BAB II

LANDASAN TEORI

2.1 Jaringan Komputer

Jaringan komputer adalah sebuah kumpulan komputer, printer dan peralatan lainnya yang terhubung. Data bergerak melalui kabel-kabel maupun media transmisi lainnya sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, mencetak pada printer yang sama dan bersama sama menggunakan hardware/software yang terhubung dengan jaringan. Tiap komputer, printer atau periperal yang terhubung dengan jaringan disebut **node**. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan node.

Sebuah jaringan biasanya terdiri dari 2 (dua) atau lebih komputer yang saling berhubungan di antara satu dengan yang lain, dan saling berbagi sumber daya misalnya CDROM, printer, pertukaran file, atau memungkinkan untuk saling berkomunikasi secara elektronik. Komputer yang terhubung tersebut, dimungkinkan berhubungan dengan media kabel, saluran telepon, gelombang radio, satelit, atau sinar infra merah (Dian Ardiyansah, 2004).

2.1.1 Internet

Internet adalah jaringan komputer yang bersifat global dan merupakan gabungan-gabungan dari ratusan bahkan ribuan jaringan dengan skala yang lebih kecil di dalamnya, seperti jaringan *Wide Area Network* (WAN) dan *Metropolitan*

Area Network (MAN) yang dibatasi oleh area yang relatif lebih kecil, umumnya dibatasi oleh area dimana jaringan tersebut berada seperti negara.

2.2 Protokol TCP/IP

TCP/IP adalah sekelompok protokol yang mengatur komunikasi data komputer di Internet. Komputer-komputer yang terhubung ke internet berkomunikasi dengan protokol ini. Karena menggunakan bahasa yang sama, yaitu protokol TCP/IP, perbedaan jenis komputer dan system operasi tidak menjadi masalah. Personal Computer (PC) dengan Sistem Operasi Windows dapat berkomunikasi dengan komputer Macintosh atau dengan Sun SPARC yang menjalankan Solaris. Jadi, jika sebuah komputer menggunakan protokol TCP/IP dan terhubung langsung ke Internet, maka komputer tersebut dapat berhubungan dengan komputer di belahan dunia manapun yang juga terhubung ke Internet.

2.2.1 Arsitektur Protokol TCP/IP

Pada dasarnya, komunikasi data merupakan proses mengirimkan data dari satu komputer ke komputer yang lain. Untuk dapat mengirimkan data, pada komputer lain harus ditambahkan alat khusus, yang dikenal sebagai *network interface* (interface jaringan). Jenis interface jaringan ini bermacam-macam, bergantung pada media fisik yang digunakan untuk mentransfer data tersebut.

Dalam proses pengiriman data ini terdapat beberapa masalah yang harus dipecahkan. Pertama, data harus dapat dikirimkan ke komputer yang tepat sesuai dengan tujuannya. Hal ini akan menjadi rumit jika komputer tujuan transfer data ini tidak berada pada jaringan lokal, melainkan ditempat yang saling berjauhan.

Jika lokasi komputer yang saling berkomunikasi jauh (secara jaringan) maka terdapat kemungkinan data rusak atau hilang. Karenanya, perlu ada mekanisme untuk mencegah rusaknya data ini.

Hal lain yang perlu diperhatikan ialah, pada komputer tujuan transfer data mungkin terdapat lebih dari satu aplikasi yang menunggu datangnya data. Data yang dikirim harus sampai ke aplikasi yang tepat, pada komputer yang tepat, tanpa kesalahan.

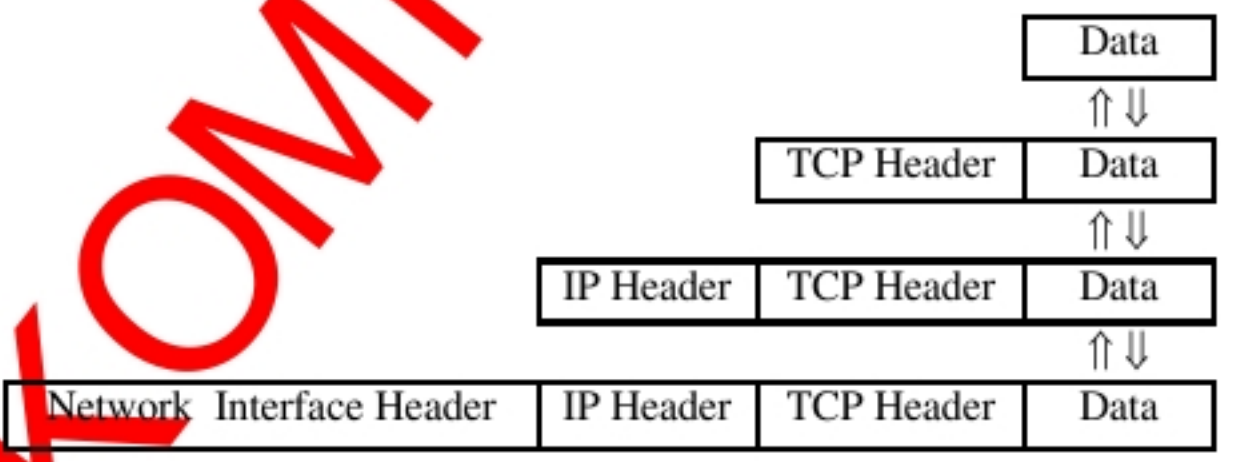
Cara alamiah untuk menghadapi setiap masalah yang rumit ialah memecahkan masalah tersebut menjadi bagian yang lebih kecil. Dalam memecahkan masalah transfer data di atas, para ahli jaringan komputer pun melakukan hal yang sama. Untuk setiap masalah komunikasi data, diciptakan solusi khusus berupa aturan-aturan untuk menangani masalah tersebut. Untuk menangani semua masalah komunikasi data, keseluruhan aturan ini harus bekerja sama satu dengan lainnya. Sekumpulan aturan untuk mengatur proses pengiriman data ini disebut sebagai protokol komunikasi data. Protokol ini diimplementasikan dalam bentuk program komputer / perangkat lunak (*software*) yang terdapat pada komputer dan peralatan komunikasi data lainnya.

TCP/IP adalah sekumpulan protokol yang didesain untuk melakukan fungsi-fungsi komunikasi data pada *Wide Area Network* (WAN). TCP/IP terdiri atas sekumpulan protokol yang masing-masing bertanggung jawab atas bagian-bagian tertentu dari komunikasi data. Berkat prinsip ini, tugas masing-masing protokol menjadi jelas dan sederhana. Protokol yang satu tidak perlu mengetahui cara kerja protokol yang lain, sepanjang ia masih bisa saling mengirim dan menerima data.

Berkat penggunaan prinsip ini, TCP/IP menjadi protokol komunikasi data yang fleksibel. Protokol TCP/IP dapat diterapkan dengan mudah di setiap jenis komputer dan interface jaringan, karena sebagian besar isi kumpulan protokol ini tidak spesifik terhadap satu komputer atau peralatan jaringan tertentu. Agar TCP/IP dapat berjalan di atas interface jaringan tertentu, hanya perlu dilakukan perubahan pada protokol yang berhubungan dengan interface jaringan saja.

Dalam TCP/IP, terjadi penyampaian data dari protokol yang berada di satu layer ke protokol yang berada di satu layer ke protokol yang berada di layer yang lain. Setiap protokol memperlakukan semua informasi yang diterimanya dari protokol lain sebagai data.

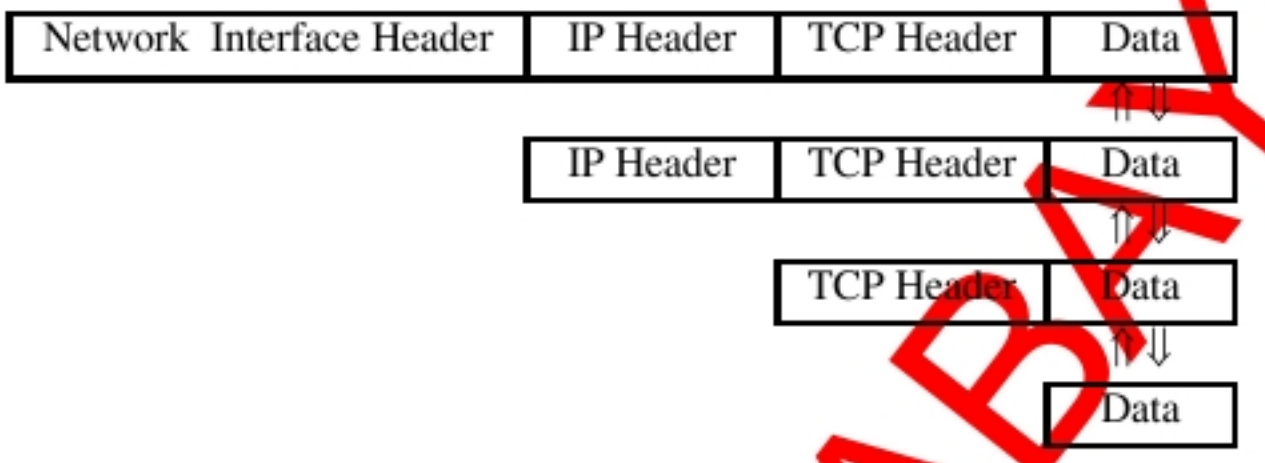
Jika suatu protokol menerima data dari protokol lain di layer atasnya, ia akan menambahkan informasi tambahan miliknya ke data tersebut. Informasi ini memiliki fungsi yang sesuai dengan fungsi protokol tersebut. Setelah itu, data ini diteruskan lagi ke protokol pada layer di bawahnya.



Gambar 2.1. Proses Enkapsulasi Data antar Layer (pihak pengirim data)

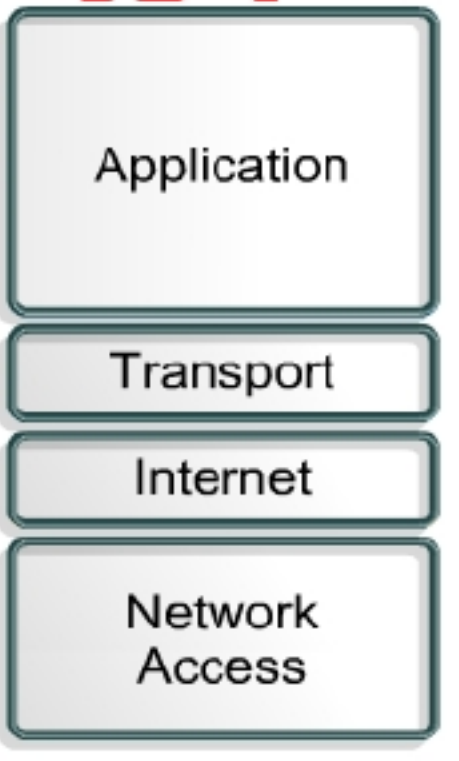
Hal yang sebaliknya terjadi jika suatu protokol menerima data dari protokol lain yang berada pada layer di bawahnya. Jika data ini dianggap valid,

protokol akan melepas informasi tambahan tersebut, untuk kemudian meneruskan data itu ke protokol lain yang berada pada layer diatasnya.(Purbo, O.W., 2001:21)



Gambar 2.2. Proses demultiplexing data antar layer (pihak penerima data)

Sekumpulan protokol TCP/IP ini dimodelkan dengan empat layer TCP/IP, sebagaimana terlihat pada gambar berikut:



Gambar 2.3 Model TCP/IP

A. Network Access Layer

Network access layer adalah layer paling bawah dari model TCP/IP. Layer ini terdiri dari protokol-protokol yang digunakan komputer untuk mengirimkan data ke komputer dan piranti lain yang tersambung dalam jaringan. Protokol yang ada dalam layer ini memiliki dua fungsi sebagai berikut:

1. Melakukan enkapsulasi datagram IP menjadi *frames* yang dikirimkan oleh jaringan.
2. Memetakan alamat IP ke alamat fisik yang digunakan oleh jaringan.

