BAB II

LANDASAN TEORI

2.1 Keamanan Data

Data dan informasi yang bersifat rahasia harus dilindungi dari pihak luar. Data dan informasi yang dapat terbaca oleh pihak luar dapat menimbulkan kerugian. Banyak aspek yang memungkinkan pihak luar dapat membaca data yang sudah tersimpan, antara lain; kurang kuatnya sistem pengaman dari sisi sistem komputer atau jaringan itu sendiri. Hal-hal yang perlu kita pertimbangkan menyangkut keamanan data diantaranya (Penerbit Andi. (2003). Memahami Model Enkripsi dan Security Data, hal: 1-4):

a. Keamanan fisik

   Komputer harus diletakan pada tempat yang dapat dikontrol karena kemungkinan penyalah-gunaan dapat terjadi (misalnya: user yang tidak disiplin, meninggalkan komputer dalam keadaan on/ hidup, sehingga orang lain yang tidak berhak dapat menggunakan fasilitas komputer tersebut).

b. Keamanan akses

   Seluruh akses terhadap sistem komputer secara administrasi harus terkontrol dan terdokumentasi, sehingga apabila ada suatu permasalahan dapat diketahui penyebabnya dan mencari solusi/ pemecahannya.

c. Keamanan file

   File atau data yang sensitif dan bersifat rahasia, memerlukan tingkatan akses keamanan dengan dibuatkan suatu kode sandi tertentu. Apabila file atau data tersebut dicuri maka isi informasinya tidak mudah didapatkan.

d. Keamanan jaringan

Bila pemanfaatan jaringan *public* dilakukan maka data yang ditransmisikan dalam jaringan harus aman dari kemungkinan dapat diketahui isi informasinya sehingga untuk informasi yang sensitif harus dibuatkan juga kode sandi tertentu untuk pengamanan pada saat transaksi.

2.1.1 Aspek yang berkaitan dengan persyaratan keamanan

Aspek yang berkaitan dengan persyaratan keamanan antara lain:

a. Secrery

Berhubungan dengan akses membaca data dan informasi. Data dan informasi dalam suatu sistem komputer hanya dapat di akses oleh orang yang berhak.

b. Integrity

Berhubungan dengan akses merubah data dan informasi. Data dan informasi yang berada di dalam suatu sistem komputer hanya dapat dirubah oleh orang yang berhak.

c. Availability

Berhubungan dengan ketersediaan data dan informasi. Data dan informasi yang berada dalam suatu sistem komputer tersedia dan hanya dapat dimanfaatkan oleh orang yang berhak.

2.1.2 Aspek yang berkaitan dengan ancaman keamanan

Aspek yang berkaitan dengan ancaman keamanan antara lain:

a. Interruption

Merupakan ancaman terhadap *avaibility*, yaitu data dan informasi yang berada dalam sistem komputer dirusak atau dibuang, sehingga menjadi tidak ada dan

tidak berguna. Contohnya: *hard-disk* yang dirusak, memotong line komunikasi.

b. Interception

Merupakan ancaman terhadap *secrery*, yaitu orang yang tidak berhak namun berhasil mendapatkan akses informasi dari dalam sistem komputer. Contohnya: dengan menyadap data melalui jaringan *public* atau menyalin secara ilegal file ataupun program.

c. Modification

Merupakan ancaman terhadap integritas, yaitu orang yang tidak berhak namun berhasil mendapatkan akses informasi dari dalam sistem komputer juga telah melakukan perubahan terhadap informasi. Contohnya: merubah program dan *coding*.

d. Fabrication

Merupakan ancaman terhadap integritas juga, yaitu orang yang tidak berhak yang meniru atau memalsukan suatu objek ke dalam sistem. Contohnya: dengan menambahkan suatu *record* ke dalam file.

