Advanced Encryption Standard (AES)

Bahan Kuliah

IF4020 Kriptografi

Latar Belakang

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

- Persyaratan algoritma baru:

1. Termasuk ke dalam kelompok algoritma kriptografi simetri berbasis *cipher* blok.
2. Seluruh rancangan algoritma harus publik (tidak dirahasiakan)
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:

1. *Rijndael* (dari Vincent **Rij**men da**n** Joan **Dae**men – Be**l**gia, 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.

Spesifikasi Algoritma Rijndael

- Rijndael mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit.
- Panjang kunci dan ukuran blok dapat dipilih secara independen.
- 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 tercepat yang dapat mencoba 1 juta kunci setiap milidetik, maka dibutuhkan waktu $5{,}4 \times 10^{18}$ tahun untuk mencoba seluruh kunci.

Algoritma *Rijndael*

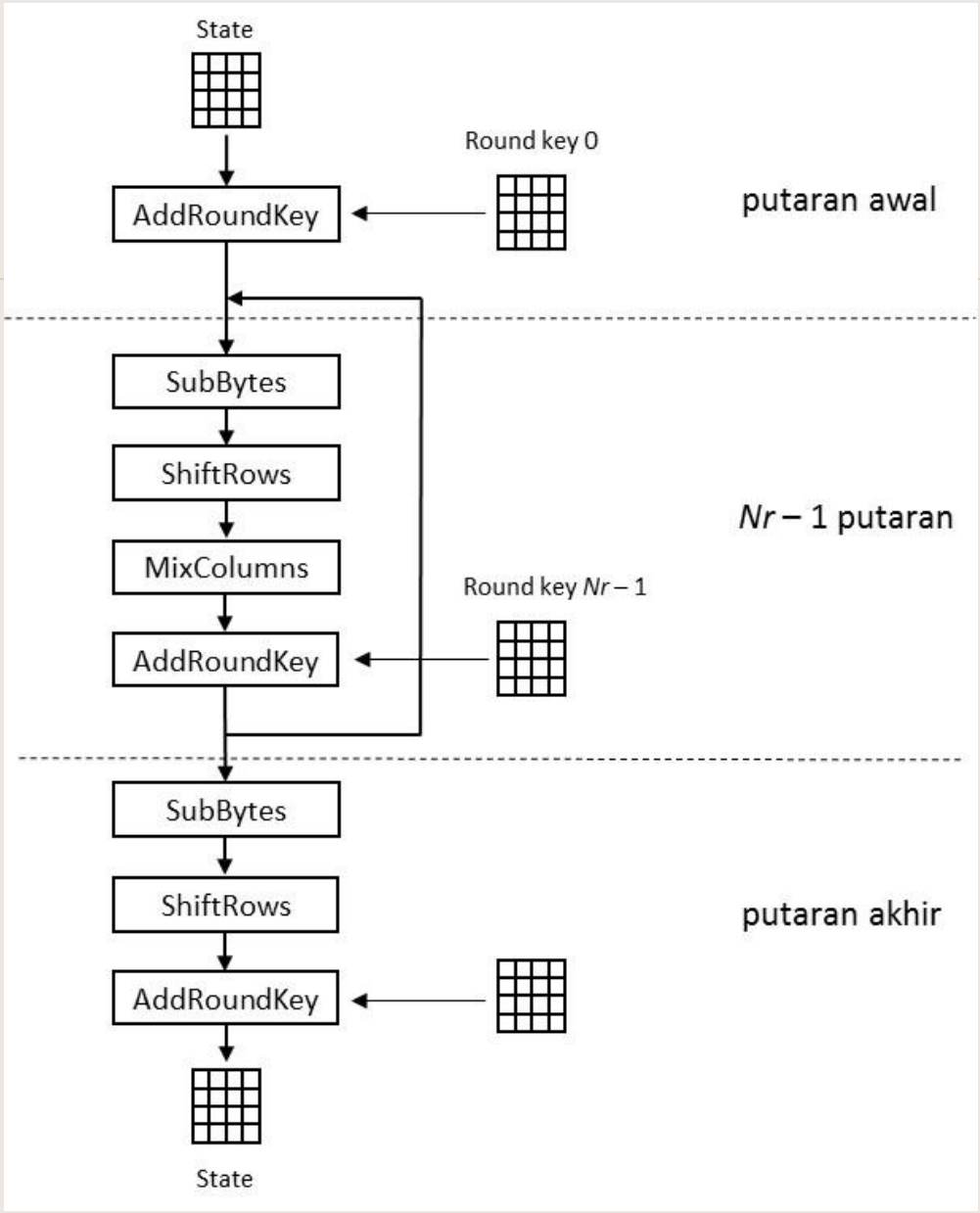
- Tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte*.
- Setiap putaran mengunakan kunci internal yang berbeda (disebut *round key*).
- *Enciphering* melibatkan operasi substitusi dan permutasi.