Tidak seperti layer di atasnya, protokol pada *network access layer* harus mengetahui secara detail jaringan fisik yang ada dibawahnya. Beberapa hal yang harus diketahui misalnya, struktur paket, ukuran frame maksimum, dan skema alamat fisik yang digunakan. Tujuannya adalah untuk memastikan agar protokol-protokol ini dapat menjalankan proses terhadap data secara benar sehingga bisa dikirimkan melewati jaringan. (Heywood, D. 1996)

B. Internet Layer

Pada layer ini terdapat *Internet Protocol* atau yang disebut dengan IP yang merupakan inti dari protokol TCP/IP. Seluruh data yang berasal dari protokol pada layer di atas IP harus dilewatkan, ialah oleh protokol IP, dan dipancarkan sebagai paket IP, agar sampai ke tujuan. Dalam melakukan pengiriman data, IP memiliki sifat yang dikenal sebagai *unreliable, connectionless, datagram delivery service*.

Unreliable berarti bahwa protokol IP tidak menjamin datagram yang dikirim pasti akan sampai ke tempat tujuan. Protokol IP hanya berjanji ia akan

melakukan usaha sebaik-baiknya (*best effort delivery service*), agar paket yang dikirim tersebut sampai ke tujuan. Jika di perjalanan terjadi hal-hal yang diinginkan (salah satu jalur putus, router down, atau host/network tujuan sedang down), protokol IP hanya memberitahukan ke pengirim paket melalui protokol Internet Control Message Protocol (ICMP), bahwa terjadi masalah dalam pengiriman paket IP ke tujuan.

Jika diinginkan kehandalan yang lebih baik, kehandalan itu harus disediakan oleh protokol yang berada diatas layer IP ini yaitu : Transmission Control Protocol (TCP) dan *application layer*. *Connectionless* berarti dalam mengirim paket dari tempat asal ke tujuan, pihak pengirim dan penerima paket IP sama sekali tidak mengadakan proses penetapan koneksi (*handshake*) terlebih dahulu.

Datagram delivery service berarti setiap paket data yang dikirim adalah independen terhadap paket data yang lain. Akibatnya jalur yang ditempuh oleh masing-masing paket data IP ke tujuannya bisa jadi berbeda satu dengan yang lainnya. Karena jalur yang ditempuh berbeda, kedatangan paket pun bisa jadi tidak berurutan. Hal ini dilakukan untuk menjamin tetap sampainya paket IP ke tujuan, walaupun salah satu jalur ke tujuan itu mengalami masalah.

Setiap paket IP membawa data yang terdiri atas:

1. *Version*, berisi versi dari protokol yang dipakai. Saat ini yang dipakai ialah IP versi 4.
2. *Header Length*, berisi panjang dari header paket IP dalam hitungan 32 bit word.
3. *Type of Service*, berisi kualitas service yang dapat mempengaruhi cara

penanganan paket IP ini.

4. *Total Length of Datagram*, panjang IP datagram dalam ukuran byte.
5. *Identification, Flags, dan Fragment Offset*, berisi beberapa data yang berhubungan dengan fragmentasi paket. Paket yang yang dilewatkan melalui berbagai jenis jalur akan mengalami fragmentasi (dipecah menjadi beberapa paket yang lebih kecil) sesuai dengan besar data maksimal yang bisa ditransmisikan melalui jalur tersebut.
6. *Time to Live,* berisi jumlah router/hop maksimal yang boleh dilewati paket IP. Setiap kali melewati satu router, isi dari field ini dikurangi satu. Jika TTL telah habis dan paket tetap belum sampai ke tujuan, paket ini akan dibuang dan router terakhir akan mengirimkan paket ICMP *time exceeded*. Hal ini dilakukan untuk mencegah paket IP terus menerus berada di dalam *network*.
7. *Protocol,* mengandung angka yang mengidentifikasikan protokol *layer* atas pengguna isi data dari paket IP ini.
8. *Header Checksum,* berisi nilai *checksum* yang dihitung dari seluruh field dari header packet IP. Sebelum dikirimkan, protokol IP terlebih dahulu menghitung *checksum* dari header paket IP tersebut untuk nantinya dihitung kembali di sisi penerima. Jika terjadi perbedaan, maka paket ini dianggap rusak dan dibuang.
9. *IP Address* pengirim dan penerima data.

   IP Address ini dikelompokkan dalam lima kelas, tiga diantaranya yaitu :

   a. Kelas A

      Format : 0nnnnnnn hhhhhhhh hhhhhhhh hhhhhhhh

      Byte Pertama : 0 - 127 (127 untuk *local loopback*)

Jumlah : 126 kelas A (0 dan 127 dicadangkan)

Range IP : 1.xxx.xxx.xxx sampai 126.xxx.xxx.xxx

Jumlah IP: 16.777.214 IP Address untuk tiap kelas A

b. Kelas B

Format : 10nnnnnn nnnnnnnn hhhhhhhh hhhhhhhh

Byte Pertama : 128 - 191

Jumlah : 16384 kelas B

Range IP : 128.0.xxx.xxx sampai 191.155.xxx.xxx

Jumlah IP: 65.532 IP Address untuk tiap kelas B

c. Kelas C

Format : 110nnnn nnnnnnnn nnnnnnnn hhhhhhhh

Byte Pertama : 192 - 223

Jumlah : 2.097.152 Kelas C

Range IP : 192.0.0.xxx sampai 223.255.255.xxx

Jumlah IP: 254 IP Address untuk tiap kelas C

Keterangan: n = network bit, h = host bit, m = multicast bit, r = bit cadangan

C. *Transport Layer*

Transport layer mempunyai dua fungsi yaitu mengatur aliran data antara dua *host* dan *reliability*.

Pada *transport layer* terdapat dua buah protokol:

1. *TCP -- a connection-oriented, reliable protocol, byte stream service.* *Connection Oriented* berarti sebelum melakukan pertukaran data, dua aplikasi pengguna TCP harus melakukan proses penetapan koneksi (*handshake*)

terlebih dahulu. *Reliable* berarti TCP menerapkan proses deteksi kesalahan paket dan retransmisi. *Byte Stream Service* berarti paket dikirimkan dan sampai ke tujuan secara berurutan.

2. *UDP -- connectionless and unreliable*. Walaupun bertanggung jawab untuk mentransmisikan pesan/data, tidak ada proses yang dilakukan guna melakukan pengecekan kesalahan diantara setiap segmen yang dilakukan oleh layer ini apakah data telah terkirim dan tiba ditujuan. Keuntungan penggunaan UDP adalah kecepatannya karena pada UDP tidak ada *acknowledgements* yang berfungsi sebagai informasi untuk menunjukkan apakah data telah terkirim dan tiba di tujuan tanpa danya kesalahan, sehingga trafik yang lewat jaringan rendah karena tidak diperlukan adanya pengiriman *acknowledgements* pada sisi pengirim, dan itu yang membuat UDP lebih cepat daripada TCP.

D. *Application Layer*

Application layer terdapat di puncak model TCP/IP yang berfungsi untuk memberikan pelayanan kepada pemakai komputer untuk dapat berkomunikasi dengan komputer lain melalui jaringan menggunakan protokol seperti TELNET, *File Transfer Protocol* (FTP), *Simple Mail Transfer Protocol* (SMTP), *Domain Name Service* (DNS).(Heywood, D. 1996)

2.3 Protokol HTTP

Hypertext Transfer Protocol (HTTP) merupakan protokol yang berbasis *request* dan *response*, yaitu *browser* akan mengirimkan *request* ke server untuk meminta informasi atau layanan lainnya, dan server melakukan *response* untuk pelayanannya. HTTP akan menentukan tipe – tipe *request* yang akan dikirim

client ke server, dan sebagai protokol juga menentukan bagaimana request dan response dijalankan. Http/1.0 menentukan tiga request yaitu GET, POST, dan HEAD, sedangkan Http/1.1 mempunyai lima method *request* tambahan yaitu OPTIONs, PUT, TRACE, DELETE dan CONNECT. Akan tetapi yang paling sering digunakan (umumnya) ialah method GET dan POST (Andrew S. Tanenbaum, 2000).

1. **Method GET :** merupakan method yang termudah dan paling sering digunakan untuk mengakses sumber static seperti HTML, gambar, dan lain – lain. sedangkan untuk mengambil informasi yang dinamis menggunakan parameter *query* tambahan pada URL yang diminta, sebagai contoh http://local*host*:8080/postget?nama=novan.
2. **Method POST :** Digunakan untuk mengakses sumber yang dinamis, pada umumnya POST digunakan untuk mengirim informasi yang bergantung pada permintaan, dan juga digunakan ketika harus mengirim informasi komplek yang besar ke server. Perbedaannya dengan method GET adalah pada method GET parameter request dikirim sebagai sebuah query string yang ditambahkan ke URL, sedangkan pada method POST parameter request dikirim di dalam tubuh request.

Response dari request HTTP adalah server merespon dengan status response dan beberapa meta informasi yang menjelaskan mengenai response tersebut. Pada HTTP, antara server dan client menggunakan Multi Purpose Internet Mail Extension (MIME) untuk mengindikasikan tipe dari isi request dan response. Web browser mengenali kode status yang digunakan untuk menjelaskan hasil respose yang terjadi kepada user. Kode status tersebut adalah kode 200

(SC_ACCEPTED) merupakan kode status OK (semuanya berjalan baik), kode 403 (SC_FORBIDDEN) merupakan kode FORBIDDEN yang menunjukkan bahwa user tidak berhak mengakses document yang diminta berkaitan dengan permission pada server, kode 404 (SC_NOT_FOUND) menunjukkan bahwa dokumen yang diminta tidak ditemukan, dan kode 500 (SC_INTERNAL_SERVER_ERROR) yang menunjukkan bahwa terjadi error pada internal server (Tanenbaum, A.S. 2000).

2.4 Kriptografi

Kriptografi merupakan ilmu dan seni untuk menjaga pesan agar aman. (*Cryptography is the art and science of keeping messages secure.*) "*Crypto*" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan). Para pelaku atau praktisi kriptografi disebut ***cryptographers***. Sebuah algoritma kriptografik (*cryptographic algorithm*), disebut ***cipher***, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat.

Proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut *Plaintext*) menjadi pesan yang tersembunyi (disebut *ciphertext*) adalah **enkripsi** (*encryption*). *Ciphertext* adalah pesan yang sudah tidak dapat dibaca dengan mudah. Menurut ISO 7498-2, terminologi yang lebih tepat digunakan adalah "*encipher*".

Proses sebaliknya, untuk mengubah *ciphertext* menjadi *Plaintext*, disebut **dekripsi** (*decryption*). Menurut ISO 7498-2, terminologi yang lebih tepat untuk

1. Algoritma dari Enkripsi dan Dekripsi

Algoritma dari enkripsi adalah fungsi-fungsi yang digunakan untuk melakukan fungsi enkripsi dan dekripsi. Algoritma yang digunakan menentukan kekuatan dari enkripsi, dan ini biasanya dibuktikan dengan basis matematika.

Berdasarkan cara memproses teks (*plaintext*), *cipher* dapat dikategorikan menjadi dua jenis: *block cipher* and *stream cipher*. Block *cipher* bekerja dengan memproses data secara blok, dimana beberapa karakter / data digabungkan menjadi satu blok. Setiap proses satu blok menghasilkan keluaran satu blok juga. Sementara itu *stream cipher* bekerja memproses masukan (karakter atau data) secara terus menerus dan menghasilkan data pada saat yang bersamaan.

2. Kunci yang digunakan dan panjangnya kunci

Kekuatan dari penyandian bergantung kepada kunci yang digunakan. Beberapa algoritma enkripsi memiliki kelemahan pada kunci yang digunakan. Untuk itu, kunci yang lemah tersebut tidak boleh digunakan. Selain itu, panjangnya kunci, yang biasanya dalam ukuran *bit*, juga menentukan kekuatan dari enkripsi. Kunci yang lebih panjang biasanya lebih aman dari kunci yang pendek. Jadi enkripsi dengan menggunakan kunci 128-bit lebih sukar dipecahkan dengan algoritma enkripsi yang sama tetapi dengan kunci 56-bit.

