

Bahan kuliah IF4020 Kriptografi

Review Beberapa Block Cipher

(Bagian 3: Advanced Encryption Standard (AES))



Oleh: Rinaldi Munir

Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

Latar Belakang

- DES dianggap sudah tidak aman karena kunci dapat ditemukan secara *brute-force*.
- Perlu diusulkan standard algoritma baru sebagai pengganti DES.
- *National Institute of Standards and Technology (NIST)* mengusulkan kepada Pemerintah Federal AS untuk sebuah standard kriptografi baru.
- *NIST* mengadakan lomba membuat standard algoritma kriptografi yang baru. Standard tersebut kelak diberi nama ***Advanced Encryption Standard (AES)***.

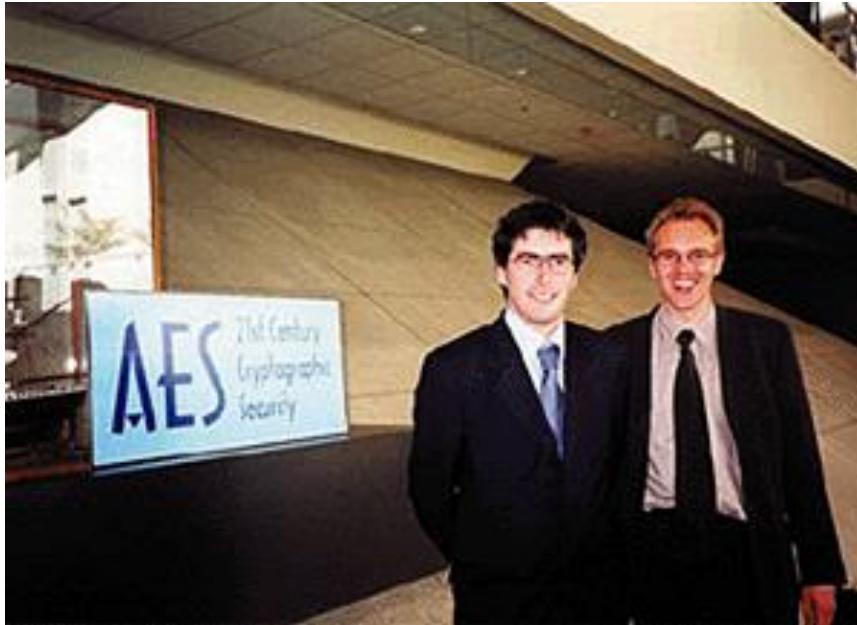
Persyaratan AES:

1. Termasuk ke dalam kelompok algoritma kriptografi simetri berbasis *cipher* blok.
2. Seluruh rancangan algoritma harus publik (tidak dirahasiakan, tidak ada yang disembunyikan)
3. Panjang kunci fleksibel: 128, 192, dan 256 bit.
4. Ukuran blok yang dienkripsi adalah 128 bit.
5. Algoritma dapat diimplementasikan baik sebagai *software* maupun *hardware*.

Lima finalis lomba AES:

1. *Rijndael* (dari Vincent Rijmen dan Joan Daemen – Belgia, 86 suara)
2. *Serpent* (dari Ross Anderson, Eli Biham, dan Lars Knudsen – Inggris, Israel, dan Norwegia, 59 suara).
3. *Twofish* (dari tim yang diketuai oleh Bruce Schneier – USA, 31 suara)
4. *RC6* (dari Laboratorium RSA – USA, 23 suara)
5. *MARS* (dari IBM, 13 suara)

- Pada bulan Oktober 2000, *NIST* mengumumkan untuk memilih Rijndael (dibaca: Rhine-doll)
- Pada bulan November 2001, Rijndael ditetapkan sebagai AES



- Diharapkan Rijndael menjadi standard kriptografi yang dominan paling sedikit selama 10 tahun.



Joan Daemen and Vincent Rijmen, the designers of the Advanced Encryption Standard
(Sumber: <https://medium.com/@stkgroup/what-is-the-advanced-encryption-standard-and-how-does-it-work-abd1ec582df8>)

Spesifikasi Algoritma Rijndael

- Rijndael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit (yaitu 128 bit, 160, 192, ..., 256 bit).
- Panjang kunci dan ukuran blok dapat dipilih secara independent (tidak harus sama. Misal: panjang blok 128 bit, kunci 256 bit).
- Setiap blok dienkripsi dalam sejumlah putaran tertentu, sebagaimana halnya pada *DES*.
- Karena AES menetapkan panjang kunci adalah 128, 192, dan 256, maka dikenal *AES-128*, *AES-192*, dan *AES-256*.

