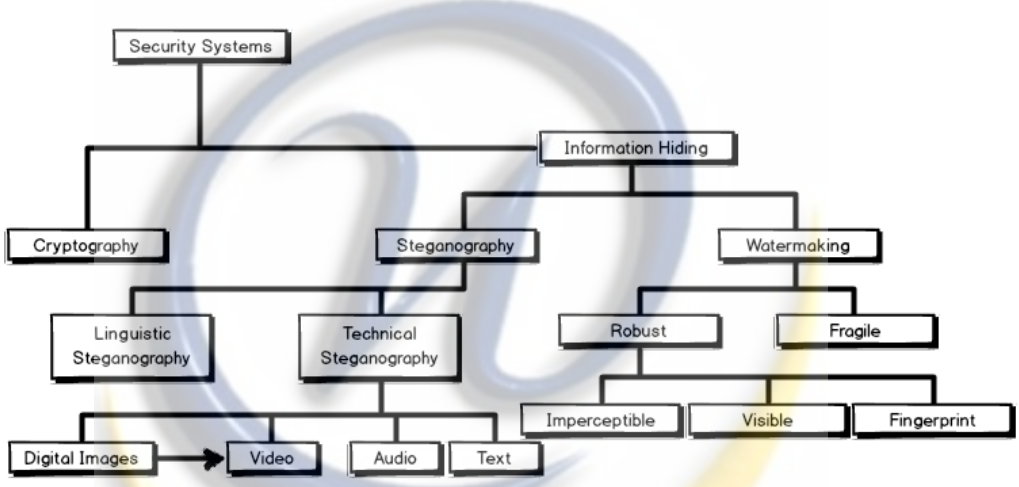
BAB II

LANDASAN TEORI

2.1 Pengertian Steganografi

Steganografi merupakan salah satu teknik menyembunyikan informasi, seperti gambar dibawah ini klasifikasi teknik penyembunyian informasi.



Gambar 2.1 Klasifikasi teknik penyembunyian informasi[10]

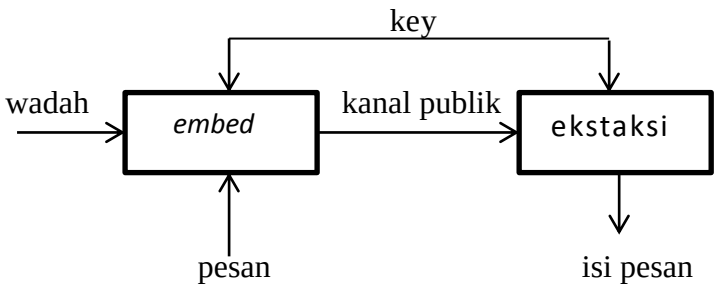
Umumnya untuk menyembunyikan informasi dapat digambarkan sebagai data tertanam oleh pesan khusus yang akan dikirim secara rahasia. Biasanya informasi tersembunyi didalam pesan dikenal sebagai wadah teks, wadah gambar, atau wadah audio. Stego-key diterapkan untuk mengendalikan proses penyembunyian dan membatasi deteksi atau mengembalikan data yang tersembunyi[10].

Steganografi (*covered writing*) didefinisikan sebagai ilmu dan seni untuk menyembunyikan pesan rahasia sehingga keberadaan pesan tidak terdeteksi oleh indera manusia. Salah satu tujuan dari steganografi adalah mengirimkan informasi rahasia melalui jaringan tanpa menimbulkan kecurigaan.

Steganografi sudah digunakan sejak dahulu kala sekitar 2500 tahun yang lalu untuk kepentingan politik, militer, diplomatik, serta untuk kepentingan pribadi. Dan sesungguhnya prinsip dasar dalam steganografi lebih dikonsentrasikan pada kerahasian komunikasinya bukan pada datanya (Johnson, 1995).

Steganografi memerlukan setidaknya dua properti. Properti pertama adalah wadah penampung (*cover*) dan yang kedua adalah data atau pesan yang disembunyikan. Untuk meningkatkan tingkat keamanan data yang disimpan, dapat dilakukan dengan menambahkan properti kunci (*key*) rahasia. Properti wadah (*cover*) yang mungkin digunakan untuk menyimpan pesan dalam steganografi sangat beragam. Medium wadah tersebut antara lain citra, suara, video ataupun teks. Adapun data yang disimpan juga dapat berupa audio, citra, video maupun teks[1].