Semakin panjang sebuah kunci, semakin besar *keyspace* yang harus dijalani untuk mencari kunci dengan cara *brute force attack* atau coba-coba karena *keyspace* yang harus dilihat merupakan pangkat dari bilangan 2. Jadi kunci 128-bit memiliki *keyspace* 2^{128}, sedangkan kunci 56-bit memiliki *keyspace* 2^{56}. Artinya semakin lama kunci baru bisa ketahuan.

3. *Plaintext*

Plaintext adalah pesan atau informasi yang akan dikirimkan dalam format yang mudah dibaca atau dalam bentuk aslinya.

4. *Ciphertext*

Ciphertext adalah informasi yang sudah dienkripsi. Kembali ke masalah algoritma, keamanan sebuah algoritma yang digunakan dalam enkripsi atau dekripsi bergantung kepada beberapa aspek. Salah satu aspek yang cukup penting adalah sifat algoritma yang digunakan. Apabila kekuatan dari sebuah algoritma sangat tergantung kepada pengetahuan (tahu atau tidaknya) orang terhadap algoritma yang digunakan, maka algoritma tersebut disebut "*restricted algorithm*".

Apabila algoritma tersebut bocor atau ketahuan oleh orang banyak, maka pesan-pesan dapat terbaca. Tentunya hal ini masih bergantung kepada adanya kriptografer yang baik. Jika tidak ada yang tahu, maka sistem tersebut dapat dianggap aman (meskipun semu). Meskipun kurang aman, metoda pengamanan dengan *restricted algorithm* ini cukup banyak digunakan karena mudah implementasinya dan tidak perlu diuji secara mendalam. Contoh penggunaan metoda ini adalah enkripsi yang menggantikan huruf yang digunakan untuk mengirim pesan dengan huruf lain. Ini disebut dengan "*substitution cipher*".

2.6 Symmetric Encryption Algorithm

Symmetric cipher ada dengan dua type yaitu : block cipher dan stream cipher. Sebuah block cipher melakukan enkripsi satu block data dalam satu waktu

biasanya 64 bit tapi beberapa algoritma enkripsi menggunakan 128 bit. Stream cipher malakukan enkripsi data setiap bit data secara satu persatu pada satu waktu.

Sebuah block cipher dapat digunakan untuk membuat sebuah stream cipher dan sebaliknya, tapi akan lebih efisien menggunakan tipe cipher yang sesuai dengan aplikasi yang dibuat. Jika anda ingin mengenkripsi satu pesan tunggal, gunakan block cipher namun jika informasi tersebut adalah stream yang konstan maka kemungkinan terbaik adalah menggunakan stream cipher.

Kekuatan dari algoritma symmetric didasarkan oleh besar panjang kunci yang digunakan. Semakin panjang kunci maka semakin sulit data untuk untuk dipecahkan. Panjang kunci direpresentasikan dalam satuan bit, biasanya panjang kunci bervariasi antara 40 sampai dengan 448 bit.

Dalam usaha untuk memecahkan sebuah pesan yang terenkripsi seorang penyerang dapat mencoba untuk mendapatkan kunci dengan metode yang disebut dengan brute-force yaitu suatu teknik untuk mendapatkan key enkripsi dengan mencoba segala kemungkinan kombinasi kunci, dengan cara tersebut akan dicoba berbagai macam kemungkinan kunci yang dapat digunakan untuk mendekripsi pesan.

Kumpulan kunci yang mungkin dapat dihasilkan disebut key space. Untuk kunci 40-bit, terdapat 2^{40} kemungkinan kunci sedangkan untuk kunci 128-bit berate memiliki 2^{128} kemungkinan kunci yang dapat dihasilkan. Setiap bit yang ditambahkan kedalam panjang kunci meningkatkan keamanan

"While it's possible for a supercomputer to break a 56-bit key in well under 24 hours, a 128-bit key will take 2^{72} times longer to crack by brute force,

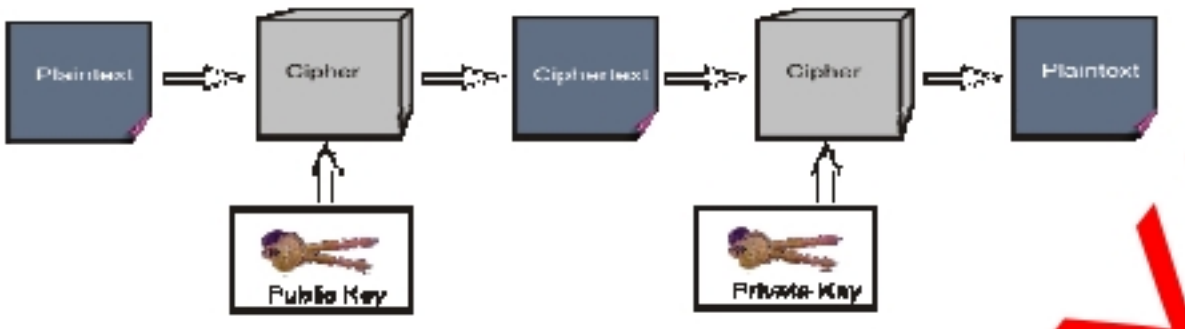
longer than the age of the universe ". **(Jess Garms & Daniel Somerfield, 2001:35)**

Algoritma symmetric enkripsi bekerja dengan cara yang sama : kunci yang sama digunakan untuk enkripsi dan dekripsi pesan. Untuk mengirim pesan rahasia ke seseorang harus ada persetujuan sebelumnya mengenai kunci yang akan digunakan secara bersama-sama. Salah satu masalah symmetric encryption adalah bagaimana sebuah kunci tetap terjaga kerahasiaannya ketika dikirimkan/digunakan kepada pihak lain. Beberapa contoh algoritma symmetric encryption adalah DES/Triple DES, BlowFish, RC4, AES, IDEA dan yang lainnya.

2.7 Asymmetric Encryption Algorithm

Asymmetric encryption yang juga dikenal sebagai public key encryption, menyelesaikan masalah yang dibutuhkan untuk bagaimana caranya agar kunci yang digunakan tetap terjaga kerahasiaannya, ketika digunakan secara bersama-sama sebelum mengirim sebuah pesan.

Berbeda dengan symmetric encryption yang menggunakan sebuah kunci yang sama untuk proses enkripsi dan dekripsi pesan. Asymmetric encryption contohnya algoritma RSA yang diterapkan pada tugas akhir ini menggunakan dua buah kunci, kunci public digunakan untuk mengenkripsi dan kunci privat untuk dekripsi. Berikut pada gambar 2.4 diagram kerja asymmetric encryption :



Gambar 2.4. Diagram asymmetric encryption

Pesan yang dienkripsi menggunakan kunci publik hanya dapat didekripsi menggunakan kunci privat yang sesuai. Demikian kunci public tidak dapat memberikan segala informasi tentang kunci privat sehingga kunci public dapat dipublikasikan dan siapapun dapat membacanya tanpa adanya kekhawatiran akan bahaya terbacanya pesan.

Seseorang yang ingin menggunakan asymmetric encryption hanya harus menjaga kerahasiaan kunci privat dan dapat mempublikasikan kunci public kepada pihak lain untuk digunakan. Setiap orang dapat mengenkripsikan pesan menggunakan kunci public milik orang tersebut dan hanya orang tertentu yang memiliki pasangan kunci privat yang dapat mendekripsi pesan.

Assymmetric encryption memerlukan panjang kunci yang jauh lebih besar dibandingkan dengan symmetric encryption. Sebuah 1024-bit kunci asymmetric encryption memberikan keamanan setara dengan 128-bit kunci symmetric. Adapun kelemahan didalam public key cryptography adalah :

A. Permasalahan bagaimana cara agar kunci public yang dibuat menjadi sulit untuk dipecahkan / digunakan tanpa memiliki pasangan kunci privat .
B. Asymmetric encryption lebih lambat jika dibandingkan symmetric encryption

Beberapa contoh algoritma asymmetric encryption adalah : RSA, El Gamal dan Rijndael.

2.8 RSA *Encryption*.

RSA adalah algoritma enkripsi asimetri yang berarti memiliki dua kunci yaitu kunci publik dan kunci privat. Untuk membangkitkan kedua kunci, pilihlah dua bilangan acak yang sangat besar, **p** dan ***q*** .Untuk mendapatkan keamanan yang maksimum, pilih dua bilangan **p** dan **q** hampir sama besarnya.

Hitung $n = p * q$ **(3)**

Kemudian secara acak, pilihlah kunci enkripsi **e**, sedemikian sehingga **e** dan **(p - 1)(q - 1)** relatif prima artinya **e** dan **(p - 1)(q - 1)** tidak memiliki faktor persekutuan bersama. Kemudian dengan algoritma Euclidan yang diperluas, hitunglah kunci dekripsi **d**, sedemikian sehingga:

$$e.d = 1 \bmod (p-1)(q-1) \quad \textbf{(4)}$$

atau

$$e.d - 1 = k(p-1)(q-1) \quad \textbf{(5)}$$

Di mana **k** adalah konstanta integer.

Perhatikan bahwa **d** dan **n** juga relatif prima. Bilangan **e** dan **n** merupakan kunci publik, sedangkan **d** kunci privat. Dua bilangan prima **p** dan **q** tidak diperlukan lagi. Namun **p** dan **q** kadang diperlukan untuk mempercepat perhitungan dekripsi.

Untuk mengenkripsi pesan **m**, pertama-tama bagi pesan ke dalam blok-blok numerik yang lebih kecil dari **n** (dengan data biner, pilih pangkat terbesar dari 2 yang kurang dari **n**). Jadi bila **p** dan **q** bilangan prima 100 digit, maka **n** akan memiliki sekitar 200 digit dan setiap blok pesan **m**, seharusnya kurang dari 200 digit panjangnya.

Jika anda perlu mengenkripsi sejumlah blok pesan yang tetap, anda dapat menambahkan sejumlah blok pesan yang tetap, anda dapat menambahkan beberapa bit 0 di sebelah kiri bilangan untuk menjamin bahwa pesan selalu kurang dari **n**. Pesan yang terenkrip **c**, akan tersusun dari blok-blok $\mathbf{c_i}$ yang hampir sama panjangnya.

Rumus enkripsinya adalah:

$$\mathbf{c_i = c_i^e \bmod n} \qquad \mathbf{(6)}$$

Untuk mendekrip pesan, ambil setiap blok pesan terenkrip $\mathbf{c_i}$ dan hitung

$$\mathbf{m_i = c_i^d \bmod n} \qquad \mathbf{(7)}$$

Contoh:

Misalkan p =409 dan q = 521

n = p * q = 213089

$\phi(n) = (p-1)(q-1) = 212160$

e = gcd{ $\phi(n)$,e} = 1;1<e< $\phi(n)$ (gcd=great common divisor atau pembagi bersama terbesar)

e = 27307 (e adalah bilangan integer yang dipilih dan relatif prima terhadap nilai $\phi(n)$)

$d = e^{-1} \bmod \phi(n) = 177283$

Misalkan pesan yang akan kita enkripsi dapat dinyatakan sebagai bilangan

m = 38901794325

Karena m harus lebih kecil dari n dan n 6 digit, maka kita ambil 5 digit per blok pesan,sehingga:

$m_1 = 38901$

$m_2 = 79432$

$m_3 = 00005$

$c_1 = m_1^e \bmod n = 38901^{27307} \bmod 213089 = 186604$

$c_2 = m_2^e \bmod n = 79432^{27307} \bmod 213089 = 80731$

$c_3 = m_3^e \bmod n = 5^{27307} \bmod 213089 = 201081$

sehingga c = 186604807312010 81

Proses dekripsi akan menjadi sebagai berikut:

$m_1 = c_1^e \bmod n = 186604^{27307} \bmod 213089 = 38901$

$m_2 = c_2^e \bmod n = 80731^{27307} \bmod 213089 = 79432$

$m_3 = c_3^e \bmod n = 201081^{27307} \bmod 213089 = 5$

Untuk dapat memecahkan pesan m dari sandi c-nya, penyerang harus dapat menemukan nilai d yang merupakan kunci privat yang hanya diketahui pemiliknya saja. Penyerang dengan mudah dapat menemukan nilai e dan n yang merupakan kunci public. Untuk menemukan d, diperlukan pemfaktoran agar diperoleh p dan q (karena pq = n) yang selanjutnya akan diperoleh d dari persamaan ed = 1 mod (p-1)(q-1). Pemfaktoran n menjadi p dan q ini dipercaya mustahil dilakukan bila n bernilai sangat besar, misalnya berorde 2^{4096}.

