BAB II

TINJAUAN PUSTAKA

II.1 Enkripsi

II.1.1 Pengertian Enkripsi

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 *chiper*. Sebuah sistem pengkodean menggunakan suatu table atau kamus yang telah didefinisikan untuk menggantikan kata dari informasi atau yang merupakan bagian dari informasi yang dikirim. (Dian Wirdasari; 2008: 174)

II.2. Citra Digital

II.2.1. Defenisi Citra

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari sebuah objek. Citra sebagai keluaran suatu sistem perekamam data dapat bersifat optik berupa foto, bersifat *analog* berupa sinyal-sinyal *video* seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpan. (T.Sutoyo, dkk ; 2009 : 9).

II.2.1.1. Defenisi Citra *Analog*

Citra *analog* adalah citra yang bersifat kontinu, seperti gambar pada minitor televisi, foto sinar x, foto yang tecetak dikertas foto, lukisan,

pemandangan alam, hasil *CT scan*, gambar-gambar yang terekam pada pita kaset, dan lain sebagainya. Citra *analog* tidak dapat direpresentasikan dalam komputer sehingga tidak bisa diproses dikomputer secara langsung. Oleh sebab itu, agar citra dapat diproses dikomputer, proses konversi *analog* ke digital harus dilakukan terlebih dahulu. (T.Sutoyo, dkk ; 2009 : 9).

II.2.1.2. Defenisi Citra Digital

Citra digital adalah citra yang dapat diolah oleh komputer. Yang menghasilkan keluaran sebuah sistem perekaman data. (T. Sutoyo,dkk; 2009: 9). Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut *pixel* (*pixel* = picture element), yaitu elemen terkecil dari sebuah citra. pixel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari *pixel* di titik itu.Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks berikut.

$$f(x,y) = \begin{vmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & \dots & \dots & f(1,M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{vmatrix}$$

Sumber : (T. Sutoyo, dkk; 2009 : 20).

Berdasarkan matriks tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f (x,y), dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y)

yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari *pixel* di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M dan kolom N hingga citra membentuk matriks ***M x N*** dan jumlah tingkat keabuan *pixel* G (T. Sutoyo, dkk; 2009 : 20).

Pengolahan citra digital adalah sebuah disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra), transformasi gambar (rotasi, translasi, skala, transformasi geometrik), melakukan pemilihan citra ciri (*feature images)* yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data. Input dari pengoalahan citra adalah citra, sedangkan outputnya adalah citra hasil pengolahan (T. Sutoyo, dkk; 2009: 5).

II.2.2. Format *File* Citra

Ada dua jenis format *file* citra yang sering digunakan dalam pengolahan citra, yaitu citra *bitmap* dan citra vektor. (T.Sutoyo, dkk ; 2009 : 25).

II.2.2.1. Format *File* Citra *Bitmap*

Citra *bitmap* sering disebut juga dengan citra raster. Citra *bitmap* menyimpan data kode citra secara digital dan lengkap (cara menyimpannya adalah per *pixel*). Citra *bitmap* dipresentasikan dalam bentuk matriks atau dipetakan dengan menggunakan bilangan biner atau sistem bilangan lain. Citra ini memiliki

kelebihan untuk memanipulasi warna, tetapi untuk mengubah objek lebih sulit. Tampilan *bitmap* mampu menunjukkan kehalusan gradiasi bayangan dan warna dari sebuah gambar. Oleh karena itu, *bitmap* merupakan media elektronik yang paling tepat untuk gambar-gambar dengan perpaduan gradiasi warna yang rumit, seperti foto dan lukisan digital. Citra *bitmap* biasanya diperoleh dengan cara *scanner*, *camera* digital, *video*, *capture*, dan lain-lain. (T.Sutoyo, dkk ; 2009 : 25).

II.2.2.2. Format *File* Citra Vektor

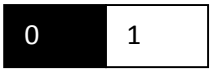
Citra vektor dihasilkan dari perhitungan matematis dan tidak berdasarkan *pixel*, yaitu data tersimpan dalam bentuk vektor posisi, dimana yang tersimpan hanya informasi vektor posisi dengan bentuk sebuah fungsi. Pada citra vektor, mengubah warna lebih sulit dilakukan, tetapi membentuk objek dengan cara mengubah nilai lebih mudah. Oleh karena itu, bila citra diperbesar atau diperkecil, kualitas citra relatif tetap baik dan tidak berubah. Citra vektor biasanya dibuat menggunakan aplikasi-aplikasi citra vektor, seperti *corelDraw, adobe Illustrator, macromedia Freehand, Autocad*, dan lain-lain. (T. Sutoyo, dkk; 2009 : 27).

II.2.3. Jenis-jenis Citra Digital

Beberapa jenis citra digital yang sering digunakan adalah citra biner, citra *grayscale*, dan citra warna. (T. Sutoyo, dkk; 2009 : 21).

II.2.3.1. Citra Biner (Monokrom)

Banyaknya warna 2, yaitu hitam dan putih. Dibutuhkan 1 bit di memori untuk menyimpan kedua warna ini. (T. Sutoyo, dkk; 2009 : 21).



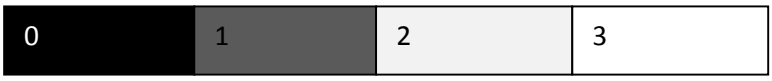
bit 0 = warna hitam

bit 1 = warna putih

II.2.3.2. Citra *grayscale* (skala keabuan)

Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna ini. (T. Sutoyo, dkk; 2009 : 21).

Citra 2 bit mewakili 4 warna dengan gradiasi warna berikut :



Citra 3 bit mewakili 8 warna dengan gradiasi warna berikut :

Semakin besar jumlah bit warna yang disediakan di memori, semakin halus gradiasi warna yang terbentuk.

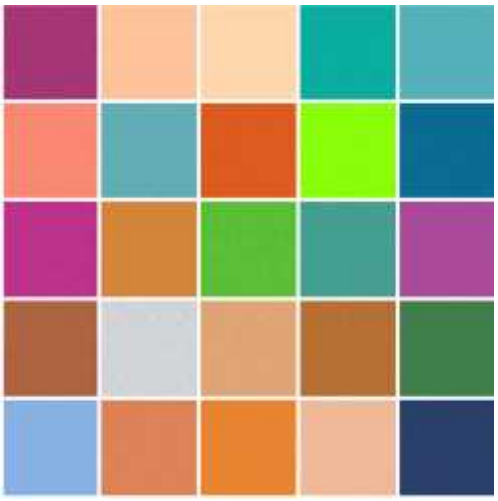
II.2.3.3. Citra Warna (*True color*)

Setiap *pixel* pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (*RGB = Red Green Blue*). Setiap warna dasar menggunakan penyimpanan 8 bit = 1 byte, yang berarti setiap warna mempunyai gradiasi sebanyak 255 warna. Berarti setiap *pixel* mempunyai kombinasi warna sebanyak $2^8.2^8.2^8 = 2^{24}$ = 16 juta warna lebih. Itulah sebabnya format ini dinamakan *true*

color karena mempunyai jumlah warna yang cukup besar sehingga bisa dikatakan hampir mencakup semua warna di alam. (T. Sutoyo, dkk; 2009 : 22).

