BAB 2
LANDASAN TEORI

2.1 Algoritma Rijndael (AES)

Pada tanggal 26 November 2001, *Federal Information Processing Standards Publication* 197 mengumumkan sebuah bentuk terstandarisasi dari algoritma Rijndael sebagai standar enkripsi baru. Standar ini dinamakan *Advanced Encryption Standard* (AES) [26]. AES merupakan algoritma *symmetric key cryptographic* (SKC) yang menggunakan kunci yang sama untuk enkripsi dan dekripsi [21]. Panjang kunci AES adalah 128 bit, 192 bit, dan 256 bit untuk enkripsi dan dekripsi, dan panjang blok yang sudah ditetapkan sebesar 128 bit [18, 27].

AES menggunakan jumlah putaran yang bervariasi, tergantung pada ukuran kunci/blok, seperti pada Tabel 2.1 [28].

Tabel 2.1. Jumlah Putaran Berdasarkan Ukuran (1 word = 32 bit)

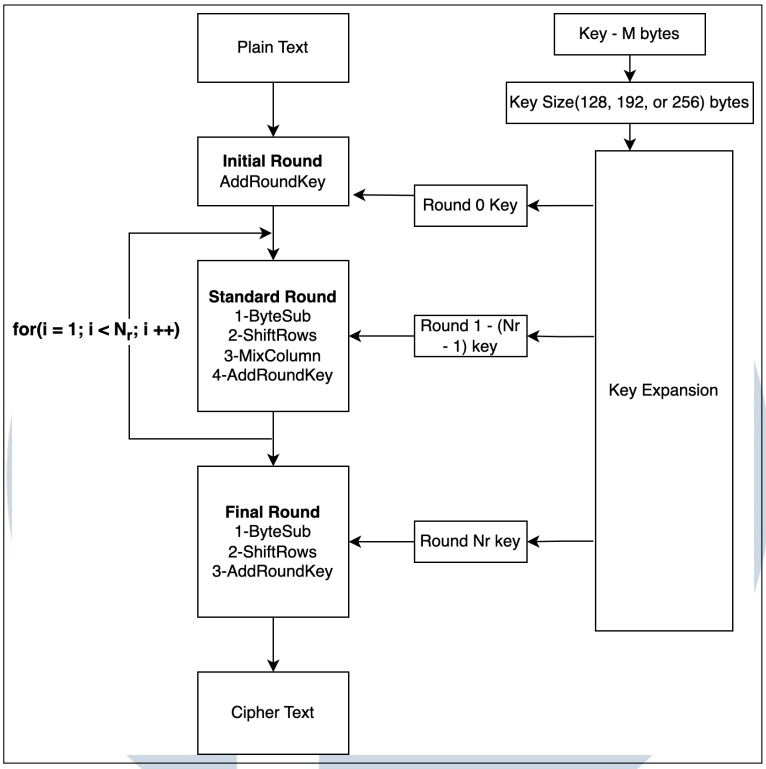
	Panjang Kunci *(Nk words)*	Panjang Blok *(Nb words)*	Jumlah Putaran *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

2.1.1 Enkripsi

Gambar 2.1 adalah algoritma dari proses enkripsi AES dan dilanjutkan dengan ilustrasi proses enkripsi AES pada Gambar 2.2 [29].

```
Rijndael(State,CipherKey)
{
KeyExpansion(CipherKey,ExpandedKey);
AddRoundKey(State,ExpandedKey[0]);
for(i = 1; i < Nr ; i++) Round(State,ExpandedKey[i]);
FinalRound(State,ExpandedKey[Nr]);
}
```

