BAB II

TINJAUAN PUSTAKA

II.1 Sistem Keamanan Data

Keamanan data merupakan hal yang sangat penting dalam menjaga kerahasiaan informasi terutama informasi sensitif yang hanya boleh diketahui oleh pihak yang berhak saja. Informasi yang merupakan hasil pengolahan dari data, mempunyai nilai yang berbeda lagi setiap orang. Seringkali sebuah informasi menjadi sangat berharga dan tidak semua orang diperkenankan untuk mengetahuinya, namun selalu saja ada pihak yang berusaha untuk mengetahui informasi dengan cara-cara yang tidak semestinya bahkan bermaksud untuk merusaknya. (sumber : Harry Abdulrachman, Erwin Gunadhi : 2015 : 23)

Keamanan merupakan komponen yang vital dalam komunikasi data elektronis. Masih banyak yang belum menyadari bahwa keamanan (*Securyti*) merupakan sebuah komponen penting yang tidak murah. Teknologi kriptografi sangat berperan juga dalam proses komunikasi yang digunakan untuk melakukan enkripsi (pengacakan) data yang ditransaksikan selama perjalanan dari sumber ke tujuan dan juga melakukan dekripsi (menyusun kembali) data yang telah teracak tersebut. Berbagai sistem yang telah dikembangkan adalah seperti sistem *private key* dan *public key*. Penguasaan algoritma-algoritma populer digunakan untuk mengamankan data juga sangat penting. Contoh-contoh algoritma ini antara lain : DES, IDEA, RC5, RSA, AES dan ECC (*Elliptic Curve Cryptography*). (sumber : Febriansyah : 2012 : 23)

Dari sisi tindakan pihak yang bertanggung jawab, keamanan jaringan komputer terbagi dua level, yaitu :

1. Keamana fisik peralatan mulai dari server, terminal / client router sampai dengan *cabling*.
2. Keamanan sistem data sekitarnya ada penyeludup yang berhasil mendapatkan akses ke seluruh fisik jaringan komputer.

II. 2 Sejara Kriptografi

Kriptografi berasal dari bahasa Yunani yaitu *cryptos* yang artinya "*secret*" (yang tersembunyi) dan gráphein yang artinya "*writting*" (tulisan). Jadi, kriptografi berarti "*secret writting*" (tulisan rahasia). Definisi yang dikemukakan oleh Schneier (1996), Kriptografi adalah ilmu ataupun seni yang mempelajari bagaimana membuat suatu pesan yang dikirim oleh pengirim dapat disampaikan kepada penerima dengan aman. (Emy Setyaningsih,S.Si, M.Kom, 2015 : 8)

II.2.1. Terminologi Kriptografi

Ada beberapa istilah-istilah yang penting dalam kriptografi, yaitu :

1. Pesan (*Plaintext dan Ciphertext*) : Pesan (*message*) adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Pesan asli disebut plainteks (*plaintext*) anda teks-jelas (*cleartext*). Sedangkan pesan yang sudah disandikan disebut cipherteks (*chipertext*).
2. Pengirim dan Penerima : Komunikasi data melibatkan pertukaran pesan antaradua entitas. Pengirim (*sender*) adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima (*receiver*) adalah entitas yang menerima pesan.
3. Penyadap (*eavesdropper*) adalah orang yang mencoba menangkap pesan selama

ditransmisikan.

4. Kriptanalisis dan Kriptologi : Kriptanalisis (*cryptanalysis*) adalah suatu ilmu dan seni membuka (*breaking*) *ciphertext* menjadi *plaintext* tanpa mengetahui kunci yang digunakan. Pelakunya disebut kriptanalis . kriptologi (*cryptology*) adalah studi mengenai kriptografi dan kriptanalisi.
5. Enkripsi dan Dekripsi : Proses untuk menyandikan plaintext menjadi ciphertext disebut enkripsi (*encryption*). Sedangkan proses untuk memperoleh kembali plaintext dari ciphertext disebut dekripsi (*decryption*).
6. Chiper dan Kunci : algoritma matematis untuk menyadikan plaintext menjadi ciphertext. Kunci (*key*) adalah parameter yang digunakan untuk transformasi *encriphering* dan *dechipering*. (Emy Setyaningsih, S.Si, M.Kom : 2015 : 8)