Penyimpanan citra *true color* didalam memori berbeda dengan citra *grayscale*. Setiap *pixel* dari citra *grayscale* 256 gradiasi warna diwakili oleh 1 byte. Sedangkan 1 *pixel* citra *true color* diwakili oleh 3 byte, dimana masing-masing byte merepresentasikan warna merah (*Red*), hijau (*Green*), dan biru (*Blue*). Seperti ditunjukkan pada gambar II.1. dibawah ini :

Citra warna



Penyimpanan dalam memori

R=50 G=65	R=40 G=45	R=50 G=95	R=80 G=85	R=70 G=75
R=80 G=40	R=50 G=35	R=50 G=90	R=20 G=65	R=50 G=65
R=80 G=45	R=20 G=65	R=50 G=95	R=50 G=35	R=70 G=55
R=60 G=55	R=30 G=60	R=30 G=45	R=50 G=60	R=40 G=95
R=40 G=45	R=90 G=55	R=30 G=65	R=50 G=55	R=50 G=65

Gambar II.1. Contoh Penyimpanan Citra Warna Dalam Memori
Sumber (T. Sutoyo, dkk; 2009 : 23).

II.2.4. Elemen-Elemen Citra Digital

Berikut adalah elemen-elemen yang terdapat pada citra digital : (T. Sutoyo, dkk; 2009 : 24).

1. Kecerahan (*Brightness*)

   Kecerahan (*Brightness*) merupakan intensitas cahaya yang dipancarkan *pixel* dari citra yang dapat ditangkap oleh sistem penglihatan. Kecerahan pada sebuah titik (*pixel*) di dalam citra merupakan intensitas rata-rata dari suatu area yang melingkupi.

2. Kontras (*Contrast*)

   Kontras (*Contrast*) menyatakan sebaran terang dan gelap dalam sebuah citra. Pada citra yang baik, komposisi gelap dan terang secara merata.

3. Kontur (*Contour*)

   Kontur (*Contour*) adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel-pixel* yang bertetangga. Karena adanya perubahan intensitas inilah mata mampu mendeteksi tepi-tepi objek di dalam citra.

4. Warna

   Warna sebagai persepsi yang ditangkap sistem visual terhadap panjang gelombang cahaya yang dipantulkan oleh objek.

5. Bentuk (*Shape*)

   Bentuk (*Shape*) adalah properti intrinsik dari objek 3 dimensi, dengan pengertian bahwa bentuk merupakan properti intrinsik utama untuk sistem visual manusia.

6. Tekstur (*Texture*)

   Tekstur (*Texture*) dicirikan sebagai distribusi spasial dari derajat keabuan di dalam sekumpulan *pixel-pixel* yang bertetangga. Tekstur adalah sifat-sifat atau karakteristik yang dimiliki oleh suatu daerah tersebut. Tekstur adalah

keteraturan pola-pola tertentu yang terbentuk dari susunan *pixel-pixel* dalam citra digital. (T. Sutoyo, dkk; 2009: 24).

II.3. Algoritma

II.3.1. Algoritma Kriptografi

Algoritma ditinjau dari asal usul kata, kata algoritma mempunyai sejarah yang menarik, kata ini muncul didalam kamus *Webster* sampai akhir tahun 1957 hanya menemukan kata *algorism* yang mempunyai arti proses perhitungan dengan bahasa Arab. Algoritma berasal dari nama penulis buku Arab yang terkenal yaitu Abu Ja'far Muhammad ibnu Musa al-khuwarizmi (al-Khuwarizmi dibaca oleh orang barat menjadi *algorism*). Kata *algorism* lambat laun berubah menjadi *algorithm*. (Dony ariyus; 2006: 13).

Defenisi terminologinya algoritma adalah urutan langkah-langkah logis untuk penyelesaian masalah yang disusun secara sistematis. Algoritma kriptografi merupakan langkah- langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut (Dony ariyus; 2006: 13). Algoritma kriptografi terdiri dari tiga fungsi dasar yaitu :

1. Algoritma Enkripsi : Enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan terjaga kerahasiannya. Pesan asli disebut plainteks yang dirubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan *cipher* atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata, maka kita akan melihatnya didalam kamus atau daftar istilah-istilah. Beda halnya dengan

enkripsi, untuk merubah plainteks ke bentuk cipherteks kita menggunakan algoritma yang dapat mengkodekan data yang kita inginkan.

2. Algoritma Dekripsi : dekripsi merupakan kebalikan dari enkripsi, pesan yang telah dienkripsi dikembalikan kebentuk asalnya (Plainteks) disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan yang digunakan untuk enkripsi.
3. Kunci : kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi, kunci terbagi jadi dua bagian kunci pribadi (*private key*) dan kunci umum (*public key*).

II.3.2. Tujuan Kriptografi

Dari paparan awal dapat dirangkumkan bahwa kriptografi bertujuan untuk memberi layanan keamanan. Yang dinamakan aspek-aspek keamanan sebagai berikut : (Dony Ariyus; 2006: 8).

1. *Authentication*

Adalah layanan yang ditujukan untuk menjaga agar penerima informasi dapat memastikan keasslian pesan, bahwa pesan itu datang dar orang yang dimintai informasi. Dengan kata lain, informasi itu benar-benar datang dari orang yang dikehendaki.

2. *Integrity*

Keaslian pesan yang dikirim melalui jaringan dan dapat dipastikan bahwa informasi yang dikirim tidak dimodifikasi oleh orang yang tidak berhak dalam perjalanan informasi tersebut.

3. *Non-repudiation*

Merupakan hal yang berhubungan dengan si pengirim. Pengirim tidak dapat mengelak bahwa dialah yang mengirim informasi tersebut.

4. Authority

Adalah layanan agar informasi yang berada pada sistem jaringan tidak dapat dimodifikasi oleh pihak yang tidak berhak untuk mengaksesnya.

