BAB II

TINJAUAN PUSTAKA

2.1 Keamanan Data atau Informasi

Masalah keamanan data merupakan salah satu aspek terpenting pada sebuah sistem informasi. Masalah keamanan data seringkali tidak begitu dipedulikan bahkan ditiadakan. Informasi menentukan hampir setiap elemen dalam kehidupan manusia. Hanya sedikit hal yang bisa dilakukan di zaman modern tanpa melibatkan pengumpulan, penukaran, pembuatan atau pengaksesan data atau informasi. Informasi saat ini sudah menjadi sebuah komoditi yang sangat penting, bahkan ada yang mengatakan bahwa kita sudah berada disebuah *information-based society*. Kemampuan untuk mengakses dan menyediakan informasi secara cepat dan akurat menjadi sangat esensial bagi sebuah organisasi, baik yang berupa organisasi komersial(perusahaan), perguruan tinggi, lembaga pemerintah maupun individual(pribadi). Hal tersbut dimungkinkan dengan perkembangan yang pesat dibidang teknologi computer dan telekomunikasi, yang juga disebut dengan *end user computing.* Hal yang mendorong *end user computing* adalah meningkatnya pengetahuan manusia tentang computer, banyaknya jasa informasi yang tersedia, perangkat keras yang murah dan banyaknya perangkat lunak yang tersedia di pasaran.

Kemajuan sistem informasi memiliki banyak keuntungan dalam kehidupan manusia. Akan tetapi, aspek negatifnya juga banyak terjadi, seperti kejahatan komputer yang meliputi pencurian, penipuan, pemerasan dan banyak lagi yang

lainnya. Jatunya informasi ke pihak lain dapat menimbulkan kerugian bagi pemilik informasi tersebut. Untuk itu diperlukan adanya suatu cara/teknik untuk mengamankan data atau informasi. Zaman sekarang ini banyak cara/teknik untuk mengamankan informasi diantaranya kriptografi dan steganografi. Cara tersebut yang akan dibahas dalam penulisan tugas akhir ini.

2.2 Kriptografi

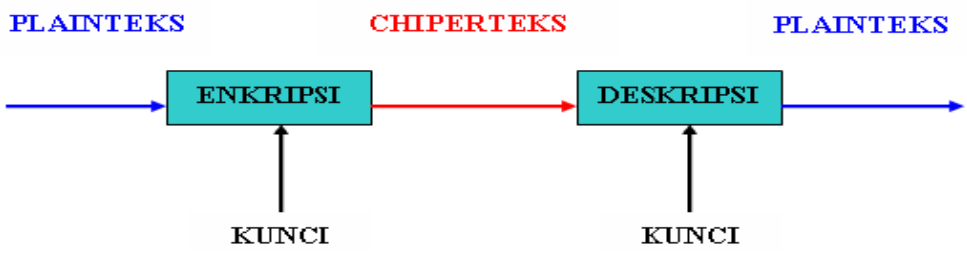
Kriptogafi berasal dari bahasa Yunani. Menurut bahasa tersebut kata kriptografi dibagi menjadi dua kata, yaitu *kripto* dan *graphia*. Kripto berarti *secret* (rahasia) dan *graphia* berarti *writing* (tulisan). Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan berita. Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data.

Didalam kriptografi kita akan menemukan berbagai istilah atau terminologi. Beberapa istilah yang penting untuk diketahui diantaranya :

1. ***Plaintext*** *(P)* adalah data atau informasi yang dapat dibaca dan dimengerti maknanya.
2. ***Ciphertext*** *(C)* adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
3. **Enkripsi** (fungsi E) adalah proses pengubahan *plaintext* menjadi *ciphertext.*
4. **Dekripsi** (fungsi D) adalah kebalikan dari enkripsi yakni mengubah *ciphertex*t menjadi *plaintext* sehingga berupa data awal/asli.

5. **Kunci** (K) adalah parameter yang digunakan untuk transformasi *enchipering* dan *dechipering*. Kunci biasanya berupa *string* atau deretan bilangan.

Secara umum, kriptografi terdiri dari dua buah bagian utama yaitu bagian enkripsi dan bagian dekripsi. Enkripsi adalah proses transformasi informasi menjadi bentuk lain sehingga isi pesan yang sebenarnya tidak dapat dipahami, hal ini dimaksudkan agar informasi tetap terlindung dari pihak yang tidak berhak menerima. Sedangkan dekripsi adalah proses kebalikan enkripsi, yaitu transformasi data terenkripsi ke data bentuk semula. Proses transformasi dari plainteks menjadi cipherteks akan dikontrol oleh kunci. Peran kunci sangatlah penting, kunci bersama-sama dengan algoritma matematisnya akan memproses plainteks menjadi cipherteks dan sebaliknya. Adapun blok proses enkripsi-dekripsi secara umum dapat kita lihat pada gambar di bawah ini :



Gambar 2.1 Proses enkripsi dan dekripsi

Dengan menggunakan simbol P untuk *plaintext* C untuk *chipertext* E untuk enkripsi D untuk dekripsi dan K untuk *key*, maka fungsi matematik dari enkripsi dan dekripsi dapat dituliskan dengan persamaan sebagai berikut :

$EK(P) = C$ dan $DK(C) = P$ .....(2.1)

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi, yaitu :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun terutama yang menjaga otoritas atau kunci rahasia untuk membuka informasi yang telah disandikan.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data system harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan dan pensubtitusian data lain kedalam data sebenarnya.
3. Autentifikasi, adalah berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan system maupun kesatuan informasi itu sendiri.
4. Non-repudiasi, adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan atau yang membuat.

2.2.1 Algoritma Sandi

Algoritma sandi adalah algoritma yang berfungsi untuk melakukan tujuan kriptografi. Pada implementasinya sebuah algoritmas sandi harus memperhatikan kualitas layanan/*Quality of Service* dari keseluruhan sistem dimana dia diimplementasikan. Algoritma sandi yang handal adalah algoritma sandi yang kekuatannya terletak pada kunci, bukan pada kerahasiaan algoritma itu sendiri.

Teknik dan metode untuk menguji kehandalan algoritma sandi adalah kriptanalis. Secara umum berdasarkan kesamaan kuncinya, algoritma sandi dibedakan menjadi :

A. Algoritma Sandi Simetris

Disebut sebagai algoritma kunci simetris, karena dalam proses enkripsi dan dekripsinya menggunakan kunci yang sama. Algoritma enkripsi dan deskripsi bisa merupakan algoritma yang sudah umum diketahui, namun kunci yang dipakai harus terjaga kerahasiaanya, dan hanya diketahui oleh pihak pengirim dan penerima saja. Kunci ini disebut sebagai *private key*. Sebelum berkomunikasi kedua pihak harus bersepakat lebih dahulu tentang kunci yang dipergunakan. Pendistribusian kunci dari satu pihak ke pihak lainnya memerlukan suatu kanal tersendiri yang terjagaan kerahasiaannya. Algoritma kunci simetris memiliki beberapa kekurangan dan kelebihan yaitu :

1. Kekurangan
    a. Perlu adanya kesepakatan untuk jalur yang khusus untuk kunci, hal ini akan menimbulkan masalah yang baru karena tidak mudah u menentukan jalur yang aman untuk kunci, masalah ini sering disebut dengan *"Key Distribution Problem"*.
    b. Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
    c. Apabila kunci sampai hilang atau dapat ditebak maka kriptosistem ini tidak aman lagi.