Skema penyembunyian data dalam steganografi secara umum adalah data atau informasi yang ingin disembunyikan disimpan dalam sebuah wadah (*cover*) melalui suatu algoritma steganografi tertentu. Untuk menambah tingkat keamanan data, dapat diberikan kunci, agar tidak semua orang mampu mengungkapkan data yang disimpan dalam berkas wadah (*cover*). Hasil akhir dari proses penyimpanan data ini adalah sebuah berkas stego (*stego data/stego file*). Pertimbangan pemilihan penggunaan kunci dari segi tipe serta panjang kunci adalah suatu hal yang juga berperan penting dalam pengamanan data yang tersimpan dalam steganografi[2].



Gambar 2.2 Proses Steganografi[2]

2.1.1 Teknik Steganografi Audio

Pada audio terdapat beberapa teknik dalam penyisipan pesan (informasi), diantaranya adalah *Low Bit Coding, Phase Coding, Spread Spectrum, dan Echo Data Hiding.*

1. *Low Bit Coding*

Adalah cara untuk menyimpan data ke dalam file audio, mengganti bit yang paling tidak penting atau *low significant bit* (LSB) pada setiap titik samping dengan string berkode biner (coded binary string), Kelemahan metode ini adalah lemahnya kekebalan terhadap manipulasi. Pada prakteknya, metode ini hanya berguna pada lingkungan digital-to-digital yang tertutup atau dengan kata lain pengiriman pesan digital ke digital melalui publik berisikan pesan rahasia/pribadi tertutup yang biasa digunakan kepentingan militer, politik, diplomatik serta kepentingan pribadi.

2. *Phase Coding*

Adalah merekayasa fasa dari sinyal masukan. Dengan mensubstitusi awal fasa dari tiap awal segmen dengan fasa yang telah dibuat sedemikian rupa dan merepresentasikan pesan yang disembunyikan. Hal ini menghasilkan keluaran yang lebih baik namun dikompensasikan dengan kerumitan dalam realisasinya.

3. *Spread Spectrum*

Adalah penyebaran spektrum, pesan dikodekan dan disebar ke setiap spektrum frekuensi yang memungkinkan. Maka dari itu akan sangat sulit bagi yang akan mencoba memecahkannya kecuali ia memiliki akses terhadap data tersebut atau dapat merekonstruksi sinyal *random* yang digunakan untuk menyebarkan pesan pada range frekuensi.

4. *Echo Data Hiding*

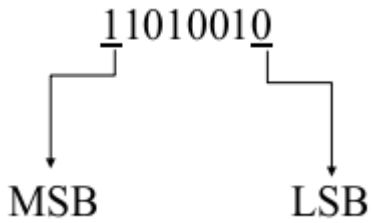
Adalah menyembunyikan pesan melalui teknik echo, menyamarkan pesan ke dalam sinyal yang membentuk echo, pesan disembunyikan dengan

menvariasikan tiga parameter dalam echo yaitu besar amplitude awal, tingkat penurunan atenuasi dan offset.

2.2 Pengertian *Low Bit Coding*

Pada dasarnya, metode steganografi *Low Bit Coding* pada *audio* sama saja dengan metode steganografi *Least Signifikan Bit* (LSB) pada *image* (citra). Pada metode ini sebagian bit pada *file audio* diubah menjadi nilai lain dalam representasi biner. Perubahan dapat dilakukan dengan berbagai cara dan algoritma, misalnya mengubah nilai biner 0 menjadi 1 atau sebaliknya, melakukan operasi XOR antara nilai biner pada *file* dengan nilai biner pada kunci, karena dalam representasi biner, maka perubahan yang mungkin terjadi adalah nilai 1 menjadi 0, atau nilai biner 0 menjadi 1. Suatu *file audio* dapat memiliki satu *channel* (*mono*) atau dua *channel* (*stereo*). Secara umum, kapasitas satu *channel* adalah kbps per kilohertz, karena ukuran channel dapat mencapai 44000 byte, maka kapasitas maksimal yang dapat ditampung oleh satu channel adalah 44kbps per kilohertz[3].

