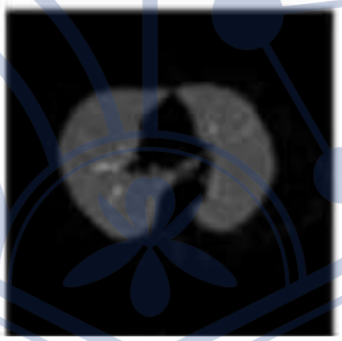
BAB II
KAJIAN LITERATUR

2.1 Citra

Citra adalah representasi visual dari objek di dunia nyata baik yang dapat dipahami oleh komputer, biasanya terdiri dari kombinasi titik, garis, bidang dan warna yang digunakan untuk menciptakan representasi visual tersebut [11], [12]. Citra juga dapat didefinisikan sebagai fungsi $f(x, y)$ berukuran M baris dan N kolom, dengan b dan c merupakan koordinat spasial, dan amplitudo a di titik koordinat (x, y) dinamakan intensitas atau tingkat keabuan oleh citra tersebut. Citra dapat terbagi menjadi 2 berdasarkan sifatnya yaitu citra yang bersifat analog dan citra yang bersifat digital [11].

2.1.1 Citra Analog

Citra analog adalah citra yang bersifat kontinu seperti gambar pada televisi, foto sinar X, lukisan dan sebagainya. Citra analog tidak dapat direpresentasikan dalam komputer, sehingga diperlukan adanya proses digitalisasi terlebih dahulu agar citra analog dapat diproses oleh komputer. Citra analog dihasilkan dari alat-alat analog seperti kamera analog, CT *Scan*, sensor gelombang pendek pada sistem radar dan lainnya [11]. Contoh citra analog seperti foto CT *Scan* dapat dilihat pada gambar 2.1.

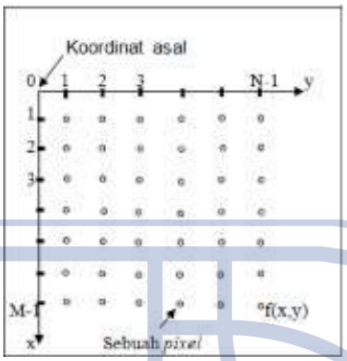


Gambar 2.1 Foto CT *Scan* [13]

2.1.2 Citra Digital

Citra digital adalah representasi numerik dari gambar dua dimensi yang dapat diproses oleh komputer dengan melakukan pendekatan berdasarkan *sampling* dan kuantisasi. *Sampling* menyatakan besar kecilnya ukuran titik (*pixel*) pada citra, dan kuantisasi menyatakan jumlah warna yang ada pada citra tersebut. Setiap *pixel* memiliki nilai numerik yang menggambarkan tingkat kecerahan atau warna pada posisi tertentu dalam citra. Citra

digital dapat diperoleh dari berbagai sumber seperti kamera digital, atau hasil simulasi komputer [11], [12], [14]. Koordinat citra digital terhadap sumbu (x, y) suatu bidang dua dimensi dapat dilihat seperti gambar 2.2.



Gambar 2.2 Koordinat Citra Digital [15]

Citra digital secara matematis dapat dituliskan ke dalam bentuk matriks seperti yang terlihat pada persamaan (1) berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad \dots\dots(1)$$

Besar intensitas yang diterima sensor di setiap titik (x, y) disimbolkan oleh $f(x, y)$ dan besarnya tergantung pada intensitas yang dipantulkan oleh objek. Ini berarti $f(x, y)$ sebanding dengan energi yang dipancarkan oleh sumber cahaya, sehingga besar intensitas $f(x, y)$ dapat dinyatakan seperti yang terlihat pada persamaan (2) berikut:

$$0 < (x, y) < \infty \quad \dots\dots(2)$$

2.1.3 Jenis Citra

Berikut merupakan beberapa jenis citra digital yang sering digunakan:

1. Citra Biner

   Citra biner merupakan citra yang hanya memiliki dua nilai *pixel* yang mungkin yaitu hitam dan putih. Hal ini merupakan hasil dari proses segmentasi di mana *pixel* dikelompokkan menjadi dua kelas berdasarkan ambang tertentu. Citra ini hanya memerlukan 1 bit untuk menyimpan kedua warna, sering kali 0 untuk hitam dan 1 untuk putih [1-3]. Bentuk citra biner dari citra *grayscale* dapat dilihat pada persamaan (3) berikut:

$$g(x,y) = \begin{Bmatrix} 1 \; if \; f(x,y) \geq T \\ 0 \; if \; f(x,y) < T \end{Bmatrix} \quad \text{.....(3)}$$

Di mana $g(x,y)$ adalah bentuk citra biner dari citra *grayscale* $f(x,y)$ dan T adalah *thresholding* (penentuan ambang) citra. *T-score* ini dapat ditentukan dengan dua cara.

1) Nilai Ambang Global

   $T = T\{f(x,y)\}$

   Dengan T tergantung pada nilai *gray level* dari *pixel* pada posisi x, y.

2) Nilai Ambang Lokal

   $T = T\{A(x,y), f(x,y)\}$ [14]

   Citra biner dapat dilihat seperti pada gambar 2.3.

Gambar 2.3 Citra Biner [14]

2. Citra Grayscale

   Citra *grayscale* adalah citra yang mengandung warna keabuan dan memiliki tingkat kecerahan sebagai informasi. Setiap *pixel* dalam citra *grayscale* direpresentasikan oleh satu angka, biasanya dalam rentang 0 hingga 255 dengan nilai 0 untuk warna hitam dan nilai 255 untuk warna putih [12], [14]. Contoh citra *grayscale* dapat dilihat seperti pada gambar 2.4.

Gambar 2.4 Citra *Grayscale* [16]

3. Citra Warna

   Citra warna merupakan citra yang mengandung informasi kecerahan dan warna. Representasi yang paling umum untuk citra warna adalah mode RGB (Merah, Hijau, Biru), setiap *pixel* dalam citra RGB direpresentasikan oleh tiga nilai yang masing-masing mewakili tingkat merah, hijau dan biru dalam rentang 0 hingga 255. Setiap warna dasar menggunakan penyimpanan 8 bit (1 *byte*) yang berarti setiap warna mempunyai gradasi sebanyak 255 warna. Dengan demikian, maka setiap *pixel* mempunyai kombinasi lebih dari 16 juta warna sehingga format RGB ini juga dinamakan dengan *true color* karena hampir mencakup semua warna di alam [11], [12]. Contoh citra warna dapat dilihat seperti pada gambar 2.5.