2. Kelebihan
    a. Waktu proses untuk enkripsi dan dekripsi relatif cepat, hal ini disebabkan karena efisiensi yang terjadi pada pembangkit kunci.
    b. Karena cepatnya proses enkripsi dan dekripsi, maka algoritma ini dapat digunakan pada sistem secara real-time seperti saluran telepon digital.

Contoh kriptografi yang menggunakan algoritma kunci simetris diantaranya :

1. DES *(Data Encryption Standard)*
2. IDEA *(International Data Encryption Algorithm)*
3. *Twofish*
4. *Rijndael*

B. Algoritma Sandi Asimetris

Algoritma kunci asimetris disebut juga algoritma kunci publik. Disebut kunci publik karena kunci yang digunakan pada proses enkripsi dapat diketahui oleh orang banyak tanpa membahayakan kerahasiaan kunci dekripsi, sedangkan kunci yang digunakan untuk proses dekripsi hanya diketahui oleh pihak tertentu (penerima). Mengetahui kunci publik semata tidak cukup untuk menentukan kunci rahasia. Pasangan kunci publik dan kunci rahasia menentukan sepasang transformasi yang merupakan invers satu sama lain, namun tidak dapat diturunkan satu dari yang lain. Dalam sistem kriptografi kunci publik ini, proses enkripsi dan dekripsi menggunakan kunci yang berbeda, namun kedua kunci

tersebut memiliki hubungan matematis (karena itu disebut juga sistem asimetris). Algoritma kunci asimetris memiliki beberapa kekurangan dan kelebihan diantaranya :

1. Kekurangan
    a. Kecepatan yang lebih rendah bila dibandingkan dengan algoritma simetris.
    b. Untuk tingkat keamanan sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritma simetris.
2. Kelebihan
    a. Masalah keamanan pada distribusi kunci dapat diatasi.
    b. Manajemen kunci pada suatu sistem informasi dengan banyak pengguna menjadi lebih mudah, karena jumlah kunci yang digunakan lebih sedikit.

Contoh kriptografi yang menggunakan algoritma kunci asimetris diantaranya :

1. RSA *(Rivert Shamir Adelman)*
2. DSA *(Digital Signature Algorithm)*
3. DH *(Diffie Hellman)*

2.2.2 Algoritma RSA *(Rivert Shamir Adelman)*

RSA adalah salah satu contoh kriptografi yang menerapkan konsep *public key*. Saat ini RSA bisa dikatakan sebagai algoritma kunci publik paling populer.

Algoritma ini pertama kali dipublikasikan di tahun 1977 oleh Ron Rivest, Adi Shamir, dan Leonard Adleman dari *Massachusetts Institute of Technology (MIT).*

Nama RSA sendiri adalah singkatan dari nama belakang penemu algoritmanya. Clifford Cocks, seorang matematikawan Inggris sebenarnya juga telah mengembangkan algoritma yang hampir sama dengan RSA ini pada tahun 1973. Namun algoritma buatannya tidak begitu dikenal oleh publik, dan baru dipublikasi pada tahun 1997 karena merupakan proyek rahasia.

Walau begitu algoritma yang dikembangkan Rivest, Shamir, dan Adleman tidak berhubungan dengan pekerjaan Cocks. Pada algoritma RSA terdapat 3 langkah utama yaitu *key generation* (pembangkitan kunci), enkripsi, dan dekripsi.

Kunci pada RSA mencakup dua buah kunci, yaitu *public key* dan *private key*. *Public key* digunakan untuk melakukan enkripsi, dan dapat diketahui oleh orang lain. Sedangkan *private key* tetap dirahasiakan dan digunakan untuk melakukan dekripsi.

Dalam melakukan enkripsi dan dekripsi secara umum algoritma RSA memiliki besaran-besaran sebagai berikut :

1. p dan q, bilangan prima rahasia
2. $n = p \cdot q$, tidak rahasia
3. $\Phi(n) = (p - 1) \cdot (q - 1)$, rahasia
4. SK, kunci privat rahasia
5. PK, kunci publik tidak rahasia
6. M, pesan yang dienkripsi

Kunci publik PK dalam RSA merupakan pembangkitan bilangan secara acak pencarian bilangan yang relatif prima terhadap Φ(n). Sedangkan kunci privat SK dibangkitkan dengan menggunakan persamaan PK**.**SK = 1 (mod Φ(n)). Dalam enkripsi biasa pesan dienkripsi dengan kunci publik baru didekripsi dengan kunci privat. Namun pada praktek tanda tangan digital digunakan sebaliknya.

2.3 Steganografi

Steganografi adalah seni dan ilmu menulis atau menyembunyikan pesan tersembunyi dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Sebaliknya, kriptografi menyamarkan arti dari suatu pesan, tapi tidak menyembunyikan bahwa ada suatu pesan. Kata steganografi (*steganography*) berasal dari bahasa Yunani yaitu *steganos*, yang artinya "tersembunyi atau terselubung" dan *graphein* yang artinya "menulis". Steganografi adalah seni dan ilmu menulis atau menyembunyikan pesan dengan suatu cara sehingga selain si pengirim dan si penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa ada suatu pesan rahasia. Sebaliknya, kriptografi menyamarkan arti dari suatu pesan, tapi tidak menyembunyikan pesan tersebut. Pada metode steganografi cara ini sangat berguna jika digunakan pada cara steganografi komputer karena banyak format file digital yang dapat dijadikan media untuk menampung/menyembunyikan pesan. Format yang biasa digunakan diantaranya:

1. Format image : bitmap (bmp), gif, jpeg.
2. Format audio : wav, voc, mp3.

3. Format lain : teks file, html, pdf.

Kini, istilah steganografi termasuk penyembunyian data digital dalam file-file komputer. Contohnya, si pengirim mulai dengan file gambar biasa, lalu mengatur warna setiap pixel ke-100 untuk menyesuaikan suatu huruf dalam alphabet (perubahannya begitu halus sehingga tidak ada seorangpun yang menyadarinya jika tidak benar-benar memperhatikannya).

Tujuan dari steganografi adalah menyembunyikan data/pesan pada suatu media. Media penampung data/pesan yang akan disembunyikan dapat berupa gambar digital, suara, video dan media lainnya. Jika pada media yang telah disisipi pesan rahasia tersebut terlihat mencurigakan, maka tujuan dari steganografi tersebut tidak tercapai.

Teknik steganografi sudah dikenal sejak jaman Yunani dan Romawi kuno. Misalnya dengan mencukur kepala budak, lalu pesan rahasia ditulis pada kulit kepalanya. Setelah rambut budak tersebut tumbuh, budak terebut dikirim untuk menyampaikan pesan rahasia tersebut.

Ada beberapa kriteria yang harus diperhatikan dalam penyembunyian data, yaitu :

1. Fidelity

   Setelah disisipi dengan data, media penampung harus terlihat baik. Pengamat tidak mengetahui terdapatnya data rahasia pada media penampung.

2. Recovery

   Data yang disembunyikan harus dapat diekstrasi kembali.

Untuk format atau media yang berbeda, metode yang digunakan untuk menyembunyikan pesannya juga berbeda-beda. Beberapa metode yang biasanya digunakan, yaitu:

1. Least Significant Bit (LSB)

Dasar dari metode ini adalah pengetahuan akan bilangan biner atau bilangan basis 2, yang hanya terdiri dari ’1’ dan ’0’. Kedua bilangan yang menjadi dasar dari kerja komputer ini sering disebut dengan istilah *bit*. Susunan dari beberapa *bit* akan membentuk suatu informasi. Istilah yang umum dikenal adalah *byte*, yaitu kumpulan delapan *bit* data.

