BAB II

KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

Berdasarkan judul penelitian ini "Steganografi Ciphertext 3DES Pada Citra Digital Menggunakan Modifikasi Metode Gifshuffle" penulis menemukan beberapa pustaka yang relevan untuk mendukung penelitian ini antara lain:

[PEN-05] menyatakan GifShuffle merupakan sebuah algoritma untuk menyisipkan pesan ke dalam citra berformat GIF. Cara kerjanya adalah dengan mengubah susunan palet warna pada citra tersebut di mana setiap perubahan berkorespondensi dengan sebuah karakter yang dideklarasikan sebelumnya. Hasil citra yang telah disisipi pesan tidak terlalu banyak mengalami perubahan sedangkan untuk hasil pengujian dengan melakukan operasi citra setelah disisipi pesan yaitu pesan gagal untuk diekstraksi karena palet warna ikut berubah.

[GHO-10] menyatakan implementasi 3DES pada kombinasi *Very High Speed Integrated Circuit Hardware Description Language* dan FPGA menghasilkan kecepatan yang tinggi pada kinerja perangkat keras FPGA.

Dari penelitian-penelitian yang dijelaskan tersebut hanya berfokus pada algoritma 3DES saja atau algoritma GifShuffle saja, tidak ada proses penggabungan dari kedua algoritma tersebut sehingga yang membedakan penelitian ini dengan penelitian sebelumnya adalah dalam penelitian ini digabungkan dua buah algoritma yaitu algoritma 3DES dan algoritma GifShuffle yang mana dengan penggabungan ini diharapan mampu memenuhi kebutuhan akan keamanan data atau informasi yang akan dipertukarkan.

2.2 Steganografi

Istilah steganografi berasal dari bahasa Yunani yaitu *steganos* yang artinya adalah penyamaran atau penyembuyian dan *graphein* yang artinya adalah tulisan. *Wax tablet* diperkenalkan di Yunani pada tahun 400 SM untuk penyembunyian pesan. Pesan ditulis di atas media kayu kemudian disembunyikan dengan melapisi lilin sebagai penutupnya. Ketika Perang Dunia II, untuk menulis dokumen yang

tersembunyi, pesan ditulis dengan menggunakan tinta yang tak terlihat, untuk melihat apa yang ditulis, dokumen harus dipanaskan agar tinta tak terlihat tersebut menjadi gelap dan pesan yang disembunyikan dapat dibaca.

Steganografi modern, dengan kemajuan teknologi komputer digital, fokus steganografi cenderung kearah menyembunyikan pesan dalam media digital, misalnya citra digital dengan format BMP, GIF, suara dan musik digital, bahkan dengan algoritma steganografi tertentu, tidak hanya media digital, namun berkas HTML dapat pula disisipi pesan dengan cara merubah urutan *tag*-nya. Steganografi ini disebut dengan steganografi digital [LUT-09].

Steganografi membutuhkan dua properti yaitu media penampung dan pesan rahasia. Media penampung yang umum digunakan adalah gambar, suara, video, atau teks. Pesan yang disembunyikan dapat berupa sebuah artikel, gambar, daftar barang, kode program, atau pesan lain. Keuntungan steganografi dibandingkan dengan kriptografi adalah bahwa pesan yang dikirim tidak menarik perhatian sehingga media penampung yang membawa pesan tidak menimbulkan kecurigaan bagi pihak ketiga. Ini berbeda dengan kriptografi di mana pesan hasil enkripsi menimbulkan kecurigaan bahwa pesan tersebut merupakan pesan rahasia [MUN-06].

Penyembunyian pesan rahasia ke dalam media penampung pasti mengubah kualitas media tersebut. Kriteria yang harus diperhatikan dalam penyembunyian pesan adalah [MUN-06]:

1. *Imperceptibility*. Keberadaan pesan rahasia tidak dapat dipersepsi secara langsung. Misalnya, jika media yang digunakan untuk menyembunyikan pesan (*covertext*) berupa citra, maka media yang berisi pesan (*stegotext*) sukar dibedakan oleh mata dengan *covertext*-nya. Jika *covertext* berupa audio (misalnya berkas mp3, wav, midi, dan sebagainya), maka indera telinga tidak dapat mendeteksi perubahan pada audio *stegotext*-nya.
2. *Fidelity*. Mutu media penampung tidak berubah banyak akibat penyisipan. Perubahan tersebut tidak dapat dipersepsi secara langsung. Misalnya, jika *covertext* berupa citra, maka penyisipan pesan membuat citra *stegotext* sukar dibedakan oleh mata dengan citra *covertex*-nya. Jika *covertext* berupa audio

(misalnya berkas mp3, wav, midi, dan sebagainya), maka audio *stegotext* tidak rusak dan indera telinga tidak dapat mendeteksi perubahan tersebut.

3. *Recovery*. Pesan yang disembunyikan harus dapat diungkapkan kembali (*reveal*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu pesan rahasia di dalam *stegotext* harus dapat diambil kembali untuk digunakan lebih lanjut.

2.2.1 Algoritma GifShuffle

Salah satu algoritma steganografi yang dapat digunakan untuk menyisipkan pesan pada citra berformat GIF adalah algoritma GifShuffle. Algoritma GifShuffle dibuat oleh Matthew Kwan, seorang sarjana ilmu komputer dari University of Melbourne yang kemudian mendirikan sebuah perusahaan yang bergerak di bidang enkripsi surat elektronik bernama Unicypt Pty Ltd.

GifShuffle akan melakukan penukaran terhadap palet warna dari sebuah berkas GIF. Sehingga dapat diartikan bahwa GifShuffle adalah algoritma yang memanfaatkan penukaran posisi ke 256 palet warna dalam berkas citra berformat GIF. Hal tersebut aman dilakukan karena dua buah berkas GIF dengan palet warna yang berbeda akan ditampilkan secara sama persis [PEN-05].

