

BAB II

TINJAUAN PUSTAKA

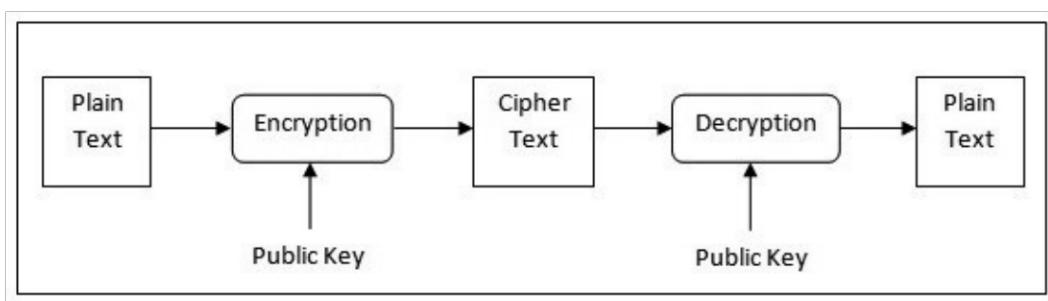
2.1 Kriptografi

2.1.1. Pengenalan Kriptografi

Kriptografi berasal dari bahasa Yunani yaitu ‘*crypto*’ yang berarti rahasia dan ‘*graphia*’ yang berarti tulisan, sehingga secara asal bahasa kriptografi berarti tulisan rahasia (*secret message*). Dalam kriptografi terdapat dua proses untuk mengamankan pesan yaitu enkripsi sebagai proses yang mengacak isi pesan rahasia (*plaintext*) menjadi pesan acak (*ciphertext*) dan dekripsi sebagai proses mengembalikan pesan yang diacak ke pesan rahasia. Kriptografi adalah ilmu mengenai proses enkripsi dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi sehingga pesan aman. Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. (Kromodimodimoeljo, S., 2010).

2.1.2. Kriptografi Kunci Simetris

Dalam kriptografi kunci simetris, jika seseorang mengetahui cara mengenkripsi pesan rahasia menjadi pesan acak, maka orang tersebut juga mengetahui cara mendekripsi pesan acak yang dihasilkan karena kunci enkripsi dan dekripsi menggunakan kunci yang sama (Odeh, A., dkk, 2015). Keamanan kriptografi kunci simetris terletak pada kerahasiaan kunci. Proses dari kriptografi kunci simetris dapat dilihat dari gambar di bawah ini:

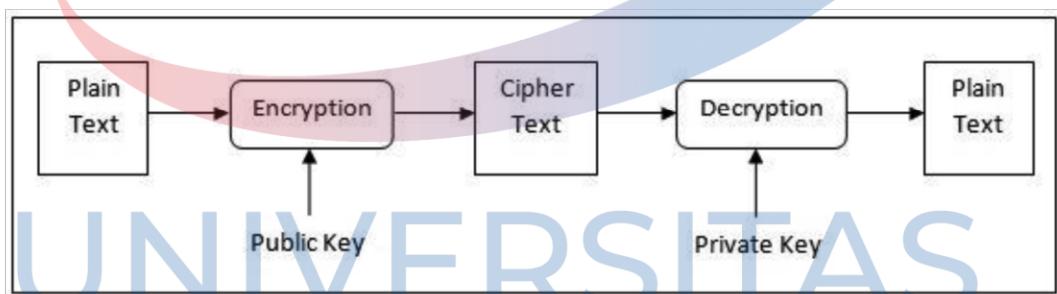


Gambar 2.1. Ilustrasi kriptografi kunci simetris (Odeh, A., dkk, 2015)

Pada gambar ilustrasi diatas menunjukkan bahwa pengirim dan penerima harus memiliki kunci yang sama. Agar komunikasi tetap aman kunci harus tetap dirahasiakan. Contoh kriptografi simetris adalah DES, Blowfish, Twofish, Triple-DES, IDEA, AES, dan lain-lain.

2.1.3. Kriptografi Kunci Asimetris

Untuk mengatasi kelemahan kriptografi kunci simetris pada tahun 1970an muncul konsep baru dalam kriptografi yaitu kriptografi kunci asimetris, pada kriptografi ini seseorang yang mengetahui cara mengenkripsi pesan rahasia belum tentu mengetahui juga cara mendekripsi pesan acak yang dihasilkan karena kunci enkripsi dan dekripsi yang digunakan berbeda (Odeh, A., dkk, 2015). Bagian dari kunci asimetris yaitu : Kunci enkripsi (*public key*), kunci yang boleh semua orang tau dan kunci rahasia dan kunci dekripsi (*private key*), kunci yang dirahasiakan. Proses dari kriptografi kunci asimetris dapat dilihat dari gambar di bawah ini:



Gambar 2.2. Ilustrasi kriptografi kunci asimetris (Odeh, A., dkk, 2015)

Pada gambar ilustrasi diatas menunjukkan bahwa kunci enkripsi (K1) dengan kunci dekripsi (K2) berbeda. Contoh kriptografi asimetris adalah RSA, Diffie-Helman, DSA, El-Gamal, NTRU dan lain-lain.

2.2. NTRU

2.2.1. Pengenalan NTRU

NTRU diperkenalkan pada tahun 1996 yang ditemukan oleh Jeffery Hoffstein, Jill Pipher, dan Joseph Silverman. Algoritma ini dipublikasikan pada tahun 1998 dan berganti nama menjadi NTRUEncrypt (Hoffstein, J., dkk, 1998).

Pendekatan yang dilakukan untuk menghasilkan algoritma NTRU adalah dengan menggunakan struktur matematika *polynomial ring* $Z[X]/(X^N - 1)$.

2.2.2. Polinomial Pada NTRU

Operasi-operasi aritmetika pada polinomial $R[x]$ adalah sebagai berikut (Security Innovation, 2014):

a. Penjumlahan Polinomial

Penjumlahan pada polinomial dilakukan dengan menjumlahkan semua suku dari polinomial (suku dengan pangkat yang sama dijadikan satu dengan menjumlahkan koefisien). Sebagai contoh $R[Z_3Z]$, misalkan $a = x^4+x^2+2x+2$ dan $b = x^5+2x^2+2x$, maka $a+b = x^5+x^4+x^2+x+2$. Hasil akhir dari penjumlahan polinomial, pada koefisien dilakukan aritmetika modulus 3.

b. Pengurangan polinomial

Pengurangan pada polinomial dilakukan dengan mengurangkan semua suku dari polinomial (suku dengan pangkat yang sama dijadikan satu dengan mengurangkan koefisien). Sebagai contoh $R[x]$, misalkan $a = x^4+x^2+2x+2$ dan $b = x^5+2x^2+2x$, langkah pertama ganti tanda setiap koefisien pada b sehingga menjadi $b = -x^5-2x^2-2x$, maka $a-b = (x^4+x^2+2x+2) + (-x^5-2x^2-2x) = 2x^5+x^4+2x^2+2$. Hasil akhir pada koefisien dilakukan aritmetika modulus 3.

c. Perkalian polinomial

Perkalian pada polinomial dilakukan dengan mengalikan semua suku dari polinomial pertama dengan setiap suku dari polinomial kedua dan menjumlahkan semua hasil perkalian. Sebagai contoh $R[x]$, misalkan $a = x^4+x^2+2x+2$ dan $b = x^5+2x^2+2x$, maka $a*b = x^4+x^2+2x+2 * x^5+2x^2+2x = x^9+x^7+x^6+x^5+2x^4+x^3+2x^2+x$.