Dalam satu *byte* data, *bit* yang paling berpengaruh terhadap informasi yang dikandungnya biasanya adalah *bit* paling awal/paling kiri. *Bit* inilah yang dinamakan *Most Significant Bit (MSB).* Semakin ke kanan, *bit-bit* tersebut semakin kecil pengaruhnya terhadap keutuhan data yang dikandung. *Bit* paling akhir/paling kanan inilah yang dinamakan *Least Significant Bit (LSB)*. Contohnya pada file *image* pesan dapat disembunyikan dengan menggunakan cara menyisipkannya pada *bit* rendah atau *bit* yang paling kanan (LSB) pada data *pixel* (titik) yang menyusun file tersebut. Seperti kita ketahui untuk file bitmap 24 *bit* maka setiap *pixel* (titik) pada gambar tersebut terdiri dari susunan tiga warna merah, hijau dan biru (RGB) yang masing-masing disusun oleh bilangan 8 *bit* (*byte*) dari 0 sampai 255 atau dengan format biner 00000000 sampai 11111111. Dengan demikian pada setiap *pixel* file bitmap 24 *bit* kita dapat menyisipkan 3 *bit* data.

A. Contoh 8 *bit* pixel :

1 pixel :	(00	01	10	11)
	white	red	green	blue
Insert 0011 :	(00	**0<u>0</u>**	**1<u>1</u>**	11)
	white	white	blue	blue

B. Contoh 24 *bit* pixel :

Contohnya huruf A dapat kita sisipkan dalam 3 pixel, misalnya data raster original adalah sebagai berikut :

(00100111	11101001	11001000	)
red	blue	green	
(00100111	11001000	11101001	)
red	green	blue	
(11001000	00100111	11101001	)
green	red	blue	

Sedangkan representasi biner huruf A adalah 100000111. Dengan menyisipkan-nya pada data *pixel* diatas maka akan dihasilkan :

(00100111	**1110100<u>0</u>**	11001000	)
red	green	green	
(**0010011<u>0</u>**	11001000	**1110100<u>0</u>**	)
white	green	green	
(**1100100<u>1</u>**	00100111	11101001	)
blue	red	blue	

Terlihat hanya empat *bit* rendah yang berubah, untuk mata manusia maka tidak akan tampak perubahannya. Secara rata-rata dengan metoda ini hanya setengah dari data *bit* rendah yang berubah, sehingga bila dibutuhkan dapat digunakan *bit* rendah kedua bahkan ketiga.

Colormap (peta warna) dalam 8 *bit* warna *image* (gambar) maksimumnya 24 *bit* warna *image* (gambar) dari 256 *color* (warna). Bagaimanapun untuk meminimisasi gangguan tambahan ketika *least significant bit (LSB)* berubah, *colormap* (peta warna) dimulai dari hanya 240 *color* (warna) dan ke-16 *color* (warna) lainnya akan ada atau ditambah pada saat hasil akhir dari sebuah gambar.



Gambar 2.2 Gambar dengan 15x16 grid

Kekurangan dari LSB *insertion* yaitu :

Dapat diambil kesimpulan dari contoh 8 *bit* pixel, menggunakan LSB *Insertion* dapat secara drastis merubah unsur pokok warna dari pixel. Ini dapat menunjukkan perbedaan yang nyata dari *cover image* menjadi *stego image*, sehingga tanda tersebut menunjukkan keadaan dari steganografi. Variasi warna kurang jelas dengan 24 *bit image*, bagaimanapun file tersebut sangatlah besar.

Antara 8 *bit* dan 24 *bit image* mudah diserang dalam pemrosesan *image*, seperti *cropping* (kegagalan) dan *compression* (pemampatan).

2. Algoritma dan Transformasi

Metode Steganografi yang lain adalah menyembunyikan data dalam fungsi matematika yang disebut algoritma compression. Dua fungsi tersebut adalah *Discrete Cosine Transformation (DCT)* dan *Wavelet Transformation*. Fungsi DCT dan *Wavelet* 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*).

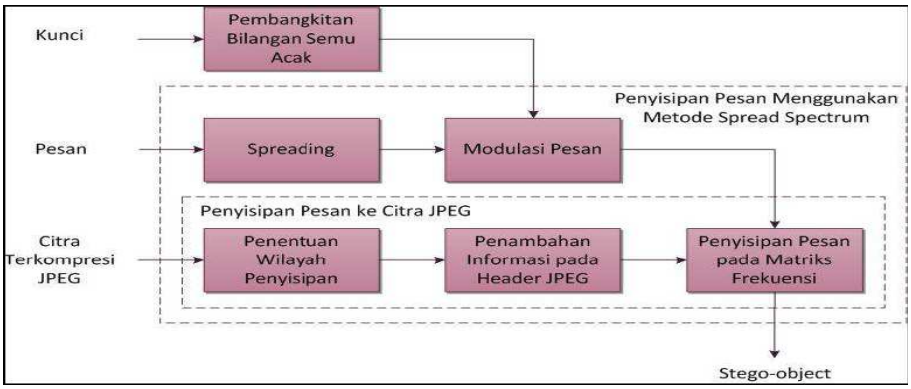
3. Redundant Pattern Encoding

Penerapan steganografi dengan metode ini adalah dengan menggambar pesan kecil pada kebanyakan gambar. Keuntungan dari metode ini adalah dapat bertahan dari *cropping*, kerugiannya yaitu tidak dapat menggambar pesan yang lebih besar, sehingga pesan menjadi agak terbatas dan mungkin saja pesan yang ingin disampaikan oleh pengirim bahkan terlewatkan oleh penerima.

4. Spread Spectrum

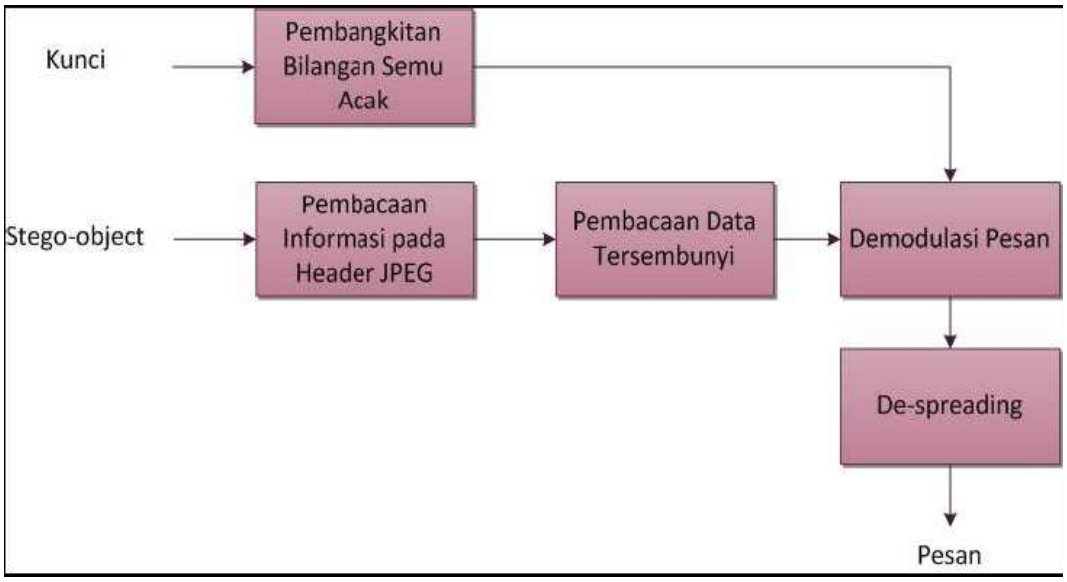
Metoda *Spread Spectrum* mentransmisikan sebuah sinyal pita informasi yang sempit ke dalam sebuah kanal pita lebar dengan penyebaran frekuensi. Penyebaran ini berguna untuk menambah tingkat redundansi. Besaran redundansi ditentukan oleh faktor pengali *cr* yang bernilai skalar. Panjang *bit- bit* hasil penyebaran ini menjadi *cr* kali panjang *bit-bit* awal.

