible polynomials* ini adalah :

$$m(x) = x^8 + x^4 + x^3 + x + 1 = \text{`11b'} \qquad (2.3)$$

atau dalam notasi heksadesimal adalah {01}{1b}. Sebagai contoh: {57}•{83} = {c1}, karena

$(x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) =$

$x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 =$

$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

dan

$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$ modulo $(x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$

Pengurangan *modulo* oleh *m(x)* memastikan bahwa hasilnya akan berupa *polynomials biner* dengan derajat kurang dari 8, sehingga dapat dipresentasikan dengan 1 *byte* saja.

2.4.1.3 *Devided* (Pembagian)

Pembagian dalam GF(2^8) dengan representasi *polynomials* antara *polynomials* g(x) dengan h(x) menghasilkan *polynomials* dimana g(x) dan r(x) dimana :

$g(x) = g(x)\ h(x) + r(x)$ $\qquad r(x) < h(x)$

g(x) adalah hasil bagi, r(x) adalah sisa bagi.

Contoh : 'GF' dibagi dengan '19' = x^2

$x^6 + x^5 + x^3 + x^2 + x + 1 = g(x)\ (x^4 + x^3 + 1) + r(x)$

$= x^2\ (x^4 + x^3 + 1) + r(x)$

$-r(x) = (x^2\ (x^4 + x^3 + 1)) - (x^6 + x^5 + x^3 + x^2 + x + 1)$

$r(x) = x^2\ (x^4 + x^3 + 1) + (x^6 + x^5 + x^3 + x^2 + x + 1)$

$r(x) = x^3 + x + 1$

dengan g(x) *modulo* h(x) adalah: $x^3 + x + 1$

2.5 Rijndael

Pada tahun 1972 dan 1974 *National Bureau of Standards* (sekarang dikenal dengan nama *National Institute of Standards and Technology,* NIST) menerbitkan permintaan kepada publik untuk pembuatan standar enkripsi. Hasil dari permintaan pada saat itu adalah DES (*Data Encryption Standard*), yang banyak digunakan di dunia. DES adalah sebuat algoritma *cryptography* simetris dengan panjang *key* 56 *bit* dan blok data 64 *bit*. Dengan semakin majunya teknologi, para *cryptografer* merasa bahwa panjang kunci untuk DES terlalu

pendek, sehingga keamanan algoritma ini dianggap kurang memenuhi syarat. Untuk mengatasi hal itu, akhirnya muncul *triple DES*.

Triple DES pada waktu itu dianggap sudah memenuhi syarat dalam standar enkripsi, namun teknologi yang tidak pernah berhenti berkembang akhirnya juga menyebabkan standar ini dianggap kurang memenuhi syarat dalam standar enkripsi. Akhirnya NIST mengadakan kompetisi untuk standar *cryptography* yang terbaru, yang dinamakan AES (*Advanced Encryption Standard*).

Persyaratan AES adalah :

- Algoritma harus dipublikasikan secara luas untuk diperiksa keamanannya.
- Algoritma haruslah merupakan simetris *Block Cipher.*
- Algoritma harus dapat diimplementasikan secara cepat dan tepat dalam *hardware* dan *software.*
- Algoritma harus memiliki input-an data 128*bit*.
- Algoritma memiliki kunci fleksibel : 128, 192 dan 256*bit.*

Dari hasil seleksi yang dilakukan oleh NIST, akhirnya NIST memilih 5 finalis AES, yaitu : *Mars* (IBM Amerika), *RC6* (RSA Corp, Amerika), *Rijndael* (Belgia), *Serpent* (Israel, Norwegia dan Inggris), dan *Twofish* (Counterpane Amerika). Kompetisi ini akhirnya dimenangkan oleh *Rijndael* dan secara resmi diumumkan oleh NIST pada tahun 2001. Sejak saat itu Algoritma *Rijndael* sering disebut juga dengan AES.

Rijndael diambil dari nama pembuatnya Dr. Vincent Rijmen dan Dr. Joan Daemen. *Rijndael* sendiri dipilih jadi pemenang bukan karena algoritma paling aman, namun karena memiliki keseimbangan antara keamanan dan fleksibilitas dalam berbagai *paltform software* dan *hardware.*

2.5.1 Representasi Data

Input dan output dari algoritma *Rijndael* terdiri dari urutan data sebesar 128 *bit*. Urutan data yang sudah terbentuk dalam satu kelompok 128 *bit* tersebut disebut juga sebagai blok data atau *plaintext* yang nantinya akan dienkripsi menjadi *ciphertext*. *Cipher key* dari *Rijndael* terdiri dari *key* dengan panjang 128 *bit*, 192 *bit*, atau 256 *bit*. Urutan bit diberi nomor urut dari 0 sampai dengan *n*-1

dimana *n* adalah nomor urutan. Urutan data 8 *bit* secara berurutan disebut sebagai *byte* dimana *byte* ini adalah unit dasar dari operasi yang akan dilakukan pada blok data (Wihartantyo, 2004).

Dalam algoritma *Rijndael*, data sepanjang 128 *bit* akan dibagi-bagi menjadi *array byte* dimana setiap *array byte* ini terdiri dari 8 *bit* data input yang saling berurutan. *Array byte* ini direpresentasikan dalam bentuk :

$a_0 a_1 a_2 ... a_{15}$

Dimana:

$a_0 = \{ input_0, input_1, ..., input_7 \}$

$a_1 = \{ input_8, input_9, ..., input_{15} \}$

$a_{15} = \{ input_{120}, input_{121}, ..., input_{127} \}$

$a_n = \{ input_{8n}, input_{8n+1}, ..., input_{8n+7} \}$

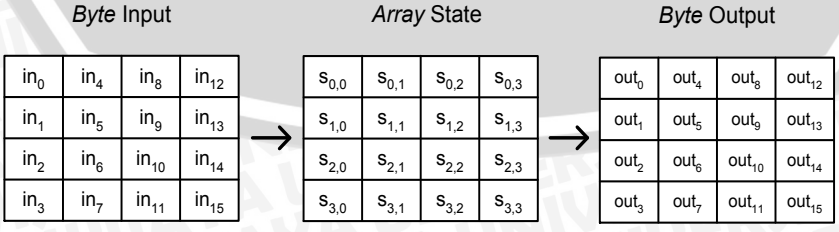
Operasi algoritma *Rijndael* dilakukan pada suatu *state* dimana *state* sendiri adalah suatu *array byte* dua dimensi. Setiap *state* pasti mempunyai jumlah baris yang tetap, yaitu 4 baris, sedangkan jumlah kolom tergantung dari besarnya blok data. Baris pada state mempunyai indeks nomor *row* (*r*) dimana $0 \leq r < 4$, sedangkan kolom mempunyai indeks nomor *column* (*c*) dimana $0 \leq c < Nb$. *Nb* sendiri adalah besarnya blok data dibagi 32.

Pada saat permulaan, input *bit* pertama kali akan disusun menjadi suatu *array byte* dimana panjang dari *array byte* yang digunakan pada *Rijndael* adalah sepanjang 8 *bit* data. *Array byte* inilah yang nantinya akan dimasukkan atau di-*copy* ke dalam *state* dengan urutan :

$s[r,c] = in[r+4c]$ *untuk* $0 \leq r < 4$ *dan* $0 \leq c < Nb$

sedangkan dari *state* akan di-*copy* ke *output* dengan urutan :

$out[r+4c] = s[r,c]$ *untuk* $0 \leq r < 4$ *dan* $0 \leq c < Nb$



Gambar2.13 *Byte Input, Array State,* dan *Byte Output*

Dari gambar 2.9 kita dapat membaca matrik tersebut dari atas kebawah, maka kita akan mempunyai *array* 1 dimensi 32 *bit* sebagai berikut :

$w_0 = out_0out_1out_2out_3$ $w_1 = out_4out_5out_6out_7$

$w_2 = out_8out_9out_{10}out_{11}$ $w_3 = out_{12}out_{13}out_{14}out_{15}$

Pada algoritma *Rijndael*, jumlah *block cipher*, *block output*, dan *state* adalah 128 *bit* . Dengan besar data 128 *bit* , berarti *Nb* = 4 yang menunjukkan panjang data tiap baris adalah 4 *byte* . Dengan *block cipher* atau *block* data sebesar 128 *bit* , *key* yang digunakan pada algoritma *Rijndael* tidak harus mempunyai besar yang sama dengan *block cipher*. *Cipher key* pada algoritma *Rijndael* bisa menggunakan kunci dengan panjang 128 *bit* , 192 *bit* , atau 256 *bit* . Perbedaan panjang kunci akan mempengaruhi jumlah *round* yang akan diimplementasikan pada algoritma *Rijndael* ini. Di bawah ini adalah tabel yang memperlihatkan jumlah *round* (*Nr*) yang harus diimplementasikan pada masing-masing panjang kunci.

Di bawah ini adalah tabel yang memperlihatkan jumlah *round* (*Nr*) yang harus diimplementasikan pada masing-masing panjang kunci.

	Jumlah Key (Nk)	Besar Block (Nb)	Jumlah Round (Nr)
AES – 128	4	4	10
AES – 192	6	4	12
AES – 256	8	4	14

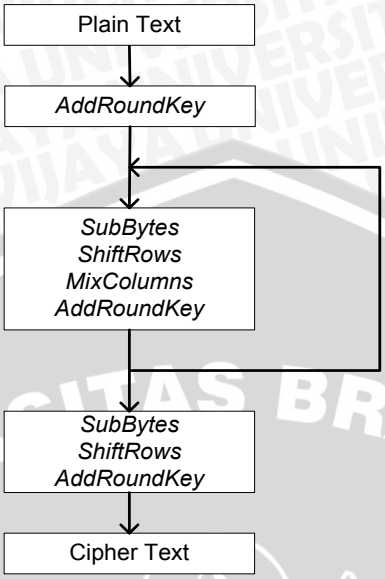
Tabel2.2 Perbandingan Jumlah *Round* dan *Key*

Dari tabel2.2 terlihat bahwa *Rijndael*- 128*bit* menggunakan panjang kunci $N_k = 4$ *word* yang setiap *word*-nya berisi 32*bit* sehingga total total kuncinya 128*bit*. Ukuran blok masukan 4 *word* (128*bit*), sedangkan jumlah *round*-nya (N_r) sebanyak 10 *round*. *Rijndael*-256 memiliki panjang kunci 256*bit*, blok masukan *plaintext* 128*bit*, dan jumlah *round*-nya 14.

2.5.2 Algoritma Enkripsi *Rijndael*

Proses enkripsi pada algoritma *Rijndael* terdiri dari 4 jenis transformasi *bytes*, yaitu *SubBytes*, *ShiftRows*, *Mixcolumns*, dan *AddRoundKey*. Pada awal proses enkripsi, input yang telah dikopikan 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 *Rijndael* disebut sebagai *round function*.

Round yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, *state* tidak mengalami transformasi *MixColumns*.



Gambar2.14 Diagram Alir Proses Enkripsi

Operasi enkripsi *Rijndael* dapat dinyatakan dengan kode semu (*pseudo code*) berikut ini :

```
Cipher (byte in [4*Nb], byte out [4*Nb], word w[Nb*(Nr+1)]) //nama fungsi
Begin
Byte state [4, Nb]
State = in                          //masukkan input
      AddRoundkey (state, w)
            For Nr = 1 to Nr -1
                  SubByte (state)
                  ShiftRows (state)
                  MixColoumns (state)
                  AddRoundkey (state, w + round *Nb )
            End
      SubByte (state)               //proses terakhir
      ShiftRows(state)
AddRoundkey (state, w)
Out = state                         //output
end
```

Dari *Pseudo code* diatas dapat kita menarik kesimpulan bahwa enkripsi dilakukan dengan fungsi *cipher* yang memiliki parameter masukkan in = 16 *byte*,

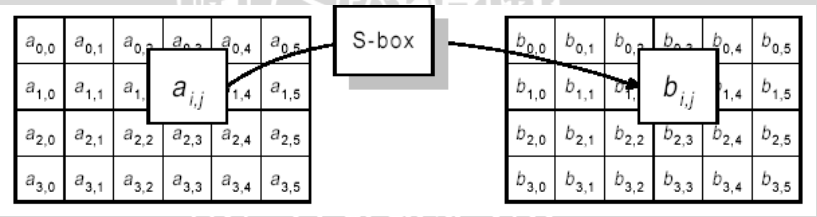
keluaran out = 16 *byte* dan *array* 1 dimensi w 44 *byte* untuk *Rijndael-128bit*. Proses yang dilakukan pada setiap *round* identik (dari Nr ke 0 sampai Nr-1), kecuali untuk *round* terakhir Nr. Proses yang identik itu terdiri dari *SubByte(), ShiftRows(), MixColoumns() and AddRoundKey()*. Sedangkan pada *round* terakhir(Nr) tidak dilakukan proses *MixColoumns()*.

2.5.2.1 *SubBytes*

SubBytes merupakan transformasi *byte* dimana setiap elemen pada *state* akan dipetakan dengan menggunakan suatu tabel substitusi (*S-Box*).

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Tabel2.3 Contoh *S-Box*



Gambar2.15 Proses *SubByte*

Hasil yang didapat dari pemetaan dengan menggunakan tabel *S-Box* ini sebenarnya adalah hasil dari dua proses transformasi *bytes*, yaitu :

1. *Invers* perkalian dalam GF(2^8) adalah fungsi yang memetakan 8 *bit* ke 8 *bit* yang merupakan *invers* dari elemen *finite field* tersebut. Suatu *byte* a merupakan *invers* perkalian dari *byte* b bila a•b = 1, kecuali {00} dipetakan ke dirinya sendiri. Setiap elemen pada *state* akan dipetakan pada tabel *invers*. Sebagai contoh, elemen "01010011" atau {53} akan dipetakan ke {CA} atau "11001010".

2. Transformasi *affine* pada *state* yang telah dipetakan. Transformasi *affine* ini apabila dipetakan dalam bentuk matriks adalah sebagai berikut :

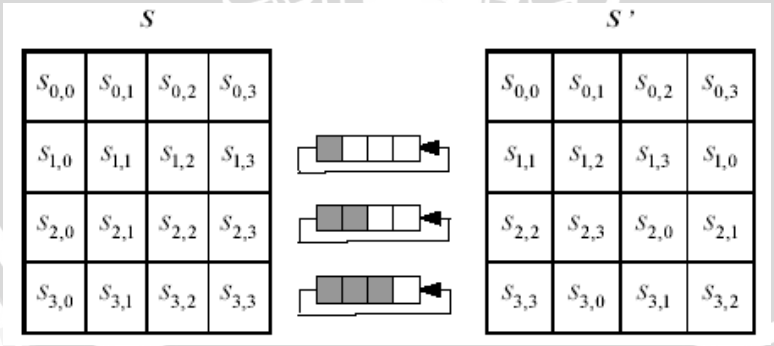
$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Gambar2.16 Matriks *Affine*

$b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0$ adalah urutan *bit* dalam elemen *state* atau *array byte* dimana b_7 adalah *most significant bit* atau *bit* dengan posisi paling kiri.

2.5.2.2 *ShiftRows*

Transformasi *Shiftrows* pada dasarnya adalah proses pergeseran *bit* dimana *bit* paling kiri akan dipindahkan menjadi *bit* paling kanan (rotasi *bit*). Transformasi ini diterapkan pada baris 2, baris 3, dan baris 4. Baris 2 akan mengalami pergeseran *bit* sebanyak satu kali, sedangkan baris 3 dan baris 4 masing-masing mengalami pergeseran *bit* sebanyak dua kali dan tiga kali.



Gambar2.17 Transformasi *ShiftRows*

2.5.2.3 *MixColoumns*

Mixolumns mengoperasikan setiap elemen yang berada dalam satu kolom pada *state*. Elemen pada kolom dikalikan dengan suatu *polynomials* tetap

a(x) = {03}x^3 + {01}x^2 + {01}x + {02}. Secara lebih jelas, transformasi *mixcolumns* dapat dilihat pada perkalian matriks berikut ini :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar2.18 Matriks Transformasi *MixColumns*

Melakukan proses penambahan pada operasi ini berarti melakukan operasi *bitwise* XOR. Maka hasil dari perkalian matriks diatas dapat dianggap seperti perkalian yang ada di bawah ini :

$$s'_{0,c} = (\{02\} \bullet s_{o,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

2.5.2.4 *AddRoundkey*

Pada proses *AddRoundKey*, suatu *round key* ditambahkan pada *state* dengan operasi *bitwise* XOR. Setiap *round key* terdiri dari Nb *word* dimana tiap *word* tersebut akan dijumlahkan dengan *word* atau kolom yang bersesuaian dari *state* sehingga :

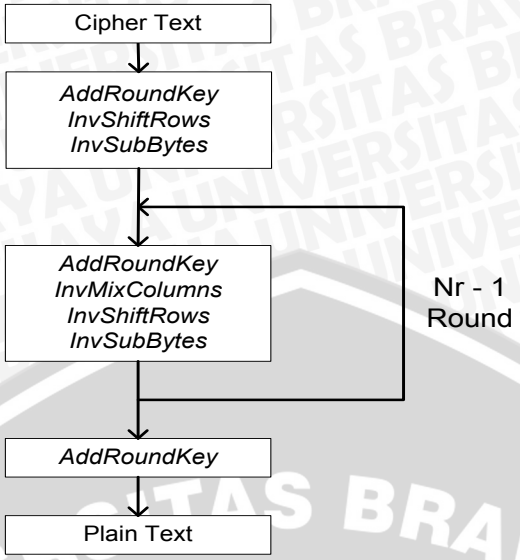
$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [w_{round*Nb+c}]$

untuk 0d c < Nb

Dimana [w_i] adalah *word* dari *key* yang bersesuaian dimana *i = round*Nb+c.* Transformasi *AddRoundKey* diimplementasikan pertama kali pada *round* = 0, dimana *key* yang digunakan adalah *initial key* (*key* yang dimasukkan oleh *cryptografer* dan belum mengalami proses *key expansion*).

2.5.3 Algoritma Dekripsi *Rijndael*

Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma *Rijndael*. Transformasi *byte* yang digunakan pada *invers cipher* adalah Inv*ShiftRows*, Inv*SubBytes*, Inv*MixColumns*, dan *AddRoundKey*.



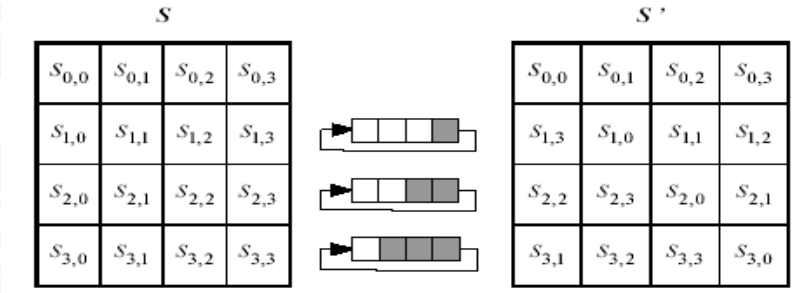
Gambar2.19 Diagram Alir Proses Dekripsi

Operasi dekripsi *rijndael* dapat dituliskan dengan *pseudo code* / kode semu beikut ini :

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]) // See Sec. 5.1

   for round = Nr-1 step -1 downto 1
      InvShiftRows(state)                    // See Sec. 5.3
      InvSubBytes(state)                     // See Sec. 5.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
      InvMixColumns(state)                   // See Sec. 5.3
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

2.5.3.1 Inv*ShiftRows*

InvShiftRows adalah transformasi byte yang berkebalikan dengan transformasi ShiftRows. Pada transformasi Inv*ShiftRows*, dilakukan pergeseran bit ke kanan sedangkan pada *ShiftRows* dilakukan pergeseran bit ke kiri. Pada baris kedua, pergeseran bit dilakukan sebanyak 3 kali, sedangkan pada baris ketiga dan baris keempat, dilakukan pergeseran bit sebanyak dua kali dan satu kali.



