BAB II
KAJIAN LITERATUR

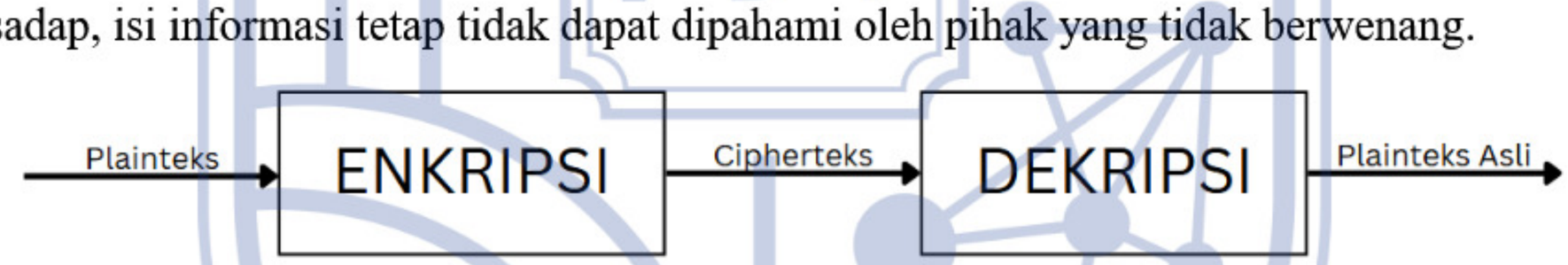
2.1 Kriptografi

Kriptografi merupakan ilmu yang mempelajari teknik untuk menjaga keamanan informasi melalui proses transformasi data sehingga tetap terlindungi dari akses pihak yang tidak berwenang. Kriptografi memiliki peran penting antara pengirim dan penerima untuk memastikan bahwa pesan yang terkirim tidak dapat dibaca atau dimodifikasi oleh penyusup. Pesan asli (p*laintext*) diubah menjadi *Ciphertext* melalui proses yang disebut enkripsi, dan hanya dapat dikembalikan ke bentuk semula melalui dekripsi oleh pihak yang memiliki kunci yang benar [10]. Secara matematis, proses ini di representasikan melalui fungsi enkripsi E dan fungsi dekripsi D sebagai berikut:

$$E(M) = C \quad \text{.....(1)}$$

$$D(C) = M \quad \text{.....(2)}$$

di mana M adalah *message* atau *plaintext,* sedangkan C adalah *Ciphertext.* Transformasi matematis ini bertujuan menjamin kerahasiaan data sehingga meskipun pesan berhasil disadap, isi informasi tetap tidak dapat dipahami oleh pihak yang tidak berwenang.



Gambar 2. 1 Enkripsi dan Dekripsi [10]

Kriptografi modern memanfaatkan berbagai algoritma dan pendekatan matematika untuk menjamin keamanan dalam penyimpanan maupun pengiriman data. Penerapan kriptografi sangat penting dalam menjaga keamanan komunikasi jaringan komputer, perlindungan data digital, serta interaksi aman antar pengguna yang saling berkomunikasi melalui media internet [11][12]. Tujuan utama kriptografi adalah menyediakan tingkat keamanan yang ditambahkan ke data selama pemrosesan, penyimpanan, dan komunikasi [10]. Untuk mencapai tujuan tersebut kriptografi menyediakan beberapa layanan keamanan yang meliputi:

1. Kerahasiaan (*confidentiality*)

   Kerahasiaan bertujuan untuk menjaga agar informasi hanya dapat diakses dan dibaca oleh pihak yang berwenang [10].

2. Autentikasi (*authentication*)

   Autentikasi bertujuan untuk memverifikasi identitas pihak yang berkomunikasi serta memastikan keaslian sumber pesan yang diterima [10].

3. Integritas data (*data integrity*)

   Integritas data bertujuan untuk menjamin bahwa informasi tetap utuh dan tidak mengalami perubahan selama proses pengiriman [10].

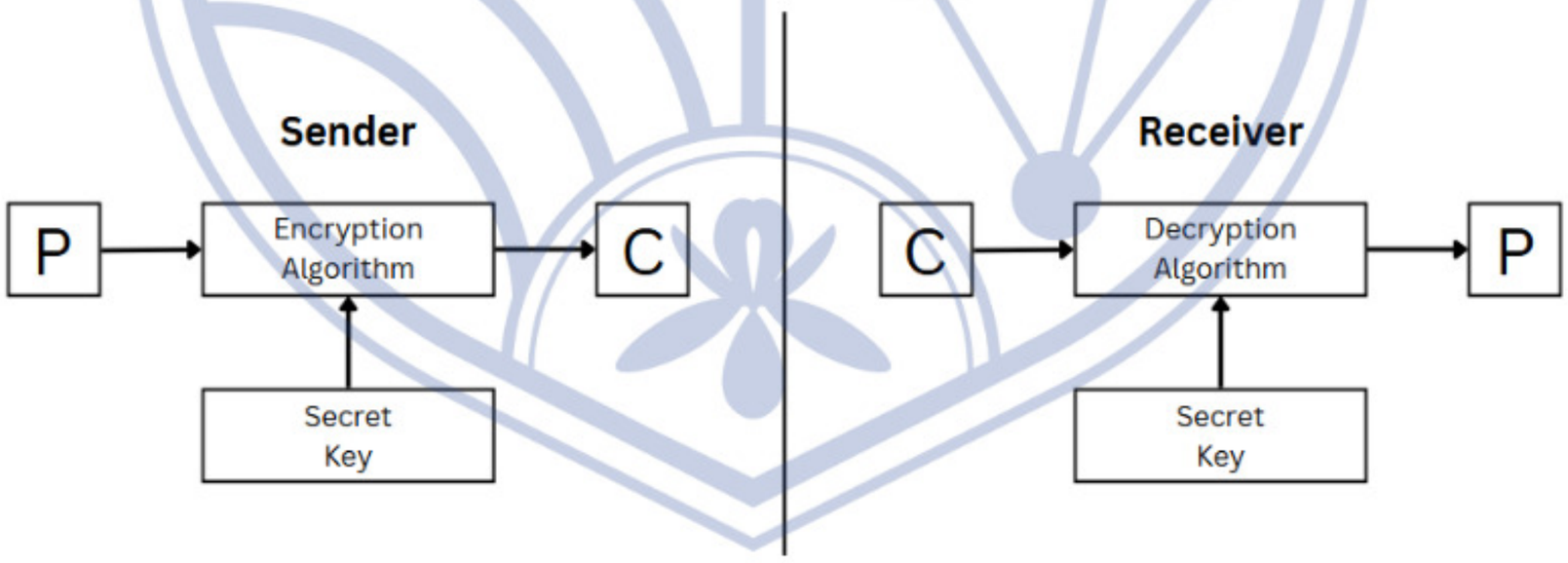
4. Tidak ada penyangkalan (*non-repudiation*)

   Non-penyangkalan bertujuan untuk mencegah pihak yang terlibat dalam komunikasi menyangkal tindakan yang telah dilakukannya dalam proses pertukaran informasi [10].

Jenis kriptografi secara umum terbagi menjadi dua kategori utama:

1. Kriptografi Kunci Simetris (*Symmetric Key Cryptography*)

Kriptografi simetris merupakan algoritma yang menggunakan satu kunci rahasia yang sama untuk proses enkripsi dan dekripsi. Pada sistem ini, kunci enkripsi dapat dihitung dari kunci dekripsi dan sebaliknya, bahkan pada sebagian besar kasus kedua kunci tersebut identik. Oleh karena itu, pengirim dan penerima harus terlebih dahulu menyepakati kunci tersebut sebelum dapat melakukan komunikasi aman. Kriptografi simetris terbagi menjadi dua kategori, yaitu *stream cipher* yang memproses data per bit atau per byte dan *block cipher* yang memproses data dalam kelompok (block) berukuran tetap, misalnya 64-bit atau 128-bit [10][11].

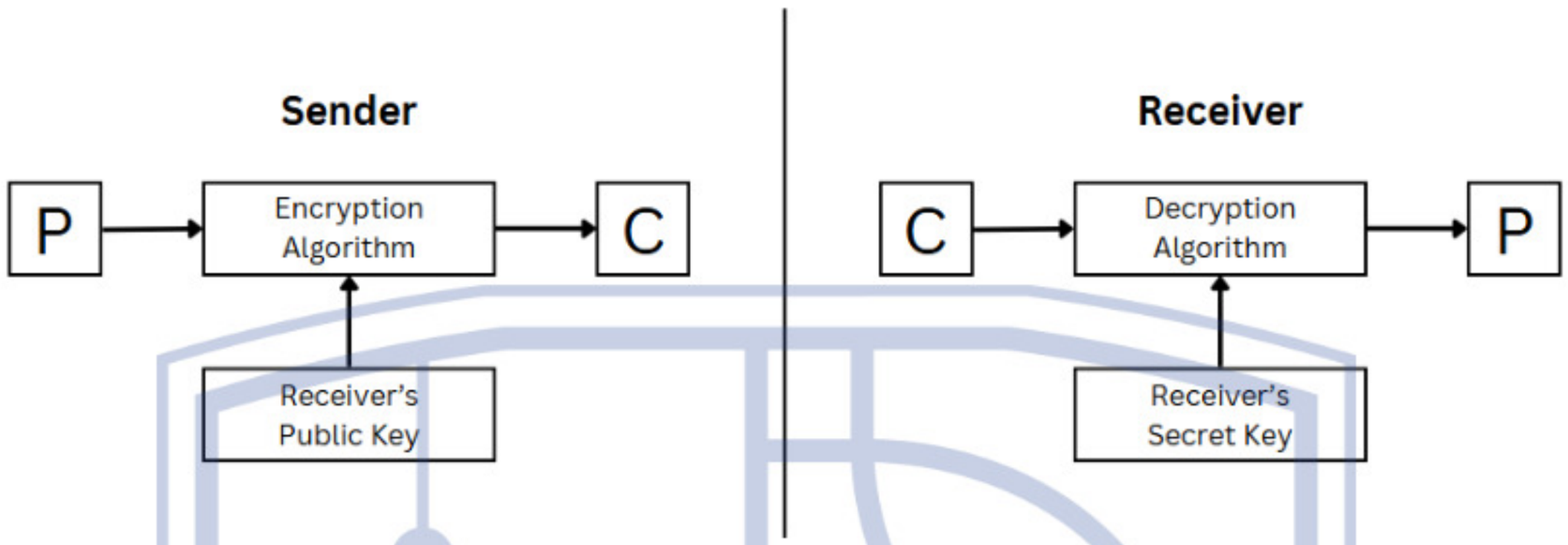


Gambar 2. 2 Algoritma Simetris [10]

