BAB II
TINJAUAN PUSTAKA

2.1 Steganografi

Steganografi (*steganography*) adalah ilmu dan seni menyembunyikan pesan rahasia (*hiding message*) sedemikian sehingga keberadaan (eksistensi) pesan tidak terdeteksi oleh indera manusia. Kata steganografi berasal dari Bahaya Yunani yang berarti "tulisan tersembunyi" (*covered writing*). Steganografi membutuhkan dua properti: wadah penampung dan data rahasia yang akan disembunyikan. Steganografi digital menggunakan media digital sebagai wadah penampung, misalnya citra, suara, teks, dan video. Data rahasia yang disembunyikan juga dapat berupa citra, suara, teks, atau video.

Steganografi dapat dipandang sebagai kelanjutan kriptografi. Jika pada kriptografi, data yang telah disandikan (*ciphertext*) tetap tersedia, maka dengan steganografi cipherteks dapat disembunyikan sehingga pihak ketiga tidak mengetahui keberadaannya. Di negara-negara yang melakukan penyensoran informasi, steganografi sering digunakan untuk menyembunyikan pesan-pesan melalui gambar (*images*), video, atau suara (*audio*) (Munir Rinaldi., 2004).

Kriteria steganografi yang baik meliputi tiga hal, yaitu :

1. *Imperceptible/Undetectability,* yaitu keberadaan pesan tidak dapat persepsi oleh indrawi. Jika pesan disisipkan ke dalam sebuah citra. Citra yang telah disisipi pesan harus tidak dapat dibedakan dengan citra asli oleh mata. Begitu pula dengan suara, telinga harus mendapati perbedaan antara suara asli dan suara yang telah disisipi pesan.
2. *Fidelity*, yaitu mutu *cover-object* tidak jauh berubah akibat *embedded*. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Pengamat tidak mengetahui kalau di dalam citra tersebut terdapat data rahasia.
3. *Recovery,* yaitu data yang disembunyikan harus dapat diungkapkan kembali. Tujuan steganogarfi adalah menyembunyikan informasi, maka sewaktu-waktu

informasi yang disembunyikan harus dapat diambil kembali untuk dapat digunakan lebih lanjut sesuai keperluan (Munir Rinaldi., 2004).

Terdapat banyak metode yang digunakan dalam melakukan penyembunyian data kedalam data lainnya. Berikut adalah penjelasan mengenai beberapa metode yang banyak digunakan dalam steganografi.

2.1.1 Metode Steganografi pada Teks

a. Metode Spasi Terbuka

Terdapat beberapa cara untuk memanfaatkan spasi terbuka dalam data text guna menyembunyikan informasi. Metode ini dapat berhasil karena buku bacaan pada umumnya menambahkan satu spasi tambahan pada akhir baris atau diantara dua kata sehingga tidak terbaca aneh. Bagaimanapun, metode spasi terbuka hanya dapat digunakan dengan memakai ASCII (*American Standard Character Interchange*) format. Bender et al memberikan tiga metode untuk mengungkap white space dalam proses penyembunyian. Spasi terbuka antar kalimat akan menghasilkan nilai "0" apabila hanya terdapat sebuah spasi yang ditambahkan diantara kalimat tersebut. Dengan menambahkan dua spasi akan menghasilkan nilai "1". Metode ini dapat berhasil, tetapi membutuhkan data dalam jumlah besar untuk menyembunyikan sebuah informasi kecil. Dan juga terdapat banyak *software word-processing* yang akan secara otomatis membetulkan spasi antara kalimat, sehingga metode ini seringkali gagal. Metode spasi *end-of-line (EOL)* mengutarakan *white space* pada akhir dari masing-masing baris. Data disembunyikan menggunakan jumlah spasi yang telah ditentukan sebelumnya dari akhir untuk masing- masing kalimat. Sebagai contoh dua spasi akan menyembunyikan satu bit, empat spasi akan menyembunyikan dua bit dan delapan spasi akan menghasilkan tiga bit dan seterusnya. Teknik ini lebih baik dibandingkan metode spasi terbuka antar kalimat, karena dengan meningkatkan jumlah spasi akan dapat menyembunyikan lebih banyak data. Salah satu kekurangan dari tehnik ini adalah dapat hilangnya informasi tersembunyi jika *hard copy* data yang diberikan.

Pada akhirnya, pemerataan kanan dari text dapat digunakan pula untuk menyembunyikan informasi rahasia pada data text. Penghitungan dan pengontrolan

spasi diantara kata dapat menyembunyikan informasi dalam data text yang terlihat tidak penting. Sebuah spasi antara kata akan menghasilkan nilai "0" dan dua buah spasi akan menghasilkan nilai "1". Bagaimanapun, pendekatan ini akan mempersulit untuk mengeluarkan informasi penitng dari media data text tersebut karena akan semakin tidak mungkin untuk membedakan sebuah spasi biasa dengan spasi yang berfungsi untuk penyembunyian data. Untuk mewujudkan hal ini, Bender et al menggunakan Manchester coding untuk mengelompokkan bit-bit.

Sehingga "01" diinterpretasikan sebagai "1" dan "10" diinterpretasikan sebagai "0". Dimana "00" dan "11" akan dianggap sebagai null bit string (Munir Rinaldi., 2006).

b. Metode Syntactic

Metode *Syntactic* sebagaimana yang telah di sarankan oleh Bender et al, mengutarakan penggunaan punktuasi dan struktur text untuk menyembunyikan informasi tanpa secara signifikan mengubah arti dari pesan pembawa. Sebagai contoh terdapat dua frase "*bread, butter, and milk*" dan "*bread, butter and milk*" secara gramatikal benar tetapi berbeda dalam penggunaan koma.

Salah satu dapat digunakan secara alternatif dalam pesan text guna mengintepretasikan nilai "1" apabila salah satu metode dipakai dan nilai "0" untuk metode lain yang dipakai.

c. Metode Semantic

Metode *Semantic* menggunakan dua sinonim sebagai nilai *primer* atau *sekunder*. Nilai tersebut akan diterjemahkan kedalam biner "1" atau "0". Bender et al menggunakan sebuah contoh dimana kata "*big*" berfungsi sebagai *primer* dan "*large*" berfungsi sebagai *sekunder*. Oleh karena itu, dalam menguraikan isi sebuah pesan akan menterjemahkan atas penggunaan primer sebagai "1" dan sekunder sebagai "0". Bender et al menyebutkan masalah yang dapat muncul dengan penggunaan metode ini adalah ketika sinonim tidak dapat digantikan karena dapat mengubah arti dari struktur kalimat. Sebagai contoh dalam memanggil seseorang

dalam bahasa Inggris dengan "*cool*" mempunyai arti berbeda dibandingkan dengan memanggilnya "*chilly*".

2.1.2 Metode Steganografi pada Gambar

Sudah banyak metode yang digunakan untuk menyembunyikan pesan di dalam sebuah *image* tanpa mengubah tampilan *image*, sehingga pesan yang disembunyikan tidak akan terlihat. Berikut akan dibahas beberapa metode umum yang digunakan pada *image steganography*.

a. Least Significant Bit Insertion (LSB)

Cara paling umum untuk menyembunyikan pesan adalah dengan memanfaatkan *Least Significant Bit* (*LSB*). Walaupun terdapat kekurangan pada metode ini, tetapi kemudahan implementasinya membuat metode ini tetap digunakan sampai sekarang. Contoh ilustrasinya sebagai berikut : jika digunakan *image* 24 *bit* warna sebagai media, sebuah *bit* dari masing-masing komponen *Red*, *Green*, dan *Blue*, dapat digunakan sehingga 3 *bit* dapat disimpan pada setiap *pixel*. Sebuah *image* 800x 600 *pixel* dapat digunakan untuk menyembunyikan 1.440.000 *bit* (180.000 bytes) data rahasia. Misalnya, di bawah ini terdapat 3 *pixel* dari *image* 24 *bit* warna :

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

jika diinginkan untuk menyembunyikan karakter A (**100000011**) dihasilkan :

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001000 00100111 11101001)

dapat dilihat bahwa hanya 3 *bit* saja yang perlu diubah untuk menyembunyikan karakter A ini.