2.2.2 Skema GifShuffle

Algoritma GifShuffle mengganti susunan palet warna dari sebuah berkas GIF. Berkas dengan format GIF mengandung 256 palet warna maka dapat disimpulkan bahwa total penyimpanan maksimum dari format ini adalah 1683 bit [KWA-03].

Langkah-langkah penyisipan pada algoritma GifShuffle yang dilakukan pada penelitian ini dilakukan modifikasi terhadap pengambilan warna gif dikarenakan peneliti mengalami kesulitan untuk mengambil palet warna gif sehingga palet warna yang digunakan disini diambil dari warna gambar. Proses iterasi berhenti pada saat warna yang digunakan sudah dapat memuat *chipertext* dan hasil warna disimpan pada palet warna. Pada proses penyimpanan warna yang sudah digunakan disimpan pada pixel awal hingga sejumlah warna yang digunakan. Langkah-langkah proses penyisipan adalah sebagai berikut:

1. Pesan yang akan disisipkan dirubah ke dalam sebuah bentuk biner dengan representasi 1/0.
2. Menambahkan angka 1 di depan kumpulan representasi biner yang tadi diperoleh dan dikonversi menjadi bentuk desimal. Biasanya langkah ini akan menghasilkan sebuah bilangan yang sangat besar karena konversi dari biner yang besar. Namakan bilangan yang diperoleh ini sebagai *M*.
3. Menghitung jumlah warna yang terkandung dalam berkas GIF yang ingin disisipkan. Namakan jumlah yang diperoleh ini sebagai *N*. Apabila *M>N!-1* maka pesan yang ingin disisipkan berukuran terlalu besar sehingga proses penyisipan tidak dapat dilakukan.
4. Mengurutkan warna dalam palet warna sesuai dengan urutan yang "natural". Setiap warna dengan format RGB dikonversikan ke bilangan desimal dengan aturan (merah* 65536 + hijau * 256 + biru). Kemudian diurutkan berdasarkan besar bilangan desimal yang mewakili warna tersebut.
5. Melakukan iterasi terhadap variabel *i* dengan nilai *i* dari 1 sampai *M*=0. Setiap warna dengan urutan *N-i* dipindahkan keposisi baru yaitu *M* mod *i*, kemudian *M* dibagi dengan *i*.
6. Kemudian palet warna yang baru hasil iterasi pada langkah 5 dimasukkan ke dalam palet warna berkas GIF. Apabila ada sebuah tempat yang diisi oleh 2 buah warna maka warna yang sebelumnya menempati tempat tersebut akan digeser satu tempat ke samping.
7. Kemudian berkas GIF ini akan dikompresi ulang dengan palet warna yang baru untuk menghasilkan berkas yang baru dengan ukuran dan gambar yang sama namun telah disisipi pesan.

Langkah-langkah ekstraksi pada algoritma GifShuffle yang dilakukan pada penelitian ini dilakukan modifikasi pada pengambilan warna. Warna yang digunakan diambil berdasarkan jumlah warna pada proses penyisipan. Langkah-langkah algoritma GifShuffle pada proses ekstraksi pesan adalah sebagai berikut:

1. Mengambil warna sesuai dengan jumlah warna pada proses penyisipan.
2. Memberikan penomoran sesuai dengan posisinya untuk setiap warna pada palet warna citra GIF yang telah disisipkan pesan. Inisialisasi awal *N*=kode gambar
2. Warna kemudian diurutkan berdasarkan rumus:

(Nilai merah * 65536) + (nilai hijau * 256) + (nilai biru)

3. M diberi nilai 0.
4. Iterasi variabel *i* dari 0 sampai *N-1*.

   M = *M**(*N*–*i*) + posisi warna ke-*i*

   Iterasi variabel *j* dari *i*+1 sampai *N-1*.

   Jika nilai posisi warna ke *j*> nilai posisi warna ke *i*, maka nilai posisi warna ke *i* dikurangkan dengan 1.
5. Setelah nilai m diperoleh, maka nilai m dikonversikan ke bilangan biner untuk memperoleh pesan asli.

2.3 Kriptografi

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar aman. "*Crypto*" berarti rahasia (*secret*) dan "*graphy*" berarti tulisan (*writing*) [RAH-01].

Kriptografi adalah suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga. Kriptografi adalah ilmu pengetahuan dan seni menjaga pesan agar tetap aman [SCH-94].

Konsep kriptografi sendiri telah lama digunakan oleh manusia misalnya pada peradaban Mesir dan Romawi walau masih sangat sederhana. Prinsip-prinsip yang mendasari kriptografi yakni:

a) Otentik. Penerima pesan dapat memastikan keaslian pengirimnya. Orang lain tidak dapat berpura-pura menjadi pengirim pesan.
b) Integritas. Penerima harus dapat melakukan pengecekan terhadap keaslian pesan yang diterima. Seorang penyusup seharunya tidak dapat melakukan perubahan selama data masih dalam perjalanan.
c) *Non-repudiation*. Pengirim seharusnya tidak dapat menyangkal pesan yang dikirimkan bukan berasal dari pengirim tersebut. Tanpa kriptografi, seseorang dapat mengelak bahwa dialah pengirim pesan yang sesungguhnya.
d) Otoritas. Informasi yang berada pada sistem seharusnya hanya dapat dimodifikasi oleh pihak yang berwenang [WAH-03].

Berbeda dengan kriptografi klasik yang menitikberatkan kekuatan pada kerahasiaan algoritma yang digunakan (yang artinya apabila algoritma yang digunakan telah diketahui maka pesan sudah jelas "bocor" dan dapat diketahui isinya oleh siapa saja yang mengetahui algoritma tersebut), kriptografi modern lebih menitikberatkan pada kerahasiaan kunci yang digunakan pada algoritma tersebut (oleh pemakainya) sehingga algoritma tersebut dapat saja disebarkan ke kalangan masyarakat tanpa takut kehilangan kerahasiaan bagi para pemakainya.

