BAB II

TINJAUAN PUSTAKA

2. 1. Sejarah Perusahaan

Perusahaan yang saat ini bernama PT. Citra Bumi Madani pada dasarnya mulai melakukan kegiatan pada tahun 1993. Pada awal kegiatannya perusahaan ini belum mempunyai badan hukum, melainkan lebih merupakan suatu kegiatan usaha di bidang survey pemetaan yang bersifat pribadi / partikelir (individual / personal). Selain menyediakan jasa di bidang survey pengukuran, perusahaan ini juga menyediakan jasa trainer untuk melatih personil dibidang survey pemetaan, terutama dalam melatih pengoperasian alat-alat pengukuran type terbaru maupun metoda-metoda pengukuran baru. Seiring dengan hal tersebut, maka asset dalam bentuk peralatan dan juga jumlah personil juga semakin bertambah diperlukan penataan organisasi dan manajemen yang lebih baik. Dari tahun ketahun kepercayaan pengguna jasa dibidang pengukuran terhadap perusahaan ini semakin meningkat, khususnya permintaan jasa pengukuran dari perusahaan-perusahaan yang bergerak dibidang tambang seperti tambang batubara, tambang galena dan tambang nikel yang tersebar di seluruh indonesia.

2. 2. Keamanan Informasi

Keamanan informasi (*information security*) merupakan perlindungan terhadap informasi ketika informasi dikirim dari sebuah sistem ke sistem lainnya. Sistem kemanan informasi memiliki empat tujuan yang sangat mendasar, yaitu:

1. *Confidentiality*

   Menjamin apakah informasi yang dikirim tersebut tidak dapat dibuka atau tidak dapat diketahui oleh orang lain yang tidak berhak. Untuk data yang penting, dibutuhkan sekali tingkat kerahasiaan yang tinggi, yang hanya bisa diakses oleh pihak-pihak tertentu saja (pihak yang berhak).

2. *Integrity*

   Menjamin keutuhan dan keaslian data, sehingga upaya pihak-pihak yang tidak bertanggung jawab untuk melakukan penduplikatan dan perusakan data dapat dihindari.

3. *Availability*

   Menjamin pengguna yang sah agar dapat mengakses informasi dan sumber miliknya sendiri. Tujuannya adalah untuk memastikan bahwa pihak-pihak yang memang berhak tidak ditolak untuk mengakses informasi yang memang menjadi haknya.

4. *Legitimate Use*

Menjamin kepastian bahwa sumber tidak digunakan atau informasi tidak diakses oleh pihak-pihak yang tidak bertanggung jawab (pihak-pihak yang tidak berhak).

2. 2. 1. Pengenalan Kriptografi

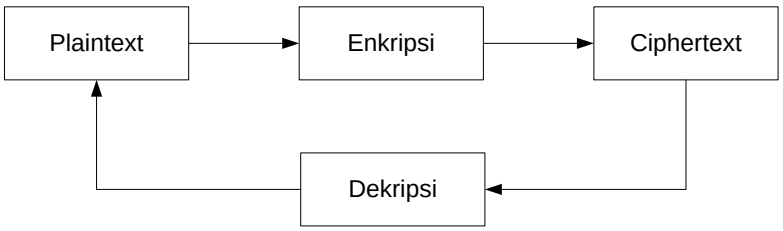
2. 2. 1. 1. Definisi dan Sejarah

Kriptografi adalah ilmu untuk mempelajari penulisan secara rahasia dengan tujuan bahwa komunikasi dan data dapat dikodekan (*encode* / *encrypt*) dan didekodekan (*decode* / *decrypt*) kembali untuk mencegah pihak-pihak lain yang ingin mengetahui isinya, dengan menggunakan kode-kode dan aturan-aturan tertentu dan metode lainnya sehingga hanya pihak yang berhak saja yang dapat mengetahui isi pesan sebenarnya.

Sebelum tahun 1970-an, teknologi kriptografi digunakan terbatas hanya untuk tujuan militer dan diplomatik. Akan tetapi kemudian bidang bisnis dan perorangan mulai menyadari pentingnya melindungi informasi berharga.

Kriptografi berasal dari bahasa Yunani yaitu kripto yang artinya tersembunyi dan grafia yang artinya sesuatu yang tertulis. Jadi kriptografi merupakan proses transformasi suatu data yang memiliki makna ke dalam bentuk data acak yang tidak memiliki makna, gunanya untuk menjaga kerahasiaan data dengan menerapkan rumus matematika tertentu, kemudian dengan transformasi balik, data acak ini dapat diubah kembali ke dalam bentuk data semula.

Sumber data yang akan ditransformasikan / dikodekan disebut *plaintext*. Proses untuk menyembunyikan isi sesungguhnya dari sebuah *plaintext* ke dalam bentuk yang hampir tidak dapat dibaca tanpa pengetahuan yang cukup dan sesuai disebut enkripsi. Sebuah sumber data yang telah ditransformasikan disebut *ciphertext*. Proses untuk mengembalikan *ciphertext* ke *plaintext* disebut dengan dekripsi. Proses enkripsi dan dekripsi dapat dilihat pada gambar 2.1 berikut ini.



Gambar 2.1 Proses enkripsi dan dekripsi

Kriptografi mempunyai sebuah sejarah yang panjang dan menarik. Penghargaan untuk referensi non teknis terlengkap adalah *The Codebreakers* oleh *Kahn*. Buku ini meliputi kriptografi dari penggunaannya yang pertama dan terbatas oleh bangsa Mesir 4000 tahun yang lalu, hingga abad XX dimana kriptografi memegang peranan penting sebagai akibat dari kedua perang dunia. Diselesaikan pada tahun 1963, buku Kahn mencakup aspek-aspek sejarah yang paling penting (pada waktu itu) pada perkembangan kriptografi. Para pengguna utama adalah mereka yang berhubungan dengan militer, pelayanan diplomatik dan pemerintahan secara umum. Kriptografi digunakan sebagai suatu alat untuk melindungi rahasia dan strategi-strategi nasional.

Perkembangan komputer dan sistem komunikasi pada tahun 1960-an mengakibatkan kebutuhan pihak swasta akan alat untuk melindungi informasi

dalam bentuk digital dan untuk menyediakan layanan keamanan. Dimulai dengan usaha *Feistel* di IBM pada awal tahun 1970-an dan mencapai puncaknya pada tahun 1977 dengan pengadopsian sebagai Standar Pemrosesan Informasi pemerintah Amerika Serikat bagi enkripsi informasi rahasia, *DES, Data Encryption Standard*, adalah mekanisme kriptografi yang paling terkenal dalam sejarah. Metode ini menjadi alat standar untuk mengamankan *electronic commerce* bagi banyak institusi financial di seluruh dunia. Sejarah perkembangan kriptografi dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1. Sejarah Perkembangan Kriptografi

Tahun	Keterangan
~ 1900 A.D.	Pertama kali digunakannya teknik transformasi *cyptography* di "*tomb inscription*", merupakan penggunaan kriptografi yang pertama kali diketahui.
~ 475 A.D.	Sparta menggunakan kriptografi untuk komunikasi dan juga merancang alat untuk mengenkripsi (skytale) yang menghasilkan *transposition cipher*.
~ 350 A.D.	"Aenas The Tactician" mengeluarkan tulisan pertama mengenai keamanan komunikasi dan kriptografi.
~ 60 A.D.	Julius Caesar menjadi orang yang pertama kali yang diketahui menggunakan substitution *cipher*.

1412	"Treatise" tertua yang diketahui dalam kriptanalis yang diter*bit*kan oleh Alkalkas Handi (Mesir).
1917	Edward Hugh Hibern mengembangkan mesin motor yang pertama.
1971	IBM mengembangkan teknik enkripsi Lucifer.
1975	DES diumumkan (disetujui tahun 1977).
1976	Presentasi terbuka pertama tentang konsep public key oleh Diffie dan Helman.
1977	Merkle mengembangkan algoritma knapsack dan memberikan hadiah $100 bagi yang dapat memecahkan kuncinya (algoritma dengan satu kali pengulangan). Algoritma Rivest – Shamir – Aldeman (RSA) diumumkan kepada umum.

2. 2. 2. Aplikasi Kriptografi

2. 2. 2. 1. Confidentiality dan Privacy

Confidentiality dan *privacy* terkait dengan kerahasiaan data atau informasi. Pada sistem *e-government* kerahasiaan data-data pribadi (*privacy*) sangat penting. Hal ini kurang mendapat perhatian di sistem *e-government* yang sudah ada. Bayangkan jika data pribadi anda, misalnya data KTP atau kartu keluarga, dapat diakses secara *online*. Maka setiap orang dapat melihat tempat dan tanggal lahir anda, alamat anda, dan data lainnya. Data ini dapat digunakan untuk melakukan

penipuan dan pembobolan dengan mengaku-aku sebagai anda (atau keluarga anda).