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

Catatan: 1 *word* = 32 bit

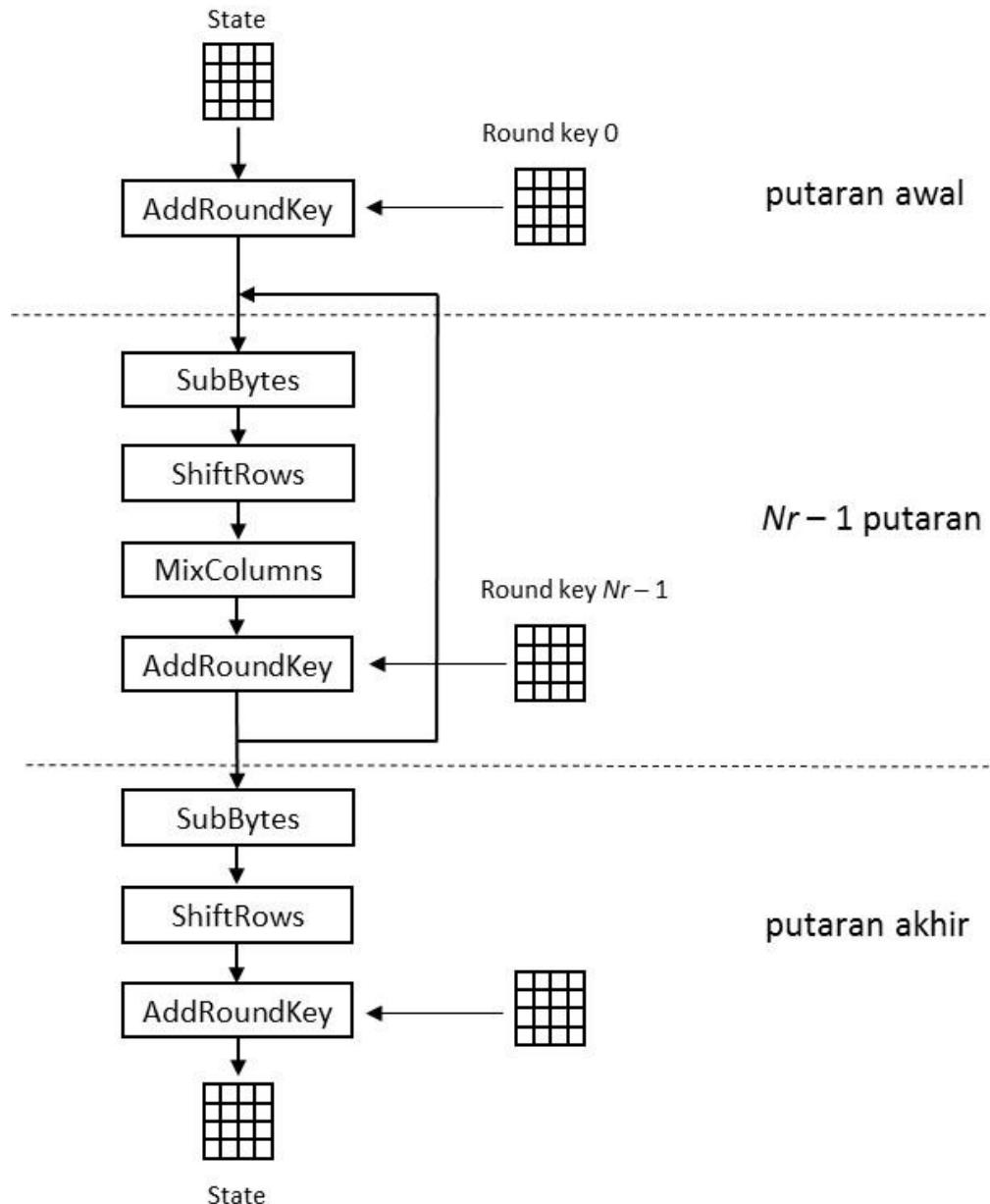
- Secara de-fakto, hanya ada dua varian AES, yaitu AES-128 dan AES-256, karena akan sangat jarang pengguna menggunakan kunci yang panjangnya 192 bit.

- Dengan panjang kunci 128-bit, maka terdapat sebanyak $2^{128} = 3,4 \times 10^{38}$ kemungkinan kunci.
- Jika komputer tercepat dapat mencoba 1 juta kunci setiap detik, maka akan dibutuhkan waktu $5,4 \times 10^{24}$ tahun untuk mencoba seluruh kunci.
- Jika komputer tercepat yang dapat mencoba 1 juta kunci setiap milidetik, maka dibutuhkan waktu $5,4 \times 10^{18}$ tahun untuk mencoba seluruh kunci.
- Artinya, AES diharapkan tahan terhadap serangan *exhaustive key search attack (brute force attack)*

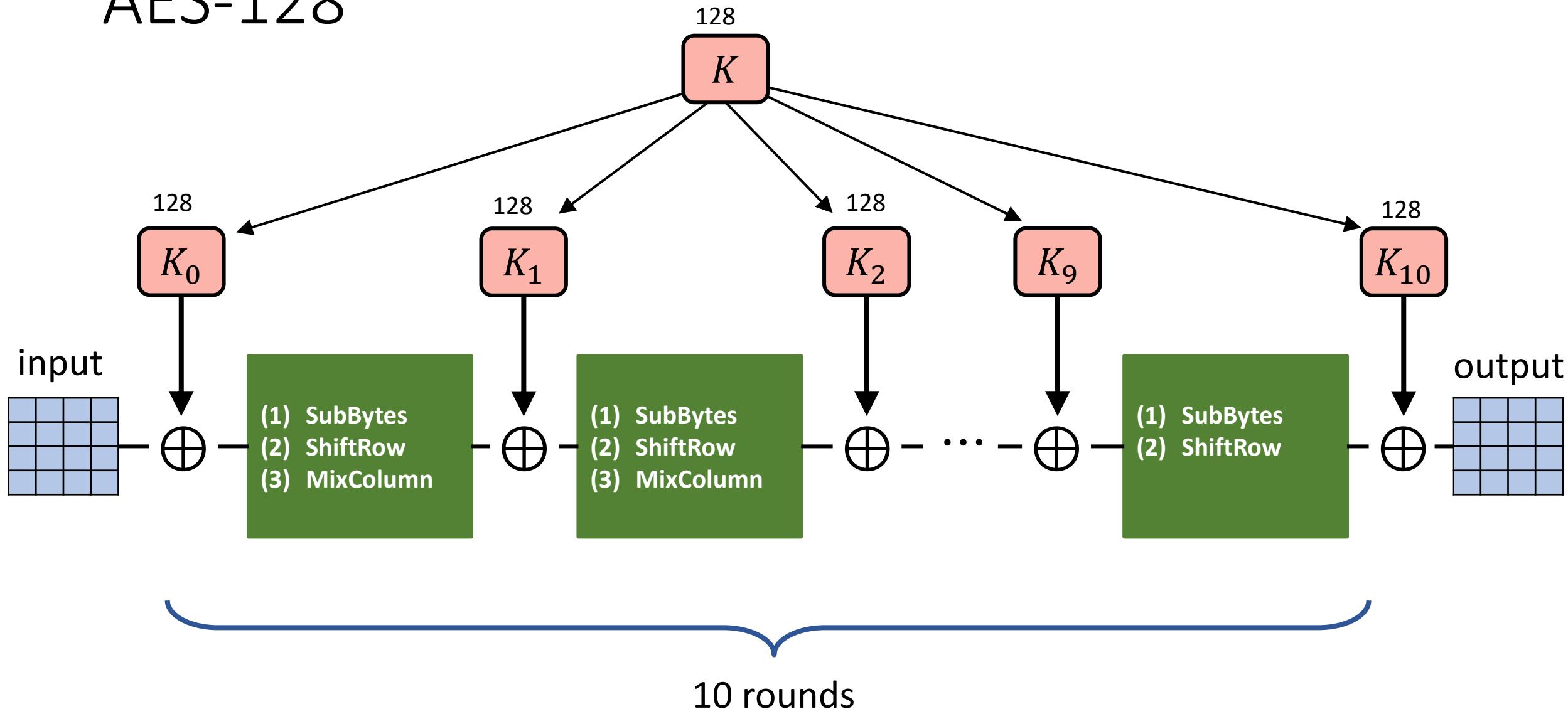
Algoritma *Rijndael*

- Yang dijelaskan di sini Rijndael 128-bit (anjang kunci 128 bit)
- Tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte*.
- *Rijndael* tidak menggunakan jaringan Feistel seperti cipher blok lainnya.
- *Enciphering* dilakukan dalam sejumlah putaran (iterate cipher). Setiap putaran menggunakan kunci internal yang berbeda (disebut *round key*).
- *Enciphering* melibatkan operasi substitusi dan permutasi.