Berikut adalah istilah-istilah yang digunakan dalam bidang kriptografi:

a) *Plaintext* (M) adalah pesan yang hendak dikirimkan (berisi data asli).
b) *Ciphertext* (C) adalah pesan terenkrip yang merupakan hasil enkripsi.
c) Enkripsi (fungsi E) adalah proses pengubahan *plaintext* menjadi *ciphertext*.
d) Dekripsi (fungsi D) adalah kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga berupa data awal/asli.
e) Kunci adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi.

Proses utama dalam suatu algoritma kriptografi adalah enkripsi dan dekripsi. Enkripsi merubah sebuah *plaintext* ke dalam bentuk *ciphertext*. Pada mode *Elekctronic Code Book* (ECB), sebuah blok pada *plaintext* dienkripsi ke dalam sebuah blok *ciphertext* dengan panjang blok yang sama.

Blok *cipher* memiliki sifat bahwa setiap blok harus memiliki panjang yang sama (misalnya 128 bit). Namun apabila pesan yang dienkripsi memiliki panjang blok terakhir tidak tepat 128 bit, maka diperlukan mekanisme *padding*, yaitu penambahan bit untuk menggenapi menjadi panjang blok yang sesuai.

Padding pada blok terakhir bisa dilakukan dengan berbagai macam cara, misalnya dengan penambahan bit-bit tertentu. Salah satu contoh penerapan *padding* dengan cara menambahkan jumlah total *padding* sebagai byte terakhir pada blok terakhir *plaintext*. Misalnya panjang blok adalah 128 bit (16 byte) dan pada blok terakhir terdiri dari 88 bit (11 byte) sehingga jumlah *padding* yang diperlukan adalah 5 byte, yaitu dengan menambahkan angka nol sebanyak 4 byte, kemudian menambahkan angka 5 sebanyak satu byte. Cara lain dapat juga menggunakan penambahan karakter *end-of-file* pada byte terakhir lalu diberi *padding* setelahnya.

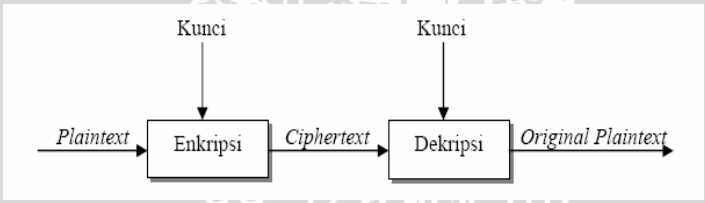
Dekripsi merupakan proses kebalikan dari proses enkripsi, merubah *ciphertext* kembali ke dalam bentuk *plaintext*. Untuk menghilangkan *padding* yang diberikan pada saat proses enkripsi, dilakukan berdasarkan informasi jumlah *padding* yaitu angka pada byte terakhir [MUN-06].

2.3.1 Teknik Kriptografi

Pesan atau informasi yang dapat dibaca disebut sebagai *ciphertext*. Teknik yang membuat pesan menjadi tidak dapat dibaca adalah enkripsi. Proses enkripsi membuat pesan yang awalnya dapat dibaca menjadi tidak dapat dibaca, pesan hasil dari enkripsi tersebut dikenal dengan istilah *ciphertext*. Dekripsi merupakan proses kebalikan dari enkripsi, yaitu membuat *ciphertext* menjadi *ciphertext*.

Kriptosistem adalah algoritma kriptografi, ditambah seluruh kemungkinan *plaintext*, *ciphertext*, dan kunci-kuncinya. *Plaintext* dapat berupa aliran bit, file teks, bitmap, aliran suara yang didigitasi, gambar video digital dan sebagainya.

Dalam proses yang ditunjukkan pada Gambar 2.1, kriptosistem terdiri dari dua proses, proses yang pertama adalah enkripsi, *plaintext* disandikan dengan suatu kunci lalu dihasilkan *ciphertext*. Proses yang kedua adalah dekripsi, *ciphertext* diuraikan dengan menggunakan kunci sehingga menghasilkan *plaintext* yang sama dengan sebelumnya.



Gambar 2.1 Kriptosistem

Sumber: [WAH-03]

Setiap kriptosistem yang baik memiliki karakteristik sebagai berikut:

a) Keamanan sistem bukan pada kerahasian algoritma yang digunakan, tetapi terletak pada kerahasiaan kunci.
b) Kriptosistem memiliki ruang kunci (*keyspace)* yang besar.
c) Kriptosistem menghasilkan *ciphertext* yang terlihat acak dalam setiap test statistiknya.

d) Kriptosistem mampu menahan semua serangan yang telah ada sebelumnya [WAH-03].

2.3.2 Keamanan Kriptografi

Komponen-komponen yang menjadi aspek keamanan dalam ruang lingkup kriptografi adalah sebagai berikut:

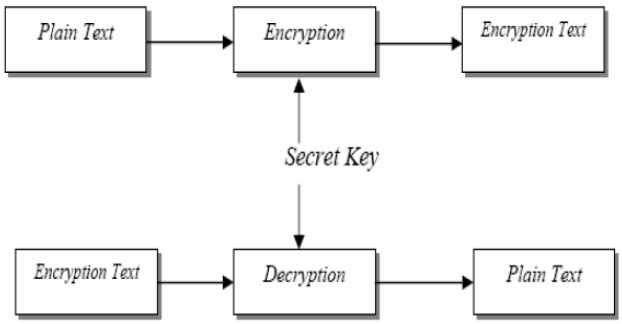
a) Autentikasi. Penerima pesan dapat memastikan keaslian pengirimnya. Orang lain tidak dapat berpura-pura menjadi pengirim pesan.
b) Integritas. Penerima harus dapat melakukan pengecekan terhadap keaslian pesan yang diterima. Seorang penyusup seharunya tidak dapat melakukan perubahan selama data masih dalam perjalanan.
c) *Non-repudiation*. Pengirim seharusnya tidak dapat menyangkal pesan yang dikirimkan bukan berasal dari pengirim tersebut. Tanpa kriptografi, seseorang dapat mengelak bahwa dialah pengirim pesan yang sesungguhnya.
d) Otoritas. Informasi yang berada pada sistem seharusnya hanya dapat dimodifikasi oleh pihak yang berwenang [WAH-03].