5. Confidentiality

Adalah layanan yang merupakan usaha untuk menjaga informasi dari orang yang tidak berhak mengakses. Kerahasiaan ini biasanya berhubungan dengan informasi yang diberikan ke pihak lain.

6. Privacy

Adalah layanan yang lebih ke arah data-data yang bersifat pribadi.

7. *Availability*

Adalah layanan yang merupakan aspek availabilitas berhubungan dengan ketersedianan informasi ketika dibutuhkan. Sistem informasi yang diserang atau dijebol dapat menghambat atau meniadakan akses ke informasi.

8. *Access Control*

Adalah layanan merupakan aspek yang berhubungan dengan cara pengaturan akses ke informasi. Hal ini biasanya berhubungan dengan masalah otentifikasi dan privasi. Kontrol akses seringkali dilakukan dengan menggunakan kombinasi *user id* dan *password* ataupun dengan mekanisme lain.

II.3.3. Macam-macam Algoritma kriptografi

II.3.3.1. Algoritma Simetri

Algoritma ini juga sering disebut dengan algoritma klasik, karena memakai kunci yang sama untuk melakukan enkripsi dan dekripsinya. Algoritma ini sudah ada lebih dari 4000 tahun yang lalu. Mengirim pesan dengan menggunakan algoritma ini, sipenerima pesan harus diberitahu kunci dari pesan tersebut agar dapat mendekripsi pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci, jika kunci tersebut diketahui oleh orang lain maka, orang tersebut dapat melakukan enkripsi dan dekripsi terhadap pesan tersebut. Sandi *vigenere* adalah sandi yang menggunakan algoritma simetri. Karena sandi *vigenere* meggunakan kata kunci yang sama untuk kunci *private* dan kunci publik nya. (Dony Ariyus; 2006: 14).

II.3.3.2. Algoritma Asimetri

Algoritma asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsinya berbeda. Pada algoritma asimetri kunci dibagi menjadi dua bagian :

a. Kunci umun (*public key*) adalah kunci yang boleh semua orang tahu (dipublikasikan)
b. Kunci pribadi (*private key*) adalah kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

Kunci-kunci tersebut saling berhubungan satu dengan yang lainnya. Dengan kunci publik orang dapat mengenkripsi pesan tapi tidak bisa mendekripsiknya, hanya orang yang memiliki kunci pribadi yang dapat

mendekripsi pesan tersebut. Algoritma asimetri dapat melakukan pengiriman pesan yang lebih aman dari pada algoritma simetri. (Dony Ariyus; 2006: 15).

II.3.3.3. *Hash* Function

Fungsi *hash* sering disebut dengan fungsi *hash* satu arah (*one-way function*), *message digest*, *fingerprint*, fungsi kompresi dan *message authentication code (MAC)*, hal ini merupakan suatu fungsi matematika yang mengambil input panjang variabel dan mengubahnya kedalam urutan biner dengan panjang yang tetap. Fungsi *hash* biasanya diperlukan bila ingin membuat sidik jari (*digital signature*) dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda yang menandakan bahwa pesan tersebut benar-benar dari orang yang diinginkan. (Dony Ariyus; 2006: 15).

II.3.3.4 Algoritma Kriptografi Klasik

Kriptografi klasik merupakan suatu algoritma yang menggunakan satu kunci untuk mengamankan data, teknik ini sudah digunakan beberapa abad yang lalu. Pada dasarnya, algoritma kriptografi klasik dapat dikelompokkan ke dalam dua macam *cipher*, yaitu : (Dony Ariyus; 2006: 16).

1. *Cipher* substitusi (*substitution cipher*)

Di dalam *cipher* substitusi setiap unit plainteks diganti dengan satu unit *cipherteks*. Satu “unit” di isini berarti satu huruf, pasanga huruf, atau dikelompokkan lebih dari dua huruf. Algoritma substitusi tertua yang diketahui adalah *Caesar cipher* yang digunakan oleh kaisar *romawi* , Julius Caesar (sehingga dinamakan juga *caesar cipher*), untuk mengirimakan pesan yang dikirimkan kepada gubernurnya.

2. *Cipher* transposisi (*transposition cipher*)

Pada *cipher* transposisi, huruf-huruf di dalam plainteks tetap saja, hanya saja urutannya diubah. Dengan kata lain algoritma ini melakukan transpose terhadap rangkaian karakter di dalam teks. Nama lain untuk metode ini adalah permutasi atau pengacakan (*scrambling*) karena transpose setiap karakter di dalam teks sama dengan mempermutasikan karakter-karkater tersebut.

II.4. Tranposisi

II.4.1. Pengertian Transposisi

Cipher transposisi adalah suatu algoritma enkripsi yang melakukan enkripsi dengan mengubah urutan dari plainteks. Transposisi adalah pada dasarnya membuat *ciphertext* dengan menggantikan posisi objek-objek *plaintext* tanpa menggantikan objek *plaintext* tersebut, jadi pada proses transposisi tidak diperlukan karakter lain. (Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:90)

II.4.2. Enkripsi Teknik Substitusi

Substitusi adalah penggantian setiap karakter *plaintext* dengan karakter lain. Dengan kata lain teknik substitusi adalah salah satu teknik enkripsi simetris yang mana dilakukan penggantian setiap objek *plaintext* dengan objek lain, teknik ini menerapkan konsep korenspondensi satu-satu untuk tiap objek *plaintext* yang disandikan. Objek yang akan disubstitusikan dalam pembuatan skripsi ini adalah *pixel.* Adapun langkah-langkahnya dapat diilustrasikan sebagai berikut:

1. Nilai *pixel-pixel* dari gambar dimasukkan kedalam sebuah matrik dengan ordo sama dengan ukuran gambar.

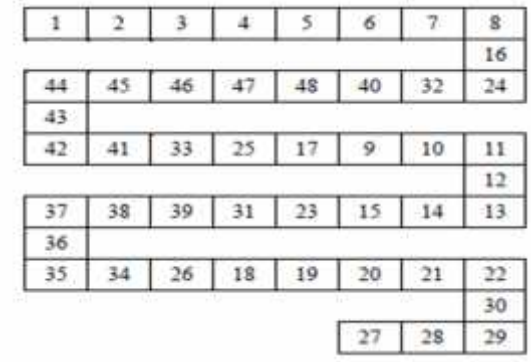
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

Gambar II.2. Matrik Susunan *Pixel* Gambar
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:90)

2. Dari matrik gambar II.2. dilakukan pembacaan secara spiral dimulai dari sudut kiri atas ke arah kanan, ke bawah, ke kiri, ke atas hingga berakhir di pusat matrik sehingga menghasilkan gambar II.3. Penggunaan metode pembacaan spiral merupakan penggabungan dua metode pembacaan secara horisontal dan vertikal, dan penggunaan metode pembacaan secara spiral akan menghasilkan bentuk untaian *pixel* gambar yang lebih rumit dibaca dari pada menggunakan metode pembacaan secara horisontal ataupun vertikal saja.