II.3 Algoritma Kriptografi

Algoritma-algoritma kriptografi dapat dibedakan menjadi dua macam yaitu simetrik dan asimetrik. Algoritama yang digunakan satu kunci untuk proses enkripsi dan dekripsi data. Sedangkan algoritma asimentrik (model enkripsi kunci public) menggunakan kunci yang berbeda dalam proses enkripsi dan dekrispi pesan. (Sumber : I Gede Andika Putra : 2012 : 30).

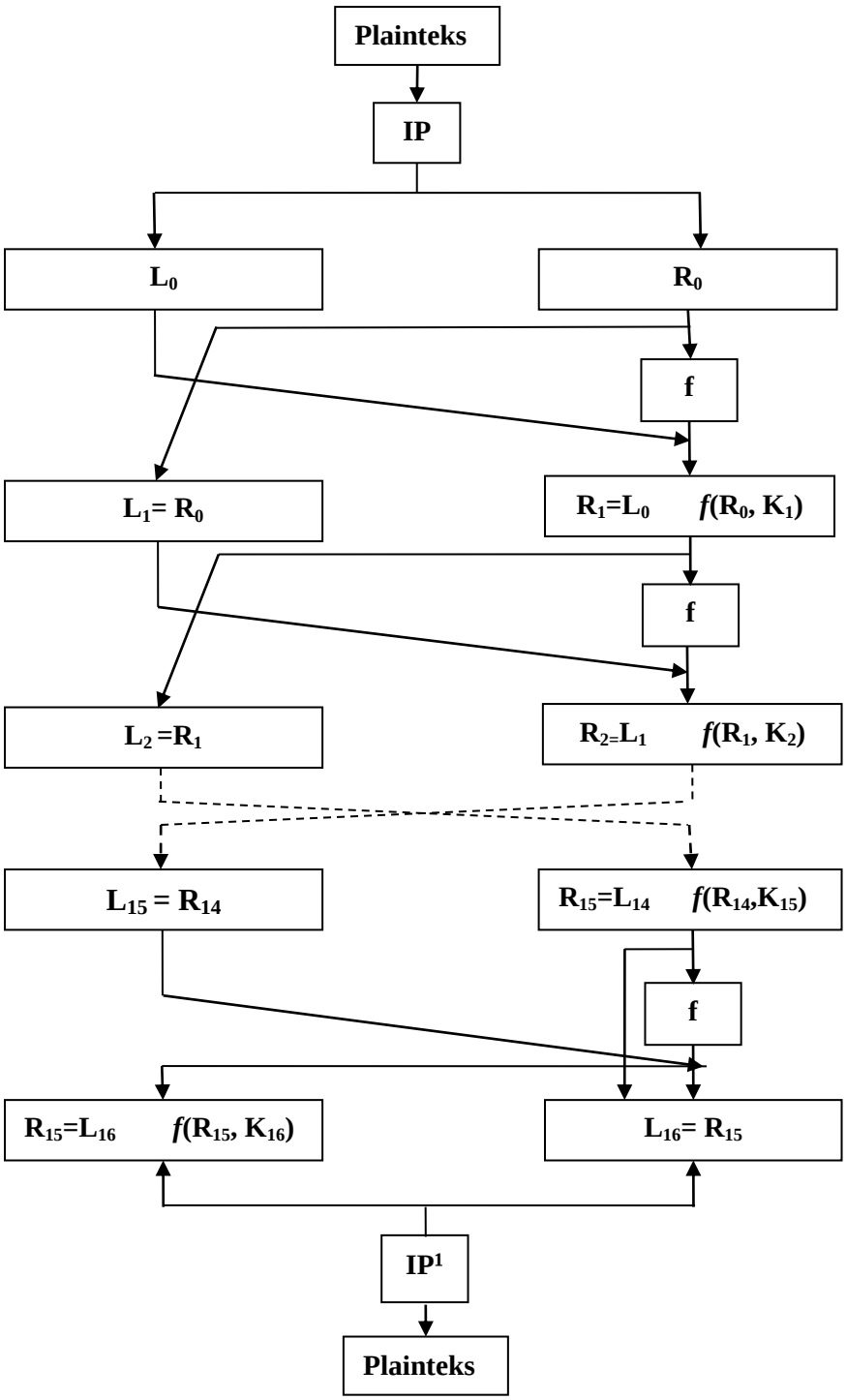
II.4 Algoritma DES (*Data Encryption Standard*)