- 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


State
Round key 0
AddRoundKey
putaran awal
SubBytes
ShiftRows
MixColumns
Nr – 1 putaran
Round key Nr – 1
AddRoundKey
SubBytes
ShiftRows
putaran akhir
AddRoundKey
State

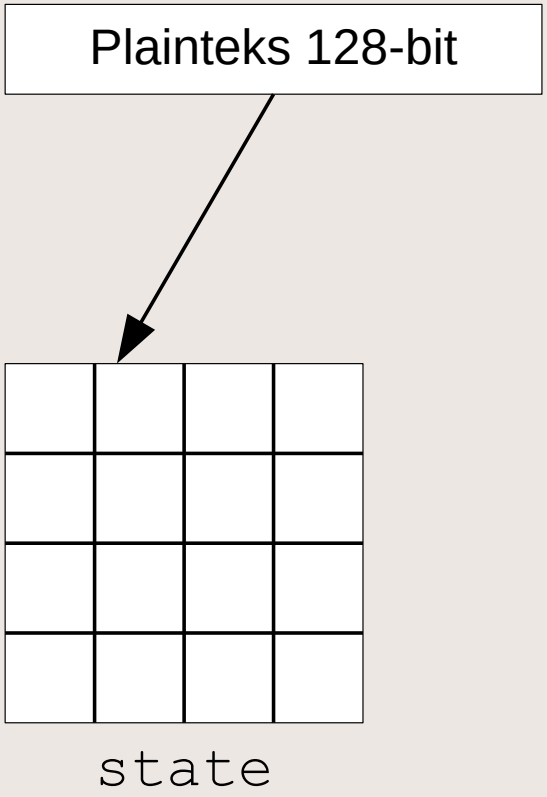
```
#define LENGTH 16            /* Jumlah byte di dalam blok atau kunci */
#define NROWS 4              /* Jumlah baris di dalam state */
#define NCOLS 4              /* Jumlah kolom di dalam state */
#define ROUNDS 10            /* Jumlah putaran */
typedef unsigned char byte;  /* unsigned 8-bit integer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

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 */

}
```

Algoritma Rijndael mempunyai 3 parameter:

1. `plaintext` : *array* berukuran 16-*byte*, yang berisi data masukan.
2. `ciphertext` : *array* berukuran 16-*byte*, yang berisi hasil enkripsi.
3. `key` : *array* berukuran 16-*byte*, yang berisi kunci *ciphering* (disebut juga *cipher key*).