Low Bit Coding merupakan salah satu *Steganography* yang paling banyak dipakai dalam menyembunyikan pesan. Cara kerjanya adalah dengan memasukkan tiap *bit* dari pesan yang ingin disembunyikan ke dalam akhir dari 1 *byte* pada data audio[3]. Contoh: pada *file audio* setelah di ubah menjadi *file* biner maka menjadi seperti berikut:



Gambar 2.3 LSB[3]

MSB : *Most Significant Bit*

LSB : *Least Significant Bit*

Pada gambar 2.3, menandakan bahwa bit 1 dari depan menyatakan bit MSB dan bit 0 dari bilangan biner terakhir adalah bit LSB. Dapat dilihat contoh dibawah ini.

1. Jika pesan = 8 bit, maka jumlah byte yang digunakan = 8 byte

00110011 10100010 10100011 00100110

01011001 01101110 10110101 00010101

Misalkan binary dari embedded message: e = 01100101

Hasil penyisipan pada bit LSB:

00110010 10100011 10100011 00100110

01011000 01101111 10110100 00010101

Pada contoh diatas, hanya sebagian yang berubah dari Least Significant Bit. Berdasarkan teori maka didapatkan bahwa ukuran file asli tidak mengalami perubahan yang begitu besar sehingga sulit terdeteksi oleh indra manusia

Cara penyembunyian pesan dengan menggunakan *Low Bit Encoding* pada *file audio WAVE* tergantung dari *format file WAVE* tersebut apakah *file WAVE* mempunyai *format bit/sample* sama dengan 8-bit atau 16-bit. Jadi penyembunyian pesan diterapkan pada bit terakhir yaitu bit ke-8 atau pada bit ke-16.

Decoding untuk menampilkan pesan rahasia ke nilai awal ataupun pesan asli yaitu pada proses ini pesan asli yang telah disisipi *file* audio diekstraksi. Proses ekstraksi dimulai dengan menginputkan pesan yang sudah disisipi *file* audio kemudian ekstrak dan simpan nama pesan yang sudah diekstrak. Pada proses ini maka bit-bit yang sudah diganti pada proses encodingkan diubah kembali ke nilai awal. Adapun prosedur ekstraksi pesan rahasia ini adalah :

1. *Input* pesan rahasia yang sudah disisipkan ke dalam *file audio*.
2. Kemudian ekstrak pesan stego tersebut.

3. Pesan rahasia yang sudah disisipkan pada file audio akan dibaca kembali.
4. Pesan akan diekstrak dan dikembalikan ke bentuk semula.
5. Simpan pesan rahasia.

Untuk menampilkan pesan asli setelah proses *encoding* yaitu melalui proses *decoding*. Adapun langkah-langkah proses *decoding* yaitu sebagai berikut. Buka pesan rahasia yang sudah disisipkan ke dalam media audio:

Gambar 2.4 Pesan Rahasia yang telah disisipkan dibiner audio

Ekstrak pesan *stego* tersebut dengan mengambil setiap bit akhir pada *file stego*. Contoh membaca pesan baris pertama yang diberi tanda merah.

Berikut langkah-langkahnya:

a. Membuat blok-blok data ke dalam 8 *byte* per blok. Untuk setiap blok dikerjakan langkah “b” sampai dengan langkah “c”untuk i = 0,1,2,3,.,7.

b. Mengambil nilai bit terakhir bytepesan ke-i dengan meng-and-kan dengan 1.

c. Menyimpan hasil setelah di-and-kan dengan 1, dan mengalikan dengan nilai posisi bit, yaitu : $(2^{(7-i)})$.

d. Menjumlahkan semua hasil perhitungan untuk i=0 sampai dengan i=7.

e. Menentukan karakter *ASCII* yang bersesuaian dengan hasil perhitungan. Sebagai contoh akan dilakukan *decoding* untuk membaca informasi yang disisipkan dengan mengambil nilai bit lsb dari media penampung file WAVseperti berikut ini:

Contohnya hasil dari baris pertama:

01010010=>01010010 and 1=0 nilai=0x2^7=0

01001001=>01001001 and 1=1 nilai=1x2^6=64

01000110=>01000111 and 1=1 nilai=1x2^5=32

01000110=>01000110 and1=0 nilai=0x2^4=0

11101000=>11101000 and 1=0 nilai=0x2^3=0

01001111=>01001110 and 1=0 nilai=0x2^2=0

00000100=>00000101 and1=1 nilai=1x2^1=2

00000000=>00000001 and1=1 nilai=1x2^0=1 +

= 99 kode ASCII yaitu “c”

Hasil dari baris pertama berisikan huruf ”c”

2.3 Auto Digital

Audio Digital adalah harmonisasi bunyi yang dibuat melalui perekaman konvensional maupun suara sintetis yang disimpan dalam media berbasis teknologi komputer. Format digital dapat menyimpan data dalam jumlah besar, jangka panjang dan berjaringan luas[11].