Algoritma DES merupakan algoritma enkripsi yang paling banyak digunakan di dunia yang diadopsi oleh NIST (*National Institue of Standards and Technology*) sebagai standar pengolah informasi Federal AS. Sejara DES dimulai dari pemerintah pemerintah amerika serikat untuk memasukkan proposal enkripsi. Algoritma DES dibuat di IBM dan

merupakan modifikasi dari algoritma terdahulu yang bernama Lucifer. Lucifer merupakan algoritma *cipher blok* yang beroperasi pada blok masukan 64 bit dan kuncinya berukuran 18 bit. Pengurangan jumlah bit kunci pada DES dilakukan dengan alasan agar mekanisme algoritma ini bias diimplementasikan dalam satu *chip*. Hosrt Feistel merupakan salah satu periset yang mula-mula mengembangkan DES ketika bekerja di IBM Watson Laboratory di Yorktown Heights, New York.

Algoritma DES terbagi menjadi tiga kelompok, yaitu pemrosesan kunci, enkripsi data 64 bit, dan dekripsi data 64 bit, dimana kelompok yang satu dengan yang lain saling berinteraksi dan terkait. Algoritma DES dirancang untuk mengenkripsi dan mendekripsi data dalam blok data yang terdiri atas 64 bit dibawah kontrol kunci 64 bit. Dekripsi data harus dikerjakan menggunakan kunci yang sama dengan yang dipakai untuk mengenkripsi data, dengan penjadwalan alamat kunci bit yang diubah sehingga proses membaca merupakan kebalikan dari proses menulis.

Dekripsi mengekripsikan 64 bit plaintext menjadi 64 bit ciphertext dengan menggunakan 56 bit kunci internal yang dibangkitkan dari 64 bit kunci eksternal. Kunci ekternal merupakan kunci yang dimasukkan oleh pengguna pada sistem, sedangkan kunci internal merupakan kunci yang digunakan untuk melakukan enkripsi pada setiap putaran DES (ada 16 putaran) yang diperoleh dari kunci eksternal yang telah diproses. (Emy Setyaningsih, S.Si, M.Kom : 2015 :)



Gambar II.1. Skema algoritma DES

Skema i berikut :

1. *Blok plaintext* dipemutasi dengan matriks permutasi awal (*initial permutation atau IP*). Plaintext merupakan data yang akan dienkripsi. Plaintext ini direpresentasikan sebagai bit,

misalnya huruf P direpresentasikan sebagai 01010000. Keseluruhan plaintext dibagi ke dalam blok-blok. Setiap blok terdiri atas 64 bit.

2. Dalam proses enchiperblok plainteks terbagi menjadi dua bagian yaitu bagian kiri (L) dan bagian kanan (R), yang masing masing memiliki panjang 32 bit. Pada setiap putaran i blok R merupakan masukan untuk fungsi transformasi fungsi f. Pada fungsi f blok R dikombinasikan dengan kunci internal Ki. Keluaran dari fungsi ini di XOR kan dengan blok L yang langsung diambil dari blok R sebelumnya. Ini merupakan 1 putaran DES. Secara matematis, satu putaran DES dinyatakan sebagai berikut :