Hasil akhir pada koefisien dilakukan aritmetika modulus 3.

d. Pembagian polinomial

Pembagian pada polinomial dilakukan dengan membagi semua suku dari polinomial pembilang dengan setiap suku dari polinomial penyebut dan mengurangkannya. Sebagai contoh $R[x]$, misalkan $a = 2x^3-3x^2+x+5$ dan $b = 2x^2-x-1$, maka $a:b = 2x^3-3x^2+x+5 : 2x^2-x-1 = x-1$ dengan sisa bagi $x+4$. Hasil akhir dari pembagian polinomial, pada koefisien dilakukan aritmetika modulus 3.

Polynomial ring pada *NTRU* memiliki operasi aritmetika modulus, penyederhanaan polinomial dan invers dari hasil penyederhanaan polinomial yang merupakan bagian penting dalam *NTRU*. *Ring* ini memiliki *irreducible polynomial* $X^N - 1$, dimana *irreducible polynomial* mirip dengan bilangan prima yaitu tidak bisa difaktorkan ke dalam bentuk polinomial lain serta tidak bisa habis dibagi kecuali dengan dirinya sendiri dan satu.

Aritmetika modulus pada *NTRU* digunakan pada hasil operasi-operasi yang terdapat dalam pada operasi polinomial dengan cara membagi setiap koefisien dan menyimpan sisanya sebagai hasil. *NTRU* menggunakan penyederhanaan polinomial pada hasil penggabungan dari operasi aritmetika modulus atau yang disebut dengan *truncated polynomial* dengan rumus X^{N-1} sehingga persamaan yang terbentuk menjadi $a = a_0 + a_1X + a_2X^2 + a_3X^3 + \dots + a^{N-1}X^{N-1}$, sedangkan pengembalian dari penyederhanaan polinomial sebelum di lakukan penyederhanaan untuk proses dekripsinya. Bentuk ini digunakan untuk pembangkitan pada polinomial. Contoh dari *truncated polynomial* yaitu jika diketahui $N = 3$ dan polinomial $a = x + x^2 + x^3$ dan $b = x + x^2$, maka hasil perkalian menjadi $x + x^2 + x^3 + x^5$, karena x^3 dan x^5 melebihi $(N - 1)$, maka lakukan *truncated polynomial* seperti berikut $x^3 = 1 * x^3 = 1$ dan $x^5 = (x^2) * (x^3) = x^2$, sehingga hasil dari perkalian menjadi $= 1 + x + 2x^2 + x^3$.

UNIVERSITAS MIKROSKIL

2.2.3. Proses pada *NTRU*

Pada *NTRU* terdapat parameter untuk melakukan proses enkripsi dan dekripsi yang ditunjukan di bawah ini:

N (Jumlah dari koefisien di dalam sebuah polinomial).

q (Modulus besar yang mereduksi koefisien)

p (Modulus kecil yang mereduksi koefisien)

f (Polinomial *private key*)

g (Digunakan untuk pembangkitan *public key*)

h (Polinomial *public key*)

r (Polinomial acak yang digunakan untuk mengenkripsi pesan)

d_f (Jumlah koefisien dengan nilai 1 dalam polinomial f)

Dimana N adalah sebuah bilangan prima $251 - 2499$, q adalah bilangan bulat $251 - 2499$ dan p adalah bilangan bulat yang tidak dapat membagi q ($p \nmid q$) juga diasumsikan nilainya 3. Untuk mencapai tingkat keamanan (*security level*) IEEE P1363.1TM/D10 merekomendasikan penggunaan parameter pada implementasi *NTRU* sesuai dengan tabel berikut.

Tabel 2.1. Tabel standar parameter tingkat keamanan *NTRU* (Whyte, W., dkk,

2008)

Parameter set	N	q	df	Known strength	Recommended security level
ees401ep1	401	2048	113	154.88	112
ees541ep1	541	2048	49	141.766	112
ees659ep1	659	2048	38	137.861	112
ees449ep1	449	2048	134	179.899	128
ees613ep1	613	2048	55	162.385	128
ees761ep1	761	2048	42	157.191	128
ees653ep1	653	2048	194	276.736	192
ees887ep1	887	2048	81	245.126	192
ees1087ep1	1087	2048	63	236.586	192
ees853ep1	853	2048	268	376.32	256
ees1171ep1	1171	2048	106	327.881	256
ees1499ep1	1499	2048	79	312.949	256

Tahapan melakukan proses enkripsi dan dekripsi adalah sebagai berikut (Gauravaram, P., dkk, 2014) :

- Plaintext* diubah dahulu ke dalam bentuk *polynomial*
- Tentukan parameter set yang digunakan untuk membangkitkan kunci enkripsi dan kunci dekripsi.
 - Bangkitkan sebuah polinomial f dimana jumlah dari koefisien dengan nilai 1 adalah sama dengan parameter d_f dan f adalah dapat di inverskan dengan modulo p dan modulo q .
 - Bangkitkan sebuah polinomial acak g .
 - Untuk polinomial f temukan f^1 modulo q dan f^1 modulo p dimana $f * f_q \equiv 1 \pmod{q}$ dan $f * f_p \equiv 1 \pmod{p}$. (2.1)
 - Hitung $h = p * f_q * g \pmod{q}$. (2.2)
 - *Public key* adalah h dan *private key* adalah f .
- Pesan di enkripsi dengan *NTRU* menggunakan kunci enkripsi, berikut langkahnya.
 - Bangkitkan sebuah polinomial acak r di dalam $[-1, 0, 1]$.