Gambar2.20 Transformasi *InvShiftRows*

2.5.3.2 Inv*SubBytes*

InvSubBytes juga merupakan transformasi bytes yang berkebalikan dengan transformasi SubBytes. Pada Inv*SubBytes*, tiap elemen pada *state* dipetakan dengan menggunakan tabel *inverse S-Box*. Tabel ini berbeda dengan tabel *S-Box* dimana hasil yang didapat dari tabel ini adalah hasil dari dua proses yang berbeda urutannya, yaitu transformasi *affine* terlebih dahulu, baru kemudian perkalian *invers* dalam GF(2^8).

$$\begin{bmatrix} b_7' \\ b_6' \\ b_5' \\ b_4' \\ b_3' \\ b_2' \\ b_1' \\ b_0' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Gambar2.21 Matriks Invers*Affine*

Perkalian *invers* yang dilakukan pada transformasi InvSubBytes ini sama dengan perkalian *invers* yang dilakukan pada transformasi SubBytes

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Tabel 2.4 Inverse *S-Box*

2.5.3.3 Inv*MixColumns*

Pada Inv*MixColumns*, kolom-kolom pada tiap *state* (*word*) akan dipandang sebagai *polynomials* atas GF(2^8) dan mengalikan *modulo* $x^4 + 1$ dengan *polynomials* tetap $a^{-1}(x)$ yang diperoleh dari :

$$a^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}.$$

Atau dalam matriks :

$$s'(x) = a(x) \otimes s(x)$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar2.22 Matriks Inv*MixColumns*

Hasil dari perkalian diatas adalah :

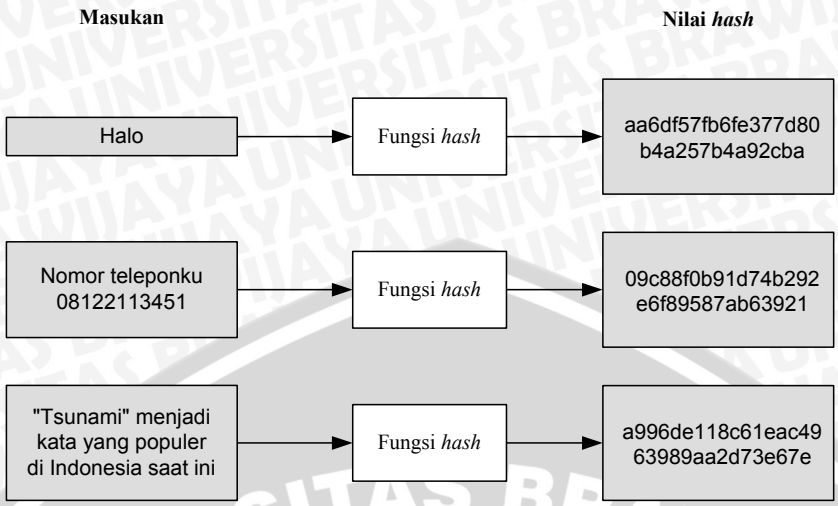
$$s'_{0,c} = (\{0E\} \bullet s_{o,c}) \oplus (\{0B\} \bullet s_{1,c}) \oplus (\{0D\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$
$$s'_{1,c} = (\{09\} \bullet s_{o,c}) \oplus (\{0E\} \bullet s_{1,c}) \oplus (\{0B\} \bullet s_{2,c}) \oplus (\{0D\} \bullet s_{3,c})$$
$$s'_{2,c} = (\{0D\} \bullet s_{o,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0E\} \bullet s_{2,c}) \oplus (\{0B\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{0B\} \bullet s_{o,c}) \oplus (\{0D\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0E\} \bullet s_{3,c})$$

2.5.3.4 *Inverse AddRoundKey*

Transformasi *Inverse AddRoundKey* tidak mempunyai perbedaan dengan transformasi *AddRoundKey* karena pada transformasi ini hanya dilakukan operasi penambahan sederhana dengan menggunakan operasi *bitwise* XOR.

2.6 Fungsi *Hash*

Fungsi *hash* adalah: fungsi yang menerima masukan *string* yang panjangnya sembarang, lalu mentransformasikannya menjadi *string* keluaran yang panjangnya tetap (*fixed*) (umumnya berukuran jauh lebih kecil daripada ukuran *string* semula).



Gambar2.23 Contoh *Hash*

Fungsi *hash* satu-arah (*one-way function*) adalah: fungsi *hash* yang bekerja dalam satu arah. Pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula (*irreversible*).

Persamaan fungsi *hash*:

$h = H(M)$

M = pesan ukuran sembarang

h = nilai *hash* (*hash value*) atau Pesan-ringkas (*message-digest*)

$h <<<< M$

Sifat-sifat fungsi *Hash* (H) satu arah adalah sebagai berikut (Munir, 2006):

1. Fungsi . H dapat diterapkan pada blok data berukuran berapa saja.
2. H menghasilkan nilai (h) dengan panjang tetap (*fixed-length output*).
3. H (x) mudah dihitung untuk setiap nilai x yang diberikan.
4. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga $H(x) = h$. Itulah sebabnya fungsi H dikatakan fungsi *hash* satu-arah (*one way hash function*).
5. Untuk setiap x yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga $H(y) = H(x)$.
6. Tidak mungkin mencari pasangan x dan y sedemikian sehingga $H(x) = H(y)$.
7. Ada beberapa fungsi *hash* satu-arah yang sudah dibuat orang, antara lain:

Beberapa contoh algoritma dari fungsi *Hash* satu arah : *MD2*, *MD4*, *MD5*, *Secure Hash Function* (*SHA*), *Snefru*, *N-hash*, *RIPE-MD*, *Tiger*, *Haval* dan lain-lain.

Algoritma	Ukuran (*bit*)	Ukuran Blok (*bit*)	Kolisi
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD-128	128	512	Ya
RIPEMD-160	160	512	Tidak
SHA	160	512	Ya
SHA-1	160	512	Cacat
SHA-256	256	512	Tidak
SHA-512	512	1024	Tidak
WHIRPOOL	512	512	Tidak

Tabel2.5 Perbandingan fungsi *Hash*

Manfaat dari aplikasi yang menggunakan fungsi *Hash* satu arah adalah sebagai berikut:

1. Menjaga integritas data
   - Fungsi *hash* sangat peka terhadap perubahan 1 bit pada pesan
   - Pesan berubah 1 bit, nilai *hash* berubah sangat signifikan.
   - Bandingkan nilai *hash* baru dengan nilai *hash* lama. Jika sama, pesan masih asli. Jika tidak sama, pesan sudah dimodifikasi.
2. Menghemat waktu pengiriman.
   - Misal untuk memverifikasi suatu salinan arsip dengan arsip asli.
   - Salinan dokumen berada di tempat yang jauh dari basisdata arsip asli
   - Ketimbang mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih mangkus mengirimkan *message digest*-nya.
   - Jika *message digest* salinan arsip sama dengan *message digest* arsip asli, berarti salinan arsip tersebut sama dengan arsip master.

3. Menormalkan panjang data yang beraneka ragam.
   - Misalkan *password* panjangnya bebas (minimal 8 karakter)
   - *Password* disimpan di komputer *host* (*server*) untuk keperluan otentikasi pemakai komputer.
   - *Password* disimpan di dalam basisdata.
   - Untuk menyeragamkan panjang *field password* di dalam basisdata, *password* disimpan dalam bentuk nilai *hash* (panjang nilai *hash* tetap).

2.7 Sekilas Tentang Java

Java adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. Java dikembangkan pertama kali oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naugton, Chris Warth, Ed Frank dan Mike Sheridan dari suatu perusahaan perangkat lunak yang bernama Sun Microsystems pada tahun 1990-an. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama ”Oak”, namun pada tahun 1995 diganti namanya menjadi ”Java”.

Alasan utama pembentukan bahasa Java adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control*, sehingga Java harus bersifat *portable* atau sering disebut dengan *platform independent* (tidak bergantung pada *platform*). Itulah yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah ”*Write once, run everywhere*”, yang berarti kode program hanya ditulis sekali namun dapat dijalankan dibawah *platform* manapun, tanpa harus melakukan perubahan kode program.

Kebanyakan bahasa komputer yang ada memiliki keterbatasan migrasi sistem yang berbeda. Java diciptakan sebagai sebuah bahasa baru dengan implementasi yang berbeda. Bahasa Java merupakan bahasa berorientasi objek yang diturunkan dari C++ dengan banyak penyempurnaan. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa Java memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk digunakan. Kini universitas-universitas di berbagai negara berpaling dari Pascal atau C++ kemudian memilih Java sebagai bahasa untuk belajar pemrograman.

2.7.1 Sejarah Perkembangan Java

Bahasa pemrograman Java pertama lahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot *Duke* yang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java Oak* pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan TV kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.

2.7.2 Arsitektur Java

Secara arsitektur, Java tidak berubah sedikitpun semenjak awal mula bahasa tersebut diliris. Kompiler Java (yang disebut dengan ***javac*** atau *Java Compiler*) akan mentransformasikan kode-kode dalam bahasa Java kedalam suatu *bytecode*. Apa itu *bytecode*? *Bytecode* merupakan sekumpulan perintah hasil kompilasi yang kemudian dapat dieksekusi melalui sebuah mesin komputer abstrak, yang disebut JVM (*Java Virtual Machine*). JVM juga sering dinamakan *interpreter*, karena sifatnya yang selalu menerjemahkan kode-kode yang tersimpan dalam *bytecode* dengan cara baris demi baris.

2.7.3 Versi-Versi Java

Versi awal Java ditahun 1996 sudah merupakan versi *release* sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

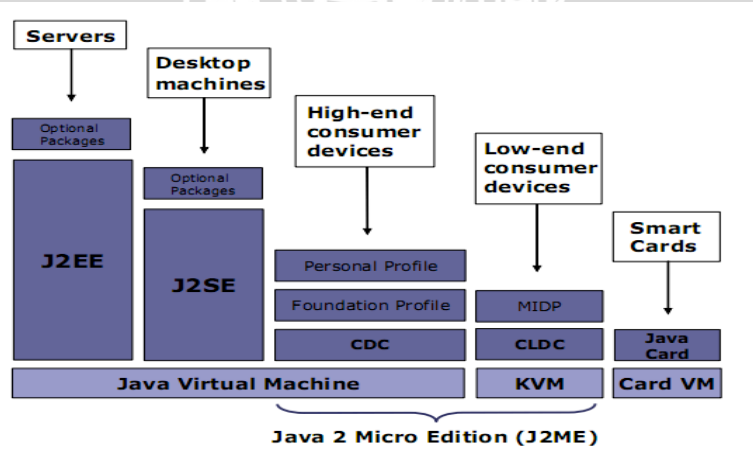
- `java.lang`: Peruntukan kelas elemen-elemen dasar.
- `java.io`: Peruntukan kelas *input* dan *output*, termasuk penggunaan berkas.
- `java.util`: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas kelas penanggalan.
- `java.net`: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
- `java.awt`: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI)
- `java.applet`: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

Pada awal perilisannya, versi Java masih disebut dengan JDK (*Java Development Kit*). Dalam JDK semua kebutuhan untuk untuk pengembangan program dan eksekusi program masih tergabung menjadi satu. Penamaan ini berlaku sampai dengan Java 1.1. Namun sekarang, setelah Java 1.2, Sun

Microsystems menamainya dengan JSDK (*Java Software Development Kit*) dalam hal ini kebutuha untuk pengembangan program dipisahkan dengan kebutuhan eksekusi. Bagian *software* yang digunakan untuk kebutuhan eksekusi program disebut dengan JRE (*Java Runtime Enviroinent*). Selanjutnya, Java 1.2 disederhanakan penamaannya menjadi ”Java 2”.

Sun Microsystems telah mendefinisikan tiga buah edisi dari Java 2, yaitu sebagai berikut:

- ✓ **Java *2 Standart Edition (J2SE***), yang digunakan untuk mengembangkan aplikasi-aplikasi *dekstop* dan *applet* (aplikasi Java yang dapat dijalankan didalam *browser web*) pada PC (*Personal Computer*).
- ✓ ***Java 2 Enterprise Edition (J2EE)***, merupakan *superset* dari J2SE yang meperbolehkan kita untuk mengembangkan aplikasi-aplikasi berskala besar (*enterprise*), yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi server dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi *web* dengan menggunakan *Sevlet* dan JSP (*Java Server Pages*) dan teknologi lainnya seperti CORBA dan XML.
- ✓ ***Java 2 Micro Edition (J2ME)***, merupakan *subset* dari J2SE yang digunakan untuk menangani pemrograman didalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh.



Gambar2.24 Lingkungan Kerja Teknologi Java

2.7.4 Kelebihan Java

- ***Multiplatform***. Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip *tulis sekali, jalankan di mana saja*. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk meninterpretasikan *bytecode* tersebut.
- **OOP** (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut *Object*. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi fungsi yang advance seperti komunikasi antara komputer sekalipun.

  Ada dua karakteristik yang utama pada sebuah objek, yakni:

  · Setiap objek memiliki atribut sebagai status yang kemudian akan disebut sebagai state .

  · Setiap objek memiliki tingkah laku yang kemudian akan disebut sebagai behaviour.
- **Perpustakaan Kelas Yang Lengkap**, Java terkenal dengan kelengkapan *library*/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya. Kelengkapan

perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

- **Bergaya C++**, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
- **Pengumpulan sampah** otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

Java sebagai bahasa pemrograman yang banyak disukai orang karena konsep pemrogramannya yang konsisten dengan teori orientasi objek serta aman untuk di gunakan, maka Java memiliki beberapa keunggulan :

- Sederhana
- Terdistribusi
- Netral Arsitektur
- Interpreter
- Multithreading
- Berorientasi Objek
- Aman
- Portable
- Powerfull
- Dinamis

2.7.5 Kekurangan Java

- **Tulis sekali, perbaiki di mana saja** - Masih ada beberapa hal yang tidak kompatibel antara platform satu dengan *platform* lain. Untuk J2SE, misalnya *SWT-AWT bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
- **Mudah didekompilasi**. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET Platform. Dengan demikian, algoritma

yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di*reverse-engineer*.

- **Penggunaan memori yang banyak**. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, Delphi dan Object Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena trend memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.7.6 Tahap Kompilasi Java

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.java'.
2. Kompilasi. Pengkompilasi membentuk *bytecodes* dari program menjadi bentuk berkas '.class'.
3. Muat. Pemuat kelas memuat *bytecodes* ke memori.
4. Verifikasi. Peng-verifikasi memastikan *bytecodes* tidak mengganggu sistem keamanan Java.
5. Jalankan. Penerjemah menerjemahkan *bytecodes* ke bahasa mesin.tidak bisa di pakai‘’

2.7.7 *Integrated Development Environment*

Banyak pihak telah membuat IDE (*Integrated Development Environment* - Lingkungan Pengembangan Terintegrasi) untuk Java. Yang populer saat ini (Juli 2006) antara lain:

- Dr. Java, program gratis yang dikembangkan oleh Universitas Rice, Amerika Serikat
- BlueJ, program gratis yang dikembangkan oleh Universitas Monash, Australia.
- NetBeans (*Open Source*- Common Development and Distribution License (CDDL))

  NetBeans disponsori Sun Microsystems, dan versi terkininya memilki Matisse, sebuah GUI Editor yang menurut pendapat umum merupakan yang terbaik.

- Eclipse JDT (*Open Source*- Eclipse Public License)
  Eclipse dibuat dari kerja sama antara perusahaan-perusahaan anggota 'Eclipse Foundation' (beserta individu-individu lain). Banyak nama besar yang ikut dalam 'Eclipse Foundation', termasuk IBM, BEA, Intel, Nokia, Borland. Eclipse bersaing langsung dengan Netbeans IDE. *Plugin* tambahan pada Eclipse jauh lebih banyak dan bervariasi dibandingkan IDE lainnya.
- IntelliJ IDEA (*commercial*, *free 30-day trial*)
- Oracle JDeveloper (*free*)
- Xinox JCreator (ada versi berbayar maupun *free*)
  JCreator ditulis dalam C/C++ sehingga lebih cepat (dan menggunakan memori lebih sedikit) dari kebanyakan IDE.

2.7.8 Komponen Java

2.7.8.1 JVM (*Java Virtual Machine*)

Java dapat berjalan pada sebuah sistem operasi membutuhkan *Java Virtual Machine* (JVM). JVM sendiri terdiri dari *Java Runtime Environment* (JRE) dan *Java Development Kit* (JDK). Sun Microsystems mengeluarkan tiga kelas paket Java, yaitu J2-SE JRE (hanya berisi JRE), J2- SE SDK (berisi JDK + JRE), dan J2-EE SDK (berisi JDK+JRE dan tools untuk aplikasi *enterprise*). Untuk versi SE (*Standard Edition*) tersedia gratis pada situs www.java.com.

2.7.8.2 IDE (*Integrated Development Environment*)

IDE (*Integrated Development Environment*) adalah sebuah editor pemrograman sebuah bahasa. Untuk Java sendiri ada banyak IDE yang tersedia dipasaran baik yang bersifat gratis (*freeware*) ataupun yang berbayar. Beberapa IDE yang populer antara lain, Jcreator (www.jcreator.com), Netbeans (www.netbeans.org), Jbuilder (www.borland.com/jbuilder), dan lain-lain.

2.7.8.3 Class

Unit yang paling mendasar dalam pemrograman java adalah class. Class adalah komponen aplikasi yang menangani kode dan data dalam pemrograman java.

2.8 Java Mobile

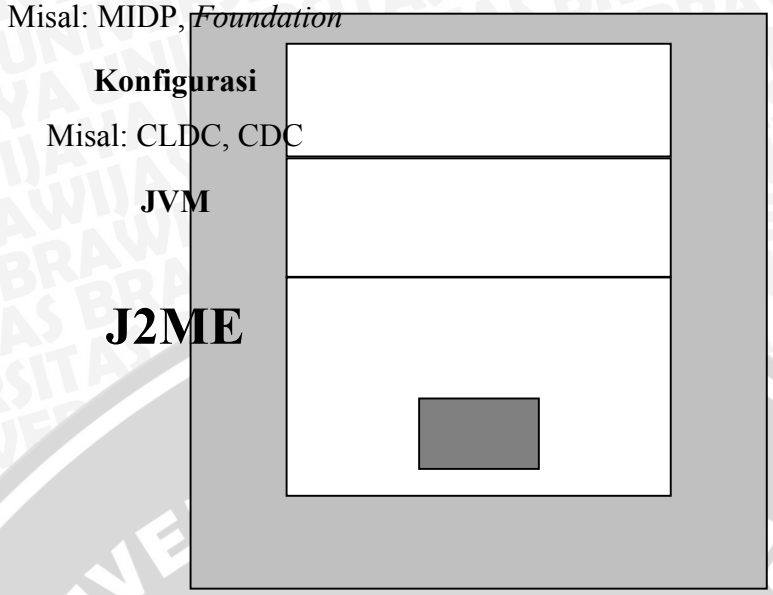
Java Mobile biasa dikenal dengan istilah J2ME (Java2 MicroEdition) merupakan salah satu bagian dari paket pemrograman Java. J2ME merupakan sebuah kombinasi yang terbentuk antara sekumpulan *interface* Java yang sering disebut dengan Java API (*Aplication Programig Interface*) dengan JTM *(Java Virtual Machine*) yang didesain khusus untuk alat, yaitu JVM dengan ruang yang terbatas. Kombinasi tersebut kemudian digunakan untuk melakukan pembuatan aplikasi-aplikasi yang dapat berjalan di atas alat (dalam hal ini *mobile device*)

J2ME merupakan superset dari J2SE, yang artinya Java API yang ada di J2ME sebagian diadopsi dari Java API J2SE. Jika pada J2SE menggunakan JVM (*Java Virtual Machine*) sebagai interprenternya, lain halnya pada J2ME yang menggunakan *Kilo Virtual Machine* (KVM) sebagai interprenternya. Berdasarkan spesifikasi perangkat kerasnya J2ME memiliki 2 macam konfigurasi yaitu CLDC (Connected Limited Device Configuration) dan CDC (*Connected Device Configurations*). Perbedaannya dapat dilihat pada tabel dibawah ini.