Proses penyisipan pesan menggunakan metode *Spread Spectrum* ini terdiri dari tiga proses, yaitu *spreading*, modulasi, dan penyisipan pesan ke citra.Untuk lebih jelasnya dapat dilihat dari gambar skema penyisipan pesan dibawah ini :



Gambar 2.3 Skema Penyisipan Pesan Spread Spectrum

Proses ekstraksi pesan menggunakan metode *Spread Spectrum* ini terdiri dari tiga proses, yaitu pengambilan pesan dari matriks frekuensi, demodulasi, dan de- *spreading*. Untuk lebih jelasnya dapat dilihat di gambar skema ekstraksi pesan dibawah ini :

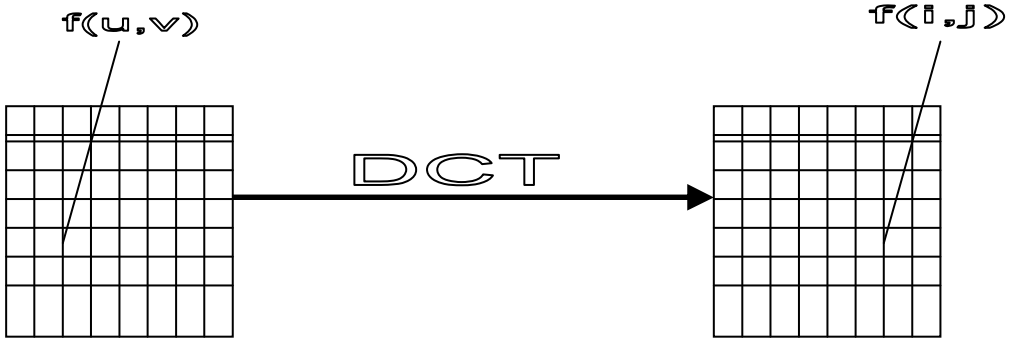


Gambar 2.4 Skema Ekstraksi Pesan Spread Spectrum

Namun yang akan di bahas dalam penulisan tugas akhir ini hanya dua metoda steganografi yaitu *Discrete Cosine Transformation (DCT)* dan *Redundant Pattern Encoding.*

2.3.1 Discrete Cosine Transformation (DCT)

Metode Steganography yang lain adalah menyembunyikan data dalam fungsi matematika yang disebut algoritma compression. Dua fungsi tersebut adalah Discrete Cosine Transformation (DCT) dan Wavelet Transformation. Fungsi DCT dan Wavelet yaitu mentransformasi data dari satu tempat (domain) ke tempat (domain) yang lain. Fungsi DCT yaitu suatu teknik yang digunakan untuk melakukan konversi sinyal kedalam komponen frekuensi pembentuknya dengan memperhitungkan nilai real dari hasil transformasinya.



Gambar 2.5 Diagram Transformasi DCT

Fungsi Transformasi DCT :

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(i,j) \cos\left(\frac{\pi.u}{2N}.(2i+1)\right) \cos\left(\frac{\pi.v}{2M}.(2j+1)\right) \quad .....(2.2)$$

Fungsi *Invers* DCT :

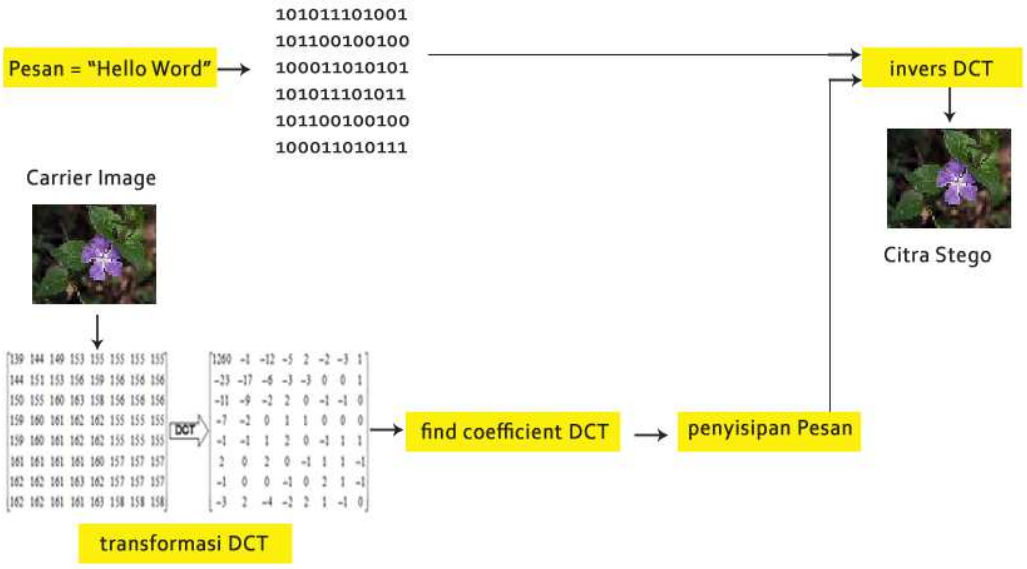
$$f(i,j) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} f(u,v) \cos\left(\frac{\pi(2i+1)u}{2N}\right) \cos\left(\frac{\pi(2j+1)v}{2M}\right) \quad \text{.....(2.3)}$$

Keterangan :

Ukuran image M x N

f(u,v) : nilai koefisien DCT pada matrik 8x8 kolom ke-u baris ke-v

F(i,j) : nilai data yang hendak ditransformasikan pada matrik 8x8 kolom ke-i baris ke-j

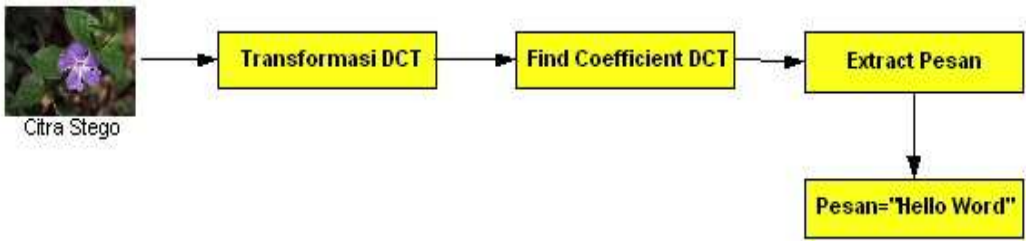


Gambar 2.6 Proses Penyisipan Pesan DCT

Pada proses penyisipan pesan terdapat tugas-tugas seperti berikut :

1. Lakukan Transformasi DCT dari citra digital.
2. Dapatkan *coefficient DCT*.

3. Gantikan *bit* pada citra tersebut dengan *bit* pesan yang akan dimasukan agar tidak terlihat (dapat digunakan LSB *insertion).*
4. Lakukan transformasi *invers dct.*
5. Simpan sebagai citra tersisip pesan.