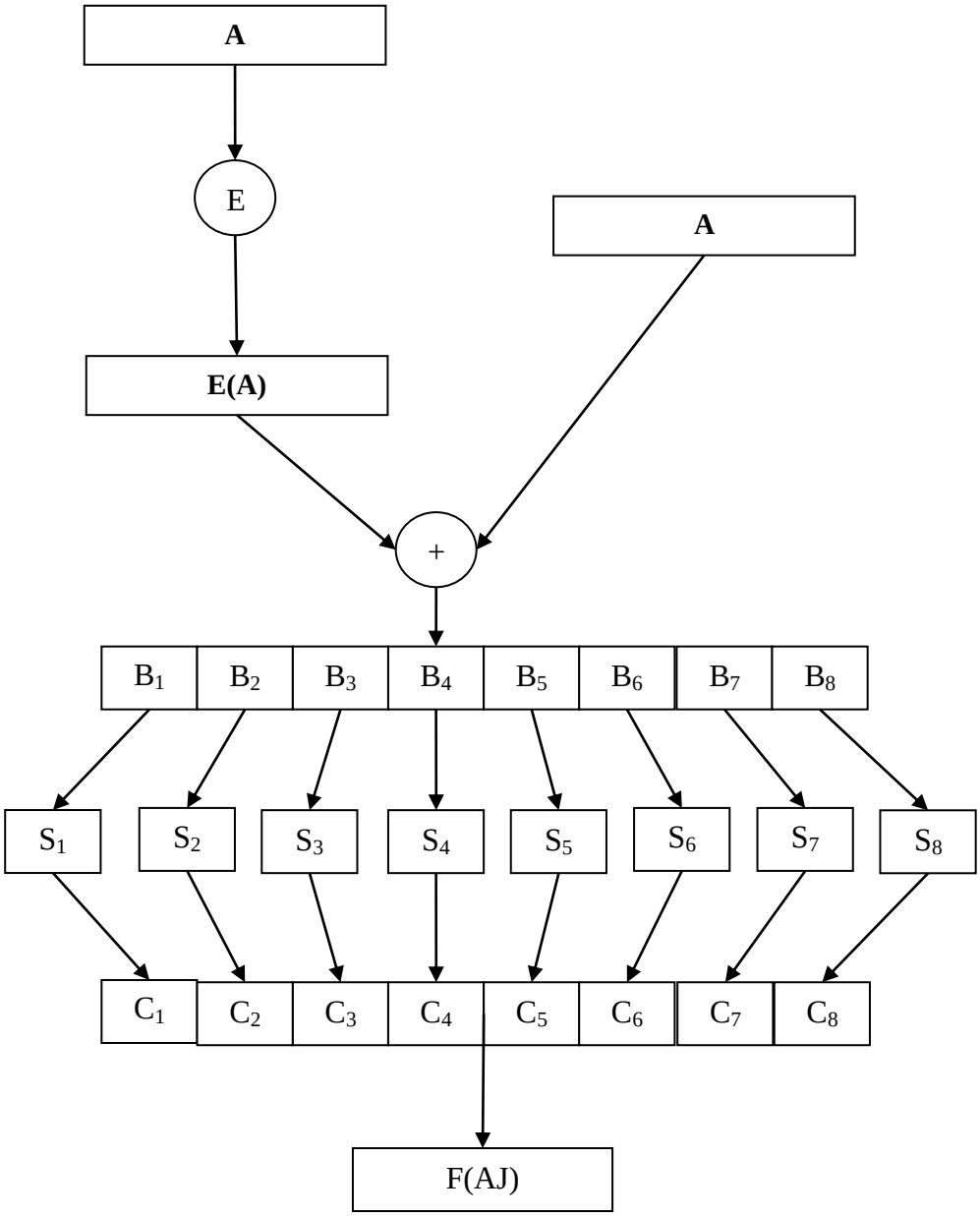
$$\mathbf{Li = Ri - 1}$$
$$\mathbf{Ri = Li - 1 + \mathit{f}(Ri - 1, Ki)}$$