Misalkan perhatikan contoh lain berikut ini :

p = 61 q = 53 p.q = 3233

e = 17 d = 2753

Kunci publik(pq,e)

Kunci privat d

$C = \text{enkrip}(T) = (T^{17}) \bmod 3233$

$T = \text{dekrip}(C) = (C^{2753}) \bmod 3233$

$\text{Encrypt}(123) = (123^{17}) \bmod 3233$

= 337587917446653715596592958817679803 mod 3233 = 855

Decrypt(855) = (855^{2753}) mod 3233

= 5.043288895841606873442289912739 4e+8071 mod 3233

= 123

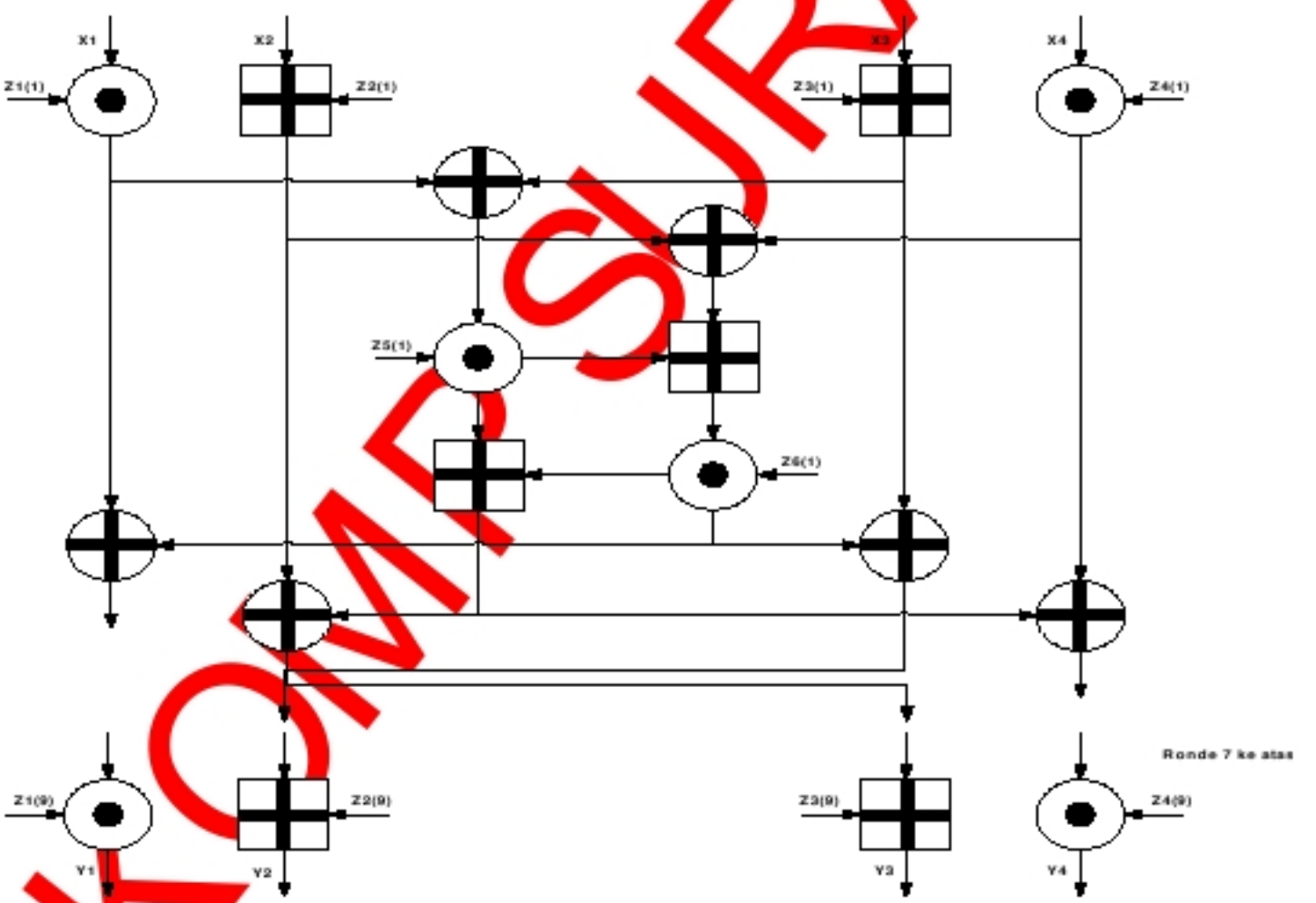
2.9 IDEA *Encryption*

IDEA adalah algoritma enkripsi simetri atau algoritma konvensional yang berarti menggunakan satu kunci enkripsi dan kunci dekripsi yang sama. IDEA merupakan *cypher blok* yang beroperasi pada blok *plaintext* 64-bit. Panjang kuncinya 128 bit. Algoritma yang sama digunakan untuk enkripsi dan deripsi. Sebagaimana algorima enkripsi yang lain, IDEA menggunakan permutasi dan substitusi untuk *confusion* dan *difusion*, IDEA menggunakan operasi aljabar yang tidak kompatibel sebagi berikut:

1. XOR
2. Penambahan modulo 2^{16}
3. Perkalian modulo $2^{16} + 1$ (operasi ini menggantikan kotak-S)

Kotak –S adalah sebuah daftar tabel berisi nilai tertentu yang telah ditetapkan oleh pihak tertentu yang digunakan untuk proses substitusi pada enkripsi data, algoritma enkripsi DES merupakan contoh teknik enkripsi yang menggunakan kotak-S untuk proses substitusi saat proses enkripsi data, kotak-S tersebut dibuat berdasarkan saran dari dari lembaga NSA sehingga pengguna tidak bisa yakin bahwa struktur internal algoritma DES terbebas dari titik-titik lemah yang mungkin dapat membuka cipher.

Pada algoritma enkripsi IDEA. Blok data 64 bit dibagi menjadi 4 sub-blok 16 bit X_1,X_2, X_3, X_4. Empat sub-blok ini menjadi masukan bagi iterasi tahap pertama algoritma. Total terdapat 8 iterasi. Pada setiap iterasi, 4 sub-blok di XOR-kan, ditambahkan, dikalikan dengan yang lain dengan 6 sub-kunci 16 bit. Di antara iterasi, sub blok kedua dan ketiga saling dipertukarkan. Akhirnya 4 sub-blok dikombinasikan dengan 4 sub-kunci dalam transformasi keluaran. Perlu diperhatikan, dalam beberapa literatur mengenai IDEA, kunci K sering disebut sebagai Z, sebagai mana pula dalam gambar 2.5 di bawah.



Gambar 2.5 Diagram blok IDEA (Kurniawan, Yusuf. 2004:72).

Pada setiap tahapan urutan berikut ini dikerjakan:

1. Kalikan X_1 dengan sub-kunci pertama Z1(1) atau K1(1).
2. Tambahkan X_2 dengan sub-kunci kedua.

3. Tambahkan X_3 dengan sub-kunci ketiga.
4. Kalikan X_4 dengan sub-kunci keempat.
5. XOR hasil langkah (1) dengan (3).
6. XOR hasil langkah (2) dengan (4).
7. Kalikan hasil langkah (5) dengan sub-kunci kelima.
8. Tambahkan hasil langkah (6) dan (7).
9. Kalikan hasil langkah (8) dengan sub-kunci ke-6.
10. Tambahkan hasil langkah (7) dan (9).
11. XOR hasil langkah (1) dengan (9).
12. XOR hasil langkah (3) dengan (9).
13. XOR hasil langkah (2) dengan (10).
14. XOR hasil langkah (4) dengan (10).

Keluaran setiap tahapan adalah 4 sub-blok yang merupakan hasil langkah (11), (12), (13) dan (14). Pertukarkan 2 sub-blok dalam (blok 2 dan 3) kecuali tahapan terakhir, dan itu adalah input untuk tahapan berikutnya.

Setelah tahapan ke-8, terdapat transformasi keluaran akhir.

1. Kalikan X_1 dengan sub-kunci pertama.
2. Tambahkan X_2 dengan sub-kunci kedua.
3. Tambahkan X_3 dengan sub-kunci ketiga.
4. Kalikan X_4 dengan sub-kunci ke-4.

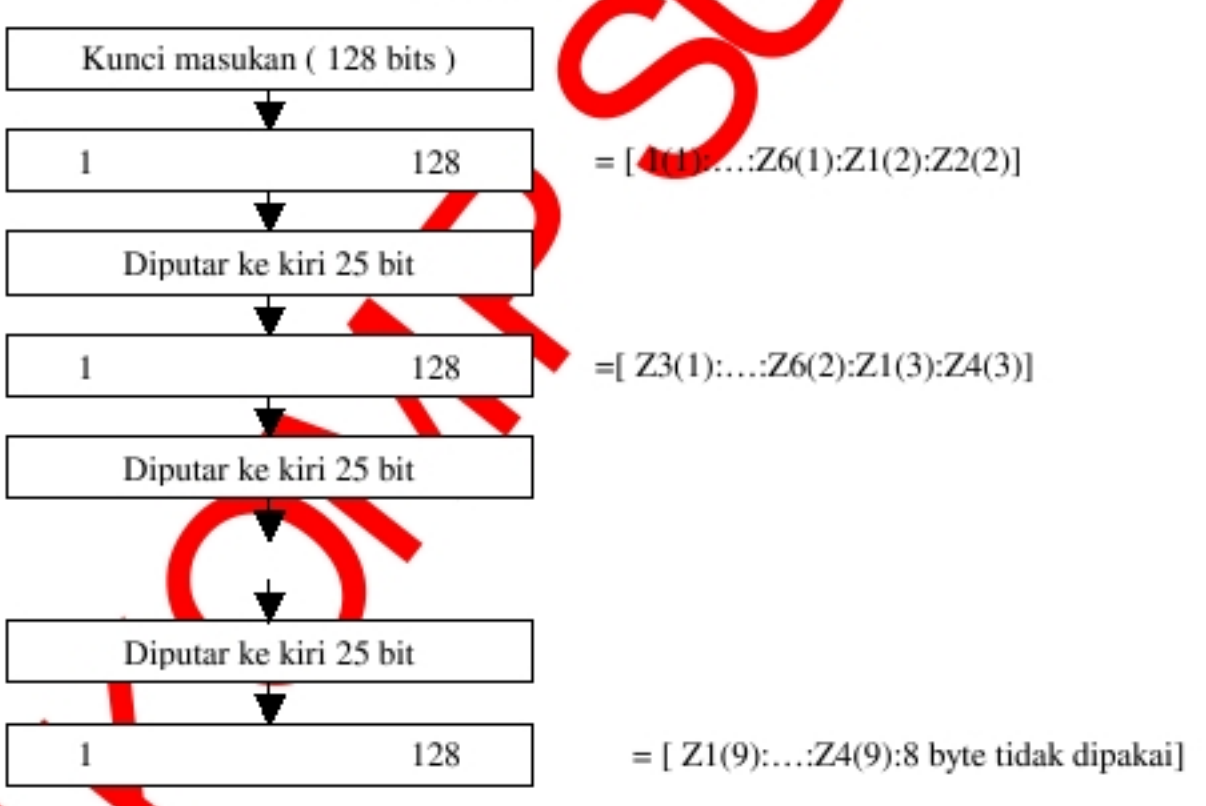
Akhirnya 4 sub-blok digabungkan kembali untuk menghasilkan *cipher*text.

Membuat sub-kunci tidak sulit. Algoritma menggunakan 52 sub-kunci (6 untuk setiap tahapan dari 8 tahapan, dan 4 lebih untuk transformasi keluaran). Sehingga jumlah sub-kunci total = 6 * 8 + 4 = 52 buah. Satu sub-kunci berukuran

16 bit. Pertama, kunci 128 bit dibagi ke dalam 8 sub-kunci 16 bit (ingat 8*16 = 128 bit). Tersedia 8 sub-kunci tahap pertama dan 2 sub-kunci berikutnya untuk iterasike-2. Jadi pada tahap ke-2 ini, terdapt kekurangan 4 sub-kunci.