Gambar 2.1. Algoritma AES enkripsi



Gambar 2.2. Proses enkripsi algoritma AES

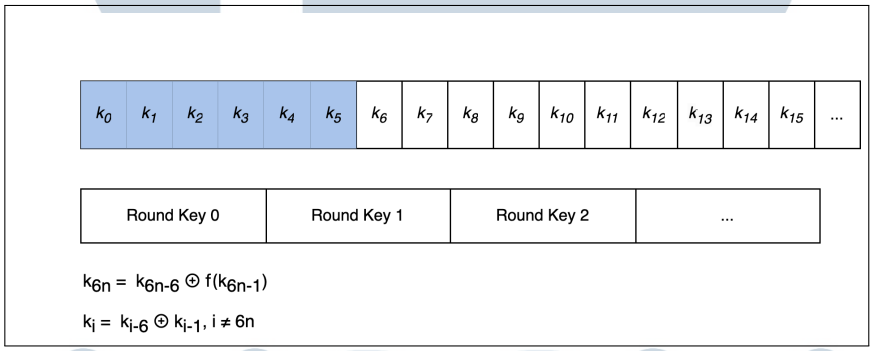
Proses enkripsi AES yang berada pada Gambar 2.2 diawali dengan pembuatan *round key* untuk setiap proses transformasi *AddRoundKey* dari *cipher key*. Algoritma dari pembuatan seluruh *round key* tersebut dinamakan *key schedule*. Gabungan dari seluruh *round key* dinamakan *expanded key* yang dilambangkan dengan *K*. *ExpandedKey* memiliki jumlah bit yang sama dengan panjang blok dikalikan dengan jumlah ronde (Nr) ditambah satu [29].

Algoritma dari proses *KeyExpansion* ditunjukkan pada Gambar 2.3 dan dilanjutkan dengan ilustrasi proses *key expansion* dan seleksi *round key* pada Gambar 2.4 [29].

UNIVERSITAS MULTIMEDIA NUSANTARA

```
KeyExpansion(byte K[4][Nk], byte W[4][Nb(Nr + 1)])
                                                    /* for Nk ≤ 6 */
{
for(j = 0; j < Nk; j++)
     for(i = 0; i < 4; i++) W[i][j] = K[i][j];
for(j = Nk; j < Nb(Nr + 1); j++)
     {
     if (j mod Nk == 0)
          {
          W[0][j] = W[0][j − Nk] ⊕ S[W[1][j − 1]] ⊕ RC[j/Nk];
          for(i = 1; i < 4; i++)
               W[i][j] = W[i][j − Nk] ⊕ S[W[i + 1 mod 4][j − 1]];
          }
     else
          {
          for(i = 0; i < 4; i++)
               W[i][j] = W[i][j − Nk] ⊕ W[i][j − 1];
          }
     }
}
```

Gambar 2.3. Algoritma fungsi *key expansion* pada enkripsi AES



Gambar 2.4. Proses *key expansion* dan pemilihan *round key* dengan contoh $N_b = 4$ dan $N_k = 6$

Setelah proses *KeyExpansion* sudah selesai, proses enkripsi dilanjutkan dengan empat jenis transformasi *byte*, yaitu *ByteSub*, *ShiftRows*, *MixColumn*, dan *AddRoundKey*. *Input* yang telah disalin ke dalam *state* akan mengalami transformasi *byte* bernama *AddRoundKey*. Setelah itu, *state* akan menjalankan transformasi *ByteSub*, *ShiftRows*, *MixColumn*, dan *AddRoundKey* sebanyak $(i = 1; i < N_r; i++)$ Putaran. Proses tersebut dinamakan *Standard Round*. Untuk proses putaran terakhir atau *Final Round* sedikit berbeda dari putaran sebelumnya karena hanya berisikan transformasi *ByteSub*, *ShiftRows* dan *AddRoundKey*, yang berarti *state* tidak mengalami transformasi *MixColumn* [27]. Gambar 2.5 merupakan algoritma enkripsi dari proses *Round* dan *FinalRound* [29].