Gambar 2.5 Citra Warna [17]

2.1.4 Elemen Citra Digital

Citra digital memiliki elemen-elemen dasar. Elemen dasar yang terdapat pada citra digital dapat dijelaskan lebih lanjut antara lain:

1. Kecerahan (*Brightness*)

   Brightness merupakan intensitas cahaya yang dipancarkan *pixel* dari citra yang dapat ditangkap oleh sistem penglihatan. Kecerahan pada sebuah titik (*pixel*) di dalam citra merupakan intensitas rata-rata dari suatu area yang melingkupinya.

2. Kontras (*Contrast*)

   Contrast menyatakan sebaran terang dan gelap dalam sebuah citra, pada citra yang baik komposisi gelap dan terang akan tersebar secara merata pada citra. Kontras yang baik membantu membedakan objek utama dari latar belakang dengan lebih jelas.

3. Kontur (*Contour*)

   Contour adalah keadaan yang ditimbulkan oleh perubahan intensitas pada *pixel-pixel* yang bertetangga. Karena adanya perubahan intensitas inilah mata mampu mendeteksi tepi-tepi objek di dalam citra.

4. Warna (*Colour*)

   Warna merupakan persepsi yang ditangkap sistem visual terhadap panjang gelombang cahaya yang dipantulkan oleh objek.

5. Bentuk (*Shape*)

   Shape adalah properti intrinsik dari objek tiga dimensi, dengan pengertian bahwa bentuk merupakan properti intrinsik utama untuk sistem visual manusia.

6. Tekstur (*Texture*)

   Texture adalah sifat-sifat atau karakteristik yang dimiliki oleh suatu daerah yang cukup besar, sehingga sifat-sifat tersebut dapat berulang dalam daerah tersebut. *Texture* juga dapat diartikan sebagai keteraturan pola-pola tertentu yang terbentuk dari susunan *pixel-pixel* dalam citra digital. Informasi *texture* dapat digunakan untuk membedakan sifat-sifat permukaan suatu benda dalam citra yang berhubungan dengan kasar dan halus terlepas dari warna permukaan tersebut [11].

2.1.5 Format Citra

Citra digital mengandung data yang bersifat *redundant*, yaitu data yang bernilai sama tetapi muncul berulang kali. Untuk mengatasi hal tersebut, teknik kompresi sering digunakan terhadap citra digital untuk memperkecil ukurannya. Beberapa format citra digital antara lain sebagai berikut [11], [12], [14], [18], [19]:

1. BMP

   Bitmap (.bmp) merupakan format citra yang dikembangkan oleh Microsoft untuk sistem operasi Windows. Format BMP tidak terkompresi, sehingga menghasilkan kualitas citra yang sangat tinggi tetapi ukuran *file* juga menjadi sangat besar.

2. JPG/JPEG

   Joint Photographic Experts Group (.jpg, .jpeg) adalah format *file* citra yang bersifat *lossy* yang berarti *file* dikompresi untuk mendapatkan ukuran yang lebih kecil. Kompresi yang dilakukan mengakibatkan berkurangnya kualitas namun biasanya hal ini tidak terlalu terlihat, format jenis ini sangat umum ditemukan pada internet dan merupakan format yang sangat populer untuk kamera digital.

3. TIFF

   Tagged Image File Format (.tif, .tiff) merupakan format *file* gambar yang bersifat *loseless* yang berarti tidak perlu melakukan kompresi atau kehilangan informasi gambar atau menurunnya kualitas gambar.

2.2 Steganografi

Steganografi merupakan cara untuk menyembunyikan informasi atau data dalam bentuk apa pun (teks, gambar, suara, video) ke dalam media yang lebih besar (gambar, suara, video) sedemikian rupa sehingga tidak seorang pun mencurigai adanya keberadaan informasi atau data selain pengirim dan penerima [20].

2.2.1 Metode Steganografi

Metode steganografi pada citra digital terdiri dari metode sebagai berikut:

1. *Least Significant Bit* (LSB)

   LSB merupakan metode sederhana sekaligus pendekatan steganografi yang paling sering digunakan karena komputasinya tidak terlalu kompleks dan pesan yang disembunyikan cukup aman. Pendekatan LSB dilakukan dengan cara mengubah nilai bit terkecil yang terletak pada barisan paling kanan atau bit ke-8 dari suatu *byte* data [21], [22].

2. *Most Significant Bit* (MSB)

   MSB merupakan sebuah pendekatan steganografi yang sedikit dimodifikasi dari pendekatan LSB. Teknik ini dilakukan dengan mengubah bit yang memiliki nilai paling besar atau bit yang terletak pada posisi paling kiri atau bit pertama dari suatu *byte* data [21], [23]. Contoh MSB dapat dilihat seperti pada gambar 2.6.

00110101	**1**1010110	**0**1101010
11110100	**0**0111001	**0**1100001
11110001	**0**0010001	11100001

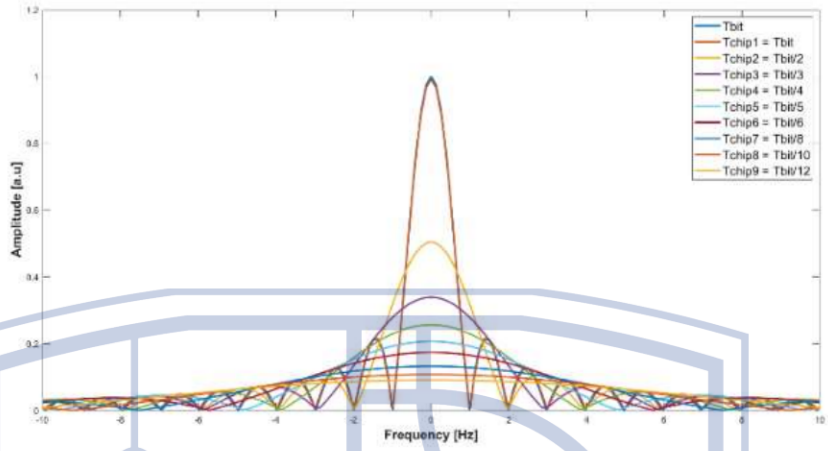
Gambar 2.6 MSB [21]

3. *Spread Spectrum*

   Spread Spectrum menyisipkan pesan dengan menyebarkannya menggunakan *pseudonoise code*, sehingga sulit dikenali karena dianggap sebagai *noise* oleh sistem. Pesan disisipkan dengan cara memperlakukan gambar penutup sebagai *noise* dan menambahkan *noise* semu [21], [24].