Jika pesan = 10 *bit*, maka jumlah *byte* yang digunakan = 10 *byte*.

Contoh susunan *byte* yang lebih panjang :

00110011 10100010 11100010 10101011 00100110

10010110 11001001 11111001 10001000 10100011

Pesan : **1110010111**

Hasil penyisipan pada *bit* LSB :

00110011 10100011 11100011 10101010 00100110

10010111 11001000 11111001 10001001 10100011

Pada metode *LSB*, ukuran data yang akan disembunyikan bergantung pada ukuran *cover-object*. Perubahan pada *LSB* ini akan terlalu kecil untuk terdeteksi oleh mata manusia sehingga pesan dapat disembunyikan secara efektif. Proses ekstraksi pesan dapat dengan mudah dilakukan dengan mengekstrak *LSB* dari masing-masing *pixel* pada steganografi secara berurutan dan menuliskannya ke *output file* yang akan berisi pesan tersebut. Keuntungan metode LSB adalah mudah dalam pengimplementasian dan proses *encoding* yang cepat. Pada perkembangannya metode steganografi *LSB* selain diterapkan pada media *image*, juga bisa diterapkan pada media audio (Munir Rinaldi., 2004).

b. Algorithms and Transformation

Algoritma *compression* adalah metode steganografi dengan menyembunyikan data dalam fungsi matematika. 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*).

c. Redundant Pattern Encoding

Redundant Pattern Encoding adalah menggambar pesan kecil pada kebanyakan gambar. Keuntungan dari metode ini adalah dapat bertahan dari *cropping* (kegagalan), kerugiannya yaitu tidak dapat menggambar pesan yang lebih besar.

d. Spread Spectrum method

Spread Spectrum steganografi terpencar-pencar sebagai pesan yang diacak (*encrypt*) melalui gambar (tidak seperti dalam LSB). Untuk membaca suatu pesan, penerima memerlukan algoritma yaitu *crypto-key* dan *stego-key.* Metode ini juga masih mudah diserang yaitu penghancuran atau pengrusakan dari kompresi dan proses image (gambar) (Munir Rinaldi., 2006).

2.1.3 Metode Steganografi pada Suara

Cara untuk mengaplikasikan steganografi pada file *audio* terdiri dari beberapa cara yang lazim digunakan dan prinsip kerja atau algoritma yang digunakan sama seperti pada metode steganografi pada gambar. Berikut adalah beberapa teknik yang digunakan:

a. Low Bit coding

Cara ini lazim digunakan dalam teknik digital steganografi yaitu mengganti LSB input setiap samplingnya dengan data yang dikodekan. Dengan metode ini keuntungan yang didapatkan adalah ukuran pesan yang disispkan *relative* besar, namun berdampak pada hasil audio yang berkualitas kurang dengan banyaknya noise.

b. Phase coding

Metode kedua yang digunakan ini adalah merekayasa fasa dari sinyal masukan. Teori yang digunakan adalah dengan mensubstitusi awal fasa dari tiap awal segment dengan fasa yang telah dibuat sedemikian rupa dan merepresentasikan pesan yang disembunyikan. Fasa dari tiap awal segment ini dibuat sedemikian rupa sehingga setiap segmen masih memiliki hubungan yang berujung pada kualitas suara yang tetap terjaga. Teknik ini menghasilkan keluaran yang jauh lebih baik daripada metode pertama namun dikompensasikan dengan kerumitan dalam realisasinya.

c. Spread Spectrum

Metode yang ketiga adalah penyebaran *spektrum*. Dengan metode ini pesan dikodekan dan disebar ke setiap spectrum *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*.

d. Echo Hiding

Metode terakhir yang sering digunakan adalah menyembunyikan pesan melalui teknik echo. Teknik menyamarkan pesan ke dalam sinyal yang membentuk echo. Kemudian pesan disembunyikan dengan menvariasikan tiga parameter dalam echo.

Besar *amplitude* awal, tingkat penurunan *atenuasi*, dan *offset*. Dengan adanya *offset* dari echo dan sinyal asli maka echo akan tercampur dengan sinyal aslinya, karena sistem pendengaran manusia yang tidak memisahkan antara echo dan sinyal asli. Keempat metode di atas memiliki kesamaan yaitu menggunakan kelemahan dari sistem pendengaran manusia. Maka dari itu teknik steganografi dalam MP3 juga akan menggunakan kelemahan ini untuk menyembunyikan pesan (Munir Rinaldi., 2006).

2.2 Kriptografi

Kriptografi (*cryptography*) berasal dari Bahasa Yunani: "cryptos" artinya "*secret*" (rahasia), sedangkan "*graphein*" artinya "*writing*" (tulisan). Jadi, kriptografi berarti "*secret writing*" (tulisan rahasia). Ada beberapa definisi kriptografi yang telah dikemukakan di dalam berbagai literatur, seperti:

1. Bruce Schneier di dalam bukunya "*Applied Cryptography*" menyatakan bahwa: Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan (*Cryptography is the art and science of keeping messages secure*).
2. Menezes, Alfred J., Paul C. van Oorschot dan Scott A. Vanstone dalam buku mereka "*Handbook of Applied Cryptography*" menyatakan bahwa: Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan

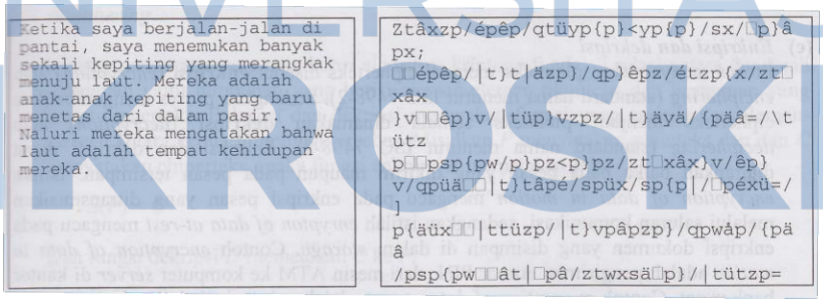
dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi (Munir Rinaldi., 2006).

Di dalam kriptografi, akan sering ditemukan berbagai istilah atau terminologi. Beberapa istilah yang penting untuk diketahui diberikan di bawah ini.

1. Plainteks dan Cipherteks.

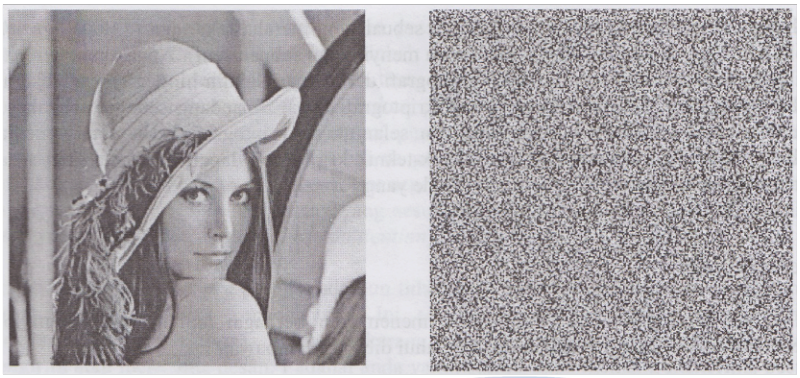
Pesan (*message*) adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan adalah plainteks (*plaintext*) atau teks-jelas (*cleartext*). Pesan dapat berupa data atau informasi yang dikirim (melalui kurir, saluran telekomunikasi) atau yang disimpan di dalam media perekaman (kertas, storage). Pesan yang tersimpan tidak hanya berupa teks, tetapi juga dapat berbentuk citra (*image*), suara/bunyi (*audio*) dan video atau berkas biner lainnya.