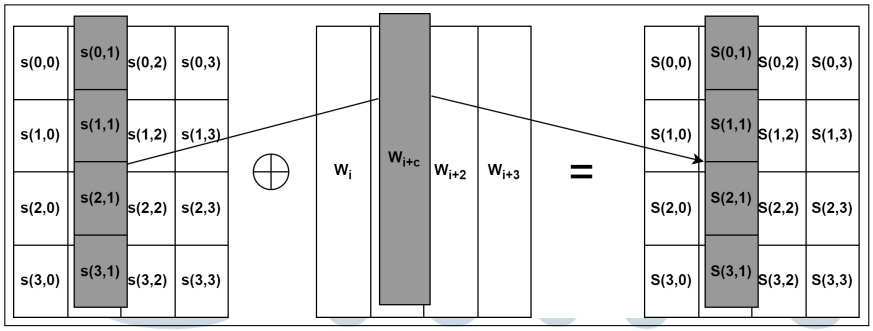
```
Round(State,ExpandedKey[i])
{
SubBytes(State);
ShiftRows(State);
MixColumns(State);
AddRoundKey(State,ExpandedKey[i]);
}


FinalRound(State,ExpandedKey[Nr])
{
SubBytes(State);
ShiftRows(State);
AddRoundKey(State,ExpandedKey[Nr]);
}
```

Gambar 2.5. Algoritma fungsi *Round* dan *FinalRound* pada enkripsi AES

Berikut adalah penjelasan dari keempat transformasi tersebut:

1. *AddRoundKey*

   Transformasi *AddRoundKey* merupakan operasi XOR sederhana antara *state* dan *Roundkey* [28]. Proses Transformasi *AddRoundKey* diilustrasikan pada Gambar 2.6 [28].

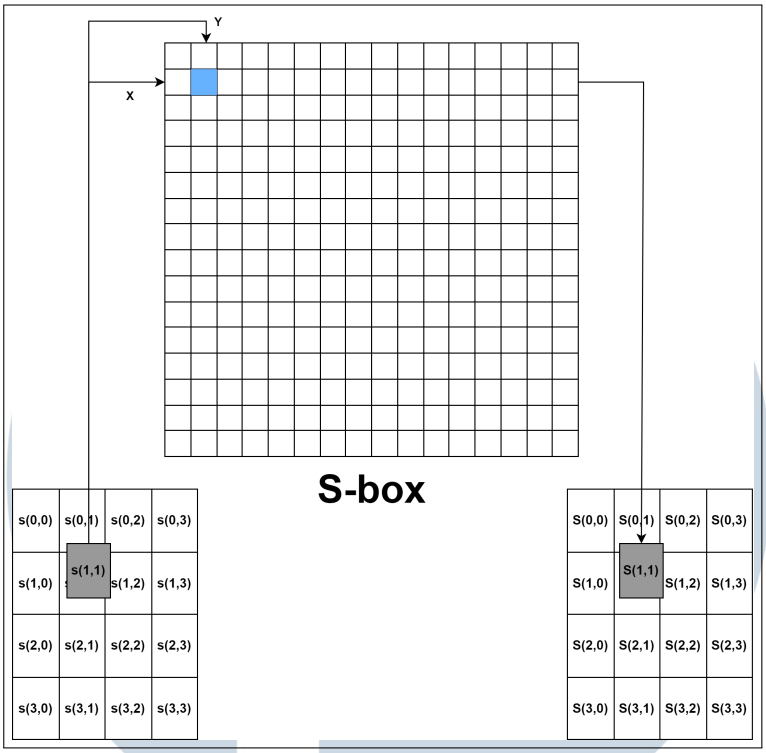


Gambar 2.6. Proses transformasi *AddRoundKey*

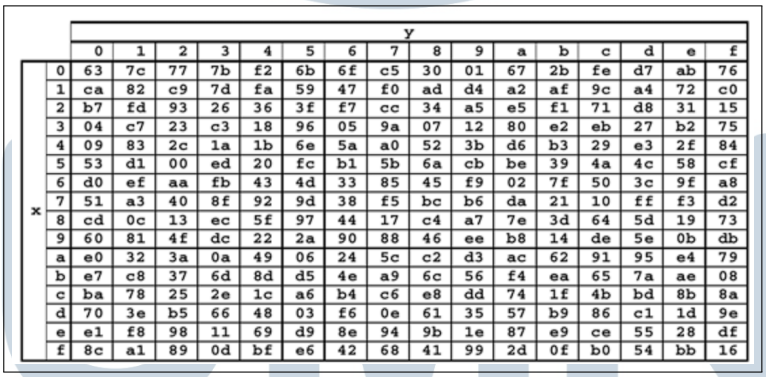
2. *ByteSub*

   Transformasi *ByteSub* merupakan transformasi pada *state* dengan melakukan pergantian nilai *state* dengan tabel S-box seperti pada Gambar 2.7 [19]. Tabel S-box dapat dilihat pada Gambar 2.8 [28].