Ancaman atau serangan terhadap kerahasiaan data ini dapat dilakukan dengan menggunakan penerobosan akses, penyadapan data (*sniffer, key logger*), *social engineering* (yaitu dengan menipu), dan melalui kebijakan yang tidak jelas (tidak ada).

Untuk itu kerahasiaan data ini perlu mendapat perhatian yang besar dalam implementasi sistem *e-government* di Indonesia. Proteksi terhadap data ini dapat dilakukan dengan menggunakan *firewall* (untuk membatasi akses), segmentasi jaringan (juga untuk membatasi akses), enkripsi (untuk menyandikan data sehingga tidak mudah disadap), serta kebijakan yang jelas mengenai kerahasiaan data tersebut.

Pengujian terhadap kerahasian data ini biasanya dilakukan secara berkala dengan berbagai metode.

Ada beberapa jenis informasi yang tersedia didalam sebuah jaringan komputer. Setiap data yang berbeda pasti mempunyai grup pengguna yang berbeda pula dan data dapat dikelompokkan sehingga beberapa pembatasan kepada pengunaan data harus ditentukan. Pada umumnya data yang terdapat didalam suatu perusahaan bersifat rahasia dan tidak boleh diketahui oleh pihak ketiga yang bertujuan untuk menjaga rahasia perusahaan dan strategi perusahaan. *Backdoor*, sebagai contoh, melanggar kebijakan perusahaan dikarenakan menyediakan akses yang tidak diinginkan kedalam jaringan komputer perusahaan.

Kerahasiaan dapat ditingkatkan dan didalam beberapa kasus pengenkripsian data. Kontrol akses adalah cara yang lazim digunakan untuk membatasi akses kedalam sebuah jaringan komputer. Sebuah cara yang mudah tetapi mampu untuk membatasi akses adalah dengan menggunakan kombinasi dari *username*-dan-*password* untuk proses otentikasi pengguna dan memberikan akses kepada pengguna (*user*) yang telah dikenali. Didalam beberapa lingkungan kerja keamanan jaringan komputer, ini dibahas dan dipisahkan dalam konteks otentikasi.

Dalam menjaga kerahasiaan data, kriptografi mentransformasikan data jelas (*plaintext*) ke dalam bentuk data sandi (*ciphertext*) yang tidak dapat dikenali. *Ciphertext* inilah yang kemudian dikirimkan oleh pengirim (*sender*) kepada penerima (*receiver*). Setelah sampai di penerima, *ciphertext* tersebut ditranformasikan kembali ke dalam bentuk *plaintext* agar dapat dikenali.

2. 2. 2. 2. Otentikasi (Authentication)

Otentikasi merupakan identifikasi yang dilakukan oleh masing – masing pihak yang saling berkomunikasi, maksudnya beberapa pihak yang berkomunikasi harus mengidentifikasi satu sama lainnya. Informasi yang didapat oleh suatu pihak dari pihak lain harus diidentifikasi untuk memastikan keaslian dari informasi yang diterima. Identifikasi terhadap suatu informasi dapat berupa tanggal pembuatan informasi, isi informasi, waktu kirim dan hal-hal lainnya yang berhubungan dengan informasi tersebut. Aspek ini berhubungan dengan metode untuk menyatakan bahwa informasi betul-betul asli, orang yang mengakses atau

memberikan informasi adalah betul-betul orang yang dimaksud, atau *server* yang kita hubungi adalah betul-betul *server* yang asli.

Masalah pertama, membuktikan keaslian dokumen, dapat dilakukan dengan teknologi *watermarking* dan *digital signature*. *Watermarking* juga dapat digunakan untuk menjaga "*intelectual property*", yaitu dengan menandai dokumen atau hasil karya dengan "tanda tangan" pembuat. Masalah kedua biasanya berhubungan dengan akses kontrol, yaitu berkaitan dengan pembatasan orang yang dapat mengakses informasi.

2. 2. 2. 3. Integritas (Integrity)

Aspek integritas (*integrity*) terkait dengan keutuhan data. Aspek ini menjamin bahwa data tidak boleh diubah tanpa ijin dari pihak yang memiliki hak. Acaman terhadap aspek integritas dilakukan dengan melalui penerobosan akses, pemalsuan (*spoofing*), virus yang mengubah atau menghapus data, dan *man in the middle attack* (yaitu penyerangan dengan memasukkan diri di tengah-tengah pengiriman data). Proteksi terhadap serangan ini dapat dilakukan dengan menggunakan *digital signature, digital certificate, message authentication code, hash function*, dan *checksum*. Pada prinsipnya mekanisme proteksi tersebut membuat kode sehingga perubahan satu bit pun akan mengubah kode.

2. 2. 2. 4. Nonrepudiation

Aspek ini menjaga agar seseorang tidak dapat menyangkal telah melakukan sebuah transaksi. Sebagai contoh, seseorang yang mengirimkan email untuk memesan barang tidak dapat menyangkal bahwa dia telah mengirimkan

email tersebut. Aspek ini sangat penting dalam hal *electronic commerce*. Penggunaan *digital signature, certificates,* dan teknologi kriptografi secara umum dapat menjaga aspek ini. Akan tetapi hal ini masih harus didukung oleh hukum sehingga status dari *digital signature* itu jelas legal.

Definisi dari *nonrepudition* adalah sebagai berikut :

1. Kemampuan untuk mencegah seorang pengirim untuk menyangkal kemudian bahwa dia telah mengirim pesan atau melakukan sebuah tindakan.
2. Proteksi dari penyangkalan oleh satu satu dari entitas yang terlibat didalam sebuah komunikasi yang turut serta secara keseluruhan atau sebagian dari komunikasi yang terjadi.

Proses tranformasi dari *plaintext* menjadi *ciphertext* disebut proses *encipherment* atau enkripsi (*encryption*), sedangkan proses mentransformasikan kembali *ciphertext* menjadi *plaintext* disebut proses dekripsi (*decryption*).

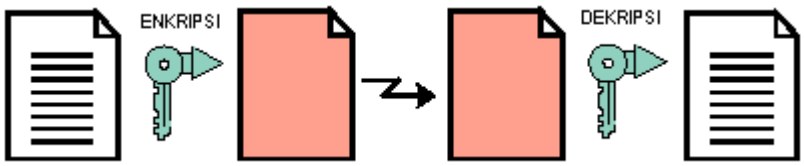
Untuk mengenkripsi dan mendekripsi data, kriptografi menggunakan suatu algoritma (*cipher*) dan kunci (key). *Cipher* adalah fungsi matematika yang digunakan untuk mengenkripsi dan mendekripsi. Sedangkan kunci merupakan sederetan bit yang diperlukan untuk mengenkripsi dan mendekripsi data.

2. 2. 3. Jenis-jenis Kriptografi

2. 2. 3. 1. Kunci Simetris

Ini adalah jenis kriptografi yang paling umum dipergunakan. Kunci untuk mengenkripsi pesan sama dengan untuk mendekripsi pesan yang telah terenkripsi. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci yang memiliki tersebut termasuk pihak-pihak yang

tidak diinginkan dapat membuat dan membongkar rahasia *ciphertext*. Masalah yang paling jelas disini terkadang bukanlah masalah pengiriman *ciphertext*-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan. Contoh algoritma kunci simetris yang terkenal adalah DES (*Data Encryption Standard*) dan RC-4 ada pada gambar 2.2 berikut.



Gambar 2.2 Penggunaan Kunci simetris

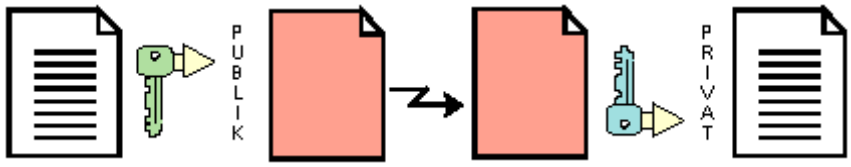
Kelebihan kriptografi kunci simetri (kunci rahasia) ini adalah memiliki kecepatan proses enkripsi dan dekripsi yang jauh lebih cepat dibandingkan dengan sistem kriptografi lainnya. Kriptografi kunci simetri juga memiliki kelemahan. Adapun kelemahan dari kriptografi kunci simetri ialah :

1. Masalah distribusi kunci. Bagaimana caranya agar kunci pembuka (untuk mendekripsi) tersebut bisa dengan aman sampai ke penerima. Padahal diketahui bahwa jaringan *internet* benar-benar tidak aman. Jika kunci sampai hilang atau dapat ditebak oleh pihak ketiga yang tidak bertanggungjawab, maka kriptosistem ini sudah tidak aman lagi.
2. Masalah efisiensi jumlah kunci. Jika ada pengguna sebanyak n, maka akan dibutuhkan n(n-1)/2 kunci, sehingga untuk jumlah pengguna yang sangat banyak, sistem ini tidak efisien.