2. Kriptografi Kunci Asimetris *(Asymmetric / public key cryptography)*

   Kriptografi asimetris atau *public-key* dikembangkan untuk mengatasi kelemahan utama kriptografi simetris, yaitu masalah distribusi kunci. Pada sistem ini, setiap pengguna memiliki sepasang kunci, yaitu kunci publik yang dapat dibagikan kepada semua pihak dan kunci privat yang dijgaga kerahasiaannya. Pesan dapat dienkripsi menggunakan public *key*

dan hanya dapat didekripsi menggunakan private *key*, atau sebaliknya pesan dapat dienkripsi dengan private *key* dan diverifikasi menggunakan public *key*, misalnya pada implementasi tanda tangan digital. Sistem ini juga umum digunakan untuk aplikasi seperti *digital signature, key exchange,* dan autentikasi pengguna [10][11].



Gambar 2. 3 Algoritma Asimetris [10]

2.2 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) adalah standard kriptografi kunci simetris modern yang ditetapkan oleh NIST pada tahun 2001, yang secara teknis didasarkan pada algritma *Rijndael.* AES dirancang untuk memproses data dalam block 128bit yang diatur dalam bentuk matriks 4x4 *byte* yang disebut *state array.* AES menyediakan tiga pilihan panjang kunci. Panjang kunci ini secara langsung menentukan jumlah putaran (*rounds*) yang harus dieksekusi oleh algoritma, yang detailnya di sajikan dalam Tabel 2.1, menunjukkan korelasi langsung antara panjang kunci dan kompleksitas keamanan yang ditawarkan.\

Table 2. 1 Kombinasi *Key* Block Round [10]

	Key length		**Block size**		**Number of rounds**
	Nk	(in bits)	*Nb*	(in bits)	*Nr*
AES-128	4	128	4	128	10
AES-192	6	192	4	128	12
AES-256	8	256	4	128	14

1. Notasi dan Konvensi

Pada subbab ini dijelaskan notasi dan konvensi yang digunakan dalam algoritma Advanced Encryption Standard (AES), yang mencakup representasi input dan output, byte, array byte, state, serta state sebagai array kolom.

1. Input dan Output

   Data Input dan Output selalu berukuran 128 bit. Ukuran bit ini disebut block, dan jumlah bit di dalamnya disebut panjang blok. Sementara itu, cipher *key* pada AES hanya boleh memiliki panjang 128, 192, atau 256 bit.

2. Byte

   Unit dasar yang digunakan pada algoritma AES, yaitu kumpulan 8bit yang diperlakukan sebagai satu kesatuan. Byte-byte dari input, output, atau Cipher *Key* akan membentuk sebuah array. Array tersebut ditulis sebagai an atau a[n], dengan indeks n bergantung pada panjang kunci atau blok, yaitu:

   - Kunci 128 bit -> indeks 0 – 15
   - Kunci 192 bit -> indeks 0 – 23
   - Kunci 256 bit -> indeks 0 – 31

   Setiap byte dalam AES disusun dari delapan bit $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. Byte ini juga dapat dipandang sebagai elemen dalam finite field menggunakan bentuk polinomial:

   $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_i x^i$ .....(3)

   Sebagai contoh Byte {01100011} mewakili elemen polinomial $x^6 + x^5 + x + 1$.

3. Array Byte

   Array byte pada AES dituliskan dalam bentuk a0 a1 a2 ... a15. Byte-byte ini berasal dari input 128bit. Setiap byte dibentuk dari urutan bit input seperti gambar berikut:

   Table 2. 2 Indeks byte dan bit [10]

Bit index in sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
Byte index	0								1								…
Bit index in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…

4. State

   Algoritma AES memproses data menggunakan sebuah array dua dimensi berisi byte yang disebut state. State terdiri dari empat baris, dan jumlah kolomnya adalah Nb, yaitu panjang blok (128 bit) dibagi 32. Karena panjang blok AES selalu 128 bit, maka Nb=4, sehingga state berukuran 4x4 byte.

   Setiap byte di dalam state memiliki dua indeks:

   - r untuk baris ($0 \leq r < 4$)
   - c untuk kolom ($0 \leq c < Nb$)

   Sebuah byte pada state bisa dituliskan sebagai $s_{r,c}$ atau $s_{[r,c]}$.

5. State sebagai array of kolom

Setiap kolom pada state array AES terdiri dari empat byte yang bersama-sama membentuk satu word berukuran 32 bit. Indeks baris menunjukkan posisi byte di dalam word, sedangkan indeks kolom menunjukkan posisi word di dalam state. Dengan demikian, State dapat dipandang sebagai array satu dimensi yang berisi 32-bit words. Bentuk penyajiannya dapat dituliskan sebagai berikut:

$w_0 = s_{0,0}\ s_{1,0}\ s_{2,0}\ s_{3,0}$ .....(4)

$w_1 = s_{0,1}\ s_{1,1}\ s_{2,1}\ s_{3,1}$ .....(5)

$w_2 = s_{0,2}\ s_{1,2}\ s_{2,2}\ s_{3,2}$ .....(6)

$w_3 = s_{0,3}\ s_{1,3}\ s_{2,3}\ s_{3,3}$ .....(7)

2. Spesifikasi Algoritma

Pada algoritma AES, ukuran blok input, output, dan state selalu tetap yaitu 128 bit. Ukuran ini dinyatakan sebagai Nb = 4, yang berarti state memiliki empat kolom word berukuran 32 bit. Sementara itu, panjang Cipher *Key* (k) dalam AES dapat berukuran 128, 192, atau 256 bit, dan masing-masing direpresentasikan sebagai Nk = 4, 6, atau 8, sesuai dengan jumlah word 32bit yang membentuk kunci.

Jumlah putaran (round) yang dijalankan dalam AES ditentukan oleh panjang kuncinya. Nilainya dinyatakan sebagai Nr, yaitu 10 putaran untuk kunci 128 bit (Nk = 4), 12 putaran untuk kunci 192 bit (Nk = 6), dan 14 putaran untuk kunci 256 bit (Nk = 8). Hanya kombinasi ukuran kunci, ukuran blok, dan jumlah putaran ini yang sesuai dengan standar AES.

Dalam proses Cipher maupun Inverse Cipher, algoritma AES menjalankan beberapa langkah utama, yaitu:

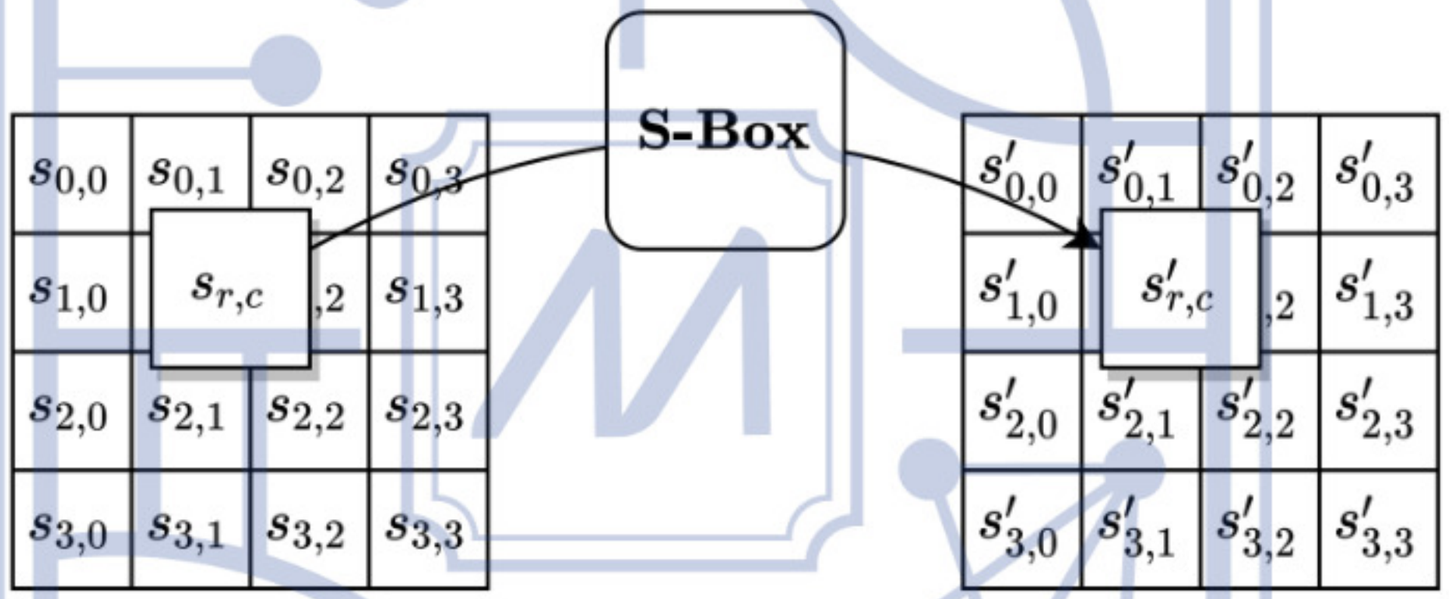
1. SubBytes

SubBytes merupakan langkah substitusi pada AES yang mengganti setiap byte dalam State menggunakan tabel S-Box. Setiap byte dipetakan ke baris dan kolom S-Box berdasarkan nilai heksadesimalnya. Hasil dari tabel tersebut menjadi byte baru.