Gambar II.3. Pembacaan Matrik Secara Spiral
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:90)

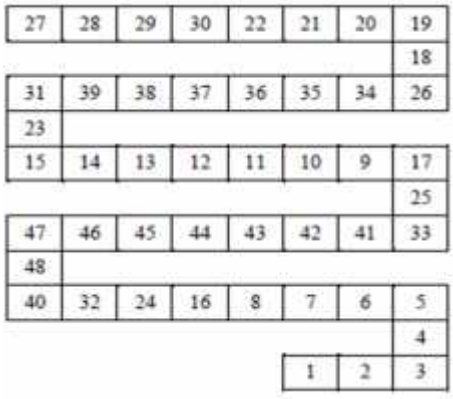
3. Dari hasil pembacaan matrik gambar II.3 dihasilkan sebuah untaian *pixel* seperti padagambar II.4.



1	2	3	4	5	6	7	8
							16
44	45	46	47	48	40	32	24
43							
42	41	33	25	17	9	10	11
							12
37	38	39	31	23	15	14	13
36							
35	34	26	18	19	20	21	22
							30
					27	28	29

Gambar II.4. Susunan untaian *pixel* hasil pembacaan spiral
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:90)

4. Dari untaian gambar II.4, dilakukan proses substitusi posisi *pixel* secara urut dengan konsep korespondensi satu-satu. Posisi *pixel* pertama digantikan dengan posisi pixel terakhir, sehingga didapat hasil untaian baru seperti pada gambar II.5.



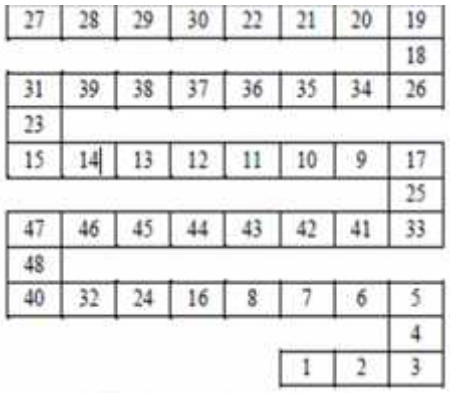
27	28	29	30	22	21	20	19
							18
31	39	38	37	36	35	34	26
23							
15	14	13	12	11	10	9	17
							25
47	46	45	44	43	42	41	33
48							
40	32	24	16	8	7	6	5
							4
					1	2	3

Gambar II.5. Susunan Untaian *Pixel* Setelah Dilakukan Substitusi.
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:90)

II.4.3. Enkripsi Teknik Transposisi

Pada teknik transposisi ini pembacaan matrik dilakukan dengan cara pembacaan kolom perkolom sesuai dengan kunci yang digunakan. Adapun langkah-langkahnya dapat diilustrasikan seperti berikut :

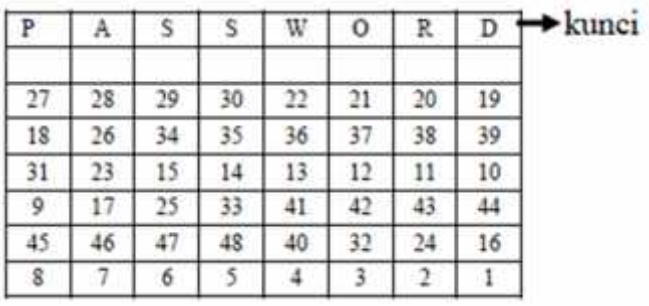
1. Susunan *pixel* hasil substitusi merupakan *plaintext* pada proses transposisi.



27	28	29	30	22	21	20	19
							18
31	39	38	37	36	35	34	26
23							
15	14	13	12	11	10	9	17
							25
47	46	45	44	43	42	41	33
48							
40	32	24	16	8	7	6	5
							4
					1	2	3

Gambar II.6 : Hasil Subtitusi Dijadikan Sebagai *Plaintext* Proses Transposisi
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:91)

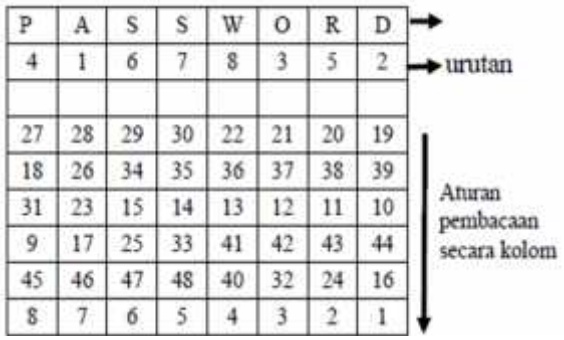
2. Untai dari matrik *plaintext* (gambar II.6) tersebut dimasukkan ke dalam matrik dengan ordo n dikali x, dengan n adalah panjang kunci yang digunakan dan x adalah jumlah *pixel* dibagi panjang kunci. Ilustrasinya terdapat pada gambar II.7.



P	A	S	S	W	O	R	D	→ kunci
27	28	29	30	22	21	20	19	
18	26	34	35	36	37	38	39	
31	23	15	14	13	12	11	10	
9	17	25	33	41	42	43	44	
45	46	47	48	40	32	24	16	
8	7	6	5	4	3	2	1	

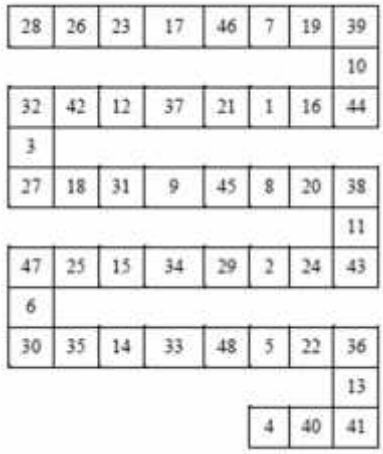
Gambar II.7. Matrik Susunan Awal Proses Transposisi
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:91)

3. Dari matrik gambar II.7, dilakukan pembuatan untaian *ciphertext* dengan pembacaan kolom per kolom sesuai dengan urutan abjad kunci yang telah diurutkan. Ilustrasinya pada gambar II.8.