2. 2. 3. 2. Kunci Asimetris

Pada pertengahan tahun 70-an Whitfield Diffie dan Martin Hellman menemukan teknik enkripsi asimetris yang merevolusi dunia kriptografi. Kunci asimetris adalah pasangan kunci-kunci kriptografi yang salah satunya dipergunakan untuk proses enkripsi dan yang satu lagi untuk dekripsi. Semua orang yang mendapat kunci publik dapat menggunakan untuk mengenkripsi suatu pesan, sedangkan hanya satu orang saja yang memiliki rahasia tertentu dalam hal ini kunci privat untuk melakukan pembongkaran terhadap sandi yang dikirim untuknya.

Dengan cara seperti ini, jika Anto mengirim pesan untuk Badu, anto dapat merasa yakin bahwa pesan tersebut hanya dapat dibaca oleh Badu, karena hanya Badu yang bisa melakukan deskripsi dengan kunvi privatnya. Tentunya Anto harus memilki kunci publik Badu untuk melakukan enkripsi. Anto bisa mendapatkannya dari Badu, ataupun dari pihak ketiga seperti Tari.



Gambar 2.3 Penggunaan Kunci Asimetris

Teknik enkripsi asimetris ini jauh lebih lambat ketimbang enkripsi dengan kunci simetris. Oleh karena itu, biasanya bukanlah pesan itu sendiri yang disandikan dengan kunci asimetris, namun hanya kunci simetrislah yang disandikan dengan kunci asimetris. Sedangkan pesannya dikirm setelah

disandikan dengan kunci simetris tadi. Contoh algortima terkenal yang menggunakan kunci asimetris adalah RSA (merupakan singkatan penemunya yakni Rivest Shamir dan Adleman).

2. 2. 4. Digital Signature

Bedasarkan sejarahnya, penggunaan digital signature berawal dari penggunaan teknik kriptografi yang digunakan untuk mengamankan informasi yang hendak ditransmisikan/disampaikan kepada orang yang lain yang sudah digunakan sejak ratusan tahun yang lalu. Dalam suatu kriptografi suatu pesan dienkripsi (*encrypt*) dengan menggunakan suatu kunci (*key*). Hasil dari enkripsi ini adalah berupa *chipertext* tersebut kemudian ditransmisikan/diserahkan kepada tujuan yang dikehendakinya. Chipertext tersebut kemudian dibuka/didekripsi (*decrypt*) dengan suatu kunci untuk mendapatkan informasi yang telah enkripsi tersebut. Terdapat dua macam cara dalam melakukan enkripsi yaitu dengan menggunakan kriptografi simetris (*symetric crypthography/secret key crypthography*) dan kriptografi simetris (*asymetric crypthography*) yang kemudian lebih dikenal sebagai *public key crypthography.*

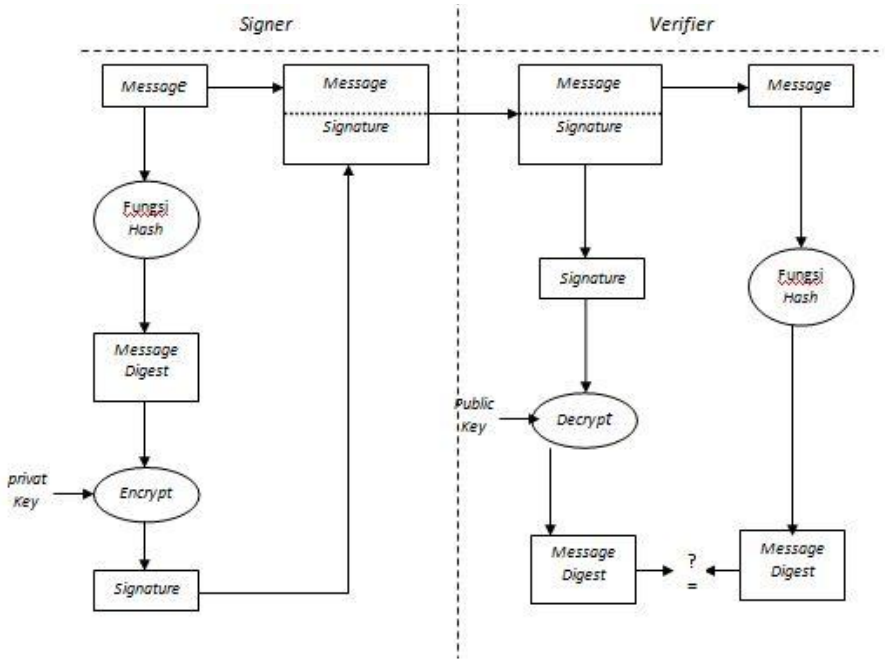
Secret key crypthografi atau yang dikenal sebagai kriptografi simetris, menggunakan kunci yang sama dalam melakukan enkripsi dan dekripsi terhadap suatu pesan (*message)*, disini pengirim dan penerima menggunakan kunci yang sama sehingga mereka harus menjaga kerahasian (*secret*) terhadap kuci tersebut. Salah satu algoritma yang terkenal dalam kriptografi simetris ini adalah *Data Encryption standard* (DES).

Tanda tangan digital (*digital signature*) adalah mekanisme otentikasi yang memungkinkan pembuat pesan menambahkan sebuah kode yang bertindak sebagai tanda tangannya. Tanda tangan dihasilkan berdasarkan pesan yang diinginkan pesan yang ingin ditandatangani dan berubah-ubah sesuai dengan pesan. Tanda tangan digital dikirmkan bersama-sama dengan pesan kepada penerima.Tanda tangan digital memungkinkan penerima informasi untuk menguji terlebih dahulu keaslian informasi yang didapat dan juga untuk meyakinkan bahwa data yang diterima itu dalam keadaaan utuh. Oleh karena itu, tanda tangan digital kunci publik (*public key gital signature*) menyediakan layanan *authentication* (keaslian) dan *data integrity* (keutuhan data). Selain itu tanda tangan digital juga menyediakan layanan *non-repudiation* yang artinya melindungi pengirim dari sebuah klaim yamg menyatakan bahwa dia telah mengirim informasi padahal tidak.Tanda tangan digital memberikan pelayanan yang tujuannya sama dengan tanda tangan (berupa tulisan tangan biasa). Tetapi bagaimanapun juga tanda tangan berupa tulisan tangan relatif lebih mudah ditiru/dipalsukan oleh orang lain. Sedangkan tanda tangan digital hampir tudak mungkin dipalsukan, bahkan bisa berfungsi ganda yaitu sekaligus memperlihatkan sekilas isi informasi mengenai identitas yang menandatangani.

Bahkan beberapa orang lebih cenderung menggunakan tanda tangan digital dari pada menggunakan enkripsi biasa. Contohnya, seseorang mungkin tidak perduli jika setiap orang mengetahui bahwa dia telah mentransfer uang sejumlah satu milyar ke rekening lain, tetapi orang tersebut pasti ingin tahu dan lebih peduli apakah *teller* bank tersebut benar-benar asli, benar-benar ada dan

benar-benar sah. Perasaan tidak tenang akan *teller* gadungan dalam sistem transaksi lewat *internet* sangat wajar terjadi.

Pada konsep yang telah diketahui sebelumnya, proses enkripsi informasi biasanya menggunakan kunci publik. Tetapi pada konsep tanda tangan digital, informasi justru dibubuhi tanda tangan digital (dienkripsi) dengan kunci rahasia yang dimilki sumber. Apabila informasi tadi bisa diverifikasi (didekripsi) dengan kunci publik sumber yang telah tersebar ini berarti bahwa informasi tersebut adalah benar-benar asli dari sumber. Konsep tanda tangan digital dapat dilihat pada gambar 2.4 berikut :



Gambar 2.4 Konsep Umum *Digital Signature*

2. 2. 4. 1. Proses Perhitungan Buat *Signature*

1. Cari nilai x

   $x = a^r mod\ p$ ..... (1)

   Dimana r = nilai random yang < q

   Contoh :

   x = $a^r mod\ p$

   = $3^5 (mod\ 23)$

   = 243 (mod 23)

   = 13

2. M = file yang akan dikirim

   h = H (M || X) ..... (2)

   || = or

   Contoh :

   Dimana diasumsikan pesan M = 12345

   h = H (M || X)

   = H (12345 || 13)

   h = H (m||x) (mod q)

   = H (12345||13) (mod 7)

   = a11784b83ea003cd66491c7e1de07296d9d9242c (hexadecimal)

   = 919671992759145855242593220263016201851705566252 (mod 7) (decimal)'