- Untuk mengenkripsi m , gunakan polinomial yang dipilih secara acak r dan h untuk menghitung polinomial *ciphertext* menggunakan

$$e = r * h + m \text{ (mod } q\text{).} \quad (2.3)$$

- d. Penerima pesan acak terlebih dahulu menghitung f_p dan mengikuti langkah-langkah berikut untuk menghitung polinomial pesan asli.
 - Hitung polinomial a sebagai berikut : $a = f * e \text{ (mod } q\text{).}$ (2.4)
 - Hitung polinomial b sebagai berikut : $b = a \text{ (mod } p\text{)}$ (2.5)
 - Hitung polinomial m sebagai berikut : $m = f_p * b \text{ (mod } p\text{).}$ (2.6)
- e. Ubah polinomial m ke string untuk mendapatkan pesan asli.

2.3. Wavelet Transform

Dalam pengolahan citra terdapat suatu teknik domain transformasi yaitu *wavelet transform*, dimana sebuah *wavelet transform* mengubah citra ke dalam domain frekuensi. Beberapa *Wavelet transform* yaitu (Adi, P. W., dkk, 2015):

2.3.1. Discrete Wavelet Transform (DWT)

DWT mendekomposisi sebuah citra menjadi empat *subbands* LL, HL, LH dan HH. *Subband* LL merepresentasikan nilai rataan dari frekuensi rendah, sedangkan *subband* HL, LH dan HH masing-masing mereprsentasikan fitur horizontal, vertikal dan diagonal dari sebuah citra. Pada dasarnya, *wavelet transform* diselesaikan dengan menerapkan *wavelet filter* secara horizontal dan kemudian secara vertical (Adi, P. W., dkk, 2015).

2.3.2. Integer Wavelet Transform (IWT)

Sebuah *DWT* menghasilkan koefisien *floating point*. Walaupun masukannya berupa bilangan bulat sebagai representasi nilai piksel sebuah citra, koefisien *wavelet* tidak lagi terdiri dari bilangan bulat. Sehingga, pengubahan koefisien transformasi selama proses penyisipan akan menghasilkan citra rekonstruksi yang tidak dijamin memiliki nilai bulat. Sementara pembulatan ke bawah ataupun ke atas

pada *floating point* dapat menyebabkan hilangnya informasi rahasia (Adi, P. W., dkk, 2015).

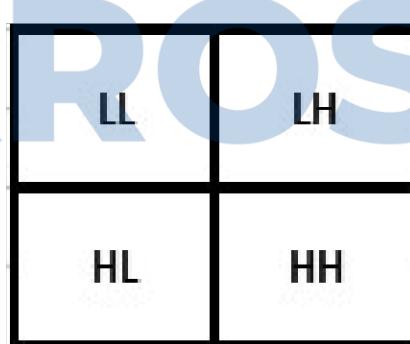
Disisi lain, *IWT* disebut juga generasi *wavelet* kedua lebih fleksibel dan mampu mendefenisikan basis *wavelet* pada sebuah *irregular grid* atau *interval*. Koefisien-koefisien dalam *IWT* direpresentasikan sebagai *finite precision numbers* yang memetakan bilangan bulat ke bilangan buat.

2.3.3. Integer Haar Wavelet Transform (IHWT)

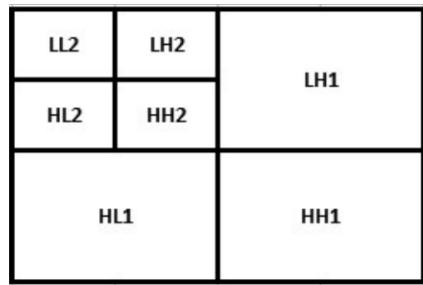
Metode IHWT dikembangkan dari DHWT melalui sebuah *lifting scheme* yang digunakan pada media gambar dengan cara mengubah bilangan bulat nilai pixel dari sebuah gambar ke dalam koefisien wavelet integer dan sebaliknya. Untuk mengubah gambar menjadi *wavelet subband*, sebuah citra dibagi menjadi 2X2 blok yang tidak saling tumpang tindih seperti matriks di bawah ini (Adi, P. W., dkk, 2015):

$$\begin{bmatrix} I_{m,n} & I_{m,n+1} \\ I_{m+1,n} & I_{m+1,n+1} \end{bmatrix} \quad (2.7)$$

Representasi blok 2X2 dari citra dimana diasumsikan $I_{m,n}$ adalah sebuah pixel pada baris m dan kolom n dari setiap blok citra. Lalu, IHWT akan diterapkan pada setiap blok gambar untuk mendapatkan *wavelet coefficient* pada *subbands* LL, HL, LH, dan HH. *Subbands wavelet* dapat dilihat pada gambar di bawah ini.



Gambar 2.3. Dekomposisi tingkat 1 (Adi, P. W., dkk, 2015)



Gambar 2. 4. Dekomposisi tingkat 2 (Adi, P, W., dkk, 2015)

$$LL = \left\lfloor \frac{\left| I_{m,n} + I_{m,n+1} \right| + \left| I_{m+1,n} + I_{m+1,n+1} \right|}{2} \right\rfloor \quad (2.8)$$

$$HL = \left\lfloor \frac{I - I_{m,n+1} + I_{m+1,n} - I_{m+1,n+1}}{2} \right\rfloor \quad (2.9)$$

$$LH = \left\lfloor \left| \frac{I_{m,n} + I_{m,n+1}}{2} \right| \right\rfloor - \left\lfloor \left| \frac{I_{m+1,n} + I_{m+1,n+1}}{2} \right| \right\rfloor \quad (2.10)$$

$$HH = I_{m,n} - I_{m,n+1} - I_{m+1,n} + I_{m+1,n+1} \quad (2.11)$$

Inverse Integer Haar Wavelet (IIHWT) dilakukan pada sekelompok *coefficient* untuk merekonstruksi kembali ke citra *cover* aslinya dari masing-masing *subband*.

$$\begin{bmatrix} I'_{m,n} & I'_{m,n+1} \\ I'_{m+1,n} & I'_{m+1,n+1} \end{bmatrix} \quad (2.12)$$

Maka, representasi blok 2X2 dari citra yang direkonstruksi menjadi

$$I'_{m,n} = LL + \left\lfloor \frac{LH+1}{2} \right\rfloor + \left\lfloor \frac{HL + \left\lfloor \frac{HH+1}{2} \right\rfloor + 1}{2} \right\rfloor \quad (2.13)$$

$$I'_{m,n+1} = I'_{m,n} - \left(HL + \left\lfloor \frac{HH+1}{2} \right\rfloor \right) \quad (2.14)$$

$$I'_{m+1,n} = LL + \left\lfloor \frac{LH+1}{2} \right\rfloor - LH + \left\lfloor \frac{HL + \left\lfloor \frac{HH+1}{2} \right\rfloor - HH+1}{2} \right\rfloor \quad (2.15)$$

$$I'_{m+1,n+1} = I'_{m+1,n} - \left(HL + \left\lfloor \frac{HH+1}{2} \right\rfloor \right) \quad (2.16)$$

2.4. Steganografi

2.4.1. Pengenalan Steganografi

Kata steganografi (*steganography*) berasal dari bahasa Yunani, yaitu ‘*stegano*’ yang berarti tersembunyi dan ‘*graphien*’ yang berarti menulis, sehingga steganografi dapat diartikan sebagai seni dan ilmu menyembunyikan pesan. Steganografi juga dapat dikatakan sebagai komunikasi yang tersembunyi karena pesannya yang terbuka selalu terlihat, tetapi tidak terdeteksi bahwa adanya pesan yang di rahasiakan. Sehingga sering disebut *hidden in plain sight* yang artinya bersembunyi di depan mata (Cox, I. J., dkk, 2008).