CLDC (Connected Limited Device Configuration	CDC (Connected Device Configuration)
Mengimplementasikan sebagian fitur dari J2SE	Mengimplementasikan seluruh fitur dari J2SE
Menggunakan KVM (Kilo Virtual Machine)	Menggunakan CVM
Digunakan pada handphone,PDA, Pager yang memiliki memori terbatas. (160-512) kb.	Digunakan pada perangkat internet TV, Nokia Communicator yang memiliki memori minimal 2 Mb.
Prosessor 16 / 32 bit	Prosessor 32 bit

Tabel2.6 Perbedaan CLDC dan CDC

J2ME sendiri pada dasarnya terdiri dari 2 buah bagian, yaitu: konfigurasi, profil, dan paket-paket opsional. Seperti yang ditunjukkna pada gambar dibawah ini.



Gambar2.25 Bagian-bagian didalam *platform* J2ME

Berikut ini penjelasan dari masing-masing bagian tersebut.

2.8.1 Konfigurasi

Konfigurasi merupakan bagian yang berisi JVM dan beberapa library kelas lainnya. Perlu diperhatikan bahwa JVM yang dimaksud disini bukanlah JVM tradisional seperti yang terdapat pada J2SE, melainkan JVM yang sudah didesain secara khusus untuk alat.

Terdapat dua buah konfigurasi yang disediakan oleh Sun Micrasystems, yaitu CLDC (*Connected Limited Device Configuration*) dan CDC (*Connected Device Configuration*). Target alat dari CLDC adalah alat-alat kecil, seperti telepon seluler, PDA, dan pager. Pada sisi lain, CDC merupakan superset dari CLDC sehingga semua yang kelas didefinsikan didalam CLDC akan ada juga dalam CDC.

Terdapat tiga buah paket dari J2SE yang didukung oleh CLDC, yaitu sebagai berikut:

- `java.lang`
- `java.io`
- `java.util`

Dengan kata lain, kelas-kelas dan *interface* lain yang terdapat pada J2SE akan dikeluarkan atau tidak semua diikutkan ke dalam J2ME, termasuk paket-paket penting seperti `java.awt` (untuk kebutuhan pengembangan aplikasi GUI – *Graphical User Interface*) dan `java.sql` (untuk kebutuhan koneksivitas dengan *database* melalui driver JDBC – *Java Database Connectivity*).

2.8.2 Profil

Profil merupakan bagian perluasan dari konfigurasi. Artinya, selain sekumpulan kelas yang terdapat pada konfigurasi, terdapat juga kelas-kelas yang didefinisikan lagi didalam profil. Dengan kata lain, profil akan membantu secara fungsional yaitu dengan menyediakan kelas-kelas yang tidak terdapat di level konfigurasi.

Adapun profil yang sangat populer penggunaannya adalah profile yang disediakan oleh Sun Microsystems, yaitu yang dinamakan dengan MIDP (*Mobile Information Device Profile*).

Beberapa profil yang tersedia untuk kebutuhan spesifik, antara lain:

- PDAP (*Personal Digital Assistant Profile*) yaitu profil untuk PDA yang memperluas fungsi-fungsi pada konfigurasi CLDC dan digunakan khusus untuk menambah kemampuan-kemampuan lebih apabila dibandingkan dengan penggunaan profil MIDP.
- *Foundation Profile*, yaitu profil yang digunakan untuk konfigurasi CDC. Profil ini menambah beberapa kelas dari J2ME ke dalam konfigurasi CDC, dan berperan juga sebagai pondasi untuk membentuk profil baru lainnya.
- *Personal Profile*, yaitu profil yang mendefinisikan ulang PersonalJava sebagai profil yang dapat digunakan sebagai profil dalam J2ME. Profil ini merupakan hasil perluasan dari *Foundation Profile.*
- RMI *Profile*, yaitu profil yang menambahkan dukungan RMI (*Remote Method Invocation*) ke dalam konfigurasi CDC

2.8.3 Paket-Paket Opsional

Paket-paket opsional merupakan paket-paket tambahan yang dibutuhkan oleh aplikasi sehingga pada saat proses *deployment* paket-paket tersebut perlu didisribusikan juga sebagai bagian dari aplikasi bersangkutan. Sebagai cacatan

bahwa paket-paket opsional ini buka merupakan paket yang dibuat oleh perusahaan alat yang digunakan.

2.9 MIDlet

Mungkin sebagian kita telah mengenal Applet sebagai aplikasi Java yang berjalan pada Web Browser di Internet. Sedangkan untuk aplikasi pada paket J2ME diberi nama MIDlet.

MIDlet adalah aplikasi yang dibuat menggunakan Java 2 Micro Edition dengan profile MIDP (*Mobile Information Device Profile*). MIDP dikhususkan untuk digunakan pada handset dengan kemampuan CPU, memori, keyboard, dan layar yang terbatas, misalnya pada handphone, PDA, Palm, Pocket PC, smart phone hingga PDA-Phone.

3.0 Sistem operasi Symbian

3.0.1 Sejarah

Pada awal mulanya di tahun 1998, symbian merupakan sebuah perusahaan patungan dari beberapa pemain di dunia ponsel yaitu Psion, Ericsson, Nokia, dan Motorola. Kemudian pada tahun 1999, Matsushita(Panasonic) bergabung kedalamnya. Pada tahun-tahun berikutnya banyak para pembuat ponsel yang mulai mengadopsi teknologi symbian. Tahun 2000, Sony dan Sanyo melisensi Symbian OS, tahun 2001 fujitsu dan siemens membeli lisensi juga. Tahun 2002 Samsung membeli lisensi, ditahun yang sama pula Siemens dan Sony Ericsson menjadi salah satu pemegang saha mnya. Samsung menjadi salah satu pemegang saham ditahun 2003. Di tahun 2004, Psion dan Motorola menjual sahamnya kepada nokia, sekarang Symbian dikuasai sebagian besar sahamnya oleh Nokia yang merupakan pemimpin didalam penjualan ponsel-ponselnya diseluruh dunia.
Ponsel pertama yang bersistem operasi Symbian OS adalah Ericsson R380 smartphone yang dikeluarkan tahun 2000. Dilanjutkan dengan dikeluarkannya dan diumumkannya ponsel-ponsel dengan sistem operasi symbian, seperti Nokia 9210 communicator dan Nokia 7650 di tahun 2001. Dilanjutkan dengan diumumkannya ponsel 800 buatan Sony ericsson yang mengadopsi symbian versi 7.0 ditahun 2002. Tahun 2003 mulai banyak ponsel-ponsel yang berbasiskan Symbian OS, diantaranya Nokia 3660, Nokia 3620, Nokia N-Gage, Nokia 6600, Nokia 7700, Sendo X, Siemens SX1, Sony Ericsson P900, BenQ P30, Foma

2102v, dan Motorola A920. Versi Sistem Operasi Symbian bermula dari dikeluarkannya ponsel Ericsson

R380 smartphone yang berbasis Symbian versi 5 yang merupakan versi unicode dari EPOC versi 5 dari Psion. Versi pertama yang menerapkan platform terbuka adalah Symbian OS versi 6.0 ditahun 2000 yang dipergunakan untuk ponsel Nokia 9210 dan 9290 Communicator. Kemudian dilanjutkan dengan versi 6.1 di awal tahun 2001 yang dipergunakan untuk ponsel Series 60 Platform yaitu Nokia 7650 dan 3650 imaging phones. Symbian OS versi 7.0 dikeluarkan pada tahun 2002 dan dipergunakan untuk ponsel Sony Ericsson P800 dan P900. Symbian OS versi 7.0s dikeluarkan tahun 2003 untuk ponsel-ponsel Nokia. Berikut merupakan daftar ponsel-ponsel yang berbasiskan sistem operasi Symbian.

3.0.2 Spesifikasi

Symbian OS adalah sistem operasi 32 bit, dengan konsep little endian dan berjalan pada beberapa tipe arsitektur mikroprosessor ARM. Symbian proses bekerja dengan prinsip preemptive multitasking. Dukungan terhadap device-device terintegrasi dalam kernel sebagai kernel extension yang ditulis dalam DLL (dynamic linking library) yang terpisah. Kernel berjalan dalam mode privileged dan memberikan servis ke aplikasi yang berjalan dalam mode unprivileged lewat user library. Symbian OS juga memberikan kumpulan-kumpulan library seperti networking (TCP/IP, PPP, FTP), Communication (Bluetooth, IrDA). Untuk mengakses servis-servis tersebut dengan menggunakan konsep hubungan client-server. Client menggunakan servis API yang diberikan oleh server untuk berkomunikasi dengan server. Semua hubungan komunikasi client-server diatur oleh kernel. Symbian OS memiliki beberapa kelebihan diantaranya sebagai berikut:

1. Small, kaya feature.
2. Platform terbuka untuk aplikasi-aplikasi third-party.
3. Konektifitas yang baik dengan perangkat lain.
4. Platform yang berkembang.
5. High performance, 32 bit OS dengan preemptive multitasking.
6. Long battery life.
7. Dukungan dan komitmen dari pembuat ponsel dunia.

8. Aplikasi yang dapat dirancang sekali dan berjalan pada beberapa device.

Banyak developer-developer yang mengembangkan aplikasi-aplikasi untuk sistem operasi ini, didasarkan karena beberapa pertimbangan, diantaranya sebagai berikut:

1. Symbian OS ditulis dalam C++, sistem operasi seluruhnya berbasis sistem object oriented sehingga flexible, efisien, reuseability, dan extendability.
2. API (Application Programming Interface) yang jelas, memudahkan membuataplikasi-aplikasinya.
3. Mempunyai multitasking dan manajemen memori yang efisien.
4. Proses berdasarkan event driven daripada multithreaded, sehingga menghemat memori untuk context switching.

Dalam masalah kehandalan, Symbian OS dirancang sedemikian rupa sehingga tidak terjadi kehilangan data dan device sangat jarang sekali reboot, karena Symbian OS mempunyai kemampuan sebagai berikut:

1. Mencegah terjadinya memori leak dengan manajemen memori yang efektif
2. Melepas sumber daya seketika sudah tidak digunakan lagi.
3. Menangani dengan baik error out of memory dengan error-handling framework yang efektif

Symbian OS mempunyai beberapa design sesuai dengan device family-nya atau yang disebut sebagai DFRD(Device Family Reference Design). DFRD ini merupakan spesifikasi dalam hal user interface dan hardware configuration, sehingga banyak tipetipe ponsel berbeda yang memanfaatkan sistem operasi ini, Macam-macam DFRD yaitu:

1. Crystal, untuk yang kaya feature seperti tipe Communicator, dimana mempunyai full keyboard, tampilan besar lcd mendatar dan menampung banyak informasi untuk kepentingan bisnis.
2. Pearl, untuk smartphone dimana seperti ponsel standar dengan lcd kecil dan keyboard yang terbatas.
3. Quartz, untuk ponsel dengan kemampuan pen-based atau touch-screen yang tidak memerlukan keyboard.

Sebagai sistem operasi untuk perangkat komunikasi bergerak, Symbian OS merupakan awal untuk ponsel masa depan. Berikut merupakan kemampuan-

kemampuan penting yang dimiliki Symbian OS yang dapat menangani kebutuhan akan masa depan:

1. Integrasi menyeluruh antar contact info, messaging,browsing dan telepon wireless.
2. Messaging (internet mail dengan POP3,IMAP4,SMTP,MHTML), standar attachment termasuk Microsoft word doc, Fax, Text messaging dengan SMS.
3. Protokol telepon bergerak (2G voice dan circuit-switched data, 2.5G packetswitched data, 3G, dan SMS).
4. Protokol komunikasi (TCP/IP,WAP,Bluetooth,IrDA,serial)
5. Security (enkripsi, secure protokol komunikasi termasuk HTTPS,WTLS dan SSL), certificate-base install aplikasi.
6. Engine aplikasi (contact,schedule,messaging,browsing,voice,office,utility dan system control)
7. Object exchange (OBEX untuk appointment dan business card)
8. Multimedia (mendukung beberapa format audio,video dan image).
9. Internasional lokalisasi (unicode karakter,handwriting recognition).
10. Programming dan content development (C++,Java,WAP dan web) Sinkronisasi dengan PC
11. Support beberapa tipe tampilan (keyboard base dan/atau pen-base)

3.0.3 Komponen

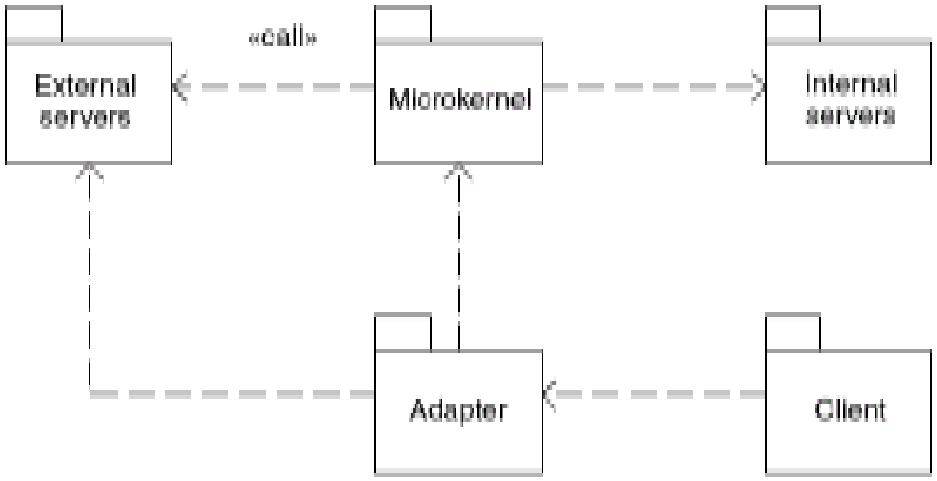
Symbian OS merupakan sebuah sistem operasi yang mengatur seluruh sumber daya yang ada didalam ponsel. Symbian OS disusun dari enam komponen, yaitu:

1. Kernel
2. Middleware
3. Application Engine
4. UI Framework
5. Synchronization
6. JVM

3.0.3.1 Kernel

Merupakan inti dari sistem operasi yang terdiri dari device driver, data table, dan program yang memungkinkan user berinteraksi dengan perangkat keras. Kernel merupakan program yang berjalan setiap waktu dan mengatur layanan-layanan

yang diberikan ke user. yang penting yang berada di kernel sedangkan fungsi yang lain ada dalam middleware, sehingga membuat kernel sangat ringkas dan arsitekturnya menjadi lebih modular. Dari awalnya Symbian OS merupakan sistem operasi 32-bit yang mendukung multitasking dan multithreading. Ukuran microkernel sekitar 5% dari keseluruhan sistem operasi, yang berkisar antara 500kB sampai 15MB tergantung pada ada tidaknya dukungan java dan aplikasi-aplikasi lain yang ikut diinstal. Pemisahan inti dan komponen lain membuat sistem sangat modular, yang akan meningkatkan portabilitas platform dan membuat proses upgrade dan perubahan platform lebih mudah dilakukan. Berikut merupakan arsitektur microkernel Symbian OS



Komponen microkernel menerapkan atomic service yang diperlukan keseluruhan aplikasi di sistem, mengontrol sumber daya seperti memori, proses, thread, dan IPC. Fungsi yang tidak mungkin dimasukkan ke kernel kerena alasan kompleksitas dan besar, dipisah ke internal server. Internal server mengembangkan fungsi inti, misalnya untuk menangani graphic dan media penyimpanan, dapat mempunyai proses sendiri atau share library. External server menggunakan servis dari microkernel dan internal server untuk memberikan servis ke client. External server menangani komunikasi (serial comm server,,socket server, message server, telephony server), graphics (window, font, bitmap server), audio(media server), storage media (file server). External server dieksekusi pada proses tersendiri, setiap external server memberikan client side API yang membungkus IPC , antara client dengan server. Sedangkan Adapter

memberikan interface yang transparan untuk client seperti detail komunikasi.

3.0.3.2 Middleware

Merupakan kumpulan library, data storage, dan program yang mengimplementasikan sistem servis. Kesemuanya itu tidak perlu diletakkan dalam kernel. Manajemen data, komunikasi dan graphics termasuk servis sistem tersebut. Sebagai contoh window system yaitu yang mengatur bagaimana user berinteraksi dengan perangkat keras, hal ini tidak cukup penting untuk diletakkan didalam kernel.

Symbian OS menggunakan server untuk implementasi middleware. Idenya adalah server yang dapat mengatur servis dari beberapa client dan merespon permintaanpermintaan tersebut. Dengan membuat sebuah layer baru untuk middleware, designer dapat dengan mudah merancang sistem servis baru tanpa harus merubah kernel.

3.0.3.3 Application Engine

Aplikasi user level dapat memanfaatkan servis yang diberikan pada level middleware yaitu dengan menggunakan application engine. Application engine yang melakukan koordinasi untuk mengakses sumber daya yang tidak begitu penting.

Application engine diantaranya adalah sebagai berikut:

Agenda engine, Contact engine, Sheet engine, Alarm server and WorldTime engine, Spell engine, dan Help engine.

3.0.3.4 User Interface Framework

User interface merupakan faktor utama untuk sebuah ponsel, yang menyebabkan ponsel mudah digunakan, mudah dirubah dan diprogram. Symbian OS menerapkan framework untuk user interface sehingga mudah diadaptasi. Dalam Symbian OS ada dua user interface dalam frameworknya yang menggunakan komponen yang umum seperti kontrol dan dialog yaitu Uikon dan Eikon.

3.0.3.5 Synchronization

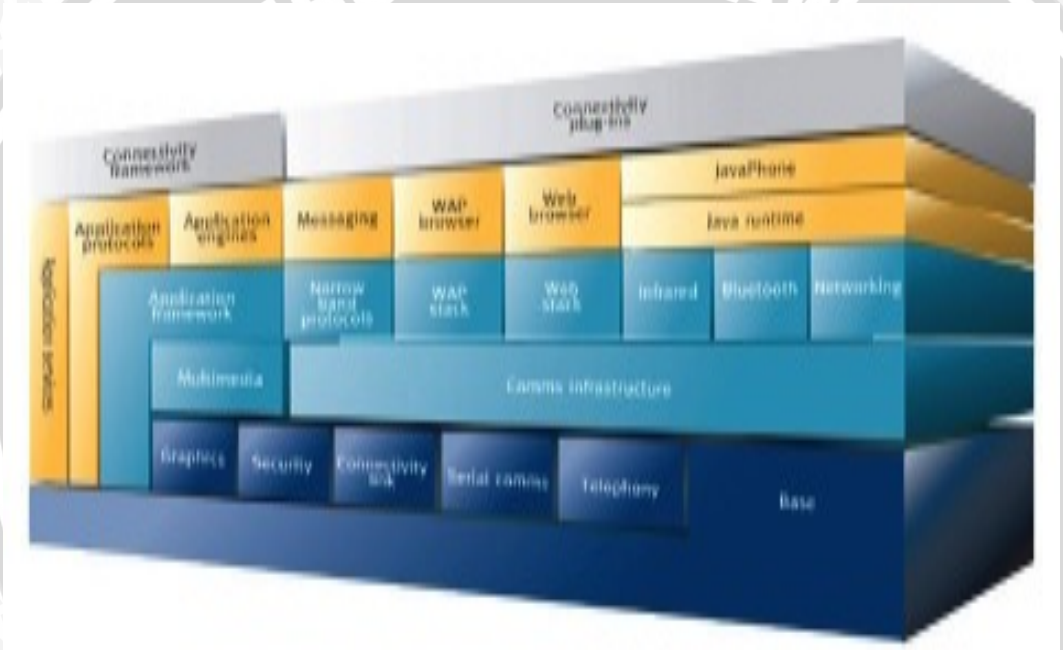
Sinkronisasi dengan peralatan lain ataupun dengan komputer desktop menjadi perhatian dalam teknologi ponsel. Symbian OS menerapkan 3 bagian untuk sinkronisasi yaitu Connection manager yang melakukan inisialisasi koneksi antar device dan mendeteksi jikalau ada device yang ingin melakukan koneksi.