Penyimpanan bentuk audio (suara) dalam format digital memiliki beberapa keuntungan dibandingkan penyimpanan dalam bentuk konvensional seperti kaset atau CD. Adapun keuntungan tersebut adalah :

a. Format yang beragam dapat disesuaikan dengan teknologi yang digunakan
b. Kualitas copy yang serupa dengan master memudahkan penggandaan dari pihak perusahaan rekaman tanpa menurunkan mutu
c. Proses penjualan dengan pendekatan single atau satu lagu terbukti jauh lebih efektif dan efisien ketimbang medium konvensional seperti kaset atau CD

Namun, peyimpanan dalam bentuk digital ini juga memiliki beberapa kelemahan, antara lain :

a. Kemudahan perekaman dan penggandaan rekaman memacu terjadinya pembajakan yang tentu saja akan merugikan
b. Penyebaran audio digital di internet tidak bisa sepenuhnya dikontrol oleh label sehingga mempengaruhi pemasukan untuk label[11].

2.4 Pengertian *Audio Digital WAV*

WAV disebut dengan sebutan singkat untuk *Wave form Audio Format*. standar format *file audio* yang dikembangkan oleh *Microsoft* dan *IBM*. WAV merupakan varian dari format bitstream RIFF dan mirip dengan format IFF dan AIFF yang digunakan komputer Amiga dan Macintosh. Format suara WAV merupakan standar dari RIFF (*Resource Interchage File Format*). Format suara WAV dipilih karena format ini banyak digunakan, dan memiliki kualitas suara yang sangat baik.

Baik WAV maupun AIFF kompatibel dengan operating *system Windows* dan *Macintosh*. meski WAV dapat menampung audio dalam bentuk terkompresi, umumnya format WAV merupakan audio yang tidak terkompresi. Sehingga jika ingin menyimpan dan dapat terbaca oleh sebuah komputer maka suara tersebut harus disimpan dalam bentuk digital hal ini bisa dilakukan dengan mengambil

sampel sejumlah bagian gelombang perdetiknya, lalu disimpan ke komputer dalam bentuk format WAV.

Jenis format *Wave* ini merupakan jenis *file Wave* yang paling umum dan hampir dikenal oleh setiap program. Format *Wave PCM (Pulse Code Modulation)* adalah *file Wave* yang tidak terkompresi, akibatnya ukuran file sangat besar jika *file* mempunyai durasi yang panjang[4].

2.5 Teks

Berkas ASCII (American Standard Code For Information Interchange) atau teks (bahasa Inggris: 'plain text') dalam istilah komputer, adalah suatu jenis berkas komputer yang berupa teks tidak terformat. Lawan dari jenis berkas ini adalah teks berformat. Jenis berkas ini biasanya disunting dengan menggunakan editor teks. Berkas ini biasanya hanya mengandung teks-teks yang diformat dengan menggunakan pengkodean ASCII. Berkas ini hanya terdiri atas karakter, angka, tanda baca, tabulasi, dan karakter pemisah baris (*carriage return*).

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Tabel 2.1: Tabel ASCII

2.6 UML

UML (*Unified Modeling Language)* adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

2.6.1 Bagian-Bagian UML

Bagian-bagian utama dari UML adalah *view, diagram, model element,* dan *general mechanism.*

a. ***View***

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram.

b. ***Use case view***

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya. *Use case view* digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *Use case view* digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

c. ***Logical view***

Mendeskripsikan bagaimana fungsionalitas dari sistem,

struktur statis (*class*, *object* dan *relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *Logical view* digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state, sequence, collaboration*, dan *activity diagram* untuk model dinamisnya. *Logical view* digunakan untuk perancang (*designer*) dan pengembang (*developer*).

d. ***Component view***

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrasi lainnya. *Component view* digunakan untuk pengembang (*developer*).

e. ***Concurrency view***

Membagi sistem ke dalam proses dan prosesor. *Concurrency view* digambarkan dalam diagram dinamis (*state, sequence, collaboration*, dan *activity diagram)* serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

f. ***Deployment view***

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *Deployment view* digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

g. ***Diagram***

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu.