- Garis besar Algoritma *Rijndael* yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitan *round key*):
 1. *AddRoundKey*: melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
 2. Putaran sebanyak $Nr - 1$ kali. Proses yang dilakukan pada setiap putaran adalah:
 - a. *SubBytes*: substitusi *byte* dengan menggunakan tabel substitusi (*S-box*).
 - b. *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.
 - c. *MixColumns*: mengacak data di masing-masing kolom *array state*.
 - d. *AddRoundKey*: melakukan *XOR* antara *state* sekarang *round key*.
 3. *Final round*: proses untuk putaran terakhir:
 - a. *SubBytes*
 - b. *ShiftRows*
 - c. *AddRoundKey*



AES-128



```

#define LENGTH 16           /* Jumlah byte di dalam blok atau kunci */
#define NROWS 4            /* Jumlah baris di dalam state */
#define NCOLS 4            /* Jumlah kolom di dalam state */
#define ROUNDS 10           /* Jumlah putaran */
typedef unsigned char byte; /* unsigned 8-bit integer */

rijndael (byte plaintext[LENGTH], byte ciphertext[LENGTH],
          byte key[LENGTH])
{
    int r;                /* pencacah pengulangan */
    byte state[NROWS][NCOLS]; /* state sekarang */
    struct{byte k[NROWS][NCOLS];} rk[ROUNDS + 1]; /* kunci pada setiap putaran */

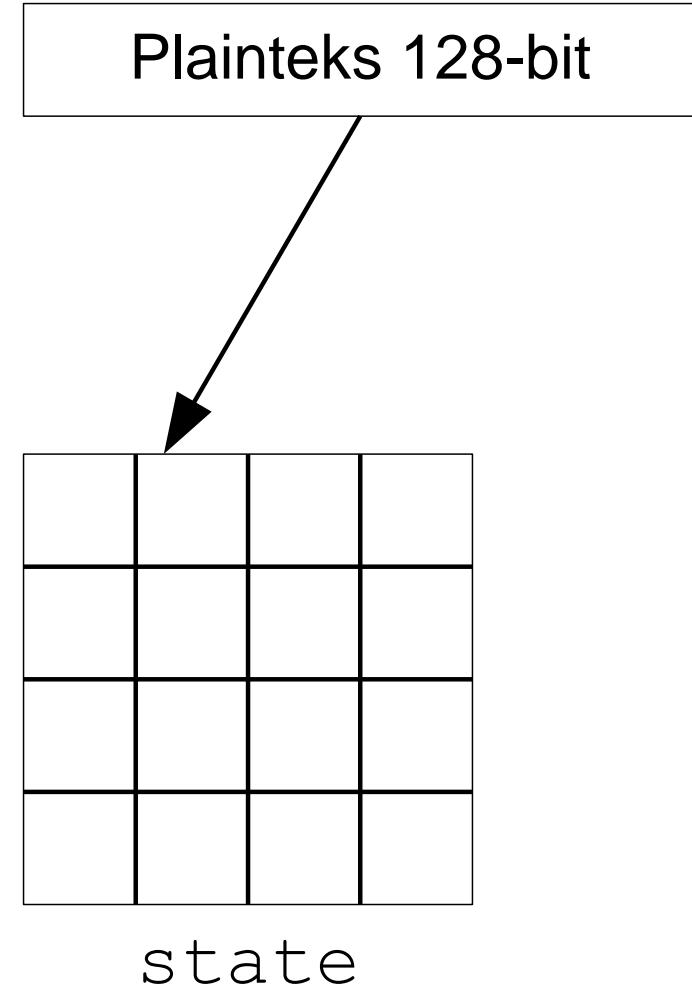
    KeyExpansion(key, rk); /* bangkitkan kunci setiap putaran */
    CopyPlaintextToState(state, plaintext); /* inisialisasi
                                              state sekarang */
    AddRoundKey(state, rk[0]); /* XOR key ke dalam state */

    for (r = 1; r<= ROUNDS - 1; r++)
    {
        SubBytes(state); /* substitusi setiap byte dengan S-box */
        ShiftRows(state); /* rotasikan baris i sejauh i byte */
        MixColumns(state); /* acak masing-masing kolom */
        AddRoundKey(state, rk[r]); /* XOR key ke dalam state */
    }
    SubBytes(state); /* substitusi setiap byte dengan S-box */
    ShiftRows(state); /* rotasikan baris i sejauh i byte */
    AddRoundKey(state, rk[ROUNDS]); /* XOR key ke dalam state */

    CopyStateToCiphertext(ciphertext, state); /* blok cipherteks yang dihasilkan */
}

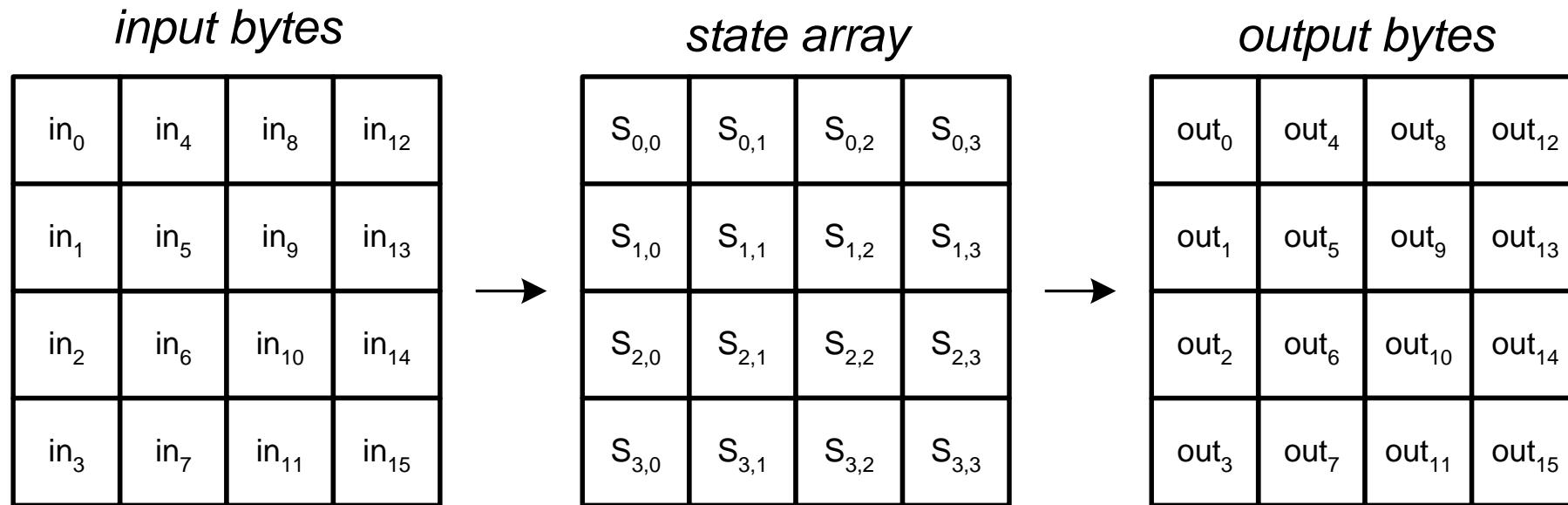
```