Connectivity server yang mengimplementasi sesi sinkronisasi seperti file browsing, file sinkronisasi, back and restoration. File Converter yang melakukan transfer data antar format yang berbeda.

3.0.3.6 JVM

Symbian OS mengimplementasikan teknologi java yang dikenal dengan J2ME. JVM merupakan salah satu komponen dalam Symbian OS yang untuk perangkat ponsel dikenal dengan KVM (kilo virtual machine). JavaPhone dan PersonalJava yang merupakan bagian dari J2ME specification juga diimplementasikan dalam Symbian OS.

3.0.4 Arsitektur



Gambar diatas menunjukkan Arsitektur Symbian OS versi 6.x. Subsystem yang lebih atas mempunyai ketergantungan dengan beberapa subsystem yang dibawahnya, walaupun tidak selalu demikian. Sebagai contoh, WAP stack, Bluetooth, Infrared dan Narrow band merupakan protokol-protokol yang ada dalam comms infrastructure. Messaging juga bergantung pada WAP stack, tetapi java tidak bergantung dengan infrared.

Komponen-komponen dalam arsitektur diatas selengkapnya sebagai berikut:

1. Base, Terdiri atas Sistem runtime yang sangat dasar, low-level security.
2. Application framework, Terdiri atas API untuk manajemen data, text, clipboard, graphics, internationalization, dan inti komponen GUI.

3. Multimedia, Terdiri atas audio recording dan playback, fungsi-fungsi yang berhubungan dengan image.
4. Communication infrastructure and network stacks, Terdiri atas stack komunikasi yang luas termasuk TCP/IP,GSM,GPRS dan WAP. Komunikasi Personal seperti infrared, Bluetooth dan serial.
5. Messaging, Terdiri atas internet mail,SMS dan Fax.
6. Browsing, Terdiri atas WML dan HTML browsing engine.
7. Application protocols,service and engines, Terdiri atas engine untuk manajemen contact,schedule dan to-do list manajemen dan aplikasi-aplikasi yang lain. Java, Terdiri atas PersonalJava 3.0 spec JVM-based java runtime system dengan JavaPhone 1.0 APIs.
8. Connectivity, Terdiri atas converter dan viewer untuk format data foreign termasuk attachment mail Microsoft word. Framework komunikasi untuk berhubungan dengan PC.
9. Tools, Terdiri atas tool untuk membuat aplikasi, ROMs dan untuk debug target aplikasi.

3.0.4.1 Base

1. **Kernel dan User Library**

Komponen E32 menggabungkan kernel ekern.dll dan user library euser.dll. Kernel berjalan dalam mode privileged, memiliki device driver, melakukan manajemen daya, alokasi memori untuk dirinya dan mode user yang mempunyai proses unprivileged. Kernel berjalan secara dasar pada ARM microprocessor. User library memberikan sevis ke program user berupa:

1. Proses,thread,program dan menajemen memori.
2. Error handling dan cleanup framework.
3. Descriptors: string dan buffer data biner.
4. Class container: array dan lists.
5. Active object: untuk even driven multitasking tanpa memerlukan multithreading.
6. Client server architecture: untuk simple dan efisien IPC.
7. HAL (Hardware abstraction layer): memberikan interface yang konsisten untuk tipe perangkat keras yang berbeda.

8. Lokalisasi: currency,time dan date format.
9. Miscellaneous: seperti timer.

1. **Base peripherals**

File server yang mendukung VFAT, ROM dan Sistem file Flash (log Flash file system) dalam internal chip, CF card dan MMC card. VFAT file sistem dengan mode 'rugged' yang memberikan proteksi terhadap daya loss.

- **Security**

Modul dasar security adalah cryptography module dan certificate management module. Security termasuk algoritma standar cryptography, hash key generation, random number generation dan certificate management. Cryptography module termasuk didalamnya sbb:

1. Raw cryptography algorithm: untuk enkripsi dan dekripsi simetris seperti DES,3DES,RC2,RC4,RC5 dan asimetris seperti RSA,DSA,DH. Hash function: MD5,SHA,HMAC. Random number generator sebagai basis untuk key cryptography.

3.0.4.2 Application framework

- **Text**

Mendukung Unicode standar versi 3.0, semua 16-bit karakter unicode dapat digunakan dalam text. Semua karakter disimpan dan diambil sesuai aslinya, diimport dan diexport sebagai plain text. Mendukung perubahan warna text seperti pada PC. Gambar dapat diletakkan disamping text dengan pengaturan transparency dan background scrolling. Dua level undo/redo yang dapat disimpan. Mendukung perubahan size, shape, color dan blink rate kursor.

- **Internationalization**

Mendukung Chinese dan Japanese karakter untuk text input menggunakan handwriting recognition atau keyboard. Perubahan antara unicode dan karakter set lain melalui mekanisme plug-in, karakter yang diimplementasikan diantaranya: UTF-7, UTF- 8, modified UTF-7, modified UTF-8, 7 bit SMS (atau 7-bit GSM), Code Page 1252, dan ISO 8859-1. Nama file VFAT filename mendukung 8.3 name format.

- **Uikon**

Inti Komponen GUI Symbian OS termasuk didalamnya dialog framework,

concrete control, GUI environment, perubahan look and feel. GUI ini fungsi-fungsinya diberikan oleh EIKON dalam versi 5 Symbian OS.

3.0.4.3 Multimedia

Memberikan kemampuan recording, playback audio dan fungsi-fungsi yang berhubungan dengan image. Audio framework terdiri atas share library yang dapat melakukan pembacaan dan penulisan terhadap format audio yang umum seperti WAV, AU, WVE dan RAW dalam format yang berlainan. Image framework terdiri atas share library yang dapat melakuan pembacaan dan penulisan terhadap format-format gambar yang umum seperti JPEG, BMP, MBM and GIF (read only), WBMP (read only) dan Smart Messaging images (read only). Server interface memberikan plug-in yang generic, plug-in audio local dan telephony 8/16 bit PCM, ALaw, DTMF, tone dan tune generator. Format baru audio dan image dapat ditambah pada saat runtime dengan menambah plugin library-nya.

3.0.4.4 Communication infrastructure and network stacks

- **Networking**

Protokol TCP/IP untuk dapat melakukan koneksi lewet internet yang digunakan oleh aplikasi seperti email dan web.

- **GSM telephony & communications**

Telephony framework yang memberikan interface dasar untuk GSM voice, data dan fax. Symbian OS versi 6.1 juga mempunyai kemampuan mendukung GSM phase 2+ 11 SIM application toolkit, Class 3 (ETSI 11.14 R98), dengan kombinasi class ‘a’ (mobile phones yang hanya mendukung satu SIM) dan class ‘b’.

- **GPRS data communications**

Versi 6.1 memperkenalkan dukungan GPRS (General Packet Radio Service) class B phones. Dengan fungsi kelas B ini maka ponsel dapat melakukan hubungan telepon lewat GSM bersamaan dengan penggunaan GPRS, jika Paket data protokol aktif, servis GPRS akan otomatis suspend dan resume. Class B sekarang ini didukung oleh banyak jaringan GPRS.

- **WAP stack**

Peningkatan yang penting dalam versi 6.1 adalah dukungan untuk WAP 1.2.1,

fungsi push dan GPRS sebagai bearer. WAP stack mendukung protokol spesifikasi versi 1.1 dan 1.2.1 class C dari WAP Forum. WAP stack dapat menggunakan bearer GSM CSD dan GPRS UDP untuk koneksi browsing, GSM CSD, GPRS UDP, GSM SMS dan GPRS SMS untuk connectionless push. WAP stack mempunyai layer WSP (session protocol for WAP), WTP (transaction protocol for WAP), WTLS (transport layer security protocol for WAP) dan WDP (datagram protocol for WAP).

- **Bluetooth stack**

Bluetooth diimplementasikan sesuai spesifikasi versi 1.0 Bluetooth system architecture. Bluetooth stack mengimplementasi penuh Generic Access Profile, Serial Port Profile dan General Object Exchange Protocol. Stack terdiri atas protocol module, security manager, communications server module dan Service Discover Protocol server module.

- **Infrared**

Infrared IrDA stack berada dalam modul protokol soket server (irda.prt) yang menerapkan IrDA layer IrLAP v1.1, IrLMP v1.1 dan IrTinyTP.v1.1. Symbian OS Versi 6.0 dan yang terbaru, meningkatkan fungsi infrared dengan menambahkan feature seperti slow infrared (SIR) dengan throughput 9.6 Kbps to 115.2 Kbps, IrOBEX v1.0 (object exchange), IrTRANP v1.0 (varian dari fungsi GET/PUT IrOBEX). APIs IrCOMM v1.0 mendukung fungsi fax/modem yang diimplementasikan dalam serial communications server module.

3.0.4.5 Messaging

Messaging framework mendukung pengiriman dan penerimaan pesan SMS, email dan fax. Framework memanfaatkan polymorphic MTMs (message type modules) untuk menangani tipe pesan yang spesifik. Perubahan besar pada symbian 6.0 adalah penambahan watchers yang menangani pesan masuk dan BIO messaging yang mendukung pengiriman pesan ke system daripada ke user. Symbian versi 6.1 menambahkan dukungan GPRS dan 2D kompresi fax.

3.0.4.6 Browsing

- **Web engine**

Arsitektur web engine dibagi dalam beberapa komponen inti yaitu: Rendering engine yang bertanggung jawab untuk rendering dokumen dan menjaga struktur

pages. Services engine yang memberikan fungsi umum, seperti network status monitoring, bookmark, history list, proxy and authentication support. Web control yang bertanggung jawab menampilkan actual pages. Web engine mendukung HTTP/1.1 dalam RFC2068, juga dapat mengakses secure web
sites HTTPS dengan SSL 3.0 and TLS 1.0. Dengan Symbian OS Version 6.1, Web browser berjalan secara transparan melalui koneksi jaringan GPRS.

- **WAP browsing engine**

Browse engine mendukung WML 1.1 dan WML 1.2.1 juga WMLScript. Symbian OS Versi 6.1 juga mendukung PRE element. Engine tidak mendukung fungsi pilihan seperti access key sebuah elemen, mode connectionless HTTP ke WSP, WTA, vCard dan vCalendar, class attributes, fieldset.

3.0.4.7 Application protocols, services and engines

- **Application engines**

Yang menjadi bagian application engine adalah: agenda engine, contacts model, sheet engine, alarm and world server, spell engine dan help engine.

- **Application services**

Application service merupakan gabungan komponen-komponen yang digunakan oleh application engine. Ada beberapa servis, diantaranya: Task scheduler: menjadwal pemanggilan aplikasi atau melakukan inisialisasi feature application. System agent: memberikan informasi status mobile phone dan interface fisiknya. log engine: menyimpan penggunaan phone terutama telephony dan messaging. Alarm and world server: memberikan fungsi alarm , sound playing, country codes, world country and city information database.

- **Software installation**

Memberikan proses instalasi yang cepat dan mempunyai keamanan yang baik. Instalasi bisa berasal dari PC ataukah dari ponselnya dengan melakukan instal paket file dalam bentuk .sis. Paket dapat berupa aplikasi program, engine, dan user interface. Symbian OS mempunyai daftar aplikasi yang telah terinstal sehingga dapat dengan mudah untuk dilakukan proses reinstall ataupun uninstall

3.0.4.8 Java

- **Java**

Java pada Symbian OS versi 6.x merupakan implementasi PersonalJava

application environment specification 1.1.1a. Spesifikasi tersebut merupakan PersonalJava 3.0.x hybrid reference yang berdasarkan pada Java 1.1.6. Symbian OS memberikan implementasi lengkap PersonalJava, namun tanpa RMI dan JDBC.

- **JavaPhone**

Komponen JavaPhone memberikan kumpulan APIs yang mengembangkan runtime PersonalJava untuk mengakses fungsi native yang penting terutama telephony, agenda, contact dan power monitoring dan juga serial komunikasi. Symbian OS menerapkan JavaPhone 1.0 reference.

3.0.4.9 Connectivity

Onboard converters

Symbian OS Version 6.x memberikan fungsi converter antara Symbian OS and Windows formats untuk spreadsheet and word documents, dan dari Symbian OS Rich Text ke HTML.

- **Connection manager**

Mengatur koneksi antara PC dan Symbian ponsel. Koneksi dapat melewati serial, infrared dan pada versi 6.1 dapat digunakan bluetooth.

- **Symbian Connect**

Merupakan program symbian pada PC untuk melakukan komunikasi dengan Symbian OS, pada versi 6.1 dapat mendukung Microsoft Windows ME.

3.0.4.10 Tools

- **Software development**

Tool dasar untuk membuat program-program dengan C++ dan Java dan untuk membuat ROMs dengan tujuan mikroprosessor ponsel yang menggunakan ARM4 atau Thumb binaries yang efisien memproses unicode. Dapat pula membuat program untuk target ARMI binary format dimana dapat berjalan baik pada Thumb atau ARM4 ROMs

- **In-target debugging**

Pada Symbian OS Version 6.1 untuk melakukan debug pada target mesin dengan menggunakan GNU Debugger GDB. Debug ini hanya bisa untuk user-mode programs.

3.0.5 Manajemen Proses

Proses merupakan unit memori yang terproteksi, sedangkan Thread merupakan unit eksekusi. Satu proses tidak dapat mengakses secara langsung memori dari proses lain, kecuali dibuat memori yang global. Setiap aplikasi dan server di symbian mempunyai prosesnya sendiri-sendiri, tetapi user juga dapat membuat proses baru. Semua proses dapat mengakses share library dan sistem ROM. Dalam mode user, dapat dibuat 4 prioritas proses, yaitu:

1. EPriorityLow=150
2. EPriorityBackground=250
3. EPriorityForeground=350
4. EPriorityHigh=450

Prioritas tersebut digunakan untuk menghitung keseluruhan prioritas sebuah thread yang dibuat dalam proses. Developer dapat juga menggunakan nilai absolut prioritas dalam hal ini prioritas proses tidak digunakan. Ada 5 level prioritas yang tersedia untuk thread yang dieksekusi di mode user, yaitu:

1. EpriorityMuchLess
2. EpriorityLess
3. EpriorityNormal
4. EpriorityMore
5. EPriorityMuchMore

Kernel mempunyai prioritas tertinggi untuk proses dan thread. Thread dijadwalkan berdasar pada prioritasnya atau round robin jika thread mempunyai prioritas yang sama. Penjadwalan bersifat preemptive yang berarti yang mempunyai prioritas tinggi dapat menginterupsi thread yang lebih rendah. Pada beberapa kasus preemptive multitasking tidak diperlukan, dan dapat digantikan dengan multitasking yang cooperative yaitu prinsip active object. Seperti telah disebutkan diatas ada dua mode eksekusi yaitu user mode dan kernel mode. Kernel mode mempunyai prioritas yang tertinggi dibandingkan proses dan thread dalam user mode. Kedua mode tersebut diatur dalam dua library yaitu EUser dan EKern. Kedua library tersebut menyediakan kumpulan servis-servis untuk aplikasi. EUser digunakan untuk menangani proses dan thread, manajemen memori, active object, dan sebagainya. EKern digunakan untuk mengakses device driver dan sebagainya.

3.0.6 Manajemen Memori

Setiap proses harus mempunyai setidaknya satu thread. Proses yang kosong tidak dapat dieksekusi, tetapi thread dapat dieksekusi. Thread yang baru dibuat dengan fungsi Create(), dimana prototipenya sebagai berikut:

TInt Create (const TDesC& aName, TThreadFunction aFunction, TInt aStackSize, TInt aHeapMinSize,TInt aHeapMaxSize,
TAny *aPtr, TOwnerType aType=EOwnerProcess);

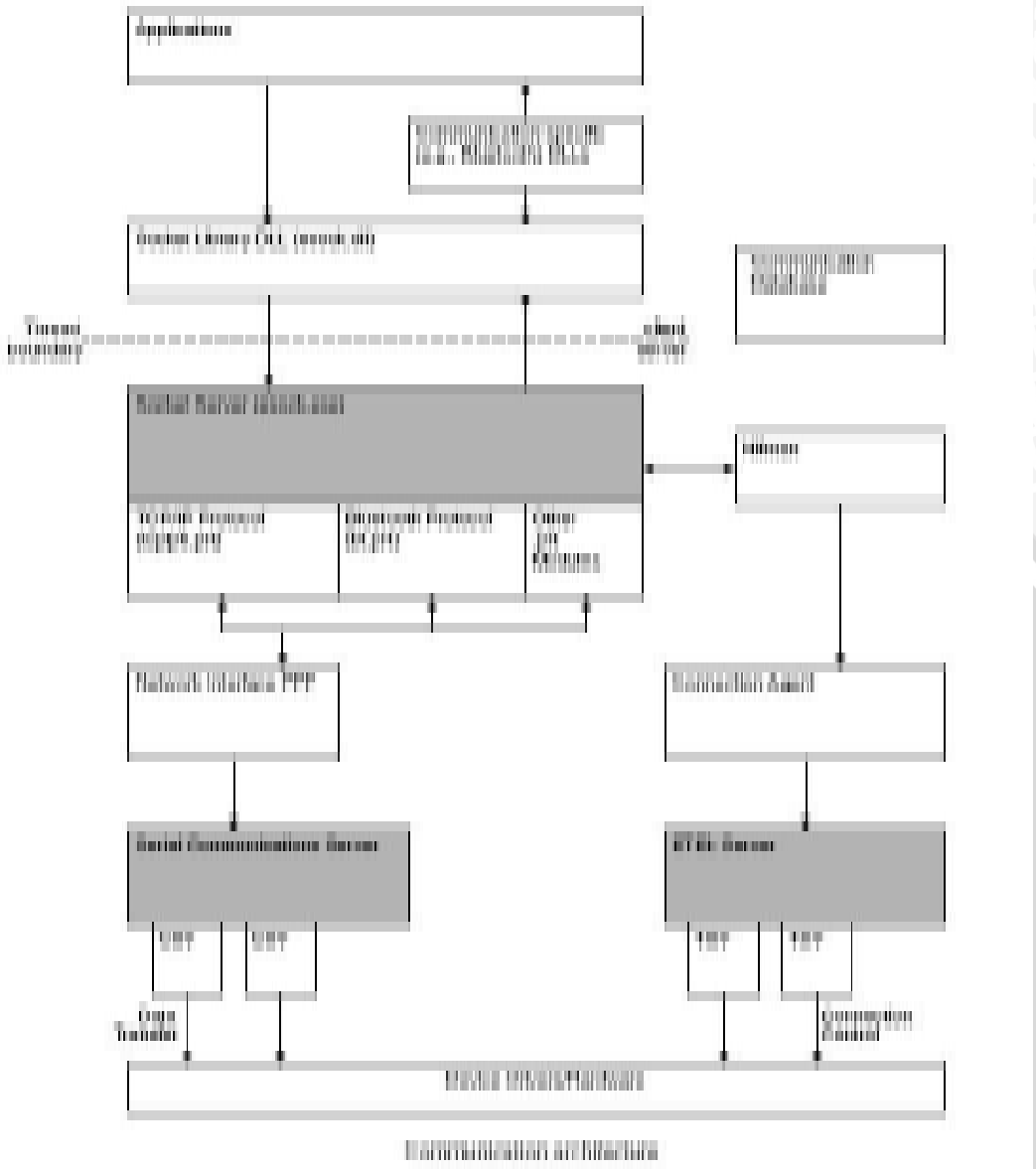
Untuk membuat thread diperlukan memori stack sebesar 8kB defaultnya, sedangkan untuk memori heap minimum 256Bytes sampai batas maksimum memori yang tersedia. Ketika thread dibuat, chunk memori baru dialokasikan untuk thread tersebut. Chunk merupakan area memori di virtual memori yang mempunyai alamat yang berdekatan. Secara fisik, chunk terdiri atas alokasi memori yang dibutuhkan tergantung pada arsitektur, 4kB jika pada arsitektur ARM. Bagian bawah dari chunk adalah stack dan diatasnya adalah heap. Stack selalu tumbuh kebawah, jadi tidak akan pernah mencapai memori heap. Heap dapat dishare antara thread dengan menggunakan versi fungsi Create() yang berbeda. Heap dan stack memainkan peranan yang berbeda dalam penyimpanan object yang dibuat dan data yang lain. Yang harus diperhatikan dalam penggunaan stack adalah stack mudah terjadi overflow. Semua alokasi di stack secara otomatis di hapus ketika tidak lagi dibutuhkan, namun untuk object yang dialokasikan di heap harus ada pointer untuk mengalokasikannya dan untuk menghapusnya. Class RHeap memberikan beberapa fungsi untuk menghitung banyaknya alokasi di heap atau menentukan keseluruhan yang teralokasi di heap. Semua object yang berada dalam heap harus mempunyai tipe class C, yang berarti berasal dari turunan kelas yang umum yaitu CBase. Semua yang berasal dari CBase dapat ditaruh dalam cleanup stack dan dihapus dengan fungsi PopAndDestroy(). Dapat juga digunakan User::Alloc() untuk melakukan alokasi di memori heap. Dikarenakan heap tidak secara otomatis dilakukan proses penghapusan, maka ada resiko heap akan menyebabkan memori leak. Untuk itu dalam Symbian OS digunakan cleanup stack untuk menyimpan variabel secara otomatis menangani penghapusannya sehingga tidak terjadi memori leak.