Kemudian kunci digeser ke kiri 25 bit secara memutar (rotasi) dan kemudian dibagi menjadi 8 subkunci. Empat sub-kunci pertama digunakan untuk iterasi ke-2, melengkapi kekurangan sebelumnya. Empat sub-kunci berikutnya untuk iterasi / tahapan ke-3. Kunci 128 bit dirotasi lagi ke kiri sebanyak 25 bit untuk mendapatkan 8 sub-kunci berikutnya, dan seterusnya sampai berakhirnya algoritma. Adapun *key schedule* untuk algoritma IDEA dapat digambarkan sebagaimana pada gambar 2.6 di bawah.

Key Schedule untuk IDEA



Gambar 2.6 *Key Schedule* untuk Algoritma IDEA (Kurniawan, Yusuf. 2004:73)

2.9.1 Dekripsi IDEA

Dekripsi dilakukan dengan cara serupa dengan proses enkripsi kecuali pembentukan sub-kuncinya yang berbeda. Blok *cipher* 64-bit dijadikan masukan dan keluarannya adalah *Plaintext*.

Empat sub-kunci pertama untuk dekripsi adalah:

K_d = Kunci dekripsi

K_d (1) = 1 / K (49) (inversi perkalian mod $2^{16} + 1$)

K_d (2) = - K (50) (inversi penjumlahan mod 2^{16})

K_d (3) = - K (51)

K_d (4) = 1 / K (52)

Pembentukan kunci dekripsi berikut berulang 8 kali, dengan menambahkan 6 ke indeks kunci dekripsi dan mengurangi 6 dari setiap indek kunci enkripsi.

K_d (5) = K (47)

K_d (6) = K (48)

K_d (7) = 1 / K (43)

K_d (8) = - K (45)

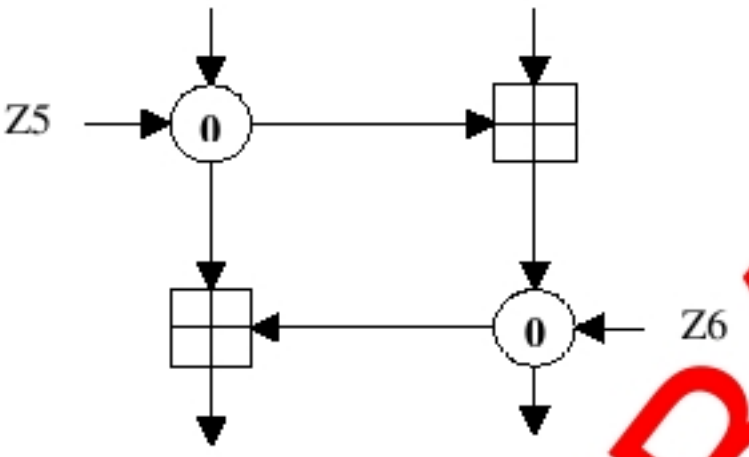
K_d (9) = - K (44)

K_d (10) = 1 / K (46)

2.9.2 Prinsip Desain IDEA

1. Kunci cukup panjang: 128 bit
2. *Confusion*: menggabungkan 3 kelompok operasi yang "tidak kompatibel" (XOR, penjumlahan mod 2^{16} dan perkalian mod 2^{17})

3. *Diffusion*: diberikan oleh kotak MA (*Multiply-Add*)



Gambar 2.7 Gambar Pembentukan *Subkey* IDEA (Kurniawan, Yusuf. 2004).

Dimana + adalah penjumlahan mod 2^{16} dan • adalah perkalian mod 2^{17}. Model ini merupakan jumlah komponen minimum yang diperlukan untuk menghasilkan difusi.

1. Keserupaan antara enkripsi dan dekripsi, perbedaan hanya pada *key schedule*.
2. Scalable: versi mini (2 bit, 4 bit atau simbol 8 bit) dapat dipelajari untuk analisis kelemahan.
3. Transparan: tidak tergantung pada tabel "misterius" kotak-S yang sering dicurigai memiliki "jalan belakang".
4. Cepat pada sofware dan hardware.
5. Sedapat mungkin dapat dibuktikan "aman", misalnya "aman sempurna" dalam satu ronde dengan kunci sekali pakai.

2.10 Performance algoritma enkripsi

Enkripsi data bukan hanya mempu memberikan rasa aman pengguna terhadap informasi yang dimiliki tetapi juga harus dapat memberikan performance yang terbaik di dalam melakukan proses *encrypt* maupun *decrypt*. Berikut adalah

daftar *performance* dari beberapa algoritma enkripsi yang ada yang akan digunakan dalam penelitian maupun yang sudah diterapkan pada penelitian sebelumnya.

Performance chart

Performance Comparison		IDEA [Mbit/sec]	AES [Mbit/sec]	3DES [Mbit/sec]	Source
Embedded Systems	C167/20MHz	0.22	0.06	0.03	Ascom Systec 2002, CH
	ARM7/20MHz	0.42	0.16	0.10	Ascom Systec 2002, CH
	MCF5397/ 90MHz	1.47	0.65	0.54	Ascom Systec 2002, CH
High-Level Languages	P-II/450MHz	24.38	17.06	7.04	Ascom Systec 2002, CH
	P-III/800MHZ	72.00	110.00	25.00	Johns Hopkins University, US
	P-4/2.0GHz	180.00	277.00	62.00	Johns Hopkins University, US
Assembler	P-II/450MHz	249.75	243.00	48.80	EncryptNT (IDEA), AU
	P-III/800MHZ	444.00	432.00	86.75	Lipmaa/Bosselaers, FI/ BE
	P-4/2.0GHz	2'050.00	1'080.00	216.89	EncryptNT (IDEA), AU
Hardware	VINCI/ 25MHz	178.00	n/a	43.33	Ascom Systec 1997/ The Free-IP Project
	IDEACrypt/ 40MHz	300.00	n/a	n/a	Ascom Systec 1997, CH
	Linesdancer/ 250MHz	3'600.00	1'208.00	1'017.33	Aspex Tech./The Free-IP Project, GB/US
	ASIC 120MHz	7'900.00	n/a	n/a	Development 2004

Gambar 2.8 Daftar *Performance* Algoritma IDEA

Gambar 2.8 di atas didapatkan berdasarkan artikel penelitian yang dikeluarkan oleh Media Crypt dengan judul "*IDEA performance overall*" yang diperoleh penulis melalui website www.mediacrypt.com. Dari gambar diatas menunjukkan bahwasannya algoritma enkripsi IDEA memiliki performance lebih baik jika dibandingkan dengan algoritma enkripsi seperti AES dan DES, hal ini

dapat disimpulkan dari jumlah output data yang dapat dienkripsi per detiknya (mbit/sec).

Footprints and performance

IDEA - International Data Encryption Algorithm

Symmetric block cipher algorithm with 64-bit data blocks and fixed-length 128-bit key.

Platform	Codesize [Byte]	Stacksize [Byte]	Datasize [Byte]	Performance [kbit/sec]
Infineon C167 20 MHz	2020	154	0	216
ARM7 TDMI 20 MHz	2388	204	0	416
ColdFire MCF5307 90	2528	344	0	1472

Computation speed and stacksize is given for ECB encryption. No additional memory must be accounted for the calculation context, the key and the data block. It is included in the stack size values shown. For completeness the memory taken up by these three objects is given - calculation context: 116 bytes, key: 16 bytes, data block: 8 bytes.

AES - Advanced Encryption Algorithm

Symmetric block cipher algorithm with 128-, 192- or 256-bit data blocks and 128-, 192- or 256-bit key.

Platform	Codesize [Byte]	Stacksize [Byte]	Datasize [Byte]	Performance [kbit/sec]
Infineon C167 20 MHz	4072	784	2079	56
ARM7 TDMI 20 MHz	2952	688	2079	160
ColdFire MCF5307 90	3414	628	2079	648

Computation speed and stacksize is given for ECB encryption, 256-bit data and 256-bit key. No additional memory must be accounted for the calculation context (ctx), the key and the data block. It is included in the stack size values shown. For completeness the memory taken up by these three objects is given - calculation context: 524 bytes, key: 32 bytes, data block: 32 bytes.

Gambar 2.9 Perbandingan *Performance* Algoritma IDEA

Gambar 2.9 di atas didapatkan berdasarkan artikel penelitian yang dikeluarkan oleh Media Crypt dengan judul "*IDEA performance embedded system*" yang diperoleh penulis melalui website www.mediacrypt.com. Dari gambar diatas menunjukkan bahwasannya algoritma enkripsi IDEA memiliki performance lebih baik jika dibandingkan dengan algoritma enkripsi seperti AES dan DES dalam hal penggunaan resource memory stack dan besaran data size yang dibutuhkan serta jumlah output data yang dapat dienkripsi per detiknya (kbit/sec).

Table 1. RSA Performance [milliseconds]

Measurement Point	33 MHz	50 MHz	66 MHz	83 MHz	100 MHz	133 MHz	166 MHz	333 MHz
512 bit modulus 512 bit exponent	8.09 [ms]	5.34 [ms]	4.05 [ms]	3.22 [ms]	2.67 [ms]	2.01 [ms]	1.61 [ms]	0.80 [ms]
512 bit modulus 3 bit exponent	0.43 [ms]	0.29 [ms]	0.22 [ms]	0.17 [ms]	0.14 [ms]	0.11 [ms]	0.09 [ms]	0.04 [ms]
1024 bit modulus 1024 bit exponent	54.71 [ms]	36.11 [ms]	27.35 [ms]	21.75 [ms]	18.05 [ms]	13.57 [ms]	10.86 [ms]	5.42 [ms]
1024 bit modulus 3 bit exponent	1.48 [ms]	0.97 [ms]	0.74 [ms]	0.59 [ms]	0.49 [ms]	0.37 [ms]	0.29 [ms]	0.15 [ms]
2048 bit modulus 2048 bit exponent	406.50 [ms]	268.29 [ms]	203.25 [ms]	161.62 [ms]	134.14 [ms]	100.86 [ms]	80.81 [ms]	40.28 [ms]
2048 bit modulus 3 bit exponent	5.40 [ms]	3.56 [ms]	2.70 [ms]	2.15 [ms]	1.78 [ms]	1.34 [ms]	1.07 [ms]	0.53 [ms]

These execution times can easily be converted into 'number of connections per second', (1/execution time = ops/sec), with the understanding that generation of protocol messages, transmission of messages, and receipt of responses are not included. The deterministic portion of connection set-up is the RSA sign/verify execution time (in milliseconds), which is provided.

Gambar 2.10 Daftar *Performance* Algoritma RSA

Gambar 2.10 didapatkan berdasarkan artikel penelitian yang dikeluarkan oleh FreeScale semi conductor dengan judul "*Understanding Public-Key Performance*" PublicKeyPerfWP rev 0.02/2005 yang ditulis oleh Matthew Short dan Geoffrey Waters yang diperoleh penulis melalui website www.freescale.com.

Dari gambar Gambar 2.10 menunjukkan bahwasannya algoritma enkripsi RSA memiliki performance yang berbeda-beda pada processor dengan clock rate dalam Mhz serta panjang kunci yang beragam, pengujian dilakukan berdasarkan jumlah waktu yang dibutuhkan untuk proses enkripsi dalam satuan miliseconds.

3.2 Efficiency analysis of the scheme

Finally, in [15] it is claimed that the chaos-based encryption-hash parallel algorithm "saves certain computation time when compared with traditional hashing and cryptographic methods". This assertion might be interpreted in the sense that their algorithm is faster than traditional hashing and cryptographic methods, when in fact it is several orders of magnitude slower. Table 1 of [10] gives some results to illustrate the performance of the proposed chaotic cryptographic and hashing algorithm. The performance depends on the values of p and r. The best speed achieved is between 7.7 and 11.5 KB/s in a 1.8 GHz processor. On the other hand, traditional encryption algorithms, such as DES or AES, achieve speeds of 21.3 and 61.0 MB/s respectively in a 2.1 GHz processor [20]. With respect to traditional hashing algorithms, MD5 and SHA-1, the two most widely used, achieve speeds of 216.6 and 67.9 MB/s respectively in a 2.1 GHz processor [20]. Thus, the claim is proved to be inadequate. As a consequence, this algorithm is also very inefficient (between 1,000 and 10,000 times slower) when compared to similar traditional algorithms.