   = 5

3. Cari nilai y

Y = (r + (s x h)) (mod q) ..... (3)

Contoh :

Y = (5 + 4 x 5) (mod 7)

= 25 (mod 7)

= 4

4. Hasil signature

Signature = h ; y ..... (4)

Contoh :

(h , y) = (5,4)

$z = a^y . \lambda^h (mod\ p)$

$= 7^4 . 24^5 (mod\ 23)$

= (2401 x 7962624) (mod 23)

= 9

M = 12345 dan z :

h’ = H (m||z) (mod q)

= H(12345||9) (mod 7)

= 4

2. 2. 4. 2. Perhitungan Check Signature

Dimana :

M = file yang akan di check

Rumus :

1. H’ = SHA-256 (M || z)

   || = Or ……..... (5)

   Contoh :

   h’ = H (m||z) (mod q)

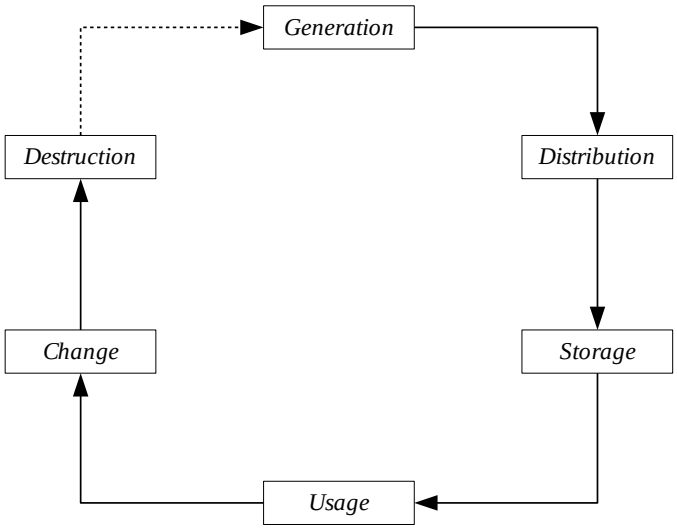
   = H(12345||9) (mod 7)

   = 4

2. - Jika h mod q = h’ mod q (valid) ..... (8)
   - Jika h mod q ≠ h’ mod q (tidak valid) ..... (9)

2. 2. 4. 3. Manajemen kunci

Kekuatan sistem kriptografi secara total bergantung pada keamanan kunci. Kunci perlu dilindungi selama fase daur hidupnya. Daur hidup kunci dimulai dari pembangkitan kunci (*generation*) sampai kunci tidak diperlukan lagi untuk kemudian dihancurkan (*destruction*). Secara garis besar, daur hidup kunci digambarkan pada Gambar 2.6 sebagai berikut.



Gambar 2.6 Daur hidup kunci

Tujuan manajemen kunci adalah menjaga keamanan dan integritas kunci pada semua fase di dalam daur hidupnya. Pada umumnya setiap kunci akhirnya diganti dengan kunci lain. Jadi, keseluruhan fase membentuk siklus (lingkaran) karena penghancuran kunci biasanya diikuti dengan penggantiannya dengan kunci baru (digambarkan dengan garis putus-putus). Manajemen kunci yang dibahas difokuskan pada algoritma kriptografi simetri karena manajemen kunci untuk algoritma kunci-publik sangat berbeda dengan algoritma simetri.

2. 2. 4. 4. Penggunaan Kunci (*key usage*)

Setiap kunci digunakan sesuai tujuannya. Misalnya ada kunci yang digunakan untuk mengenkripsi pesan, dan ada kunci yang digunakan untuk mengenkripsi kunci lainnya. Supaya setiap kunci mempunyai penggunaan yang unik, maka perlu diberi label pada setiap kunci, yang dalam hal ini label menspesifikasikan penggunaan kunci. Misalnya, label tersebut menspesifikasikan

‘kunci untuk mengenkripsi data’, ‘kunci untuk mengenkripsi kunci’, ‘kunci untuk pembangkitan bilangan acak’, dan sebagainya.

2. 2. 4. 5. Perhitungan Untuk Pembuatan Kunci

Proses pembentukan kunci pada perangkat lunak adalah sebagai berikut :

Dimana :

1. Pilih p yang bilangan prima
2. Pilih q yang bilangan prima dan (p-1)mod q
3. Pilih s dimana s < q
4. Pilih λ (a^smod p) mod p

   s = private

   λ = public

Contoh :

1. Pilih p = 23
2. Pilih q = 11

   = (p – 1) mod q

   = 22 mod 11

   = 0
3. Pilih s = 8

   = s < q

   = 8 < 11
4. Pilih λ (a^s mod p)^s mod p

   a =8

$\lambda = 8^{8}(mod\ 23)$

= 16777216 (mod 23)

= 4

Jadi, s = 8 dan λ = 4

2. 2. 5. Verifiable Ring Signature

Verifiable Ring Signature ini adalah metode yang akan digunakan dalam pembuatan perangkat lunak ini. *Verifiable Ring Signature* dapat digunakan oleh salah satu anggota kelompok pengguna yang berada dalam sebuah lingkar kerahasiaan. Untuk menandatagani sebuah dokumen, dengan menggunakan metode ini diperlukan kunci privat dari individu yang mengirim dan semua kunci publik dari anggota lingkar kerahasiaan tersebut. Kunci privat dan publik harus sama dalam lingkar kerahasiaan tersebut.

Verifiable ring signature ini mengambil sekuritas dari permasalahan menghitung logaritma diskrit. Metode ini juga menggunakan bilangan prima dalam proses pembentukan kuncinya. Proses pembentukan kunci privat dan publiknya sama seperti skema otentikasi, hanya saja metode tanda tangan digital ditambahkan sebuah fungsi *hash*.

2. 2. 6. Fungsi HASH SHA-256

Sebuah fungsi *hash* (*hash function* atau *hash algorithm*) adalah suatu cara untuk menghasilkan sebuah *digital* "*fingerprint*" kecil dari sembarang data. Fungsi ini memecahkan dan mencampurkan data untuk menghasilkan *fingerprint* yang sering disebut sebagai nilai *hash* (*hash value*). Nilai *hash* ini sering direpresentasikan dengan sebuah *string* pendek dari huruf-huruf dan angka-angka yang kelihatan acak (berbentuk heksadesimal). Sebuah fungsi *hash* yang baik adalah suatu fungsi yang tidak (jarang) memiliki output nilai *hash* yang sama untuk input yang berbeda.

Secure Hash Algoritm, SHA-256 ini dikembangkan oleh NIST (*National Institute of Standard and Technology*). SHA-256 dapat diterapkan dalam penggunakan tanda tangan digital. Untuk suatu pesan yang panjangnya $< 2^{64}$, SHA-256 akan menghasilkan keluaran sebanyak 256 bit dari pesan tersebut dan pesan keluaran itu disebut *message digest*. Panjang jarak *message digest* dapat berkisar antara 256 sampai 512 bit tergantung algoritmanya. Berdasarkan cirinya SHA-256 dapat digunakan dengan algoritma kriptografi lainnya seperti *Digital Signature* atau dalam generasi angka yang acak (*bits*).

SHA-256 dikatakan aman karena proses SHA-256 dihitung secara infisibel untuk mencari pesan yang sesuai untuk menghasilkan *message digest* atau dapat juga digunakan untuk mencari dua pesan yang berbeda yang akan menghasilkan *message digest* yang sama.

Menurut jenisnya SHA dapat dispesifikasikan menjadi 4 bagian yaitu: SHA-1, SHA-256, SHA-384, dan SHA-512. Berikut ini merupakan daftar-daftar properti dari keempat SHA.

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security[2] (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

Gambar 2.7 Daftar properti dari keempat SHA

Untuk SHA-256 ukuran blok pesan -m bit- dapat ditentukan tergantung dari algoritmanya. Pada SHA-256 masing-masing blok pesan mempunyai 512 bit dimana dapat dilakukan dengan 16 urutan sebesar 32 bit.

SHA-256 digunakan untuk menghitung *message digest* pada pesan atau *file* data yang diberikan sebagai *input*. Tujuan pengisian pesan adalah untuk menghasilkan total dari pesan yang diisi menjadi perkalian dari 512 bits.

2. 2. 7. Jaringan LAN

Jaringan LAN aalah jaringan yang dibatasi oleh area yang relatif kecil, umumnya dibatasi oleh area lingkungan seperti sebuah perkantoran di sebuah

gedung dan biasanya jangkauan tidak lebih dari 1 kilometer persegi. Beberapa model konfigurasi LAN biasanya berupa sebuah komputer yang dijadikan sebagai *file server* yang digunakan untuk menyimpan perangkat lunak ataupun sebagai perangkat lunak yang dapat digunakan oleh komputer-komputer yang terhubung kedalam jaringan lokal.