2.2 Pengertian Dasar Kriptografi

Definisi dari *cryptography* adalah sebagai berikut:

"*Cryptography* adalah suatu ilmu ataupun seni mengamankan pesan (Munir, Rinaldi. (2006). Kriptografi, hal: 2), dan dilakukan oleh *cryptographer*. Sedangkan *crypanalysis* adalah suatu ilmu dan seni membuka (*breaking*) *ciphertext* dan orang yang melakukannya disebut *cryptanalyst*. *Cryptographic system* atau *cyptosystem* adalah suatu fasilitas untuk mengkonversikan *plaintext* ke *ciphertext* dan sebaliknya."

Definisi dari enkripsi adalah sebagai berikut:

"Enkripsi adalah sebuah proses yang melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti (tidak terbaca). Enkripsi dapat diartikan sebagai kode atau *cipher*."

Pengertian lain dari kriptografi ialah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, ke-absahan data, integritas data, serta autentikasi data (Munir, Rinaldi. (2006). Kriptografi, hal: 2).

Proses enkripsi adalah sebuah proses yang dilakukan untuk menyembunyikan isi pesan yang ada dalam sebuah pesan. Hal ini dimaksudkan agar isi pesan tersebut hanya dapat dipahami oleh pihak tertentu, yang ditujukan oleh pengirim pesan tersebut.

Sedangkan proses dekripsi (standar nama menurut ISO 7498-2) adalah kebalikan dari proses enkripsi, yaitu suatu proses menterjemahkan pesan yang telah dienkripsi kembali menjadi bentuk semula, sehingga pesan tersebut dapat dimengerti maksudnya. Data asli sebuah pesan disebut *plaintext* dan hasil enkripsinya disebut *ciphertext*. Ukuran *ciphertext* bisa lebih kecil, lebih besar atau sama dengan ukuran *plaintext*.

Berikut ini adalah gambaran proses enkripsi dan dekripsi.

Gambar 2.1 Proses Enkripsi dan Dekripsi

Keterangan gambar:

a. *Plaintext* adalah data aslinya yang belum diacak

b. *Ciphertext* adalah data yang telah diacak

c. E = proses enkripsi

d. D = proses dekripsi

Selanjutnya *plaintext* dinotasikan dengan P. *Ciphertext* dinotasikan dengan C yang ukurannya lebih kecil, lebih besar atau sama dengan ukuran P. Fungsi enkripsi E berfungsi untuk mengubah P menjadi C dengan notasi matematikanya **E(P)=C.** Sedangkan fungsi dekripsi D berfungsi untuk mengubah C menjadi P dan dalam matematika dinotasikan dengan **D(C)=P**.

Pada dasarnya kriptografi memiliki algoritma yang berupa fungsi matematika yang saling berhubungan, dimana satu fungsi digunakan untuk proses enkripsi data dan yang lainya digunakan untuk proses dekripsi data. Kedua fungsi ini merupakan fungsi utama yang selalu berkaitan antara yang satu dengan yang lainnya. Jika salah satu fungsi ini tidak bekerja maka akan terjadi kesalahan.

2.3 Metode Enkripsi Data

Enkripsi (standar nama menurut ISO 7498-2) adalah sebuah proses yang melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti secara langsung. Enkripsi seperti pemaparan diatas dapat diartikan sebagai kode atau *chiper*. Sebuah sistem pengkodean menggunakan suatu tabel atau kamus yang telah di definisikan untuk mengganti kata dari informasi atau yang merupakan bagian informasi yang dikirim.