Adapun jenis diagram antara lain :

1. ***Use Case Diagram***

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*..

2. ***Class Diagram***

Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

3. ***Component Diagram***

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class*. Komponen dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga *component diagram* merepresentasikan dunia riil yaitu *component software* yang mengandung *component, interface dan relationship*.

4. *Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes, executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

5. *State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu class dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

6. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah sistem. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antar *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

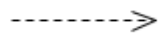
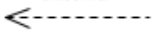
7. Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration* diagrams menggambarkan object dan hubungannya (mengacu ke konteks).

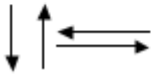
8. *Activity Diagram*

Menggambarkan rangkaian aliran dari aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

Tabel 2.2: *Use Case Diagram*[9]

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2	<< include >> 	*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
3	<< extend >> 	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
4		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
5		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
6		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor

Tabel 2.3: *Activity Diagram*[9]

NO	GAMBAR	NAMA	KETERANGAN
1		*Activity*	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		*Activity Final Node*	Bagaimana objek dibentuk dan diakhiri
5		*Decision*	Diguanakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu
6		*Line Connector*	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya

2.6.2 Tujuan Penggunaan UML

Tujuan dari penggunaan UML adalah sebagai berikut :

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahas pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

Memberikan model yang siap pakai, bahasa pemodelan *visual* yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum. UML bisa juga berfungsi sebagai sebuah (*blue print*) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi secara detail tentang *coding* program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram (*reserve engineering*)[9].

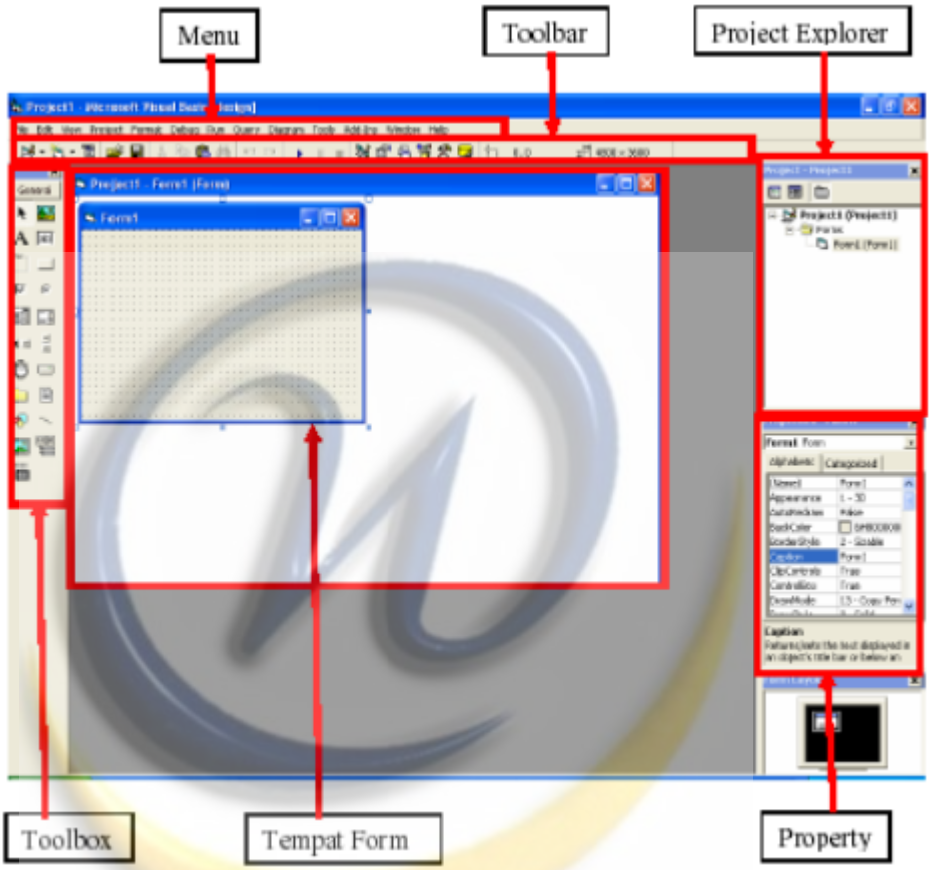
2.7 Pengertian *Microsoft Visual Studio*