2.3.3 Algoritma Kriptografi

Terdapat dua jenis algoritma kriptografi berdasar jenis kuncinya, yaitu:

1. Algoritma Simetris

Algoritma simetris disebut juga sebagai algoritma konvensional, yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan dekripsinya. Pada Gambar 2.2 diperlihatkan skema algoritma simetris yang hanya membutuhkan satu buah kunci yang sama. Keamanan algoritma simetris tergantung pada kuncinya. Algoritma simetris sering juga disebut algoritma kunci rahasia, algoritma kunci tunggal atau algoritma satu kunci. Dua kategori yang termasuk pada algoritma simetris ini adalah algoritma *block cipher* dan *stream cipher*.



Gambar 2.2 Algoritma Kriptografi Simetris

Sumber: [MUN-06]

Kelebihan algoritma kriptografi simetris adalah:

a) Algoritma ini dirancang sehingga proses enkripsi/dekripsi membutuhkan waktu yang singkat.
b) Ukuran kunci relatif lebih pendek.
c) Algoritmanya bisa menghasilkan *cipher* yang lebih kuat.
d) Autentikasi pengiriman pesan langsung diketahui dari *ciphertext* yang diterima, karena kunci hanya diketahui oleh pengirim dan penerima pesan saja.

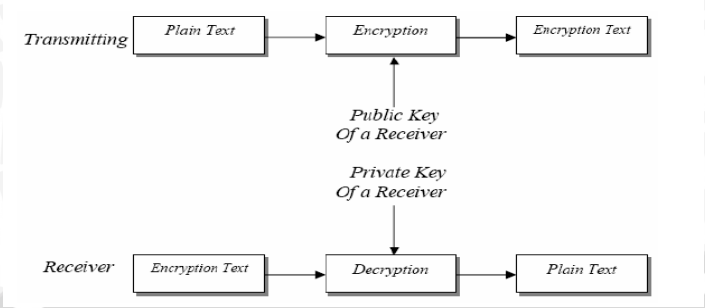
Kelemahan algoritma kriptografi simetris adalah:

a) Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini.
b) Kunci harus sering diubah, mungkin pada setiap sesi komunikasi [MUN-06].

2. Algoritma Asimetris

Algoritma asimetris atau biasa disebut algoritma kunci publik dirancang sedemikian sehingga kunci yang digunakan untuk mengenkripsi dan mendekripsi berbeda. Pada Gambar 2.3 diperlihatkan skema algoritma asimetris yang menggunakan dua buah kunci. Sehingga kunci dekripsi tidak dapat dihitung dari kunci enkripsi. Algoritma tersebut disebut *public-key* karena kunci enkripsi dapat dibuat secara *public*. Orang asing dapat menggunakan kunci enkripsi tersebut untuk mengenkripsi sebuah pesan, tetapi hanya orang tertentu dengan kunci dekripsi sepadan dapat mendekripsi pesan tersebut. Dalam sistem ini kunci

enkripsi sering disebut *public key* sedangkan key dekripsi sering disebut *private key*.



Gambar 2.3 Algoritma Kriptografi Asimetris

Sumber: [MUN-06]

Kelebihan algoritma kriptografi asimetris adalah:

a) Hanya *Private key* yang harus benar-benar rahasia/aman.
b) Sangat jarang untuk perlu merubah *public key* dan *private key.*

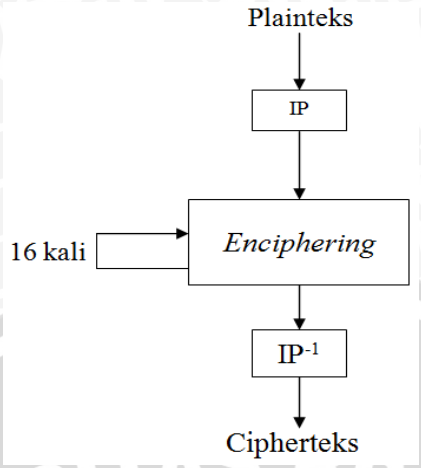
Kelemahan algoritma kriptografi asimetris adalah:

a) Ukuran kunci lebih besar dari pada algoritma kunci simetris.
b) Tidak adanya jaminan bahwa *public key* benar-benar aman [MUN-06].

2.4 Algoritma DES

DES termasuk ke dalam sistem kriptografi simetris dan tergolong jenis *cipher* blok. DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit *plaintext* menjadi 64 bit *ciphertext* dengan menggunakan 56 bit kunci internal (*internal key*) atau sub-kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) yang panjangnya 64 bit. Skema global dari algoritma DES ditunjukkan pada Gambar 2.4. Skema global dari algoritma DES adalah sebagai berikut:

a) Blok *plaintext* dipermutasi dengan matriks permutasi awal (*initial permutation* atau IP).
b) Hasil permutasi awal kemudian di-*enciphering* sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
c) Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP^{-1}) menjadi blok *ciphertext*.



Gambar 2.4 Skema Global Algoritma DES

Sumber: [BAN-06]

Permutasi awal ditunjukkan pada Gambar 2.5. Skema algoritma DES ditunjukkan pada Gambar 2.6.