- Selama kalkulasi plainteks menjadi cipherteks, status data sekarang disimpan di dalam matriks dua dimensi, state.
- state bertipe *byte* dan berukuran $Nrows \times Ncols$
- Untuk blok data 128-bit, ukuran state adalah 4×4 , seperti gambar di samping.



- Pada awal enkripsi, 16-byte data masukan, $in_0, in_1, \dots, in_{15}$ disalin ke dalam *array state*, direalisasikan oleh fungsi:

`CopyPlaintextToState(state, plaintext)`



- Pada akhir enkripsi, state disalin ke dalam *ciphertext*, direalisasikan oleh fungsi:

`CopyStateToCiphertext(ciphertext, state)`

Contoh elemen state dalam notasi HEX:

23	A2	BC	4A
D4	03	97	F3
16	48	CD	50
FF	DA	10	64

Transformasi *SubBytes()*

- *SubBytes()* melakukan operasi substitusi dengan memetakan setiap byte dari *array state* dengan menggunakan *S-box*.

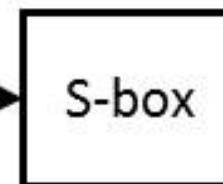
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	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

state

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

state'

b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	b_{33}



Transformasi *SubBytes*

Contoh:

state

23	A2	BC	4A
D4	03	97	F3
16	48	CD	50
FF	DA	10	64

state'

26	3A	65	D6
48	7B	88	0D
47	52	BD	53
16	57	CA	43

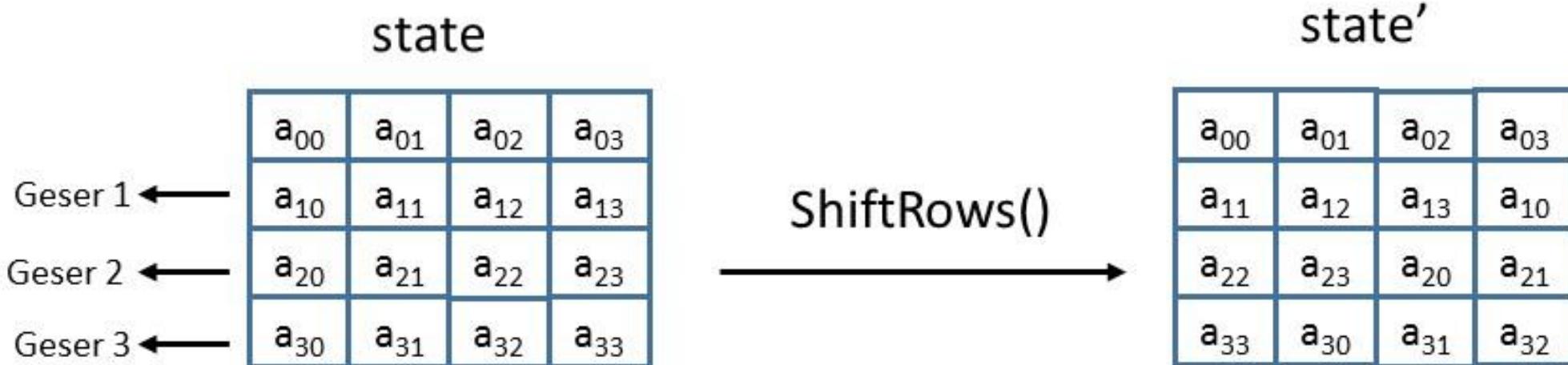


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	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

Proses substitusi 23 menjadi 26

Transformasi *ShiftRows()*

- Transformasi *ShiftRows()* melakukan operasi permutasi dengan pergeseran secara *wrapping* (siklik) pada 3 baris terakhir *array state*.
- Jumlah pergeseran bergantung pada nilai baris (r). Baris $r = 1$ digeser sejauh 1 *byte*, baris $r = 2$ sejauh 2 *byte*, dan baris $r = 3$ sejauh 3 *byte*. Baris $r = 0$ tidak digeser.



26	3A	65	D6
48	7B	88	0D
47	52	BD	53
16	57	CA	43

ShiftRows()