Kini steganografi lebih mengarah untuk menyembunyikan pesan rahasia (*hiding message*) sehingga keberadaan pesan tidak terdeteksi dari pihak yang tidak berkepentingan. Steganografi membutuhkan dua property : wadah penampung (*cover*) dan pesan rahasia yang akan disembunyikan. Pesan rahasia yang disembunyikan dapat berupa citra, suara, teks, dan video.

Steganografi yang dibahasi disini adalah menyembunyikan pesan rahasia di dalam citra warna saja. Meskipun demikian, penyembunyian pesan rahasia juga bisa dilakukan pada wadah berupa teks, suara dan video.

2.4.2. Kriteria Steganografi yang baik

Ada beberapa kriteria yang harus diperhatikan dalam penerapan steganografi (Cox, I. J., dkk, 2008):

1. *Imperceptibility*. keberadaan pesan rahasia dalam media penampung tidak dapat di deteksi oleh inderawi manusi. Misalnya, jika *cover text* berupa citra maka citra *stego* sulit ditemukan perbedaannya dengan *cover text* secara kasat mata.
2. *Fidelity*, mutu dari media penampung tidak berubah banyak akibat penyisipan. Pengujian terhadap *Fidelity* di lakukan dengan beberapa metode, salah satu metode yang paling banyak digunakan adalah dengan mengukur nilai PSNR.
3. *Robustness*, tingkat ketahanan citra hasil steganografi terhadap operasi dasar seperti rotasi dan resize merupakan aspek yang cukup penting. Pesan

yang terkandung di dalam citra *stego* seharusnya tidak rusak walaupun citra diputar, diperbesar atau diperkecil.

2.4.3. Metode Steganografi

Terdapat berberapa metode dasar pada steganografi dan proses penyisipannya:

1. Least Significant Bit (LSB)

LSB adalah metode steganografi sederhana yang menukar *bit* yang paling kecil ke dalam beberapa *byte* media penyembunyian. *LSB* juga merupakan bagian dari barisan data biner (basis dua) yang mempunyai nilai paling kecil dimana letaknya berada pada paling kanan dari barisan bit (Andrian, 2013).

Dalam proses penyisipan dengan *LSB* dibutuhkan representasi biner dari data yang akan disembunyikan. Misalkan kita memiliki tiga piksel yang berdekatan dengan kode RGB sebagai berikut :

00110101	11010110	11101010
11110100	00111001	11100001
01110001	10010001	11100001

Pesan yang akan disisipkan adalah karakter “R”, yang nilai binernya adalah “01010010”, maka akan dihasilkan citra hasil dengan urutan bit sebagai berikut:

00110100	11010111	11101010
11110101	00111000	11100000
01110001	10010000	11100001

Pada contoh di atas, dapat dilihat bahwa sebagian bit *LSB* yang ada pada citra asal digantikan dengan bit dari pesan yang akan disisipkan. Satu karakter = 1 byte = 8 bit, pesan akan memerlukan 8 lokasi tempat penyisipan bit pesan. Jika menggunakan citra *grayscale* berarti memerlukan 8 bit dari citra yang akan disisipkan pesan. Jika menggunakan citra RGB berarti memerlukan 8 bit dari 3 *plane* piksel dari citra yang akan disisipi pesan. Pada saat penyisipan pesan, ada piksel yang berubah dari piksel asal, ada juga piksel yang tidak berubah sama sekali. Hal ini

dikarenakan nilai dari bit sisip sama dengan bit *LSB* dari piksel citra yang akan disisipi pesan.

2. Pixel Value Differencing (PWD)

CD (*Coefficient Difference*) adalah metode yang diadopsi dari PVD, PVD merupakan metode steganografi yang beroperasi pada domain spasial yang menggunakan selisih nilai piksel yang satu dengan nilai piksel yang lain, dimana hasil selisih kedua piksel tersebut nantinya akan digunakan untuk menyisipkan pesan pada media lain yang ingin disembunyikan. Setelah pesan sudah disisipkan pada media penampung pesan, maka nilai piksel-piksel tersebut akan berubah, nilainya akan berubah sesuai dengan rumus perhitungan PVD (Wang, C. M., dkk, 2008).

Proses penyisipan pada metode ini dilakukan dengan cara membandingkan dua piksel yang bertetangga $P_{(i)}$ dan $P_{(i+1)}$ dengan menggunakan persamaan $d_i = |p_i - p_{i+1}|$. Hasil dari perbandingan tersebut digunakan untuk mengetahui berapa banyak *bit* yang dapat disisipkan ke dalam dua piksel yang dibandingkan. Metode ini menggunakan skema Wu dan Tsai yaitu $R = \{[0-7], [8-15], [16-13], [32-63], [64-127], [128-255]\}$. Skema ini digunakan untuk mengetahui terdapat di *range* mana selisih dari dua piksel tersebut, jika telah diketahui dimana letak *range* itu, maka kita dapat mengetahui batas bawah (L_i) dan batas atas (U_i). Setelah mengetahui batas bawah dan batas atas range-nya, kita dapat menghitung lebar dari *optimum range* (W_i) dengan persamaan $W_i = U_i - L_i + 1$. Langkah selanjutnya yaitu mencari jumlah *bit* sisip (t_i) yang dapat disisipkan melalui persamaan $t_i = \log_2 (W_i)$.

Penyisipan pesan dapat dilakukan dengan mengambil sebanyak t_i *bit* dari pesan yang akan disisipkan. Setelah mengetahui berapa banyak bit sisip lalu ubahlah bit-bit sisip ke dalam desimal (**b**). Selanjutnya dihitung nilai *difference value* yang baru untuk penyisipan ke dalam citra menggunakan persamaan $d'_i = b + L_i$. Untuk menentukan nilai piksel yang telah disisipkan pesan menggunakan beberapa aturan yang harus dipenuhi, dimana m didapat dari selisih d'_i dengan d_i

menggunakan persamaan $m = d'_i - d_i$. Proses-proses tersebut dilakukan terus hingga bit sisip tersisipi semuanya ke dalam citra stego.