   Proses penyisipan pesan menggunakan metode *Spread Spectrum* ini terdiri dari tiga proses, yaitu *spreading*, modulasi, dan penyisipan pesan ke citra. Sedangkan proses ekstraksi pesan menggunakan metode *Spread Spectrum* ini terdiri dari tiga proses,

yaitu pengambilan pesan dari matriks frekuensi, demodulasi, dan *de-spreading* [21]. Contoh dasar *spread spectrum* dapat dilihat seperti pada gambar 2.7.



Gambar 2.7 Prinsip Dasar *Spread Spectrum* [25]

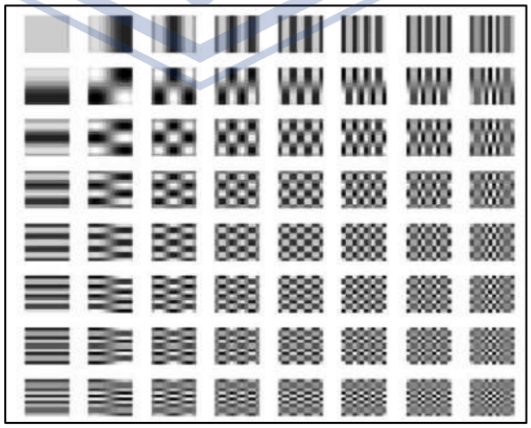
4. *Discrete Cosine Transform* (DCT)

DCT merupakan metode yang mengubah data dari domain waktu atau ruang menjadi domain frekuensi. Metode ini sering digunakan dalam kompresi gambar, seperti JPEG, karena mampu mengurangi kebutuhan penyimpanan. Dalam steganografi, pesan disisipkan ke dalam koefisien frekuensi [21], [26]. Persamaan perhitungan DCT dapat dilihat pada persamaan (4) berikut:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad \text{.....(4)}$$

Dan untuk $u = 0, \ldots, N-1$ dan $v = 0, \ldots, N-1$ yang di mana $N = 8$ dan $C(k) = \begin{cases} \frac{1}{\sqrt{2}} \; jika \; k \leq 0 \\ 1 \; jika \; k > 0 \end{cases}$

Gambar dasar DCT lengkap dengan ukuran N = 8 dapat dilihat sebagai berikut pada gambar 2.8.



Gambar 2.8 Dasar DCT Lengkap Ukuran N=8 [26]

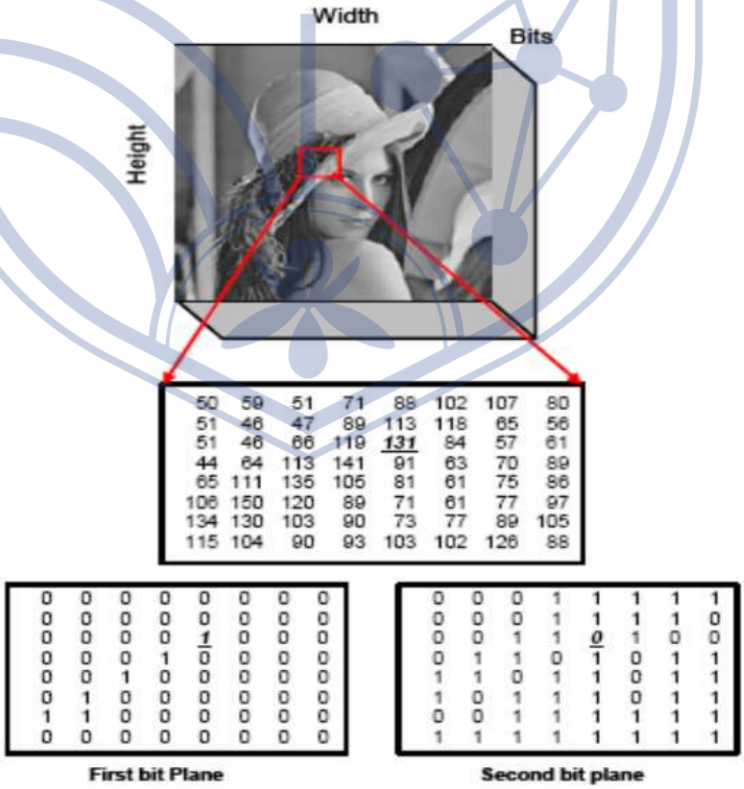
5. *Discrete Wavelet Transform* (DWT)

   DWT merupakan metode matematika yang digunakan untuk mendekomposisi citra digital menjadi sub-band dengan berbagai frekuensi dan orientasi spasial, seperti vertikal, horizontal, dan diagonal. Metode ini bekerja dengan memisahkan citra menjadi empat sub-band utama: LL (*low-low*), LH (*low-high*), HL (*high-low*), dan HH (*high-high*). LL berisi informasi frekuensi rendah dan mewakili citra kasar, sementara sub-band lainnya menampung detail frekuensi tinggi yang sering kali tidak terlalu sensitif terhadap penglihatan manusia [21], [27]. Persamaan umum untuk DWT dapat dilihat seperti pada persamaan (5) berikut:

   $$DWT\{f(t)\} = W_{\emptyset}(j_{\emptyset}, k) + W_{\emptyset}(j, k) \quad \text{.....(5)}$$

6. *Bit-Plane Complexity Segmentation* (BPCS)

   BPCS merupakan teknik steganografi yang memanfaatkan kompleksitas segmen gambar untuk menyembunyikan informasi rahasia. Metode ini membagi gambar menjadi beberapa *bit plane* yang diklasifikasikan menjadi dua kategori utama: *informative regions* (daerah informatif) dan *noise-like regions* (daerah mirip *noise*). Pesan rahasia disisipkan pada *noise-like regions*, di mana perubahan *pixel* tidak mudah dikenali oleh mata manusia, sehingga meningkatkan *imperceptibility* pesan dalam gambar [21], [28]. Konsep *bit plane slicing* dapat dilihat pada gambar 2.9.