3.0.7 Manajemen I/O

Dalam permasalahan input dan output, baik itu berupa keyboard input, pointer input, dan I/O request yang lain, semua berjalan dalam mode asynchronous yang diberikan oleh asynchronous service providers. Symbian OS memberikan mekanisme yang mudah untuk menggunakan asynchronous service tersebut. Ada 2 level asynchronous yaitu: Low-level dan High-level asynchronous. Pada low-level, asynchronous service provider memberikan servis dengan fungsi request dan cancel. Status request akan disimpan kedalam request status, dan service provider akan memberikan signal bahwa request telah selesai dengan menggunakan thread request semaphore. Pada high-level, menggunakan active scheduler yang melakukan wait loop yang digunakan dalam multiple asynchronous, dan active object yang melakukan fungsi request dan cancel, sekaligus menangani penyelesaian permintaan.

3.0.8 Komunikasi

Ponsel tidak akan bermakna tanpa ada pertukaran informasi, untuk melakukan hal tersebut diperlukan suatu teknologi komunikasi, baik berupa teknologi telephony maupun teknologi pertukaran data yang lain. Struktur microkernel Symbian OS mempunyai efek terhadap arsitektur komunikasi. Service komunikasi harus melewati sistem server, yang melakukan penambahan atau pengurangan tergantung perangkat keras yang didukung oleh smartphone. Arsitektur komunikasi terdiri atas server komunikasi dan modul tambahan. Modul tambahan dapat ditambah dan dibuang kapanpun pada saat runtime tanpa harus melakukan rebooting OS. Berikut ini adalah arsitektur komunikasinya.



3.0.8.1 Server komunikasi

Server komunikasi dalam Symbian OS memberikan client-side API, ada empat server komunikasi, yaitu:

1. Server serial komunikasi (C32) yang memberikan interface umum untuk komunikasi serial.
2. Server soket (ESock) yang memberikan interface umum untuk komunikasi endpoint yang lebih dikenal dengan soket.
3. Server Telephony (ETel) yang memberikan interface umum untuk inisialisasi, kontrol dan memutuskan telephone call.
4. Server Message (MTMs) yang memberikan akses ke data message.

Penggunaan server komunikasi tergantung pada aplikasi, jika aplikasi menghendaki pengiriman bit sederhana maka dapat digunakan serial server. Serial server menggunakan protokol dependen yang mempunyai ekstensi .csy. Hal ini yang membedakan antara modul protokol RS-232 dengan infrared. CSY mempergunakan device driver di kernel untuk mengakses perangkat keras. Device driver pun merupakan modul tambahan yang dapat di load dan unload kapanpun.

3.0.8.2 Modul komunikasi

Device driver dibagi menjadi dua bagian dalam Symbian OS yaitu physical dan logical device driver untuk meningkatkan modularity dan reusability. PDD (physical device driver) berhubungan langsung dengan perangkat keras, sedangkan LDD (logical device driver) bertanggung jawab terhadap buffer data, control flow, DFC (delay function call) dan interrupt handling. Interupsi device ditangani dalam dua tahap, ketika interupsi terjadi, pertama akan menjalankan servisrutin yang akan mengenali device kemudian akan men-set flag di kernel untuk memanggil DFC. Selanjutnya DFC akan dipanggil dalam user mode. Untuk serial komunikasi dalam symbian, PDD-nya adalah euart1.pdd an LDDnya adalah ecomm.ldd. Perubahan hanya terjadi pada protokol modul yaitu ecuart.csy untuk RS-232, ircomm.csy untuk infrared, dan btcomm.csy untuk bluetooth. Protokol digunakan sebagai fungsi untuk error detection, error correction, efisiensi dan adaptasi flow control, dan mendukung beberapa koneksi simultan. Implementasi protokol dalam Symbian OS di akses melalui soket server ESock, yang mirip dengan interface BSD soket. Protokol high-level mulai dari network hingga application layer di terapkan dalam modul terpisah yang dinamakan protocol modules dan mempuyai ekstensi .prt, misalkan tcpip.prt adalah modul protokol TCP/IP, irda.prt adalah modul protokol IrDa, bt.prt adalah modul protokol Bluetooth dan wapprot.prt adalah untuk WAP. Berikut ini merupakan gambar level modul komunikasi dan modul protokol dalam Symbian OS.

Modul komunikasi untuk telepon adalah TSY modul yang memberikan standar fungsi telepon seperti establishing, controlling dan terminating call. Modul TSY digunakan oleh server telephony ETel. Modul phonetsy.tsy yang memberikan semua service untuk keperluan telephony. Selain standar juga ada modul yang lain seperti hayes, gsmbsc, dan gprstsy. Modul komunikasi untuk menangani

creation, sending, receiving dan editing message adalah MTMs. Modul message ini dapat menangani email message seperti (SMTP, POP3 dan IMAP4), dapat pula menangani FAX, SMS dan MMS. Untuk mengirimkan message dalam aplikasinya digunakan CSendAs class yang melakukan pengelompokan data message tanpa menggunakan interface MTM secara langsung. Object hasil dari CSendAs akan membuat session ke message server dan mengambil registry MTM sehingga dapat memanfaatkan servis-servis-nya.

3.0.8.3 Keamanan komunikasi

Ada tiga jenis ukuran keamanan dalam berkomunikasi yaitu confidentiality, integrity dan availability. Confidentiality berarti tidak ada data yang bisa didapat oleh orang yang tidak berkepentingan. Integrity berarti tidak ada data yang bisa diubah oleh orang yang tidak mempunyai hak akses. Availability berarti data dan service selalu tersedia untuk yang menginginkannya, tidak boleh terjadi serangan denial of service. Mekanisme proteksi dapat dilakukan dengan beberapa cara, diantaranya untuk meningkatkan tingkat confidentiality dapat dilakukan dengan melakukan enkripsi, walaupun tidak ada algoritma enkripsi yang tidak bisa dipecahkan, tetapi tingkat kesulitan memecahkannya yang perlu dipertimbangkan. Untuk meningkatkan integrity, dapat digunakan message certificate seperti digital signature. Availability dapat ditingkatkan dengan pengecekan keaslian dengan pengecekan password.

Symbian OS mendukung berberapa algoritma enkripsi diantaranya yaitu RSA, DES, 3DES,RC2, RC4, SHA-1,MD5, dan Diffie-Hellman. Keamanan dalam tingkatBprotokol, dapat digunakan secure shell (SSH) untuk remote system yang akan melakukan enkripsi password dan data. Dalam WAP (wireless application protocol) diberikan layer terpisah untuk keamanan yaitu WTLS (Wap transfer layer security). Ketika menggunakan soket, dapat dipergunakan TLS (transfer layer security) dan SSL (secure socket layer). Dalam tool Symbian OS, ada generator untuk membuat private-public key yang merupakan asymetric cryptography dan dapat mengeluarkan permintaan certificate. Private key dipakai untuk tanda digital instalasi file, sehingga system installer dapat mengenalinya. Proses instal yang aman pertama dengan mengecek tanda pada file yang telah terinstal dengan menggunakan public key dalam certificate developer untuk

meyakinkan bahwa paket instalasinya telah tertanda private key developer. Kemudian mengecek tanda pada certificate developer dengan organisasi public key dari organisasi certificate untuk meyakinkan bahwa pasangan kunci tersebut adalah kepunyaan seseorang yang ada dalam certificate tersebut.

Ada permasalahan dalam keamanan, bahwa tidak akan ada metode keamanan yang berguna jika salah satu software tidak menerapkannya. Misalnya dalam penggunaan protokol bluetooth untuk koneksi ke service provider yang tidak menerapkan keamanan, maka dapat terjadi data-datanya tidak bisa dipercaya keasliannya. Dalam symbian untuk mengecek keabsahan suatu password dari input external harus dicek isinya dan besarnya dengan menggunakan software, dapat diterapkan dengan menggunakan ASSERT macro.

BAB III
METODOLOGI PENELITIAN

Dalam penyusunan skripsi ini, dirancang suatu aplikasi yang dapat mengirim dan membaca MMS yang hanya bisa terbaca oleh orang yang mempunyai *account* (pengirim dan penerima) melalui proses penyandian kriptografi rijndael menggunakan program Java. Metode penelitian yang digunakan pada penyusunan skripsi ini adalah:

3.1 Studi Literatur

Mempelajari literatur-literatur atau buku serta dokumen-dokumen yang berhubungan dengan algoritma *rijndael* enkripsi dan dekripsi, multimedia serta *object-oriented design* dan literatur-literatur terkait.

3.2 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Metode analisis yang akan digunakan adalah *Object-Oriented Analysis* yaitu menggunakan bahasa algoritma yang di aplikasikan melalui program java, perangkat lunak yang dibangun diharapkan dapat menghasilkan solusi yang optimal, mudah diimplementasikan oleh pengguna, dan menghasilkan solusi optimal dengan cepat serta dapat menghasilkan kenyamanan dalam sebuah pengiriman sebuah MMS.

Perangkat Keras :

1. PC atau Laptop
2. Handphone dengan platform java yaitu tipe Samsung monte

Perangkat Lunak :

1. SDK
2. Netbeans
3. Emulator J2ME

Adapun kebutuhan yang harus dipenuhi agar pengguna dapat menggunakan aplikasi yang dibuat antara lain:

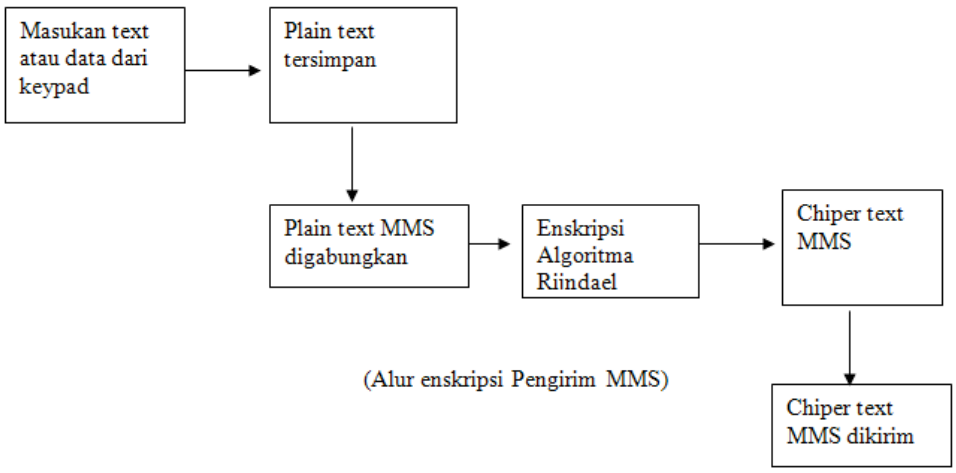
1. Memiliki kemampuan fitur aplikasi MMS pada umumnya untuk melakukan pengiriman ,
   penerimaan dan penyimpanan pesan
2. Aplikasi dapat melakukan proses enkripsi dan dekripsi dengan algorithma Rijndael

3. Memiliki antarmuka menu yang dapat dimengerti oleh pengguna dan memiliki tingkat keamanan

3.3 Perancangan

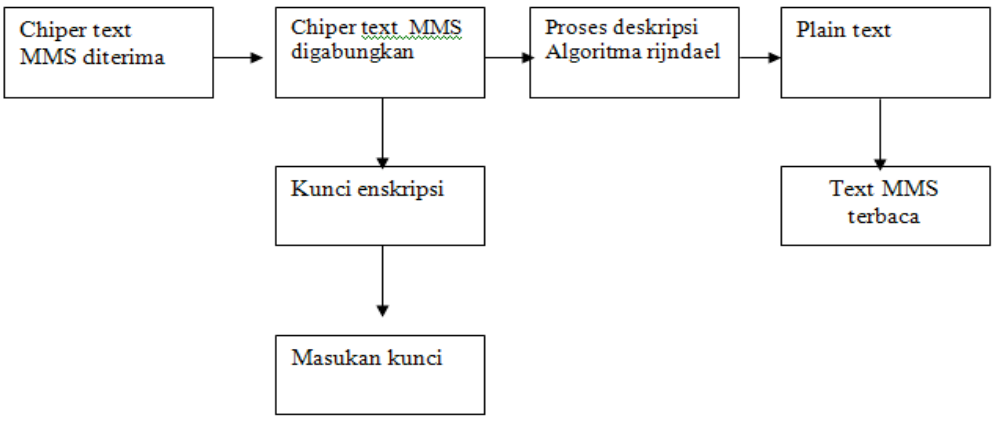
Perancangan aplikasi dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan. Perancangan algoritma deskripsi dan enkripsi

3.4 Alur Sistem



(Alur enskripsi Pengirim MMS)

(Alur deskripsi receiver MMS)



3.5 Perancangan Secara Umum

Menjelaskan proses perancangan secara umum untuk mengimplementasikan kedalam J2ME dengan tahapan perancangan sebagai berikut

1. Perancangan antarmuka aplikasi yaitu mendesain aplikasi agar sesuai dengan perangkat mobile dan bahasa pemrograman yang dipakai.
2. Perancangan implementasi algorithma Rijndael berupa perhitungan komponen wordbit , putaran dan panjang kunci yang dibutuhkan untuk diterapkan pada aplikasi.
3. Perancangan perangkat lunak agar dapat melakukan proses menulis , membaca pengiriman dan penerimaan pesan yang dibutuhkan aplikasi.

3.6 Implementasi

Implementasi dilakukan dengan mengacu kepada perancangan perangkat lunak. Pada proses implementasi, perancangan dilakukan dengan menggunakan bahasa Java 2 Micro Edition.

3.7 Pengujian

Pengujian dilakukan untuk menjamin dan memastikan bahwa sistem yang telah dirancang memiliki tingkat kesalahan yang minimal. Untuk pengujian perangkat lunak dibuat agar tidak terjadi kegagalan dari alur logika yang berjalan. Pengujian dilakukan dengan parameter input dari keypad berupa plaintext dan kunci untuk menentukan hasil enkripsi berupa panjang chipertext yang akan dibandingkan dengan plaintext dan waktu yang diperlukan untuk proses enkripsinya , kemudian tingkat keberhasilan dekripsi dan waktu yang diperlukan untuk proses dekripsi dengan parameter kunci dan chipertext.

3.8 Kesimpulan dan Saran

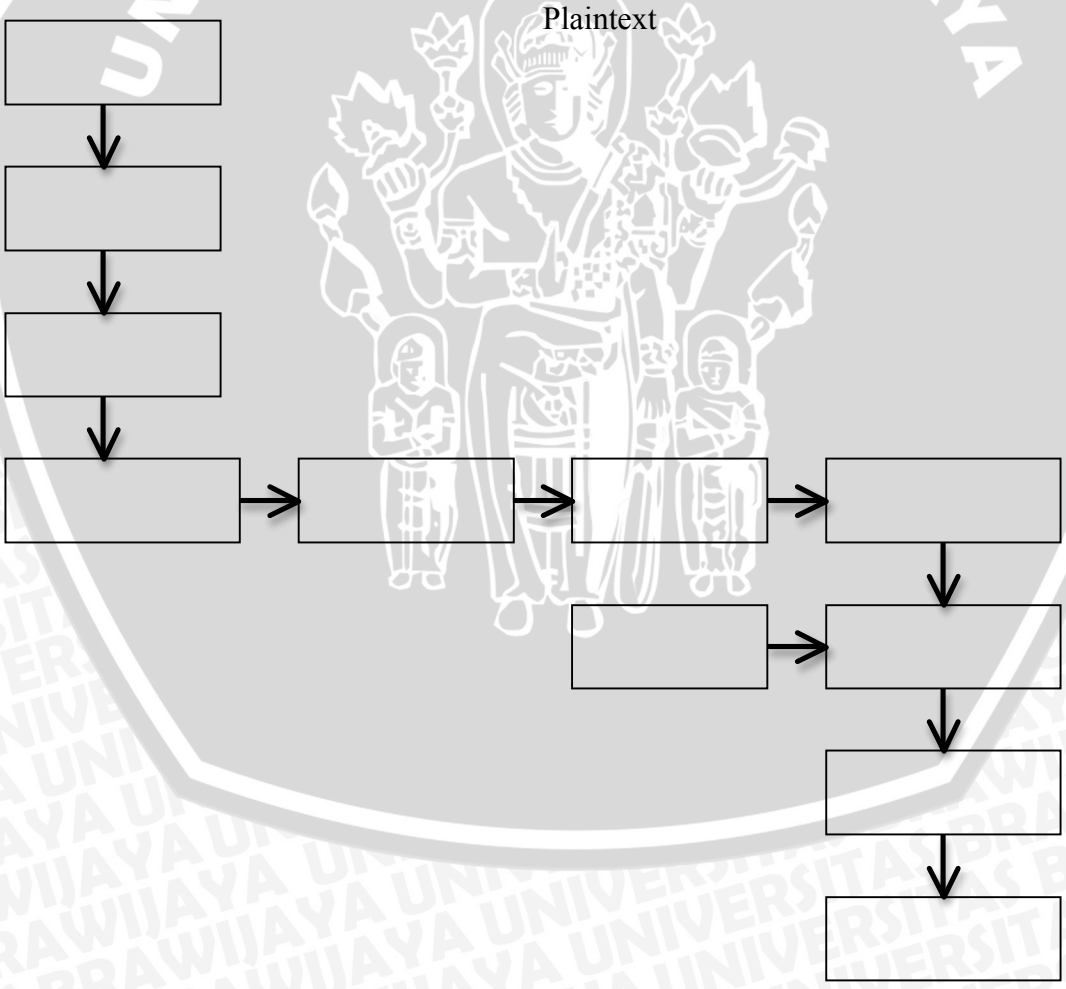
Pada tahap ini, diambil kesimpulan dari hasil pengujian dan analisis terhadap rancangan kriptografi rijndael enskripsi dan deskripsi MMS menggunakan J2ME, Untuk Menentukan kemampuan tingkat keberhasilan pengkodean rahasia dan *Programming pada kriptografi rijndael*. Tahap selanjutnya adalah pembuatan saran untuk perbaikan terhadap penelitian selanjutnya sehingga dapat menyempurnakan kekurangan-kekurangan yang ada.