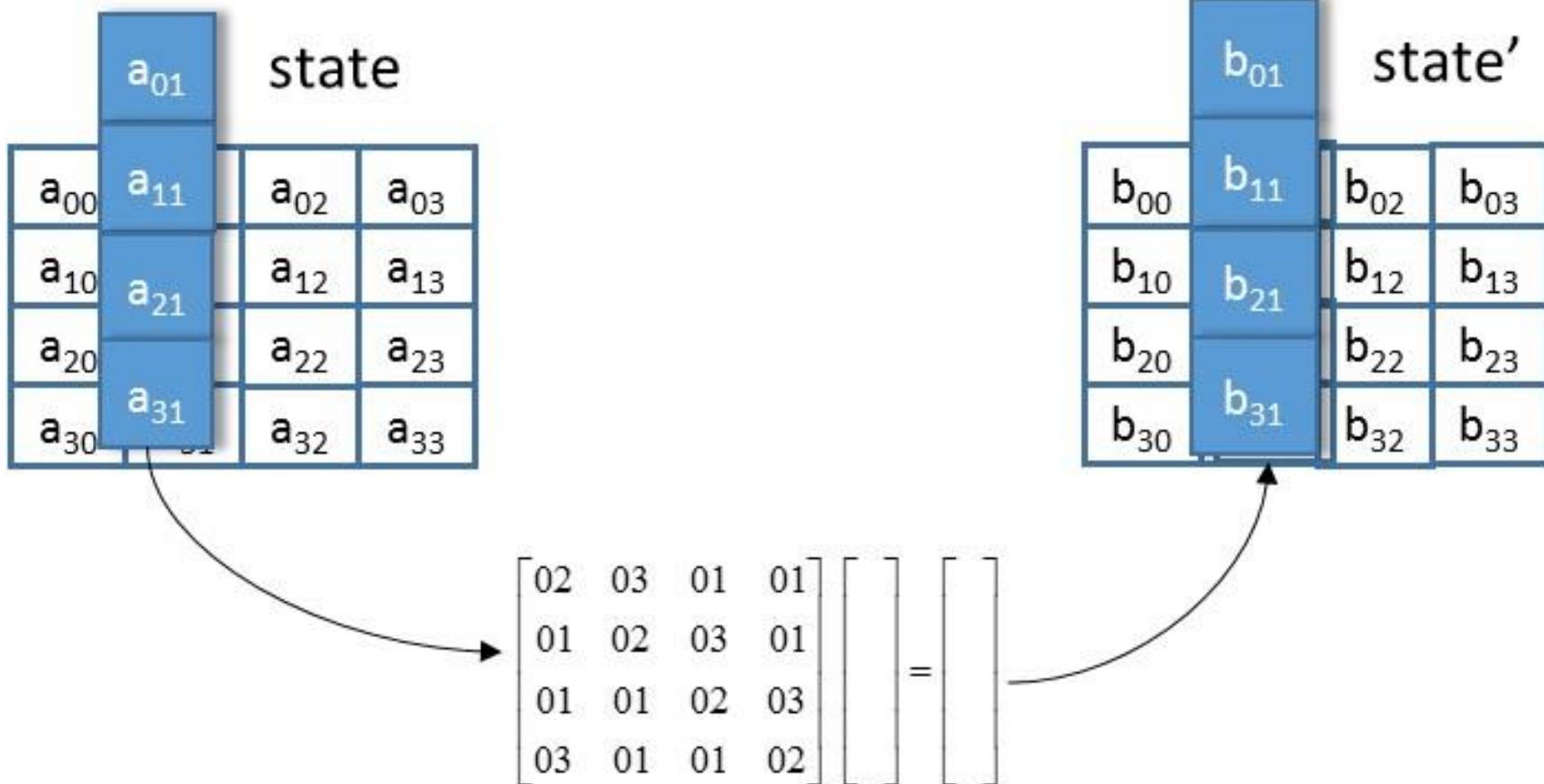


26	3A	65	D6
7B	88	0D	48
BD	53	47	52
43	16	57	CA

Transformasi *MixColumns()*

- Transformasi *MixColumns()* mengalikan matriks *state* dengan sebuah matriks tertentu sbb:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$



$$s'(x) = a(x) \otimes s(x)$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{3,c})$$

Contoh:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 26 \\ 7B \\ BD \\ 43 \end{bmatrix} = \begin{bmatrix} 3F \\ 4F \\ F9 \\ 2A \end{bmatrix}$$

$$(02 \bullet 26) \oplus (03 \bullet 7B) \oplus (01 \bullet BD) \oplus (01 \bullet 43) = 3F$$

$$(01 \bullet 26) \oplus (02 \bullet 7B) \oplus (03 \bullet BD) \oplus (01 \bullet 43) = 4F$$

$$(01 \bullet 26) \oplus (01 \bullet 7B) \oplus (02 \bullet BD) \oplus (03 \bullet 43) = F9$$

$$(03 \bullet 26) \oplus (01 \bullet 7B) \oplus (01 \bullet BD) \oplus (02 \bullet 43) = 2A$$

$$(02 \bullet 26) \oplus (03 \bullet 7B) \oplus (01 \bullet BD) \oplus (01 \bullet 43) = 3F$$

$$\begin{aligned}(02 \bullet 26) &= (0000\ 0010) \times (0010\ 0110) \\&= x \times (x^5 + x^2 + x) \bmod (x^8 + x^4 + x^3 + x + 1) \\&= (x^6 + x^3 + x^2) \bmod (x^8 + x^4 + x^3 + x + 1) \\&= x^6 + x^3 + x^2 \\&= (01001100) \\&= 4C\end{aligned}$$

$$\begin{aligned}(03 \bullet 7B) &= (0000\ 0011) \times (0111\ 1011) \\&= (x + 1) \times (x^6 + x^5 + x^4 + x^3 + x + 1) \bmod (x^8 + x^4 + x^3 + x + 1) \\&= ((x^7 + x^6 + x^5 + x^4 + x^2 + x) + (x^6 + x^5 + x^4 + x^3 + x + 1)) \bmod (x^8 + x^4 + x^3 + x + 1) \\&= (x^7 + (1+1)x^6 + (1+1)x^5 + (1+1)x^4 + x^3 + x^2 + (1+1)x + 1) \bmod (x^8 + x^4 + x^3 + x + 1) \\&= (x^7 + x^3 + x^2 + 1) \bmod (x^8 + x^4 + x^3 + x + 1) \\&= (x^7 + x^3 + x^2 + 1) \\&= (1000\ 1101) = 8D\end{aligned}$$

$$(01 \bullet BD) = BD = 10111101$$

$$(01 \bullet 43) = 43 = 01000011$$

$$(02 \bullet 26) \oplus (03 \bullet 7B) \oplus (01 \bullet BD) \oplus (01 \bullet 43) = 3F$$

Selanjutnya, XOR-kan semua hasil antara tersebut:

$$(02 \bullet 26) = 4C = 0100\ 1100$$

$$(03 \bullet 7B) = 8D = 1000\ 1101$$

$$(01 \bullet BD) = BD = 1011\ 1101$$

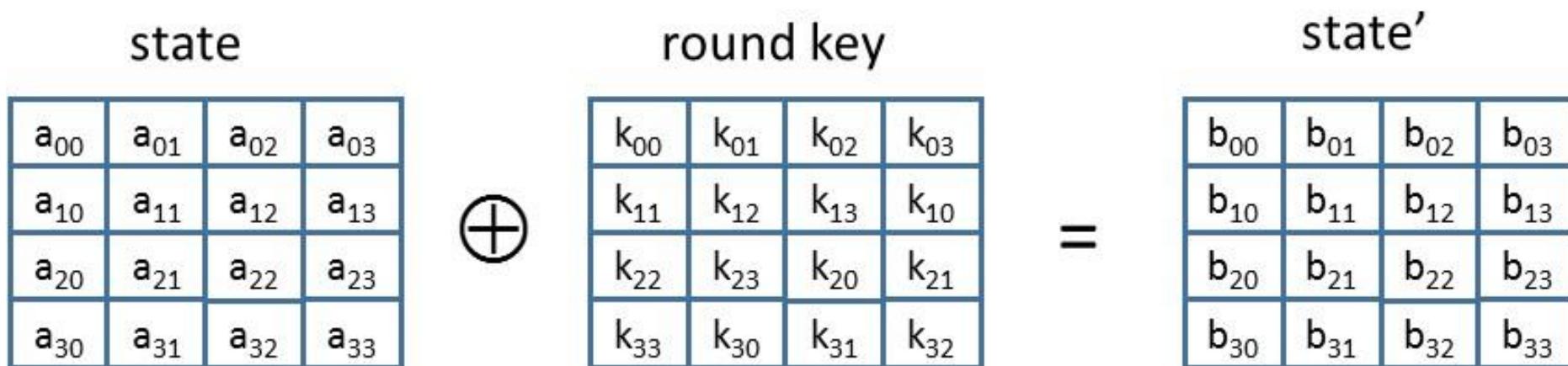
$$\begin{aligned}(01 \bullet 43) &= 43 = \underline{\underline{0100\ 0011}} \oplus \\&\quad 0011\ 1111 = 3F\end{aligned}$$