Table 2. 3 S-Box pada AES [13]

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Adapun transformasi SubBytes pada state dapat dilihat pada gambar berikut :



Gambar 2. 4 Penerapan SubBytes() S-box untuk byte pada state [13]

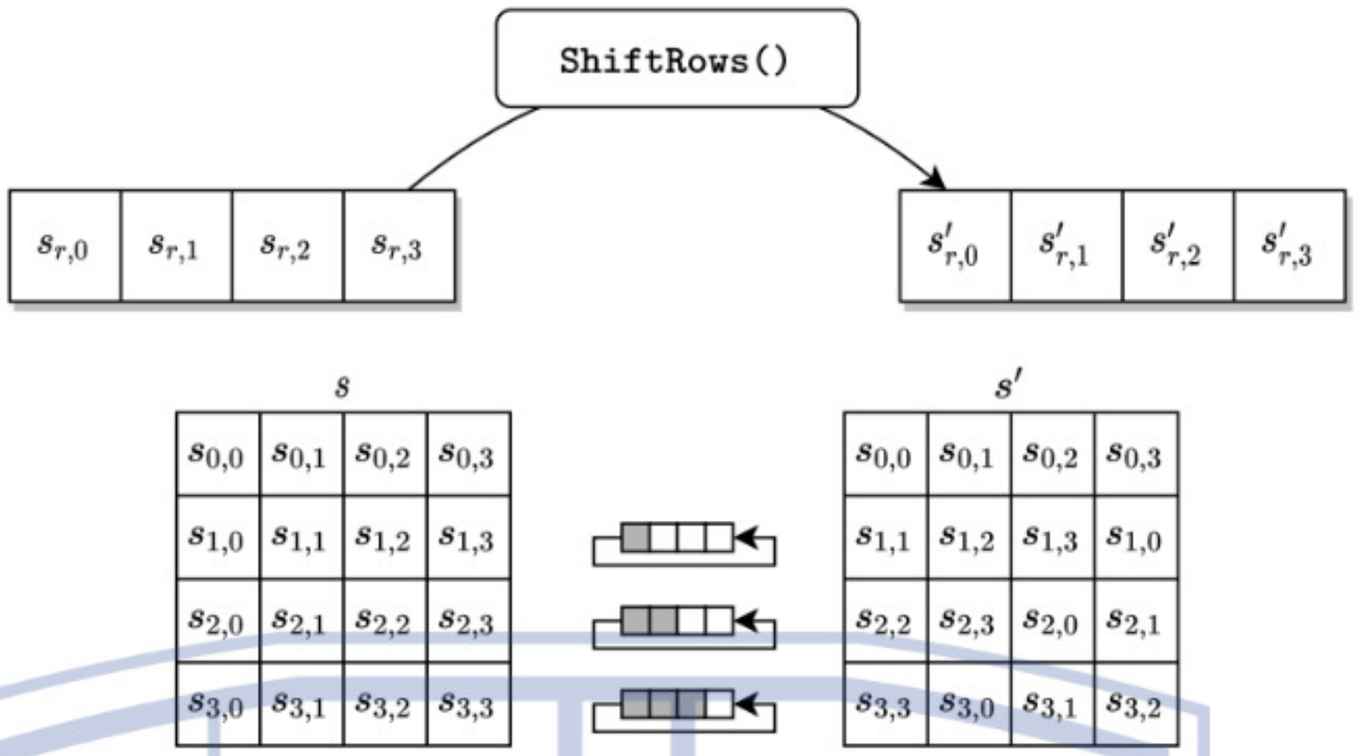
Contoh jika byte yang akan di proses adalah {53}, maka proses subtitusi dilakukan dengan mencari nilai pada table S-Bos sesuai indeks heksadesimalnya. Byte {53} berarti baris 5 dan kolom 3 pada S-Bos. Dengan mengikuti percabangan heksadesimal seperti pada gambar S-Bos nilai hasil subtitusinya adalah {ed}.

2. ShiftRows

ShiftRows adalah trasformasi pada state dimana byte-byte pada tiga baris terakhir digeser secara siklis. Jumlah pengeseran ditentukan oleh indeks baris r. Aturannya adalah :

$s'_{r,c} = s_{r,(c+r)\ mod\ 4}$ Untuk $0 \leq r < 4$ dan $0 \leq c < 4$ .....(8)

Ilustrasi ShiftRows dapat dilihat pada gambar di bawah ini. Dalam tampilan tersebut, setiap baris digeser ke kiri sebanyak r posisi, dan byte yang “keluar” dari sisi kiri akan kembali muncul di ujung kanan baris. Baris pertama (r = 0) tidak mengalami perubahan.[13]



Gambar 2. 5 Penggeseran secara cyclic ShiftRows tiga baris terakhir dalam state [13]

3. MixColumns

MixColumns adalah transformasi yang bekerja pada state dengan mengalikan setiap kolom, yang terdiri dari empat byte, dengan matriks tetap tunggal. Matriks tetap yang digunakan didefinisikan sebagai:

[a0, a1, a2, a3] = [{02}, {01}, {01}, {03}] .....(9)

Secara matematis, transformasi ini diwakili oleh perkalian matriks berikut untuk setiap kolom c (dimana $0 \leq c < 4$):

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad \text{.....(10)}$$

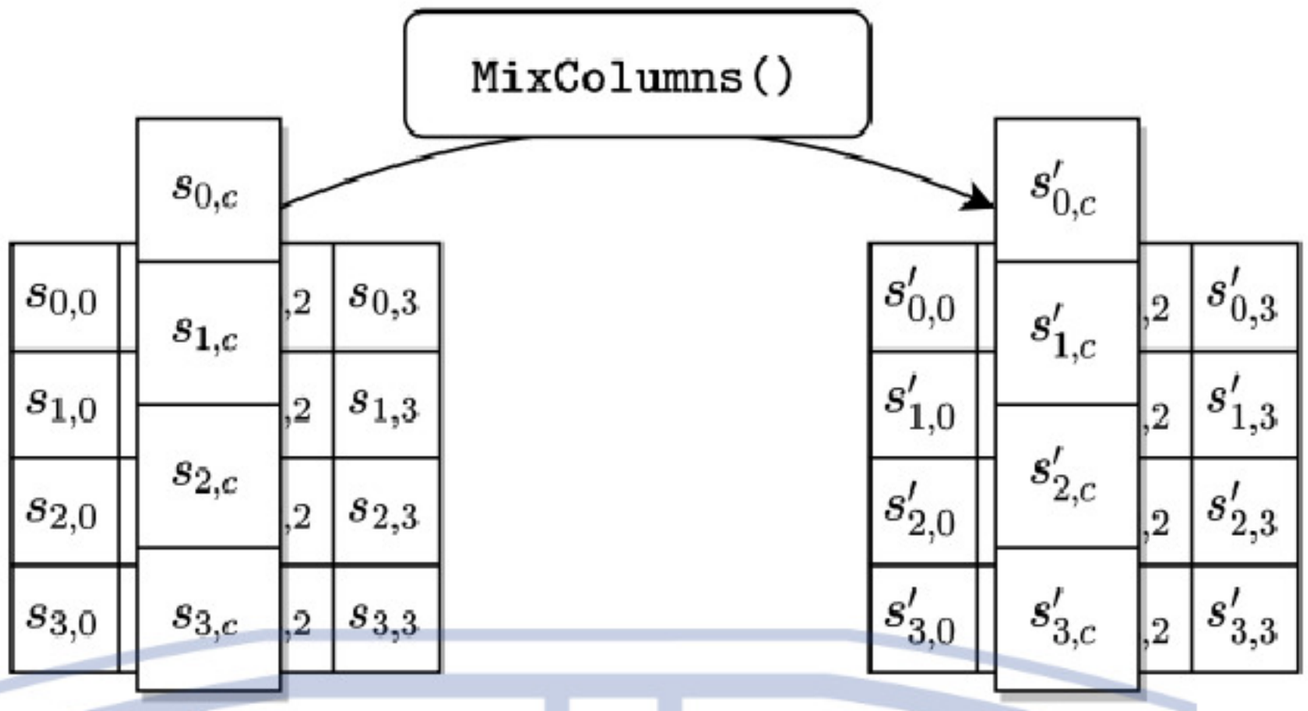
Matriks ini menunjukkan bagaimana byte output individual ($s'_{i,c}$) dihitung sebagai hasil dari operasi perkalian (ditandai dengan $\cdot$) dan XOR ($\oplus$) dari byte input ($s_{i,c}$) pada kolom yang sama. Ekspresi untuk byte output didefinisikan sebagai:

$$s'_{o,c} = (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \quad \text{.....(11)}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c} \quad \text{.....(12)}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c}) \quad \text{.....(13)}$$

$$s'_{3,c} = (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \cdot s_{3,c}) \quad \text{.....(14)}$$