(a) Matriks Masukan

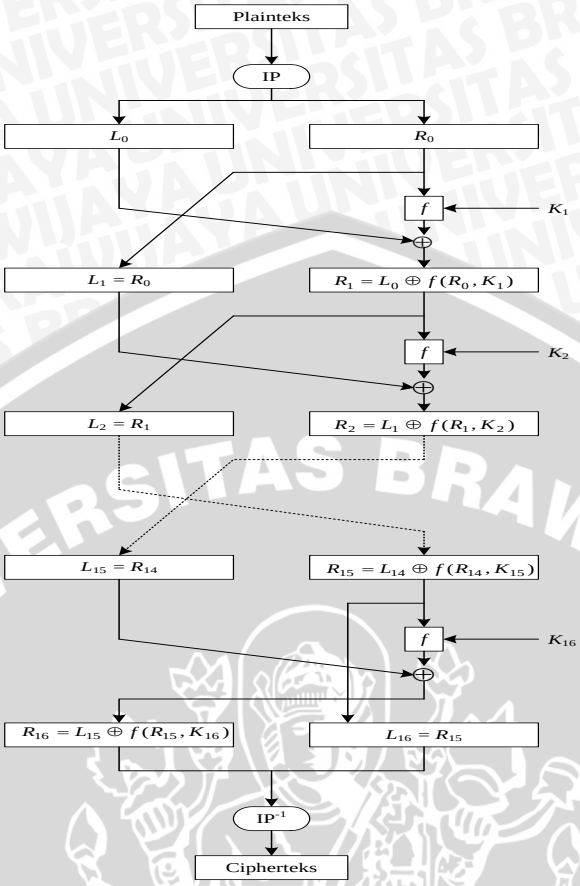
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Matriks Inisial Permutasi (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Gambar 2.5 Permutasi Awal DES

Sumber: [BAN-06]

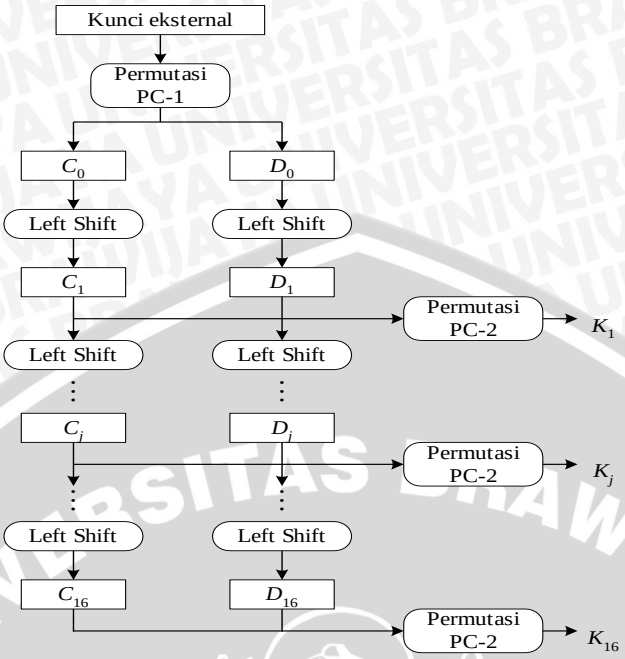


Gambar 2.6 Skema Algoritma DES

Sumber: [BAN-06]

(L_{16}, R_{16}) merupakan keluaran dari putaran ke-16, maka (R_{16}, L_{16}) merupakan pra *ciphertext* dari *enciphering* ini. *Ciphertext* yang sebenarnya diperoleh dengan melakukan permutasi awal balikan, IP^{-1}, terhadap blok pra *ciphertext* [DHI-00].

Pembangkitan kunci internal digunakan pada setiap putaran. Proses pembangkitan kunci internal ditunjukkan pada Gambar 2.7.



Gambar 2.7 Proses Pembangkitan Kunci-kunci Internal DES

Sumber: [BAN-06]

Permutasi dengan PC-1 dilakukan setelah proses pembentukan 8 kelompok. Permutasi PC-1 ditunjukkan pada Gambar 2.8.

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Gambar 2.8 PC-1

Sumber: [BAN-06]

Pergeseran bit (*Left Shift*) tiap putaran berbeda-beda. Jumlah bit yang digeser setiap putaran ditunjukkan pada Gambar 2.9.

Jumlah pergeseran-bit pada tiap putaran

Putaran, *i*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Jumlah pergeseran-	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Gambar 2.9 Jumlah Pergeseran-bit Pada Tiap Putaran

Sumber: [BAN-06]

Pembentukan kunci internal dilakukan dengan melakukan permutasi *C,D* setiap putaran dengan PC-2. PC-2 ditunjukkan pada Gambar 2.10.

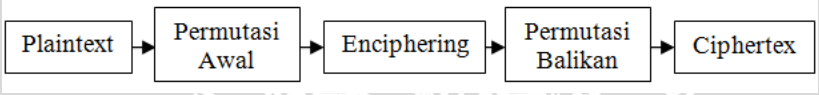
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Gambar 2.10 Kunci PC-2

Sumber: [BAN-06]

2.4.1 Enkripsi DES

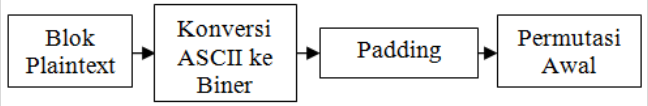
Proses enkripsi DES untuk menghasilkan *ciphertext* melalui beberapa proses, yaitu permutasi awal, proses *enciphering*, dan proses permutasi balikan. Proses enkripsi DES dijelaskan pada Gambar 2.11.



Gambar 2.11 Blok Diagram Enkripsi DES

Sumber: [BAN-06]

Permutasi awal dilakukan terhadap blok *plaintext* sebelum putaran pertama. Hal ini bertujuan untuk mengacak *plaintext* sehingga urutan bit-bit didalamnya berubah. Sebelum melakukan proses permutasi awal, terdapat beberapa proses yang harus dilakukan. Proses-proses tersebut dijelaskan pada Gambar 2.12.