3. Jika (L16, R16) merupakan keluaran dari putaran ke-16,maka (R16, L16) merupakan pra-cipherteks (preciphertext) dari encipheringini.Cipherteks yang sebenarnya diperoleh dengan melakukan permutasi awal balikan, IP-1, terhadap blok pra-cipherteks.
4. Hasil enkripsi kemudian dipermutasikan dengan matriks permutasi balikan (*inverse initial permutation atau* IP^{-1}) menjadi *blok ciphertext.*

   Fungsi *f* diambil dari masukan pertama argument A, dengan panjnag bitstring 32, dan argument yang kedua j dari panjang bitstring 48 dan prosedur keluaran dari bitstring adalah 32 seperti langkah berikut :

   1. Argumen pertama A di perluas ke bitstring dengan panjang 48 menurut fix dari fungsi ekspansi E. E(A) yang berisi 32 bit dari A, permutasi dengan cara tertentu dengan 16 bit, sedikitnya muncul dua kali.
   2. Perhitungan E(A) *j* dan hasilnya ditulis pada penggabungan 6- bitstring *B* =

$B_1B_2B_3B_4B_5B_6B_7B_8$.



Gambar

II.2. Rincian DES Fungsi *f*

(Sumber : Dony Arius;2008:113)

3. Langkah berikutnya menggunakan beberapa letak- S $S_1, \ldots, S_8$. Setiap S_1 adalah 4 x 6 larik yang dimasukkan dari integer 0 – 15. Dengan member suatu bitstring dengan panjang 6, seperti $B_j = b_1b_2b_3b_4b_5b_6$, maka akan dapat dihitung $S_j(B_j)$ dengan mengikuti dua bit b_1b_6, tentunya dengan representasi binary row r of S_j ($0 \leq c \geq 15$). Kemudian $S_j(B_j)$ di defenisikan sebagai entry S_j(r, c) yang dituliskan ke dalam b inari dengan panjang bitstring empat. Dengan begitu dapat dihitung $C_j = S_j(B_j)$, $1 \leq j \leq 8$.

4. Bitstring $C = C_1C_2C_3C_4C_5C_6C_7C_8$ dengan panjang 32 adalah permutasi dari hasil bitstring P(C) yang didefenisikan untuk $f(A,j)$.

   Fungsi f di jelaskan pada gambar diatas. Pada dasarnya berisi subsitusi dengan menggunakan S-box (kotak-S) dengfan mengikuti permutrasi P. 16 iterasi dari f yang terdiri dari system kripto.

   Data enkrip dalam blok-blok 64 bit menggunakan kunci 56 bit. DES mentransformasikan masukan 64n bit dalam beberapa tahap enkripsi ke dalam keluaran 64 bit. Dengan demikian DES termasuk l ama blok kode dengan tahapan pemakaian kunci yang sama, uintuk dekripsinya.

II.5 Algoritma AES (*Anvanced Encryption Standard*)

Pada bulan Januari 1997 inisiatif AES diumumkan dan pada bulan September 1997 publik diundang untuk mengajukan proposal *block cipher* yang cocok sebagai kandidat untuk AES. Pada tahun 1999 NIST mengumumkan lima kandidat finalis yaitu MARS, RC6, Rijndael, Serpent, dan Twofish. Algoritma AES dipilih pada bulan Oktober 2001 dan standarnya diperkenalkan pada bulan November 2002. Algoritma AES (*Advanced Encryption Standard*) adalah algoritma *cipher* blok yang menggunakan teknik subtitusi, permutasi dari sejumlah

putaran pada setiap blok yang akan dienkripsi. Sistem permutasi dan subtitusi (*S-box)* yang diguanakan pada AES tidak menggunakan jaringan *Feistel* sebagaimana *cipher blok* pada umumnya.(Emy Setyaningsih, S.Si, M.Kom : 2015 : 168)

Algoritma Rijndael dirancang untuk memiliki properti berikut :

1. Ketahanan terhadap semua jenis serangan yang diketahui.
2. Kesederhanaan rancangan.
3. Kekompakan kode dan kecepatan pada berbagai *platform*.