Gambar II.8. Aturan Pembacaan Matrik Transposisi
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:91)

4. Kemudian matrik gambar II.8 dilakukan pembacaan kolom dimulai dari huruf "A" pada urutan ke-1, huruf "D" pada urutan ke- 2 dan seterusnya hingga huruf "W" pada urutan terakhir. Maka didapatkan bentuk untai seperti pada gambar II.9.



Gambar II.9. Untaian Hasil Pembacaan Kolom
(Sumber: Aji Supriyanto dan Eka Ardhianto, 2008:91)

5. Nilai *pixel* yang didapat dari untaian transposisi (gambar II.9) kemudian dimasukkan kembali ke dalam matrik baru sesuai dengan ordo ukuran gambar. Bentuk hasil akhirnya adalah pada gambar II.10.

28	26	23	17	46	7	19	39
10	44	16	1	21	37	12	42
32	3	27	18	31	9	45	8
20	38	11	43	24	2	29	34
15	25	47	6	30	35	14	33
48	5	22	36	13	41	40	4

Gambar II.10. Matrik Hasil Proses Transposisi
(Sumber : Aji Supriyanto dan Eka Ardhianto, 2008:91)

II.5. Pemrograman Visual Basic

II.5.1. *Visual basic 2010*

Visual basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi windows. *Visual basic 2010* merupakan aplikasi pemrograman yang menggunakan teknologi .NET Framework. Teknologi .NET Framework merupakan komponen windows yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi, dan halaman web. Teknologi .NET Framework mempunyai 2 komponen utama, yaitu CLR (Common Language Runtime) dan class library. CLR digunakan untuk menjalankan aplikasi yang berbasis .NET, sedangkan Library adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi. (Wahana Komputer; 2010: 2).

II.5.2. Sistem *Visual Basic 2010*

Sebelum menginstal *visual basic 2010*, komputer harus memenuhi beberapa persyaratan agar *visual basic 2010* dapat dijalankan dengan baik.

Adapun, persyaratan (system requirments) yang harus dipenuhi dapat dilihat pada tabel II.1. berikut :

Tabel II.1. Sistem *Requirements Visual Basic 2010*

Sistem	Syarat minimal	Syarat yang direkomendasikan
Arsitektur	x86 dan x64	
Sistem Operasi	Microsoft Windows 7 Service pack 2 Microsoft Windows Server 2003 Windows Vista	
Prosesor	CPU 1.6 GHz (Giga Hertz)	Windows XP dan Windows Server 2003: CPU 2,2 GHz atau yang lebih tinggi. Windows Vista: CPU 2,4 GHz.
RAM	Windows XP dan Windows Server 2003 384 MB (Mega Byte) Windows Vista: 768 MB.	RAM 10 24 MB/1GB atau yang lebih besar.
Harddisk	Tanpa MSDN R Ruang kosong harddisk pada drive penginstalan 2 GB. Sisa ruang harddisk kosong 1 GB.	Kecepatan harddisk 7200 RPM atau yang lebih tinggi.
	Tanpa MSDN R Ruang kosong harddisk pada drive penginstalan 3,8 GB (MSDN diinstal full). 2,8 GB untuk menginstal MSDN default. Kecepatan harddisk 5400 RPM.	
Display Layar	1024x768 display	1280x1024 display

(Sumber : Andi, 2010, 2.)

II.5.3. Mengenal Area Kerja *Visual basic 2010*

Setelah berhasil menginstal *visual studio 2008* yang didalamnya terdapat *visual basic 2010*, maka selanjutnya adalah mencoba menjalankan dan mengenal lingkungan kerja *visual basic 2010* Lingkungan kerja *visual basic* atau disebut *Integrated Development Environment* (*IDE*) adalah suatu lingkungan kerja tempat programmer melakukan pemrograman yang didukung oleh *compiler*, *editor* baik *editor grafis* maupun kode, dan lain sebagainya untuk memudahkan pemrograman. (Wahana Komputer; 2010: 3).

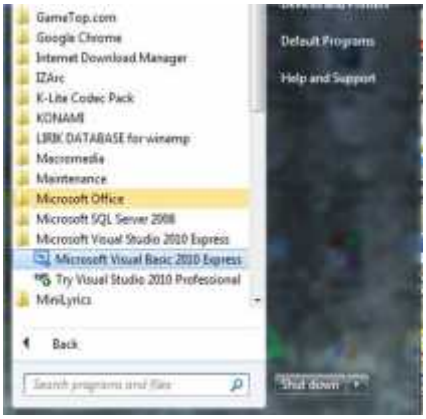
II.5.3.1. Membuka *IDE Visual basic 2010*

Langkah membuka lingkungan kerja *visual basic 2010* sebagai berikut :

1. Ikuti salah satu cara berikut untuk membuka *IDE* visual basic :
   a. Klik ganda shortcut *visual basic 2010* pada desktop.

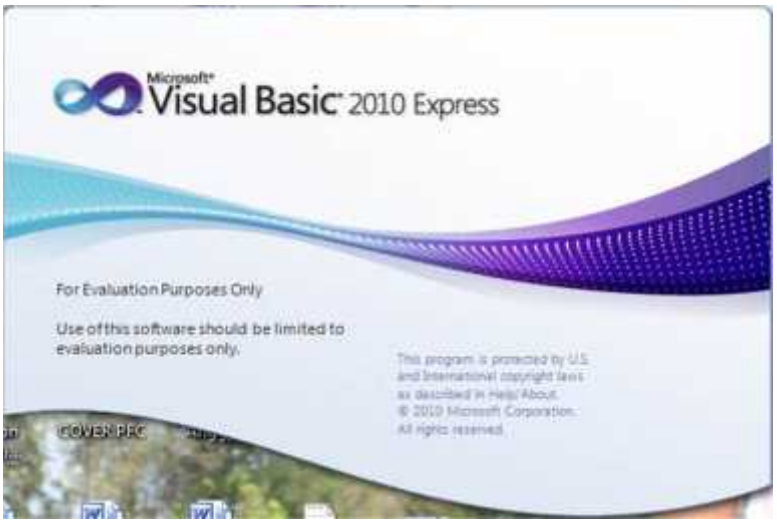
Gambar II.11. Ikon *Shortcut* Pada Desktop
Sumber : Andi, 2010, 4.

   b. Melalui menu *start* > *Microsoft Visual Studio 2010* > *Microsoft Visual Studio 2010.*



Gambar II.12. Membuka *Visual Studio 2010* Melalui Menu
Sumber : Andi, 2010, 4.