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 2.7. Proses transformasi *ByteSub*



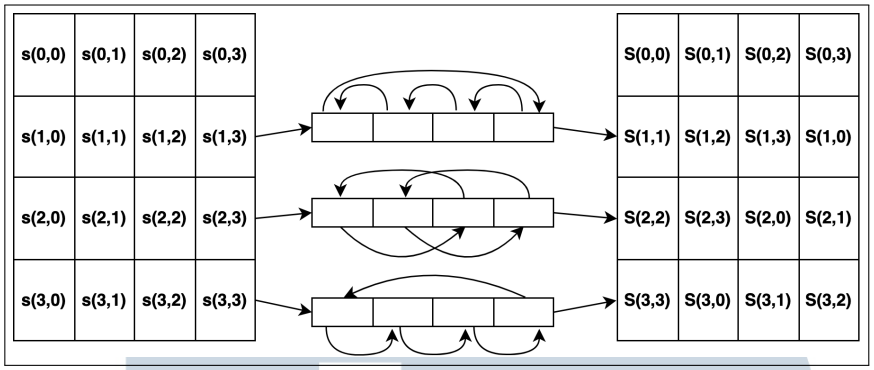
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	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

Gambar 2.8. Tabel S-box

3. *ShiftRows*

   Transformasi *ShiftRows* menggeser baris-baris *state* secara konsisten dengan besaran tertentu.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 2.9. Proses Transformasi *ShiftRows*

Berdasarkan ilustrasi pada Gambar 2.9 [17], tidak ada pergeseran di baris satu, terjadi pergeseran ke kiri sebanyak *c1 byte* pada baris kedua, *c2 byte* pada baris ketiga, dan *c3 byte* pada baris keempat [28]. Nilai dari *c1, c2,* dan *c3* bergantung dari panjang blok (*Nb*) sesuai dengan yang ada pada Tabel 2.2 [28].

Tabel 2.2. Nilai c1, c2, dan c3 berdasarkan panjang blok (*Nb*)

Nb	**c1**	**c2**	**c3**
4	1	2	3
6	1	2	3
8	1	3	4

4. *MixColumn*

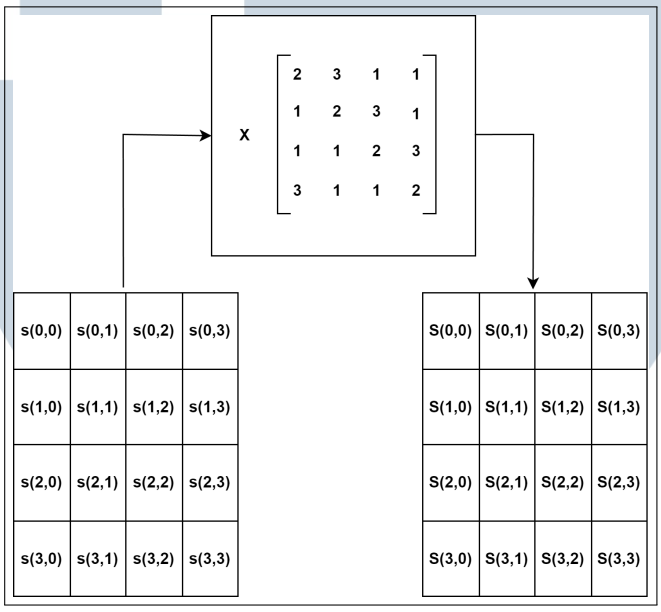
Tahap transformasi *MixColumn* merupakan permutasi *bricklayer* yang diterapkan pada kolom per kolom dari sebuah state. Kolom-kolom *state* tersebut dianggap sebagai polinomial di atas $GF(2^8)$ dan dikalikan modulo x^4+1 dengan polinomial tetap $c(x)$ sebagai berikut.

$$c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02. \tag{2.1}$$

Polinomial tetap ini dapat dituliskan sebagai perkalian matriks dengan persamaan $S(x) = c(x) \,.\, s(x) \pmod{x^4+1}$.