Jadi, $(02 \bullet 26) \oplus (03 \bullet 7B) \oplus (01 \bullet BD) \oplus (01 \bullet 43) = 3F$

Persamaan lainnya diselesaikan dengan cara yang sama.

Transformasi *AddRoundKey()*

- Transformasi ini melakukan operasi XOR antara *round key* dan *state*, dan hasilnya disimpan di *state*.



Contoh:

3F	B2	CD	F7
4F	D2	E1	9E
F9	2E	1F	7F
2A	B9	4B	F4



4F	5A	7B	10
8C	CD	D1	23
67	2A	FF	45
28	0D	93	2C

=

70	E8	B6	E7
C3	1F	30	BD
9E	04	E0	3A
02	B4	D8	D8

S-box

Elemen-elemen di dalam *S-box* terlihat seperti diisi secara acak, namun sebenarnya ia dihasilkan dari proses perhitungan sebagai berikut:

1. Inisialisasi *S-box* dengan nilai yang menaik dari baris ke baris. Baris ke-0 diisi dengan nilai 00, 01, 02, ..., 0F. Baris ke-1 diisi dengan nilai 10, 11, 12, ..., 1F, dan seterusnya untuk baris-baris lain.
2. Untuk setiap nilai pada baris y kolom x , tentukan balikannya dalam $\text{GF}(2^8)$. Nilai 00 dipetakan ke dirinya sendiri. Penjelasan tentang $\text{GF}(2^8)$ lihat pada lampiran.
3. Hasil dari langkah 2 dikonversi ke vektor kolom bit $(b_0, b_1, \dots, b_8)^T$.

4. Kalikan vektor kolom bit $(b_0, b_1, \dots, b_8)^\top$ dengan sebuah matriks *affine* sebagai berikut:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

5. Selanjutnya, konversi hasil perhitungan $(b'_0, b'_1, \dots, b'_8)^\top$ ke dalam heksadesimal, menjadi elemen *S-box*(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

S-box di dalam Rijndael

Pembentukan Kunci Putaran

- Setiap putaran di dalam algoritma *Rijndael* menggunakan kunci putaran atau *round key*. Kunci putaran dibangkitkan dari kunci eksternal dari pengguna yang dinamakan *cipher key* dan disimbolkan dengan peubah *key*. Pembangkitan semua kunci putaran dilakukan oleh fungsi *KeyExpansion()*.
- Pembangkitan kunci putaran di dalam Algoritma Rijndael tergolong rumit dan agak sukar diterangkan. Tinjau sebuah larik *key* yang panjangnya 16 byte (16 elemen) dan $Nr = 10$ putaran. Sepuluh kunci putaran akan disimpan di dalam larik *rk*. Elemen awal *key* langsung menjadi *rk[0]*. Elemen-elemen kunci lainnya akan disimpan di dalam *rk[1]*, *rk[2]*, ..., *rk[10]*.

ExpansionKey()

Algoritma:

1. Salin elemen-elemen *key* ke dalam larik $w[0]$, $w[1]$, $w[2]$, $w[3]$. Larik $w[0]$ berisi empat elemen pertama key, $w[1]$ berisi empat elemen berikutnya, dan seterusnya.
2. Mulai dari $i = 4$ sampai 43, lakukan:
 - a) Simpan $w[i-1]$ ke dalam peubah *temp*
 - b) Jika i kelipatan 4, lakukan fungsi g berikut:
 - Geser $w[i-1]$ satu *byte* ke kiri secara sirkuler
 - Lakukan substitusi dengan *S-box* terhadap hasil pergeseran tersebut

- XOR-kan hasilnya dengan *round constant* ($Rcon$) ke $i/4$ (atau $Rcon[i/4]$). Nilai $Rcon$ berbeda-beda untuk setiap $j = i/4$, yaitu $Rcon[j] = (RC[j], 0, 0, 0)$, dengan $RC[1]=1$, $RC[j] = 2 \bullet RC[j-1]$, simbol \bullet menyatakan perkalian yang didefinisikan di dalam $GF(2^8)$. Nilai $RC[j]$ di dalam heksadesimal adalah [STA11]: $RC[1]=01$, $RC[2]=02$, $RC[3]=04$, $RC[4]=08$, $RC[5]=10$, $RC[6]=20$, $RC[7]=40$, $RC[8]=80$, $RC[9]=1B$, $RC[10]=36$.
 - Simpan hasil fungsi g ke dalam peubah $temp$
- c) XOR-kan $w[i-4]$ dengan $temp$

Sebagai contoh, misalkan *key* dalam hekadesimal adalah

$$key = (54, 77, 6F, 20, 4F, 6E, 65, 20, 4E, 69, 6E, 65, 20, 54, 77, 6F)$$

Dari *key* langsung diperoleh $rk[0]$ sebagai berikut:

54	4F	4E	20
77	6E	69	54
6F	65	6E	77
20	20	65	6F

Proses di bawah ini hanya menunjukkan pembangkitan $rk[1]$:

1. $w[0] = (54, 77, 6F, 20); w[1] = (4F, 6E, 65, 20); w[2] = (4E, 69, 6E, 65);$
 $w[3] = (20, 54, 77, 6F)$
2. Mulai dari $i = 4$:
 - a) $temp = w[3]$
 - b) $i = 4$ adalah kelipatan 4, maka jalankan fungsi g terhadap $w[3]$:
 - Geser $w[3]$ satu byte ke kiri secara sirkuler: $w[3] = (54, 77, 6E, 20)$
 - Substitusi hasil pergeseran tersebut dengan S -box: (20, F5, 9F, B7)
 - XOR-kan hasil di atas dengan $Rcon[1] = (01, 00, 00, 00)$, menghasilkan (21, F5, 9F, B7)
 - c) Jadi, $g(w[3]) = (21, F5, 9F, B7)$

