RPS22006 Teknologi Kriptografi Militer

Block Cipher

Oleh: Rinaldi Munir

Program Studi Magister Terapan
Cyber Defense Engineering
Faculty of Defense Technology
Universitas Pertahanan

Cipher Blok (*Block Cipher*)

- Berbeda dengan *cipher* alir, pada *cipher* blok plainteks dibagi menjadi blok-blok bit dengan panjang sama. Ukuran blok yang umum adalah 64 bit, 128 bit, 256 bit, dsb.

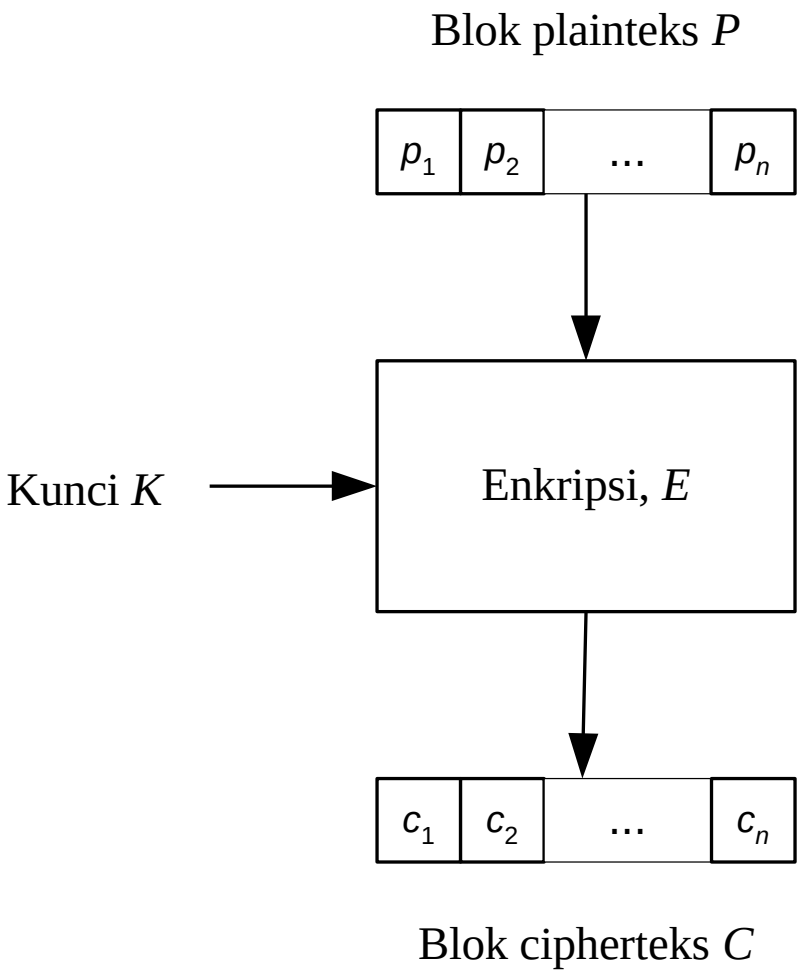
- Panjang blok cipherteks = panjang blok plainteks.

- Enkripsi dilakukan pada setiap blok plainteks dengan menggunakan bit-bit kunci

- Panjang kunci eksternal (yang diberikan oleh pengguna) tidak harus sama dengan panjang blok plainteks.