Komputer-komputer yang terhubung dengan suatu *file server* biasanya disebut *workstation*. Biasanya kemampuan *workstation* lebih kurang dibawah dari *file server*-nya dan mempunyai aplikasi lain di dalam media penyimpanannya selain aplikasi untuk jaringan. Kebanyakan LAN menggunakan media kabel untuk menghubungkan antara satu komputer dengan komputer lainnya. LAN merupakan jaringan komunikasi yang terbatas pada daerah yang kecil.

2. 2. 7. 1. Media Transmisi

Kabel adalah alat penghubung untuk mengirim informasi dari satu komputer ke komputer yang lain. Ada beberapa macam tipe kabel yang umum digunakan pada LAN. Dalam beberapa kasus sebuah jaringan lain menggunakan kabel yang berbeda. Pemilihan kabel berdasarkan topologi jaringan, protokol jaringan, dan ukurannya.

Adapun jenis-jenis kabel tersebut adalah :

1. Kabel *Unshielded Twisted Pair* (UTP)

Kabel *twisted pair* terdiri dari dua tipe yaitu *shielded* dan *unshielded*. *Unshielded twisted pair* (UTP) adalah yang paling populer dan umumnya merupakan pilihan yang terbaik untuk jaringan sederhana.

Kualitas kabel UTP berbeda dengan kabel telepon, kabel jenis ini mempunyai empat pasangan kabel di dalamnya. Setiap pasangan adalah jenis kabel kembar. Jenis konektor untuk kabel ini adalah konektor RJ-45.

2. Kabel *Shielded Twisted Pair* (STP)

Kekurangan kabel jenis ini adalah sangat sensitif terhadap sinyal radio dan listrik. Kabel seperti ini sangat baik digunakan dimana lingkungan pengaruh listrik kurang, serta biasanya digunakan pada jaringan yang menggunakan topologi *Token Ring*.

3. Kabel *Coaxial*

Kabel koaksial adalah kabel yang memiliki satu *Copper Conductor* pada bagian tengahnya. Sebuah lapisan plastik menutupi di antara konduktor dan lapisan pengaman serat besi. Lapisan serat besi tersebut membantu menutupi gangguan dari arus listrik, lalu lintas kendaraan atau mesin dan komputer.

Selain sangat sulit untuk konfigurasi, kabel ini pula sangat tidak tahan terhadap serangan dari sinyal-sinyal tertentu. Tetapi memiliki kelebihan karena dapat mendukung penggunaan kabel yang panjang di antara jaringan daripada kabel *Twisted Pair*. Ada dua jenis tipe kabel ini yaitu kabel *thick coaxial* dan kabel *thin coaxial*.

Kabel *thin coaxial* disebut juga dengan 10Base2 (*thinnet*) dimana angka 2 menunjukan pada panjang maksimum untuk setiap segmen kabel tersebut yaitu

200 meter, namun kenyataannya hanya dapat menjangkau sampai 185 meter. Kabel ini sangat populer terutama pada penggunaan jaringan yang linear.

Kabel *thick coaxial* disebut juga dengan 10Base5 (*thicket*) dimana angka 5 menunjukan pada panjang maksimum untuk setiap segmen kabel tersebut yaitu 500 meter, dan satu kekurangan dari kabel jenis ini adalah tidak lentur dan sangat relatif sulit untuk mengkonfigurasinya. Tipe konektor untuk kabel jenis ini adalah konektor *Bayone-Neill-Concelman* (BNC).

4. Kabel *Fibre Optic*

Kabel *fibre optic* (serat optik) mempunyai kemampuan mentransmisikan sinyal melalui jarak yang relatif jauh daripada kabel *coaxial* ataupun kabel *twisted*, serta memiliki kecepatan yang baik. Kabel ini sangat baik digunakan untuk fasilitas konferensi radio atau layanan interaktif. 10BaseF merujuk pada spesifikasi untuk kabel *fibre optic* dengan membawa sinyal *Ethernet*.

Ketika akan melakukan konfigurasi kabel ini ada beberapa petunjuk yang bisa dilakukan, diantaranya :

a. Hendaknya menggunakan kabel yang relatif panjang untuk menghindari adanya kekurangan.

b. Ujilah tiap-tiap bagian dari jaringan yang akan dikonfigurasi, meskipun hal tersebut dalam kondisi baru.

c. Jika kabel melintasi lantai sebaiknya dilindungi dengan *protector* kabel.

d. Berikan label pada masing-masing ujungnya.

e. Gunakan pengikat kabel untuk menyatukan kabel-kabel jangan menggunakan isolasi atau plester.

2. 2. 7. 2. Topologi LAN

Topologi secara fisik dari suatu jaringan lokal merujuk kepada konfigurasi kabel, komputer dan perangkat lainnya. Adapun jenis-jenis topologi LAN adalah sebagai berikut :

1. *Linear Bus* (Garis Lurus)

Topologi *linear Bus* terdiri dari satu jalur kabel utama dimana pada masing-masing ujungnya diberikan sebuah *terminator*. Semua *nodes* pada jaringan terkoneksi pada sebuah kabel utama (*backbone*). Jaringan-jaringan *Ethernet* dan *Local Talk* menggunakan topologi ini.

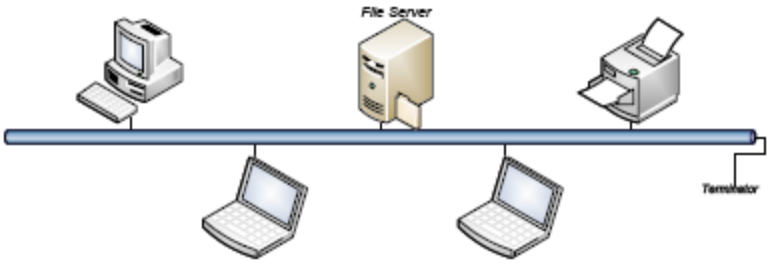
Kelebihan dari topologi *linear bus* adalah :

a. Mudah dalam mengkonfigurasi komputer atau perangkat lain ke dalam sebuah kabel utama.
b. Tidak terlalu banyak menggunakan kabel dibandingkan dengan topologi *star* / bintang.

Kekurangan dari topologi *linear Bus* adalah :

a. Seluruh jaringan akan mati jika ada kerusakan pada kabel utama.
b. Membutuhkan *terminator* pada kedua sisi kabel utamanya.
c. Sangat sulit mengidentifikasi permasalahan jika jaringan sedang *down* atau rusak.

d. Sangat tidak disarankan dipakai sebagai salah satu solusi pada penggunaan jaringan di gedung besar.



Gambar 2.8 1Topologi Linear Bus

Pada gambar 2.8 memperlihatkan topologi jaringan *linear bus*, pada gambar tersebut kita dapat melihat *backbone* dan *terminator* dari *backbone*.

2. *Star* (Bintang)

Topologi model ini dirancang yang mana setiap *nodes* terkoneksi ke jaringan melewati sebuah *concentrator*.

Data yang dikirim ke jaringan lokal akan melewati *concentrator* sebelum melanjutkan ke tempat tujuannya, *concentrator* akan mengatur dan mengendalikan keseluruhan fungsi jaringan dan juga bertindak sebagai *repeater*. Konfigurasi jaringan model ini menggunakan kabel *Twisted Pair* dan dapat digunakan pada kabel *coaxial* atau kabel *fibre optic*.

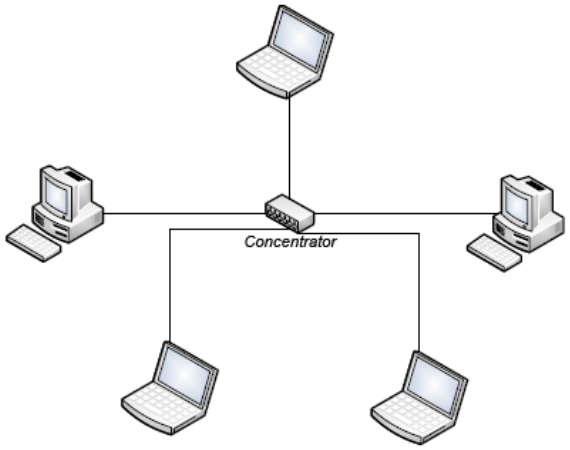
Kelebihan dari topologi *Star* (bintang) adalah :

a. Mudah dalam pemasangan dan pengkabelan.
b. Tidak mengakibatkan gangguan pada jaringan ketika akan memasang atau memindahkan perangkat jaringan lainnya.

c. Mudah untuk mendeteksi kesalahan dan memindahkan perangkat-perangkat lainnya.