Agar pesan tidak dapat dimengerti maknanya oleh pihak lain, maka pesan perlu disandikan ke bentuk lain yang tidak dapat dipahami. Bentuk pesan yang tersandi disebut cipherteks (*ciphertext*) atau kriptogram (*cryptogram*). Cipherteks harus dapat ditransformasikan kembali menjadi plainteks semula agar pesan yang diterima bisa dibaca. Gambar 1 dan 2 memperlihatkan contoh dari dua buah plainteks, masing-masing berupa teks dan gambar, serta cipherteks yang berkoresponden (Munir Rinaldi., 2006).



Gambar 2.1 Plainteks berupa Teks dan Cipherteksnya

(a) Plainteks (b) Cipherteks



Gambar 2.2 Plainteks berupa Gambar dan Cipherteksnya

(a) Plainteks (b) Cipherteks

2. Pengirim dan penerima

Komunikasi data melibatkan pertukaran pesan antara dua entitas. Pengirim (*sender*) adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima (*receiver*) adalah entitas yang menerima pesan. Entitas di sini dapat berupa orang, mesin (komputer), kartu kredit, dan sebagainya. Jadi, orang bisa bertukar pesan dengan orang lainnya (contoh: Alice berkomunikasi dengan Bob), sedangkan di dalam jaringan komputer, mesin (komputer) berkomunikasi dengan mesin (contoh: mesin ATM berkomunikasi dengan komputer *server* di bank). Pengirim tentu menginginkan pesan dapat dikirim secara aman, yaitu ia yakin bahwa pihak lain tidak dapat membaca isi pesan yang ia kirim. Solusinya adalah dengan cara menyandikan pesan menjadi *cipherteks* (Munir Rinaldi., 2006).

3. Enkripsi dan dekripsi

Proses menyandikan *plainteks* menjadi *cipherteks* disebut enkripsi (*encryption*) atau *enciphering*. Sedangkan, proses mengembalikan *cipherteks* menjadi *plainteks* semula dinamakan dekripsi (*decryption*) atau *deciphering*. Enkripsi dan dekripsi dapat diterapkan baik pada pesan yang dikirim maupun pada pesan tersimpan. Istilah *encryption of data in motion* mengacu pada enkripsi pesan yang ditransmisikan melalui saluran komunikasi, sedangkan istilah *encryption of data at-rest* mengacu pada enkripsi dokumen yang disimpan di dalam *storage*. Contoh *encryption of data in motion* adalah pengiriman nomor PIN dari mesin

ATM ke komputer *server* di kantor bank pusat. Contoh *encryption of data at-rest* adalah enkripsi *file* basis data di dalam *hard disk* (Munir Rinaldi., 2006).

4. *Cipher* dan kunci

Algoritma kriptografi disebut juga *cipher* yaitu aturan untuk *enciphering* dan *deciphering*, atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Beberapa *cipher* memerlukan algoritma yang berbeda untuk *enciphering* dan *deciphering*.

Konsep matematis yang mendasari algoritma kriptografi adalah relasi antara dua buah himpunan yaitu himpunan yang berisi elemen-elemen plainteks dan himpunan yang berisi *cipherteks*. Enkripsi dan dekripsi merupakan fungsi yang memetakan elemen-elemen antara kedua himpunan tersebut. Misalkan P menyatakan *plainteks* dan C menyatakan *cipherteks*, maka fungsi enkripsi E memetakan P ke C: (Munir Rinaldi., 2006).

$$E(P) = C$$

Dan fungsi dekripsi D memetakan C ke P,

$$D(C) = P$$

Karena proses enkripsi kemudian dekripsi mengembalikan pesan ke pesan asal, maka kesamaan berikut harus benar,

$$D(E(P))= P$$

Macam – Macam Algoritma Kriptografi

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang digunakan, yaitu:

1. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya)
2. Algoritma Asimetri (menggunakan kunci yang berbeda untuk enkripsi dan dekripsi).
3. Fungsi Hash.

2.2.1 Algoritma Simetris

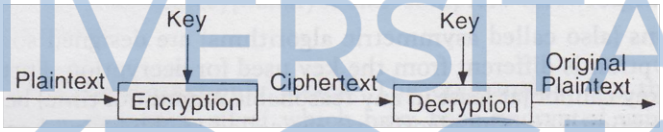
Algoritma simetris, yang kadang-kadang disebut algoritma konvensional, adalah algoritma dimana kunci enkripsi dapat dikalkulasi dari kunci dekripsi dan demikian juga sebaliknya. Pada kebanyakan algoritma simetris, kunci enkripsi dan kunci dekripsi adalah sama. Algoritma ini, yang juga disebut sebagai algoritma kunci rahasia (*secret-key algorithm*), algoritma kunci tunggal (*single-key algorithm*) atau algoritma satu kunci (*one-key algorithm*), memerlukan pengirim dan penerima untuk setuju (sepakat) pada sebuah kunci sebelum mereka dapat berkomunikasi secara aman. Sekuritas dari algoritma ini tergantung sepenuhnya pada kunci, sehingga membocorkan atau memberitahukan kunci kepada orang lain berarti bahwa orang tersebut dapat melakukan proses enkripsi dan dekripsi terhadap suatu pesan. Proses enkripsi dan dekripsi dengan sebuah algoritma simetris dapat didenotasikan sebagai berikut:

$$E_K(M) = C$$

$$D_K(C) = M$$

(Schneier, 1996)

Ilustrasi dari algoritma kunci-simetris ini dapat dilihat pada gambar 3:



Gambar 2.3 Ilustrasi Algoritma Kunci Simetris

Sumber: Schneier, 1996

Algoritma kunci-simetri mengacu pada metode enkripsi yang dalam hal ini baik pengirim maupun penerima memiliki kunci yang sama. Algoritma kunci-simetri modern beroperasi dalam mode bit dan dapat dikelompokkan menjadi dua kategori:

1. *Cipher* aliran (*stream cipher*)

   Algoritma kriptografi beroperasi pada *plaintext* / *ciphertext* dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan / didekripsikan bit per bit. *Cipher* aliran mengenkripsi satu bit setiap kali.

2. *Cipher* blok (*block cipher*)

   Algoritma kriptografi beroperasi pada *plaintext* / *ciphertext* dalam bentuk blok *bit*, yang dalam hal ini rangkaian *bit* dibagi menjadi blok-blok *bit* yang panjangnya sudah ditentukan sebelumnya. Misalnya panjang blok adalah 64 *bit*, maka itu berarti algoritma enkripsi memperlakukan 8 karakter setiap kali enkripsi (1 karakter = 8 bit dalam pengkodean ASCII). *Cipher* blok mengenkripsi satu blok *bit* setiap kali (Munir Rinaldi., 2006).

2.2.2 Algoritma Asimetri

Algoritma sering juga disebut dengan algoritma kunci public, karena untuk melakukan enkripsi dan dekripsi menggunakan kunci yang berbeda. Pada algoritma asimetri kunci terbagi dua bagian, yaitu:

1. Kunci umum (*public key*), yaitu kunci yang boleh diketahui oleh semua orang.
2. Kunci rahasia (*private key*), yaitu kunci yang dirahasiakan (hanya boleh diketahui oleh orang yang diinginkan).

Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci publik orang dapat mengenkripsi pesan tetapi tidak bisa mendekripsinya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut. Algoritma asimetri dapat mengirim pesan lebih baik dari pada algoritma simetri. Contoh, Bob mengirim pesan ke Alice menggunakan algoritma asimetri. Hal yang harus dilakukan adalah:

1. Bob memberitahukan kunci publiknya ke Alice.
2. Alice mengenkripsi pesan dengan menggunakan algoritma kunci publik Bob.
3. Bob mendekripsi pesan dari Alice dengan kunci rahasianya.
4. Begitu juga sebaliknya jika Bob ingin mengirim pesan ke Alice.

Algoritma yang menggunakan kunci publik adalah sebagai berikut:

1. *Digital Signature Algorithm* (DSA).
2. RSA.
3. *Diffie-Hellman* (DH).
4. *Elliptic Curve Cryptography* (ECC), dan lain sebagainya (Ariyus Doni., 2008).

2.2.3 Fungsi Hash

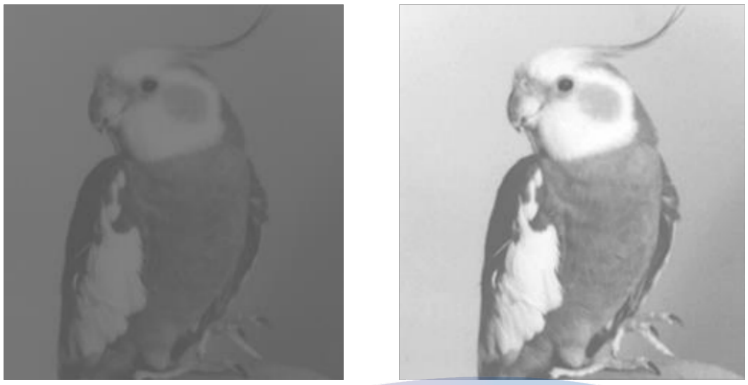
Fungsi Hash sering disebut dengan fungsi Hash satu arah (*one-way function), message digest, fingerprint,* dan *message Authentication code* (MAC), merupakan suatu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam suatu urutan biner dengan panjang yang tetap. Dinamakan sebagai fungsi kompresi karena biasanya, masukan fungsi satu arah ini selalu lebih besar daripada keluarannya, sehingga seolah-olah mengalami kompresi. Namun kompresi hasil fungsi ini tidak dapat dikembalikan ke aslinya sehingga disebut fungsi satu arah. Fungsi Hash biasanya digunakan bila ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda bahwa pesan tersebut benar-benar berasal dari orang yang diinginkan (Ariyus Doni., 2008).

2.3 Citra Digital

Citra (*image*) istilah lain untuk gambar sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi "sebuah gambar bermakna lebih dari seribu kata" (*a picture is more than a thousand words*). Maksudnya tentu sebuah gambar dapat memberikan informasi yang lebih banyak daripada informasi tersebut disajikan dalam bentuk kata-kata (tekstual) (Munir Rinaldi., 2004).

2.3.1 Definisi Pengolahan Citra

Pengolahan citra pada dasarnya adalah memproses citra sehingga menghasilkan kualitas citra yang lebih baik. Sebagai contoh pada gambar 2.4, terlihat bahwa citra (a) tampak agak gelap, lalu dengan operasi pengolahan citra (b) kontrasnya diperbaiki sehingga menjadi lebih terang dan tajam.



(a) (b)

Gambar 2.4 (a) Citra burung nuri yang gelap, (b) Citra burung nuri yang telah diperbaiki kontrasnya sehingga terlihat jelas dan tajam

(Munir Rinaldi., 2004)

Tujuan dari pengolahan citra adalah:

a. Proses memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia dan komputer.
b. Teknik pengolahan citra dengan mentransformasikan citra menjadi citra lain, contoh pemampatan citra (*image compression).*
c. Pengolahan citra merupakan proses awal *(preprocessing)* dari komputer visi (pengolahan citra dan pengenalan pola).

Di dalam bidang komputer, sebenarnya ada tiga bidang studi yang berkaitan dengan data citra, yaitu:

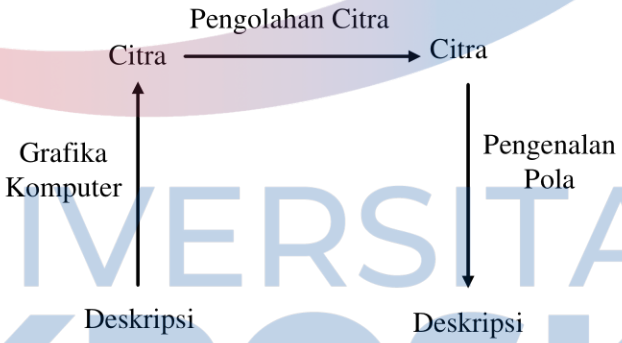
1. Grafika Komputer (*Computer Graphics*), bertujuan menghasilkan citra (lebih sering disebut grafik atau *picture*) dengan primitif-primitif geometri seperti garis, lingkaran dan sebagainya. Primitif-primitif geometri tersebut memerlukan data deskriptif untuk melukiskan elemen-elemen citra. Contoh data deskriptif seperti koordinat titik, panjang garis, jari-jari, lingkaran, tebal garis, warna dan sebagainya. Grafika komputer memainkan peranan penting dalam visualisasi dan *virtual reality*.
2. Pengolahan Citra (*Image Processing*), bertujuan memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (dalam hal ini

komputer). Teknik-teknik pengolahan citra mentransformasikan suatu citra menjadi citra lain. Jadi masukannya adalah citra dan keluarannya juga citra.

3. Pengenalan Pola (*Pattern Recognition/ Image Interpretation*), mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek didalam citra. Manusia bisa mengenali objek yang dilihatnya karena otak manusia telah belajar mengklafikasikan objek-objek di alam sehingga mampu membedakan suatu objek dengan objek yang lain.

Kemampuan sistem visual manusia inilah yang coba ditiru oleh komputer. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut dan memberikan keluaran berupa deskripsi objek di dalam citra (Munir Rinaldi., 2004).

Hubungan antara bidang grafika komputer, pengolahan citra dan pengenalan pola dapat ditunjukkan pada gambar 2.5.



Gambar 2.5 Tiga bidang studi yang berkaitan dengan citra
(Munir Rinaldi., 2004)

a. Operasi Pengolahan Citra

Operasi-operasi yang dilakukan dalam pengolahan citra beraneka ragam. Namun, secara umum operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut:

1. Perbaikan Kualitas Citra (*Image Enhancement*)

Perbaikan atau memodifikasi citra dilakukan untuk meningkatkan kualitas penampakan citra/menonjolkan beberapa aspek informasi yang terkandung dalam citra *(image enhancement).* Contoh-contoh operasi perbaikan citra:

i. Perbaikan kontras gelap/terang
ii. Perbaikan tepian objek (*edge enhancement*)
iii. Pemberian warna semu (*pseudocoloring*)
iv. Penapisan derau (*noise filtering*)

Contoh operasi perbaikan kualitas citra dapat dilihat pada gambar 2.6. Operasi ini menerima masukan sebuah citra yang gambarnya hendak dibuat tampak lebih tajam. Bagian citra yang ditajamkan adalah tepi-tepi objek.