Gambar 2.11 Hasil Analisa *Performance* Algoritma Chaotic

Gambar 2.11 didapatkan berdasarkan artikel penelitian yang ditulis oleh Gonzalo Alvarez di Instituto de Fisica Aplicada, Consejo Superior de Invetigaciones Cientificas, Serrano 144 – 28006 Madrid, Spain dengan judul "*Security problems with a chaos-based deniable authentication scheme*" yang diterbitkan pada arXiv:nlin.CD/0412023 v1 9 Dec 2004.

Dari hasil analisa yang dilakukan oleh Gonzalo Alvares menyimpulkan bahwasannya algoritma chaotic map tidaklah efisien jika dibandingkan dengan algoritma enkripsi yang serupa seperti AES dan DES, algoritma chaotic map memiliki performance antara 1.000 kali sampai dengan 10.000 kali lebih lambat

Tabel 1. Tabel *Performance* Algoritma Enkripsi

Algorithm	Megabytes(2^20 bytes) Processed	Time Taken	MB/Second
MD5	1.02e+003	4.726	216.674
SHA-1	256	3.766	67.977
MDC/MD5	256	5.377	47.610
Luby-Rackoff/MD5	64	4.307	14.860
DES	128	5.998	21.340
DES-XEX3	128	6.159	20.783
DES-EDE3	64	6.499	9.848
IDEA	64	3.375	18.963

Tabel 1 di atas, didapat berdasarkan penelitian yang ditulis oleh Weidai, yang diperoleh dari situs www.eskimo.com/~weidai/benchmark.html. Dari tabel diatas algoritma IDEA memiliki performance yang lebih baik jika dibandingkan dengan algoritma DES-EDE3 untuk jumlah data yang diproses sebesar 64 Mb

2.11 Bahasa JAVA

Java merupakan teknologi dimana teknologi tersebut mencakup java sebagai bahasa pemrograman yang memiliki sintaks dan aturan pemrograman tersendiri, juga mencakup java sebagai platform dimana teknologi ini memiliki virtual machine dan library yang diperlukan untuk menulis dan menjalankan program yang ditulis dengan pemrograman java.

Adapun kelebihan Java yaitu :

1. Java didesain untuk menghilangkan alokasi memori dan dealokasi memori secara manual. Java memiliki garbage collection otomatis yang mencegah adanya memory leak. Memory leak adalah masalah yang sering dihadapi programmer C dan C++ dimana memori yang digunakan untuk objek atau variable yang sudah tidak digunakan tidak didealokasikan sehingga

menyebabkan kehabisan memori karena proses alokasi maupun dealokasi yang tidak diatur dengan baik.

2. Java memiliki array yang tidak memerlukan pointer sehingga memudahkan java programmer.
3. Java menghilangkan multiple inheritance pada C++ dan menggunakan interface yang memiliki kemampuan yang sama tetapi lebih sederhana.

Karakteristik Java antara lain :

A. Sederhana

Java tidak memiliki sintaks yang aneh tapi banyak menggunakan sintaks C++ yang sudah banyak dikenal sehingga java tidak menyulitkan bagi para programmer. Bahkan java memberikan banyak keunggulan dan kemudahan dibandingkan C++.

B. Berorientasi Objek

Java merupakan pemrograman berorientasi objek yang murni. Dalam pemrograman Java semua adalah objek, terkecuali tipe data primitif.

C. Aman

Aman karaena program Java memiliki library security serta policy yang membatasi akses applet di computer client.

D. Portabel

Portabel karena Java dapat dijalankan diberbagai platform tanpa perubahan kode sama sekali.

E. Multithreading

Java memiliki kemampuan untuk menangani dan menjalankan banyak thread sekaligus.

F. Dinamis

Java merupakan teknologi yang terus berkembang dan hal ini tampak nyata sekali dengan library yang terus ditingkatkan kemampuan dan kelengkapannya.

2.12 Java Thread

Thread merupakan kemampuan yang disediakan java untuk membuat aplikasi yang tangguh dimana kita dapat memiliki thread-thread dalam program yang memiliki fungsi dan tugan tersendiri. Dengan adanya thread kita dapat membuat yang lebih efisien dalam hal kecepatan maupun penggunaan sumber daya, karena kita dapat membagi proses dalam aplikasi kita pada waktu yang sama. Jadi thread dapat diumpamakan semacam bagian dari program yang mampu melakukan proses secara tersendiri.

Dengan konsep pemrograman berorientasi objek kita akan lebih mudah dalam berurusan dengan thread. Sebenarnya dalam setiap program yang memiliki method main maka otomatis terdapat thread utama yang menjalankan program. Apabila kita menginginkan thread tambahan untuk membuat program kita berjalan lebih efisien dan cepat maka kita dapat melakukan dengan 2 cara :

1. Membuat Sub Class dari thread
2. Mengimplementasikan Interface Runnable

Dari kedua cara ini kita perlu memperhatikan bahwa tiap class yang ingin anda jadikan thread maka perlu memiliki method run() yang akan dipanggil apabila thread dijalankan. Perbedaan utama antara mengimplementsikan interface

Runnable dan membuat Sub Class Thread yang tampak saat membuat objek Thread supaya lebih jelasnya perhatikan cara konstruksi dari masing-masing cara :

1. Membuat Sub Class dari Thread

```
Class Thread namavar = new Class Thread();
namavar.start();
```

atau kita juga dapat langsung denagan cara :

```
new Class Thread().start();
```

dengan membuat sub class thread maka kita mendapatkan objek thread dengan cara biasa yaitu dengan keyword new dan diikuti nama class yang menjadi sub class thread. Kemudian unutk menjalankan thread, kita memanggil method start();

2. Mengimplementasikan Interface Runnable

Cara ini memiliki perbedaan dengan cara membuat objek thread biasa. Jadi pertama-tama kita membuat objek class kita yang mengimplementasikan interface runnable lalu menjadikannya sebagai argument untuk konstruktor thread seperti kode berikut :

```
ObjekRunnable objek = new ObjekRunnable();
Thread namavar = new Thread(ObjekRunnable);
```

atau cara singkat seperti berikut :

```
new Thread(new ObjekRunnable());
```

(Budi Susanto,2004:36).

2.13 Java Socket

Pada bahasa pemrograman Java, anda membutuhkan Socket untuk membuat suatu hubungan ke mesin atau proses lain, baru kemudian anda akan mendapatkan input stream dan output stream untuk pertukaran datanya. Terdapat dua belah class yang sudah tersedia pada java yang mendukung type koneksi

Connection-Oriented yaitu java.net.ServerSocket yang digunakan server untuk listen koneksi dan java.net.Socket yang digunakan oleh client untuk menginisialisasi koneksi. Setelah client membentuk suatu koneksi Socket dengan proses server, ServerSocket akan mengembalikan status server ke client melalui koneksi yang telah terbentuk sebelumnya.

Java juga menyediakan suatu class yang mendukung tipe connectionless, yaitu java.net.DatagramSocket. Secara umum pada paket java.net berisi class-class dan interface API (Aplication Programming Interface) level rendah (TCP dan UDP) dan level tinggi. Dengan API level rendah anda akan dapat bekerja dengan Socket TCP dan datagram UDP sedangkan API level tinggi memungkinkah anda membangun lebih cepat sebuah program jaringan oleh karena beberapa fungsi utama level rendah dapat diwakili dengan class atau interface level tinggi contohnya adalah URL, URL Connection.

Sebelum kita tinjau constructor dan method yang tersedia dari class Socket, ServerSocket atau DatagramSocket, perlu diketahui dahulu sebuah class yang berfungsi untuk mengambil informasi alamt ip suatu computer. Class tersebut adalah java.net.InetAddress. Class ini bersifat static dan tidak memiliki constructor, namun menyediakan beberapa fungsi yang dapat digunakan untuk mendapatkan alamat IP sesungguhnya antara lain :

A. Method getbyname(nama host) akan menerima sebuah string nama host dan mengembalikan alamat ip sesungguhnya.

Untuk Memanggil method diatas anda dapat langsung memanggilnya tanpa harus membuat instance objek dari class InetAddress

Class Socket adalah sebuah class yang dirancang sebagai wakil sebuah koneksi menggunakan protocol TCP. Ketika sebuah Socket dibuat,sebuah koneksi akan terbentuk ke suatu mesin atau proses yang dituju. Pada java 2.class Socket memiliki kurang lebih 8 buah constructor , berikut ini adalah salah satu diantranya yang digunakan :

B. Socket(InetAddress address,int port) digunakan untuk Membuat sebuah stream Socket dan koneksi ke suatu nomor port pada sebuah computer yang memiliki alamat ip

Selain constructor juga ada dua method dari class Socket yang pasti akan digunakan, yaitu getInputStream() dan getOutputStream(), dimana kedua-duanya mengembalikan suatu objek stream yang dapat digunakan untuk berkomunikasi dengan Socket. Method close() disediakan untuk memberitahukan kepada protocol untuk menutup koneksi (Budi Susanto,2004:6).

2.14 Java Server Socket

Class Server Socket menyatakan suatu koneksi TCP yang berfungsi untuk listen yang siap menerima suatu permintaan dari proses lain. Class ServerSocket ini digunakan ketika anda akan membangun suatu aplikasi server yang menerapkan tipe koneksi Connection Oriented.

Ketika dibuat sebuah objek baru dari class ServerSocket untuk dapat mengirim dan menerima melalui stream, maka perlu dibuat input stream dan output stream dari objek Socket yang dihasilkan ketika dari class ServerSocket menerima permintaan dari client melalui method accept()

Class ServerSocket memiliki suatu constructor yang menerima sebuah nomor port yang akan digunakan sebagai nomor akses layanan yang disediakannya. Serupa dengan class Socket pada class ServerSocket juga menyediakan beberapa method lain.

Ada dua method penting dari class ServerSocket yaitu method accept() yang akan menghasilkan sebah objek class Socket yang terkoneksikan dengan client. Method kedua yang penting adalah close() yang akan menutup sesi listen untuk permintaan yang datang dari Socket.

2.15 Penanganan Kegagalan (Java Exception)

Penanganan terhadap suatu kegagalan dalam berinterkoneksi dengan client maupun proses yang lain dapat dilakukan dengan menangkap kegagalan tersebut dalam blok perintah :

```
try{
        //perintah-perintah yang dikerjakan
}
catch ( ClassException objek ){
        //perintah-perintah yang dikerjakan jika ada kegagalan dari suatu
        //perintah didalam blok try
}
```

Terdapat beberapa class penanganan kegagalan atau disebut dengan class Exception karena semua class penanganan kegagalan ini diturunkan dari class Exception, berikut ini diantaranya :

1. ClassNotFoundException

Exception ini akan dipanggil ketika aplikasi mencoba memanggil sebuah class sesuai dengan nama string yang diberikan namun class tersebut tidak dapat ditemukan

2. InvalidClassException

Exception ini akan dipanggil ketika Serialization runtime menemukan salah satu masalah dengan class tersebut , antara lain :

A. Versi class serial yang tidak sesuai dengan deskripsi class class yang dibaca melalui stream
B. Class tersebut mengandung data types yang tidak dikenali
C. Class tersebut tidak memiliki hak akses terhadap constructor tanpa argument

3. StreamCorruptedException

Exception ini akan dipanggil ketika informasi control yang dibaca melalui object stream melanggar ketentuan internal konsistensi object stream

4. IOException

Exception ini akan dipanggil ketika exception telah terjadi sebagi akibat dari proses signal I/O. class ini adalah class utama exception yang dihasilkan oleh kegagalan yang disebabkan oleh operasi interrupt I/O

5. OptionalDataException

Exception ini akan dipanggil ketika terdapat adanya indikasi kegagalan operasi pembacaan sebuah object yang berisikan data primitive. Exception ini dapat terjadi dalam dua kemungkinan, antara lain :

A. Sebuah usaha membaca sebuah object ketika element selanjutnya didalam stream adalah data primitive. Pada kasus ini, panjang data OptionalDataException di set sesuai dengan jumlah bytes data primitive yang terbaca melalui stream, dan setiap eof (end of file) diset dengan nilai false.