Proses ekstraksi pesan dari citra stego menggunakan metode *PVD* dimulai dengan mengurutkan semua piksel pada gambar yang telah disisipkan pesan, sesuai cara pengambilan pesannya. Kemudian dihitung selisih nilai *difference value* baru. Nilai *difference value* baru tersebut digunakan untuk mengetahui nilai *continuous ranges* (R) yang sudah didefinisikan. Dengan demikian didapatkan pesan yang telah disisipkan.

Berdasarkan informasi tersebut dapat diketahui seberapa panjang data rahasia yang disisipkan pada kedua piksel, sehingga pesan rahasia yang telah disisipkan didapatkan kembali. Proses algoritma *PVD* untuk ekstraksi pesan dimulai dengan menghitung selisih nilai d_i dengan persamaan $d_i = |P'_{(i)} - P'_{(i+1)}|$, dimana $P'_{(i)}$ dan $P'_{(i+1)}$ adalah piksel yang telah disisipkan pesan. Kemudian cari nilai $U_{(i)}$ dan $L_{(i)}$ yang merupakan nilai batas atas dan nilai batas bawah pada range table dengan menggunakan nilai d_i yang telah diperoleh. Setelah mengetahui batas bawah dan batas atas *range*-nya, kita dapat menghitung lebar dari *optimum range* (W_i) dengan persamaan $W_i = U_i - L_i + 1$. Lalu kita dapat mengetahui pesan yang telah disisipkan dengan persamaan $b' = d_i - L_i$. Jika stego-image tidak berubah maka $b' = b$. Kemudian ubah ke biner. Setelah itu gunakan perhitungan t_i dengan persamaan $t_i = \lfloor \log_2 (W_i) \rfloor$ untuk mengetahui seberapa panjang bit sisip yang diekstraksi. Lakukan sampai semua piksel telah berhasil di proses.

Perbedaan antara *PVD* dengan *CD* adalah nilai pada prosesnya, pada *PVD* nilai yang diproses yaitu nilai piksel dari citra sedangkan *CD* nilai yang diproses yaitu nilai koefisien dari citra.

3. Modulus Function

Metode ini memodifikasi sisa dari 2 pixel bersebelahan $P_{(i,x)}$ dan $P_{(i,y)}$ untuk mendapatkan kualitas citra stego yang lebih baik, tahapan proses penyisipan dan ekstraksi yang dilakukan metode *modulus function* adalah sebagai berikut (Wang, C. M., dkk, 2008) :

a. Proses Penyisipan

Step 1 : Diberikan sebuah sub-block citra F , F_i yang tersusun dari 2 pixel berurutan $P_{(i,x)}$ dan $P_{(i,y)}$ dari citra sampul, dapatkan nilai selisih d_i , sub-range R_j sehingga $R_j \in [ij,uj]$, width $w_j = u_j - i_j + 1$, Kapasitas penyembunyian t_i bits, dan nilai desimal t'_i dari t_i untuk setiap F_i dengan $t_i = \lfloor \lg(w_j) \rfloor$.

Step 2 : Hitung nilai sisa dari $P_{rem(i,x)}$, $P_{rem(i,y)}$, dan $F_{rem(i)}$ dari $P_{(i,x)}$, $P_{(i,y)}$ dan sub-block F_i masing-masing dengan menggunakan Persamaan berikut :

$$\begin{aligned} P_{rem(i,x)} &= P_{(i,x)} \bmod t'_i, \\ P_{rem(i,y)} &= P_{(i,y)} \bmod t'_i, \\ F_{rem(i)} &= (P_{(i,x)} + P_{(i,y)}) \bmod t'_i. \end{aligned} \quad (2.17)$$

Step 3: Sisip bit-bit t_i dari data rahasia ke dalam F_i dengan mengubah $P_{(i,x)}$ dan $P_{(i,y)}$ dimana $F_{rem(i)} = t'_i$. Pendekatan optimal untuk mengubah $P_{(i,x)}$ dan $P_{(i,y)}$ untuk mencapai distorsi (aturan) minimum adalah sebagai berikut :

Case 1 : $F_{rem(i)} > t'_i$ dan $m \leq (2^{t_i})/2$ dan $P_{(i,x)} \geq P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} - \lfloor m/2 \rfloor, P_{(i,y)} - \lfloor m/2 \rfloor); \quad (2.18)$$

Case 2 : $F_{rem(i)} > t'_i$ dan $m \leq (2^{t_i})/2$ dan $P_{(i,x)} < P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} - \lfloor m/2 \rfloor, P_{(i,y)} - \lfloor m/2 \rfloor); \quad (2.19)$$

Case 3 : $F_{rem(i)} > t'_i$ dan $m > (2^{t_i})/2$ dan $P_{(i,x)} \geq P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} + \lfloor m_1/2 \rfloor, P_{(i,y)} + \lfloor m_1/2 \rfloor); \quad (2.20)$$

Case 4 : $F_{rem(i)} > t'_i$ dan $m > (2^{t_i})/2$ dan $P_{(i,x)} < P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} + \lfloor m_1/2 \rfloor, P_{(i,y)} + \lfloor m_1/2 \rfloor); \quad (2.21)$$

Case 5 : $F_{rem(i)} \leq t'_i$ dan $m \leq \frac{2^{t_i}}{2}$ dan $P_{(i,x)} \geq P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} + \lfloor m/2 \rfloor, P_{(i,y)} + \lfloor m/2 \rfloor); \quad (2.22)$$

Case 6 : $F_{rem(i)} \leq t'_i$ dan $m \leq (2^{t_i})/2$ dan $P_{(i,x)} < P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} + \lfloor m/2 \rfloor, P_{(i,y)} + \lfloor m/2 \rfloor); \quad (2.23)$$

Case 7 : $F_{rem(i)} \leq t'_i$ dan $m > (2^{t_i})/2$ dan $P_{(i,x)} \geq P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} - \lfloor m_1/2 \rfloor, P_{(i,y)} - \lfloor m_1/2 \rfloor); \quad (2.24)$$

Case 8 : $F_{rem(i)} \leq t'_i$ dan $m > (2^{t_i})/2$ dan $P_{(i,x)} < P_{(i,y)}$

$$(P'_{(i,x)}, P'_{(i,y)}) = (P_{(i,x)} - \lfloor m_1/2 \rfloor, P_{(i,y)} - \lfloor m_1/2 \rfloor). \quad (2.25)$$

Dalam pendekatan diatas, $m = |F_{rem(i)} - t'_i|$, $m_1 = 2^{t_i} - |F_{rem(i)} - t'_i|$ dan $P'_{(i,x)}, P'_{(i,y)}$ adalah nilai baru pixel setelah penyisipan bit-bit t_i dari data rahasia ke dalam sub-block F_i . Setelah Step 3, jika $P'_{(i,x)}$ atau $P'_{(i,y)}$ melebihi nilai batas 0 atau 255, maka jalankan Step 4 untuk memperbaiki $P'_{(i,x)}$ dan $P'_{(i,y)}$. Jika tidak, tujuan untuk menyembunyikan data rahasia akan diselesaikan setelah pergantian dari $(P_{(i,x)}, P_{(i,y)})$ oleh $(P'_{(i,x)}, P'_{(i,y)})$ di dalam citra sampul.