Gambar 2.7 Proses pembacaan pesan DCT

Pada proses pembacaan pesan tugas-tugas yang perlu dilakukan diantaranya :

1. Lakukan transformasi dct dari citra tersisip pesan.
2. Dapatkan *coefficient DCT.*
3. Extract *bit* data dari koefisien terebut.
4. Gabungkan *bit-bit* tersebut menjadi sebuah pesan aktual

2.3.2 Redundant Pattern Encoding

Penerapan steganografi dengan metode ini adalah dengan menggambarkan pesan kecil pada kebanyakan gambar. Algoritma dari *redundant pattern encoding* yaitu memasukkan redundansi (penggandaan) ke dalam informasi/pesan yang akan disembunyikan (dengan catatan pesan hasil penggandaan sama dengan aslinya/utuh) dan kemudian menyebarkan pesan itu ke keseluruhan gambar. Sebuah generator yang bekerja secara *pseudorandom* (suatu

algoritma untuk menghasilkan urutan angka yang mendekati sifat nomor acak) digunakan untuk menyeleksi dua area dari gambar (*patch A* dan *patch B*). *Patch* adalah metode yang menandai area gambar.

Keseluruhan *pixel* pada *patch A* akan ditinggikan tingkat cahayanya. Lain halnya dengan *patch B,* yang justru diturunkan tingkat cahayanya (digelapkan). Dengan kata lain, intensitas pada *pixel* di suatu *patch* dinaikkan dengan nilai yang konstan, sementara *patch* lainnya diturunkan dengan nilai konstan yang sama. Perubahan kontras pada bagian patch akan mengenkripsi tiap satu *bit* dan perubahannya biasanya sangat kecil dan halus.

Kerugian dari teknik ini adalah hanya satu *bit* yang dapat dimasukkan pesan rahasia. Suatu gambar dapat digunakan untuk dimasukkan lebih dari satu *bit* dengan cara membagi-bagi gambar tersebut ke sub- gambar dan menggunakan cara yang sama untuk memasukkan pesan ke dalamnya. Keuntungan untuk menggunakan teknik ini sendiri adalah pesan rahasianya akan disebarkan ke gambar tersebut secara keseluruhan. Sehingga, apabila ada sebuah patch yang dihancurkan, yang lain masih tetap bertahan. Namun, tetap saja hal ini bergantung kepada ukuran pesan itu sendiri. Ini disebabkan pesan tersebut hanya bias diulang dan tersebar ke seluruh bagian gambar apabila ukurannya cukup dan kecil.

Cara ini digunakan tidak bergantung terhadap jenis gambarnya (transfer domain) dan terbukti sebagai cara yang cukup ampuh dan baik untuk memasukkan pesan.

2.4 Citra Digital

Secara harafiah, citra (*image*) adalah gambar pada bidang dua dimensi (*dwimatra)*. Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam.

2.4.1 *Bitmap*

Bitmap adalah representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan di memori komputer. Dikembangkan oleh Microsoft dan nilai setiap titik diawali oleh satu *bit* data untuk gambar hitam putih, atau lebih bagi gambar berwarna. Kerapatan titik-titik tersebut dinamakan resolusi, yang menunjukkan seberapa tajam gambar ini ditampilkan, ditunjukkan dengan jumlah baris dan kolom, contohnya 1024x768. Kelemahan dari citra bitmap adalah tidak dapat memperbesar atau memperkecil ukuran resolusinya. Resolusinya bergantung pada citra aslinya. Ukuran filenya juga relative besar, karena tersimpan dalam pixel. Selain itu citra bitmap apabila dilakukan proses kompresi kualitas citranya akan mengalami penurunan. Yang paling penting dari kriteria ini adalah kedalaman warna (berapa banyak *bit* per pixel yang didefinisikan dari sebuah warna). Bitmap dengan mengikuti kriteria tadi maka dapat dilihat :

1. *bit* = 16 warna (16 gray scales).
2. 8 *bit* = 256 warna (256 gray scales).
3. 24 *bit* = 16.777.216 warna.

Secara umum dapat dikatakan : Semakin banyaknya warna, maka akan diperlukan keamanan yang ketat atau tinggi dikarenakan bitmap memiliki area yang sangat luas dalam sebuah warna yang seharusnya dihindarkan. Dilihat dari kedalaman atau kejelasan dari sebuah warna, bitmap dapat mengambil sejumlah data tersembunyi dengan perbandingan sebagai berikut (ukuran ratio dari bitmap dalam *byte* = ukuran dari data yang disembunyikan) :

1. 4 *bit* = 16 warna : 4 : 1
2. 8 *bit* = 256 warna : 8 : 1
3. 24 *bit* = 16.777.216 warna : 8 : 1

Untuk menampilkan citra *bitmap* pada monitor atau mencetaknya pada printer, komputer menterjemahkan *bitmap* ini menjadi pixel (pada layar) atau titik tinta (pada printer). Beberapa format file bitmap yang populer adalah BMP, PCX dan TIFF. Contoh gambar bitmap dapat dilihat di bawah ini :



Gambar 2.8 Gambar berformat BMP atau bitmap

2.4.2 *Joint Photographic Group Experts (JPEG)*

Adalah standar kompresi file yang dikembangkan oleh *Group Joint Photographic Experts* menggunakan kombinasi DCT dan pengkodean *Huffman* untuk mengkompresikan suatu file citra. JPEG adalah suatu algoritma kompresi yang bersifat *lossy*, (yang berarti kualitas citranya agak kurang bagus). JPEG adalah teknik kompresi grafis *high color bit-mapped*. Merupakan teknik dan standar *universal* untuk kompresi dan dekompresi citra tidak bergerak untuk digunakan pada kamera digital dan sistem pencitraan menggunakan komputer yang dikembangkan oleh *Joint Photographic Experts Group.*

Umumnya digunakan untuk kompresi citra berwarna maupun *gray scale*. Format ini benar-benar didesain memanjakan fotographer. Format ini bisa mensupport sampai 16.7 juta warna. Jumlah tersebut cukup untuk keperluan apapun bahkan pencitraan warna yg tidak bisa dicerna mata manusia. Selain jumlah kombinasi warna, algoritma kompresi JPEG bekerja dengan meresonansi informasi-informasi image keluar. Dan tergantung setting yg diberikan, informasi tersebut bisa dan bisa tidak dapat dicerna mata. Contoh gambar dengan format JPEG dapat dilihat di bawah ini :



Gambar 2.9 Gambar berformat JPEG

2.4.3 *Graphics Interchange Format (GIF)*

Formst GIF itu baik digunakan untuk website. Kombinasi warna yg tersedia sebanyak 256 warna. Jumlah kombinasi ini cukup membuatnya dipakai bagi keperluan grafis apapun, tentu saja degan pengecualian keperluan photografi. Kita bisa memakainya sebagai *icon, favicon, logo, line grafis,* ataupun *image-image* tombol. GIF juga bisa dianimasikan. Beberapa karakteristik format gambar GIF sebagai berikut :

1. Mampu menayangkan maksimum sebanyak 256 warna karena format GIF menggunakan 8-*bit* untuk setiap pixelnya.
2. Mengkompresi gambar dengan sifat *lossless*
3. Mendukung warna transparan dan animasi sederhana

Sebenarnya tersedia pilihan format lain seperti flash atau format animasi berbasis vektor. Akan tetapi format-format itu biasanya lebih *konsumtif-bandwidth.* Gif mempunyai kemampuan mengkompres area-area gambar sewarna. Dengan kemampuan ini, banyak item-item grafis website bisa didesain dgn ukuran sekecil mungkin. Metode yang paling umum mereduksi ukuran file GIF ialah dengan mereduksi jumlah warna dalam *palettle*. Perlu digaris-bawahi, GIF telah menggunakan skema kompresi internal LZW. Skema ini bisa membuat gambar berukuran sekecil mungkin tanpa kehilangan data penting apapun. Namun tidak tau pasti bagaimana algoritma LZW ini bekerja. Ada 2 *feature* Gif yg menarik perhatian, yaitu *transparency* dan *interlacing*. Contoh gambar yang memiliki format GIF dapat dilihat dibawah ini :



Gambar 2.10 Gambar berformat GIF

2.4.4 *Portable Network Graphics (PNG)*

PNG (*Portable Network Graphics*) adalah salah satu format penyimpanan citra yang menggunakan metode pemadatan yang tidak menghilangkan bagian dari citra tersebut (*lossless compression*).