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{2.2}$$

Proses dari transformasi *MixColumn* diilustrasikan pada Gambar 2.10 [29].



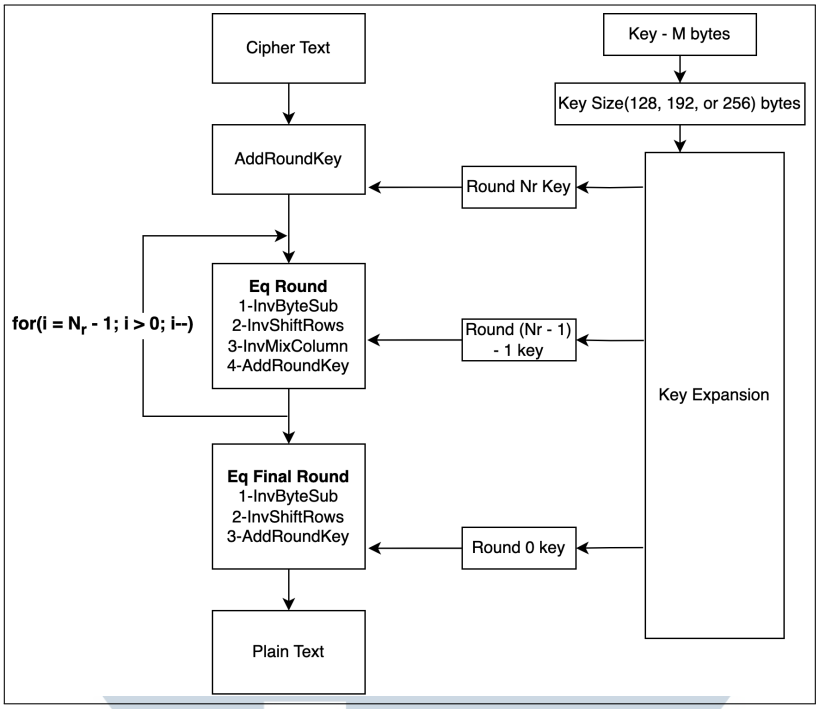
Gambar 2.10. Proses Transformasi *MixColumn*

2.1.2 Dekripsi

Gambar 2.11 adalah algoritma dari proses dekripsi AES dan dilanjutkan dengan ilustrasi proses dekripsi AES pada Gambar 2.12 [29].

```
InvRijndael(State,CipherKey)
{
EqKeyExpansion(CipherKey,EqExpandedKey);
AddRoundKey(State,EqExpandedKey[Nr]);
for(i = Nr − 1; i > 0; i−−) EqRound(State,EqExpandedKey[i]);
EqFinalRound(State,EqExpandedKey[0]);
}
```

Gambar 2.11. Algoritma AES dekripsi



Gambar 2.12. Proses dekripsi algoritma AES

Proses dekripsi AES juga diawali dengan pembuatan *round key* untuk setiap proses transformasi *AddRoundKey* dari *cipher key*. Seluruh *round key* untuk dekripsi tersebut dibuat pada proses *EqKeyExpansion*. Algoritma dari proses *EqKeyExpansion* ditunjukkan pada Gambar 2.13 [29].

```
EqKeyExpansion(CipherKey,EqExpandedKey)
{
KeyExpansion(CipherKey,EqExpandedKey);
for(i = 1; i < Nr ; i++)
     InvMixColumns(EqExpandedKey[i]);
}
```

Gambar 2.13. Fungsi *key expansion* pada algoritma AES dekripsi

Transformasi *byte* yang digunakan dalam proses dekripsi merupakan kebalikan dari transformasi yang digunakan saat enkripsi, yaitu *InvShiftRows*, *InvByteSub*, *InvMixColumns*, dan *AddRoundKey* [28, 29]. Gambar 2.14 merupakan algoritma dekripsi dari proses *EqRound* dan *EqFinalRound* [29].