Step 4 : Mempertimbangkan 3 situasi di berikut dimana masalah *falling-off-boundary* terjadi dan perbaiki $P'_{(i,x)}$ dan $P'_{(i,y)}$ sebagai berikut :

Case 1 : Jika $P_{(i,x)} \approx 0$, $P_{(i,y)} \approx 0$ dan $P'_{(i,x)} < 0$ atau $P'_{(i,y)} < 0$, kemudian atur ulang $P'_{(i,x)}$ dan $P'_{(i,y)}$ untuk menjadi $P''_{(i,x)}$ dan $P''_{(i,y)}$ dengan:

$$(P''_{(i,x)}, P''_{(i,y)}) = (P'_{(i,x)} + (2^{t_i})/2, P'_{(i,y)} + (2^{t_i})/2). \quad (2.26)$$

Case 2 : Jika $P_{(i,x)} \approx 255$, $P_{(i,y)} \approx 255$ dan $P'_{(i,x)} > 255$ atau $P'_{(i,y)} > 255$, kemudian atur ulang $P'_{(i,x)}$ dan $P'_{(i,y)}$ untuk menjadi $P''_{(i,x)}$ dan $P''_{(i,y)}$ dengan:

$$(P''_{(i,x)}, P''_{(i,y)}) = (P'_{(i,x)} + (2^{t_i})/2, P'_{(i,y)} + (2^{t_i})/2). \quad (2.27)$$

Case 3 : Jika $P_{(i,x)}$ dan $P_{(i,y)}$ membentuk kontras kuat (i.e. $d_i > 128$), kemudian, atur ulang $P'_{(i,x)}$ dan $P'_{(i,y)}$ dengan:

$$P''_{(i,y)} = \begin{cases} (0, P'_{(i,y)} + P'_{(i,x)}), \\ \text{if } P'_{(i,x)} < 0 \text{ dan } P'_{(i,y)} \geq 0; \\ (P'_{(i,x)} + P'_{(i,y)}, 0), \\ \text{if } P'_{(i,x)} \geq 0 \text{ dan } P'_{(i,y)} \geq 0; \\ (255, P'_{(i,x)} + (P'_{(i,x)} - 255)), \\ \text{if } P'_{(i,x)} > 255 \text{ dan } P'_{(i,y)} \geq 0; \\ (P'_{(i,x)} + (P'_{(i,y)} - 255), 255), \\ \text{if } P'_{(i,x)} \geq 0 \text{ dan } P'_{(i,y)} > 255. \end{cases} \quad (2.28)$$

Untuk ilustrasi memodifikasi sisa dari dua piksel berurutan dengan *modulus function* dapat dilihat pada tabel di bawah ini:

Tabel 2.2. Ilustrasi *modulus function* (Wang, C. M., dkk, 2008)

Secret Data (decimal value)	$P_{(i,x)} = 32$, $P_{rem(i,x)} = 0$	$P_{(i,y)} = 32$, $P_{rem(i,y)} = 0$	The total remainder $F_{rem(i)}$
0	No modify	No modify	0
1	+1	No modify	1
2	+1	+1	2
3	+2	+1	3
4	+2	+2	4
5	-2	-1	5
6	-1	-1	6
7	-1	No modify	7

Setelah step 4, $(P'_{(i,x)}, P'_{(i,y)})$ dapat dikoreksi sehingga *range* dari $(P''_{(i,x)}, P''_{(i,y)})$ tidak dapat menuju lebih kecil dari 0 atau lebih besar dari 255. Terakhir, $(P''_{(i,x)}, P''_{(i,y)})$ digunakan menggantikan $(P_{(i,x)}, P_{(i,y)})$ dalam citra sampul.

Sebuah contoh sederhana bagaimana meregulasi nilai sisa untuk menyembunyikan data ditunjukkan dalam Tabel 2.1. Asumsi kita mempunyai *sub-block* F_i dengan dua nilai piksel beruntun $P_{(i,y)} = 32$. Kemudian, nilai sisa $F_{rem(i)}$

dari F_i adalah 0. Jika ketiga bit (sebagai contoh, $t_i = 3$, dan $t'_i=2^3 = 8$) dari data rahasia dipilih untuk disisipkan ke dalam F_i , $P_{(i,x)}$ dan $P_{(i,y)}$ akan dimodifikasi untuk menampung 3-bit data rahasia tersebut. Tabel 2.1 menunjukkan bahwa skema ini mempunyai kinerja lebih baik dalam mengurangi selisih antara $(P_{(i,x)}, P_{(i,y)})$ dan $(P'_{(i,x)}, P'_{(i,y)})$.

b. Proses Ekstraksi

Sebuah ilustrasi untuk memecahkan permasalahan penyimpangan dari batas dengan memodifikasi ulang $(P'_{(i,x)}, P'_{(i,y)})$. Tabel ilustrasinya dapat dilihat di bawah ini:

Tabel 2.3. Ilustrasi memecahkan permasalahan penyimpangan dari batas dengan memodifikasi ulang $P_{(i,x)}'$, $P_{(i,y)}'$ (Wang, C. M., dkk, 2008)

Secret Data (decimal value)	$P_{(i,x)} = 0$, $P_{rem(i,x)} = 0$	$P_{(i,y)} = 0$, $P_{rem(i,y)} = 0$	The total remainder $F_{rem(i)}$
0	No modify	No modify	0
1	+1	No modify	1
2	+1	+1	2
3	+2	+1	3
4	+2	+2	4
5	$(-2) + 4 = 2$	$(-1) + 4 = 3$	5
6	$(-1) + 4 = 3$	$(-1) + 4 = 3$	6
7	$(-1) + 4 = 3$	$(0) + 4 = 4$	7

Selanjutnya, diberikan contoh untuk menunjukkan bagaimana mekanisme dari menjaga nilai pixel melewati kisaran [0,255] setelah penyisipan data rahasia. Seperti yang ditunjukkan pada Tabel 2.2, asumsikan kembali $P_{(i,x)} = 0$ dan $P_{(i,y)} = 0$ dalam contoh sebelumnya, dan permasalahan penyimpangan dari batas terjadi sebagaimana sehingga , $P'_{(i,x)} < 0$ atau , $P'_{(i,y)} < 0$ saat nilai desimal dari data rahasia adalah 5, 6, atau 7. Akan tetapi, $P'_{(i,x)}$ dan $P'_{(i,y)}$ dapat diatur ulang dengan

menambahkan sampai 4 secara bersamaan. Setelah itu, nilai-nilai dari $P''_{(i,x)}$ dan $P''_{(i,y)}$ akan masuk dalam kisaran 0-255.