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi. Baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya dalam bentuk aplikasi *Console*, aplikasi *Windows*, ataupun aplikasi *Web*. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment (IDE)*, dan dokumentasi (umumnya berupa *MSDN Library*). Kompiler yang dimasukkan ke dalam paket *Visual Studio* antara *lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro,* dan *Visual SourceSafe.*

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di *Windows*) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* diatas *.NET Framework*). Selain itu, *Visual Studio* juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas *.NET Compact Framework)*[5].

1. Antar Muka *Visual Studio*

Interface atau antar muka *Visual Studio*, berisi *menu, toolbar, toolbox, form, project explorer* dan *property* seperti terlihat pada Gambar 2.5 berikut:



Gambar. 2.5 *Interface Visual Studio*[5]

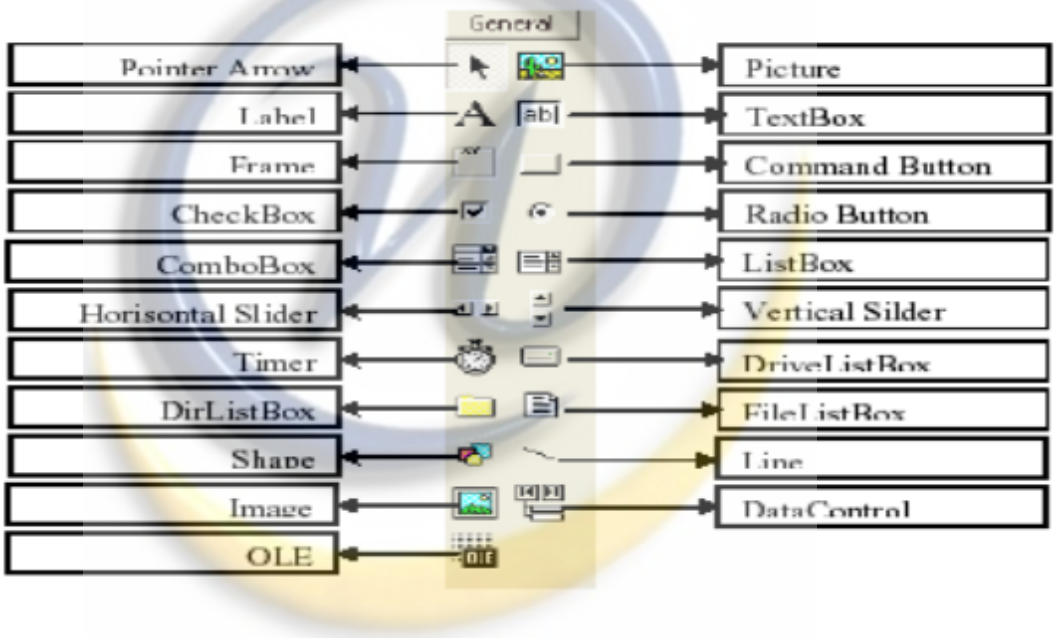
Pembuatan program aplikasi menggunakan *Visual Studio* dilakukan dengan membuat tampilan aplikasi pada *form*, kemudian diberi *script* program di dalam komponen-komponen yang diperlukan. *Form* disusun oleh komponen-komponen yang berada di *[Toolbox]*, dan setiap komponen yang dipakai harus diatur propertinya lewat jendela *[Property]*.

Menu pada dasarnya adalah operasional standar di dalam sistem operasi *windows*, seperti membuat *form* baru, membuat *project* baru, membuka *project* dan menyimpan *project*. Di samping itu terdapat fasilitas-fasilitas pemakaian

Visual Studio pada *menu*. Untuk lebih jelasnya *Visual Studio* menyediakan bantuan yang sangat lengkap dan detail dalam MSDN (*Microsoft Developer Network*).

a. Toolbox

Toolbox berisi komponen-komponen yang bisa digunakan oleh suatu project aktif, artinya isi komponen dalam toolbox sangat tergantung pada jenis project yang dibangun. Komponen standar dalam toolbox dapat dilihat pada Gambar. 2.6 berikut ini.