(a) (b)

Gambar 2.6 (a) Citra Lena asli, (b) Citra Lena setelah ditajamkan (Munir Rinaldi., 2004)

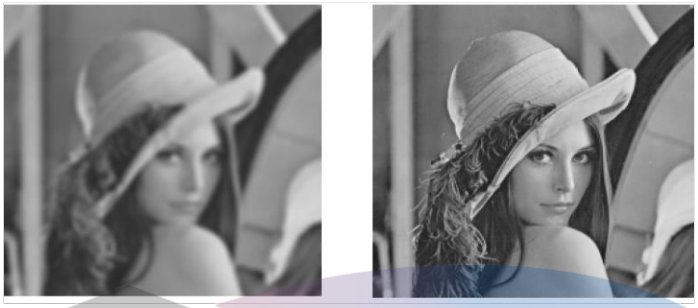
2. Pemugaran Citra (*Image Restoration*)

Operasi ini bertujuan untuk menghilangkan/meminimumkan kerusakan pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui. Contoh-contoh operasi pemugaran citra:

i. Penghilangan kesamaran (*deblurring*).
ii. Penghilangan derau (*noise*)

Gambar 2.7 adalah contoh operasi pemugaran citra. Citra masukan adalah citra yang tampak kabur (*blur*). Kekaburan gambar mungkin disebabkan pengaturan fokus lensa yang tidak tepat atau kamera bergoyang pada pengambilan

gambar. Melalui operasi *deblurring*, kualitas citra masukan dapat diperbaiki sehingga tampak lebih baik.



(a) (b)

Gambar 2.7 Kiri: Citra Lena yang kabur (blur), kanan: Citra Lena setelah deblurring

(Munir Rinaldi., 2004)

3. Pemampatan Citra (Image Compression)

Operasi ini bertujuan untuk merepresentasikan citra dalam bentuk lebih baik, sehingga pemakaian memori lebih sedikit namun tetap mempertahankan kualitas gambar (misal dari .BMP menjadi .JPG). Perhatikan gambar 2.8, gambar sebelah kiri adalah citra berformat BMP dan berukuran 258 KB. Hasil pemampatan citra berformat JPG dapat mereduksi ukuran citra semula sehingga menjadi 49 KB.

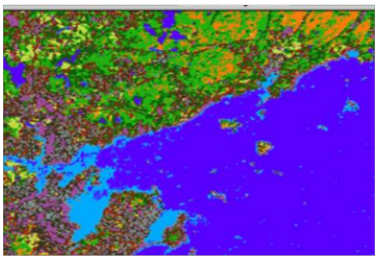
Gambar 2.8 (a) Citra boat.bmp (258 KB) sebelum dimampatkan, (b) citra boat.jpg (49 KB) setelah dimampatkan

(Munir Rinaldi., 2004)

4. Segmentasi Citra (Image Segmentation)

Jenis operasi ini bertujuan untuk memecahkan suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan

pengenalan pola. Pada gambar 2.9 di bawah ini merupakan contoh dari segmentasi citra.



Gambar 2.9 Contoh dari segmentasi citra

(Munir Rinaldi., 2004)

5. Analisis Citra (Image Analysis)

Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Analisis citra diperlukan dalam mengekstraksi ciri-ciri tertentu yang dimiliki citra untuk membantu dalam pengidentifikasian objek. Contoh-contoh operasi analisis citra adalah:

i. Pendeteksian tepi objek (*edge detection*)
ii. Ekstraksi batas (*boundary*)
iii. Representasi daerah (*region*)

Gambar 2.10 merupakan contoh operasi pendeteksian tepi pada citra kamera. Operasi ini menghasilkan semua tepi (*edge*) di dalam citra.

Gambar 2.10 (a) Citra camera, (b) Citra hasil pendeteksian seluruh tepi

(Munir Rinaldi., 2004)

6. Rekonstruksi Citra (Image Reconstruction)

Jenis operasi ini bertujuan membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto *rontgen* dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh (Putra Darma., 2010).

b. Aplikasi Pengolahan Citra

Pengolahan citra mempunyai aplikasi yang sangat luas dalam berbagai bidang kehidupan. Di bawah ini contoh aplikasi dalam beberapa bidang yang memanfaatkan pengolahan citra (Putra Darma., 2010).

1. Bidang perdagangan
   i. Pembacaan kode batang (*bar code*) yang tertera pada barang (umum digunakan di pasar swalayan/supermarket).
   ii. Pengenalan huruf/angka pada suatu formulir secara otomatis.
2. Bidang militer
   i. Mengenali sasaran peluru kendali melalui sensor visual.
   ii. Mengidentifikasi jenis pesawat musuh.
3. Bidang kedokteran
   a. Mendeteksi kelainan tubuh dari foto sinar X.
   b. Rekonstruksi foto janin hasil USG.
4. Bidang biologi
   Pengenalan jenis kromosom melalui gambar mikroskopik.
5. Komunikasi data
   Pemampatan citra yang ditransmisi.
6. Hiburan
   Pemampatan video (*MPEG*)
7. Robotika
   Visualy-guided autonomous navigation
8. Pemetaan
   Klasifikasi penggunaan tanah melalui foto udara/LANDSAT
9. Geologi

Mengenali jenis batu-batuan melalui foto udara/LANDSAT

10. Hukum
    i. Pengenalan sidik jari
    ii. Pengenalan foto narapidana.

2.3.2 Format Citra

Sebuah format *file* citra harus dapat menyatukan kualitas citra, ukuran *file* dan kompabilitas dengan berbagai aplikasi. Format *file* citra standar yang digunakan saat ini terdiri dari beberapa jenis. Format- format ini digunakan untuk menyimpan citra dalam sebuah *file*. Setiap format memiliki karakteristik masing-masing (Putra Darma., 2010).

Terdapat dua jenis format *file* citra yang sering digunakan dalam pengolahan citra, yaitu citra *bitmap* dan citra *vektor*. Citra *bitmap* ini menyimpan data kode citra secara digital dan lengkap (cara penyimpanannya adalah per piksel). Citra *bitmap* direpresentasikan dalam bentuk matriks atau dipetakan dengan menggunakan bilangan biner atau sistem bilangan yang lain. Citra ini memiliki kelebihan untuk memanipulasi warna, tetapi untuk mengubah objek lebih sulit. Tampilan *bitmap* mampu menunjukkan kehalusan gradasi bayangan dan warna dari sebuah gambar. Tetapi bila tampilan diperbesar maka tampilan di monitor akan tampak pecah-pecah (kualitas citra menurun). Contoh format *file* citra antara lain adalah BMP, GIF, TIF, dan lain-lain. Sedangkan pada format *file* citra *vektor* merupakan citra yang dihasilkan dari perhitungan matematis dan 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.

a. *Bitmap* (*.bmp)

Citra *bitmap* merupakan 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

untuk gambar berwarna. Banyaknya titik dalam 1 *inchi* dikenal dengan dpi (*dot per inchi*).

Pada format *bitmap*, citra disimpan sebagai suatu matriks dimana masing-masing elemennya digunakan untuk menyimpan informasi warna untuk setiap piksel. Jumlah warna yang dapat disimpan dalam *bitmap* ditentukan dengan satuan *bit per pixel* (bpp). Semakin besar ukuran *bit per pixel* dari suatu *bitmap*, semakin banyak jumlah warna yang dapat disimpan. Format *bitmap* ini cocok digunakan untuk menyimpan citra digital yang memiliki banyak variasi dalam bentuk maupun warnanya, seperti foto, lukisan dan *frame video.*

b. *Joint Photographic Group* (.jpg)

Format JPEG atau JPG adalah suatu format citra yang memiliki beberapa kelebihan dan kekurangan dibandingkan format-format lainnya. Kelebihan paling umum adalah ukuran *file* (*size)* dan kualitas citra yang dihasilkan lebih fleksibel dibandingkan lainnya. Sedangkan kekurangannya adalah kualitas citranya tidak sesuai dengan aslinya. Format JPEG mendukung mode warna RGB, CMYK dan *Grayscale*, tetapi tidak mampu menampilkan citra dengan latar belakang transparan. Format JPEG menterjemahkan informasi tersebut menjadi komponen *luminance* (komponen cahaya) dan dua komponen *chromatic* (komponen perubahan warna dari hijau ke merah dan dari biru ke kuning).