Gambar 2.9 Konsep *Bit Plane Slicing* Jika *Pixel* Memiliki Nilai 131 [29]

2.2.2 Karakteristik Steganografi

Secara umum, karakteristik yang perlu diperhatikan saat menyembunyikan suatu data melalui sebuah medium seperti yang dituliskan oleh [26], [30], [31] sebagai berikut:

1. *Imperceptibility*

   Imperceptibility mengacu pada kemampuan steganografi untuk memastikan bahwa keberadaan informasi tersembunyi tidak dapat dideteksi oleh pihak yang tidak berwenang. Efektivitas *imperceptibility* diukur menggunakan metrik kualitas seperti PSNR (*Peak Signal-to-Noise Ratio*) dan SSIM (*Structural Similarity Index Metric*).
2. *Recovery*

   Recovery merupakan kemampuan untuk mengambil kembali pesan tersembunyi dari media penampung secara akurat tanpa kehilangan data atau kerusakan. Proses pemulihan ini bergantung pada ketahanan metode ekstraksi dan integritas algoritma steganografi. Efektivitas *recovery* diukur menggunakan metrik seperti SSIM.
3. *Fidelity*

   Fidelity mengacu kepada seberapa baik media penampung dapat mempertahankan kualitas aslinya setelah penyisipan pesan. Steganografi dengan *fidelity* tinggi memastikan bahwa gambar yang telah disisipi pesan tetap terlihat mirip dengan gambar aslinya, dengan degradasi yang minimal. *Fidelity* biasanya diukur menggunakan MSE (*Mean Squared Error*) dan PSNR (*Peak Signal-to-Noise Ratio*), di mana nilai MSE yang lebih rendah dan PSNR yang lebih tinggi menunjukkan *fidelity* yang lebih baik.
4. *Robustness*

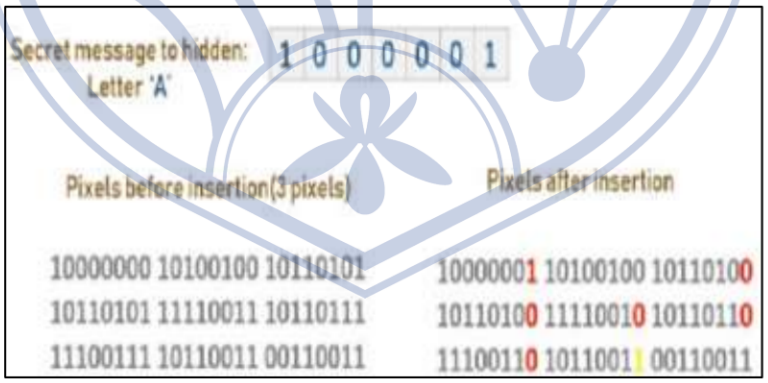
   Robustness mengacu pada seberapa baik pesan tersembunyi dapat bertahan terhadap berbagai bentuk pemrosesan gambar, kompresi, atau modifikasi lainnya pada *stego-image*. Metode steganografi yang tangguh memastikan bahwa pesan tetap utuh meskipun gambar mengalami transformasi seperti pemotongan (*cropping*), perubahan ukuran (*scaling*), atau penambahan *noise*. Karakteristik ini sangat penting agar informasi tersembunyi tetap dapat diambil meskipun gambar stego mengalami konversi format atau serangan tertentu. *Bit Error Rate* (BER) biasanya digunakan untuk mengukur *robustness* di mana nilai BER rendah berarti pesan semakin mirip dengan pesan aslinya [32].

2.2.3 Least Significant Bit (LSB)

Least Significant Bit adalah salah satu teknik steganografi paling populer dan sederhana yang digunakan untuk menyembunyikan pesan rahasia dalam citra digital. Metode ini bekerja dengan menyisipkan bit pesan rahasia ke dalam bit paling tidak signifikan dari setiap *pixel* pada gambar. Bit pesan rahasia diubah ke dalam bentuk biner dan kemudian didistribusikan di antara LSB dari setiap *pixel*. Karena perubahan ini terjadi pada bit terkecil, dampaknya pada kualitas visual gambar sangat minim, sehingga pesan tersembunyi sulit untuk dideteksi [22], [23].

LSB tergolong dalam teknik substitusi dalam *media-based steganography*, di mana bit paling kanan dari data citra diubah dengan bit informasi rahasia. Pada setiap *byte pixel*, bit LSB diganti dengan bit pesan yang akan disisipkan. Ini membuat citra hasil (*stego-image*) terlihat hampir sama dengan citra sampul (*cover image*) yang asli. Metode ini sering disebut sebagai *sequential embedding*, karena penyisipan dilakukan secara berurutan dari *pixel* pertama hingga terakhir.

Metode ini mudah dilakukan, karena bit pesan rahasia langsung dimasukkan ke dalam bit yang memiliki nilai paling rendah pada gambar. Namun, karena penyisipan dilakukan secara berurutan dari bit yang paling kanan, pesan menjadi mudah diekstraksi jika seseorang mencurigai adanya informasi rahasia dalam gambar. Kemudahan ekstraksi ini merupakan salah satu kelemahan dari teknik LSB [33]. Proses penyisipan pesan menggunakan metode LSB dapat dilihat pada gambar 2.10.



Gambar 2.10 Penyisipan Pesan LSB [34]

Seperti yang terlihat pada gambar, terdapat pesan yang mau disisipkan dan citra sampul. Pada gambar pesan yang disisipkan ditandai dengan warna merah, perubahan tersebut hanya mengubah nilai *byte* satu bit lebih tinggi atau satu bit lebih rendah. Manusia

tidak akan dapat melihat perbedaan antara citra sampul dan *stego-image* karena perbedaan yang sangat kecil, terutama jika pesan hanya disisipkan pada *channel* B dari gambar RGB [34].

2.3 Diffie-Hellman Key Exchange (DHKE)

Diffie-Hellman Key Exchange adalah algoritma pertukaran kunci asimetris yang memungkinkan dua pihak untuk berbagi kunci rahasia melalui saluran komunikasi yang tidak aman tanpa harus berbagi kunci sebelumnya [35], [36]. Pertukaran kunci ini didasarkan pada kesulitan dalam menghitung logaritma diskrit, yaitu sebuah permasalahan matematika yang juga menjadi dasar keamanan pada algoritma *El Gamal*. Nilai kunci yang dihasilkan bergantung kepada para peserta dan informasi mengenai kunci publik dan privat mereka masing-masing [37].