Gambar. 2.6 Komponen standar dalam *Toolbox*[5]

Toolbox Visual Studio dengan semua kontrol intrinsik. Jendela *Toolbox* merupakan jendela yang sangat penting. Dari jendela ini dapat mengambil komponen-komponen *(object)* yang akan ditanamkan pada f*orm* untuk membentuk *user interface*.

b. Variabel

Variabel adalah tempat dalam memori komputer yang diberi nama (sebagai pengenal) dan dialokasikan untuk menampung data. Sesuai data yang ditampung maka variabel harus mempunyai tipe data yang sesuai dengan isinya.

c. Operator

Operator digunakan untuk menghubungkan variabel dengan variable lain untuk melakukan berbagai manipulasi dan pengolahan data.

2. Konsep Dasar Pemrograman Dalam *Visual Studio*

Konsep dasar pemrograman Visual Studio adalah pembuatan form dengan mengikuti aturan pemrograman *Property*, Metode dan *Event*. Hal ini berarti:

a. *Property*: Setiap komponen di dalam pemrograman Visual Studio dapat diatur propertinya sesuai dengan kebutuhan aplikasi.
b. Metode: Bahwa jalannya program dapat diatur sesuai aplikasi dengan menggunakan metode pemrograman yang diatur sebagai aksi dari setiap komponen. Metode merupakan tempat untuk mengekpresikan logika pemrograman dari pembuatan suatu program aplikasi.
c. *Event:* Setiap komponen dapat beraksi melalui *event*, seperti *event click* pada *command button* yang tertulis dalam layar *script Command1_Click*[5].

2.8 Tinjauan Literatur Steganografi

2.7.1 *LEAST SIGNIFICANT BIT*

Penelitian ini melakukan proses penyembunyian text ke dalam file citra menggunakan metode *Last Significant Bit* yang ditulis oleh Hidayat[7]. Metode steganografi berbasiskan *Least significant bit* (LSB) proses penyisipan bit-bit data ke dalam byte-byte RGB citra adalah dengan menggunakan teknik penyisipan pada LSB. LSB *(Least Significant Bit)* adalah bit yang mempunyai nilai paling rendah, atau bit yang berada pada posisi paling kanan. Penyisipan LSB dilakukan dengan memodifikasi bit terakhir dalam satu byte data. Bit yang diganti adalah

LSB karena perubahan pada LSB hanya menyebabkan perubahan nilai byte satu lebih tinggi atau satu lebih rendah.

2.7.2 *SEQUENTIAL DAN SPREADING*

Penelitian yang ditulis oleh I Nyoman[8]. Teknik steganografi pada penelitian ini diimplementasikan pada data citra dengan format JPEG menggunakan metode *sequential* dan *spreading*. Metode *sequential* melakukan penyisipan secara berurutan pada *koefisien* dari DCT (*Discrete Cosine Transformation)* yaitu mentransformasi data dari satu tempat (domain) ke tempat (domain) yang lain. Fungsi DCT yaitu mentransformasi data dari tempat spatial (*spatial domain*) ke tempat frekuensi (*frequency domain*), sedangkan metode *spreading* melakukan penyisipan secara acak berdasarkan proses *hasing* yang digunakan. Proses pengujian yang dilakukan terdiri dari perbandingan kapasitas perhitungan dengan kapasitas pengujian, perhitungan statistik *error measurement*, pengujian dengan metode MOS untuk mengukur kualitas data citra serta ketahanan teknik steganografi yang digunakan terhadap penyerangan yang dilakukan.

Hasil pengujian menunjukan bahwa teknik steganografi dengan transformasi DCT bisa menghasilkan data hiding dengan tingkat validitas mencapai 100% dengan catatan bahwa data citra memiliki kapasitas penyisipan yang memadai. Penyisipan data yang dilakukan tidak berpengaruh terlalu banyak pada kualitas data citra yang dihasilkan, serta nilai PSNR yang dimiliki data citra terstego lebih besar sama dengan 30 dB.