c. *Tagged Image Format* (.tif)

Format .tif merupakan format penyimpanan citra yang dapat digunakan untuk menyimpan citra *bitmap* hingga citra dengan warna palet terkompresi. Format ini dapat digunakan untuk menyimpan citra yang tidak terkompresi dan juga citra terkompresi.

d. *Portable Network Graphics* (.png)

Format .png adalah format penyimpanan citra terkompresi. Format ini dapat digunakan pada citra *grayscale*, citra dengan palet warna, dan juga citra *fullcolor*. Format .png juga mampu menyimpan informasi hingga kanal alpha dengan penyimpanan sebesar 1 hingga 16 *bit* per kanal.

e. *Graphics Interchange Format* (.gif)

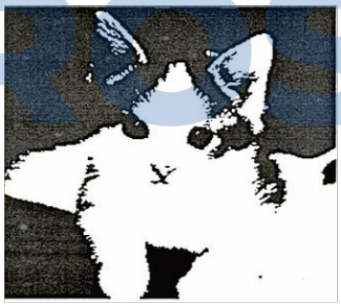
Format ini dapat digunakan pada citra warna dengan palet 8 *bit*. Penggunaan format ini umumnya pada aplikasi *web*. Kualitas yang rendah menyebabkan format ini tidak terlalu populer dikalangan peneliti pengolahan citra digital (Putra Darma., 2010).

2.3.3 Jenis Citra

Nilai suatu piksel memiliki nilai dalam rentang tertentu, dari nilai minimum sampai nilai maksimum. Jangkauan yang digunakan berbeda-beda tergantung dari jenis warnanya. Namun secara umum jangkauannya adalah 0-255. Citra dengan penggambaran seperti ini digolongkan ke dalam citra integer. Berikut adalah jenis-jenis citra berdasarkan nilai pikselnya:

a. Citra Biner

Citra biner adalah citra digital yang hanya memiliki dua kemungkinan nilai piksel yaitu hitam dan putih. Citra biner juga disebut sebagai citra B&W (*Black and White)* atau monokrom. Hanya dibutuhkan 1 *bit* untuk mewakili nilai setiap piksel dari citra biner. Citra biner sering kali muncul sebagai hasil dari proses pengolahan seperti segmentasi, pengembangan, morfologi, ataupun *difhering*. Pada gambar 2.11 di bawah ini merupakan contoh dari citra biner.

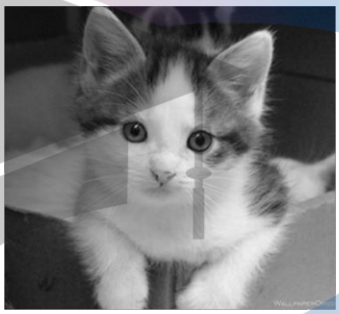


Gambar 2.11 Citra biner

(Putra Darma., 2010)

b. Citra *Grayscale*

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, dengan kata lain nilai bagian *red* = *green* = *blue.* Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan, dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih. Pada gambar 2.12 merupakan contoh citra *grayscale* yang memiliki kedalaman warna 8 *bit* (256 kombinasi warna keabuan).



Gambar 2.12 Citra grayscale

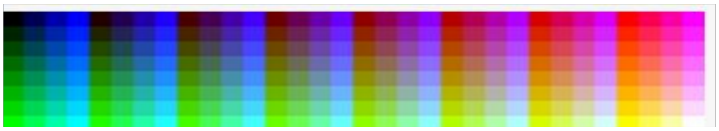
(Putra Darma., 2010)

c. Citra Warna (8 *Bit*)

Setiap piksel dari citra warna (8 *bit*) hanya diwakili oleh 8 *bit* dengan jumlah warna maksimum yang dapat digunakan adalah 256 warna. Ada dua jenis warna 8 *bit*. Pertama, citra warna 8 *bit* dengan menggunakan palet warna 256 dengan setiap paletnya memiliki pemetaan nilai (*colormap)* RGB tertentu. Model ini lebih sering digunakan.

Bit-7	*Bit*-6	*Bit*-5	*Bit*-4	*Bit*-3	*Bit*-2	*Bit*-1	*Bit*-0
R	R	R	G	G	G	B	B

Kedua, setiap piksel memiliki format 8 *bit*. Bentuk kedua ini dinamakan 8 *bit true color.* Berikut adalah warna-warna dari citra warna 8 *bit.*



Gambar 2.13 di bawah ini merupakan contoh dari citra warna 8 *bit.*

Gambar 2.13 Citra warna 8 bit dengan palet

(Putra Darma., 2010)

d. Citra Warna (16 *Bit*)

Setiap piksel dari citra warna 16 *bit* diwakili dengan 2 *byte* memori (16 *bit*). Warna 16 *bit* memiliki 65.536 warna. Dalam formasi *bit*, nilai merah dan biru mengambil tempat di 5 *bit* di kanan dan kiri. Komponen hijau memiliki 5 *bit* ditambah 1 *bit* ekstra. Pemilihan komponen hijau dengan deret 6 *bit* dikarenakan penglihatan manusia lebih sensitif terhadap warna hijau.

Bit- 15	*Bit-* 14	*Bit-* 13	*Bit-* 12	*Bit-* 11	*Bit-* 10	*Bit-* 9	*Bit-* 8	*Bit-* 7	*Bit-* 6	*Bit-* 5	*Bit-* 4	*Bit-* 3	*Bit-* 2	*Bit-* 1	*Bit-* 0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

Berikut adalah warna-warna dari citra warna 16 *bit.*

Gambar 2.14 merupakan contoh citra yang dihasilkan dari warna 16 *bit*.

Gambar 2.14 Citra warna 16 bit

(Putra Darma., 2010)

e. Citra Warna (24 *Bit*)

Setiap piksel dari citra warna 24 *bit* diwakili dengan 24 *bit* sehingga total 16.777.216 variasi warna. Variasi ini sudah lebih dari cukup untuk memvisualisasikan seluruh warna yang dapat dilihat oleh penglihatan manusia. Setiap poin informasi piksel (RGB) disimpan ke dalam 1 *byte* data. 8 *bit* pertama menyimpan nilai biru, diikuti dengan nilai hijau pada 8 *bit* kedua dan pada 8 *bit* terakhir merupakan warna merah. Gambar 2.15 merupakan contoh dari citra 24 *bit*.

Gambar 2.15 Citra 24 bit

(Putra Darma., 2010)

2.3.4 Pengukuran Kualitas Citra

Pengukuran kualitas citra bertujuan untuk mengukur efektifitas dari algoritma yang digunakan. Pengukuran kualitas citra ini terdiri dari dua kelas. Pertama, didefinisikan secara pengukuran matematika seperti *Mean Square Error* (MSE), *Peak Signal to Noise Ratio* (PSNR), dan lain sebagainya. Kedua, metode pengukuran berdasarkan karakteristik sistem penglihatan manusia (HSV/*Human Visual System*) dalam usaha untuk menunjukkan persepsi kualitas suatu citra (Iain E. G. R., 2002).

a. MSE (*Mean Square Error*).

MSE adalah nilai *error* kuadrat rata-rata antara *cover image* dengan citra ter-steganografi, secara matematis dapat dirumuskan sebagai berikut:

$$\mathrm{MSE} = \frac{1}{3mn}\sum_{l=1}^{3}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[\mathrm{I(i,j)} - \mathrm{K(i,j)}]^2$$

Dimana : MSE = Nilai *Mean Square Error* dari citra

I (*i,j*) = nilai piksel di citra asli.