2. Pada saat pertama kali menjalankan *visual basic 2010*, akan muncul kotak dinformasi proses setting *IDE visual basic 2010*. Setelah proses setting selesai, kotak *splash screen* sesuai dengan gambar II.13. akan muncul :

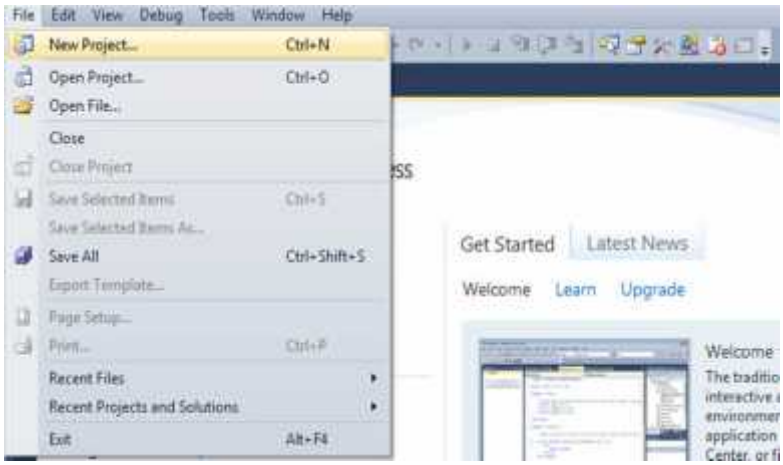


Gambar II.13. *Splash screen* visual studio 2010.
Sumber : Andi, 2010, 4

3. Setelah itu, akan terbuka lingkungan kerja atau *IDE* dari *visual basic 2010* yang ditunjukkan pada gambar II.14. berikut :

Gambar II.14. Area kerja *visual basic 2010* saat pertama kali dijalankan
Sumber : Andi, 2010, 5.

4. Untuk membuat sebuah project baru menggunakan Visual Studio .NET 2010, klik menu **File | New | Project.**



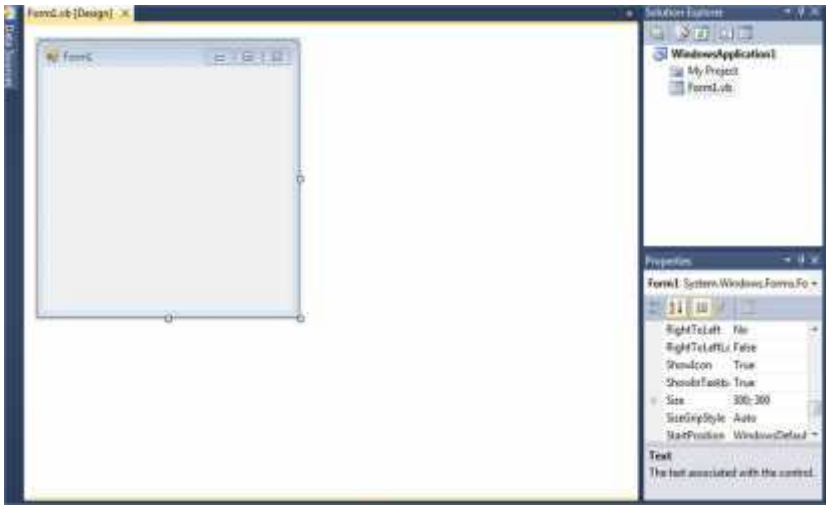
Gambar II.15. Menu New Project
Sumber : Andi, 2010, 5

5. Setelah itu akan muncul kotak dialog **New Project**. Pada kotak dialog **New Project** terdapat beberapa pilihan tool untuk pengembangan aplikasi, seperti Visual Basic, Visual C# dan Visual C++. Pilih **Visual Basic** kemudian pilih

Windows Form Application. Beri nama project yang akan dibuat pada bagian **Name** dan direktori tempat menyimpan project pada bagian **Location**.

Gambar II.16. Kotak Dialog *New Project*
Sumber : Andi, 2010, 5

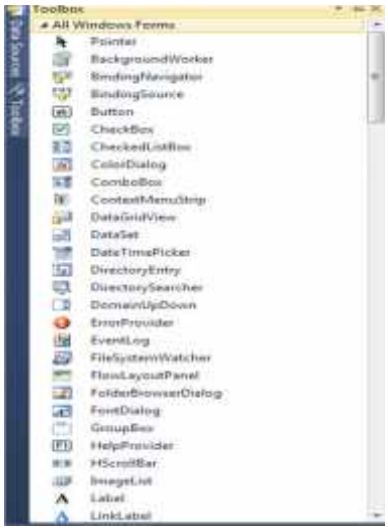
6. Selanjutnya muncul Visual Basic IDE tempat untuk membangun aplikasi Visual Basic .NET 2010.



Gambar II.17. IDE Visual Studio .NET 2010
Sumber : Andi, 2010, 5

Pada *project visual basic* untuk *windows application* secara *default* telah terdapat sebuah *form*. *Form* tersebut bernama *Form1*. Pada *form* inilah tempat untuk meletakkan kontrol-kontrol atau komponen-komponen untuk membuat sebuah aplikasi *windows. Form* dan kontrol-kontrol dari program aplikasi inilah yang biasanya disebut dengan GUI (*Graphical User Interface*) atau antar muka dari program. Jadi *user* akan berinteraksi dengan sebuah program aplikasi melalui GUI. Pada IDE *Visual Basic .NET* 2010 terdapat *menu, toolbar, toolbox, server explorer, solution explorer* dan *properties window***.**

7. *Toolbox* adalah tempat dimana kontrol-kontrol dan komponen-komponen diletakkan. Kontrol dan komponen yang terdapat pada *toolbox* dipakai dalam pembuatan program aplikasi. Untuk membuat objek kontrol dan komponen pada *form* program aplikasi diambil dari kontrol-kontrol yang ada pada *toolbox*. Untuk menampilkan *windows toolbox, klik* pada tombol *toolbox* yang terdapat pada *toolbar*.



Gambar II.18. *Toolbox*
Sumber : Andi, 2010, 5

II.6. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia perkembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi perkembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (sharing) dan mengkomunikasikan rancangan dengan baik. (Munawar; 2005: 17).