PNG dibaca "ping", namun biasanya dieja apa adanya - untuk menghindari kerancuan dengan istilah "ping" pada jaringan komputer. Format PNG ini diperkenalkan untuk menggantikan format penyimpanan citra GIF. Secara umum PNG dipakai untuk Citra Web. Untuk Web, format PNG mempunyai 3 keuntungan dibandingkan format GIF diantaranya :

1. *Channel Alpha* (transparansi)
2. *Gamma* (pengaturan terang-gelapnya citra en:"*brightness*")
3. Penayangan citra secara progresif (*progressive display*)

Selain itu, citra dengan format PNG mempunyai faktor kompresi yang lebih baik dibandingkan dengan GIF (5% - 25% lebih baik dibanding format GIF). Satu fasilitas dari GIF yang tidak terdapat pada PNG format adalah dukungan terhadap penyimpanan multi-citra untuk keperluan animasi. Untuk keperluan pengolahan citra, meskipun format PNG bisa dijadikan alternatif

selama proses pengolahan citra - karena format ini selain tidak menghilangkan bagian dari citra yang sedang diolah (sehingga penyimpanan berulang ulang dari citra tidak akan menurunkan kualitas citra) namun format JPEG masih menjadi pilihan yang lebih baik. Contoh gambar yang memiliki format PNG dapat dilihat dibawah ini :



Gambar 2.11 Gambar berformat PNG

2.5 Pengukuran *Error* Citra

MSE (Mean Square Error) adalah tingkat kesalahan sinyal-sinyal video atau piksel-piksel citra hasil pemrosesan sinyal terhadap sinyal/citra original. Rumus untuk menghitung *MSE* pada citra digital adalah :

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}[I(x,y) - I'(x,y)]^2 \quad \text{.....(2.4)}$$

Sedangkan *Peak Signal to Noise Ratio (PSNR)* merupakan nilai (rasio) yang menunjukan tingkat toleransi *noise* tertentu terhadap banyaknya *noise* pada suatu sinyal video/citra. *Noise* adalah kerusakan sinyal pada bagian tertentu dalam sebuah video/citra sehingga mengurangi kualitas sinyal tersebut. Dengan kata lain

PSNR merupakan suatu nilai yang menunjukkan kualitas suatu sinyal video/citra. Untuk menghitung nilai *PSNR* digunakan rumus sebagai berikut :

$$PSNR = 10 \log_{10}\left(\frac{255^2}{MSE}\right) \quad \text{.....(2.5)}$$

Dimana : I (x,y) nilai *pixel* dari citra asli

I’ (x,y) nilai *pixel* dari hasil steganografi

M, N adalah dimensi *image*

2.6 .NET Platform

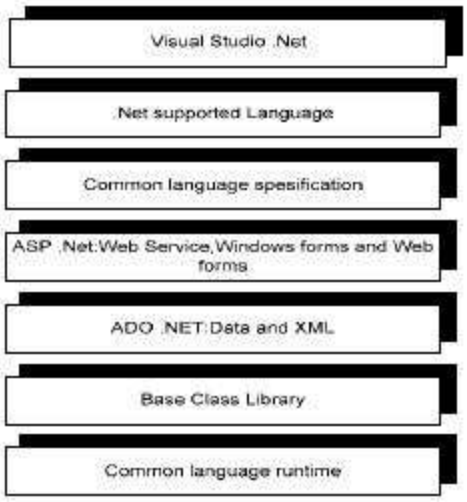
.NET merupakan sebuah platform yang dibangun oleh microsoft dengan komponen utamanya yaitu Framework .NET. Framework .NET merupakan suatu komponen window yang terintegrasi dan dibuat agar dapat menjalankan berbagai macam aplikasi berbasis .NET termasuk pembangunan aplikasi *Web Service* (XML).

2.7 *.NET Framework*

.NET framework adalah sebuah framework atau bingkai kerja bagi aplikasi-aplikasi .NET yang dibuat menggunakan Visual Studio .NET tanpa terpengaruh bahasa pemrograman yang digunakan.

.NET framework menyediakan berbagai macam *tool* dan teknologi yang diperlukan untuk membangun aplikasi yang terdistribusi. Penyediaan *tool* dan teknologi akan menjadikan sesuatu yang konsisten dan bahasa pemrograman yang

bebas ke semua model ditingkatan sama suatu aplikasi, lalu menyediakan interoperabilitas tanpa klaim dan memudahkan migrasi dari teknologi yang ada pada Microsoft .NET. Arsitektur dari Framework .NET seperti gambar di bawah ini :



Gambar 2.12 Arsitektur Framework .NET

Framework .NET terdiri atas dua elemen penting, yaitu *Common Language Runtime* (CLR) dan *Framework Class Library* (FCL). CLR merupakan sebuah lingkungan untuk menjalankan program pada saat runtime, kemudian FCL menyediakan layanan yang dibutuhkan oleh aplikasi .NET.

2.7.1 *Common Language Runtime (CLR)*

CLR adalah tulang punggung .NET framework, pada CLR inilah tersedia *library-library* yang diperlukan oleh aplikasi-aplikasi yang dibuat. Program aplikasi yang dijalankan tidak langsung berhubungan dengan sistem operasi, namun melalui CLR terlebih dahulu.

Tugas dari CLR adalah sebagai berikut :

1. Mengelola dan mengeksekusi kode program
2. Mengelola alokasi memori ketika program di eksekusi
3. Memberikan layanan seperti JIT kompilasi
4. Mengelola penanganan kesalahan atau eksepsi
5. Melakukan proses *debug* dan mengatur keamanan program
6. Bersifat *assembly* saat proses *deployment* (penyebaran) atau kompilasi
7. Memungkinkan adanya variasi dan integrasi berbagai bahasa pemrograman yang ada dilingkungan Framework .NET

2.7.2 *Framework Class Library (FCL)*

FCL adalah sebuah kumpulan *class-class* yang berjumlah ribuan di dalam .NET framework. *Class-class* ini menyediakan akses ke window API dan fungsi-fungsi yang umum seperti struktur data, manipulasi string, IO, *stream, theread, security, network programming, windows programming, web programming*, data akses, dan lain-lain. Ini adalah *library* standar yang pernah disertakan pada programming *enviorenment* atau pada bahasa pemrograman tertentu. Penggunaan *class-class* ini sangat mudah, bahkan *inheritance* dan *polymorphisme* dapat diterapkan pada *class-class* ini.

2.7.3 C#

C# merupakan bahasa pemrograman yang didesain dengan target diimplementasikan dalam teknologi .NET. Pada tahun 2002 tepatnya bulan

Januari C# maupun framework .NET diselesaikan dan dapat diimplementasikan. Bahasa pemrograman C# merupakan bahasa baru yang sangat handal dan konsisten serta membawa kesan bahwa C# merupakan bahasa pemrograman yang modern. Bahasa C# merupakan bahasa komposisi dari berbagai bahasa diantaranya bahasa java, C++, Visual Basic dan lain-lain. Bahasa java merupakan bahasa yang paling banyak digunakan dalam sintaksis C#. Persentase penggunaan bahasa java dalam bahasa C# adalah sebanyak 70%, bahasa C++ sebanyak 10%, bahasa Visual Basic sebanyak 5% dan bahasa lain-lain sebanyak 15%.