K (*i,j*)= nilai piksel pada citra hasil pengolahan

m, n = dimensi citra.

b. *Peak Signal Noise Ratio* (PSNR)

Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR biasanya diukur dalam satuan desibel. PSNR digunakan untuk mengetahui perbandingan kualitas citra sebelum dan sesudah disisipkan pesan. Nilai PSNR dihitung dari kuadrat nilai maksimum sinyal dibagi dengan MSE seperti pada rumus berikut ini:

$$\text{PSNR} = 20^{*} log_{10}\left(-\frac{255}{\sqrt{MSE}}\right)$$

Nilai 255 merupakan nilai maksimum dari piksel citra yang digunakan, karena dalam tugas akhir ini menggunakan citra *bitmap* 24 *bit,* maka 2^{24} atau 16.777.216. Nilai MSE yang semakin rendah akan semakin baik, sedangkan semakin tinggi nilai PSNR, semakin bagus kualitas citra tersebut (Iain E.G.R., 2002).

Tabel 2.1 Kualitas Citra

PSNR (db)	Kualitas Citra
60	Sangat bagus, tidak ada noise yang jelas.
50	Baik, terdapat noise tapi kualitas gambar yang bagus.
40	Wajar, butir halus atau salju dalam gambar, beberapa detail halus hilang.
30	Buruk, banyak sekali noise.
20	Tidak dapat digunakan.

(Munir Rinaldi., 2004)

2.4 Block Cipher Baru dengan 256 Bit Kunci

Block cipher baru yang dibuat merupakan pengembangan dari metode *Hill Cipher*. Metode *block cipher* ini menggunakan 256 bit kunci dan dapat diterapkan pada semua karakter beserta karakter spesial dan digit.

2.4.1 Metode Hill Cipher

Hill cipher dapat beroperasi pada 26 buah alfabet dari a sampai z saja. Jika P adalah plaintext dan K adalah kunci, maka proses enkripsi dan dekripsi dapat didefinisikan sebagai berikut:

Enkripsi: *ciphertext*, $C = KP \bmod 26$

Dekripsi: *plaintext*, $P = K^{-1}C \bmod 26$

Dimana K^{-1} adalah invers aritmatika modular. *Hill cipher* memiliki beberapa batasan. Algoritma ini tidak dapat diterapkan pada karakter spesial dan digit. Semakin kecil blok yang diambil maka panjang kunci juga menjadi lebih pendek. Algoritma ini sangat sederhana, sehingga sangat rentan terhadap *exhaustive key search attack* dan *known plaintext attack*.

2.4.2 Algoritma Block Cipher Baru

Algoritma *block cipher* yang baru merupakan pengembangan dari *Hill Cipher* untuk menangani kelemahan dari *Hill Cipher* seperti penjabaran diatas. Algoritma baru ini memiliki panjang block sebesar 128 bit (16 karakter) dan panjang kunci sebesar 256 bit (32 karakter). Proses pembentukan sub kunci, enkripsi dan dekripsi dari algoritma ini dapat dijabarkan sebagai berikut:

1. Pembentukan Sub Kunci

   Kunci dengan panjang 32 karakter (256 bit) direpresentasikan sebagai sebuah matriks berukuran 4 x 8, dan disimbolkan sebagai K. Kemudian, hasilkan delapan buah sub kunci K_1, K_2, K_3, K_4, K_5, K_6, K_7 dan K_8.

   a. Sub kunci pertama K_1 adalah sebuah matriks berukuran 4 x 4 yang diperoleh dengan mengambil 4 kolom pertama dari K.
   b. Sub kunci kedua K_2 diperoleh dengan menukar kolom pertama dan kedua dari K_1.
   c. Sub kunci ketiga K_3 diperoleh dengan menukar kolom pertama dan ketiga dari K_1.
   d. Sub kunci keempat K_4 diperoleh dengan menukar kolom pertama dan keempat dari K_1.

e. Sub kunci kelima K_5 adalah sebuah matriks berukuran 4 x 4 yang diperoleh dengan mengambil 4 kolom sisa dari K.

f. Sub kunci keenam K_6 diperoleh dengan menukar kolom pertama dan kedua dari K_5.

g. Sub kunci ketujuh K_7 diperoleh dengan menukar kolom pertama dan ketiga dari K_5.

h. Sub kunci kedelapan K_8 diperoleh dengan menukar kolom pertama dan keempat dari K_5.

2. Enkripsi

Pesan akan dipecahkan menjadi beberapa blok berbeda, dengan ukuran blok masing-masing sebesar 16 karakter. Jika blok terakhir kurang dari 16 karakter, maka akan ditambahkan spasi kosong pada akhir blok hingga blok berukuran 16 karakter. Setiap blok akan direpresentasikan oleh sebuah matriks berukuran 4 x 4, dan disimbolkan dengan P. Proses enkripsi dari sebuah blok pesan akan dilakukan dengan menggunakan prosedur berikut:

```
For i = 1 to 8
{
    P1 = (Ki * P) mod (127)
    P = P1
}
For i = 1 to 8
{
    P2 = stir (P)
    P3 = XOR (Ki, P2)
    P = P3
}
C = P
```

Dimana K * P adalah operasi perkalian matriks dari dua buah matriks berukuran 4 x 4.

P_1, P_2 dan P_3 adalah hasil *intermediate*.

Operasi lainnya yang digunakan adalah:

a. Operasi Stir, yang dapat didefinisikan sebagai berikut:

1) Dua bit pertama (bit 1 dan bit 2) dari setiap *byte* pada sebuah baris dari A dikombinasikan untuk membentuk *byte* pertama dari B pada setiap baris.
2) Dua bit berikutnya (bit 3 dan bit 4) dari setiap *byte* pada sebuah baris dari A dikombinasikan untuk membentuk *byte* berikutnya dari B pada setiap baris.
3) Dua bit berikutnya (bit 5 dan bit 6) dari setiap *byte* pada sebuah baris dari A dikombinasikan untuk membentuk *byte* berikutnya dari B pada setiap baris.
4) Dua bit berikutnya (bit 7 dan bit 8) dari setiap *byte* pada sebuah baris dari A dikombinasikan untuk membentuk *byte* berikutnya dari B pada setiap baris.

Operasi stir adalah operasi yang dapat balik. Contoh operasi stir:

Matrix A=

$$\begin{bmatrix} 11001100, & 00101010, & 11100110, & 11001100 \\ 11011110, & 10101010, & 00100100, & 01010111 \\ 00011001, & 11101111, & 01111000, & 11000111 \\ 11111100, & 11011011, & 11011100, & 10011001 \end{bmatrix}$$

B= Stir (A) =

$$\begin{bmatrix} 11001111, & 00101000, & 11100111, & 00101000 \\ 11100001, & 01101001, & 11100101, & 10100011 \\ 00110111, & 01101100, & 10111001, & 01110011 \\ 11111110, & 11010101, & 11101110, & 00110001 \end{bmatrix}$$

Stir (Stir (A)) =

$$\begin{bmatrix} 11001100, & 00101010, & 11100110, & 11001100 \\ 11011110, & 10101010, & 00100100, & 01010111 \\ 00011001, & 11101111, & 01111000, & 11000111 \\ 11111100, & 11011011, & 11011100, & 10011001 \end{bmatrix}$$

b. Operasi XOR

Jika A dan B adalah matriks dengan ukuran yang sama, maka XOR (A, B) adalah operasi *exclusive-or* bit dengan bit.

3. Dekripsi

Ciphertext block C dapat dikonversi ke *plaintext block* P dengan prosedur berikut:

```
For i = 8 to 1
{
    C1 = XOR (Ki, C)
    C2 = stir (C1)
    C = C2
}
For i = 8 to 1
{
    C3 = {Kmi^-1 * C} mod (127)
    C = C3
}
P = C
```

Dimana P adalah *plaintext* asli dan C_1, C_2, C_3 adalah hasil *intermediate*. K_{mi}^{-1} adalah invers modular aritmatika dari matriks K_i. (Gandharba Swain, et. al., 2012).