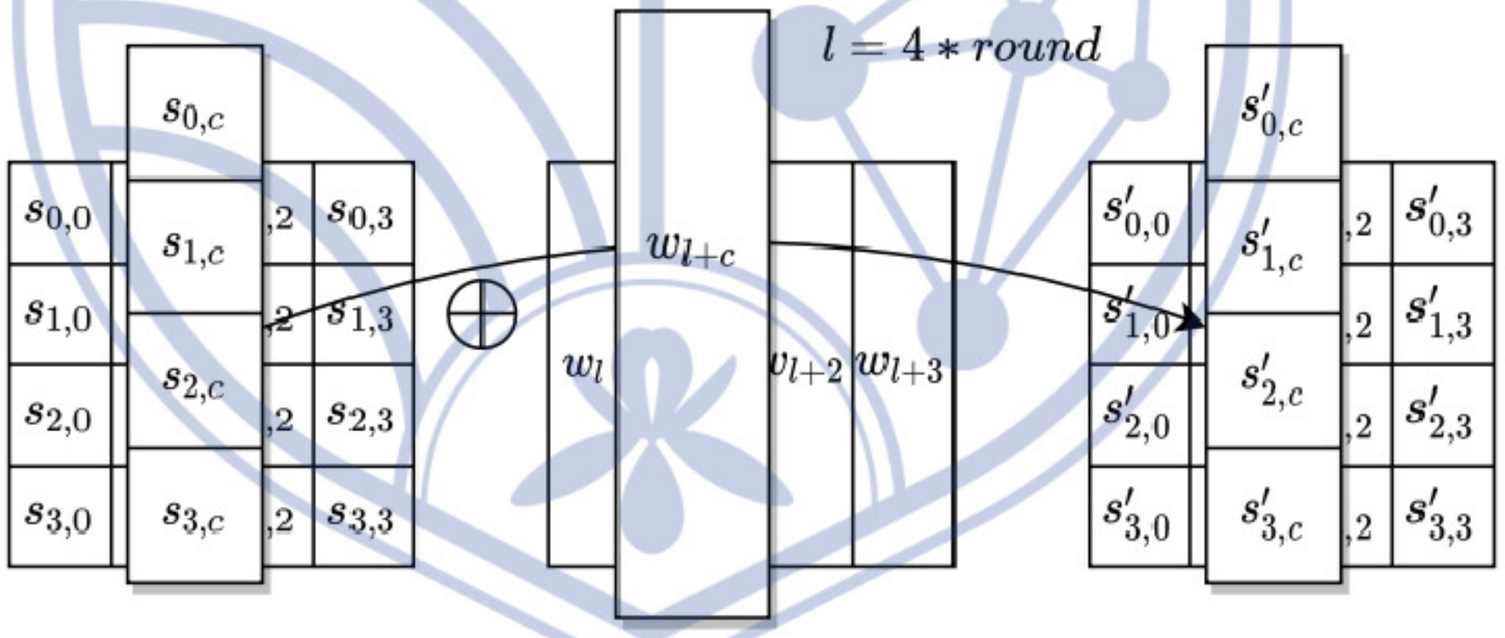
Gambar 2. 6 Ilustrasi MixColumns [13]

4. AddRound*Key*

AddRound*Key* merupakan transformasi dimana kunci putaran (round *key*) digabungkan dengan state melalui operasi XOR bitwise. Setiap kunci putaran terdiri dari empat kata (word) yang berasal dari penjadwalan kunci (*key* schedule). Empat kata ini secara berurutan dikombinasikan (XOR) dengan empat kolom state. Operasi ini dapat didefinisikan sebagai:

$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w(4 \cdot \text{round} + c)]$ untuk $0 \leq c <$..(15)

dimana round berada dalam rentang $0 \leq \text{round} \leq N_r$, dan $w[i]$ adalah *key* schedule words.



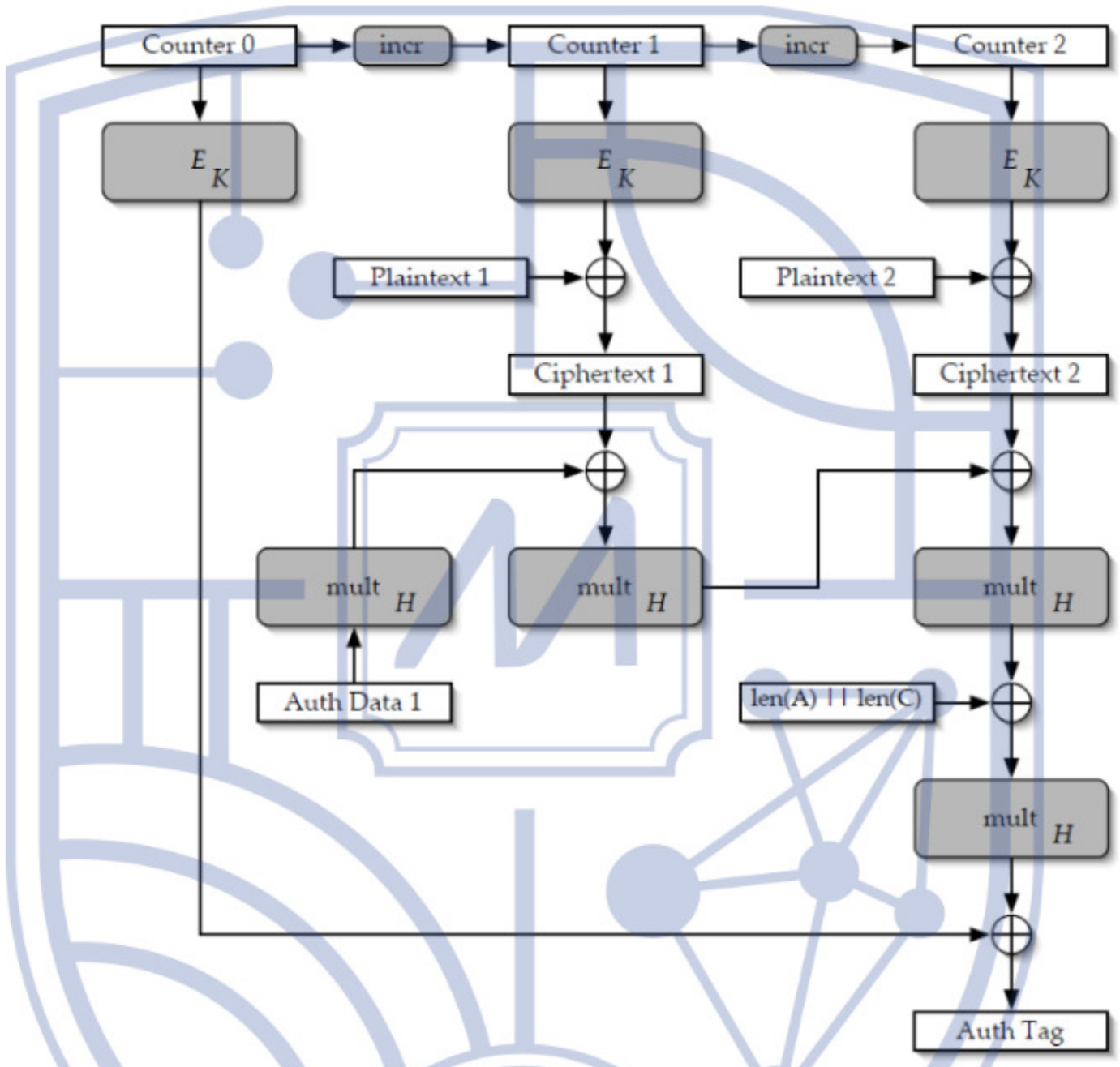
Gambar 2. 7 Ilustrasi AddRound*Key* [13]

2.3 Galois/*Counter* Mode (GCM)

Galois/*Counter* Mode (GCM) adalah mode operasi block cipher yang dirancang untuk menyediakan layanan keamanan ganda berupa kerahasiaan data (confidentiality) dan keaslian data (authenticity) secara efisien [4]. Menurut McGrew dan Viega (2005), GCM menggabungkan variasi dari mode *Counter* (CTR) untuk proses enkripsi dengan mekanisme

hashing universal di atas bidang Galois biner GF (2128) untuk proses autentikasi. Keunggulan utama GCM terletak pada kemampuannya untuk melakukan pemrosesan secara paralel, sehingga mampu mendukung kecepatan jaringan tinggi dengan latensi yang sangat rendah [14].

Alur kerja GCM secara visual dapat direpresentasikan melalui diagram blok enkripsi yang menunjukkan bagaimana *Counter* dienkripsi dan di-XOR dengan pesan asli, sementara fungsi perkalian bidang digunakan untuk menghasilkan tag pengaman.



Gambar 2. 8 Diagram blok enkripsi terauthentikasi GCM [14].

Inti dari proses autentikasi pada GCM adalah fungsi GHASH. Fungsi ini bekerja dengan mengalikan data dengan sebuah hash sub*key* (H) di dalam bidang matematika Galois. Persamaan dasar untuk fungsi GHASH adalah sebagai berikut:

$$X_i = (X_{i-1 \oplus Pi}) \cdot H \quad \text{.....(16)}$$

Dimana:

- X_i: Hasil komputasi pada blok ke i
- P_i: Blok data (baik *Ciphertext* maupun data tambahan)
- H: Hash sub*key* yang dihasilkan dari enkripsi blok nol menggunakan kunci AES
- $(\cdot)$: Operasi perkalian dalam bidang GF (2^{128}) [14]

2.4 AES 256 GCM

AES-256-GCM merupakan penerapan algoritma AES dengan panjang kunci 256-bit sebagai fungsi dasar dalam mode GCM. Mekanisme ini berfungsi sebagai Authenticated Encryption with Associated Data (AEAD), sebuah istilah yang merujuk pada kemampuan algoritma untuk menjamin kerahasiaan (confidentiality) pesan sekaligus memastikan keaslian (authenticity) data tambahan atau Additional Authenticated Data (AAD). Data tambahan ini merupakan informasi yang tidak dienkripsi namun harus diautentikasi agar tidak dapat dimanipulasi oleh pihak ketiga [15].

Secara teknis, proses AES-256-GCM menghasilkan dua output utama, yaitu *Ciphertext* dan *Authentication tag*. *Authentication tag* berperan sebagai segel digital, apabila terjadi perubahan data pada media penyimpanan, seperti pada bit-bit Least Significant Bit (LSB) dalam media steganografi, maka nilai tag tersebut tidak akan valid saat proses dekripsi dilakukan. Hal ini menjamin bahwa integritas data tetap terjaga karena sistem akan mendeteksi setiap upaya modifikasi sekecil apapun [4].