Sebuah *chiper* menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (*stream*) bit dari sebuah pesan menjadi *cryptogram* yang tidak bisa dimengerti. Teknik *chiper* merupakan suatu sistem yang telah siap untuk di automasi. Teknik ini cukup ampuh digunakan dalam sistem keamanan komputer dan jaringan. Teknik *stream chiper* yang akan dipakai dalam penulisan ini ialah *Stream cipher SCOP (Secure Coprosessor).* Scop sendiri merupakan pengembangan dari *PIR* (*Private Information Retrieval*) yang digunakan pada platforrm *IBM 4758 secure prosessor* pada sistem operasi Linux. Tujuan awal dari PIR itu

sendiri adalah untuk menyembunyikan karakter ganda agar tidak dapat terbaca melalui enkripsi dataset-nya (Sean Smith, Department of Computer Science Dartmouth College)

Scop diciptakan oleh Simeon V. Malthev dan Peter T. Antonov. Scop yang merupakan pengembangan dari *PIR* diciptakan agar dapat bekerja pada kinerja software prosesor intel pentium, Scop memiliki keunggulan karena dapat mengerjakan 1,5 *clock cycles* perbyte. Algoritma enkripsi Scop tidak di patenkan dan dapat digunakan pada berbagai penggunaan tanpa pembayaran royalti kepada penciptanya. Penjelasan diatas diterangkan bahwa Scop didisain khusus untuk digunakan pada prosesor Pentium, tetapi dapat berjalan sangat cepat pada prosesor 32-bit lainya (Nopember 27, 2007 oleh Sandromedo). Dasar pengembangan Scop itu sendiri untuk menyelesaikan permasalan *privacy* pada dataset yang besar dengan menggunakan co-prosesor yang kecil. Scop juga didisain untuk memenuhi kriteria sebagai berikut:

a. Scop dapat berjalan pada memori kurang dari 2.5 kbyte.

b. Implementasi software yang sederhana, Scop menggunakan operasi sederhana seperti penambahan dan pengurangan modulus 2^{32}.

c. Keamanan Variabel, Scop merupakan algoritma ukuran kunci Variabel, dengan kunci sampai dengan 384 bit.

Algoritma ini mempunyai dua bagian yaitu *key expansion* dan enkripsi data, bagian pertama merubah kunci, maksudnya adalah *key* pada setiap data dilakukan penyandian ulang hingga panjang key 48 byte menjadi array 1540 bytes.

K*ey expansion* ini berdasarkan pada fungsi hash satu arah yang dinamakan GP8. Fungsi hash digunakan dalam kriptografi untuk membagi atribut yang mirip. Operasi utama yang digunakan dalam proses *cipher* adalah penambahan aritmatik dari 32-bit *words*.

Scop menggunakan sebuah 32-bit S-box (dinamai V) dengan 384 entri, dua buah indek 8-bit i dan j, serta tiga buah variabel sementara T1, T2, T3. S-box dibagi dua: pertama

merupakan 128 32-bit *word* dari V membentuk bagian pertama yang statis dan kemudian 256 32-bit *word* dari V membentuk bagian kedua yang berubah-ubah selama proses enkripsi. Indek j menunjuk hanya bagian yang dinamis, sedangkan indek i menunjuk bagian statis.

Keystream K 32-bit dihasilkan dengan langkah-langkah sebagai berikut:

1. T1 = V[128 + j]
2. j = (j + T3) mod 256
3. T2 = V[128 + j]
4. T3 = V[128 + j] + V[i]
5. V[128 + j] = T3
6. i = (i + 1) mod 256
7. j = (j + T2) mod 256
8. K = T1 + T2

K kemudian ditambahkan pada *plaintext* untuk menghasilkan *ciphertext* atau dikurangkan dari *ciphertext* untuk menghasilkan *plaintext*.