Kekurangan dari topologi *Star* (bintang) adalah

a. Membutuhkan lebih banyak kabel daripada topologi *linear bus*.
b. Membutuhkan *concentrator* dan apabila *concentrator* rusak maka semua *node* yang terkoneksi tidak dapat terdeteksi.
c. Lebih mahal daripada topologi *linear bus* karena biaya untuk pembelian *concentrator*.



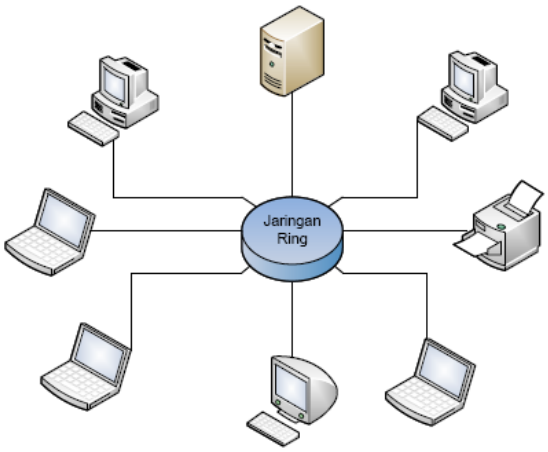
Gambar 2.9 Topologi Star

Pada gambar 2.9 memperlihatkan topologi jaringan *star*, pada gambar tersebut kita dapat melihat *concentrator* yang merupakan bagian paling vital dari topologi ini.

3. *Ring* (Cincin)

Topologi *Ring* (cincin) menggunakan teknik konfigurasi yang sama dengan topologi *star* tetapi pada topologi ini terlihat bahwa jalur media transmisi

menyerupai suatu lingkaran tertutup menyerupai cincin sehingga diberi nama topologi bintang dalam lingkaran atau *star-wired ring*.



Gambar 2.10 Topologi Ring

Pada gambar 2.10 di atas terlihat bahwa *concentrator* dari topologi *ring* berbentuk lingkaran tetapi sebenarnya yang berbentuk lingkaran itu adalah kabel untuk menghubungkan dari kartu jaringan ke *concentrator*

4. *Tree* (Pohon)

Topologi model ini merupakan perpaduan antara topologi *linear bus* dan *star*, yang mana terdiri dari kelompok-kelompok *workstation* dengan konfigurasi *star* yang terkoneksi ke kabel utama yang menggunakan topologi *Linear Bus*. Topologi ini memungkinkan untuk pengembangan jaringan yang telah ada dan memungkinkan untuk mengkonfigurasi jaringan sesuai dengan kebutuhan.

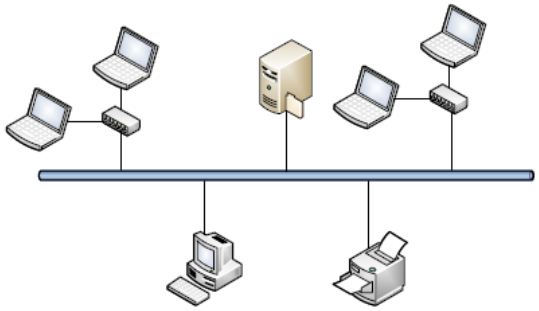
Kelebihan dari topologi *Tree* adalah :

a. Proses konfigurasi jaringan dilakukan dari titik ke titik pada masing-masing segmen.

b. Didukung oleh banyak perangkat keras dan perangkat lunak

Kekurangan dari topologi *Tree* adalah :

a. Keseluruhan panjang kabel pada tiap-tiap segmen dibatasi oleh tipe kabel yang digunakan.
b. Jika jaringan utama rusak maka keseluruhan segmen ikut rusak juga.
c. Sangat relatif sulit untuk dikonfigurasi dan proses pengkabelannya dibandingkan dengan topologi jaringan yang lain.



Gambar 2.11 Topologi Tree

Pada gambar 2.11 di atas terlihat bahwa topologi *tree* merupakan gabungan dari beberapa topologi.

2. 2. 7. 3. Model Hubungan Pada LAN

a. *Peer-to-peer*

Model hubungan *peer to peer* memungkinkan *user* membagi sumber dayanya yang ada di komputernya baik berupa *file*, layanan *printer* dan lain-lain serta mengakses sumber daya yang terdapat pada komputer lain. Namun model ini tidak mempunyai sebuah *file server* atau sumber daya yang terpusat. Pada model

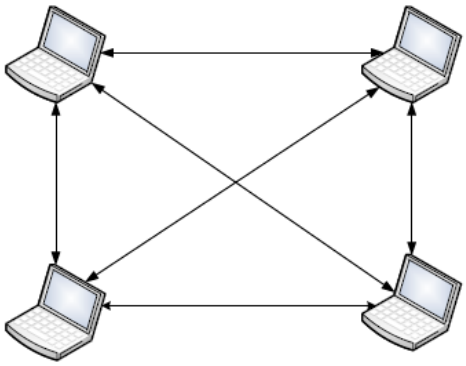
ini seluruh komputer adalah sama, yang mana mempunyai kemampuan yang sama untuk memakai sumber daya yang tersedia di dalam jaringan. Model jaringan ini didesain untuk jaringan berskala kecil dan menengah.

Kelebihan model hubungan *peer to peer* adalah :

a. Tidak terlalu mahal, karena tidak membutuhkan *dedicated file server*.
b. Mudah dalam konfigurasi programnya, hanya tinggal mengatur untuk operasi model hubungan *peer to peer*.

Kekurangan model hubungan *peer to peer* adalah :

a. Tidak terpusat, terutama untuk penyimpanan data dan aplikasi.
b. Tidak aman, karena tidak menyediakan fasilitas untuk keperluan itu.



Gambar 2.12 Model Hubungan *Peer ro Peer*

Pada gambar 2.12 memperlihatkan model hubungan *peer to peer* pada suatu jaringan.

b. Client / Server

Model hubungan *Client / Server* memungkinkan jaringan untuk mensentralisasi fungsi dan aplikasi kepada satu atau dua *dedicated file server*.

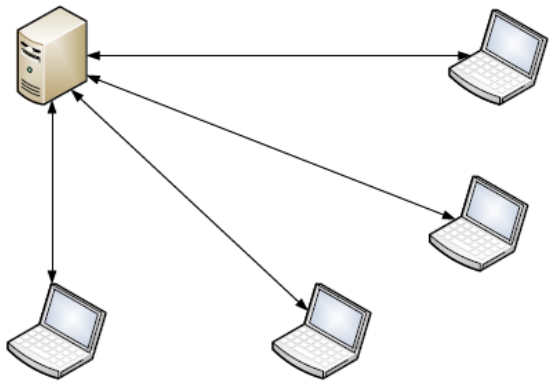
Sebuah *file server* menjadi jantung dari keseluruhan sistem, memungkinkan untuk mengakses sumber daya dan menyediakan keamanan. *Workstation* yang berdiri sendiri dapat mengambil sumber daya yang ada pada *file server*. Model hubungan ini menyediakan mekanisme untuk mengintegrasikan seluruh seluruh komponen yang ada di jaringan dan memungkinkan banyak pengguna secara bersama-sama memakai sumber daya pada *file server*.

Kelebihan model hubungan *Client Server* adalah :

a. Terpusat (sumber daya dan keamanan data terkontrol melalui server)
b. Skalabilitas
c. Fleksibel
d. Teknologi baru dengan mudah terintegrasi ke dalam sistem
e. Keseluruhan komponen (*client / network / server*) dapat bekerja bersama

Kekurangan model hubungan *Client Server* adalah :

a. Mahal
b. Membutuhkan investasi untuk *dedicated file server*
c. Perbaikan (Jaringan besar membutuhkan seorang staff untuk mengatur agar sistem berjalan secara efisien)
d. Berketergantungan
e. Ketika *server* jatuh, mengakibatkan keseluruhan operasi pada *network* akan jatuh pula



Gambar 2.13 Model Hubungan Client Server

Pada gambar 2.13 memperlihatkan model hubungan *client server* pada suatu jaringan.

2. 2. 8. Pemodelan Sistem

Dalam suatu proses pembuatan software, analisa dan rancangan telah merupakan terminologi yang sangat tua. Pada saat masalah ditelusuri dan spesifikasi dinegisiasikan, dapat dikatakan bahwa kita berada pada tahap rancangan. Merancang adalah menemukan suatu cara untuk menyelesaikan masalah, salah satu tool/model untuk merancang pembangunan software yang berbasis *object-oriented* adalah UML. Alasan mengapa UML digunakan adalah, pertama, *scalability* dimana objek lebih mudah dipakai untuk menggambarkan sistem yang besar dan komplek. Kedua, *dynamic modeling*, dapat dipakai untuk pemodelan sistem dinamis dan real time. Sebagaimana dalam tulisan pertama, penulis menjelaskan konsep mengenai obyek, OOA&D (*Obyek Oriented Analyst/ Design*) dan pengenalan UML, maka dalam tulisan kedua ini lebih ditekankan pada cara bagaimana UML digunakan dalam merancang sebuah pengembangan *software* yang disertai gambar atau contoh dari sebuah aplikasi.