Gambar 2.12 Blok Diagram Proses Permutasi Awal

Sumber: [BAN-06]

Penjelasan dari Gambar 2.12 yaitu sebagai berikut:

a) Blok *Plaintext*

Algoritma DES termasuk kedalam sistem kriptografi simetris dan tergolong jenis *cipher* blok, maka diperlukan proses pembagian blok dari

sebuah *plaintext*. Algoritma ini beroperasi pada blok berukuran 64 bit. Pembagian blok dengan menggunakan acuan kode ASCII yang membutuhkan 8 bit data, maka ukuran 1blok data adalah 8 karakter.

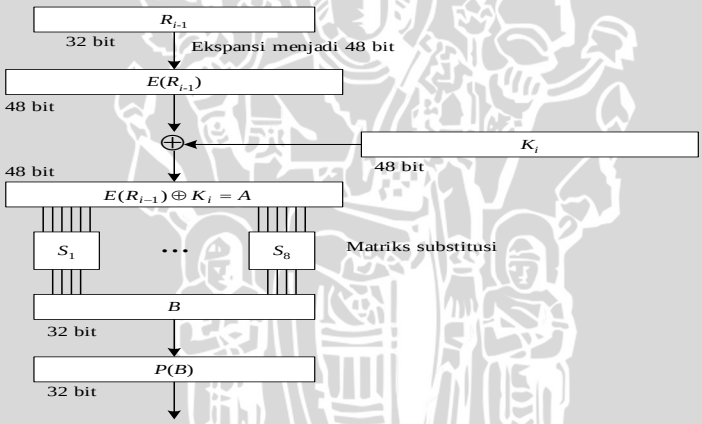
b) Konversi ASCII ke Biner

Konversi ini dilakukan dikarenakan pada permutasi awal dibutuhkan bilangan biner sebanyak 64 bit, sedangkan pada *plaintext* masih berupa karakter ASCII.

c) *Padding*

Proses *padding* adalah proses menambahan bit-bit isian pada blok terakhir dari inputan *plaintext*.

Proses *enciphering* terhadap blok *plaintext* dilakukan setelah permutasi awal. Setiap blok *plaintext* mengalami 16 kali putaran *enciphering*. Diagram komputasi fungsi *f* diperlihatkan pada Gambar 2.13.



Gambar 2.13 Rincian Komputasi Fungsi f

Sumber: [BAN-06]

E adalah fungsi ekspansi yang memperluas blok R_{i-1} yang panjangnya 32-bit menjadi blok 48 bit. Selanjutnya, hasil ekpansi, yaitu $E(R_{i-1})$, yang panjangnya 48 bit di-XOR-kan dengan K_i yang panjangnya 48 bit menghasilkan vektor *A* yang panjangnya 48-bit: $E(R_{i-1}) \oplus K_i = A$

Vektor *A* dikelompokkan menjadi 8 kelompok, masing-masing 6 bit, dan menjadi masukan bagi proses substitusi. Proses substitusi dilakukan dengan menggunakan delapan buah kotak-S (*S-box*), S_1 sampai S_8. Setiap kotak-S menerima masukan 6 bit dan menghasilkan keluaran 4 bit. Kelompok 6 bit

pertama menggunakan S_1, kelompok 6 bit kedua menggunakan S_2, dan seterusnya.

S_1:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2:

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3:

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4:

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5:

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	16
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6:

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7:

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8:

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Gambar 2.14 S-box

Sumber: [BAN-06]

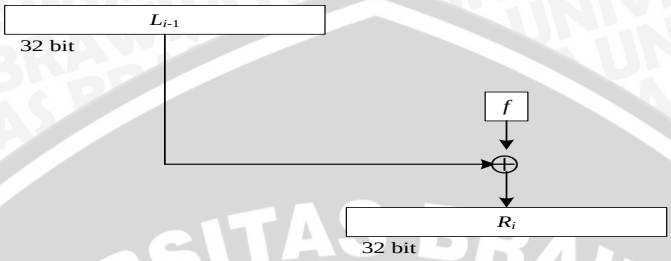
Keluaran proses substitusi adalah vektor *B* yang panjangnya 48 bit. Vektor *B* menjadi masukan untuk proses permutasi. Tujuan permutasi adalah untuk mengacak hasil proses substitusi kotak-S. Bit-bit *P*(*B*) merupakan keluaran dari fungsi *f*. Akhirnya, bit-bit *P*(*B*) di-XOR-kan dengan L_{i-1} untuk mendapatkan R_i : $R_i = L_{i-1} \oplus P(B)$ Jadi, keluaran dari putaran ke-*i* adalah $(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus P(B))$. *P-box* ditunjukkan pada Gambar 2.15. Skema perolehan R_i ditunjukkan pada Gambar 2.16.

Tabel (h) P-box

16	7	20	21	29	12	28	17	1	15	23	26	5	8	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Gambar 2.15 P-box

Sumber: [BAN-06]



Gambar 2.16 Skema Perolehan R_i

Sumber: [BAN-06]

Proses permutasi balikan dilakukan dilakukan setelah 16 kali putaran terhadap gabungan blok kiri dan blok kanan. Proses permutasi menggunakan matriks permutasi awal balikan (*inverse initial permutation* atau IP^{-1}) [DHI-00]. Matriks permutasi awal balikan ditunjukkan pada Gambar 2.17.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Gambar 2.17 IP^{-1}

Sumber: [BAN-06]

2.4.2 Dekripsi DES

Proses dekripsi terhadap *ciphertext* merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah K_1, K_2, …, K_{16}, maka pada proses dekripsi urutan kunci yang digunakan adalah K_{16}, K_{15}, …, K_1.

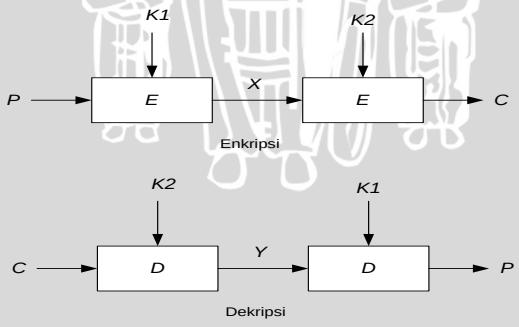
Blok (R_{16}, L_{16}) diperoleh dengan mempermutasikan *ciphertext* dengan matriks permutasi IP^{-1}. Prakeluaran dari *deciphering* adalah adalah (L_0, R_0). Dengan permutasi awal IP akan didapatkan kembali blok *plaintext* semula.