Proses awal dari algoritma diatas adalah dengan memberikan nilai kepada karakter pertama yang ingin dienkripsi. Contoh pada penulisan ini misal karakter "t", karakter t pada urutan ASCII memiliki nilai 84, yang berati T1= 84. selanjutnya nilai t tadi dilakukan perhitungan dengan urutan dan langkah seperti diatas. Maka ditemukan nilai hasilnya adalah =252 dalam urutan ASCII 252 adalah karakter "ü". Hasil pengacakan tersebut adalah t = ü, hasil ini adalah hasil yang didapat dari perhitungan secara manual dari algoritmanya, tetapi

dalam proses secara program, karakter awal harus mengalami kompresi terlebih dahulu lalu dilakukan proses enkripsi

Secara teknis urutan *key-stream* 32 didapatkan berdasarkan urutan dan langkah algoritmanya. Berikut adalah algoritma (*coding*) yang disesuaikan.

```
package my.project.scop;
    public String encrypt(String key, String file){
  byte[] v = new byte[32];
          for(int i = 0; i<32; i++){
    v[i] = state (substr(key, file, i)); ( langkah pertama )
  }

  byte i = 0;    ( variable sementara )
  byte t1 = 0;   ( variable sementara )
  byte t2 = 0;   ( variable sementara )
  byte t3 = 0;   ( variable sementara )

  for(int j = 0; j < 32; j++){        ( langkah penambahan untuk nilai j )
      t1 = v[j];                       ( nilai variable T1 )
      j = (j+t3)%256;                    ( proses mencari nilai j )
      t2 = v[j];                           ( nilai variable T2 )
      t3 = (byte)(v[j] + v[i]);              ( nilai variable T3 )
      v[i] = t3;                               ( pencocokan vaiable )
      i= (byte)((i+1)%256);                      ( langkah penambahan untuk nilai i )
      j=(byte)(j+t2)%256;                          ( proses mencari nilai j )
      K = t1 + t2;                                   ( nilai K )

      i = (byte)((i+1)%256);
      j = (byte) ((j+t2)%256);
      byte K = (byte) (t1+t2);
      for(int l =0; l<cip.length(); l++){
          enc[i]=K+state(substr(file,i,1));      ( proses enkripsi )
          dec[i] = state(substr(cip,i))-K;       ( proses dekripsi )
          cip.=chr(enc[i]);

          txt .= chr(dec[i]);
```

Prosesnya adalah pada "langkah pertama" sistem menerima input berupa data (aliran data), lalu kemudian di eksekusi oleh program dengan perintah:

"v[i] = state (substr(key, file, i));".

Perintah ini adalah perintah untuk memanggil perintah lain, yaitu perintah untuk proses kompresi. Setelah sistem menerima input data maka akan dilakukan proses kompresi

(pemadatan data), maka variable "v[i]" mempunyai nilai berupa karakter pertama dari file hasil kompresi. Berikut adalah urutan langkahnya:

Proses 1:

T1 = V[128 + j]

V = sbox (didapat dari hasil kompresi sebelumnya dengan perintah:

"substr(key, file, i"). Sedangkan 128 adalah bit array.

T1 = 84 + j (0) = 84.

T1 = 84.

Proses 2:

Setelah sbox mendapatkan nilai maka langkah selanjutnya adalah:

j = (j + T3) mod 256.

diketahui nilai j didapat dengan " for(int j = 0; j < 32; j++)".

karena proses awal j = 0.

variable T3 = 0 ,maka: j = (0 + 0)mod 256 = 0.

nilai j = 0.

Proses 3:

T2 = V[128 + j]

Diketahui pada proses 1 diatas, maka nilai

T2 = 84.

Proses 4:

Pada " t3 = (byte)(v[j] + v[i]);" untuk mencari nilai T3 maka,

T3 = V[128 + j] + V[i]

Diketahui V[128+j] = 84

V[i] = 84

Nilai T3 adalah = 84 + 84 = 168.

Proses 5:

V[128 + j] = T3

Pada " v[i] = t3;" maka sbox v[i] adalah nilai T3 = 168

Proses 6:

i = (i + 1) mod 25

pada " i= (byte)((i+1)%256);", diketahui variable i = 0, maka:

i = (i + 1) mod 256 = 1. jadi variable i = 1.