Untuk $i = 5, 6, 7$:

$$\begin{aligned}w[5] &= w[4] \oplus w[1] = (3A, EC, 96, B7), \\w[6] &= w[5] \oplus w[2] = (74, 85, F8, D2), \\w[7] &= w[6] \oplus w[3] = (54, D1, 8F, BD)\end{aligned}$$

Dari proses di atas diperoleh $rk[1]$ sebagai berikut:

75	3A	74	54
82	EC	85	D1
F0	96	F8	8F
97	B7	D2	BD

Dekripsi

```
#define LENGTH 16      /* Jumlah byte di dalam blok atau kunci */
#define Nrows 4         /* Jumlah baris di dalam state */
#define Ncols 4         /* Jumlah kolom di dalam state */
#define Nr 10           /* Jumlah putaran */
typedef unsigned char byte; /* unsigned 8-bit integer */

decrypt_rijndael (byte in[LENGTH], byte out[LENGTH], byte key[LENGTH])
{
    int round;          /* pencacah pengulangan */
    byte state[Nrows][Ncols];
    struct{byte k[Nrows][Ncols];} rk[Nr + 1]; /* kunci setiap putaran */

    KeyExpansion(key, rk); /* bangkitkan kunci setiap putaran */
    CopyInToState(state, in);

    AddRoundKey(state, rk[Nr]);

    for (round = Nr - 1; round >= 1; round--)
    {
        InvShiftRows(state);
        InvSubBytes(state);
        AddRoundKey(state, rk[round]);
        InvMixColumns(state);
    }

    /* putaran terakhir */
    InvShiftRows(state);
    InvSubBytes(state);
    AddRoundKey(state, rk[0]);

    CopyStateToOut(out, state); /* cipherteks */
}
```

InvSubBytes()

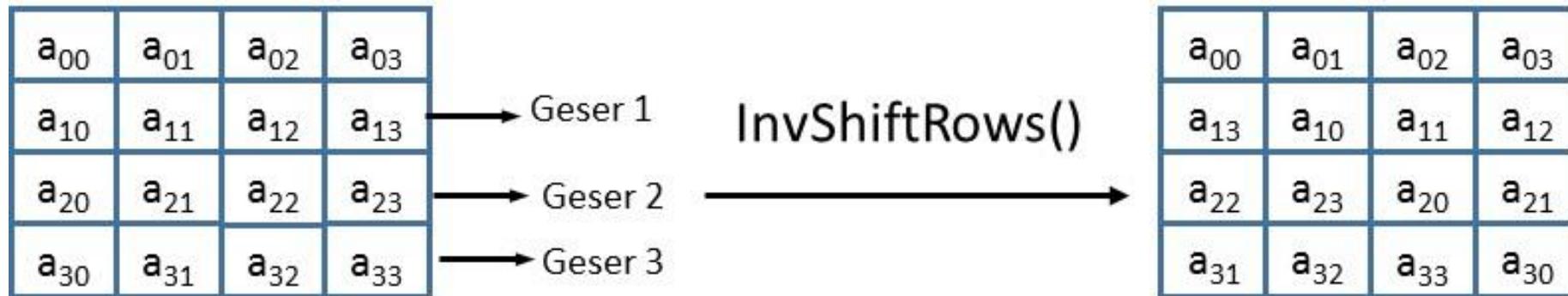
- *InvSubBytes()* sama seperti di dalam *SubBytes()*, hanya saja *S-box* yang digunakan adalah inversi dari *S-box*

	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	B	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Gambar 6.27 Inversi S-box di dalam Rijndael

InvShiftRows()

- *InvShiftRows* sama seperti *ShiftRows* namun melakukan pergeseran dalam arah berlawanan (ke kanan) untuk tiap-tiap baris pada tiga baris terakhir di dalam *state*



InvMixColumns()

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

URL yang terkait dengan AES:

1. AES Homepage, <http://www.nist.gov/CryptoToolkit>
2. J. Daemen, V. Rijmen, AES Proposal: Rijndael,
<http://www.esat.kuleuven.ac.be/~rizmen/>

Demo online AES

<https://encode-decode.com/aes128-encrypt-online/>

The screenshot shows a Firefox browser window with several tabs open at the top:

- (9) WhatsApp
- Inbox (7,447) - rinaldi@inf...
- Block cipher - Wikipedia
- aes128 encrypt & decrypt online (active tab)
- Diguyur Hujan Deras, Banj...

The main content area displays the "aes128 encrypt & decrypt online" tool from encode-decode.com. The page has a header with navigation links for encoding & decoding, hash generation, encryption & decryption, and guide & faq. A dropdown menu for supported encryptions shows "aes128".

The left sidebar contains social sharing icons for Facebook, Twitter, Google+, Pinterest, and LinkedIn, along with a link to "penasehat".

The right sidebar shows a large block of encrypted text output from the tool.

At the bottom, there are two green buttons: "Encrypt string →" and "← Decrypt string".

A footer banner at the bottom of the page encourages users to try the tool: "Give our aes128 encrypt/decrypt tool a try!"

The taskbar at the bottom of the screen shows various pinned icons and system status indicators.

Program enkripsi-dekripsi AES dengan Python

```
In [1]: #import Library crypto dan base64
from Crypto.Cipher import AES
from Crypto import Random
import base64
```

```
In [2]: def enkripsi (pesan):
    print("Plainteks: \n", pesan)
    print("Ketikkan kunci (16 karakter):")
    kunci = input()
    iv = Random.new().read(AES.block_size)
    cipher = AES.new (kunci,AES.MODE_CBC,iv)      #defenisikan AES mode CBC

    cipherteks = base64.b64encode (iv + cipher.encrypt(pesan))  #enkripsi pesan

    print("\n")
    print("Cipherteks: \n", cipherteks)      #tampilkan ciphertext
```

In [15]: `def dekripsi(pesan):`

```
    print("Cipherteks: \n", pesan)

    print("Ketikkan kunci (16 karakter):")
    kunci = input()
    iv = pesan[:16]
    cipher = AES.new(kunci, AES.MODE_CBC,iv)

    #dekripsi pesan
    plainteks = cipher.decrypt(base64.b64decode(pesan))

    print("\n")

    #tampilkan plainteks
    print ("Pesan setelah didekripsi adalah: \n", plainteks[16:])
```

```
In [17]: def ProgramEnkripsiDekripsiAES():
    print("-- Enkripsi dan dekripsi pesan dengan AES --")
    print("1: Enkripsi pesan")
    print("2: Dekripsi pesan")

    pilih = input()
    if pilih == '1':
        print("Ketikkan pesan yang akan dienkripsi:")
        pesan = input()
        n = len(pesan)
        if n % 16 != 0:
            pesan = pesan + ' ' * (16 - n % 16)      #padding dengan spasi
        print("Enkripsi pesan...")
        enkripsi(pesan)

    elif pilih == '2':
        print("Ketikkan pesan yang akan didekripsi:")
        pesan = input()
        #pesan= pesan.rjust(32)
        print("Dekripsi pesan...")
        dekripsi(pesan)

    else:
        print("Pilihan salah")
```