UNIVERSITAS MULTIMEDIA NUSANTARA

```
EqRound(State,EqExpandedKey[i])
{
InvSubBytes(State);
InvShiftRows(State);
InvMixColumns(State);
AddRoundKey(State,EqExpandedKey[i]);
}


EqFinalRound(State,EqExpandedKey[0])
{
InvSubBytes(State);
InvShiftRows(State);
AddRoundKey(State,EqExpandedKey[0]);
}
```

Gambar 2.14. Fungsi *Round* dan *FinalRound* pada algoritma AES dekripsi

1. *InvByteSub*

   Transformasi *InvByteSub* merupakan transformasi *state* yang serupa dengan transformasi *ByteSub*, hanya saja pergantian nilai state nya dengan tabel *inverse* tabel S-box [28]. Tabel *inverse* S-box dapat dilihat pada Gambar 2.15 [28].

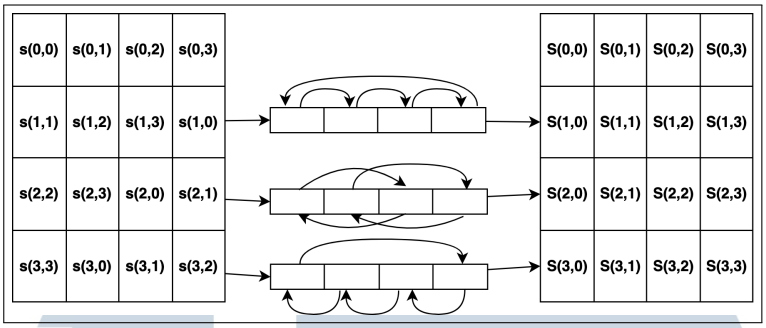
	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5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

x

Gambar 2.15. Tabel *Inverse* S-box

2. *InvShiftRows*

   Transformasi *InvShiftRows* merupakan transformasi yang serupa dengan *ShiftRows*, akan tetapi pergeseran yang ada pada *InvShiftRows* bukan kearah kiri seperti pada *ShiftRows*, melainkan kearah kanan [29]. Ilustrasi dari proses transformasi *InvShiftRows* ada pada Gambar 2.16 [28].



Gambar 2.16. Proses Transformasi *InvShiftRows*

3. *InvMixColumn*

   Transformasi *InvMixColumn* adalah transformasi yang serupa dengan *MixColumn* karena merupakan operasi *inverse* dari *MixColumn*. Setiap kolom pada operasi *InvMixColumn* diubah dengan melakukan perkalian dari kolom tersebut dengan polinomial tetap $d(x)$ sebagai berikut [29].

$$d(x) = 0B \cdot x^3 + 0D \cdot x^2 + 09 \cdot x + 0E. \tag{2.3}$$

   Apabila ditulis dalam perkalian matriks, *InvMixColumn* mengubah kolom dengan cara berikut [29].

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{2.4}$$

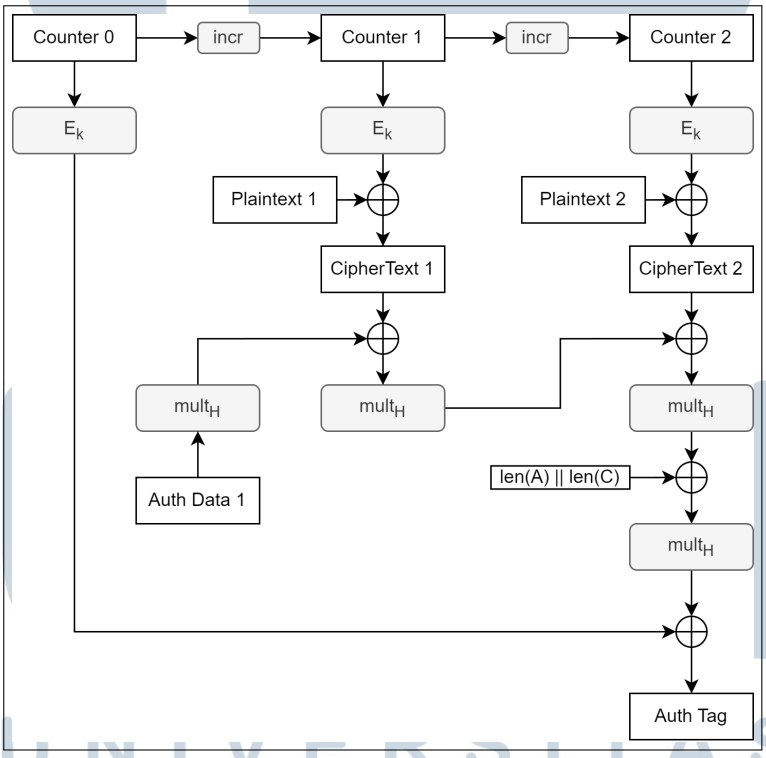
2.2 Galois/Counter Mode (GCM)