BAB IV
PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas mengenai perancangan perangkat lunak untuk proses pengiriman pesan MMS kriptografi dengan menggunakan Algoritma Rijndael meliputi enkripsi , dekripsi , pengiriman pesan dan penerimaan pesan, untuk memudahkan pemahaman cara kerja sistem.

4.1 Perancangan Secara Umum

Perancangan sistem secara umum merupakan tahap awal sebagai acuan dalam perancangan aplikasi yang akan dibuat. Perancangan ini didahului dengan pendefinisian kegiatan pelaku atau user dalam menggunakan program MMS untuk mengirimkan dan menerima pesan menggunakan teknik kriptografi serta perangkat yang digunakan meliputi blok sistem diagram dan cara kerja sistem.



Gambar 4.1 Diagram Blok Kriptografi MMS

Pengirim sebagai user menuliskan pesan rahasia pada ponsel. Ponsel akan melakukan pengolahan pesan dengan masukan kunci sedemikian rupa sehingga pesan rahasia yang ditulis ditransformasikan menjadi sebuah kode-kode hexadesimal. Kode-kode tersebut melalui proses enkripsi diubah menjadi sebuah cipher. Pesan dalam bentuk cipher ini di kirimkan dan diterima melalui layanan MMS pada jaringan GSM. Jaringan GSM akan mengantarkan pesan pada nomer tujuan sehingga pesan diterima. Setelah diterima dengan memasukan kunci yang sesuai maka cipher akan ditransformasikan kembali menjadi plaintext semula yang dijadikan sebagai pesan rahasia. Penanganan berkas MMS tersebut pada perangkat mobile dilakukan pada kelas MMSMessage.java. Mulai dari pengecekan kesesuain nomor tujuan, subjek MMS yang ingin dimasukan dan memastikan tujuan MMS yang akan dikirim, pesan MMS yang akan dikirim dan cara menambahkan MMSnya. Secara detail untuk penyimpanan MMS akan dijelaskan pada kelas dbMessage. Mulai dari mendapatkan list inbox, proses save to inbox, status sent message dan status save sent, semua diproses berdasarkan nilai vektornya. Untuk penangan penyimpanan telah disediakan database untuk kedua proses yang telah disebutkan di atas yang akan dijelaskan pada kelas database.java. Pada blok diagram secara keseluran terdapat blok J2ME. Blok tersebut merupakan tanda dimana perancangan perangkat lunak (*software*) akan dilakukan. Pada komputer akan menggunakan bahasa pemrograman Java.

4.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak dipisah menjadi dua modul program (modul program kriptografi Rijndael dan modul program MMS) untuk memudahkan perancangan. Perangkat lunak untuk *algorithma* (modul program kriptografi Rijndael) dan MMS (modul program MMS) dibangun dengan bahasa pemrograman Java. Saat program MMS dijalankan , user dapat menuliskan pesan yang ingin dikirimkan. Kemudian user diminta untuk memasukan gambar yang dapat dipilih. Pesan dan gambar yang ditulis oleh user kemudian dipetakan dalam kode hexadecimal. User juga diminta untuk memasukan kunci setelah menulis pesan dan memilih gambar. Kunci tersebut yang akan dicampur dengan kode hexadecimal sehingga menjadi *cipher*. *Cipher* akan ditampung dalam sebuah

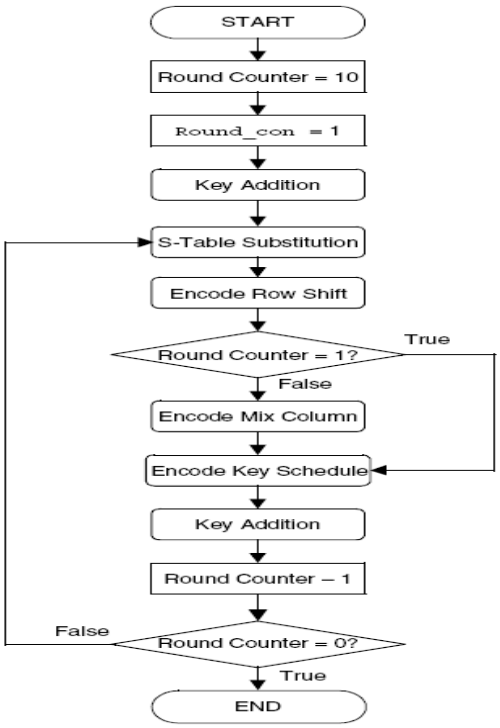
variabel untuk dikirimkan sesuai dengan nomer tujuan melalui layanan jaringan GSM.

4.2.1 Perancangan Modul Program Kriptografi

Perancangan modul Kriptografi Rijndael meliputi proses enkripsi dan proses dekripsi. Proses enkripsi berupa proses perhitungan antara data input berupa text dan gambar yang akan dirubah kedalam bentuk hexadesimal dan dilakukan proses berhitungan dengan tabel kunci yang telah dibentuk kemudian akan menjadi cipher. Dan proses dekripsi akan mengembalikan semua *cipher* yang telah di enkripsi ke dalam text dan gambar. Dalam setiap proses ini akan dijelaskan tahapan tahapan yang membentuk semua proses sehingga menjadi program yang utuh.

4.2.2 Perancangan Modul Program Enkripsi

Pada modul program enkripsi ini dijelaskan cara untuk mendapatkan *cipher* dari *plaintext* asli. Pada proses sebelumnya di asumsikan bahwa tabel sub kunci yang baru telah didapatkan sehingga tabel sub kunci yang baru tersebut akan digunakan dalam dalam proses perhitungan untuk mengacak *plaintext* asli agar dapat dirubah menjadi *cipher*. Berikut ini adalah perancangan modul enkripsi :



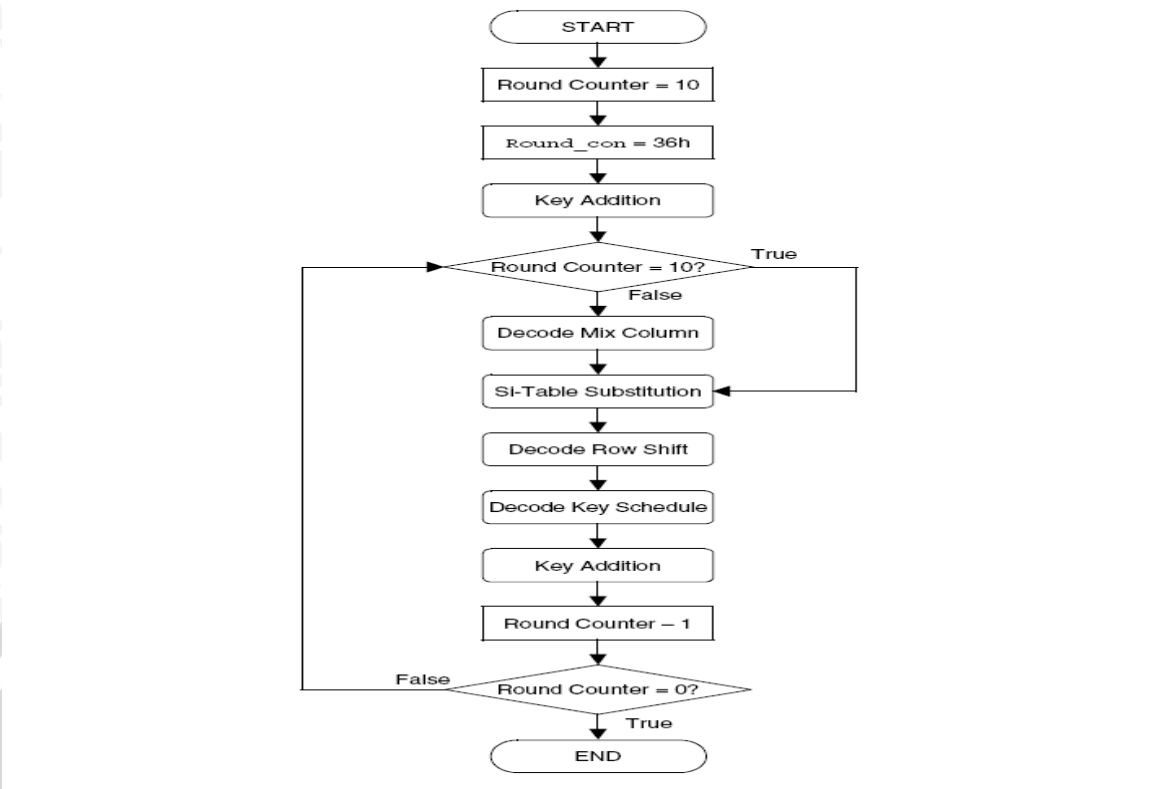
Gambar 4.2 Flowcart Modul Program Enkripsi

Berikut penjelasan flowchart modul program Enkripsi:

1. Mengeset round counter = 10 dan mengisi variable round_con =1
2. Merubah chipper key melalui metode key schedule pada rijndael menjadi round key sampai mendapatkan 10 round key
3. Melakukan metode subbyte dengan menggunakan table subtitusi atau S-Box table
4. Melakukan metode shiftRows
5. Melakukan metode MixColumns
6. Melakukan metode AddRoundKey
7. Mengulangi langkah 4-7 sebanyak 9x
8. Melakukan round ke 10 dengan menjalankan metode SubByte, AddRoundKey, ShiftRows dengan berurutan
9. Selesai dan menghasilkan chipertext

4.2.3 Perancangan Modul Program Dekripsi

Pada modul program dekripsi ini dijelaskan cara untuk mendapatkan plaintext dari cipher yang ada. Pada proses sebelumnya di asumsikan bahwa tabel sub kunci yang baru telah didapatkan dari kunci yang dimasukan sehingga tabel sub kunci yang baru tersebut akan digunakan dalam dalam proses perhitungan untuk mengembalikan plaintext asli dari cipher. Berikut ini adalah perancangan modul dekripsi :



Gambar 4.3 Flowcart Modul Program Dekripsi

Berikut penjelasan flowchart modul program Dekripsi:

1. Mengeset round counter = 10 dan mengisi variable round_con = 36h
2. Merubah chipper key melalui metode key schedule pada rijndael menjadi round key sampai mendapatkan 10 round key
3. Melakukan metode InvShiftRows
4. Melakukan metode InvSubByte dengan menggunakan table subtitusi atau S-Box table
5. Melakukan metode AddRoundKey
6. Melakukan metode InvMixColumns
7. Mengulangi langkah 4-7 sebanyak 9x
8. Melakukan round ke 10 dengan menjalankan metode InvShiftRows, InvSubByte, AddRoundKey dengan berurutan
9. Selesai dan menghasilkan plaintext

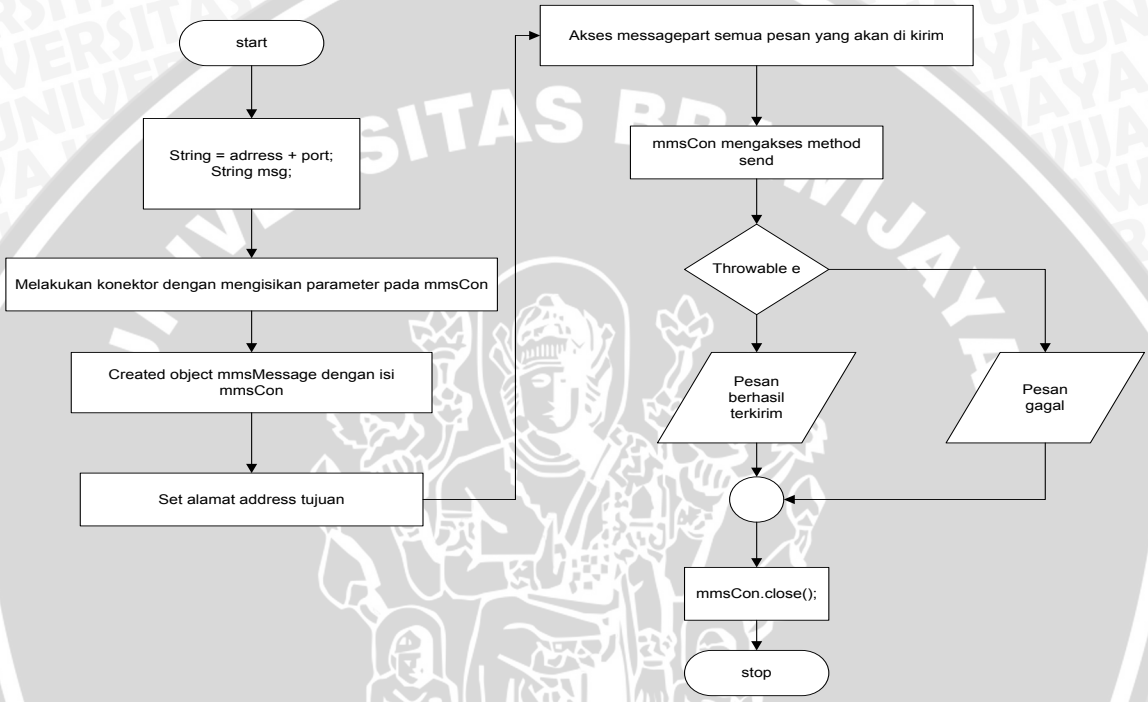
4.2.4 Perancangan Modul Program MMS Pengiriman dan Penerima

Modul program MMS ini digunakan untuk menampung cipher yang akan di kirim dan diterima. Handphone yang akan dipasangkan sebuah program MMS yang akan berhubungan dengan GSM network sebagai *wireless messeging API.*

Selain itu, program ini juga digunakan untuk menulis pesan MMS, mengatur kemudian mengirimkan MMS melalui handphone.

4.2.4.1 Pengiriman

Seperti yang didefinisikan oleh Generic Connection Framework pengiriman pesan dilaksanakan oleh antarmuka Connection. Untuk membuat sambungan berlangsung, aplikasi memperoleh obyek MessageConnection dari kelas Connector oleh menyediakan koneksi string URL yang mengidentifikasi alamat.



Gambar 4.4 Flowcart Modul Program Pengiriman

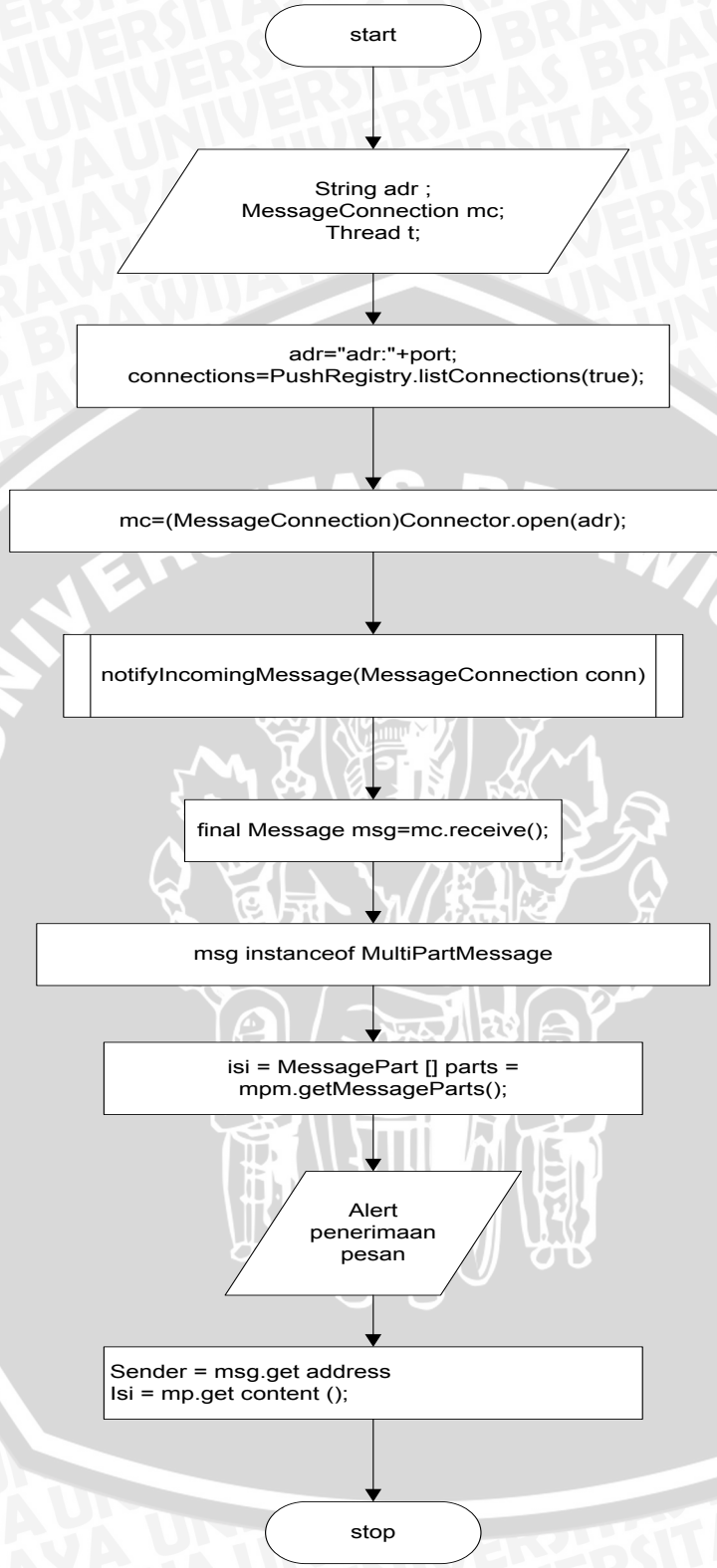
Berikut penjelasan flowchart modul program pengiriman pesan:

1. Inisialisasi nomer tujuan dan port pada sebuah String address dan melakukan passing pesan yang akan dikirim ke sebuah String msg.
2. Membuat variable mmsCon dengan type MessageConnection dalam keadaan null , Untuk membuka koneksi, MIDlet suite harus memiliki izin yang tepat untuk mengakses MessageConnection , mmsCon diatur untuk mengakses Connector agar membuka untuk alamat tujuan.
3. Membuat object baru dengan type mmsMessage berisi mmsCon yang mengakses MessageConnection dalam type MMS_MESSAGE dengan begitu maka seluruh parameter akan terisi pada mmsCon.