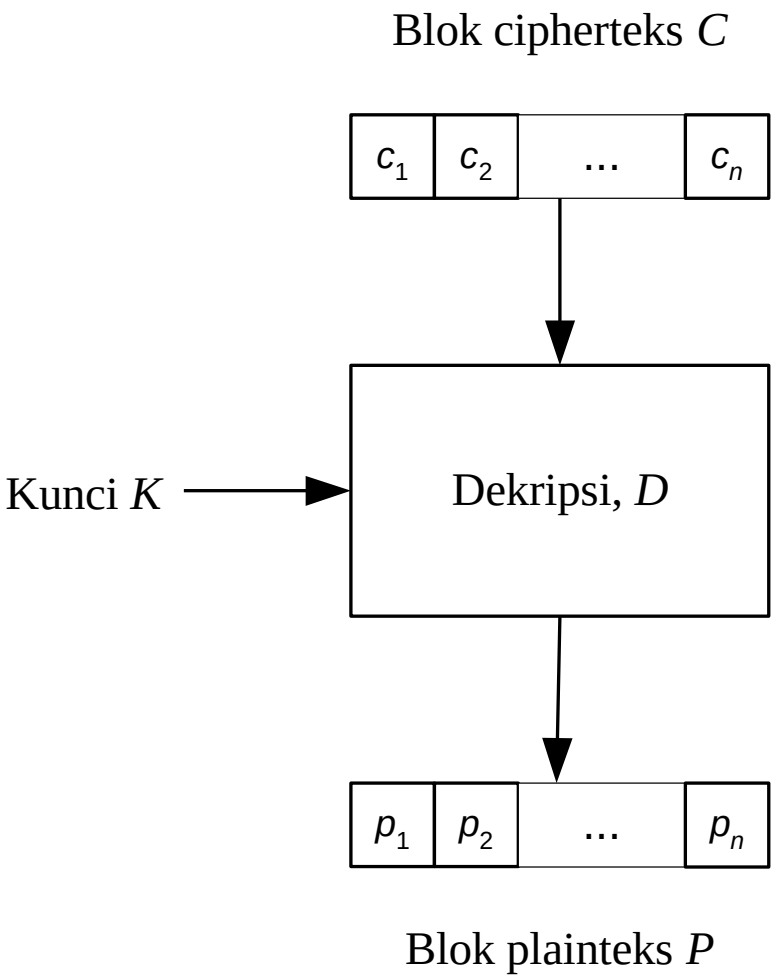
Blok plainteks berukuran n bit:

$P = (p_1, p_2, \ldots, p_n)$, $p_i \in \{0, 1\}$



Blok cipherteks berukuran n bit:

$C = (c_1, c_2, \ldots, c_n)$, $c_i \in \{0, 1\}$



- E dan D adalah fungsi enkripsi dan dekripsi yang digunakan di dalam *cipher*:

  C = E(P) dan P = D(C)

- Contoh fungsi E yang sederhana misalkan algoritmanya adalah sbb:

  1. XOR-kan blok plainteks *P* dengan *K*
  2. lalu geser secara *wrapping* bit-bit dari hasil langkah 1 satu posisi ke kiri

$$C = E_K(P) = (P \oplus K) << 1$$

- D adalah kebalikan (*invers*) dari E. Contoh fungsi D yang sederhana adalah sbb:

  1. Geser secara *wrapping* bit-bit ciphertes C satu posisi ke kanan
  2. Lalu XOR-kan blok hasil Langkah 1 dengan *K*

$$P = E_K(C) = (C >> 1) \oplus K$$

Contoh: Misalkan plainteks adalah 110111010001000101000101, dibagi menjadi tiga buah blok masing-masing berukuran 8 bit:

11011101	00010001	01000101
P1	P2	P3

Kunci yang digunakan adalah K = 01010110

Enkripsi blok pertama, 11011101, dengan E sebagai berikut:

1. P1 + K = 11011101 ⊕ 01010110 = 10001011
2. 10001011 << 1 = 00010111

Jadi, C1 = 00010111. Cara yang sama dilakukan untuk P2 dan P3.

Dekripsi blok cipherteks pertama, 00010111, dengan D sebagai berikut:

1. 00010111 >> 1 = 10001011
2. 10001011 ⊕ K = 10001011 ⊕ 01010110 = 11011101 = P1

- Lakukan proses yang sama untuk blok P2, P3, dst.

- Tentu saja algoritma enkripsi E dan D di atas sangat lemah karena mudah dipecahkan.

- *Cipher* blok modern membuat E dan D sangat rumit prosesnya sehingga sangat sukar dipecahkan oleh kriptanalis.

- Fungsi E dan D melibatkan sejumlah teknik seperti teknik substitusi, permutasi, pergeseran (*shifting*), kompresi, ekspansi, dsb.