Algoritma ini menggunakan kunci rahasia sepanjang 256-bit yang menawarkan tingkat keamanan tinggi terhadap serangan brute-force. Selain itu, digunakan Initialization Vector (IV) sepanjang 96-bit yang bersifat unik untuk setiap sesi enkripsi guna menjaga keamanan mode *Counter* di dalamnya. Seluruh proses pengamanan ini dilakukan secara efisien melalui operasi pada bidang matematika GF (2^{128})

1. Algoritma Enkripsi AES-256-GCM

Proses enkripsi pada AES-256-GCM bertujuan untuk menghasilkan pesan acak (*Ciphertext*) sekaligus tag autentikasi. Langkah-langkah sistematis dalam proses enkripsi adalah sebagai berikut [15]:

1. Pembentukan Hash Sub*key* (H)

   Hash sub*key* (H) dibangkitkan dengan mengenkripsi blok nol 128-bit menggunakan kunci AES (K), yang dinyatakan secara matematis sebagai:

   $H = E(K, 0^{128})$ .....(17)

   Dimana E melambangkan fungsi enkripsi AES. Nilai ini digunakan dalam perhitungan autentikasi GHASH [14] [15].

2. Inisialisasi *Counter*Awal (Y_0)

Blok *Counter* awal (Y_0) dibentuk dari Initialization Vector (IV). Jika IV berukuran 96-bit, maka Y_0 diperoleh dengan menggabungkan IV dengan konstanta 32-bit bernilai satu sesuai spesifikasi GCM [14] [15].

3. Enkripsi Data (GCTR)

   Enkripsi data dilakukan menggunakan mekanisme *Counter* mode, yang dalam standar NIST disebut sebagai fungsi GCTR. Setiap blok plaintext dienkripsi dengan operasi XOR terhadap hasil enkripsi AES pada nilai *Counter* yang berurutan, sehingga dihasilkan *Ciphertext* (C) [14] [15].

4. Perhitungan dan Finalisasi *Authentication tag*

   Nilai autentikasi dihitung menggunakan fungsi GHASH dengan memproses *Additional Authenticated Data* (AAD) dan *Ciphertext* secara iteratif pada medan Galois GF(2^{128}). Nilai akhir GHASH kemudian di-*XOR* dengan hasil enkripsi blok *Counter* awal (Y_0) untuk menghasilkan *authentication tag* sepanjang 128-bit [14].

2. Algoritma Dekripsi AES-256-GCM

Proses dekripsi pada AES-256-GCM diawali dengan verifikasi *authentication tag* untuk memastikan keutuhan data sebelum plaintext dikembalikan [15]. Langkah-langkah teknis proses dekripsi adalah sebagai berikut [14]:

1. Pembentukan Hash Sub*key* (H)

   Hash sub*key* (H) dibangkitkan kembali dengan mengenkripsi blok nol 128-bit menggunakan kunci yang sama, dan digunakan dalam perhitungan GHASH [14].

2. Perhitungan dan Verifikasi Tag

   Sistem menghitung ulang *authentication tag* menggunakan GHASH terhadap AAD dan *Ciphertext*. Tag hasil perhitungan dibandingkan dengan tag yang diterima. Jika tidak sesuai, sistem mengembalikan status *FAIL* dan proses dihentikan [14].

3. Dekripsi *Ciphertext* (*Counter* Mode)

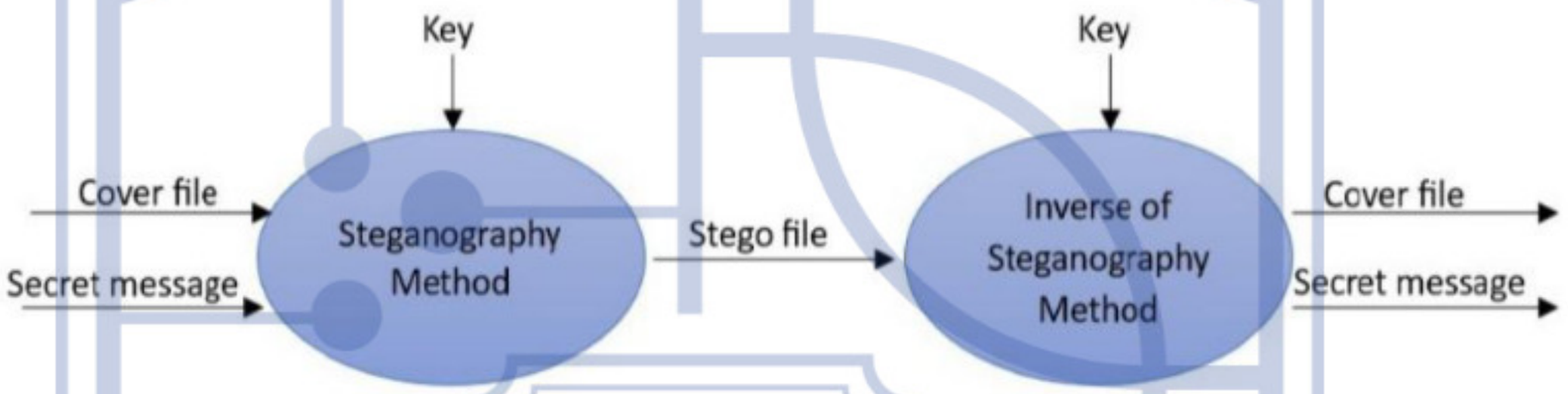
   Jika verifikasi berhasil, *Ciphertext* didekripsi menggunakan mekanisme *Counter mode* yang sama seperti pada proses enkripsi, sehingga diperoleh kembali plaintext [14].

4. Hasil Akhir (Output)

   Plaintext hanya dihasilkan apabila seluruh proses verifikasi autentikasi berhasil, sehingga keutuhan dan keaslian data tetap terjamin [14].

2.5 Steganografi

Steganografi berasal dari bahasa Yunani yaitu *steganos* yang berarti "tersembunyi" dan *graphien* yang berarti "menulis". Steganografi didefinisikan sebagai seni dan ilmu menyembunyikan pesan rahasia ke dalam suatu media penutup sehingga keberadaan pesan tersebut tidak terdeteksi. Media yang umum digunakan dalam steganografi meliputi teks, citra, audio, dan video, yang memungkinkan penyampaian informasi secara tersembunyi tanpa menimbulkan kecurigaan [16].



Gambar 2. 9 Alur steganografi

Berdasarkan gambar di atas, berikut penjelasan dari masing-masing komponen:

1. *Secret Message*: merupakan pesan atau informasi rahasia yang akan di sembunyikan. Dalam konteks citra, sebuah fungsi spesifik mendedikasikan vektor warna c(x,y) untuk setiap piksel (x, y) pada citra pesan.
2. *Cover File*: media utama yang digunakan untuk mengirimkan pesan tersembunyi. *Cover* biasanya dipilih sedemikian rupa agar terlihat biasa saja dan tidak signifikan, sehingga tidak memancing perhatian pihak luar.
3. *Stego File*: hasil dari proses penyisipan pesan rahasia ke dalam citra *cover*. Citra *stego* inilah yang nantinya dikirim dan digunakan untuk mengekstraksi kembali pesan tersembunyi di sisi penerima.
4. *Stego Key*: kode atau kunci yang mengizinkan informasi untuk disisipkan ke dalam *cover* dan diekstraksi dari media stego. Kunci ini dapat berupa angka yang dihasilkan melalui *pseudo-random* number atau bentuk lain untuk mengenkripsi posisi penyisipan pesan.
5. *Embedding Domain*: fitur dari media *cover* yang dimanfaatkan untuk menyisipkan pesan. Domain ini dapat berupa domain spasial jika bagian penyusun *cover* (seperti piksel) dimodifikasi secara langsung, atau berupa domain frekuensi (*transform*) jika

dilakukan manipulasi matematis pada media sebelum proses penyisipan dilakukan [16].

Secara lebih spesifik, implementasi steganografi citra menggunakan tiga pendekatan tradisional utama sebagai berikut:

1. Domain Spasial: Pendekatan ini menyisipkan pesan secara langsung ke dalam nilai piksel citra, seperti pada algoritma *Least Significant Bit* (LSB) dan *Pixel Value Differencing* (PVD), yang menawarkan kapasitas penyimpanan besar namun memiliki ketahanan rendah.
2. Domain Transformasi: Metode ini memanfaatkan manipulasi matematis pada koefisien frekuensi untuk menyisipkan data melalui algoritma seperti *Discrete Cosine Transform* (DCT) dan *Discrete Wavelet Transform* (DWT) guna memberikan keamanan yang lebih kuat.
3. Domain Adaptif: Dikenal sebagai metode berbasis model, pendekatan ini menggunakan teknik dinamis seperti algoritma genetika atau *machine learning* untuk menentukan lokasi penyisipan terbaik agar dapat menyeimbangkan aspek keamanan dan kapasitas secara optimal [17].

Steganografi berbeda dengan kriptografi karena kriptografi berfokus pada penyandian isi pesan agar tidak dapat dipahami oleh pihak yang tidak berwenang, sedangkan steganografi bertujuan menyembunyikan keberadaan komunikasi itu sendiri. Dalam steganografi, pesan rahasia disisipkan ke dalam media penutup yang tampak normal sehingga komunikasi tidak terlihat mencurigakan, sementara kriptografi menghasilkan pesan yang tampak acak dan dapat menarik perhatian pihak ketiga.

Citra digital merupakan media yang paling banyak digunakan dalam steganografi karena memiliki redundansi data yang tinggi serta toleransi visual manusia terhadap perubahan kecil pada nilai piksel. Perubahan intensitas piksel antara citra asli dan citra hasil penyisipan umumnya bersifat sangat kecil sehingga tidak terdeteksi oleh sistem visual manusia [17].

Karakteristik penting dari steganografi meliputi [17]:

1. Kapasitas Penyisipan (*Payload* Capacity): Jumlah data rahasia yang dapat disisipkan ke dalam media.
2. Kualitas Visual (Imperceptibility): Kualitas visual yang dihasilkan pada citra setelah penyisipan.

3. Ketahanan (Robustness): Kemampuan citra (stego-image) untuk mempertahankan pesan rahasia di dalamnya meskipun mengalami distorsi.
4. Keamanan (Security): Kemampuan untuk mencegah deteksi pesan rahasia oleh pihak ketiga atau steganalisis modern.