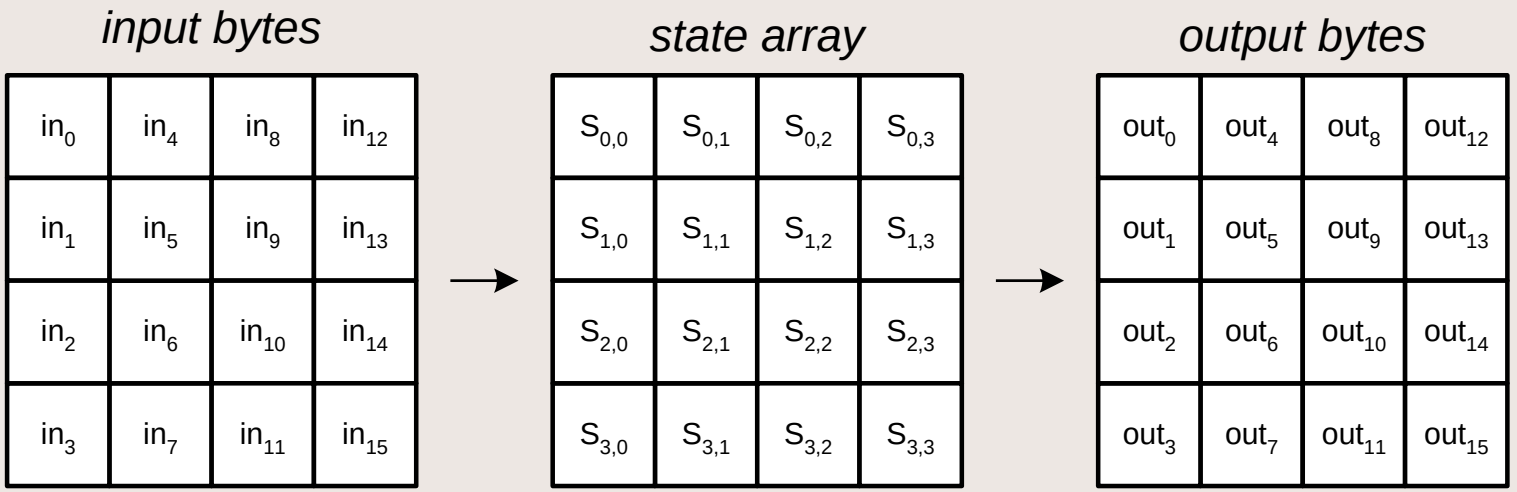
- Dengan 16 *byte*, maka blok data dan kunci yang berukuran 128-bit dapat disimpan di dalam *array* 16 elemen ($16 \times 8 = 128$).

- Blok plainteks disimpan di dalam *matrix of byte* yang bernama `state` dan berukuran `NROWS` × `NCOLS`.
- Untuk blok data 128-bit, ukuran `state` 4 × 4.



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

  `CopyPlaintextToState(state, plaintext))`



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()* memetakan setiap byte dari *array state* dengan menggunakan *S-box*.

hex		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

S-BOX

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

hex	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

x

19

	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

d4

Transformasi *ShiftRows()*

- Transformasi *ShiftRows()* melakukan pergeseran secara *wrapping* (siklik) pada 3 baris terakhir dari *array state*.

- Jumlah pergeseran bergantung pada nilai baris (*r*). Baris *r* = 1 digeser sejauh 1 *byte*, baris *r* = 2 digeser sejauh 2 *byte*, dan baris *r* = 3 digeser sejauh 3 *byte*. Baris *r* = 0 tidak digeser.

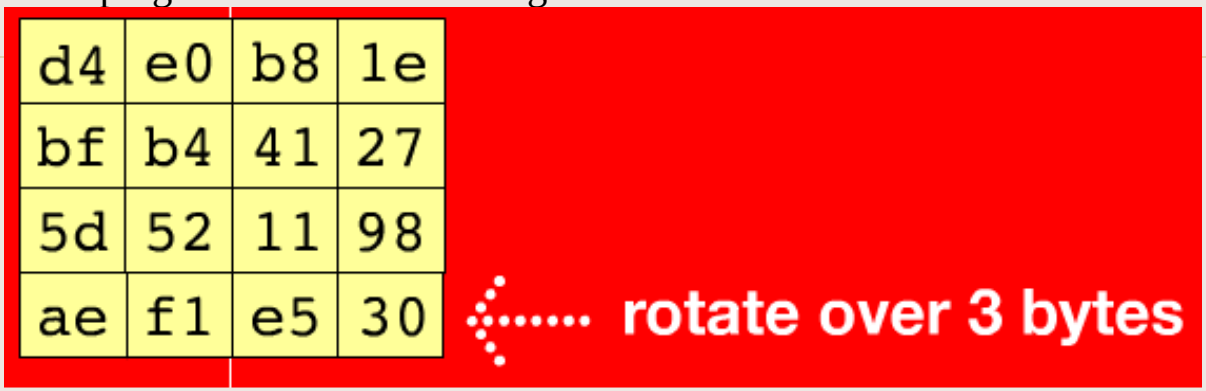
Geser baris ke-1:

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

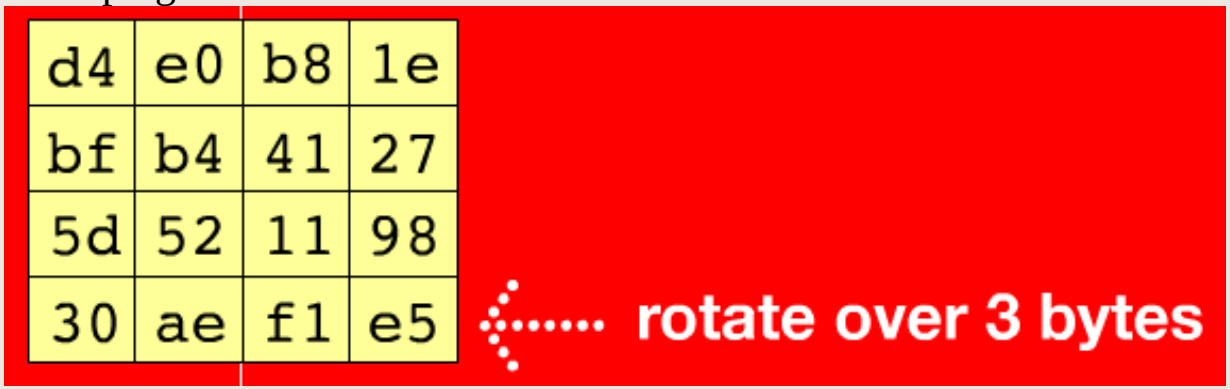
Hasil pergeseran baris ke-1 dan geser baris ke-2:

d4	e0	b8	1e
bf	b4	41	27
11	98	5d	52
ae	f1	e5	30

Hasil pergeseran baris ke-2 dan geser baris ke-3:



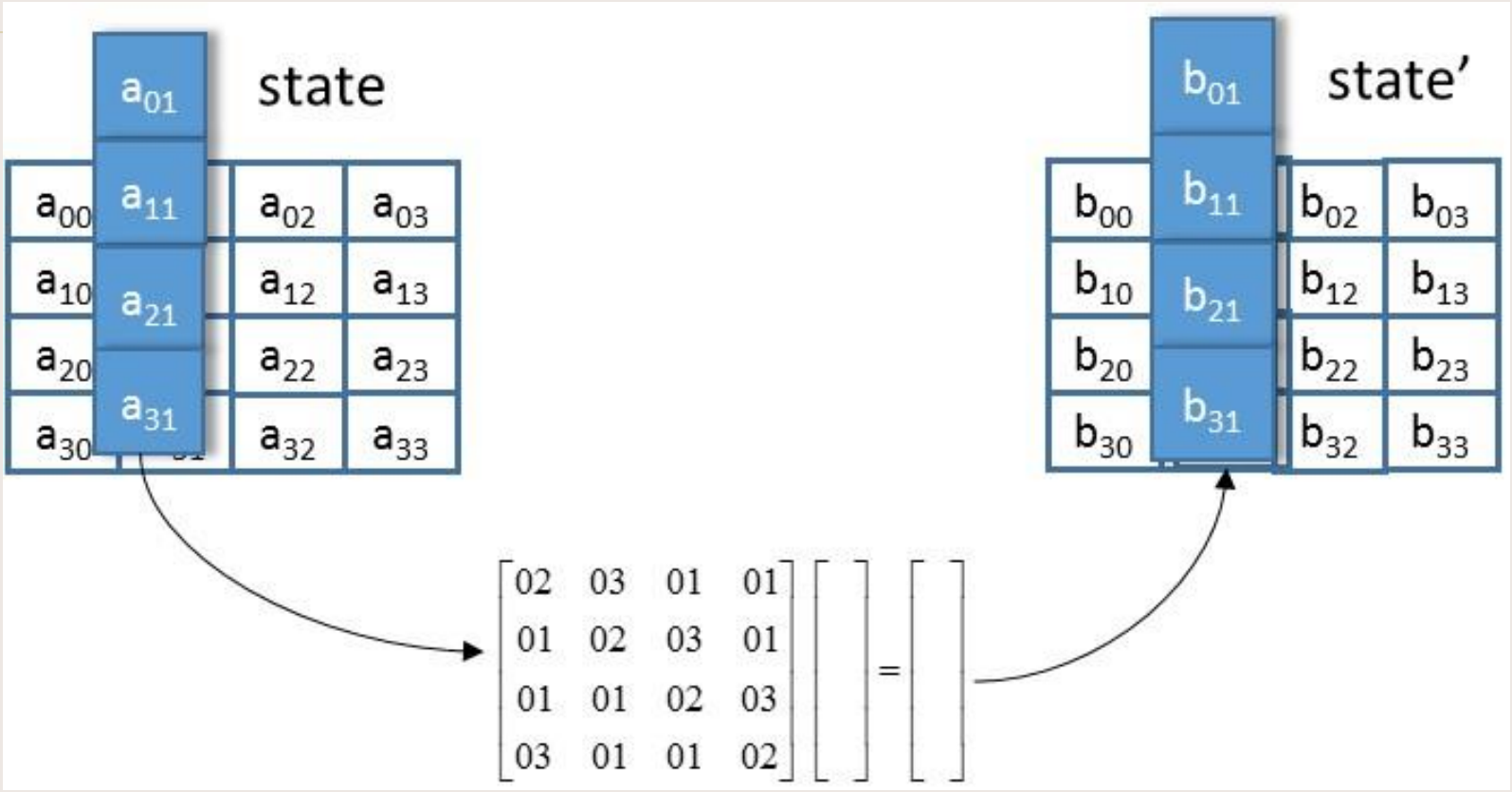
Hasil pergeseran baris ke-3:



Transformasi *MixColumns()*

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

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


state
a01
a11
a21
a31
a00
a02
a03
a10
a12
a13
a20
a22
a23
a30
a32
a33
02 03 01 01
01 02 03 01
01 01 02 03
03 01 01 02
=
state'
b01
b11
b21
b31
b00
b02
b03
b10
b12
b13
b20
b22
b23
b30
b32
b33

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

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

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

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

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

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

Contoh:

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

$(02 \bullet 26) \oplus (03 \bullet 7B) \oplus (01 \bullet BD) \oplus (01 \bullet 43) = 3F$
$(01 \bullet 26) \oplus (02 \bullet 7B) \oplus (03 \bullet BD) \oplus (01 \bullet 43) = 4F$
$(01 \bullet 26) \oplus (01 \bullet 7B) \oplus (02 \bullet BD) \oplus (03 \bullet 43) = F9$
$(03 \bullet 26) \oplus (01 \bullet 7B) \oplus (01 \bullet BD) \oplus (02 \bullet 43) = 2A$