Untuk melakukan pertukaran kunci rahasia K dengan ukuran t *byte*, kedua pihak yang berkomunikasi, harus memiliki sebuah grup siklik G dan sebuah elemen generator a dari grup tersebut. Elemen dalam grup G, jika kita menuliskan operasinya secara perkalian akan berbentuk seperti persamaan (6) berikut:

$$1, a, a^2, \dots, a^{(s-1)} \quad (6)$$

Di mana:

1. s adalah jumlah elemen dari grup G.

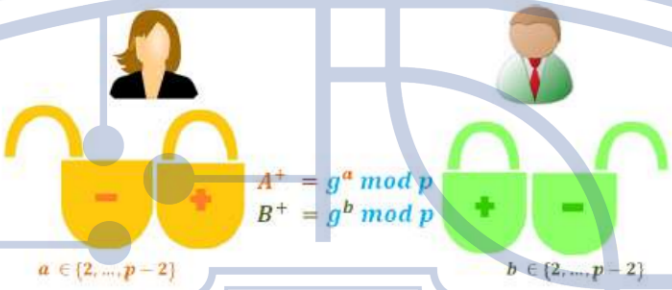
Grup G dapat berupa grup perkalian dari bilangan bulat modulo p yang relatif prima terhadap p, ditulis sebagai $(\frac{\mathbb{Z}}{p\mathbb{Z}})$. Di mana p merupakan bilangan prima besar dan a adalah elemen yang dapat menghasilkan seluruh elemen grup tersebut (generator) [37]. Prinsip kerja *Diffie-Hellman Key Exchange* adalah sebagai berikut:

1. Alice dan Bob sepakat dua parameter publik
    - p: bilangan prima besar.
    - g: bilangan yang disebut generator modulo p.
2. Alice dan Bob memilih kunci privat secara acak:
    - Alice memilih a
    - Bob memilih b
3. Masing-masing menghitung kunci publik
    - Alice menghitung $A = g^a \ mod \ p$ dan mengirimkan A ke Bob.

- Bob menghitung $B = g^b \ mod \ p$ dan mengirimkan B ke Alice.

4. Keduanya menghitung kunci rahasia bersama K
   - Alice: $K = B^a \ mod \ p$
   - Bob: $K = A^b \ mod \ p$
   - Karena sifat aritmetika modular, hasilnya sama dengan $K = g^{(a*b)} \ mod \ p$ [35], [36].

Proses *public/private key generation* dapat dilihat seperti pada gambar 2.11.

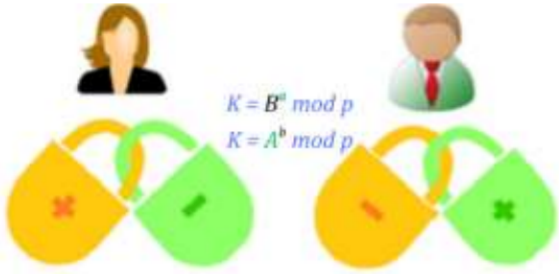


Gambar 2.11 *Public/Private Key Generation* [36]

Proses *key exchange* dapat dilihat seperti pada gambar 2.12.

Gambar 2.12 *Key Exchange* [36]

Secret key generation dapat dilihat seperti pada gambar 2.13.



Gambar 2.13 *Secret Key Generation* [36]

2.4 Generator Modulo

Adapun algoritma yang berkaitan dengan generator modulo sebagai berikut [38]:

1. *Greatest Common Divisor* (GCD)

GCD atau Faktor Persekutuan Terbesar (FPB) digunakan untuk menguji apakah sebuah bilangan yang lebih kecil dari m merupakan bilangan yang relatif prima. Dua bilangan dapat disebut relatif prima apabila $GCD\ (m, a) = 1$. Untuk mencari dua bilangan bulat yang relatif prima dapat menggunakan algoritme *Euclidean* karena algoritma ini didasarkan pada asumsi bahwa terdapat dua bilangan positif, yaitu m dan n, dengan m ≥ n. Berikut adalah tahapan algoritma *Euclidean*:

a. Apabila n = 0 maka m merupakan $FPB(m, n)$; stop. Namun jika $n \neq 0$, maka lanjut ke langkah 2.
b. Bagi nilai m dengan n dan misalkan sisanya adalah r; ganti nilai m dengan nilai n dan nilai n dengan nilai r, kemudian ulangi ke langkah a kembali.

2. Order Modulo

Bilangan asli z disebut sebagai order dari bilangan asli a dalam modulo bilangan asli m jika a dan m adalah relatif prima, $a\ z \equiv 1 \pmod m$, dan z adalah bilangan asli terkecil yang memiliki sifat ini. Karakteristik menarik yang terkait dengan order modulo, yaitu $a\ k \equiv 1 \pmod m$ dan z adalah order dari a modulo m maka z akan membagi k. Dengan demikian, z adalah bilangan asli terkecil yang memenuhi $a^k \equiv (\mathrm{mod}\ m)$ sehingga k harus < dari z. Bilangan k dapat dituliskan dalam bentuk $k = qz + r$ dengan q adalah bilangan asli dan r adalah bilangan cacah r yang memenuhi $0 \leq r < z$. Karakteristik jika dituliskan dapat dilihat pada persamaan (7) berikut:

$$\begin{aligned} &.a^k \equiv 1 \pmod m\ a \\ &a^{qz+r} \equiv 1 \pmod m \\ &(a^z)^q.a^r \equiv 1 \pmod m \\ &a^r \equiv 1 \pmod m \end{aligned} \quad \text{.....(7)}$$

3. Fungsi Phi *Euler* (∅)

Fungsi Phi *Euler* (∅) didefinisikan sebagai jumlah bilangan bulat positif yang lebih kecil dari bilangan bulat tertentu dan relatif prima terhadap bilangan bulat tertentu. Misalnya, jika kita memiliki bilangan bulat positif n, maka ∅(n) akan memberikan banyaknya bilangan bulat positif yang lebih kecil dari n dan relatif prima terhadap n. Meskipun disebut "phi", fungsi ini sama sekali tidak terkait dengan nilai phi (∅) yang dikenal sebagai bilangan emas

1,61803399. Penggunaan simbol phi (∅) di sini hanya untuk sebuah 'fungsi', jika n merupakan bilangan prima dapat dilihat seperti pada persamaan (8)

$$\emptyset(n) = n - 1 \quad \text{.....(8)}$$

4. Bilangan Prima

Bilangan prima adalah bilangan bulat positif yang lebih besar dari 1 dan hanya memiliki dua pembagi, yaitu 1 dan dirinya sendiri. Artinya, bilangan ini tidak bisa dibagi habis oleh angka lain selain dua pembagi tersebut. Contoh bilangan prima antara lain: 2, 3, 5, 7, 11, 13, 17, dan seterusnya.