AES terbagi dalam tiga jenis yaitu :

1. AES-128
2. AES-192
3. AES-256

Tabel : II.1 Perbandingan jumlah round dan kunci

	Jumlah Kunci (Nk)	Besar Blok (Nb)	Jumlah Round (Nr)
AES-128	16 *byte*	16 *byte*	10
AES-129	24 *byte*	16 *byte*	12
AES-256	32 *byte*	16 *byte*	14

(Wibowo dan Wihartantyo, 2004)

II.6 *Unified Modeling Language* (UML)

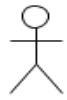
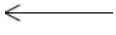
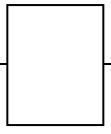
UML yang merupakan singkatan dari *Unified Modelling Language* adalah sekumpulan pemodelan konvensi yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak dalam kaitannya dengan objek. UML dapat juga diartikan sebuah bahasa grafik standar yang digunakan untuk memodelkan perangkat lunak berbasis objek. UML pertama kali

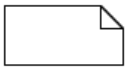
dikembangkan pada pertengahaan tahun 1990an dengan kerjasama antara James Rumbaugh, Grady Booch dan Ivar Jacobson, yang masing-masing telah mengembangkan notasi mereka sendiri di awal tahun 1990an. (Lethbride dan Leganiere, 2009:11)

II.6.1. *Use Case Diagram*

Use case diagram, adalah sebuah gambaran dari fungsi sistem yang dipandang dari sudut pandang pemakai. *Actor* adalah segala sesuatu yang perlu berinteraksi dengan sistem untuk pertukaran informasi. *System boundary* menunjukkan cakupan dari sistem yang dibuat dan fungsi dari sistem tersebut. (Lethbride dan Leganiere, 2009:11)

Tabel II.2. Simbol *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

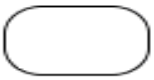
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

(Sumber :Lethbride dan Leganiere, 2009:11)

II.6.2. *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. Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada *business modeling* untuk memperlihatkan urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowchart* atau Data *Flow* Diagram pada perancangan terstruktur. Activity diagram dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram. (Lethbride dan Leganiere, 2009:13)

Tabel II.3. Simbol *Activity Diagram*

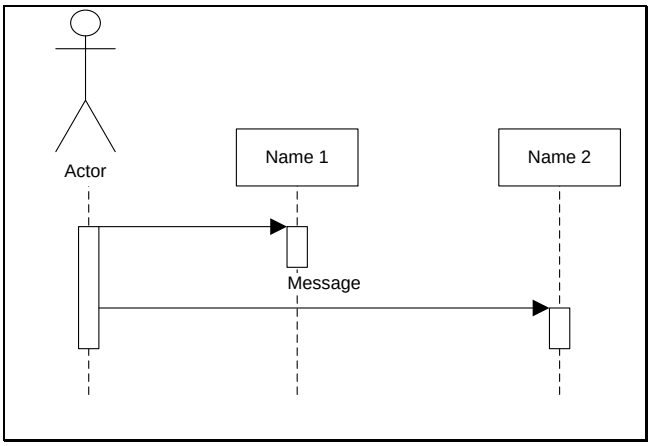
NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

(Sumber :Lethbride dan Leganiere, 2009:15)

II.6.3. *Sequence diagram*

Sequence diagram menambahkan dimensi waktu pada interaksi diantara obyek. Pada diagram ini participant diletakkan di atas dan waktu ditunjukkan dari atas ke bawah. *Life line participant* diurutkan dari setiap paricipant. Kotak kecil pada life line menyatakan *activation* : yaitu menjalankan salah satu operation dari participant. Satate bisa ditambahkan dengan menempatkannya sepanjang life line. *Message* (sederhana, synchronous atau *asynchroneous*) adalah tanda panah yang menghubungkan suatu *life line* ke *life line* yang lain. Lokasi *life line* dalam dimensi vertikal mewakili urutan waktu dalam sequence diagram. Message yang pertama terjadi adalah yang paling dekat dengan bagian atas diagram dan yang terjadi belakangan adalah

yang dekat dengan bagian bawah. Pada beberapa sistem, operasi bisa dilakukan kepada dirinya sendiri. Hal ini disebut dengan rekursif. Untuk melukiskannya digunakan anak panah dar activation kembali ke dirinya sendiri, dan sebuah kotak kecil diletakkan pada bagian atas dari *activation*.