$(02 \bullet 26) = (0000\ 0010) \times (0010\ 0110)$

$= x \times (x^5 + x^2 + x)$ **mod** $(x^8 + x^4 + x^3 + x + 1)$

$= (x^6 + x^3 + x^2)$ **mod** $(x^8 + x^4 + x^3 + x + 1)$

$= x^6 + x^3 + x^2$

$= (01001100)$

$= 4C$

$(03 \bullet 7B) = (0000\ 0011) \times (0111\ 1011)$

$= (x + 1) \times (x^6 + x^5 + x^4 + x^3 + x + 1)$ **mod** $(x^8 + x^4 + x^3 + x + 1)$

$= ((x^7 + x^6 + x^5 + x^4 + x^2 + x) + (x^6 + x^5 + x^4 + x^3 + x + 1))$ **mod** $(x^8 + x^4 + x^3 + x + 1)$

$= (x^7 + (1 + 1)x^6 + (1 + 1)x^5 + (1 + 1)x^4 + x^3 + x^2 + (1 + 1)x + 1)$ **mod** $(x^8 + x^4 + x^3 + x + 1)$

$= (x^7 + x^3 + x^2 + 1)$ **mod** $(x^8 + x^4 + x^3 + x + 1)$

$= (x^7 + x^3 + x^2 + 1)$

$= (1000\ 1101) = 8D$

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

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

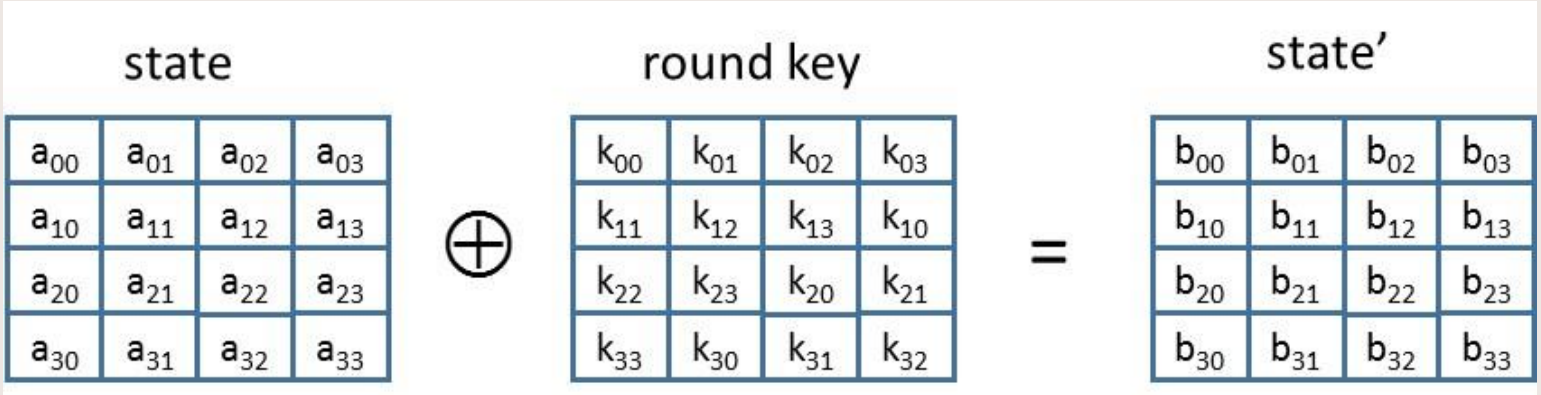
Selanjutnya, XOR-kan semua hasil antara tersebut:

$(02 \bullet 26) = 0100\ 1100$
$(03 \bullet 7B) = 1000\ 1101$
$(01 \bullet BD) = 1011\ 1101$
$(01 \bullet 43) = \underline{0100\ 0011 \oplus}$
$0011\ 1111 = 3F$

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 terhadap sebuah *round key* dengan *array state*, dan hasilnya disimpan di *array state*.



Contoh:

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

$\oplus$

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

Ekspansi Kunci

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 hasil di atas 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•*RC*[*j*-1], simbol • 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*

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/

Beberapa algoritma kriptografi simetri:

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