Selama *deciphering*, K_{16} dihasilkan dari (C_{16}, D_{16}) dengan permutasi. (C_{16}, D_{16}) tidak dapat diperoleh langsung pada permulaan *deciphering*. Tetapi karena (C_{16}, D_{16}) = (C_0, D_0), maka K_{16} dapat dihasilkan dari (C_0, D_0) tanpa perlu lagi melakukan pergeseran bit. Selanjutnya K_{15} dihasilkan dari (C_{15}, D_{15}). (C_{15}, D_{15}) diperoleh dengan menggeser C_{16} (yang sama dengan C_0) dan D_{16} (yang sama dengan C_0) satu bit ke kanan. Sisanya, K_{14} sampai K_1 dihasilkan dari (C_{14}, D_{14}) sampai (C_1, D_1). (C_{i-1}, D_{i-1}) diperoleh dengan menggeser C_i dan D_i dengan cara yang sama, tetapi pergeseran kiri (*left shift*) diganti menjadi pergeseran kanan (*right shift*) [DHI-00].

2.5 Algoritma 3DES

Algoritma 3DES adalah suatu algoritma pengembangan dari algoritma DES. Algoritma 3DES menggunakan ukuran kunci yang lebih panjang yaitu 3 kunci yang panjangnya 168-bit (masing-masing panjangnya 56-bit). Algoritma 3DES melakukan enkripsi dengan implementasi algoritma DES sebanyak 3 kali.

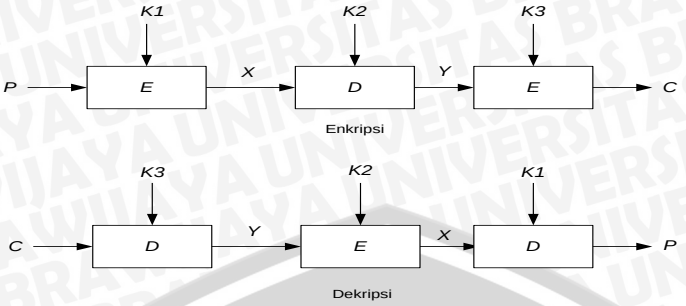
2TDES menggunakan 2 buah kunci eksternal, K_1 dan K_2 ditunjukkan pada Gambar 2.18.



Gambar 2.18 Block Diagram 3DES dengan 2 Kunci

Sumber: [BAN-06]

3TDES menggunakan 3 buah kunci eksternal, K_1, K_2, dan K_3 ditunjukkan pada Gambar 2.19



Gambar 2.19 Block Diagram 3DES dengan 3 Kunci

Sumber: [BAN-06]

Keterangan:

E: Enkripsi dengan menggunakan algoritma DES

D: Dekripsi dengan menggunakan algoritma DES

P: *Plaintext*

C: *Ciphertext*

K1: Kunci ke-1

K2: Kunci ke-2

K3: Kunci ke-3

X: Hasil enkripsi

Y: Hasil dekripsi

2.6 Citra Digital

Citra adalah gambar pada bidang 2 dimensi. Dalam tinjauan matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Ketika sumber cahaya menerangi objek, objek memantulkan kembali sebagian cahaya tersebut. Pantulan ini ditangkap oleh alat-alat pengindera optik, misalnya mata manusia, kamera, *scanner* dan sebagainya. Bayangan objek tersebut akan terekam sesuai intensitas pantulan cahaya. Ketika alat optik yang merekam pantulan cahaya itu merupakan mesin digital, misalnya kamera digital, maka citra yang dihasilkan merupakan citra digital.

Citra digital disimpan sebagai suatu file dengan format tertentu. Format citra tersebut menunjukan cara sebuah citra digital disimpan, misalnya apakah dengan suatu kompresi atau tidak. Contoh format citra digital adalah .bmp, .jpg, .png, .tif dan sebagainya. Ukuran citra digital dinyatakan dalam *pixel*. Umumnya, nilai

setiap *pixel* merupakan kuantisasi harga intensitas cahaya. Dengan demikian, suatu citra digital dapat dipandang sebagai sebuah matriks yang elemen-elemennya menunjukkan intensitas cahaya terkuantisasi. Bedanya terletak pada urutan penyebutan angka ukuran tersebut. Citra digital dengan ukuran 92x112 *pixel* sebenarnya merupakan sebuah matriks dengan ukuran 112x92, dimana 112 merupakan banyaknya baris dan 92 merupakan banyaknya kolom.

Citra merupakan fungsi kontinyu dari intensitas cahaya pada bidang dua dimensi, dinyatakan dalam bentuk fungsi: $f(x, y)$ dimana:

(x, y) : koordinat pada bidang dua dimensi

$f(x, y)$: *intensitas cahaya (brigthness) pada titik* (x, y),

besarnya $0 < f(x, y) < \infty$

Penentuan koordinat titik pada citra ditunjukkan pada Gambar 2.20.

Gambar 2.20 Penentuan Koordinat Titik Pada Citra

Digitalisasi adalah proses representasi citra dari fungsi kontinyu menjadi nilai-nilai diskrit. Citra yang dihasilkan dari digitalisasi disebut citra digital (*digital image*). Umumnya citra digital berbentuk empat persegi panjang dan dimensi ukurannya dinyatakan dengan *panjang* x *lebar*.