2. 2.8. 1. **UML (Unified Modelling Languge)**

Unified Modelling Language (UML) adalah sebuah"bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

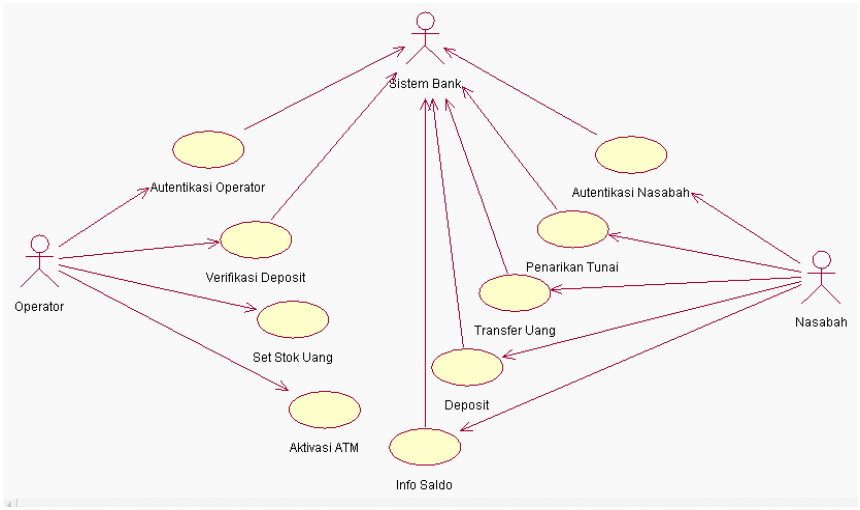
Dengan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek.

Seperti bahasa-bahasa lainnya , UML mendefinisikan notasi dan syntax. notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk bentuk tersebut dapat di kombinasikan. Notasi Uml terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : Grady Booch OOD (*Object Oriented Design*), Jim Rumbaugh OMT(Object Modeling Technique) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*).

2. 2.8. 2. **Use Case Diagram**

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan

"bagaimana". Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.



Gambar 2.14 Use Case Diagram

2. 2.8. 3. **Use Case**

Sebuah *use case* menggambarkan suatu urutan interaksi antara satu atau lebih aktor dan sistem. Dalam fase *requirements*, model *use case* mengambarkan sistem sebagai sebuah kotak hitam dan interaksi antara aktor dan sistem dalam suatu bentuk naratif, yang terdiri dari input user dan respon-respon sistem. Setiap use case menggambarkan perilaku sejumlah aspek sistem, tanpa mengurangi struktur internalnya. Selama pembuatan model use case secara pararel juga harus ditetapkan obyek-obyek yang terlibat dalam setiap use case.

2. 2.8. 4. **Aktor**

Seorang aktor mencirikan suatu bagian *outside user* atau susunan yang berkaitan dengan *user* yang berinteraksi dengan sistem [*Rumbaugh, Booch, da*

Jacobson 1999]. Dalam model use case, aktor merupakan satu-satunya kesatuan eksternal yang berinteraksi dengan sistem. Terdapat beberapa variasi bagaimana aktor dibentuk [Fowler dan Scott 1999]. Sebuah aktor sering kali merupakan manusia (*human user*). Pada sejumlah sistem informasi, manusia adalah satu-satunya aktor. Dan mungkin saja dalam sistem informasi, seorang aktor bisa saja menjadi suatu sistem eksternal. Pada aplikasi real-time dan distribusi, sebuah aktor bisa saja menjadi satu perangkat eksternal I/O atau sebuah alat pengatur waktu. Perangkat eksternal I/O dan pengatur waktu aktor secara khusus lazimnya berada dalam real-time yang tersimpan dalam sistem (*real-time embedded systems*), sistem berinteraksi dengan lingkungan eksternal melalui sensor dan aktuator.

Primary actor (aktor utama) memprakarsai sebuah use case. Jadi, suatu primary aktor memegang peran sebagai proaktif dan yang memulai aksi dalam sistem. Aktor lainnya yang berperan sebagai *secondary* aktor bisa saja terlibat dalam use case dengan menerima *output* dan memberikan *input*. Setidaknya satu actor harus mendapatkan nilai dari use case. Biasanya adalah *primary* aktor (actor utama). Bagaimanapun, dalam *real-time embedded systems*, *primary* aktor dapat berperan sebagai perangkat eksternal I/O atau pengatur waktu, penerima utama dari *use case* bisa menjadi *secondary human* aktor yang menerima sejumlah informasi dari sistem.



Gambar 2.16 Aktifitas aktor dan user case

2. 2.8. 5. **Class Diagram**

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

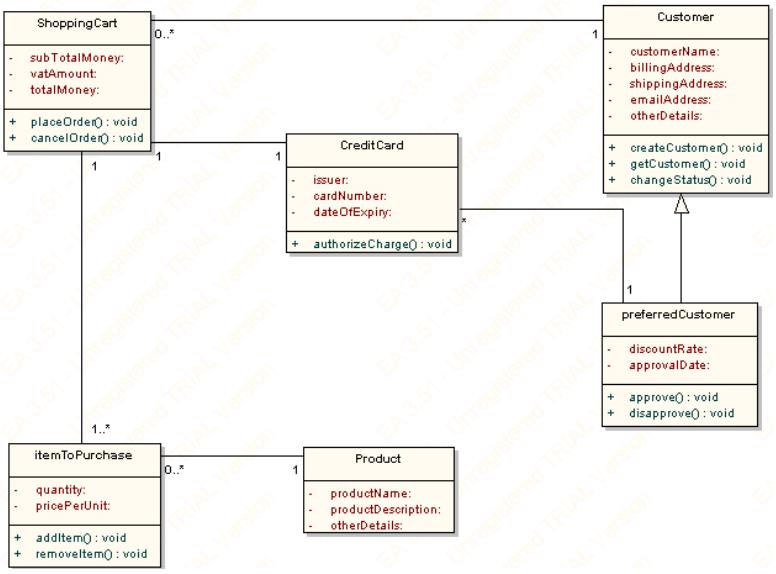
Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment,* pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (*danstereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. Private, tidak dapat dipanggil dari luar class yang bersangkutan
2. Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
3. Public, dapat dipanggil oleh siapa saja

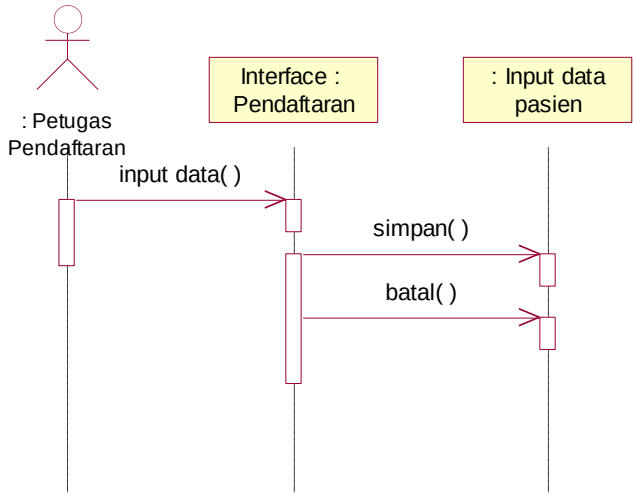


Gambar 2.17 Class Diagram

2. 2.8. 6. **Sequence Diagram**

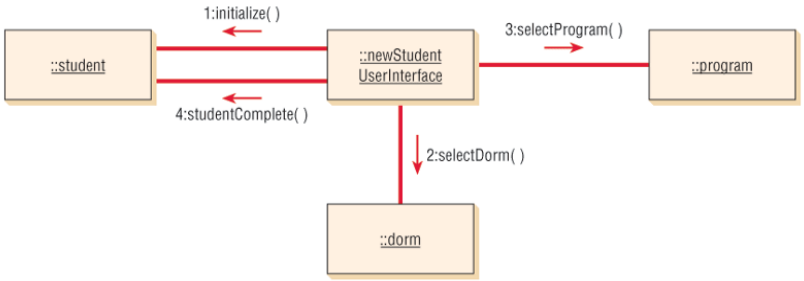
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara *internal* dan *output* apa yang dihasilkan.



2. 2.8. 7. **Collaboration Diagram**

Menunjukan informasi yang sama seperti dalam sequence diagram. Perbedaannya adalah cara mengelompokan object. Sequence berdasarkan urutan nomber dari pesan.