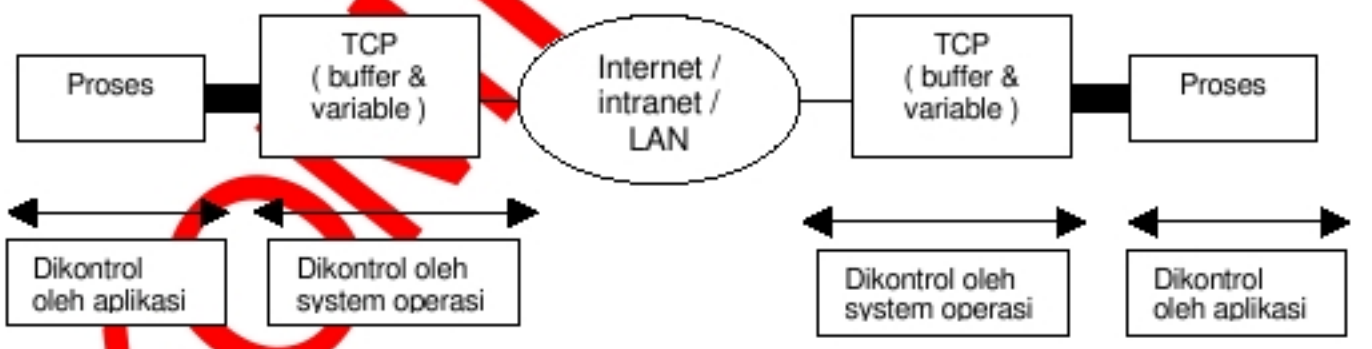
B. Sebuah usaha dilakukan untuk membaca data yang dihasilkan oleh class yang didefinisikan oleh method readObject atau readExternal

6. FileNotFoundException

Exception ini akan dipanggil ketika sebuah usaha untuk membuka file yang diberikan sesuaikan dengan pathname tidak dapat ditemukan (Budi Susanto,2004:13).

2.16 Pemrograman Socket Connection Oriented

Untuk membangun aplikasi jaringan berbasis TCP/IP dengan tipe koneksi Connection Oriented akan digunakan protocol transport TCP. Ketika sebuah proses server ataupun client saling berinterkoneksi (bertukar data), masing-masing akan terhubung atau menggunakan protocol TCP melalui sebuah socket. Sebuah proses ketika berhubungan dengan TCP akan memiliki nomor disamping alamat komputernya.



Gambar 2.12 Komunikasi antar proses dengan menggunakan Socket TCP

Pada gambar 2.12 menunjukkan suatu skema hubungan antara proses aplikasi dengan TCP Lokal dan juga hubungan antara dua computer dengan protocol TCP. Socket berperan menghubungkan antara suatu aplikasi jaringan ketika akan mengirimkan data ke computer atau proses lain menggunakan

protocol TCP harus menggunakan socket agar data yang dikirimkan dapat diterima oleh TCP. Berikutnya adalah tugas TCP yang dikontrol oleh system operasi untuk dapat menghantarkan data yang dikirimkan ke TCP di computer atau proses lainnya.

Setiap sebuah proses yang berhubungan dengan TCP akan memiliki alamat port, sehingga jika ada dua proses yang saling berkomunikasi memanfaatkan TCP. Setiap proses yang dimaksudkan memiliki alamat denagn bentuk AlamatIP:Port (Budi Susanto,2004:17).

2.17 Komunikasi Antar Objek

Penggunaan bahasa pemrograman berbasis objek mengharuskan kita dapat merancang program untuk dapat saling berkomunikasi antar objeknya. Komunikasi ini dapat berupa pemanggilan method dari objek lain ataupun secara langsung mengakses atribut dari objek lain. Agar satu objek dengan objek lain dapat saling berkomunikasi, setiap objek harus dibuat dari suatu initerface.dengan kata lain setiap objek pasti memiliki definisi interface

Pada program yang bekerja diatras suatu jaringan , objek dapat berjalan pada mesin yang berbeda. Dengan kata lain objek yang digunakan oleh suatu proses di satu mesin dapat mengakses objek di proses mesin lain. Pengaksesan ini didasarkan pada interface yang telah disepakati. Pemanggilan ke objek lain hanya dimungkinkan dengan cara memanggil method yang tersedia

Setiap proses berisi sekumpulan objek, beberapa diantara objek tersebut dapat menerima baik pemanggilan secara local maupun remote. Pemanggilan

method yang terjadi antar objek dalam satu proses yang sama disebut local method invocation (Budi Susanto,2004:103).

2.18 Pengiriman Objek Melalui Stream

Pada pemrograman socket biasanya yang dikirimkan hanyalah data stream, " Bagaimana jika yang dikirimkan adalah suatu objek ? ". Diperlukan suatu mekanisme yang dapat memperlakukan sebuah objek yang terkirim sama seperti pengiriman data lainnya. Ini yang dikenal dengan serialisasi objek.

Serialisasi objek (Object Serialization) adalah suatu teknik dimana sebuah program dapat menyimpan status objek ke dalam sebuah sebuah file dan kemudian dibaca kembali dari file ke memory atau dikirimkan melalui jaringan.

Serialisasi objek adalah sebuah mekanisme yang digunakan oleh RMI namun teknik ini juga berguna untuk semua program yang ingin menyimpan status(state) dari suatu objek ke suatu file kemudian dibaca kembali dari file tersebut untuk merekayasa ulang objek yang terbaca dikirimkan melalui jaringan menggunakan socket.

Jika diinginkan sebuah objek dapat diserialisasikan. Objek tersebut harus mengimplementasikan java.io.Serializable. Interface serializable tidak mendefinisikan method yang harus disediakan karena memang tujuannya adalah untuk memberitahukan kepada Java Virtual Machine (JVM), bahwa objek yang menerapkan serializable merupakan objek yang dapat diserialisasikan.

Selain itu menuliskan objek yang terserialisasi melalui jaringan dibutuhkan I/O stream khusus untuk objek yang akan dituliskan melewati

jaringan, yaitu menggunakan class ObjectOutputStream yang merupakan subclass dari FilterOutputStream.

Method WriteObject() digunakan untuk menuliskan sebuah object ke output stream yang telah didefinisikan . Method ini menerima sebuah parameter bertipe class object yang telah menerapkan interface Serializable

Sedangkan untuk membaca sebuah objek dari input stream dapat dilakukan dengan memanggil method readObject() yang akan membaca sebuah object dari input stream object. Oleh karena yang dibaca adalahsebuah objek hasil pengembalian dari method readObject() dapat ditampung ke dalam sebuah objek bertipe kelas object (Budi Susanto, 2004:104).

2.19 Apache Jakarta Tomcat 5 Web Server

Untuk membuat suatu aplikasi berbasis web, diperlukan adanya suatu web server, Apache Jakarta Tomcat merupakan web server yang populer saat ini untuk web yang dibuat dengan berbasis bahasa pemrograman *java*. Server Apache Jakarta Tomcat telah bekerja pada semua platform seperti UNIX. Berbagai macam varian Linux, OS/2, Windows dan lain-lain. Ada banyak versi apache jakarta tomcat saat ini, setiap versi berisi fitur-fitur seperti :

1. *Host* Virtual. Memungkinkan sebuah komputer untuk berhubungan dengan sejumlah besar web server pada saat yang bersamaan sehingga satu buah komputer yang menjalankan satu web server dapat melayani banyak halaman dari beragam situs web.
2. Server-Side Include. Perintah – perintah yang ada dalam halaman web HTML yang menyediakan fungsi server-side. SSI serupa dengan CGI yang secara

khusus digunakan untuk membuat sebuah web yang dinamis. Situs web sering mengaktifkan fitur ini untuk mengerjakan file berekstensi .shtml dan mengendalikan SSH.

3. Halaman web dinamis dari CGI. Suatu mekanisme orisinil yang dikembangkan untuk mengirimkan isi data yang dinamis ke web.
4. Handler, pengendali untuk mengendalikan beragam request pada web berdasarkan nama file. File – file tertentu seperti .asp dan seterusnya. Handler bersifat built-in dan mengendalikan isi data statis seperti HTML.
5. Variabel lingkungan.
6. Pemetaan URL ke sistem file.

2.20 Java Applet

Secara sederhana, applet adalah program java yang decompile dan dijalankan menggunakan Applet Viewer atau melalui Web Browser yang mensupport java. Apa yang dilakukan sebuah applet tergantung dari desain program. Applet dapat menampilkan gambar, image, memainkan audio, menerima input user dan memanipulasi program seperti program lainnya.

Secara teknis applet adalah subclass dari java.awt.Panel. Dengan demikian sebuah applet dapat berisi fitur-fitur dari Abstract Window Toolkit dan Java Language Package. Untuk membantu membuat ataupun mendesain, Sun telah mengembangkan Java Applet Package, sekumpulan koleksi constructor dan method yang didesain memiliki sejumlah fitur yang berguna yang dapat diakses oleh programmer (Michael Morrison,2002).

A. Applet dan World Wide Web

Kemampuan dasar applet telah membuatnya sempurna dan sesuai untuk penggunaan di internet maupun pada World Wide Web. Karena applet kecil, sehingga dapat didownload dan dijalankan dengan cepat.

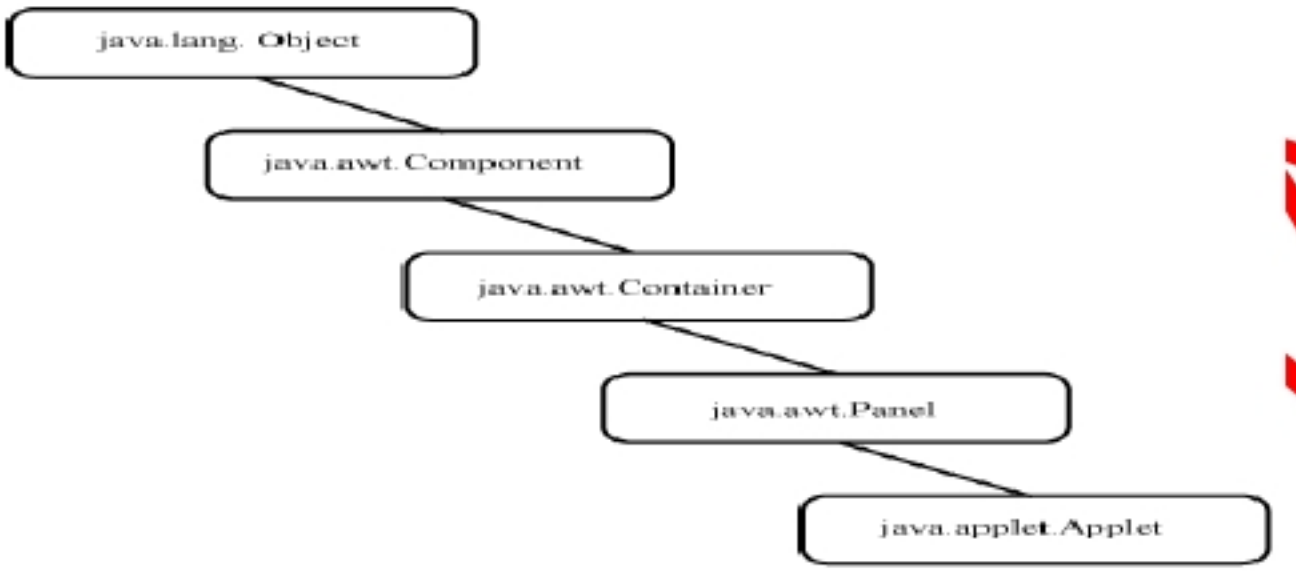
Applet dapat menambahkan sebuah tingkatan baru fungsionalitas, interaktifitas ke halaman web tanpa adanya beban layaknya aplikasi berskala penuh. Dengan ada banyaknya browser yang mendukung applet seperti Opera, Netscape, applet telah berpengaruh pada World Wide Web (Michael Morrison,2002).