Run program (enkripsi):

```
In [6]: ProgramEnkripsiDekripsiAES()
```

```
-- Enkripsi dan dekripsi pesan dengan AES --
```

```
1: Enkripsi pesan
```

```
2: Dekripsi pesan
```

```
1
```

```
Ketikkan pesan yang akan dienkripsi:
```

```
Hari ini Sabtu 18-2-2023 di Bandung
```

```
Enkripsi pesan...
```

```
Plainteks:
```

```
Hari ini Sabtu 18-2-2023 di Bandung
```

```
Ketikkan kunci (16 karakter):
```

```
abcdefghij123456
```

```
Cipherteks:
```

```
b'E8Hfq13qMGWfqv028jgvnkIhgDsJLI0teVT0DHkdS3wSnJCgeTB9BWtd50gtsF0YwcYlQ4IkUFjqUjABJjpH3g=='
```

Run program (**dekripsi**):

```
In [16]: ProgramEnkripsiDekripsiAES()
```

```
-- Enkripsi dan dekripsi pesan dengan AES --
```

```
1: Enkripsi pesan
```

```
2: Dekripsi pesan
```

```
2
```

```
Ketikkan pesan yang akan didekripsi:
```

```
E8Hfql3qMGWfqv028jgvnkIhgDsJLl0teVT0DHkdS3wSnJCgeTB9Bwt50gtsF0YwcYlQ4IkUFjqUjABJjpH3g==
```

```
Dekripsi pesan...
```

```
Cipherteks:
```

```
E8Hfql3qMGWfqv028jgvnkIhgDsJLl0teVT0DHkdS3wSnJCgeTB9Bwt50gtsF0YwcYlQ4IkUFjqUjABJjpH3g==
```

```
Ketikkan kunci (16 karakter):
```

```
abcdefghij123456
```

```
Pesan setelah didekripsi adalah:
```

```
b'Hari ini Sabtu 18-2-2023 di Bandung'
```

Beberapa algoritma kriptografi simetri:

Cipher	Pembuat	Panjang Kunci	Keterangan
<i>Blowfish</i>	Bruce Schneier	1 – 448 bit	<i>Old and slow</i>
<i>DES</i>	IBM	56 bit	<i>Too weak to use now</i>
<i>IDEA</i>	Massey dan Xuejia	128 bit	<i>Good, but patented</i>
<i>RC4</i>	Ronald Rivest	1 – 2048 bit	<i>Caution: some keys are weak</i>
<i>RC5</i>	Ronald Rivest	128 – 256 bit	<i>Good, but patented</i>
<i>Rijndael</i>	Daemen dan Rijmen	128 – 256 bit	<i>Best choice</i>
<i>Serpent</i>	Anderson, Biham, Knudsen	128 – 256 bit	<i>Very strong</i>
<i>Triple DES</i>	IBM	168 bit	<i>Second best choice</i>
<i>Twofish</i>	Bruce Schneier	128 – 256 bit	<i>Very strong; widely used</i>

Lampiran

Galois Field GF(2^m)

- Disebut juga medan berhingga biner.
- GF(2^m) atau F₂^m adalah ruang vektor berdimensi m pada GF(2). Setiap elemen di dalam GF(2^m) adalah integer dalam representasi biner sepanjang maksimal m bit.
- String biner $\alpha_{m-1} \dots \alpha_1 \alpha_0$, $\alpha_i \in \{0,1\}$, dapat dinyatakan dalam polinom
$$\alpha_{m-1}x^{m-1} + \dots + \alpha_1x + \alpha_0$$
- Jadi, setiap $a \in GF(2^m)$ dapat dinyatakan sebagai
$$a = \alpha_{m-1}x^{m-1} + \dots + \alpha_1x + \alpha_0$$
- Contoh: 1101 dapat dinyatakan dengan $x^3 + x^2 + 1$

Operasi aritmetika pada $\text{GF}(2^m)$

Misalkan $a = (a_{m-1} \dots a_1 a_0)$ dan $b = (b_{m-1} \dots b_1 b_0) \in \text{GF}(2^m)$

- **Penjumlahan:**

$a + b = c = (c_{m-1} \dots c_1 c_0)$ dimana $c_i = (a_i + b_i) \bmod 2$, $c \in \text{GF}(2^m)$

- **Perkalian:** $a \cdot b = c = (c_{m-1} \dots c_1 c_0)$ dimana c adalah sisa pembagian polinom $a(x) \cdot b(x)$ dengan *irreducible polynomial* derajat m , $c \in \text{GF}(2^m)$

Contoh: Misalkan $a = 1101 = x^3 + x^2 + 1$ dan $b = 0110 = x^2 + x$
 a dan $b \in GF(2^4)$

(i) $a + b = (x^3 + x^2 + 1) + (x^2 + x) = x^3 + 2x^2 + x + 1 \pmod{2}$

Bagi tiap koefisien dengan 2,
lalu ambil sisanya

$$\begin{aligned} &= x^3 + 0x^2 + x + 1 \\ &= x^3 + x + 1 \end{aligned}$$

Dalam representasi biner:

1101

0110 +

1011 → sama dengan hasil operasi XOR

$$\therefore a + b = 1011 = a \text{ XOR } b$$

$$\begin{aligned}\text{(ii)} \quad a \cdot b &= (x^3 + x^2 + 1) \cdot (x^2 + x) = x^5 + 2x^4 + x^3 + x^2 + x \pmod{2} \\ &= x^5 + x^3 + x^2 + x\end{aligned}$$

Karena $m = 4$ hasilnya direduksi menjadi derajat < 4 oleh
irreducible polynomial $x^4 + x + 1$

$$\begin{aligned}x^5 + x^3 + x^2 + x \pmod{f(x)} &= (x^4 + x + 1)x + x^5 + x^3 + x^2 + x \\ &= 2x^5 + x^3 + 2x^2 + 2x \pmod{2} \\ &= x^3\end{aligned}$$

$$\therefore a \cdot b = 1000$$