Proses 7:

j = (j + T2) mod 256

pada "j=(byte)(j+t2)%256;" diketahui nilai j = 0, maka

j = (0 + 84) mod 256 = 84

Proses 8:

K = T1 + T2, proses ini adalah proses terakhir untuk memperoleh nilai dari masukan T1, yang kemudian dilakukan enkripsi. Nilai K adalah 168 berarti hasil enkripsi (*chipertext*) dari variabel awal yang diinput menjadi nilai K. Untuk proses enkripsinya adalah K ditambahkan v awal, dan dikurangkan "cip" untuk mendapatkan dekripsinya.

Enkripsi = " enc[i]=K+state(substr(file,i,1)); ".

= 168 + 84 = 252

Dekripsi = "dec[i] = state(substr(cip,i))-K; ".

84 – 252 = 168

Java memiliki aplikasi siap-bangun (*built-in*) dikhususkan layanan untuk kriptografi yaitu, JCE *(Java Cryptography Extension*). JCE merupakan kerangka kriptografi berbasis Java yang dimulai untuk J2SE pada versi 1.4 hingga J2EE. JCE itu sendiri sudah memakai algoritma hash untuk kriptografi antara lain digunakan: DES, Triple_DES, Blowfish, md5 dan sha-1. Proses kriptografi yang memakai prosedur enkripsi yang sudah ada, cukup aman bila digunakan.

2.3.1 Algoritma Hash.

Fungsi hash mengurangi data dari ukuran yang berubah-ubah menjadi ukuran yang khusus. Fungsi hash dibutuhkan untuk konfigurasi sistem agar memudahkan pengecekan terhadap kelebihan data, utamanya terhadap data berkapasitas besar. Bila waktu pengecekan terlalu lama maka pesan dianggap tidak valid. Fungsi hash digunakan dalam kriptografi yaitu dalam hal membagi atribut yang mirip. Berbagai tipe fungsi hash dengan nilai bitnya adalah: md5 (128 bit), sha-1 (160 bit), ripemd-160 (160 bit), DDS (320 bit). Fungsi hash dengan kemampuan bit yang besar dapat mengatasi pengecekan data kapasitas besar dengan waktu lebih singkat, caranya yaitu data pada setiap atribut yang mirip maka pesan dibagi dalam ukuran yang lebih kecil dan panjang yang berubah-ubah dari teks dibuat dalam ringkasan pesan.

Bagaimanapun juga, fungsi hash dalam kriptografi dapat dibuat oleh siapapun, tetapi biasanya sering dikombinasikan dengan fungsi kriptografi yang punya integritas. Scop sebagai fungsi kriptografi dan merupakan kriptografi fungsi hash yang diubah dalam bentuk *bytecote* kemudian di integrasikan kedalam JCE yang merupakan API dari Java2 *Enterprise Edition* (J2EE).

2.4 Metode Kompresi Data

Compression istilah dalam bahasa Inggris yang diterjemahkan ke bahasa Indonesia yang berarti pemampatan (kompresi). Pada bidang teknik, kompresi berarti proses memampatkan sesuatu yang berukuran besar sehingga menjadi padat/ berukuran lebih kecil (Penerbit Andi. (2003). Memahami Model Enkripsi dan Security Data, hal: 16). Kompresi data, berarti proses untuk memampatkan data agar ukurannya menjadi lebih kecil.

Pemampatan ukuran berkas melalui proses kompresi hanya diperlukan sewaktu berkas tersebut akan disimpan dan/ atau dikirim melalui media transmisi atau telekomunikasi. Berkas tersebut jika ingin ditampilkan pada layar monitor, maka data yang terkompresi tersebut harus dikembalikan ke bentuk sebelumnya agar dapat dibaca kembali. Proses pengembalian ke bentuk semula berkas yang dimampatkan inilah yang disebut dekompresi.