4. Variable mmsMessage akan mengakses method setAddress dengan isi String address berupa nomer tujuan dan port.
5. Method messagepart dipanggil pada String msg yang berisi seluruh pesan yang akan dikirim untuk bisa diakses oleh mmsMessage sehingga saat ini isi nya berupa nomer tujuan , port dan isi pesan.
6. MmsCon akan mengakses method send untuk mengirimkan pesan ke address yang telah diisikan dengan menampilkan informasi apakah pesan berhasil dikirim atau gagal
7. Dan terakhir mmsCon akan memanggil method close(); untuk menutup koneksi tanda berakhirnya proses pengiriman.

4.2.4.2 Penerima

Proses penerimaan pesan ini juga didefinisikan oleh Generic Connection Framework. Aplikasi membuat object lain dengan type MessageConnection dari kelas Connector yang mencakup mekanisme untuk mendaftarkan MIDlet ketika acara pemberitahuan koneksi terdeteksi. Setelah MIDlet telah diluncurkan ia melakukan yang sama I / O operasi biasanya akan digunakan untuk membuka koneksi. Untuk aplikasi WMA kemampuan ini memungkinkan aplikasi yang akan diluncurkan jika pesan datang. Dalam rangka untuk melakukan Push pendaftaran untuk koneksi WMA suite harus meminta izin untuk menggunakan PushRegistry dan izin untuk membuka koneksi untuk menerima pesan.



Gambar 4.5 Flowcart Modul Program Penerimaan

Berikut penjelasan flowchart modul program penerimaan pesan:

1. Inisialisasi nomer pengirim dan port pada sebuah String address dan inisialisasi variable untuk menangkap pesan yang diterima. Kemudian membuat sebuah thread untuk bisa menerima pesan.
2. Membuat variabel connections yang berisi PushRegistry sebagai ijin untuk bisa menerima pesan dari jaringan GSM di setting true sehingga koneksi terbuka
3. PushRegistry telah membuka koneksi kemudian MessageConnection membuka Connector untuk mendaftarkan MIDlet ketika acara pemberitahuan koneksi terdeteksi dan mengakses setMessageListener.
4. Membuat sebuah thread bahwa ada pesan masuk pada MessageConnection dan variabel msg yang bertype Message yang mengakses method received().
5. Msg melakukan instansi dari mmsMessage , kemudian variabel isi akan dipasingkan isi pesan yang akan diterima melalui method MessagePart. Kemudian akan dibentuk sebuah tanda bahwa ada pesan yang masuk.
6. Isi dari method MessageParts() berupa nomer pengirim dan isi dari pesan akan disimpan pada variabel pesan.

4.3 Implementasi Sistem

Setelah tahap perancangan sistem tahap selanjutnya adalah tahap implementasi. Tujuan dari tahap implementasi ini merupakan proses transformasi hasil perancangan perangkat lunak. Pembahasan terdiri dari lingkungan implementasi (spesifikasi perangkat lunak dan perangkat keras), implemntasi antarmuka aplikasi dengan sintaks dari bahasa pemrograman yang digunakan.

4.3.1 Lingkungan Implementasi

Sistem dibuat dengan menggunakan Netbeans IDE 7.0 dengan Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

1. Perangkat keras :

   1.1. Komputer

   Spesifikasi :

   - Processor : Genuine Intel(R) core 2 duo P7750 2.26 Ghz
   - Memory : 1720 MB RAM
   - OS : Windows XP

- VGA : Geforce Nvidia 6600

1.2. *handphone*

Spesifikasi:

Model Samsung Monte

- 2G network (GSM 850/900/1800/1900)
- 3G network (HSDPA 900/1200)
- Display Type TFT capacitive touchscreen, 256K colors Size 240 x 400 pixels, 3.0 inches, Accelerometer sensor for UI auto-rotate, TouchWiz UI 2.0 Plus, Turn to mute, Smart unlock
- Data GPRS Class 10 (4+1/3+2 slots) 32 - 48 kbps, EDGE Class 10 236.8 kbps, 3G HSDPA 3.6 Mbps, WLAN, Wi-Fi 802.11 b/g, Bluetooth v2.1 with A2DP, USB microUSB v2.0
- Features messaging (SMS, MMS, Push Email, Palringo IM), Java (MIDP 2.0)

2. Perangkat Lunak :

2.1. Sistem operasi : Microsoft Windows XP SP 3 2002

2.2. Bahasa pemrograman: Java 2 Micro Edition (J2ME)

2.3. IDE : Netbeans 7.0 + Netbeans Mobility Pack

2.4. Tools Pemrograman : JDK 1.6_22

J2ME WTK 2.5.2

```
tion(
ublic class enkripsiRijndael {
    private boolean    doneEncrypt = false;
    private String     key = "0123456789abcdef0123456789abcdef";
    private String                 plainText   = "Rijndael";
    private byte[]                 keyBytes    = null;
    private byte[]                 cipherText  = null;
    private BufferedBlockCipher    cipher      = null;
enkripsiRijndael(){
cipher = new PaddedBufferedBlockCipher(new CBCBlockCipher(new
ijndaelEngine()));
String name = cipher.getUnderlyingCipher().getAlgorithmName();
System.out.println("Using " + name +" "+cipher.getBlockSize());
System.out.println("Key: " + key );
System.out.println("In : " + plainText );
}
```

4.3.2 Implementasi Program Algoritma Rijndael

Program kriptografi rijndael dapat dipaparkan sebagai berikut:

Penjelasan program di atas :

1. Baris 1-7 mendeklarasikan variable serta menginisialisasi kelas enkripsiRijndael.
2. Baris 8-9 membuat objek chiper sebagai penunjuk untuk mengaktifkan Application Programming Intervace (API) RijndaelEngine.
3. Baris 10-13 menampilkan informasi tampilan program tersebut.

Proses KeyAddition :

Penjelasan program di Atas :

1. Baris 1-3 mendeklarasikan kelas KeyAddition bertipe data long untuk rk.

```
private void Substitution(
    byte[]     box)
{
    A0 = applyS(A0, box);
    A1 = applyS(A1, box);
    A2 = applyS(A2, box);
    A3 = applyS(A3, box);
}
private long applyS(
    long    r,
    byte[]  box)
{
    long    res = 0;
    for (int j = 0; j < BC; j += 8)
    {
        res |= (long)(box[(int)((r >> j) & 0xff)] & 0xff)
<< j;
    }
    return res;
}
```

2. Baris 4-7 menginisialisasikan proses penjumlahan dari dua elemen dalam suatu *finite field* dilakukan dengan menjumlahkan koefisien dari pangkat *polynomials* yang bersesuaian dari dua elemen tersebut.

Proses Subtitution :

Penjelasan program di atas :

1. Baris 1-3 mendeklarasikan kelas substitution dengan tipe data byte untuk box.
2. Baris 4-7 menjalankan metode applyS
3. Baris 9-11 mendeklarasikan kelas applyS dengan tipe data long untuk variabel r dan tipe data byte untuk variabel box.
4. Baris 13-17 proses transformasi *byte* dimana setiap elemen pada *state* akan dipetakan dengan menggunakan suatu tabel substitusi (*S-Box*).

Proses MixColumn

```
xff);
xff);
xff);
xff);
ul0x3(a1)  ^ a2  ^ a3)  &

ul0x3(a2)  ^ a3  ^ a0)  &

ul0x3(a3)  ^ a0  ^ a1)  &

ul0x3(a0)  ^ a1  ^ a2)  &
```

Penjelasan program di atas :

1. Baris 1-4 mendeklarasikan kelas MixColumn dengan tipe data long untuk variabel r0 r1 r2 r3 dengan nilai awal 0.
2. Baris 5-24 menjelaskan proses mengoperasikan setiap elemen yang berada dalam satu kolom pada *state*, elemen pada kolom dikalikan dengan suatu *polynomials* tetap.

Proses InvMixColumn

```
1  private void InvMixColumn()
2      {
3          long r0, r1, r2, r3;
4          r0 = r1 = r2 = r3 = 0;
5          for (int j = 0; j < BC; j += 8)
6          {
7              int a0 = (int)((A0 >> j) & 0xff);
8              int a1 = (int)((A1 >> j) & 0xff);
9              int a2 = (int)((A2 >> j) & 0xff);
10             int a3 = (int)((A3 >> j) & 0xff);
11             a0 = (a0 != 0) ? (logtable[a0 & 0xff] & 0xff) : -1;
12             a1 = (a1 != 0) ? (logtable[a1 & 0xff] & 0xff) : -1;
13             a2 = (a2 != 0) ? (logtable[a2 & 0xff] & 0xff) : -1;
14             a3 = (a3 != 0) ? (logtable[a3 & 0xff] & 0xff) : -1;
15             r0 |= (long)((mul0xe(a0) ^ mul0xb(a1) ^ mul0xd(a2)
` mul0x9(a3)) & 0xff) << j;
16             r1 |= (long)((mul0xe(a1) ^ mul0xb(a2) ^ mul0xd(a3)
` mul0x9(a0)) & 0xff) << j;
17             r2 |= (long)((mul0xe(a2) ^ mul0xb(a3) ^ mul0xd(a0)
` mul0x9(a1)) & 0xff) << j;
18             r3 |= (long)((mul0xe(a3) ^ mul0xb(a0) ^ mul0xd(a1)
` mul0x9(a2)) & 0xff) << j;
19         }
20         A0 = r0;
21         A1 = r1;
22         A2 = r2;
23         A3 = r3;
24     }
```

Penjelasan program di atas :

1. Baris 1-4 mendeklarasikan kelas InvMixColumn dengan tipe data long untuk variabel r0 r1 r2 r3 dengan nilai awal 0.
2. Baris 5-28 menjelaskan tentang kolom-kolom pada tiap *state* (*word*) akan dipandang sebagai *polynomials*.

Proses ShiftRow

```
<<  (BC  -  shift)))  &
```

1. Baris 1-2 mendeklarasikan kelas ShiftRow dengan tipe data byte untuk objek shiftsSC
2. Baris 4-6 menjalankan metode shift
3. Baris 8-10 mendeklarasikan kelas shift bertipe data long untuk objek r dan tipe data int untuk objek shift
4. Baris 12-14 menjalankan metode shift yang digunakan untuk melakukan pergeseran *bit* dimana *bit* paling kiri akan dipindahkan menjadi *bit* paling kanan (rotasi *bit*).

```
    long[][] rk)
  {
    int r;
  KeyAddition(rk[0]);
for (r = 1; r < ROUNDS; r++)
    {
      Substitution(S);
      ShiftRow(shifts0SC);
      MixColumn();
      KeyAddition(rk[r]);
    }
  Substitution(S);
    ShiftRow(shifts0SC);
    KeyAddition(rk[ROUNDS]);
  }
```

4.3.3 Implementasi Program Enkripsi

Program enkripsi rijndael dapat dipaparkan sebagi berikut

Penjelasan program di atas :

1. Baris 1-2 mendeklarasikan kelas encryptBlock. Kelas tersebut adalah proses enkripsi rijndael secara umum dalam setiap proses pengiriman suatu pesan.
2. Baris 3-16 adalah tahapan-tahapan proses yang dilakukan untuk mengenkripsi suatu pesan.

4.3.4 Implementasi Program Dekripsi

Program Dekripsi Rijndael dapat dipaparkan sebagi berikut:

```
1  private void decryptBlock(
       long[][] rk)
2  {
       int r;
       KeyAddition(rk[ROUNDS]);
       Substitution(Si);
       ShiftRow(shifts1SC);
       for (r = ROUNDS-1; r > 0; r--)
       {
           KeyAddition(rk[r]);
           InvMixColumn();
           Substitution(Si);
           ShiftRow(shifts1SC);
       }
       KeyAddition(rk[0]);
16 }
```

Penjelasan program di atas :

1. Baris 1-2 mendeklarasikan kelas decryptBlock. Kelas tersebut adalah proses dekripsi rijndael secara umum dalam setiap proses penerimaan suatu pesan.
2. Baris 3-16 adalah tahapan-tahapan proses yang dilakukan untuk mendekripsikan suatu pesan.

4.3.5 Implementasi Program Pengiriman dan Penerima MMS

Program pengiriman MMS dapat dipaparkan sebagi berikut:

Program untuk memeriksa koneksi sebelum dikirimkan MMS :

```
public class SenderThread extends Thread {
    private MMSMessage message;
    private String appID;
    public SenderThread(MMSMessage message, String appID) {
        this.message = message;
        this.appID = appID;
    }
public void run() {
        String  address  =  message.getDestination()  +  ":"  +
appID;
        MessageConnection mmsconn = null;
        try {
mmsconn = (MessageConnection)Connector.open(address);
            MultipartMessage mmmessage =
(MultipartMessage)mmsconn.newMessage(MessageConnection.MULTIPAR
T_MESSAGE);
            mmmessage.setAddress(address);
            MessagePart[] parts = message.getParts();
            for (int i = 0; i < parts.length; i++) {
                mmmessage.addMessagePart(parts[i]);
            }
            mmmessage.setSubject(message.getSubject());
            mmsconn.send(mmmessage);
        } catch (Exception e) {
        }
        if (mmsconn != null) {
            try {
                mmsconn.close();
            } catch (IOException ioe) {
            }
}
}
}
```

```
private void promptAndSend() {
        try {
            String time = new
Date(System.currentTimeMillis()).toString();
            time = parseTime(time);
            String address = getTextField().getString();
            message.setSubject("mms ku");
            message.setDestination("mms://"+address);
            String statusMessage = "Sending message to " +
address + "...";
            sendingMessageAlert.setString(statusMessage);
            switchDisplayable(sendingMessageAlert, getList());
            send = new SenderThread(message, appID);
            send.start();
            dbm.dbSent.addElement(new database(address, time,
imgOut, msgOut, 0, mimeType));
        } catch (IllegalArgumentException iae) {
            errorMessageAlert.setString(iae.getMessage());
            getDisplay().setCurrent(errorMessageAlert);
        }
    }
```

Penjelasan program di atas :

1. Baris 1-6 membuat thread baru untuk pengiriman MMS.
2. Baris 8-terakhir menjelaskan bagaimana proses membuka koneksi pengiriman pesan.

Program untuk mengirim MMS :

Penjelasan program di atas :

1. Baris 1 mendeklarasikan kelas promptAndSend untuk proses pengiriman MMS.
2. Baris selanjutnya adalah mendeklarasikan isi informasi dari pesan tersebut yaitu berupa waktu, alamat, nama subjek.

Program penerimaan MMS dapat dipaparkan sebagai berikut :

Program untuk mengakses koneksi penerimaan MMS :

```
1  private void startReceive(){
2  String mmsConnection = "mms://:" + appID;
3  if (mmsconn == null) {
4          try {
5              mmsconn
   (MessageConnection)Connector.open(mmsConnection);
6              mmsconn.setMessageListener(this);
7          } catch (IOException ioe) {
8              System.out.println("Eror startReceive : \n");
9          }
10     }
11 connections = PushRegistry.listConnections(true);
12     if ((connections == null) || (connections.length == 0))
   {
   }
13     done = false;
14     thread = new Thread(this);
15     thread.start();
   }
```

Penjelasan program di atas :

1. Baris 1-2 mendeklarasikan kelas startReceive untuk penerimaan MMS.
2. Baris 3-10 membuka koneksi penerimaan pesan.
3. Baris 11-18 digunakan menginisialisasi pesan.

Program untuk mengakses inbox :

```
private void detailInbox(int id)
    {
            database db = (database) dbm.dbInbox.elementAt(id);
            ((database) dbm.dbInbox.elementAt(id)).status = 1;
            String nama = db.sender;
            getForm2().setTitle("From: " + nama);
            getStringItem6().setText("Dikirim pada "+db.time);
       imgIn = db.image;
            msgIn = db.message;
            mimeType = db.mime;
            System.out.println(mimeType);
String encPsn = msgIn;
            String encImg = new String(imgIn);
if(msgIn.length() > 100){
                encPsn     =     String.valueOf(msgIn.substring(0,
100)+"....");
            }
            getStringItem4().setText("\n"+encPsn);
            if(encImg.length() > 300){
                encImg     =     String.valueOf(encImg.substring(0,
300)+"....");
            }
            getStringItem5().setText("\n"+encImg);
    }
```

Penjelasan program di atas :

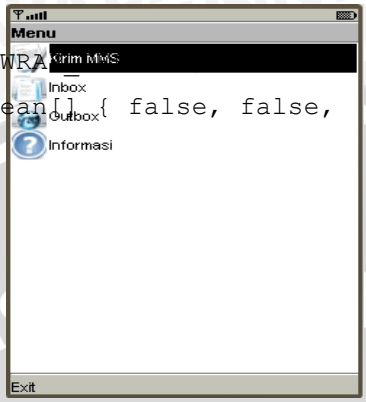
1. Baris 1 mendeklarasikan kelas detailInbox untuk metode penerimaan MMS.
2. Baris 3-4 mengakses database MMS di dalam perangkat.
3. Baris 6-10 memberikan informasi dari pesan tersebut berupa nama pengirim waktu pengiriman.
4. Baris berikutnya menjelaskan proses enkripsi

```
t = new List("Menu", Choice.IMPLICIT);
      list.append("Kirim MMS", getImage2());
      list.append("Inbox", getImage3());
      list.append("Outbox", getImage4());
      list.append("Informasi", getImage5());
      list.addCommand(exitCommand);
      list.setCommandListener(this);
      list.setFitPolicy(Choice.TEXT_WRAP_ON);
      list.setSelectedFlags(new boolean[] { false, false,
se, false });
```

4.4 Implementasi Antarmuka Program

4.4.1 Implementasi Antarmuka Program Menu

Program kriptografi memiliki tampilan menu antarmuka berupa kirim MMS, kotak Inbox, Outbox, informasi, dan exit :



Gambar4.6 User Interface Menu

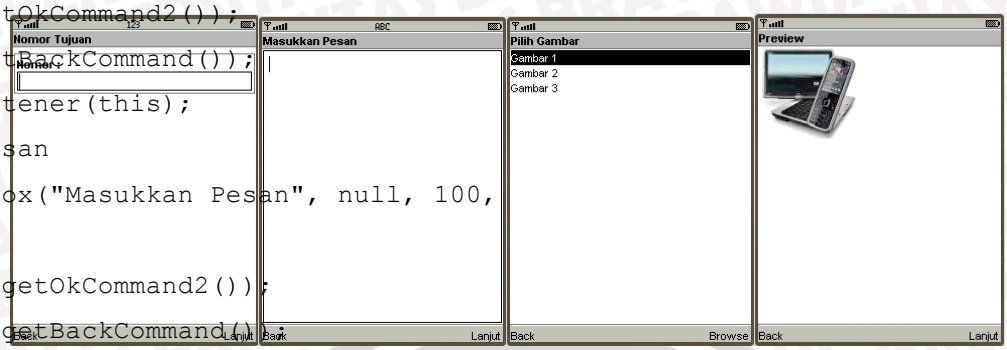
Pada aplikasi ini , antarmuka menu dibuat untuk memudahkan pengguna dalam menggunakan aplikasi yang telah dibangun. Antarmuka ini ditampikan ketika pertama kali program dijalankan. bagian script *J2ME* sebagai berikut untuk menu

dan command:

4.4.2 Implementasi Antarmuka Kirim Pesan

Untuk menu kirim pesan , antarmuka penulisan pesan ini ditampilkan ketika pengguna ingin mengirim mms, pertama masukan nomor tujuan, masukan pesan text, pilih gambar yang tersedia atau dapat memilih gambar yang kita inginkan melalui browse, kemudian preview gambar. Selain itu disediakan dua tombol yaitu back untuk kembali ke menu sebelumnya dan lanjut untuk menuju menu selanjutnya :

```
//Gambar Nomor Tujuan
form1 = new Form("Nomor Tujuan", new Item[] { getSpacer(),
getTextField() });
form1.addCommand(getOkCommand2());
form1.addCommand(getBackCommand());
form1.setCommandListener(this);
//Gambar Masukan Pesan
textBox = new TextBox("Masukkan Pesan", null, 100,
TextField.ANY);
textBox.addCommand(getOkCommand2());
textBox.addCommand(getBackCommand());
textBox.setCommandListener(this);
//Gambar Pilih Gambar
list1 = new List("Pilih Gambar", Choice.IMPLICIT);
list1.append("Gambar 1", null);
list1.append("Gambar 2", null);
list1.append("Gambar 3", null);
list1.addCommand(getBackCommand());
list1.addCommand(getOkCommand());
list1.setCommandListener(this);
list1.setSelectedFlags(new boolean[] { false, false, false });
//Gambar Preview
form = new Form("Preview", new Item[] { getImageItem() });
form.addCommand(getBackCommand());
form.addCommand(getOkCommand2());
form.setCommandListener(this);
```



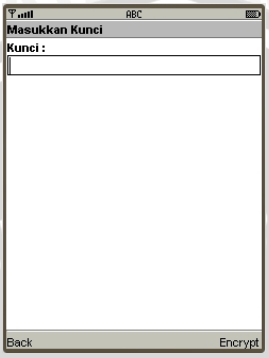
Gambar4.7 User Interface Kirim Pesan

Berikut adalah potongan program :

```
form5 = new Form("Masukkan Kunci", new Item[] { getTextField1()
});
form5.addCommand(getOkCommand3());
form5.addCommand(getBackCommand());
form5.setCommandListener(this);
```

4.4.3 Implementasi Antarmuka Masukan Kunci

Antarmuka penulisan kunci rahasia ini ditampilkan ketika pengguna diminta memasukan kunci rahasia. Pada antarmuka Masukan Kunci ini disediakan dua tombol yaitu back untuk menuju antarmuka sebelumnya dan encrypt untuk melanjutkan ke menu selanjutnya.