Pixel (*picture element)* merupakan citra digital yang berukuran *N x M* biasanya dinyatakan dengan matriks berukuran *N* baris dan *M* kolom, dan masing-masing elemen pada citra digital. Kuantisasi merupakan pembagian skala keabuan (0,L) menjadi G level yang dinyatakan dengan suatu harga bilangan bulat (integer), biasanya G diambil perpangkatan dari $2.G = 2^m$ dimana G adalah derajat keabuan dan m adalah bilangan bulat positif.

Skala Keabuan	Rentang Nilai Keabuan	*Pixel Depth*
2^1(2 nilai)	0,1	1 bit
2^2(4 nilai)	0, sampai 7	2 bit
2^3(16 nilai)	0, sampai 15	3 bit
2^8(256 nilai)	0, sampai 255	8 bit

Hitam dinyatakan dengan nilai derajat keabuan terendah, sedangkan putih dinyatakan dengan nilai derajat keabuan tertinggi, misalnya 15 untuk 16 level. Jumlah bit yang dibutuhkan untuk merepresentasikan nilai keabuan *pixel* disebut *pixel depth*. Sehingga citra dengan kedalaman 8 bit sering disebut citra-8 bit. Besarnya derajat keabuan yang digunakan untuk menentukan resolusi kecerahan dari citra yang diperoleh. Semakin banyak jumlah derajat keabuan (jumlah bit kuantisasinya makin banyak), semakin bagus gambar yang diperoleh karena kemenerusan derajat keabuan akan semakin tinggi sehingga mendekati citra aslinya [SUT-09].

Menurut jenisnya citra dibedakan menjadi citra warna, citra *grayscale*, dan citra biner.

a) Citra warna adalah setiap titik yang mempunyai warna yang spesifik yang merupakan kombinasi 3 warna dasar yaitu merah, hijau dan biru. Format citra ini sering disebut dengan citra RGB. Setiap titik pada citra warna membutuhkan data 3 *byte*.
b) Citra *grayscale* adalah citra yang terdiri dari 8 bit yang hanya menggunakan warna pada tingkatan hitam untuk nilai *pixel* 0, sampai dengan putih dengan nilai *pixel* 1.
c) Citra biner (monokrom) adalah citra yang terdiri dari 1 bit yang hanya memiliki 2 kemungkinan nilai pada setiap *pixel*, yaitu 0 dan 1. Nilai 0 adalah *background points*, biasanya bukan merupakan bagian dari citra sesungguhnya. Sedangkan nilai 1 adalah *region points*, yaitu bagian dari citra sebenarnya [AHM-05].

2.6.1 Citra GIF

Graphics Interchange Format (GIF) adalah sebuah format berkas citra yang diperkenalkan pada tahun 1987 oleh CompuServe untuk menggantikan format RLE yang hanya mampu menampilkan gambar dengan warna hitam dan putih saja.

GIF adalah salah satu format berkas citra yang paling sering ditemui di dunia digital. Hal ini terjadi karena format ini berukuran relatif kecil. Sebagai contoh untuk citra yang sama, berkas dengan format GIF dapat berukuran lebih

kecil jika dibandingkan dengan format JPG. Hal ini disebabkan karena file GIF hanya menggunakan 256 palet warna. Sehingga tentunya ukuran file akan lebih kecil. Namun 256 palet warna tersebut tidak mutlak hanya 256 warna tertentu. Warna tersebut dapat dipilih dari 24-bit palet warna RGB atau dapat disimpulkan bahwa berkas dengan format GIF akan membuang palet warna yang tidak diperlukan dan mengambil hanya 256 palet warna yang diperlukan.

Format GIF juga memiliki kekurangan. Format ini jarang sekali digunakan untuk citra fotografi karena hanya dapat menampung 256 buah warna sementara citra fotografi biasanya menggunakan lebih dari 256 warna. Jika citra fotografi direpresentasikan dalam format GIF maka citra tersebut akan mengalami penurunan kualitas yang banyak [PEN-05].

2.7 Avalanche Effect

Avalanche Effect adalah perubahan satu bit pada *plaintext* atau key yang menyebabkan perubahan yang signifikan terhadap *ciphertext*. Suatu algoritma dikatakan memiliki nilai *Avalanche Effect* yang baik jika perubahan satu bit saja pada input menghasilkan perubahan sekitar setengah jumlah bit pada output-nya. Nilai *Avalanche Effect* yang berkisar antara 45% hingga 60% menunjukkan algoritma tersebut baik untuk digunakan. Hal ini dikarenakan perubahan tersebut berarti membuat perbedaan yang cukup sulit untuk kriptanalisis melakukan serangan. Nilai A*valanche Effect* dirumuskan dengan persamaan 2.1 [END-08].

$$AvalancheEffect = \frac{\Sigma bit\_berubah}{\Sigma bit\_total} \times 100\% \qquad (2.1)$$

2.8 Peak Signal to Noise Ratio (PSNR)

PSNR adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. Informasi yang hilang akibat proses enkripsi seharusnya seminimal mungkin, sehingga kualitas citra digital setelah didekripsi akan sama bagusnya sebelum citra di enkripsi. PSNR merupakan salah satu cara untuk mengukur tingkat kesalahan akibat hilangnya informasi dan biasanya diukur dalam satuan *decibel* (dB). PSNR digunakan untuk mengetahui perbandingan kualitas citra sebelum dienkripsi (citra asli) dan sesudah didekripsi. Nilai PSNR yang tinggi akan lebih baik. Perhitungan

nilai PSNR dari citra digital berukuran N x M, dilakukan menggunakan Persamaan 2.2 [LIN-05].

$$\text{PSNR} = 20xLog10\left(255 \Big/ \sqrt{\frac{1}{MN}\sum_{Y=1}^{M}\sum_{X=1}^{N}[I(x,y)-I'(x,y)]^2}\right) \qquad (2.2)$$

Dimana:

m = panjang citra tersebut (dalam *pixel*)

n = lebar citra tersebut (dalam *pixel*)

(x,y) = koordinat masing-masing *pixel*

I = nilai bit pada citra asli

I' = nilai bit pada citra setelah didekripsi