Metode yang dipakai untuk kompresi dan dekompresi pada penulisan ini adalah LZSS (*Lempel Ziv Storer and Szymanski*) merupakan skema yang di inisialisasikan oleh Lempel dan Ziv dan dikembangkan oleh Storer dan Szymanski. Berikut istilah-istilah yang dipakai dalam algoritma LZSS:

a. *Input stream*: urutan karakter yang akan dikompres.

b. *Coding position*: posisi karakter dalam *input stream* yang akan dikodekan (awal *lookahead buffer).*

c. *Lookahead buffer*: urutan karakter dari *coding position* sampai akhir *input stream.*

d. *Window* Ukuran W yang berisi W karakter dari *coding position* kebelakang, yaitu karakter W yang terakhir diproses.

e. *Pointer* menunjuk ke cocok di *window* dan juga menetapkan panjangnya.

2.4.1 Algoritma kompresi LZSS

Berikut langkah-langkah proses kompresi LZSS:

1. Menempatkan *coding position* pada awal *input stream.*
2. Mencari deretan karakter terpanjang pada *lookahead buffer*:
    - P := pointer yang cocok.
    - L := panjang yang cocok.
3. Apakah L >= MIN_LENGTH ?
    - YA: Output P dan pindahkan coding position L karakter kedepan.

- TIDAK: output karakter pertama dari *lookahead buffer* dan pindahkan *coding position* satu karakter kedepan.

4. Jika masih ada karakter pada input stream, maka kembali ke langkah 2, jika tidak ada karakter lagi pada input stream, berarti proses kompresi selesai.

Untuk lebih jelasnya dapat dilihat contoh proses kompresi pada algoritma LZSS. Contoh ini memakai panjang minimum penyalinan *string* dengan dua, yang dapat dilihat pada Tabel 2.1 dan Tabel 2.2.

Tabel 2.1 Input stream untuk kompresi

Posisi	1	2	3	4	5	6	7	8	9	10	11
Karakter	A	A	B	B	C	B	B	A	A	B	C

Tabel 2.2 Proses kompresi algoritma LZSS (MIN_LENGTH=2)

Langkah	Posisi	Cocok	Output
1	1	--	A
2	2	A	A
3	3	--	B
4	4	B	B
5	5	--	C
6	6	B B	(3,2)
7	8	A A B	(7,3)
8	11	C	C

Pada Tabel 2.2 ada empat kolom, berikut penjelasan singkatnya:

1) Kolom Langkah menunjukan langkah proses kompresi.
2) Kolom Posisi menunjukan *coding position*.
3) Kolom Cocok menampilkan deretan karakter pada *window*.
4) Kolom *Output* penulisan ke *output stream*, format pengkodeannya (B,L). Berarti "Kembali B karakter pada *window* dan *copy* L karakter ke *output*".

2.4.2 Algoritma dekompresi LZSS

Proses dekompresi merupakan kebalikan dari proses kompresi. Untuk lebih jelasnya dapat dilihat contoh proses dekompresi pada algoritma LZSS, yang dapat dilihat pada Tabel 2.3 dan Tabel 2.4.

Tabel 2.3 : Input stream untuk dekompresi

Posisi	1	2	3	4	5	6	7	8
Karakter	A	A	B	B	C	(3,2)	(7,3)	C

Tabel 2.4 : Proses dekompresi algoritma LZSS

Langkah	Input	Output
1	A	A
2	A	A
3	B	B
4	B	B
5	C	C
6	(3,2)	B B
7	(7,3)	A A B
8	C	C

2.5 Pemrograman Berbasis Internet

Berikut ini adalah sedikit uraian mengenai teori pemrograman berbasis internet sebatas hanya yang berkaitan dengan pembuatan tugas akhir ini.

2.5.1 HTML (*HyperText Markup Language*)

Hypertext Markup Language (HTML) adalah bahasa standar dalam menulis halaman *web*, HTML merupakan pengembangan dari standar pemformatan dokumen teks. HTML sebenarnya adalah dokumen dalam bentuk ASCII atau teks biasa yang dapat diterjemahkan oleh *web browser* menjadi suatu halaman yang menarik.