Galois/Counter Mode (GCM) merupakan sebuah mode operasi untuk blok cipher yang menggunakan *hashing* universal pada medan Galois biner untuk menyediakan enkripsi terautentikasi. Dua fungsi utama dari GCM adalah enkripsi terautentikasi dan dekripsi terautentikasi. Fungsi enkripsi terautentikasi mengenkripsi data yang bersifat rahasia dan menghitung *tag* autentikasi pada kedua data rahasia dan data tambahan jika ada. Fungsi dekripsi terautentikasi mendekripsi data rahasia, tergantung pada verifikasi *tag* tersebut. Jika *tag* tidak cocok, maka dekripsi gagal.

Operasi enkripsi terautentikasi memiliki empat *input* yang masing-masing berupa *string* bit , yaitu *Secret Key (K), Initialization Vector (IV), PlainText (P),* dan *Additional Authenticated Data (AAD)*, dengan dua output, yaitu *Ciphertext (C)* dan *Authentication tag (T)* [23].

Operasi dekripsi terautentikasi memiliki lima *input*, yaitu *K, IV, C, A* dan *T*, dengan satu *output*, yaitu nilai *plaintext P* atau simbol khusus *FAIL* yang menunjukkan bahwa *input* tidak terautentikasi. *C, IV, A* dan *T* dianggap autentik untuk *K* ketika nilai *C, IV, A* dan *T* memiliki nilai yang sama dengan saat operasi enkripsi dengan *input K, IV, A,* dan *P*. Operasi dekripsi terautentikasi memiliki kemungkinan yang tinggi untuk mengembalikan *FAIL* setiap kali *input* dari operasi dekripsi tidak dibuat oleh operasi enkripsi dengan kunci yang identik [23].

Proses enkripsi dan dekripsi diilustrasikan pada Gambar 2.17 [23].



Gambar 2.17. Operasi enkripsi dan dekripsi terautentikasi GCM

Pada ilustrasi proses tersebut, *EK* menunjukkan enkripsi blok cipher menggunakan kunci *K*, *multH* menunjukkan perkalian dalam *Galois Field* dengan ukuran 2 pangkat 128 ($GF(2^{128})$), dan *incr* menunjukkan fungsi *increment* pada counter. Fungsi *incr* menghasilkan urutan nilai counter 32-bit unik untuk proses enkripsi GCM dan menambahkan 32-bit paling kanan dari input dengan operasi modulo untuk menangani *overflow* atau saat counter sudah melebihi 32-bit [23].

2.3 AES-GCM

Galois/Counter Mode (GCM) merupakan salah satu mode operasi yang dapat diterapkan dengan algoritma AES. Mode ini umumnya digunakan untuk blok *cipher* yang menggunakan transformasi kunci simetris dengan ukuran blok yang telah ditentukan menjadi 128 bit, sehingga cocok dengan algoritma AES. Gabungan dari algoritma AES dan operasi GCM dinamakan sebagai AES-GCM [22]. Jumlah panggilan blok cipher yang diperlukan untuk mengenkripsi *plaintext* p-bit dengan AES-GCM adalah $[p/128]+1$. Jumlah perkalian yang dibutuhkan pada $GF(2^{128}$ sama dengan jumlah panggilan blok *cipher*. Apabila ada data tambahan sebesar q-bit yang perlu diautentikasi, maka diperlukan perkalian $[q/128]$ tambahan. Operasi dekripsi serupa dengan operasi enkripsi dan memiliki karakteristik performa yang serupa juga [23].