2.6 Least Significant Bit (LSB)

Least Significant Bit (LSB) merupakan salah satu teknik steganografi citra yang termasuk dalam pendekatan domain spasial, di mana pesan rahasia disisipkan secara langsung ke dalam nilai piksel citra penutup. Pendekatan domain spasial mencakup berbagai teknik penyisipan data, salah satunya adalah algoritma LSB *insertion* [17].

Metode LSB bekerja dengan cara menyisipkan bit pesan rahasia ke dalam bit paling tidak signifikan dari piksel citra. Teknik ini melibatkan penggantian bit pada lapisan bit terendah dari citra penutup dengan bit pesan rahasia yang akan disembunyikan. Karena perubahan hanya terjadi pada bit dengan bobot terendah, modifikasi tersebut bersifat minimal dan tidak menimbulkan perbedaan visual yang signifikan pada citra hasil penyisipan [16].

Gambar 2. 10 Perbandingan citra sebelum dan sesudah menggunakan metode LSB [5]

Metode LSB banyak digunakan dalam steganografi citra karena teknik domain spasial dilaporkan memiliki kualitas visual yang tinggi dengan distorsi minimal, serta kapasitas penyisipan yang relatif besar dibandingkan dengan pendekatan lainnya. Karakteristik ini menjadikan LSB sebagai salah satu metode yang populer dalam penelitian steganografi citra [16].

Namun demikian, teknik LSB memiliki keterbatasan pada aspek ketahanan (*robustness*) dan keamanan (*security*). Pendekatan domain spasial dilaporkan kurang robust

dan rentan terhadap berbagai bentuk manipulasi citra, seperti kompresi, filtering, dan penambahan noise. Selain itu, teknik steganalisis statistik, khususnya RS analysis, diketahui mampu mendeteksi keberadaan pesan rahasia yang disisipkan menggunakan metode LSB [17].

2.7 Bilangan Acak

Bilangan acak digunakan dalam berbagai bidang ilmu karena banyak aplikasi komputasi memerlukan pemilihan nilai secara acak. Penggunaan bilangan acak ini mencakup simulasi statistic matematis serta kriptografi dan keamanan sistem, sehingga bilangan acak menjadi komponen penting dalam perancangan dan analisis sistem komputasi modern [18].

Persyaratan terhadap bilangan acak berbeda-beda bergantung pada konteks penggunaannya. Secara umum, deret bilangan acak diharapkan memiliki sifat keacakan dan distribusi yang merata, yang biasanya dievaluasi melalui pengujian statistik. Namun, pada aplikasi keamanan, sifat tersebut belum mencukupi karena bilangan acak juga harus sulit diprediksi agar nilainya tidak dapat ditebak oleh pihak lain [18].

Selain digunakan pada aplikasi keamanan, kualitas bilangan acak juga memengaruhi keandalan hasil perhitungan dan simulasi berbasis probabilitas. Generator bilangan acak, baik true random maupun pseudo-random, dipandang sebagai komponen penting dalam berbagai bidang ilmiah dan teknis karena digunakan untuk mendukung proses perhitungan dan pemodelan berbasis acak. Oleh karena itu, kualitas statistik deret bilangan acak yang dihasilkan menjadi faktor penting dalam keandalan hasil komputasi [19].

Karena sistem komputer bekerja secara deterministik, pembangkitan bilangan acak melalui algoritma tidak dapat menghasilkan keacakan sejati. Oleh sebab itu, bilangan pseudo-acak dihasilkan melalui algoritma deterministik berbasis *seed* yang dirancang untuk menghasilkan deret bilangan yang menyerupai bilangan acak dan dapat digunakan secara luas dalam statistik maupun kriptografi. Dengan demikian, pemahaman mengenai konsep bilangan acak dan karakteristik pembangkitannya menjadi dasar teoretis yang penting dalam analisis dan perancangan sistem komputasi [18].

2.8 Ascon XOF128

Ascon XOF128 merupakan salah satu algoritma kriptografi ringan (*lightweight cryptography*) yang di standardisasi oleh *National Institute of Standard and Technology* (NIST). Algoritma ini dirancang sebagai *Extendable-Output Fucntion* (XOF), yaitu fungsi hash yang mampu menghasilkan keluaran panjang yang dapat ditentukan oleh pengguna. Dengan sifat ini, satu pesan masukan yang sama dapat menghasilkan keluaran dengan panjang yang berbeda-beda sesuai kebutuhan, berbeda dengan fungsi hash konvensional yang panjang keluarannya bersifat tetap (misalnya 256 bit).

Ascon XOF128 menerima parameter tambahan berupa panjang keluaran $L > 0$ dalam satuan bit. Parameter ini menentukan seberapa banyak bit keluaran yang dihasilkan pada fase akhir algoritma. Meskipun panjang keluaran dapat diperluas sesuai kebutuhan, tingkat keamanan Ascon XOF128 tidak bergantung pada panjang keluaran tersebut, melainkan pada desain internal algoritma.

Angka 128 pada nama Ascon XOF128 tidak menunjukkan panjang keluaran, tetapi menunjukkan tingkat keamanan yang dicapai oleh algoritma ini. NIST menetapkan bahawa Ascon XOF128 memiliki rate sebesar 64 bit dan capacity sebesar 256 bit. Berdasarkan teori konstruksi sponge kapasitas sebesar 256 but memberikan tingkat keamanan maksimum sebesar 128 bit terhadap seangan kriptografis seperti collision attack dan preimage attack.

Ascon XOF 128 dibangun di atas algoritma inti Ascon, yang menggunakan sebuah permutasi bernama Ascon-p. Permutasi ini bekerja pada sebuah state internal berukuran 320 bit, yang direpresentasikan sebagai lima buah word 64-bit sebagai berikut [8]:

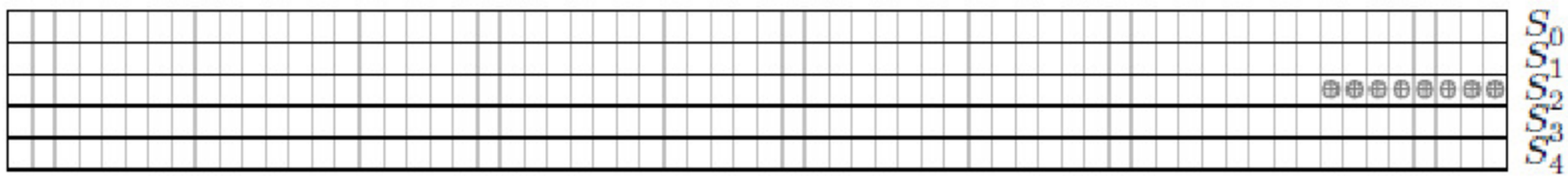
$$S = S_0 \,||\, S_1 \,||S_2 \,||\, S_3 \,||S_4 \quad \text{.....(18)}$$

Setiap word S_1 menyimpan 64 bit, sehingga keseluruhan state terdiri dari 320 bit. Permutasi Ascon-p memiliki struktur *Substitution–Permutation Network* (SPN). Dalam setiap rondenya, permutasi ini terdiri dari tiga lapisan utama yang diterapkan secara berurutan untuk mentransformasikan state. Ketiga lapisan tersebut dijelaskan sebagai berikut.

1. Constant Addition Layer

   Pada lapisan ini, sebuah konstanta ronde (*round constant*) di XOR kan ke word S_2sesuai dengan definisi permutasi Ascon-p [8].

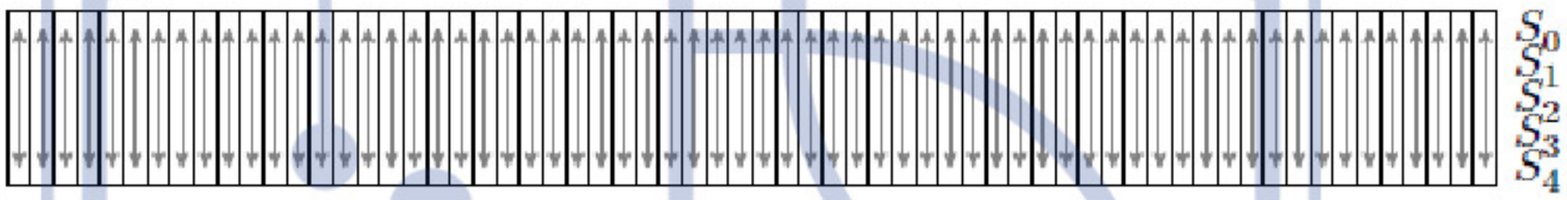
   $$S_{2=} S_2 \oplus C_i \quad \text{.....(19)}$$



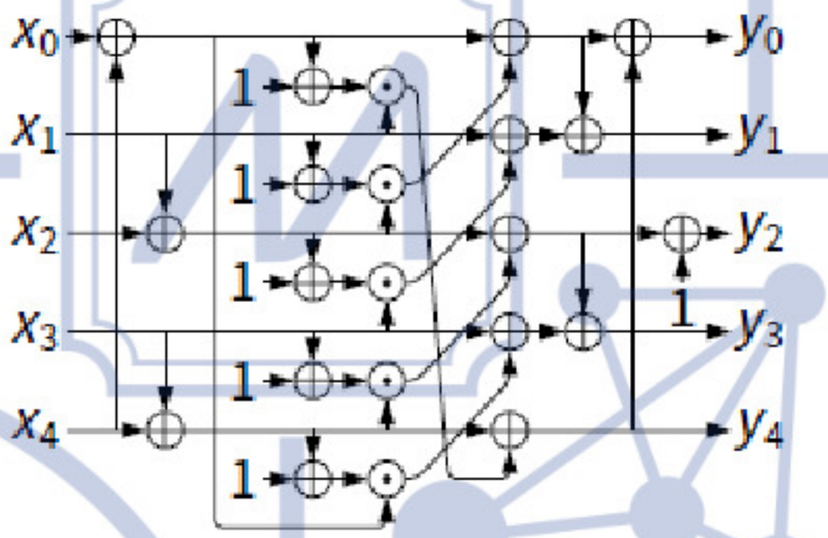
Gambar 2. 11 Constant Addition Layer [8]

2. Subtitution Layer

Substitution layer merupakan lapisan nonlinier utama pada permutasi Ascon-p. Lapisan ini bekerja dengan menerapkan S-box 5-bit secara paralel pada state Ascon.