2.8 Model Pengembangan Perangkat Lunak

Perangkat lunak *(software)* adalah perintah/program komputer yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional dan dokumen yang menggambarkan operasi dan kegunaan program.

Software memiliki dua peran, di satu sisi berfungsi sebagai sebuah produk dan di sisi lain berfungsi sebagai kendaran yang mengantarkan sebuah produk. Sebagai produk, *software* mengantarkan potensi perhitungan yang dibangun oleh *software* komputer. Baik di dalam sebuah telepon seluler, atau beroperasi di sebuah mainframe komputer, *software* merupakan transformer informasi yang memproduksi, mengatur, memperoleh, memodifikasi, menampilkan, atau memancarkan informasi, dimana pekerjaan ini dapat sesederhana suatu *bit* tunggal atau sekompleks sebuah simulasi multimedia. Sedangkan peran sebagai kendaran yang dipakai untuk mengantarkan produk, *software* berlaku sebagai dasar untuk

kontrol komputer (sistem operasi), komunikasi informasi (jaringan), dan penciptaan serta kontrol dari program – program lain (peranti dan lingkungan *software*).

Pada masa-masa awal, pemrograman masih dilihat sebagai “bentuk kesenian”. Hanya sedikit saja metode yang ada dan lebih sedikit lagi orang yang memahaminya. Para *programmer* kadang – kadang harus mempelajarinya dengan coba-coba. Tetapi sekarang *software* sudah menjadi lahan yang sangat kompetitif. *Software* yang dulu dibangun secara internal di dalam komputer sekarang sudah dapat diproduksi secara terpisah.

Software lebih merupakan elemen logika dan bukan merupakan elemen sistem fisik. *Software* memiliki ciri yang berbeda dari *Hardware*, yaitu :

1. *Software* dibangun dan dikembangkan.

2. *Software* tidak pernah usang.

3. Sebagian besar *Software* dibuat secara *custom-built*, serta tidak dapat dirakit dari komponen yang sudah ada.

Karakteristik dari produk *software* :

1. Maintainanbility

2. Dependability

3. Efficiency

4. Usability

Proses (produksi) software, yaitu:

1. Spesifikasi *software*

2. Pengembangan *software*

3. Validasi (pengetesan dan pengujian)

4. Evolusi, pengembangan lanjutan

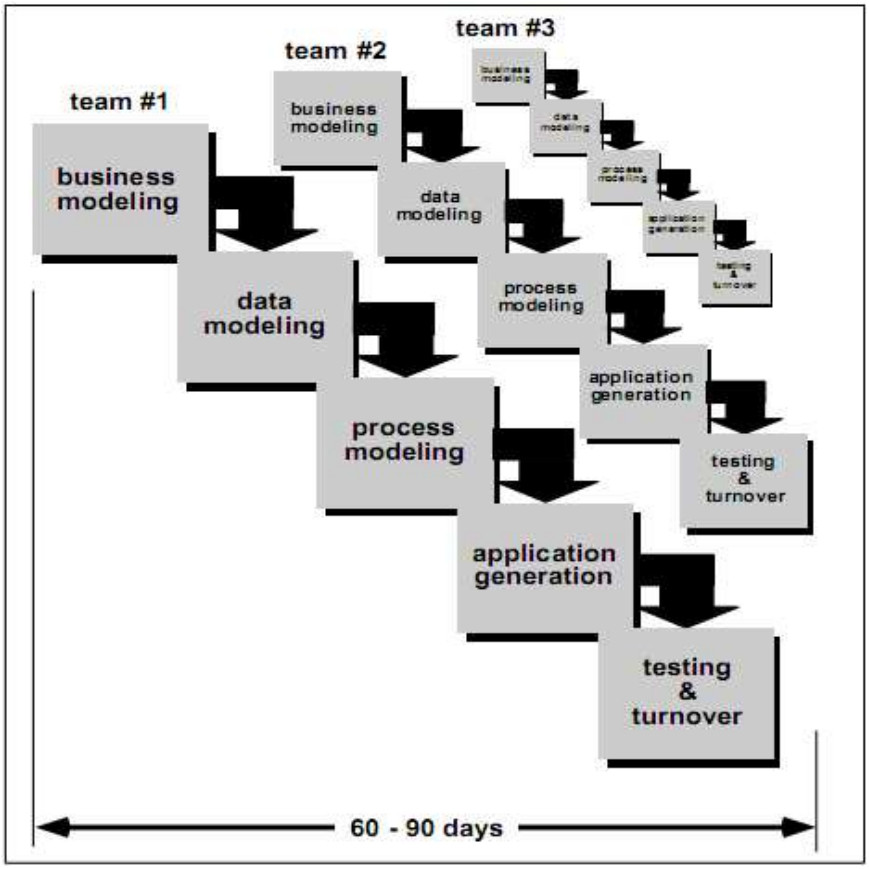
IEEE telah mengembangkan definisi yang lebih komprehensif, yaitu *software engineering* : aplikasi dari sebuah pendekatn kuantifiabel, disiplin, dan sistematis kepada pengembangan, operasi, dan pemeliharaan *software*.

Model–model proses untuk *software engineering* yang sering digunakan seperti model sekuensial linier, model prototipe, model RAD, model inkremental, model spiral, model *asembly* komponen, model pengembangan kongkuren, model metode formal, model teknik generasi keempat.

Model pengembangan perangkat lunak yang akan dibahas dalam penulisan tugas akhir ini hanyalah model RAD *(Rapid Application Development).*

2.8.1 RAD (Rapid Application Development)

Rapid Aplication Development (RAD) adalah sebuah model proses pengembangan perangkat lunak yang incremental. RAD menekankan pada siklus pengembangan yang pendek/singkat. RAD mengadopsi model waterfall dan pembangunan dalam waktu singkat dicapai dengan menerapkan *component based construction*. Waktu yang singkat adalah batasan yang penting untuk model ini. Jika kebutuhan lengkap dan jelas maka waktu yang dibutuhkan untuk menyelesaikan secara komplit *software* yang dibuat adalah misalnya 60 sampai 90 hari.



Gambar 2.13 Model RAD

Fase-fase di atas menggambarkan proses dalam model RAD. Sistem dibagi-bagi menjadi beberapa modul dan dikerjakan dalam waktu yang hampir bersamaan dalam batasan waktu yang sudah ditentukan.

1. *Bussiness Modeling*

Aliran informasi di antara fungsi–fungsi bisnis dimodelkan dengan suatu cara untuk menjawab pertanyaan–pertanyaan berikut : Informasi apa yang mengendalikan proses bisnis? Informasi apa yang di munculkan? Siapa yang memunculkanya? Ke mana informasi itu pergi? Siapa yang memprosesnya?

2. *Data Modeling*

Aliran informasi yang didefinisikan sebagai bagian dari fase *business modelling* disaring ke dalam serangkaian objek data yang dibutuhkan untuk menopang bisnis tersebut. Karakteristik (disebut atribut) masing–masing objek diidentifikasi dan hubungan antara objek–objek tersebut didefinisikan.