UML merupakan kesatuan bahasa pemodelan yang dikembangkan oleh *Booch*, *Object Modeling Technique* (OMT) dan *Object Oriented Engineering* (OOSE). Metode *Booch* dari *Grady Booch* sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan desain ke dalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi semantik dari hubungan objek dan kelas tersebut, perincian *interface* dan implementasi. Keunggulan metode *Booch* adalah pada detail dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh *Rumbaugh* didasarkan pada analisis terstuktur dan pemodelan *entity-relationship*. Tahapan utama dalam metodologi ini adalah analisis, desain sistem, desain objek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari *Jacobson* lebih memberi penekanan dan *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement* dan analisis, desain dan implementasi dan model pengujian (test Model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi sederhana namun

mencakup seluruh tahapan dalam rekayasa perangkat lunak. (Munawar; 2005: 17).

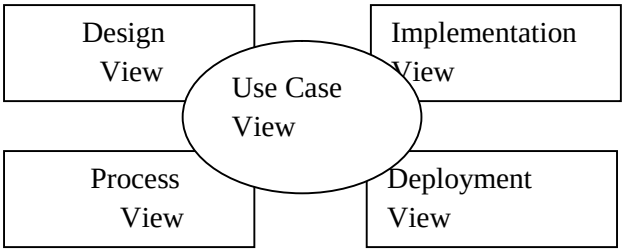
Dengan UML, metode *Booch*, OMT dan OOSE digabungkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari pada metode lainnya. Unsur-unsur yang membentuk UML ditunjukkan dalam Gambar II.19 : (Munawar; 2005: 18).

Gambar II.19. Unsur-unsur yang membentuk UML
Sumber : Graha Ilmu, 2005, 18

UML adalah hasil kerja dari konsorsium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam OOAD (*Object Oriented Analysis* dan *Design*). UML tidak hanya domain dalam penotasian dilingkungan OO tetapi juga populer di luar lingkungan OO. Ada tiga karakter penting yang melekat di UML yaitu sketsa, cetak biru dan bahasa pemrograman. Sebagai sebuah sketsa UML bisa berfungsi sebagai sebuah cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi detail tentang *coding* program (*Forward engineering*) atau bahkan membaca program dan

mengimplementasikannya kembali ke dalam diagram (*reverse engineering*). *Reverse engineering* sangat berguna pada situasi dimana kode program yang tidak terdokumentasi asli hilang atau bahkan belum perna dibuat sama sekali. Sebagai bahasa pemrograman, UML dapat diterjemahkan diagram yang ada di UML menjadi kode program siap untuk dijalankan. (Munawar; 2005: 18).

UML dibangun atas model 4+1 *view*. Model ini didasarkan pada fakta bahhwa struktur sebuah sistem dideskripsikan dalam *view* dimana salah satu diantaranya *use case view*. *use case view* ini memegang peran khusus untuk mengintegrasikan *content* ke *view* yang lain. Model 4+1 *view* ditunjukkan pada gambar II.20 : (Munawar; 2005: 20).



Gambar II.20 Model 4+1 *View*
Sumber : Graha Ilmu, 2005, 20

Kelima *view* tersebut tidak berhubungan dengan diagram yang dideskripsikan di UML. Setiap *view* berhubungan dengan perspektif tertentu dimana sistem akan diuji. *View* yang berbeda akan menekankan pada aspek yang berbeda dari sistem yang mewakili tentang sistem bisa dibentuk dengan menggabungkan informasi-informasi yang ada pada kelima *view* tersebut. (Munawar; 2005: 20).

Use case view mendefinisikan perilaku eksternal sistem. Hal ini menjadi daya tarik bagi *end user*, analis dan tester. Pandangan ini mendefenisikan kebutuhan sistem karena mengandung semua *view* yang lain yang mendeskripsikan aspek-aspek tertentu dari peran dan sering dikatakan yang mendrive proses perkembangan perangkat lunak. (Munawar; 2005: 20).

Design view mendeskripsikan struktur logika yang mendukung fungsi-fungsi yang dibutuhkan di *use case*. *Design view* ini berisi definisi komponen program, *class-class* utama bersama-sama dengan spesifikasi data, perilaku dan interaksinya. Informasi yang terkandung di *view* menjadi pergantian para progremer karena menjelaskan secara detil bagaimana fungsionalitas sistem akan diimplementasikan. (Munawar; 2005: 20).

Implementasi *view* menjelaskan komponen-komponen visi yang akan dibangun. Hal ini berbeda dengan komponen logic yang dideskripsikan pada *design view*. Termasuk disini diantaranya *file exe*, *library* dan *database*. Informasi yang ada di *view* dan integrasi sistem. (Munawar; 2005: 21).

Proses *view* berhubungan dengan hal-hal yang berkaitan dengan *concurrency do* dalam sistem. Sedangkan *deployment view* menjelaskan bagaimana komponen-komponen fisik didistribusikan ke lingkungan fisik seperti jaringan komputer dimana sistem akan dijalankan. Kedua *view* ini menunjukan kebutuhan non fungsional dari sistem seperti toleransi kesalahan dan hal-hal yang berhubungan dengan kinerja. (Munawar; 2005: 21).

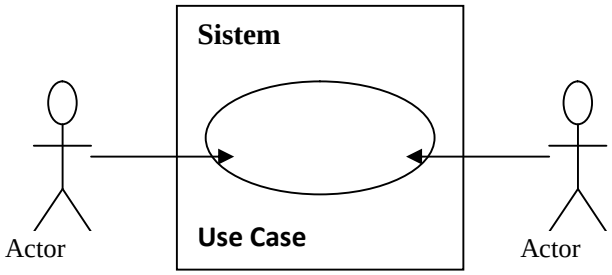
II.6.1. *Use Case* Diagram

Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna. (Munawar; 2005: 63).

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. (Munawar; 2005: 63).

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*. (Munawar; 2005: 63).

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary. Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor, use case* dan *system* ditunjukkan pada gambar II.21 : (Munawar; 2005: 64).