Dokumen HTML disebut sebagai *markup-language* karena mengandung tanda-tanda tertentu (yang selanjutnya akan kita sebut sebagai *tag*) yang digunakan untuk menentukan tampilan suatu teks dan tingkat kepentingan dari teks tersebut dalam suatu dokumen.

Tag HTML biasanya adalah *tag-tag* yang berpasangan dan ditandai dengan simbol < dan >, sedangkan 'pasangan' atau akhir perintah dari sebuah *tag* ditandai dengan tanda '/', misalnya pasangan dari tag <tag> adalah </tag>.

Setiap dokumen HTML memiliki struktur sebagai berikut:

```
<html>

<head>
   <TITLE>Judul pada title bar web browser</TITLE>
 </head>
 <body>
        Text, gambar atau isi dokumen HTML anda
 </body>
 </html>
```

Oleh karena sifatnya yang hanya dapat menampilkan informasi secara statis maka untuk membuatnya dinamis, sehingga dapat berinteraksi dengan pembacanya, HTML digunakan menjadi inang sebagai tempat untuk meletakkan bahasa pemrograman web.

2.5.2 Apache Tomcat

Web dinamis membutuhkan sebuah *web application server* atau *appserv* atau biasa disingkat sebagai *web server* saja. Web server memberikan *service* yang bekerja untuk melayani *request* dari *HTTP client* (*web browser*) ke komputer *server*. Salah satu *web server* yang dipakai dalam penulisan adalah *Apache Tomcat JSP Server*. Beberapa keunggulan *Apache Tomcat* antara lain:

a. Ekonomis; merupakan *software* yang berbasis *opensource*, penggunaan pada software ini gratis.

b. *Multi Platform*; dapat digunakan pada semua piranti perangkat lunak dan perangkat keras

2.6 *Arsitektur Java*

Bahasa Java dibuat oleh James Gosling yang dibantu rekan-rekannya yaitu Patrick Naughton, Herbert Schildt, Chris Warth, Ed Frank dan Mike Sheridan di suatu perusahaan bernama Sun Microsystems pada tahun 1991. Bahasa pemrograman ini mula-mula di inisialisasi dengan nama "OAK", namun pada tahun 1995 diganti namanya menjadi "JAVA".

Secara arsitektur, Java tidak berubah sedikit pun semenjak awal mula bahasa tersebut diperkenalkan. Kompiler Java (yang disebut dengan *javac* atau java compiler) akan mentrans-formasikan kode-kode dalam bahasa java ke dalam suatu *bytecode*. Bytecode adalah sekumpulan perintah hasil kompilasi yang kemudian dapat dieksekusi melalui sebuah mesin komputer abstrak, yang disebut JVM (*Java Virtual Machine*).

Ada beberapa versi java yaitu:

a. Java 1 masih disebut JDK (*Java Development Kit*), di mana versi Java yang digunakan adalah Java 1.1

b. Java 2, dimulai dari versi Java 1.2 yang sering disebut JSDK (*Java Software Development Kit)*.

2.6.1 JAVA 2

Sun Microsystem mendefinisikan 3 buah edisi Java 2 yaitu:

a. J2SE (*Java 2 Standard Edition*), digunakan untuk mengembangkan aplikasi desktop dan *applet* (aplikasi yang dapat dijalankan di dalam browser web).

b. J2EE *(Java 2 Enterprise Edition*), merupakan superset dari J2SE yang memungkinkan untuk mengembangkan aplikasi-aplikasi berskala besar (*Enterprise*) yang diperkenalkan pada tahun 1998, yaitu dengan melakukan pembuatan-pembuatan aplikasi-aplikasi di sisi server dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi web dengan menggunakan servlet dan JSP (Java Server Pages) dan teknologi lainnya seperti CORBA (*Common Object Request Broker Architecture*) dan XML (*Extensible Markup Language*). J2EE didesain untuk menjadi suatu bahasa yang ringan, mudah dan *portable* terhadap berbagai *platform.*

c. J2ME (*Java 2 Micro Edition*), merupakan subset dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil. Peralatan mini yang banyak memanfaatkan bahasa ini adalah *remote-control* dan *handphone.*