Dalam proses pengambilan kembali, kita dapat dengan cepat mengekstrak data rahasia tanpa menggunakan citra asal. Bagaimanapun juga, adalah esensial untuk menggunakan tabel batas R yang didesain dalam tahap penyisipan dengan tujuan menemukan kapasitas penyisipan untuk setiap sub-block F_i . Diberikan sebuah sub-block F_i dengan dua pixel berurutan dari citra stego dengan nilai piksel mereka berupa $P_{(i,x)}$ dan $P_{(i,y)}$, nilai selisih d_i dari $P_{(i,x)}, P_{(i,y)}$ dapat diturunkan dari Persamaan (2.23):

$$d_i = |P_{(i,x)} - P_{(i,y)}| \quad (2.29)$$

Setiap F_i dapat berkaitan dengan sub-range optimal R_j dari tabel original R menurut nilai selisih d_i . Maka, dapat dihitung lebar dari sub-range melalui $w_j = u_j - l_j$, dan jumlah bit-bit t_i dari data rahasia dapat di *extraction* dari F_i melalui Persamaan (2.23). Pada akhirnya, kita menghitung nilai sisa dari F_i dengan menggunakan persamaan (2.24):

$$t_i = \lfloor \lg(w_j) \rfloor \quad (2.30)$$

Dimana t_i adalah jumlah bit yang dapat disembunyikan dalam F_i . Kemudian, mentransformasikan nilai sisa $F_{rem(i)}$ ke dalam string biner dengan panjang t_i . Setelah itu, algoritma *etraction* telah selesai dilakukan. Sebagai contoh, asumsikan dua nilai pixel beruntun dari citra stego adalah $P_{(i,x)} = 34$ dan $P_{(i,y)} = 33$, dan kapasitas tersembunyi adalah 3 bit (sebagai contoh $t_i = 3$). $F_{rem(i)}$ dapat didapatkan dari $(33 + 34) \bmod 2^3 = 3$. Konversikan nilai sisa 3 ke dalam string biner yang panjangnya adalah 3, lalu kita mendapatkan $3_{(10)} = 011_{(2)}$. Hal itu terjadi jika, data rahasia $011_{(2)}$ didapatkan kembali dari sub-block dimana nilai piksel adalah $P_{(i,x)} = 34$ dan $P_{(i,y)} = 33$.

2.4.4. Proses Steganografi

1. Proses Penyisipan

Nilai sisa dari dua koefisien berurutan digunakan untuk menyisipkan pesan menggunakan *modulus function* dengan nilai *threshold* yang terlebih dahulu ditentuan. Nilai sisa *modulus function* menghasilkan sebuah nilai integer yang cocok untuk *IHWT* (Adi, P. W., dkk, 2015). Sebelum memulai sebuah proses penyisipan nilai *threshold* harus ditentukan terlebih dahulu, nilai *threshold* akan menentukan kapasitas penyembunyian juga untuk mencegah citra yang direkonstruksi keluar dari kisaran nilai pixel 0 sampai 255.

Selain itu, sebuah kunci bertipe data integer 8-bit (`uint8`) digunakan untuk mengenkripsi pesan dan mengacak lokasi penyisipan sebenarnya. Proses penyisipan dan proses ekstraksi tersebut adalah sebagai berikut :

- a. Definisikan nilai ambang (T):
 - i. Untuk menentukan kapasitas penyembunyian, dimana jumlah bit yang akan disisipkan adalah sama dengan nilai *threshold*.
 - ii. Untuk menghindari nilai pixel melebihi kisaran nilai 0 dan 255 :

$$I(m, n) = \begin{cases} \alpha T, & \text{jika } I(m, n) < \alpha T \\ 255 - \alpha T, & \text{jika } I(m, n) > 255 - \alpha T \end{cases} \quad (2.31)$$

dimana $I(m, n)$ adalah piksel citra pada baris m dan kolom n , α adalah faktor bobot integer, dan T adalah nilai Threshold.

- b. Melakukan transformasi *IHWT* pada citra sampul menjadi subband-subband.
- c. Mentransformasikan kunci dekripsi dan pesan acak yang akan disisip menjadi bit-bit.
- d. Gabungkan bit kunci dekripsi dan bit sisip menjadi 1 blok.
- e. Tentukan kunci stego dengan menghitung panjang dari pesan acak.
- f. Hitung sisa (r_k) dari 2 pixel bersebelahan:

$$r_k = (g_{ij} + g_{ij+1}) \bmod 2^T \quad (2.32)$$

dimana r_k adalah sisa dari dua pixel bersebelahan g_{ij} dan g_{ij+1} dalam grup k . Jika $r_k = \text{negatif}$, maka hitung kembali $r_k = r_k + 2^T$ sampai r_k menghasilkan nilai positif.