Gambar II.21 *Use Case* Diagram
Sumber : Graha Ilmu, 2005, 64

II.6.2. *Class* Diagram

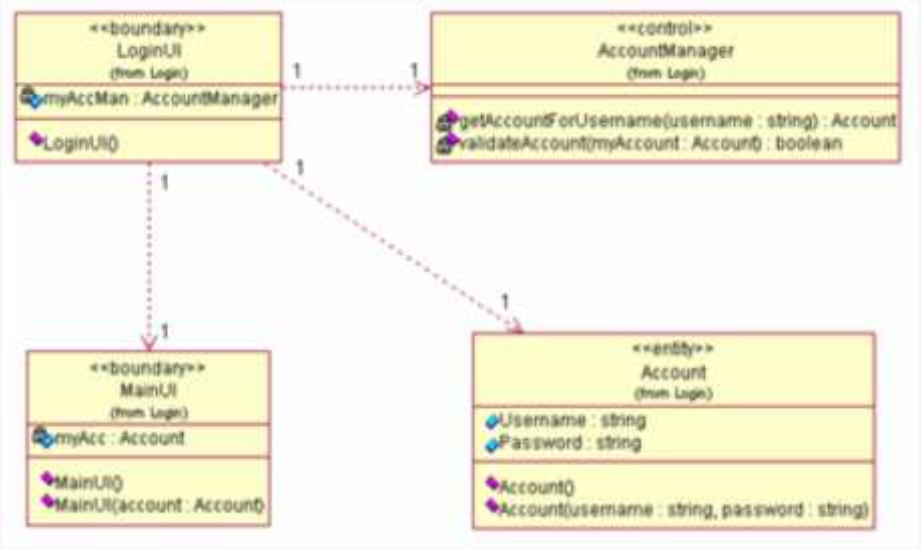
Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class* diagram membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class* diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. Seperti yang ditunjukkan pada gambar notasi *class* diagram II.22 berikut : (Haviluddin; 2011: 3).

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)

2. Atribut

3. Metoda

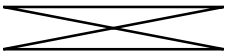


Gambar II.22. Notasi *Class* Diagram
Sumber : Haviluddin, 2011, 3

II.6.3. *Activity* Diagram

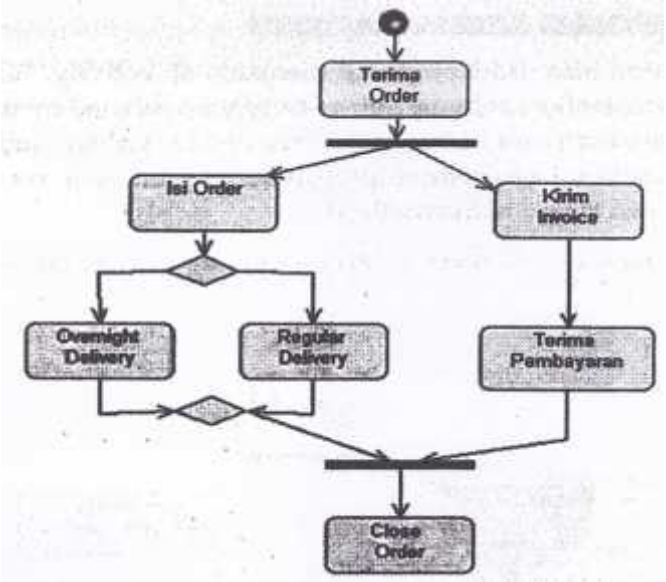
Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity* diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity* diagram bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity* diagram. Seperti yang ditunjukkan pada tabel II.2 berikut: (Munawar; 2005: 109).

Tabel II.2. Simbol-Simbol Yang Sering Dipakai Pada *Activity* Diagram

Simbol	Keterangan
	Titik Awal
	Titik Akhir
	Activity
	Pilihan untuk mengambil keputusan
	Foks: Untuk menunjukkan kegiatan yang dilakukan secara paralel
	Rake: Menunjukkan adanya dekomposisi
	Tanda Waktu
	Tanda Penerimaan
	Aliran Akhir (Flow Final)

Sumber : Graha Ilmu, 2005, 109

Contoh *activity diagram* sederhana ditunjukkan pada gambar II.23. (Munawar; 2005: 111).



Gambar II.23 Contoh *Activity Diagram* Sederhana
Sumber : Graha Ilmu, 2005, 111

II.6.4 *Squence* Diagram

Squence diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sebuah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*. (Munawar; 2005: 87).

Komponen utama *Squence* diagram terdiri dari atas objek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*. (Munawar; 2005: 87).

1. Objek / *participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram. Setiap *participant* dihubungkan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. *Activation* mewakili sebuah eksekusi operasi dari participant. Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada gambar II.24 : (Munawar; 2005: 87).

Gambar II.24 Bentuk *Participant*
Sumber : Graha Ilmu, 2005, 87

2. *Messege*

Sebuah *messege* bergerak dari suatu *participant* ke *participant* yang lain dan dari *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. (Munawar; 2005: 88).

Sebuah *message* bisa jadi *simple*, *synchronous* atau *asynchoronous*. *Message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya. Jika suatu *participant* mengirimkan sebuah *message* tersebut akan ditunggu sebelum di proses dengan urusannya. Namun jika *message asynchoronous* yang dikirimkan, maka jawabannya atas *message* tersebut tidak perlu ditunggu. Simbol *message* pada *squnence* diagram dapat dilihat pada gambar II.25 : (Munawar; 2005: 87).

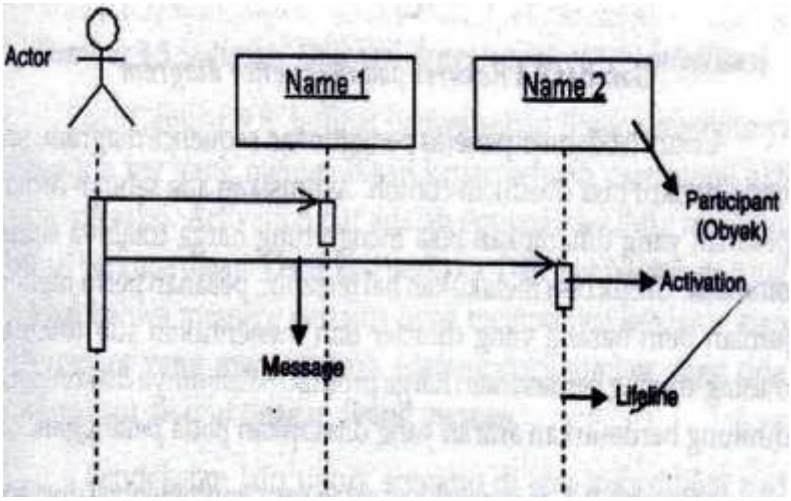
Gambar II.25. Bentuk *Messege*
Sumber : Graha Ilmu, 2005, 88

3. *Time*

Time adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat kebawah. (Munawar; 2005: 88).

Terdapat dua dimensi pada *squence* diagram yaitu dimensi dari kiri ke kanan menunjukkan tata letak participant dan dimensi dari atas ke bawah

menunjukkan lintasan waktu. Simbol-simbol yang ada pada *squence* diagram ditunjukkan pada gambar II.26 : (Munawar; 2005: 89).



Gambar II.26 Bentuk *Time*
Sumber : Graha Ilmu, 2005, 88