Gambar 2. 12 Application of substitution layer ps to Ascon State [8].



Gambar 2. 13 Circuit representation of the 5-bit S-box SBox [8]

Gambar di atas memperlihatkan bagaimana *word state* S_0 hingga S_4diproses secara bersamaan. Untuk setiap posisis bit j dengan $0 \leq j < 64$, lima bit $(x_0, x_1, x_2, x_3, x_4)$ dipetakan menjadi lima bit keluaran baru menggunakan fungsi S-box. Proses ini dilakukan 64 kali secara paralel, satu kali untuk setiap posisi bit pada *state.*

Secara matematis, pemetaan S-box Ascon didefinisikan menggunakan fungsi Boolean nonlinear sebagai berikut [8]:

$$y_0 = x_4x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1x_0 \oplus x_1 \oplus x_0 \quad \text{.....(20)}$$

$$y_1 = x_4 \oplus x_3x_2 \oplus x_3x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1 \oplus x_0 \quad \text{.....(21)}$$

$$y_2 = x_4x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1 \quad \text{.....(22)}$$

$$y_3 = x_4x_0 \oplus x_4 \oplus x_3x_0 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0 \quad \text{.....(23)}$$

$$y_4 = x_4x_1 \oplus x_4 \oplus x_3 \oplus x_1x_0 \oplus x_1 \quad \text{.....(24)}$$

Selain menggunakan persamaan Boolean, S-box Ascon juga dapat di representasikan dalam bentuk loopup table debagai berikut [8]:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
SBox(x)	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c
x	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
SBox(x)	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

Note that 5-bit inputs are represented in hexadecimal (e.g., $x = 1$ corresponds to $(0,0,0,0,1)$).

Gambar 2. 14 Lookup table representation of SBox [8]

3. Linear Diffusion Layer

Linear diffusion layer merupakan lapisan terakhir pada permutasi Ascon-p. Pada lapisan ini, setiap word pada state diproses menggunakan operasi rotasi bit dan XOR [8].

Gambar 2. 15 Application of linear diffusion layer pl to Ascon state [8].

Linear diffusion layer didefinisikan dengan transformasi berikut:

$$\Sigma_0(S_0) = S_0 \oplus (S_0 \ggg 19) \oplus (S_0 \ggg 28) \quad (25)$$

$$\Sigma_1(S_1) = S_1 \oplus (S_1 \ggg 61) \oplus (S_1 \ggg 39) \quad (26)$$

$$\Sigma_2(S_2) = S_2 \oplus (S_2 \ggg 1) \oplus (S_2 \ggg 6) \quad (27)$$

$$\Sigma_3(S_3) = S_3 \oplus (S_3 \ggg 10) \oplus (S_3 \ggg 17) \quad (28)$$

$$\Sigma_4(S_4) = S_4 \oplus (S_4 \ggg 7) \oplus (S_4 \ggg 41) \quad (29)$$

Algoritma Ascon XOF128 menggunakan permutasi tersebut dalam sebuah konstruksi sponge. Konstruksi ini terdiri dari tiga tahap utama yaitu:

1. *Initialization*

Pada tahap ini, *state* Ascon diinisialisasi menggunakan nilai *Initialization Vector* (IV) khusus untuk Ascon-XOF128, yaitu:

IV = 0x00000800000cc0003 .....(30)

Nilai IV tersebut digabungkan dengan 256 bit nol untuk membentuk state awal, kemudian diproses menggunakan permutasi Ascon-p[12] sebagai berikut [8]:

$$S \leftarrow Ascon - p[12](IV || 0^{256}) \quad (31)$$

2. *Absorbing*

Pada tahap absorbing, pesan masukkan M di partisi menjadi blok-blok berukuran 64bit sebagai berikut [8] :

$M_0, \dots, M_{n-1}, \widetilde{M_n} \leftarrow parse(M, 64)$ .....(32)

Blok terakhir $\widetilde{M_n}$, diproses menggunakan fungsi padding untuk membentuk blok penuh 64 bit [8].

$M_n \leftarrow pad(\widetilde{M_n}, 64)$ .....(33)

Setiap blok pesan M_i dengan $0 \leq i < n$ di XOR kan ke 64bit pertama state, kemudian state dipermutasi menggunakan Ascon-p [12]

$S_{[0:63]} \leftarrow S_{[0:63]} \oplus M_i$ .....(34)

$S \leftarrow Ascon - p[12](S)$ .....(35)

Setelah itu, blok terakhir M_n di XOR kan ke 64bit pertama state dan state dipermutasi kembali menggunakan Ascon-p [12][8].

3. *Squeezing Output*

Pada tahap squeezing, keluaran sepanjang L bit dihasilkan dari state Ascon. Banyaknya blok keluaran yang diambil ditentukan oleh:

$h = \left\lceil \frac{L}{64} \right\rceil$ .....(36)

State dipermutasi menggunakan Ascon-p [12], kemudian 64bit pertama state diambil sebagai blok keluaran. Proses permutasi dan pengambilan keluaran diulang hingga blok H_0 sampai H_{h-1} diperoleh. Blok terakhir H_h diambil dari 64bit pertama state tanpa dilakukan permutasi tambahan. Seluruh blok keluaran digabungkan, dan L bit pertama dari hasil penggabungan tersebut ditetapkan sebagai keluaran [8].

$S \leftarrow Ascon - p[12](S)$ .....(37)

$H_i \leftarrow S[0:63], \quad i = 0, \dots, h-1$ .....(38)

$S \leftarrow Ascon - p[12](S)$ .....(39)

$H_h \leftarrow S[0:63]$ .....(40)

$H \leftarrow (H_0 || \dots || H_h)[0:L-1]$ .....(41)

2.9 Citra

Citra dapat dipahami sebagai representasi atau gambaran dari suatu objek yang dibentuk melalui kombinasi titik, garis, bidang, dan warna. Citra merupakan kemiripan atau imitasi dari objek nyata, baik berupa objek fisik maupun manusia. Berdasarkan sifatnya, citra dibedakan menjadi citra analog dan citra digital.

Citra analog adalah citra yang bersifat kontinu, seperti tampilan pada monitor televisi, foto sinar-X, lukisan, atau hasil pencitraan medis. Citra jenis ini tidak dapat diproses secara langsung oleh komputer sehingga diperlukan proses konversi dari bentuk analog ke digital agar dapat diolah lebih lanjut.

Citra digital merupakan citra yang dapat diolah oleh komputer dan diperoleh melalui proses sampling dan kuantisasi. Sampling menentukan pembagian citra ke dalam baris dan kolom yang membentuk piksel, sedangkan kuantisasi menentukan nilai tingkat kecerahan atau warna berdasarkan jumlah bit yang digunakan. Secara matematis, citra digital dapat dinyatakan sebagai fungsi dua dimensi $f(x, y)$, dengan x dan y sebagai koordinat spasial serta nilai fungsi menyatakan intensitas atau tingkat keabuan pada titik tersebut [20].

1. Jenis Citra

Jenis citra digital ditentukan oleh cara penyimpanan nilai piksel di dalam memori. Berdasarkan karakteristik nilai tersebut, citra digital dapat dikelompokkan sebagai berikut:

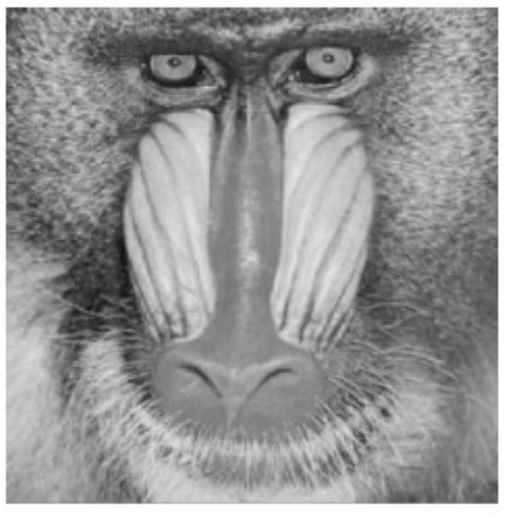
1. Citra Biner

Citra biner merupakan citra digital yang hanya memiliki dua kemungkinan nilai intensitas, yaitu hitam dan putih. Setiap piksel pada citra biner dapat direpresentasikan menggunakan satu bit, sehingga struktur datanya paling sederhana dibandingkan jenis citra lainnya [20].

Gambar 2. 16 Contoh Citra Biner [21]

2. Citra Keabuan

Citra keabuan memiliki tingkat intensitas yang bergantung pada jumlah bit yang digunakan untuk merepresentasikan setiap piksel. Sebagai contoh, citra dengan 2bit memiliki 4 tingkat keabuan, sedangkan citra dengan 3bit memiliki 8 tingkat keabuan. Semakin besar kedalaman bit yang digunakan, semakin halus gradasi keabuan yang dapat ditampilkan [20].



Gambar 2. 17 Contoh Citra Keabuan [2]

3. Citra Berwarna

Citra berwarna merepresentasikan setiap piksel sebagai kombinasi tiga warna dasar, yaitu merah, hijau, dan biru (RGB). Umumnya, setiap kanal warna direpresentasikan menggunakan 8 bit, sehingga satu piksel citra berwarna memiliki total 24bit data. Kombinasi tersebut memungkinkan citra berwarna merepresentasikan jutaan variasi warna [20].

Gambar 2. 18 Contoh Citra Berwarna [1]

2. Elemen Citra

Citra digital memiliki beberapa elemen utama yang menggambarkan karakteristik visual dari objek di dalam citra. Elemen-elemen tersebut antara lain sebagai berikut:

1. Kecerahan (*Brightness*)

   Kecerahan menyatakan tingkat intensitas cahaya yang direpresentasikan oleh suatu piksel dan dapat ditangkap oleh sistem penglihatan manusia. Nilai kecerahan pada sebuah piksel menggambarkan tingkat terang atau gelap pada area tertentu di dalam citra.