Banyak halaman web menggunakan java applet untuk berbagai macam kegunaan, diantaranya sebagai berikut :

1. Animasi
2. Menampilkan image dengan suara
3. Efek graphic seperti teks bergulung
4. Interactive programming, seperti games

B. Dasar-dasar Applet

Oleh karena hirarki dari Java class, sebuah applet adalah perluasan dari objek Panel pada Abstract Windows Toolkit (AWT). Sebuah applet adalah subclass dari Container class (lihat gambar 2.13), dan juga mengandung segala komponen dari bahasa Java seperti button, menu, scroll bar dan yang lainnya.



Gambar 2.13 Struktur class applet

Karena applet sebenarnya adalah sebuah class, applet dapat digunakan untuk membangun applet yang lainnya atau aplikasi skala penuh. Applet selain dapat digunakan bersama-sama applet lainnya, applet juga mewarisi fitur dari class diatasnya. Applet juga memiliki tingkatan yang berbeda pada waktu hidupnya. Tingkatan ini adalah init, start, stop, dan destroy. Seluruh method tersebut dapat ditulis ulang oleh programmer untuk memenuhi tugas spesifik pada setiap method yang dimiliki applet (Michael Morrison,2002).

C. Java class hirarki.

Berikut adalah urutan empat tingkatan method pada applet :

1. init();

Method init berisi inisialisasi seluruh method dan variable yang dibutuhkan applet agar dapat berfungsi, seperti gambar yang harus dipanggil. Method init dipanggil ketika applet pertama kali dijalankan.

2. **start();**

   Method start menjalankan fungsi utama dari sebuah applet. Sebagai contoh : jika sebuah applet memainkan musik, maka dapat mulai dikerjakan pada saat method start dipanggil. Method start dipanggil ketika method init telah selesai dijalankan dan applet siap untuk dieksekusi.

3. **stop();**

   Method stop dapat digunakan untuk menghentikan segala aksi yang mungkin sedang berlangsung sejak method start dipanggil ketika applet selesai atau keluar. Sebagai contoh : suara musik yang berulang-ulang perlu dihentikan ketika pengguna meninggalkan halaman web yang berisi applet. Method stop dipanggil ketika halaman web ditinggalkan atau layer browser diperkecil (minimized).

4. **destroy();**

   Method destroy otomatis dipanggil dan mengerjakan seluruh memory cleanup atau garbage collection yang perlu dikerjakan ketika sebuah applet telah selesai. Method ini dapat ditulis ulang tapi hal tersebut secara umum tidak diperlukan. Method destroy secara otomatis dipanggil ketika pengguna menutup browser.

D. Pewarisan dari Class Applet.

Applet mewarisi properti dari objek Panel pada AWT, dan attribut class Applet dari java.applet.Applet. Hal ini menjelaskan struktur dari java applet, cara mendefinisikan applet sebagai berikut :

1. public class Classname extends java.applet.Applet { }

Statement code diatas menunjukkan bahwa class tersebut merupakan perluasan dari class Applet. Terdapat sejumlah method yang dibangun untuk class Applet sehingga membuat applet mudah untuk dikembangkan. Applet mewarisi kemampuan untuk memiliki objek user interface karena seluruh applet adalah Panel.

2. Applet HTML

Saat ini Applet dapat berjalan pada beberapa UNIX platform dan Microsoft Windows. Setelah kode java dicompile langsung dapat dieksekusi menggunakan Applet Viewer, atau melalui browser yang mendukung java. Applet membutuhkan 2 (dua) komponen berikut agar dapat dieksekusi :

A. Bytecode hasil dari proses compile

B. File HTML yang berisi informasi applet dan parameter

Bytecode hasil dari proses compile adalah komponen applet yang dapat dieksekusi. File HTML dibutuhkan Applet Viewer dan Web browser agar dapat di eksekusi kodenya dengan sempurna. File HTML didasarkan pada tag < applet> dan memiliki struktur dasar sebagai berikut :

```
<html>
     <applet codebase=lokasi dari kode code=namafile.class width=90 height=90 >
     </applet>
</html>
```

Tag <applet> tag berisi nama file dari code yang akan dieksekusi, dengan format namafile.class, dan diikuti dengan besar ukuran applet. Nilai awal ukuran applet diberikan dalam satuan pixel. Hal ini disebabkan ukuran applet harus sesuai dengan ketentuan layout halaman HTML.

Tag <param> menerima nama parameter name beserta nilai yang terkandung didalamnya. Parameter tag hanya digunakan jika applet didesain untuk

mengambil suatu nilai parameter, dan dapat dilakukan sebanyak mungkin selama diperlukan to menghasilkan nilai parameter. Berikut adalah daftar tag dan nilai yang dapat digunakan pada sebuah file applet HTML :

codebase

Codebase adalah lokasi utama URL dari Java bytecode. Hal ini memungkinkan programmer untuk meletakkan bytecode pada directori yang berbeda dengan lokasi dari file applet yang akan ditampilkan / dijalankan.

align

Mirip dengan standard HTML tag <align>. Tag ini memungkinkan anda untuk menentukan alignment seperti right, left, or center dari applet.

code

Parameter code dibutuhkan oleh tag <applet>. Code menunjukkan lokasi yang spesifik dan actual dari appletyang telah dicompile.

width

Parameter width parameter digunakan oleh tag <applet> untuk menentukan lebar dari window yang akan dibuka ketika applet ditambahkan kedalam halaman web. Paramater ini dibutuhkan oleh tag <applet>.

height

Parameter height serupa dengan parameter width, menetukan tinggi dari applet window. Paramater ini dibutuhkan oleh tag <applet>.

2.21 Java Server Page (JSP)

Java Server Page adalah teknologi yang memungkinkan programmer untuk menggabungkan HTML regular, statis dengan content yang dihasilkan

secara dinamik oleh servlet. Cukup membuat file HTML menggunakan perangkat web design. Anda dapat menambahkan code untuk bagian dinais dalam sebuah tag special, kebanyakan diawalai dengan <% dan diakhiri %>.

Sebagai contoh : berikut adalah bagian sebuah halaman JSP yang menghasilkan tulisan "Thanks for using JSP" untuk alamat URL http://host/Information.jsp?title=JSP.

```
Thank for using <I><%= request.getParameter("title") %></I>
```

A. Cookie di Java Server Page

Cookie merupakan solusi mekanisme umum yang digunakan pada aplikasi web untuk menyimpan informasi –informasi penting yang disimpan di computer pengguna. Browser mengirimkan cookie ke server setiap kali browser memanggil halaman web. Bentuk informasi yang disimpana dalam cookie berupa nama=value. Cara membuat cookie sebagai berikut :

```
Cookie kue = new Cookie("UserID","Novan");
```

Untuk mendapatkan informasi cookie sbagai berikut :

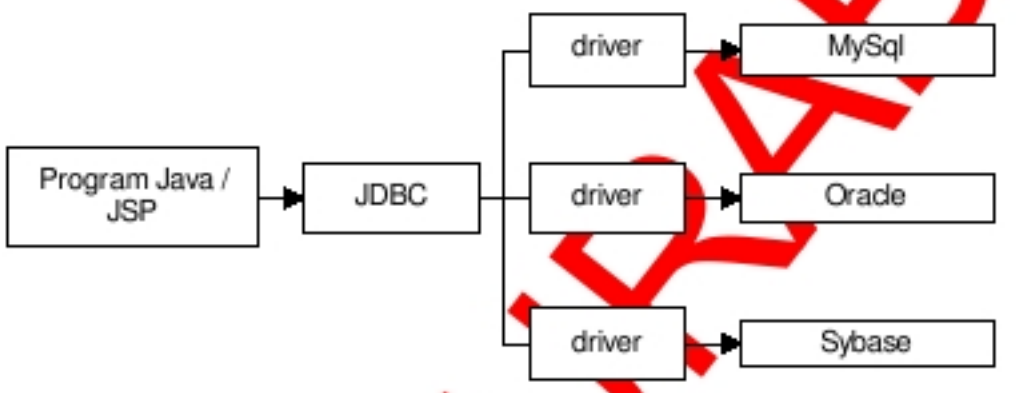
```
Cookie[] kue = request.getCookies();
```

2.22 Java Cryptography Extension (JCE)

Java Cryptography Extension terdiri dari sejumlah class dan interface yang terdapat di java.crypto paket maupun sub paket. Class-class tersebut menyediakan fungsi generik API yang digunakan untuk enkripsi dan dekripsi. Agar dapat menggunakan JCE maka harus diinstal pada java virtual mesin. Ada berbagai macam JCE yang tersedia yang dibuat oleh provider yang berbeda. Berikut beberapa diantaranya yang digunakan untuk pembuatan aplikasi adalah Bouncy Castle dan SunJCE.

2.23 Java Database Connectivity (JDBC)

JDBC adalah Application Programming Interface (API) yang dikembangkan Sun Microsystem unutk menyediakan akses data universal dalam bahasa Java. JDBC merupakan koleksi API yang terdiri dari sekumpulan class dan interface yang ditulis dalam bahasa Java sebagai standar sehingga memungkinkan pembuatan aplikasi database yang portable dengan bahasa java.



Gambar 2.14 Diagram JDBC

JDBC memudahkan saat mengirimkan perintah SQL ke system database dan mendukung dialek SQL dari berbagai macam vendor database. JDBC bukan database server, tetapi merupakan penyedia mekanisme unutk berhubungan antara aplikasi Java dengan database server. Dalam JDBC terdapat tujuh langkah untuk menangani database, yaitu : (Isack Riyanto, 2004:162)

1. Memanggil driver JDBC

Driver adalah library yang digunakan untuk berkomunikasi dengan database server dan membuat program Java yang menggunakan API JDBC dapat berinteraksi dan dimengerti oleh database server. Diperlukan driver yang berbeda untuk database server sehingga perlu mendownload atau mendapatkan driver yang sesuai. Misalkan memanggil driver unutk mysql :

```
try{
```

```
  Class.forName("com.mysql.jdbc.Driver");
  }
catch (Exception e) {
  System.out.println("Error Loading Driver : " + e);
  }
```

2. Mendefinisikan URL untuk Koneksi Database

Setelah meload driver, perlu didefinisikan lokasi secara spesifik dari database server. URLyang mengacu ke database menggunakan protocol jdbc: dan diikuti server host, port dan nama databse. Sintaks JDBC URL bisa disebut sebagai Connection String. Alamat URL disimpan dalam variable bertipe String untuk database "securechat" dengan database mysql dengan nama host "della" sebagai berikut :

```
String urldb = "jdbc:mysql://della/securechat";
```

3. Melakukan Koneksi Database

Informasi URL, username dan password harus dikirimkan agar dapat melakukan koneksi database dengan method getConnection dari class DriverManager.

```
String user = "root";
String password = "atmodiharjo";
Connection koneksi = DriverManager.getConnection(urldb,user,password);
```

4. Membuat objek Statement

Objek statement digunakan untuk melakukan query dan objek ini dapat dibuat dari objek Connection. Kode untuk membuat objek statement adalah :

```
Statement stm = koneksi.createStatement();
```

5. Melakukan Query atau Update

Objek statement digunakan untuk mengirimkan query dan mengeksekusinya dengan metoda executeQuery yang mengembalikan objek bertipe ResultSet. Berikut listing kode :

```
String strQuery = "SELECT * FROM tbluser '";
ResultSet resultSet = stm.executeQuery(strQuery);
```

6. Memproses Hasil

Untuk memproses hasil dapat menggunakan objek ResultSet karena hasil dari query disimpan dalam objek ini. Metode utama yang sering digunakan adalah metode next dan getString() seperti contoh dibawah ini :

```
while(resultSet.Next())
    {
    System.out.println(resultSet.getString(1) + " " + resultSet.getString(2));
    }
```

7. Menutup Koneksi

Sebelum menutup koneksi database, perlu merelease atau melepaskan objek ResultSet yang ada dengan kode berikut :

stm.close();

Tulis kode berikut untuk menutup koneksi database

koneksi.close();