2.6.2 JSP (*Java Server Pages*)

JSP adalah suatu teknologi web berbasis pemrograman Java dan berjalan di *platform* Java, juga merupkan bagian dari teknologi J2EE. J2EE adalah platform java untuk pengembangan sistem aplikasi *enterprise* dengan dukungan API (*Aplication Programing Interface*). J2EE juga memberikan sarana untuk membuat aplikasi yang memisahkan antara *business logic* (sistem), tampilan muka dan data (*database*).

JSP merupakan bagian dari komponen web dari aplikasi J2EE. JSP juga memerlukan JVM (*Java Virtual Machine*) supaya dapat berjalan. Selain JVM, JSP juga memerlukan server yang disebut *Web Container*. JSP menggunakan pendekatan pemrosesan di sisi *server*. Pada model ini, sumber kode JSP dijalankan pada *web server*. Keuntungan model seperti ini adalah memungkinkan untuk membuat aplikasi yang *independen* (tidak bergantung pada perangkat keras dan perangkat lunak) terhadap keberadaan sistem Java di sisi klien. Berikut alasan yang membuat JSP digunakan untuk pengembangan aplikasi web:

a. JSP menggunakan bahasa java.

b. JSP mendukung multi platform. Keunggulanya adalah memungkinkan kode dapat dipindahkan ke berbagai platform tanpa perlu melakukan perubahan pada kode tersebut.

c. JSP terintegrasi dengan J2EE.

d. JSP memudahkan pembuatan dengan aplikasi tag.

Tujuan dari JSP adalah ialah mempermudah pembuatan dan manajemen halaman web dinamis, dengan memisahkan antara logika bisnis dengan presentasi/ tampilan muka. JSP terdiri dari HTML standard dan tag script JSP.

JSP menggunakan kelas-kelas yang ada di J2EE. Kode JSP pada dasarnya kode HTML yang dilengkapi dengan tag-tag JSP. Berikut contoh pemakaian tag JSP:

```
<html>
        <head>
                <title> tag pada JSP </title>
        </head>
                <body>
                        <%   out.print(" tag-tag pada JSP");   %>
                </body>
</html>
```

Pasangan <% dan %> merupakan salah satu tag untuk model JSP. Akhiran pada berkas yang mengandung kode JSP berakhiran "*.jsp" bukan "*.htm" atau "*.html". Pada sisi klien kode JSP tidak bisa dilihat, karena sifatnya yang demikian, kode JSP dapat disembunyikan terhadap klien yang menggunakannya.

2.6.3 Web Container

Web container ialah sebuah *runtime java* yang menyediakan implementasi dari *java servlet API*, dan fasilitas lainnya untuk JSP. *Web container* bertanggung-jawab untuk inisialisasi, *invoking* dan mengatur daur hidup *servlet* dan JSP. Beberapa tipe kofigurasi *web container* pada java antara lain:

a. *Web Container* yang ada pada *J2EE Application Server*,

pada semua J2EE application server seperti WebLogic, JRun menyertakan web container.

b. *Web Container* yang ada di *web server*

Merupakan *java web server* murni seperti *sun's java web server* yang berisi *web container* yang terintegrasi. *Jakarta Tomcat* yang merupakan *web container reference implementation*, juga masuk dalam kategori ini.

c. *Web Container* pada *runtime* terpisah

di luar J2EE, cara ini merupakan yang paling umum. *Web server* seperti *Apache* membutuhkan *runtime java* terpisah untuk menjalankan *servlet*, dan sebuah *plug-in web server* untuk meng-integrasikan *runtime java* dengan sebuah *server*.