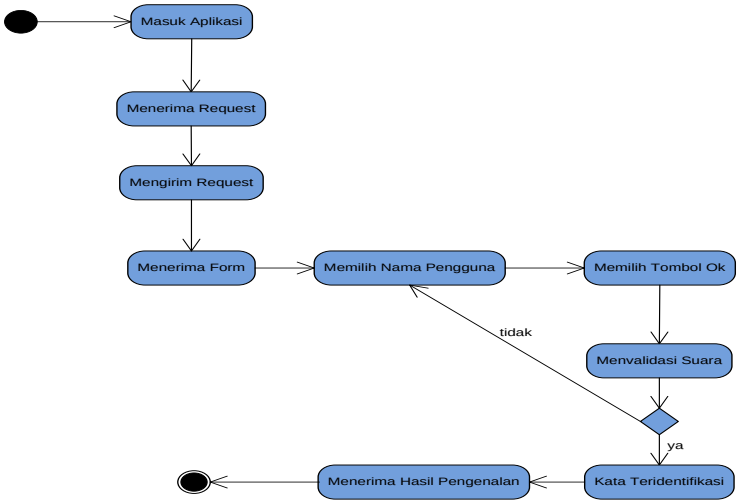
Gambar 2.19 Collaboration Diagram

2. 2.8. 8. **Activity Diagram**

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang

mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

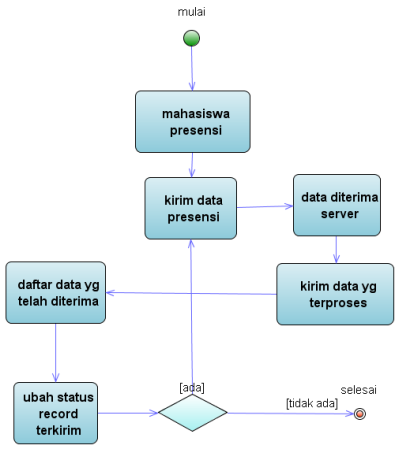
Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu activity diagram tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.



Gambar 2.20 1Activity Diagram

2. 2.8. 9. **Statechart Diagram**

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke *state* lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya statechart diagram menggambarkan class tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).



Gambar 2.21 StateChart Diagram

2. 2. 9. Metode Pengembangan Perangkat Lunak

Merupakan sebuah model *chaos* yang menggambarkan "perkembangan P/L sebagai sebuah kesatuan dari pemakai ke pengembang dan ke teknologi" disebut dengan "Prescriptive" karena Menentukan sekumpulan elemen proses (Aktivitas, Aksi, tugas, produk kerja, jaminan kualitas, dll untuk setiap proyek), setiap model proses juga menentukan alur kerjanya.

Pada saat kerja bergerak maju menuju sebuah sistem yang lengkap, keadaan yang digambarkan secara rekursif diaplikasikan kepada kebutuhan pemakai dan spesifikasi teknis P/L pengembang Saat ini, prescriptive memberikan jawaban secara definitive untuk masalah pengembangan P/L dalam setiap perubahan lingkungan komputasi

2. 2. 9. 1. Model Skuensial Linear

Pada Tugas akhir ini menggunakan metode penggunaan perangkat lunak siklus kehidupan klasik atau model air terjun. Model ini mengusulkan sebuah pendekatan kepada perkembangan software yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas sebagai berikut :

1. Rekayasa dan pemodelan sistem/informasi

Karena sistem merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke software tersebut. Pandangan sistem ini penting ketika software harus berhubungan dengan elemen-elemen yang lain seperti software, manusia, dan database. Rekayasa dan anasisis system menyangkut pengumpulan kebutuhan pada tingkat sistem dengan sejumlah kecil analisis serta disain tingkat puncak. Rekayasa informasi mancakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

2. Analisis kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khusunya pada *software*. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan interface yang diperlukan. Kebutuhan baik untuk sistem maupun software didokumentasikan dan dilihat lagi dengan pelanggan.

3. Desain

Desain software sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang berbeda; struktur data, arsitektur software, representasi interface, dan detail (algoritma) prosedural. Proses desain menterjemahkan syarat/kebutuhan ke dalam sebuah representasi software yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, desain didokumentasikan dan menjadi bagian dari konfigurasi software.

4. Generasi Kode

Desain harus diterjemahkan kedalam bentuk mesin yang bias dibaca. Langkah pembuatan kode melakukan tugas ini. Jika desain dilakukan dengan cara yang lengkap, pembuatan kode dapat diselesaikan secara mekanis.

5. Pengujian

Sekali program dibuat, pengujian program dimulai. Proses pengujian berfokus pada logika internal *software*, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarahkan pengujian untuk menemukan kesalahan – kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesuai dengan hasil yang dibutuhkan.

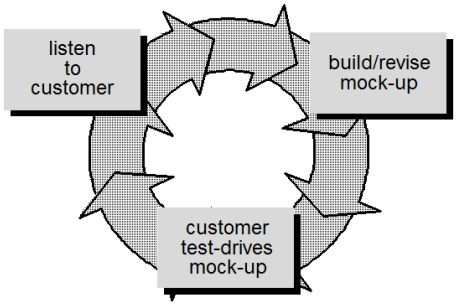
6. Pemeliharaan

Software akan mengalami perubahan setelah disampaikan kepada pelanggan (perkecualian yang mungkin adalah *software* yang dilekatkan). Perubahan akan terjadi karena kesalahan – kesalahan ditentukan, karena software

harus disesuaikan untuk mengakomodasi perubahan – perubahan di dalam lingkungan eksternalnya (contohnya perubahan yang dibutuhkan sebagai akibat dari perangkat peripheral atau sistem operasi yang baru), atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja. Pemeliharaan software mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

2. 2. 10. Model Prototype

Model prototype dibangun dari mengumpulkan berbagai kebutuhan, kemudian tim pengembang akan nertemu dengan pelanggan untuk menentukan tujuan dari perangkat lunak, dan mengidentifikasi kebutuha-kebutuhan yang telah diketahui oleh pelanggan, dan batasan-batasan apa saja yang dapat dikategorikan sebagai tugas utama. Hasilnya akan dibangun rancangan sementara yang mewakili berbagai aspek dari perangkat lunak yang kelak akan digunakan oleh pelanggan/pengguna (seperti bentuk pendekatan input yang digunakan dan bentuk output). Idealnya model prototype melayani sebuah mekanisme untuk mengidentifikasi kebutuhan perangkat lunak. Dimana jika nantinya sebuah model prototype berhasil dibuat, seorang developer harus berusaha mendayagunakan tools yang ada (semisal, report generator, windows manager) dapat bekerja dengan baik (cepat).



Gambar 2.22 Model Prototype

2. 2. 11. Sistem Operasi Java

Sistem operasi biasanya ditulis dalam sebuah kombinasi dari kode bahssa C dan assembly, terutama disebabkan oleh kelebihan performa dari bahasa tersebut dan memudahkan komunikasi dengan perangkat keras. Satu kesulitan dalam merancang sistem basis bahasa adalah dalam hal proteksi memori, yaitu memproteksi sistem opeasi dari pemakai program yang sengaja memproteksi pemakai program lainnya. Sistem operasi tradisional mengaharapkan pada

tampilan perangkat keras untuk menyediakan proteksi memori. Sistem basis bahasa mengandalkan pada tampilan keamanan dari bahasa. Sebagai hasilnya, sistem basis bahasa menginginkan pada alat perangkat keras kecil, yang mungkin kekurangan tampilan perangkat keras yang menyediakan proteksi memori.

Java adalah sebuah teknologi yang diperkenalkan oleh Sun Microsystems pada pertengahan tahun 1990. Menurut definisi dari Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. Kita lebih menyukai menyebut Java sebagai sebuah teknologi dibanding hanya

sebuah bahasa pemrograman, karena Java lebih lengkap dibanding sebuah bahasa pemrograman konvensional. Teknologi Java memiliki tiga komponen penting, yaitu:

a. Programming-language specification
b. Application-programming interface
c. Virtual-machine specification

2. 2. 11. 1. Java2

Java2 adalah generasi kedua dari Java platform (generasi awalnya adalah Java Development Kit). Java berdiri di atas sebuah mesin interpreter yang diberi nama JVM. JVM inilah yang akan membaca bytecode dalam file .class dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu, bahasa Java disebut sebagai bahasa pemrograman yang portable karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM. Platform Java terdiri dari kumpulan library, JVM, kelas-kelas loader yang dipaket dalam sebuah lingkungan rutin Java, dan sebuah compiler, debuger, dan perangkat lain yang dipaket dalam Java Development Kit (JDK). Java2 adalah generasi yang sekarang sedang berkembang dari platform Java.

2. 2. 11. 2. J2SE (Java Standard Edition)

J2SE (Java2 Second Edition). J2SE adalah kelompok dari beberapa API (Application Programming Interface) dari Java dan teknologi selain Java. J2SE adalah lingkungan dasar dari Java.