g. Sisip bit-bit sisip T ke dalam citra dengan menyesuaikan :

$$\text{i. } r_k > t_k \text{ dan } m \leq (2^T) \text{ dan } g_{ij} \geq g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = \left(g_{ij} - \left\lceil \frac{m}{2} \right\rceil, g_{ij+1} - \left\lceil \frac{m}{2} \right\rceil \right) \quad (2.33)$$

$$\text{ii. } r_k > t_k \text{ dan } m \leq 2^T \text{ dan } g_{ij} < g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} - \lfloor m/2 \rfloor, g_{ij+1} - \lfloor m/2 \rfloor) \quad (2.34)$$

$$\text{iii. } r_k > t_k \text{ dan } m > 2^T \text{ dan } g_{ij} \geq g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} + \lfloor m'/2 \rfloor, g_{ij+1} + \lfloor m'/2 \rfloor); \quad (2.35)$$

$$\text{iv. } r_k > t_k \text{ dan } m > 2^T \text{ dan } g_{ij} < g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} + \lceil m'/2 \rceil, g_{ij+1} + \lceil m'/2 \rceil); \quad (2.36)$$

$$\text{v. } r_k \leq t_k \text{ dan } m \leq 2^T \text{ dan } g_{ij} \geq g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} + \lfloor m/2 \rfloor, g_{ij+1} + \lfloor m/2 \rfloor) \quad (2.37)$$

$$\text{vi. } r_k \leq t_k \text{ dan } m \leq 2^T \text{ dan } g_{ij} < g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} + \lceil m/2 \rceil, g_{ij+1} + \lceil m/2 \rceil) \quad (2.38)$$

$$\text{vii. } r_k \leq t_k \text{ dan } m > 2^T \text{ dan } g_{ij} \geq g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} - \lceil m'/2 \rceil, g_{ij+1} - \lceil m'/2 \rceil); \quad (2.39)$$

$$\text{viii. } r_k \leq t_k \text{ dan } m > 2^T \text{ dan } g_{ij} < g_{ij+1},$$

$$(g'_{ij}, g'_{ij+1}) = (g_{ij} - \lfloor m'/2 \rfloor, g_{ij+1} - \lfloor m'/2 \rfloor) \quad (2.40)$$

dimana t_k adalah nilai desimal dari bit-bit T , $m = |r_k - t_k|$, $m' = 2^T - m$ dan (g'_{ij}, g'_{ij+1}) adalah nilai koefisien baru setelah penyisipan.

- h. Setelah kapasitas penyisipan telah tercapai, atau seluruh pesan telah disisipkan, lakukan transformasi invers IHWT pada semua koefisien subband untuk merekonstruksi citra stego.

Pada proses ekstraksi kunci yang telah didefinisikan terlebih dahulu dan nilai ambang T digunakan untuk mengekstrak data teks dari citra stego.

2. Proses Ekstraksi

Proses ekstraksi adalah sebagai berikut :

- Lakukan transformasi IHWT tingkat 1 untuk mendapatkan subband-subband wavelet dari citra stego.
- Lakukan ekstraksi menggunakan kunci stego pada seluruh bit-bit tersisip.
- Hitung sisa dari 2 koefisien bersebelahan untuk mendapatkan nilai desimal dari bit-bit tersisip.
- Dapatkan bit stream dari seluruh bit-bit tersisip.
- Pisahkan bit-bit yang tersisip menjadi kunci dekripsi dan pesan acak.
- Pengujian terhadap ketahanan citra stego adalah *noise salt and pepper*.

$$r_k' = (g_{ij} + g_{ij+1}) \bmod 2^T \quad (2.41)$$

2.5. Noise

Noise adalah titik-titik pada citra yang sebenarnya bukan merupakan bagian dari citra, melainkan ikut tercampur pada citra karena suatu sebab. Ada dua macam *noise* dan penjelasannya (Gonzalez, C., dkk, 2008), yaitu:

2.5.1. Gaussian Noise

Gaussian noise merupakan titik-titik berwarna pada citra yang jumlahnya sama dengan persentase *noise*. *Noise* ini dibangkitkan dengan cara membangkitkan biangan acak $[0,1]$. Titik-titik yang terkena *noise*, nilai fungsi citra ditambahkan dengan nilai *noise* yang diperoleh atau dapat dirumuskan sebagai berikut (Gonzalez, C., dkk, 2008):

$$F(i,j) = g(i,j) + p * a \quad (2.40)$$

Dengan penjelasan a sebagai bilangan acak distribusi *gaussian*, p sebagai persentase *noise*, $f(i,j)$ sebagai nilai intensitas citra terkena *noise* dan $g(i,j)$ sebagai nilai intensitas citra sebelum terkena *noise*.

2.5.2. Noise Salt and Pepper

Fungsi probabilitas kepadatan PDF (*Probability Density Function*) diperoleh dengan (Gonzalez, C., dkk, 2008) :

$$p(z) = \begin{cases} Pa \text{ untuk } z = a \\ Pb \text{ untuk } z = b \\ 0 \text{ lainnya} \end{cases} \quad (2.41)$$

Dimana $p(z)$ adalah fungsi probabilitas kepadatan *noise*, Pa adalah probabilitas *noise* jenis a (*pepper*) dan Pb adalah probabilitas *noise* b (*salt*). Jika $b > a$, intensitas b akan tampak sebagai titik terang pada citra. *Noise salt and pepper* dapat dibangkitkan dengan cara membangkitkan bilangan 255 (warna putih) pada titik yang secara probabilitas lebih kecil dari nilai probabilitas *noise*, dan dirumuskan dengan:

$$f(x,y) = 255 \text{ jika } p(x,y) < probNoise \quad (2.42)$$

$$f(x,y) = \text{tetap jika } p(x,y) > probNoise \quad (2.43)$$

Dimana $f(x,y)$ adalah hasil piksel citra pada titik (x,y) dan $p(x,y)$ adalah probabilitas acak.

2.6. Peak Signal to Noise Ratio (PSNR)

PSNR adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya *noise* yang berpengaruh pada sinyal tersebut. *PSNR* diukur dalam satuan desibel. *PSNR* digunakan untuk mengetahui perbandingan kualitas citra *stego* sebelum dan sesudah disisipkan pesan. Untuk menentukan *PSNR*, terlebih dahulu harus ditentukan *Mean Square Error (MSE)*. *MSE* adalah nilai error kuadrat rata-rata antara citra cover dengan citra tersteganografi, secara matematis dapat dirumuskan sebagai berikut (Wang, C. M., dkk, 2008):

$$MSE = \frac{1}{MN} \sum_{y=1}^M \sum_{x=1}^N [I(x,y) - I'(x,y)]^2 \quad (2.44)$$

Dimana :

MSE = Nilai *mean square error* citra steganografi

M = Panjang citra stego

N = Lebar citra stego

$I(x,y)$ = Nilai pixel dari citra *cover*

$I'(x,y)$ = Nilai pixel pada citra stego

Setelah diperoleh nilai MSE maka nilai $PSNR$ dapat dihitung dari kuadrat nilai maksimum dibagi dengan MSE . Semakin rendah nilai MSE maka akan semakin baik, dan semakin besar nilai $PSNR$ maka semakin baik kualitas citra steganografi. Secara matematis, nilai $PSNR$ dirumuskan sebagai berikut :

$$PSNR = 10 \log \left(\frac{MAX^2}{MSE} \right) \quad (2.45)$$

Dimana, MSE = nilai MSE dan MAX_i = nilai maksimum dari pixel citra yang digunakan.

UNIVERSITAS MIKROSKIL