Untuk menemukan bilangan prima, terutama dalam rentang angka yang kecil, salah satu metode paling sederhana yang dapat digunakan adalah metode *Sieve of Eratosthenes* (Saringan *Eratosthenes*). Metode ini bekerja dengan menyusun daftar angka dari 1 hingga n, lalu secara bertahap mengeliminasi kelipatan dari setiap bilangan prima yang ditemukan. Berikut adalah langkah-langkah algoritmanya:

a. Buat daftar angka dari 1 hingga n.
b. Tandai angka 1 sebagai bilangan prima (meskipun beberapa sumber menyatakan bahwa 1 bukan bilangan prima).
c. Tandai angka 2 sebagai bilangan prima, lalu coret semua kelipatan dari 2 karena angka-angka tersebut bukan bilangan prima.
d. Tandai angka 3 sebagai bilangan prima, kemudian coret seluruh kelipatan 3 dari daftar.
e. Ulangi proses ini untuk angka berikutnya yang belum dicoret, terus menerus hingga seluruh angka non-prima telah dieliminasi.
f. Angka-angka yang tersisa dan tidak tercoret dalam daftar tersebut adalah bilangan prima.

5. *Fast Exponential*

Fast Exponential digunakan untuk melakukan operasi pemangkatan dengan cepat pada bilangan bulat modulo. Dalam metode ini, ekspansi biner dari eksponen dimanfaatkan. Misalkan kita memiliki himpunan G, dan $g \in G$, sedangkan z adalah bilangan bulat positif. Untuk menghitung g^z menggunakan metode *fast exponentiation*, langkah-langkahnya adalah sebagai berikut:

a. Hitung $g^{2^i}, 0 \leq 1 < k$.
b. Nilai g^z adalah hasil perkalian dari nilai-nilai g^{2^i}, dengan $a_1 = 1$. Diperoleh persamaan (9):

$$g^{2^{i+1}} = \left(g^{2^i}\right)^2 \quad (9)$$

Penelitian [39] menuliskan jika p adalah bilangan prima, maka himpunan bilangan $\{1,2,\dots,p-1\}$ membentuk perkalian modulo p, yang disebut $(\frac{\mathbb{Z}}{p\mathbb{Z}})$. Akar primitif dari p adalah bilangan $g \in (\frac{\mathbb{Z}}{p\mathbb{Z}})$ yang dapat menghasilkan semua elemen grup melalui perpangkatan atau dengan kata lain g adalah generator dari $(\frac{\mathbb{Z}}{p\mathbb{Z}})$ seperti yang dapat dilihat pada persamaan (10) berikut:

$$\{g^1 \bmod p, g^2 \bmod p, \dots, g^{p-1} \bmod p\} = \{1, 2, \dots, p-1\} \quad (10)$$

Untuk mencari akar primitif dari bilangan prima, digunakan pendekatan berbasis teori ordo dan Teorema Kecil Fermat. Langkahnya adalah sebagai berikut:

1. Menghitung nilai dari $p-1$.
2. Memfaktorkan bilangan $p-1$ menjadi faktor prima seperti yang dapat dilihat pada persamaan (11)

   $$p - 1 = q_1^{k_1} \cdot q_2^{k_2\prime} \cdot \dots \cdot q_n^{k_n} \quad (11)$$

   Di mana:

   1. $p-1$ merupakan jumlah elemen dalam grup perkalian modulo p, yaitu $(\frac{\mathbb{Z}}{p\mathbb{Z}})$.
   2. $q_1, q_2, \dots, q_n$ merupakan faktor-faktor prima dari $p-1$.
   3. $k_1, k_2, \dots, k_n$ merupakan pangkat dari masing-masing faktor prima, jadi $q_i^{k_i}$ menunjukkan bahwa q_i muncul sebanyak k_i kali dalam faktorisasi.
3. Uji semua nilai g dari 2 hingga $p-1$. Untuk setiap g seperti pada persamaan (12) berikut:

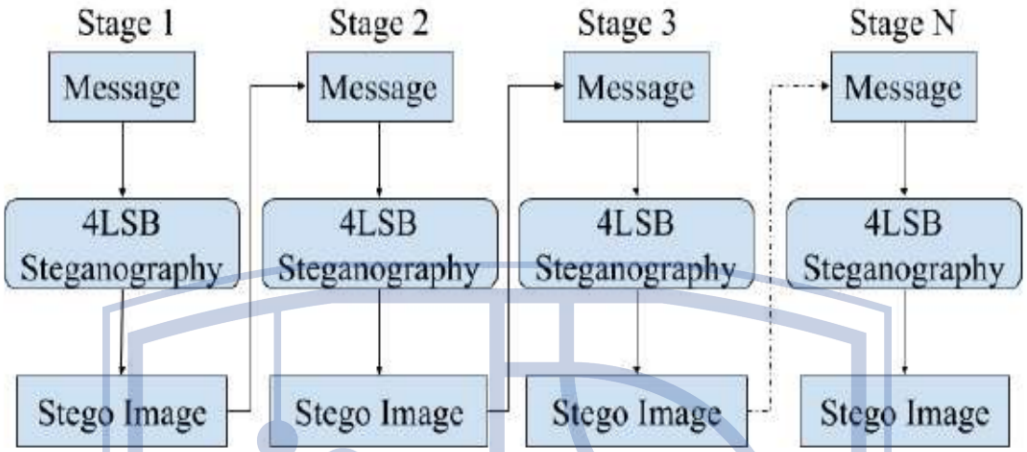
   $$g^{\frac{(p-1)}{q_i}} \bmod p \neq 1 \text{ untuk semua } q_i \quad (12)$$

   Jika semua hasil tidak sama dengan 1, maka g adalah akar primitif dari p.