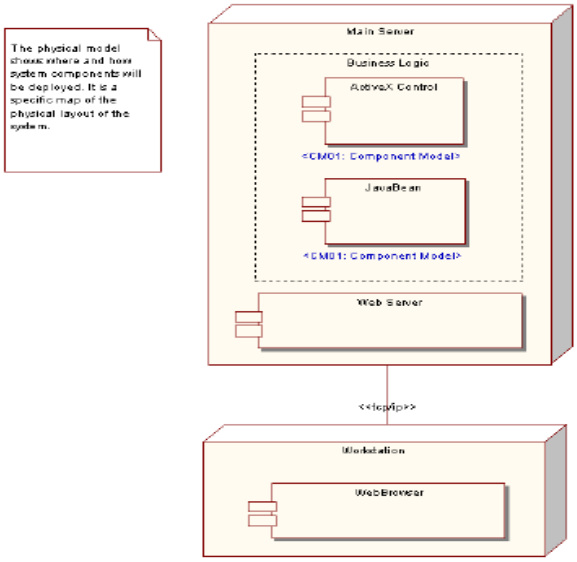
Gambar II.3. Simbol-simbol yang ada pada *sequence diagram*

(Sumber :Agus Putranto, 2009:14)

II.6.4. *Deployment diagram*

Deployment diagram menggambarkan sumber fisik dalam sistem, termasuk node, komponen dan koneksi (model implementasi sistem yang statistik). Dalam hal ini meliputi topologi *hardware* yang dipakai sistem.

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.



Gambar II.4. Simbol-simbol yang ada pada *Deployment* Diagram

(Sumber :Agus Putranto:2009:9)

II.7 Eclipse

Eclipse adalah sebuah IDE(*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent).* Berikut ini adalah sifat dari Eclipse :

1. *Multi-platform*: Target sistem operasi Eclipse adalah *Microsoft Windows, Linux,Solaris, AIX, HP-UXdan Mac OS X.*
2. *Mulit-language:* Eclipse dikembangkan dengan bahasa pemrograman *Java*, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lain seperti *C/C++, Cobol, Python, Perl, PHP*, dan lain sebagainya.
3. *Multi-role:* Selain sebagai IDE untuk pengembangan aplikasi. Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak seperti dokumentasi, pengujian

perangkat lunak, pengembangan web, dan lain sebagainya.

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah dipasang (*diinstal*). Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform*(RCP). Berikut ini adalah komponen yang membentuk RCP:

- *Core platfor*
- *OSGi*
- SWT(*Standard Widget Toolkit*)
- J*Face*
- *Eclipse Workbench*

Eclipse selalu dilengkapi dengan JDT(*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program *Java*, dan PDE(*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. (Sumber : Wina Noviani Fatimah, ST : 2011 : 2)

II.8 Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya *Google inc* membeli *android inc* yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan *android*, dibentuklah *Open Handesti Alliance*, konsorium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.* Pada saat perilisan perdana *android*, 5 November 2007, *Android* bersama *Open Handest Alliance* menyatakan mendukung pengembangan *open*

source pada perangkat *mobile*. Di lain pihak, *Google* merilis kode-kode *android* dibawah lisensi *Apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler. (Safaat H, Nazruddin, 2012)

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi *Android*, sebuah file yang ditandai dengan akhiran *.apk*. file inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*.(Sumber : Ahmad, 2015 :191)