2.4 Vulnerability Assessment

Vulnerability Assessment atau penilaian kerentanan adalah proses untuk mengidentifikasi kelemahan atau celah pada keamanan sistem komputer yang dapat dimaanfatkan oleh peretas. Tujuan dari penilaian ini adalah menemukan kerentanan atau celah pada sistem dan segera memperbaikinya sebelum peretas menemukan celah tersebut dan dilanjutkan dengan penyerangan sistem [12].

2.5 OWASP Top Ten

Open Web Application Security Project (OWASP) merupakan komunitas *nonprofit* yang terdiri dari *software developer*, *engineer*, dan *freelancer* yang menyediakan sumber daya dan alat untuk keamanan aplikasi web. Setiap 3-4 tahun sekali, OWASP menerbitkan laporan tentang 10 resiko keamanan teratas untuk aplikasi web dengan penerbitan pertama pada tahun 2003 dan yang terbaru pada tahun 2021. OWASP *Top* 10 ditujukan untuk menganalisis keamanan sistem dengan mengidentifikasi kerentanan pada aplikasi berbasis *web* [14].

2.6 Testing for Weak Encryption

Pengetesan enkripsi yang lemah bertujuan untuk mengidentifikasi penggunaan dan implementasi enkripsi yang lemah. Berikut adalah daftar pemeriksaan keamanan dasar yang harus dipenuhi [30].

1. Penggunaan *Initialization Vector* (IV) pada AES128/AES256, harus acak dan tidak bisa diprediksi.
2. Untuk enkripsi asimetris, gunakan *Elliptic Curve Cryptography* (ECC) dengan kurva yang aman seperti *Curve25519*.
3. Ketika menggunakan RSA untuk tanda tangan digital, gunakan *padding Probabilistic Signature Scheme* (PSS).
4. Tidak boleh menggunakan algoritma enkripsi yang lemah seperti MD5, RC4, DES, *Blowfish*, SHA1, 1024-bit RSA atau DSA, 160-bit ECDSA (*elliptic curves*), 80/112-bit 2TDEA (*two key triple DES*).
5. Persyaratan panjang kunci minimum untuk *Key exchange* adalah *Diffie–Hellman key exchange* dengan minimum 2048 bit, HMAC-SHA2 untuk *Message Integrity*, SHA2 256 bit untuk *Message Hash*, RSA 2048 bit untuk *Asymmetric encryption*, AES 128 bit untuk *Symmetric-key algorithm*, PBKDF2, Scrypt, dan Bcrypt untuk *Password Hashing*, dan 256 bit untuk ECDH dan ECDSA.
6. Mode *Cipher Block Chaining* (CBC) tidak boleh digunakan dalam penggunaan SSH.
7. Hindari Mode ECB pada Enkripsi Simetris.
8. Ketika menggunakan PBKDF2 untuk *password hashing*, disarankan untuk menggunakan parameter iterasi lebih dari 10.000.

2.7 OWASP Risk Rating

Menemukan kerentanan pada keamanan dan memperkirakan risiko yang terkait dengan bisnis merupakan hal yang sangat penting. Metodologi OWASP *Risk Rating* memungkinkan pengguna untuk memperkirakan tingkat risiko keseluruhan yang teridentifikasi terhadap bisnis dan membuat keputusan yang baik untuk mengatasinya. Terdapat lima kategori dalam penilaian tingkat risiko keseluruhan, yaitu *note, low, medium, high,* dan *critical*. Selain itu, metodologi ini juga dapat membuat proses penanganan risiko menjadi lebih efisien dan menghindari perdebatan mengenai prioritas karena metodologi ini akan membantu bisnis untuk menangani risiko serius yang berpotensi menimbulkan kerugian besar dan tidak terjebak pada risiko kecil yang dampaknya minimal [31].