2.5 Reversible-Enhanced Stego Block Chaining (RESBC)

Stego Block Chaining (SBC) adalah teknik steganografi yang didasarkan pada konsep *Cipher Block Chaining* (CBC) dalam kriptografi. Dalam metode ini, pesan rahasia disisipkan dalam gambar penutup menggunakan 4 *Least Significant Bits* (4 LSB) steganografi dan hasil *stego-image* dari satu tahap digunakan sebagai input untuk tahap berikutnya. SBC meningkatkan keamanan dengan menyusun tahap-tahap steganografi secara berantai, tetapi memiliki kelemahan yaitu meningkatkan ukuran data yang

ditransmisikan sehingga meningkatkan kebutuhan *bandwidth* dan waktu transmisi [40]. Implementasi teknik SBC dapat dilihat seperti pada gambar 2.14.



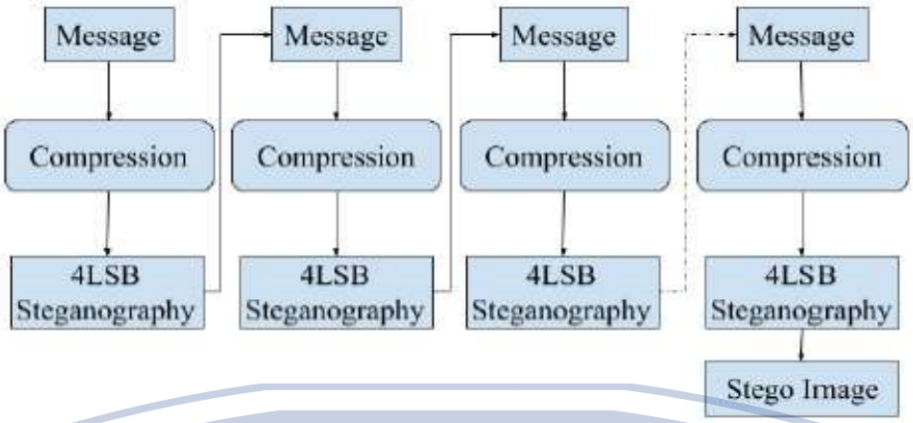
Gambar 2.14 Implementasi Teknik SBC [8]

Enhanced Stego Block Chaining (ESBC) dikembangkan untuk mengatasi masalah tersebut dengan mengurangi kebutuhan *bandwidth* dan waktu transmisi hingga setengah dari SBC. ESBC masih mempertahankan keamanan berjenjang seperti SBC, tetapi lebih efisien dalam hal kapasitas penyimpanan dan kecepatan transmisi [41].

Reversible Enhanced Stego Block Chaining (RESBC) adalah pengembangan lebih lanjut dari ESBC yang bertujuan untuk menyelesaikan masalah pemulihan informasi yang tidak sempurna pada ESBC. RESBC memastikan pemulihan data 100% serta meningkatkan efisiensi penyembunyian data dengan menggunakan teknik kompresi *lossless* seperti *Run-Length Encoding* (RLE), *Huffman Encoding* (HE), *Shannon-Fano Algorithm* (SFA), dan *Lempel-Ziv-Welch* (LZW) sebelum data disisipkan dalam gambar penutup [8].

Cara kerja RESBC yaitu:

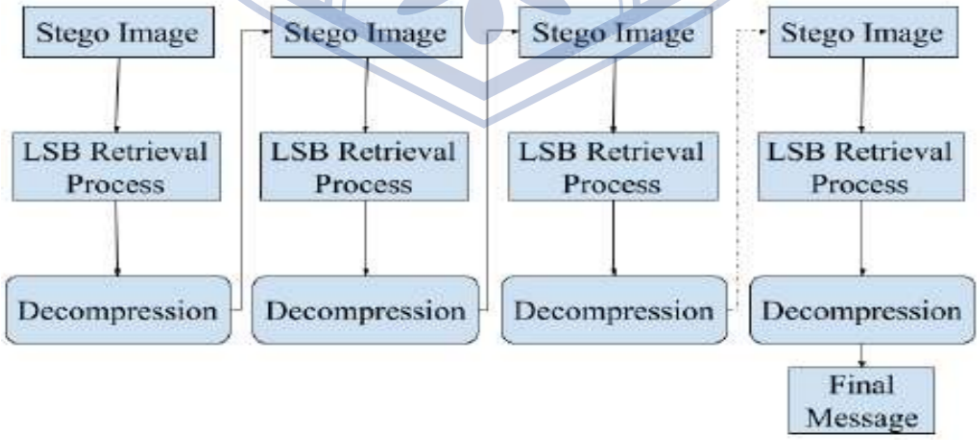
1. Pesan dikompresi menggunakan algoritma *lossless compression* seperti *Lempel-Ziv-Welch*, *Run Length Encoding* dan lainnya.
2. Pesan yang dikompresi dimasukkan ke dalam gambar sampul menggunakan teknik 4 LSB.
3. *Stego-image* yang dihasilkan kemudian akan digunakan sebagai pesan rahasia untuk tahap berikutnya, proses ini diulangi hingga tahap ke-N seperti yang terlihat pada gambar 2.15.



Gambar 2.15 Implementasi Teknik RESBC [8]

Proses penerimaan pesan rahasia yaitu:

1. Proses dimulai saat penerima memperoleh *file stego-image* yang telah disisipkan dengan pesan rahasia oleh pengirim.
2. Pada setiap *pixel* dari gambar yang diterima dari pengirim, penerima akan mengambil 4 bit LSB terakhir dari setiap *pixel* karena pada bit tersebut pesan rahasia disisipkan sebelumnya.
3. Bit yang telah dibaca membentuk pesan terkompresi, sehingga perlu melakukan proses dekompresi menggunakan algoritma kompresi yang sama dengan yang digunakan oleh pengirim.
4. Data yang didekompresi dianggap sebagai *stego-image* dan diproses kembali dengan membaca 4 bit LSB dan dekompresi seperti langkah kedua dan ketiga.
5. Proses ini diulangi hingga pesan asli dapat dipulihkan secara utuh seperti yang terlihat pada gambar 2.16.