2.5 Metode Dynamic Steganography

Metode *dynamic steganography* merupakan metode steganografi citra yang dikembangkan dari metode *LSB substitution*, karena metode *LSB substitution* tidak aman terhadap penyerangan dengan menggunakan operasi pengolahan citra. Untuk itu, maka dapat dilakukan modifikasi terhadap metode *LSB* dimana bit tidak hanya disisipkan pada bit ke-8, namun juga dapat disisipkan pada bit ke-6 dan bit-7. Karena lokasi penyisipan ditentukan pada saat proses kerja dari algoritma, maka algoritma ini dinamakan *dynamic steganography.* Kelebihan metode *dynamic steganography* terletak pada tingkat keamanan data yang lebih tinggi karena lokasi penyisipan tergantung pada *ciphertext* yang dihasilkan oleh pesan *input* (Gandharba Swain, et. al., 2012).

2.5.1 Teknik Penyisipan (Embedding) Dynamic Steganography

Citra sampul dikonversikan ke bentuk biner. Setiap piksel menjadi 3 *byte* untuk citra RGB. Panjang *ciphertext* dihitung, dan disimbolkan sebagai L. 8 piksel awal akan digunakan untuk menyimpan nilai L. Oleh karena itu, penyisipan bit *ciphertext* dimulai dari bit 9. Prosedur penyisipan dapat dirincikan sebagai berikut:

1. Input citra sampul.
2. Input *ciphertext*.
3. Set L = panjang *ciphertext*.
4. Sisipkan L dimulai dari piksel 1 menjadi 8 menggunakan lokasi 2 buah *least significant bit*.
5. Sisipkan dua bit *ciphertext* pada bit 7 dan bit 8 di piksel 9. Ambil dua bit berikutnya dari *ciphertext*. Set i = 9 dan L = L – 2.
6. Lakukan salah satu dari keempat langkah dari (a) sampai (d) berikut:
   a. Jika dua bit dari *ciphertext* yang disisipkan pada piksel ke-i adalah 00, maka dua bit berikutnya dari *ciphertext* akan disisipkan pada posisi bit 7 dan bit 8 dari piksel-(i + 1) pada citra.
   b. Jika dua bit dari *ciphertext* yang disisipkan pada piksel ke-i adalah 01, maka dua bit berikutnya dari *ciphertext* akan disisipkan pada posisi bit 8 dan bit 7 dari piksel-(i + 1) pada citra.
   c. Jika dua bit dari *ciphertext* yang disisipkan pada piksel ke-i adalah 10, maka dua bit berikutnya dari *ciphertext* akan disisipkan pada posisi bit 6 dan bit 7 dari piksel-(i + 1) pada citra.
   d. Jika dua bit dari *ciphertext* yang disisipkan pada piksel ke-i adalah 11, maka dua bit berikutnya dari *ciphertext* akan disisipkan pada posisi bit 7 dan bit 6 dari piksel-(i + 1) pada citra.
7. Set i = i + 1 dan L = L – 2. Jika (L = 0) lompat ke langkah 4, jika tidak, maka ambil dua bit berikutnya dari *ciphertext* dan kembali ke langkah 6.
8. Berhenti.

Sebagai contoh, asumsikan bit data yang akan disisipkan adalah 11001001 00011010 01100110 10011111. Anggap piksel berbeda dari citra yang akan

digunakan sebagai tempat penyisipan data disimbolkan sebagai A, B, C, D, dan seterusnya. Maka proses penyisipan bit data pada citra sampul dapat dilihat pada Tabel 1.

Tabel 2.2 Proses penyisipan bit pada Citra

Cover image pixels (each 1 byte)	Operation	Location
pixel A	embedd 11	7th and 8th
pixel B	embedd 00	7th and 6th
pixel C	embedd 10	7th and 8th
pixel D	embedd 01	6th and 7th
pixel E	embedd 00	8th and 7th
pixel F	embedd 01	7th and 8th
pixel G	embedd 10	8th and 7th
pixel H	embedd 10	6th and 7th
pixel I	embedd 01	6th and 7th
pixel J	embedd 10	8th and 7th
pixel K	embedd 01	6th and 7th
pixel L	embedd 10	8th and 7th
pixel M	embedd 10	6th and 7th
pixel N	embedd 01	6th and 7th
pixel O	embedd 11	8th and 7th
pixel P	embedd 11	7th and 6th
etc...		

(Gandharba Swain, et. al., 2012)

2.5.2 Teknik Penemuan Kembali (Retrieving) Dynamic Steganography

Proses penemuan kembali bit data dari citra stego dapat dirincikan sebagai berikut:

1. Input citra stego.
2. Input kunci enkripsi.
3. Hitung panjang bit dari pesan yang disisipkan (L) dengan mengambil bit dari piksel 1 menjadi 8. Ambil dua bit ciphertext dari posisi bit 7 dan bit 8 dari piksel 9. Simpan kedua bit dalam variabel CIPHER. Set L = L – 2 dan i = 9.
4. Lakukan salah satu dari keempat langkah dari (a) sampai (d) berikut:
   a. Jika bit yang ditarik keluar dari piksel ke-i adalah 00, maka tarik keluar bit 7 dan bit 8 dari piksel ke-(i + 1).
   b. Jika bit yang ditarik keluar dari piksel ke-i adalah 01, maka tarik keluar bit 8 dan bit 7 dari piksel ke-(i + 1).

c. Jika bit yang ditarik keluar dari piksel ke-i adalah 10, maka tarik keluar bit 6 dan bit 7 dari piksel ke-(i + 1).
d. Jika bit yang ditarik keluar dari piksel ke-i adalah 11, maka tarik keluar bit 7 dan bit 6 dari piksel ke-(i + 1).

5. Gabungkan kedua bit ini ke variabel CIPHER. Set L = L – 2.
6. Jika (L > 0) lompat ke langkah 2, jika tidak, maka lompat ke langkah 8.
7. Output = citra stego
8. Berhenti (Gandharba, et. al., 2012).

2.5.3 Algoritma Skema New Block Cipher dengan Dynamic Steganography

Algoritma pada bagian pengirim dapat dijabarkan sebagai berikut:

1. Input citra sampul.
2. Input pesan.
3. Input kunci enkripsi.
4. Pecahkan pesan rahasia ke bentuk block, setiap block terdiri dari 16 karakter (128 bit).
5. Aplikasikan proses enkripsi untuk mengkonversikan setiap block pesan menjadi sebuah *block ciphertext*.
6. Output *ciphertext*.
7. Gabungkan semua *block ciphertext* untuk memperoleh sebuah *ciphertext* lengkap.
8. Konversikan *ciphertext* ini menjadi bentuk biner.
9. Sisipkan *ciphertext* ke dalam citra sampul, sehingga diperoleh citra stego dengan menggunakan metode *dynamic steganography*.
10. Output citra stego dan kirimkan citra stego kepada penerima.

Algoritma pada bagian penerima dapat dijabarkan sebagai berikut:

1. Input citra stego.
2. Input kunci enkripsi.
3. Tarik keluar bit *ciphertext* (dua bit per piksel) dari citra stego.
4. Gabungkan semua bit dan konversikan ke bentuk teks.
5. Output *ciphertext*.

6. Setelah itu, pecahkan menjadi bentuk block dengan panjang setiap block sebesar 16 karakter.
7. Aplikasikan proses dekripsi ke setiap *block ciphertext* untuk memperoleh *block plaintext*.
8. Gabungkan semua *block plaintext* untuk memperoleh pesan rahasia.
9. Output *plaintext* (Gandharba Swain, et. al., 2012).