3. *Prosess Modelling*

Aliran informasi yang didefinisikan di dalam fase data modeling ditransformasikan untuk mencapai aliran informasi yang perlu bagi implementasi sebuah fungsi bisnis. Gambaran pemrosesan diciptakan untuk menambah, memodifikasi, menghapus, atau mendapatkan kembali sebuah objek data.

4. *Aplication Generation*

RAD mengasumsikan pemakaian teknik generasi ke empat. Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang ada (pada saat memungkinkan) atau menciptakan komponen yang bisa dipakai lagi (bila perlu). Pada semua kasus, alat–alat bantu otomatis dipakai untuk memfasilitasi konstruksi perangkat lunak.

5. *Testing and Turnover*

Karena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji. Hal ini mengurangi keseluruhan waktu pengujian. Tetapi komponen baru harus di uji dan semua *interface* harus dilatih secara penuh.

Kelemahan dalam model ini:

1. Tidak cocok untuk proyek skala besar.

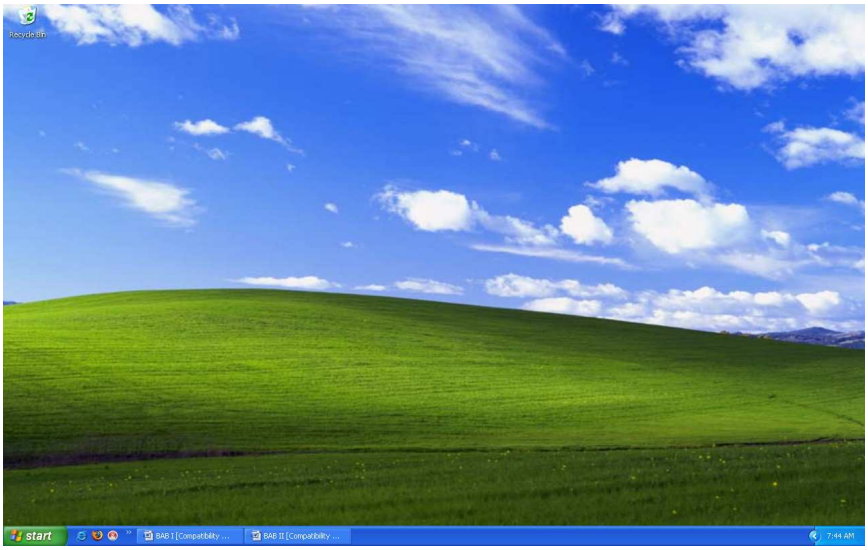
2. Proyek bisa gagal karena waktu yang disepakati tidak dipenuhi.
3. Sistem yang tidak bisa dimodularisasi tidak cocok untuk model ini.
4. Resiko teknis yang tinggi juga kurang cocok untuk model ini.

2.9 Tinjauan Perangkat Lunak

Pada bagian ini penulis akan menguraikan penjelasan singkat mengenai perangkat lunak (*software* utama) yang digunakan dalam pembuatan aplikasi ini yaitu Microsoft Windows XP SP 3 sebagai sistem operasi yang digunakan dan Microsoft Visual C# 2008 sebagai editor pembuatan aplikasi.

2.9.1 *Microsoft Windows XP*

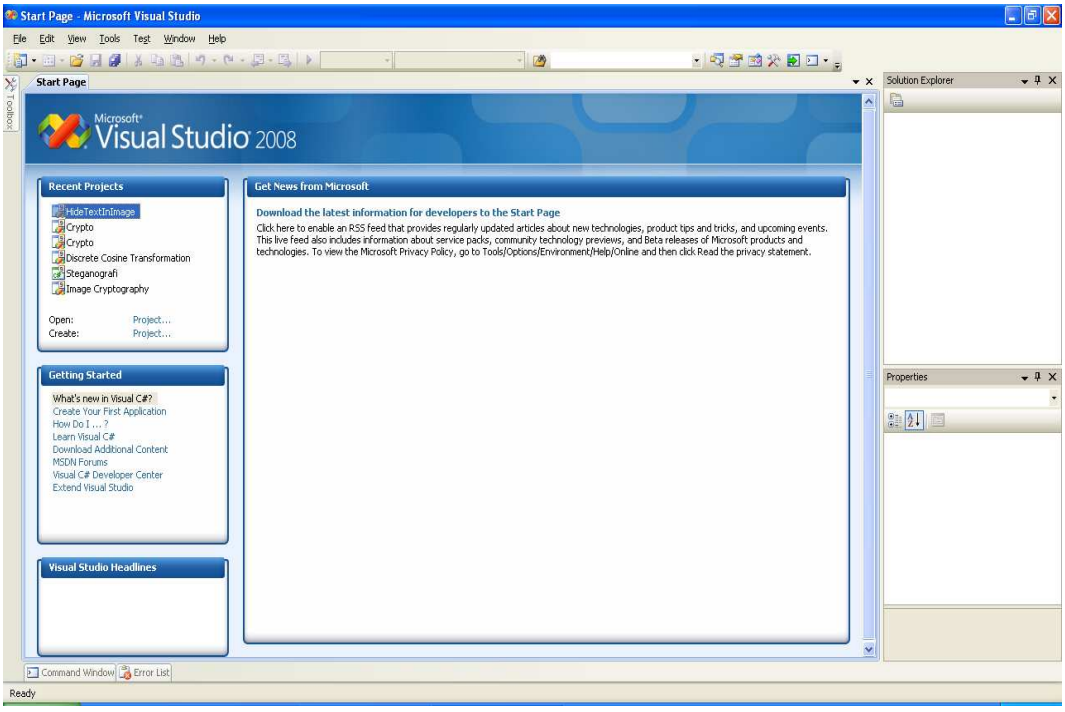
Windows adalah salah satu sistem operasi andalan Micosoft yang banyak dipakai saat ini, karena kestabilan dan kemudahan dalam pemakaiannya. Kelebihan liannya adalah kebutuhan perangkat kerasnya tidak membutuhkan sumber daya yang besar. Tampilan sistem operasi Windows XP adalah sebagai berikut :



Gambar 2.14 Microsoft Windows XP

2.9.2 *Microsoft Visual C# 2008*

C# sering dianggap sebagai penerus C++ atau versi canggih dari C++, karena ada anggapan bahwa tanda # adalah perpaduan dari 4 buah tanda tambah yang disusun sedemikian rupa sehingga membentuk tanda pagar. Akan tetapi, terlepas dari benar tidaknya anggapan tersebut C# adalah suatu bahasa pemrograman yang sangat menjanjikan. C# adalah sebuah bahasa pemrograman yang berorientasi pada objek yang dikembangkan oleh Microsoft dan menjadi salah satu bahasa pemrograman yang mendukung .NET *programming* melalui Visual Studio. Tampilan awal dari Micosoft Visual C# 2008 adalah sebagai berikut :



Gambar 2.15 Microsoft Visual C# 2008

2.9.3 Keuntungan Visual C#

Ada beberapa keuntungan menggunakan teknologi .NET dalam membangun sebuah aplikasi, baik aplikasi *web service* maupun aplikasi biasa yang sering digunakan seperti Window Form, diantaranya :

1. Mudah, Visual C# yang sangat sederhana mudah digunakan dalam mengimplementasikan pengembangan sebuah aplikasi
2. Effisien, kemudahan saat pembuatan aplikasi sangat berpengaruh pada efisiensi waktu pengerjaan aplikasi dan berdampak pula pada efisiensi biaya
3. Produktivitas, kemudahan pengerjaan aplikasi akan berdampak besar dalam menghasilkan produk yang berupa aplikasi.
4. Konsisten, kemudahan yang dihadirkan oleh Visual C# akan berdampak pada konsistensi program.