Gambar 2.16 Proses Ekstraksi Pesan RESBC [8]

2.6 Metrik Pengukuran

Untuk mengukur kualitas suatu metode steganografi diperlukan suatu pengujian secara objektif. Pengujian tersebut dilakukan dengan menggunakan metrik pengukuran, metrik pengukuran yang umum digunakan untuk menguji kualitas steganografi antara lain:

2.6.1 MSE

Mean Squared Error (MSE) adalah metrik yang digunakan untuk mengukur perbedaan antara citra asli dengan citra yang telah dimodifikasi setelah penyisipan pesan steganografi. Nilai MSE mendekati 0 dianggap memiliki perubahan yang lebih sedikit dibandingkan gambar asli yang digunakan. Semakin kecil nilai MSE, semakin sedikit perbedaan antara gambar asli dan *stego-image*, yang berarti metode steganografi lebih baik dalam mempertahankan kualitas visual gambar [5], [26], [30], [42]. Rumus MSE dapat dilihat pada persamaan (13) berikut:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(x_{i,j} - x'_{i,j}\right)^2 \quad \text{.....(13)}$$

Di mana:

1) $x_{i,j}$ adalah nilai *pixel* pada gambar asli.
2) $x'_{i,j}$ adalah nilai *pixel* pada *stego-image*.
3) M dan N adalah ukuran gambar (jumlah baris dan kolom).

2.6.2 PSNR

Peak Signal-to-Noise Ratio (PSNR) adalah metrik yang digunakan untuk mengukur kualitas citra setelah mengalami perubahan, seperti setelah penyisipan pesan steganografi. PSNR membandingkan tingkat sinyal maksimum dalam gambar dengan tingkat *noise* yang ditimbulkan akibat modifikasi, nilai PSNR dikategorikan menjadi baik jika berada diantara 30 – 50 dB dan tidak diterima jika dibawah 30 dB [5], [30], [42], [43]. Rumus PSNR dapat dilihat pada persamaan (14) berikut:

$$PSNR = 10\, log_{10} \left(\frac{255^2}{\frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(x_{i,j} - x'_{i,j}\right)^2} \right) \quad \text{.....(14)}$$

Di mana:

1) 255 adalah nilai maksimum *pixel* dalam citra 8 bit (0-255)
2) $x_{i,j}$ adalah nilai *pixel* pada gambar asli.
3) $x'_{i,j}$ adalah nilai *pixel* pada *stego-image*.

4) M dan N adalah ukuran gambar (jumlah baris dan kolom).

2.6.3 SSIM

Structural Similarity Index (SSIM) adalah metrik yang digunakan dalam pemrosesan citra untuk mengukur tingkat kemiripan antara dua gambar. SSIM menilai kualitas gambar berdasarkan tiga komponen utama: pencahayaan (*luminance*), kontras, dan struktur, yang semuanya dirancang untuk menyerupai cara manusia menilai visual.

Metrik ini banyak digunakan dalam penelitian untuk menilai kualitas hasil kompresi gambar, penghilangan *noise* (*denoising*), serta pemulihan gambar (*restoration*). Dibandingkan metrik sederhana seperti MSE atau PSNR, SSIM lebih akurat dalam mencerminkan persepsi visual manusia terhadap perbedaan gambar. SSIM memiliki rentang nilai 0 hingga 1 walaupun sebenarnya bisa mencapai -1 namun jarang terjadi sehingga tidak relevan, semakin tinggi nilai SSIM semakin baik citra sampul menyembunyikan citra pesan menandakan teknik steganografi yang digunakan dapat menyembunyikan pesan dengan efektif [5], [26], [42], [43]. Rumus SSIM dapat dilihat seperti pada persamaan (15) berikut:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C1)(2\sigma_{xy}+C2)}{(\mu_x^2+\mu_y^2+C1)(\sigma_x^2+\sigma_y^2+C2)} \quad \text{.....(15)}$$

Di mana:

1) μ_x, μ_y merupakan nilai rata-rata intensitas *pixel* dari citra x dan y, serta mewakili komponen kecerahan dari masing-masing citra.
2) σ_x^2, σ_y^2 merupakan varians dari citra x dan y yang menunjukkan kontras dari masing-masing citra.
3) $\sigma_{x,y}$ merupakan kovarians antara citra x dan y yang merepresentasikan kesamaan pola antara dua citra.
4) C_1, C_2 merupakan konstanta kecil yang digunakan untuk mencegah pembagian dengan nol. Biasanya didefinisikan oleh persamaan (16)

   $$C_1 = (K_1L)^2, C_2 = (K_2L)^2 \quad \text{.....(16)}$$

   Di mana:

   1) L merupakan rentang dinamis *pixel* (misalnya 255 untuk gambar 8-bit)
   2) $K_1 \approx 0.01, K_2 \approx 0.03$

2.7 Lempel Ziv Welch (LZW)

Lempel-Ziv-Welch (LZW) algoritma kompresi *lossless* yang dirancang untuk mengurangi ukuran *file* tanpa kehilangan data asli. Algoritma ini bertujuan untuk mengurangi ukuran data tanpa kehilangan informasi asli dengan menggantikan fragmen-fragmen teks yang berulang menggunakan indeks yang diperoleh dari sebuah *dictionary* (kamus). LZW sering digunakan dalam berbagai format *file*, termasuk teks, gambar, dan suara, karena strukturnya yang sederhana dan kemampuannya dalam mengurangi ukuran *file* tanpa mengorbankan kualitas [44], [45].

Langkah–langkah dalam proses algoritma LZW yaitu:

1. *Dictionary* (kamus) diinisiasikan dengan karakter dasar ASCII (0-255), di mana setiap karakter diberikan indeks unik.
2. P adalah karakter pertama dalam *stream* data.
3. Q adalah karakter berikutnya dalam *stream* data.
4. Lakukan pengecekan apakah P+Q terdapat pada *dictionary*:
    a. Jika iya, gabungkan P+Q menjadi *string* baru dan lanjutkan dengan menambahkan karakter berikutnya.
    b. Jika tidak, *output string* P sebagai kode, tambahkan P+Q ke *dictionary* dengan indeks baru lalu tetapkan P = Q.
5. Ulangi proses hingga semua karakter dalam *stream* data selesai diproses.
6. Ketika tidak ada karakter lagi, kode akhir untuk *string* P dikeluarkan dan proses berhenti.

Proses dekompresi LZW yaitu:

1. Kamus diinisiasikan kembali dengan karakter dasar ASCII.
2. Data terkompresi dibaca sebagai kode dan dicocokkan dengan entri *dictionary*.
3. Setiap kode yang ditemukan diterjemahkan kembali menjadi *string* aslinya dan ditambahkan ke *dictionary*.
4. Proses ini diulangi hingga seluruh data didekompresi sepenuhnya.