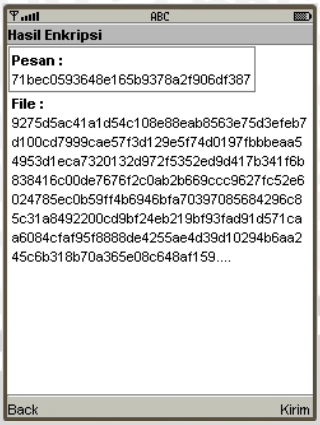
Gambar4.8 User Interface Masukan Kunci

Berikut adalah potongan program :

4.4.4 Implementasi Antarmuka Hasil Enkripsi

Antarmuka Kirim MMS ini akan ditampilkan ketika pengguna ingin mengirimkan pesan hasil enkripsi , dan nomer tujuan yang akan dikirimkan. Pada antarmuka ini ada beberapa tombol menu yaitu back ke menu utama dan kirim untuk mengirim dan melakukan konfirmasi delivery report.

```
form3 = new Form("Hasil Enkripsi", new Item[] { getStringItem2(),
    getSpacer1(), getStringItem3() });
form3.addCommand(getOkCommand1());
form3.addCommand(getBackCommand());
form3.setCommandListener(this);
```

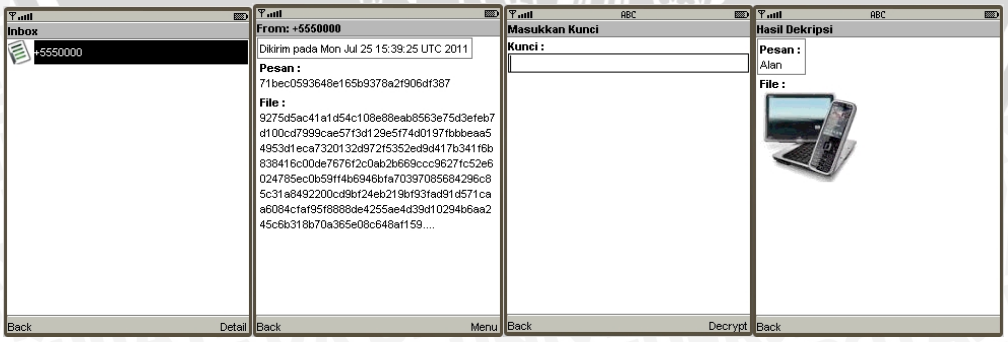


Gambar4.8 User Interface Kirim SMS

Berikut adalah potongan program :

4.4.5 Implementasi Antarmuka Penerimaan Pesan

Antarmuka penerimaan pesan ini digunakan ketika pengguna mendapatkan pesan dari aplikasi ini. Antarmuka ini akan menampilkan informasi alert mendapatkan pesan dan mengarahkan ke pesan yang dikirimkan oleh pengguna lain untuk dilakukan dekripsi terhadap pesan yang diterima. Pada antarmuka ini ada beberapa tombol menu yaitu kembali ke menu utama dan dekripsi untuk mengirim dan melakukan proses dekripsi dengan sebelumnya memasukan kunci yang tepat terlebih dahulu.



Gambar4.8 User Interface Penerimaan SMS

Berikut adalah potongan program :

```
//Gambar isi Inbox
form2 = new Form("Pesan", new Item[] { getStringItem6(),
getSpacer2(), getStringItem4(), getSpacer3(), getStringItem5()
});
form2.addCommand(getBackCommand());
form2.addCommand(getOkCommand2());
form2.addCommand(getOkCommand6());
form2.setCommandListener(this);
//Gambar masukan kunci
form6 = new Form("Masukkan Kunci", new Item[] { getTextField2()
});
form6.addCommand(getOkCommand4());
form6.addCommand(getBackCommand());
form6.setCommandListener(this);
//Gambar hasil dekripsi
form4 = new Form("Hasil Dekripsi", new Item[] { getStringItem(),
getSpacer4(), getImageItem1() });
form4.addCommand(getBackCommand());
form4.setCommandListener(this);
```

BAB V
PENGUJIAN DAN ANALISIS

Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkain pengujian. Pengujian yang dilakukan dalam bab ini adalah sebagai berikut:

1. Analisi vektor pada algorithma Rijndael
2. Pengujian validasi
3. Pengujian Citra

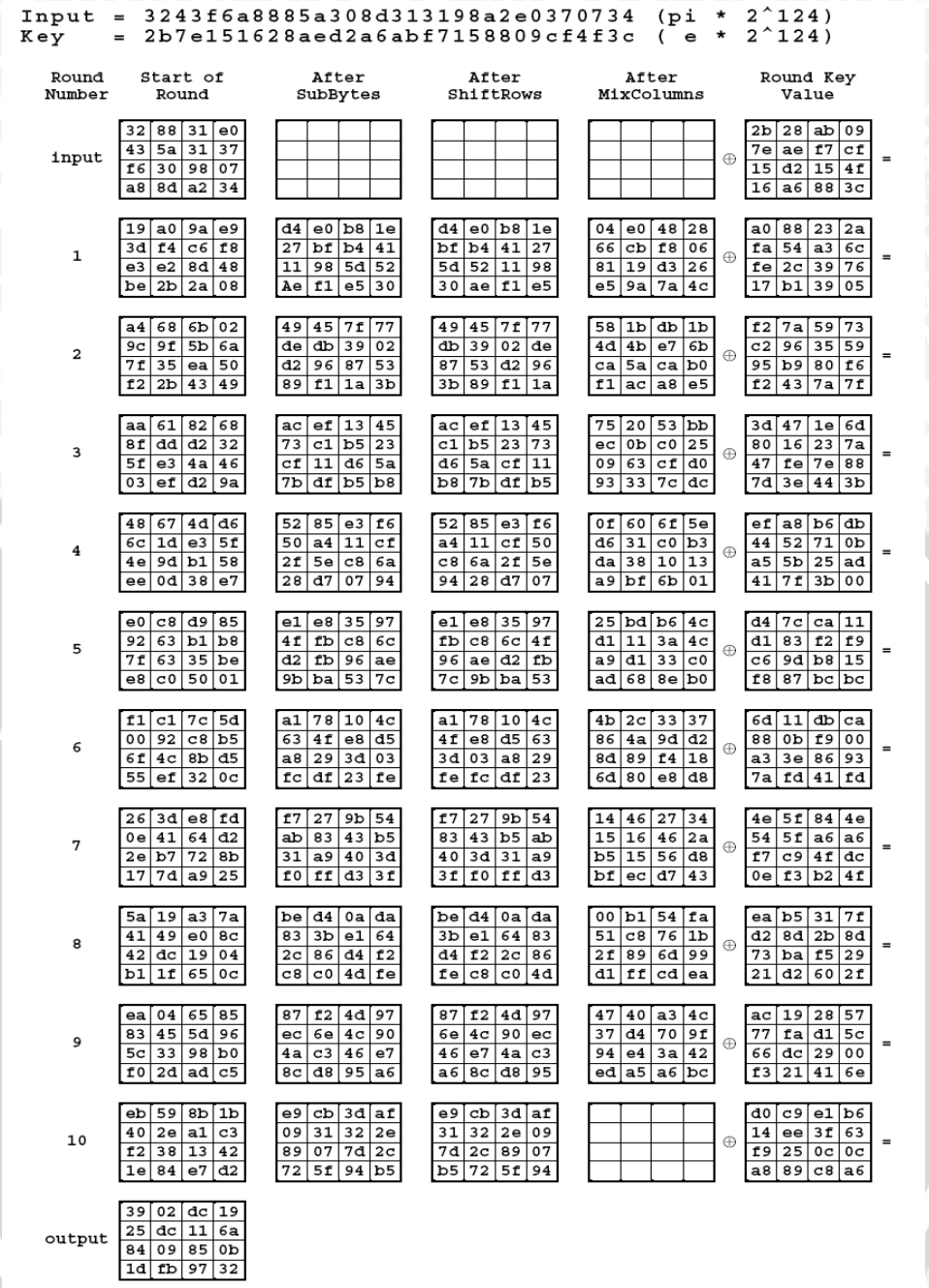
5.1 Analisis Vektor Pada Implementasi Algoritma Rijndael

Proses analisis ini dilakukan untuk mengetahui valid atau tidaknya implementasi algoritma Rijndael yang digunakan , proses analisis ini dilakukan dengan cara melakukan pengujian vektor pada implementasi algorithma Rijndael yang dirancang , dilakukan dengan cara memasukan parameter kunci dan plaintext berupa text yang ditentukan dengan mengamati keluaran ciphertextnya seperti pada data dibawah ini

Tabel 5.1 Pengujian hasil vektor test pada program antara hubungan plain, key dan cipher.

Plain	Key	Cipher
3243f6a8885a308d 313198a2e0370734	**2b7e151628aed2a6 abf7158809cf4f3c**	**3925841d02dc09fb dc118597196a0b32**

Berdasarkan hasil pengujian vektor yang telah dilakukan dapat dilakukan analisis dengan cara membandingkan hasil yang telah didapatkan dengan hasil vektor tes yang umumnya digunakan untuk melakukan validasi implementasi algritma Rijndael. Dari hasil perbandingan keluaran ciphertext yang didapatkan dapat ditarik kesimpulan bahwa implementasi algorithma yang digunakan adalah valid menggunanakan algorithma Rijndael.



Tabel Vektor Tes Algorithma Rijndael oleh NIST

5.2 Pengujian validasi.

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang perancangan. Item- item yang telah dirumuskan dalam perancangan menjadi acuan untuk melakukan pengujian validasi.

5.2.1 Pengujian Pengiriman MMS Rahasia

Nama Kasus Uji : Kasus Uji Pengiriman MMS Rahasia

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengirimkan MMS rahasia melalui layanan MMS

Prosedur Uji : Memasukan nomor tujuan dan MMS yang di kirim kemudian menekan menu tombol Kirim MMS pada aplikasi.

Hasil yang diharapkan : Aplikasi dapat mengirimkan MMS dan menampilkan informasi bahwa MMS sudah dikirimkan.

5.2.2 Pengujian Penerimaan MMS Rahasia

Nama Kasus Uji : Kasus Uji Peneriman MMS Rahasia

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk menerima MMS rahasia melalui layanan MMS.

Prosedur Uji : Sistem mendeteksi adanya MMS yang masuk dan menampilkan form penerimaan MMS pengguna menekan tombol baca

Hasil yang diharapkan : Aplikasi dapat menampilkan MMS yang sedang diterima.

5.2.3 Pengujian Enkripsi MMS Rahasia

Nama Kasus Uji : Kasus Uji Enkripsi MMS Rahasia

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan enkripsi MMS rahasia menjadi MMS acak

Prosedur Uji : Menulis MMS pada box message untuk isian MMS. Mengisi kunci pada form isian kunci dengan data private key “1234” dan data masukan seperti pada tabel berikut :

Tabel 5.2 pengujian proses enkripsi

No	Data Masukan Plain
1	Alan
2	1234

Hasil yang diharapkan : Aplikasi dapat melakukan proses pengamanan MMS dengan cara melakukan enkripsi MMS dan menampilkan form yang berisi ciphertext dari MMS.

5.2.4 Pengujian Dekripsi MMS Rahasia

Nama Kasus Uji : Kasus Uji Dekripsi MMS Rahasia

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan dekripsi MMS rahasia dari MMS acak menjadi plaintext

Prosedur Uji : Menekan menu dekrip pada form penerimaan MMS dengan data masukan MMS adalah data keluaran dari uji kasus enkripsi MMS rahasia. Memasukan private key "1234" untuk proses dekripsi.

Tabel 5.3 pengujian proses dekripsi

No	Cipher
1	2a416dd22d7b1484953a8aa0f7de19c5
2	Fbd704ec27aaceb3b07cd4c6f441644

Hasil yang diharapkan : Aplikasi dapat melakukan proses dekripsi pada ciphertext MMS dan menampilkan form dengan MMS telah terdekripsi.

5.2.5 Hasil Pengujian Validasi

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian pada maka didapatkan hasil sebagai berikut

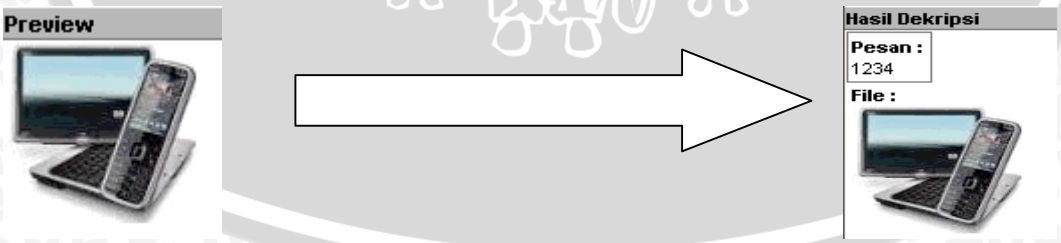
Tabel 5.4 *Test case* untuk pengujian validasi

No.	**Kasus Uji**	**Hasil yang didapatkan**	**Status**
1.	Pengiriman MMS	Aplikasi dapat mengirimkan MMS	Valid

<table>
<tr><td></td><td>rahasia</td><td>ke nomer tujuan dan berhasil.</td><td></td></tr>
<tr><td>2.</td><td>Penerimaan MMS rahasia</td><td>Aplikasi dapat melakukan proses penerimaan MMS.</td><td>Valid</td></tr>
<tr><td>3.</td><td>Enkripsi MMS rahasia</td><td>Aplikasi dapat melakukan proses dekripsi dengan hasil data pada tabel enkripsi
<table>
<tr><th>No</th><th>Cipher</th></tr>
<tr><td>1</td><td>2a416dd22d7b1484 953a8aa0f7de19c5</td></tr>
<tr><td>2</td><td>Fbd704ec27aaceb3b 07cd4c6f441644</td></tr>
</table></td><td>Valid</td></tr>
<tr><td>4</td><td>Dekripsi MMS rahasia</td><td>Aplikasi dapat melakukan proses dekripsi dan mengembalikan ciphertext ke plaintext seperti pada data enkripsi
<table>
<tr><th>No</th><th>Hasil dekripsi</th></tr>
<tr><td>1</td><td>Alan</td></tr>
<tr><td>2</td><td>1234</td></tr>
</table></td><td>Valid</td></tr>
</table>

5.3 Pengujian Citra

Pengujian ini dilakukan untuk membandingkan gambar yang dikirim melalui MMS sama dengan gambar yang diterima walaupun melalui proses enkripsi dan dekripsi sebagaimana ditunjukan oleh gambar berikut ini :



BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian maka dapat diambil kesimpulan sebagai berikut:

1. Sebuah perangkat lunak yang mengimplementasikan suatu algoritma kriptografi kunci privat untuk enkripsi MMS telah berhasil dibangun. Perangkat lunak yang dibangun tersebut dapat melakukan pengiriman MMS dan penerimaan MMS terenkripsi tersebut dengan baik. Perangkat unak tersebut menggunakan algoritma Rijndael untuk enkripi MMS. Perangkat lunak tersebut dapat ditanamkan pada telepon selular samsung monte dengan nomor XL berkode depan 0817 dan dibangun dengan menggunakan bahasa pemrograman J2ME.
2. Sistem ini dapat melakukan pengamanan MMS dengan cara menyandikan pesan sandi (ciphertext) sebelum dikirim dan mengirimkannya melalui layanan MMS dari hasil pengujian yang telah dilakukan.
3. Sistem ini dapat menerima pesan sandi (ciphertext) melalui layanan MMS dan melakukan dekripsi pada pesan sandi agar dapat dibaca oleh pengguna.
4. Algoritma Rijndael dapat diimplementasikan dengan baik untuk melakukan enkripsi MMS yang bekerja pada jaringan GSM dengan mengirimkan pesan yang berbentuk bilangan heksa.
5. Telah dilakukan pengujian pengiriman 2x mms dan berhasil 100% serta penerimaan mms dan berhasil 100% sesuai dengan mms yang dikirim.

6.2 Saran

Dalam perancangan dan pembuatan sistem keamanan ini masih terdapat kekurangan dan kelemahan, oleh karena itu masih diperlukan adanya

penyempurnaan dalam rangka pengembangan kedepan. adapun hal yang dapat disempurnakan antara lain:

1. Untuk pengembangan lebih lanjut, dapat dikembangkan pada handheld lain dan no penyedia layanan lain.
2. Untuk pengembangan lebih lanjut, dapat digunakan algoritma kriptografi yang telah dibangun ini untuk diimplementasikan pada aplikasi lain
3. Untuk kelancaran proses komunikasi sebaiknya menggunakan kartu *sellular* dari *provider* telepon *sellular* dengan sinyal GSM yang kuat di tempat pengaplikasian sistem.

DAFTAR PUSTAKA

[001] Prof.N..Penchalaiah, 2007. *Multimedia Messaging Service (MMS)*. India : Department of Computer Science Engineering ASCET Gudur.

[002] Sitorus, Syahriol dkk. 2006. *Global system for mobile communication*. Medan. USU Press

[003] Bow, Sing – Tze. 2002. *Finite Field GF Method.* NewYork : Marcel Pekker, Inc.

[004] Efford, Nick. 2000. *Addition Multiplication and Devided Method.* NewYork : Addison Wesley.

[005] Suksmono. 2005. *Algoritma Rijndael*. Diktat Kuliah AES, ITB

[006] Li, Yang. 2010. *Protactor : Java Mobile*, GA, USA

[007] Huang, Ying. 2009. *Begin Java Journey in Hours.*

[008] Xuguang, Heung. 2009. *An Introduction to Java.* Database Lab. Inha University.

[009] Hashimi, Sayed dkk. 2010. *Pro Java 2*. NewYork : Springer.

[010] Thomas, S. Huang dkk. 1997. *IEEE Transaction On Pattern Analysis.*

[011] Najib, Marsum. 2005. *Diktat Matakuliah Kalkulus 'Fungsi Hash'*. UGM. Yogyakarta