2. Kontras (*Contrast*)

   Kontras menunjukkan perbedaan distribusi antara area terang dan gelap dalam sebuah citra. Citra dengan kontras yang baik memiliki perbedaan intensitas yang jelas sehingga objek di dalam citra dapat dibedakan dengan lebih mudah.

3. Kontur (*Contour*)

Kontur muncul akibat adanya perubahan intensitas antara piksel-piksel yang saling berdekatan. Perubahan intensitas ini memungkinkan sistem visual mendeteksi batas atau tepi objek yang terdapat di dalam citra.

4. Warna (*Colour*)

Warna merupakan persepsi visual yang dihasilkan dari panjang gelombang cahaya yang dipantulkan oleh suatu objek. Informasi warna memberikan karakteristik tambahan pada citra selain tingkat kecerahan.

5. Bentuk (*Shape*)

Bentuk merupakan karakteristik intrinsik dari suatu objek yang berkaitan dengan struktur geometrinya. Informasi bentuk berperan penting dalam proses pengenalan dan identifikasi objek di dalam citra.

6. Tekstur (*Texture*)

Tekstur menggambarkan distribusi spasial tingkat keabuan atau intensitas pada sekelompok piksel yang berdekatan. Pola tekstur terbentuk dari keteraturan susunan piksel dan dapat digunakan untuk membedakan karakteristik permukaan objek, seperti kasar atau halus, tanpa bergantung pada informasi warna [20].

2.10 Mean Square Error (MSE)

Mean Square Error (MSE) merupakan metrik statistik yang digunakan untuk mengukur tingkat perbedaan antara citra asli (cover image) dan citra hasil penyisipan (stego-image) dengan menghitung rata-rata kuadrat selisih nilai intensitas piksel pada kedua citra tersebut. Nilai MSE diperoleh dengan menjumlahkan kuadrat selisih nilai piksel pada setiap posisi piksel dan membaginya dengan jumlah total piksel dalam citra. Metrik ini digunakan untuk merepresentasikan tingkat distorsi yang diakibatkan oleh proses penyisipan data ke dalam citra digital [22].

Dalam penelitian steganografi citra, MSE digunakan sebagai indikator kuantitatif untuk mengevaluasi kualitas citra stego terhadap citra aslinya. Nilai MSE yang rendah menunjukkan bahwa perbedaan antara citra asli dan citra stego relatif kecil, sehingga perubahan visual yang terjadi akibat proses penyisipan data bersifat minimal dan sulit diamati secara kasat mata. Oleh karena itu, MSE banyak digunakan dalam penelitian steganografi sebagai ukuran distorsi citra yang dihasilkan oleh suatu algoritma penyisipan pesan rahasia [17].

Selain digunakan sebagai ukuran distorsi, MSE juga berperan sebagai parameter dasar dalam perhitungan Peak Signal-to-Noise Ratio (PSNR). Hubungan ini menjadikan MSE sebagai salah satu metrik evaluasi fundamental dalam pengujian kinerja algoritma steganografi citra [17].

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad \text{.....(42)}$$

2.11 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) merupakan metrik evaluasi yang digunakan untuk mengukur kualitas citra stego dengan membandingkannya terhadap citra asli berdasarkan rasio antara nilai sinyal maksimum yang mungkin dengan tingkat kesalahan yang diwakili oleh Mean Square Error (MSE). Pada citra digital 8-bit, nilai sinyal maksimum yang digunakan dalam perhitungan PSNR umumnya bernilai 255 [22].

Dalam konteks steganografi citra, PSNR digunakan untuk mengukur tingkat ketidaktampakan (imperceptibility) perubahan visual yang dihasilkan oleh proses penyisipan pesan rahasia. Nilai PSNR yang tinggi menunjukkan bahwa citra stego memiliki tingkat kemiripan visual yang tinggi terhadap citra asli, sehingga distorsi yang terjadi akibat penyisipan data relatif kecil [17]. Secara umum, nilai PSNR sebesar 30 dB atau lebih dianggap hampir tidak dapat dibedakan oleh penglihatan manusia, sehingga kualitas visual citra hasil penyisipan masih dinilai baik [23].

PSNR juga digunakan secara luas sebagai metrik pembanding untuk mengevaluasi dan membandingkan kinerja berbagai algoritma steganografi. Suatu metode steganografi dinilai memiliki kinerja yang baik apabila mampu mempertahankan nilai PSNR yang tinggi meskipun kapasitas penyisipan data meningkat, yang menunjukkan bahwa kualitas visual citra tetap terjaga dengan baik [22].

$$PNSR = 20 \cdot log_{10}(MAX_i) - 10 \cdot log_{10}(MSE) \quad \text{.....(43)}$$

2.12 Structural Similarity Index Method (SSIM)

Structural Similarity Index Method (SSIM) merupakan metode evaluasi kualitas citra yang digunakan untuk mengukur tingkat kemiripan struktural antara dua citra, yaitu citra asli dan citra hasil penyisipan data. SSIM menilai kesamaan citra berdasarkan komponen luminance, contrast, dan structure, sehingga mampu merepresentasikan persepsi visual manusia dengan lebih baik dibandingkan metrik berbasis perbedaan nilai piksel semata [22].

Dalam steganografi citra, SSIM digunakan untuk mengevaluasi sejauh mana struktur visual citra tetap terjaga setelah proses penyisipan pesan rahasia. Nilai SSIM yang mendekati satu menunjukkan bahwa struktur citra stego sangat mirip dengan struktur citra asli, sehingga degradasi visual yang terjadi akibat penyisipan data dapat dianggap sangat kecil [22].

Penggunaan SSIM sebagai metrik evaluasi sering dikombinasikan dengan MSE dan PSNR untuk memberikan penilaian kualitas citra yang lebih komprehensif. Pendekatan ini digunakan secara luas dalam penelitian steganografi modern untuk memastikan bahwa algoritma yang diusulkan tidak hanya menghasilkan distorsi numerik yang rendah, tetapi juga mampu menjaga kesamaan struktur visual citra [17].

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \quad \text{.....(44)}$$

2.13 Chi-Square

Chi-Square merupakan metode uji statistik yang digunakan untuk menganalisis perbedaan antara distribusi frekuensi yang diamati dan distribusi frekuensi yang diharapkan. Dalam konteks steganografi dan steganalisis, uji Chi-Square digunakan untuk mendeteksi keberadaan pesan tersembunyi berdasarkan analisis distribusi statistik nilai piksel pada citra digital [17].

Analisis Chi-Square umumnya diterapkan pada teknik steganografi berbasis Least Significant Bit (LSB) untuk mengidentifikasi pola distribusi biner yang tidak alami akibat proses penyisipan pesan. Metode ini bekerja dengan membandingkan distribusi nilai piksel pada citra stego dengan distribusi alami citra untuk menentukan apakah terjadi manipulasi data yang signifikan [17].

Dalam pengujian ketahanan algoritma steganografi, uji Chi-Square digunakan sebagai metode steganalysis statistik untuk mengevaluasi adanya penyimpangan distribusi nilai piksel akibat proses penyisipan pesan rahasia. Pengujian dilakukan dengan menganalisis distribusi frekuensi pasangan nilai piksel atau intensitas citra stego, kemudian membandingkannya dengan distribusi statistik citra alami untuk mengidentifikasi perubahan histogram yang tidak wajar akibat manipulasi Least Significant Bit (LSB). Hasil analisis ini digunakan untuk menilai apakah algoritma steganografi mampu mempertahankan distribusi statistik citra agar tetap mendekati distribusi alaminya, sehingga memiliki ketahanan terhadap deteksi melalui serangan steganalysis berbasis Chi-Square [17], [22], [24].

$$X^2{}_{k-1} = \sum_{i-1}^{k} \frac{(n_i-n_i^*)}{(n_{i)}^*} \quad \text{.....(45)}$$

2.14 Lempel Ziv Welch (LZW)

Lempel–Ziv–Welch (LZW) adalah algoritma kompresi lossless yang digunakan untuk mengurangi ukuran data tanpa menghilangkan informasi asli [6]. Algoritma ini bekerja dengan cara mengenali pola atau rangkaian karakter yang muncul berulang di dalam data, kemudian menyimpan pola tersebut ke dalam sebuah kamus (dictionary) dan menggantikannya dengan kode numerik sehingga ukuran data menjadi lebih kecil [6],[7].

Pada algoritma LZW, kamus tidak disiapkan secara terpisah, tetapi dibangun secara bertahap selama proses kompresi dan dekompresi berlangsung [6]. Karena sifatnya yang lossless dan mekanismenya yang sederhana, algoritma LZW banyak digunakan untuk kompresi data teks dan juga diklasifikasikan sebagai algoritma kompresi yang dapat digunakan pada berbagai sistem yang membutuhkan data hasil dekompresi yang sama persis dengan data awal [7],[25] .

1. Proses algoritma LZW adalah sebagai berikut [26]:
2. Proses dimulai dengan membuat kamus awal yang berisi seluruh karakter ASCII dasar, di mana setiap karakter memiliki indeks tersendiri [6].
3. Data kemudian dibaca secara berurutan untuk membentuk rangkaian karakter sementara.
4. Setiap rangkaian karakter yang terbentuk dicek apakah sudah ada di dalam kamus.
5. Jika rangkaian karakter sudah ada di dalam kamus, maka pembacaan data dilanjutkan untuk membentuk rangkaian yang lebih Panjang.
6. Jika rangkaian karakter belum ada di dalam kamus, maka kode dari rangkaian sebelumnya dikeluarkan sebagai hasil kompresi dan rangkaian baru tersebut ditambahkan ke dalam kamus [7].
7. Langkah-langkah ini diulang hingga seluruh data selesai diproses.
8. Pada proses dekompresi, kode numerik yang dihasilkan diterjemahkan kembali menjadi rangkaian karakter dengan menggunakan kamus yang dibentuk selama proses kompresi [6] [25].