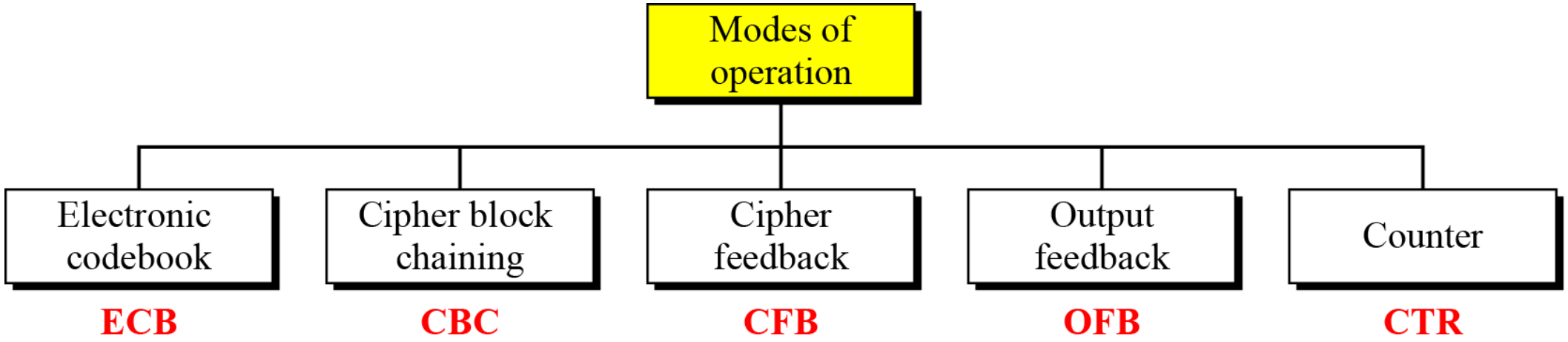
- Setiap blok tidak dienkripsi satu kali, tetapi berulang kali untuk mendapatkan cipherteks yang kuat.

Mode Operasi Cipher Blok

- Mode operasi: berkaitan dengan cara blok dioperasikan sebelum dienkripsi/dekripsi oleh fungsi E dan D.

- Ada 5 mode operasi *cipher* blok:
  1. *Electronic Code Book* (*ECB*)
  2. *Cipher Block Chaining* (*CBC*)
  3. *Cipher Feedback* (*CFB*)
  4. *Output Feedback (OFB)*
  5. *Counter Mode*

- Di dalam kuliah ini hanya dibahas mode ECB, CBC, dan Counter


Modes of operation
Electronic codebook
ECB
Cipher block chaining
CBC
Cipher feedback
CFB
Output feedback
OFB
Counter
CTR

1. Electronic Code Book (ECB)

- Setiap blok plainteks P_i dienkripsi secara individual dan independen dari blok lainnya menjadi blok cipherteks C_i .

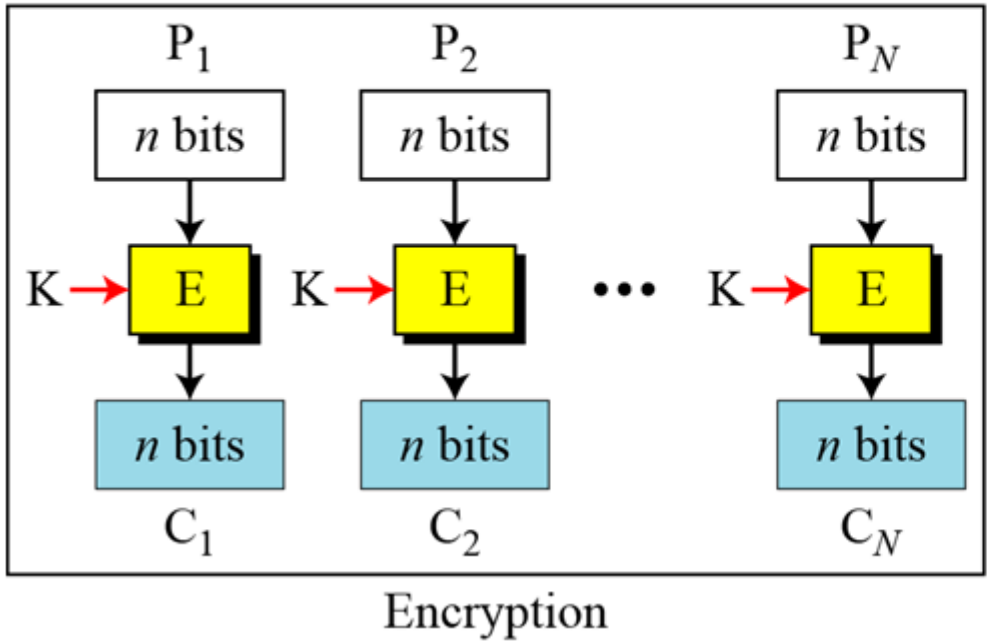
- Enkripsi: $C_i = E_K(P_i)$

  Dekripsi: $P_i = D_K(C_i)$

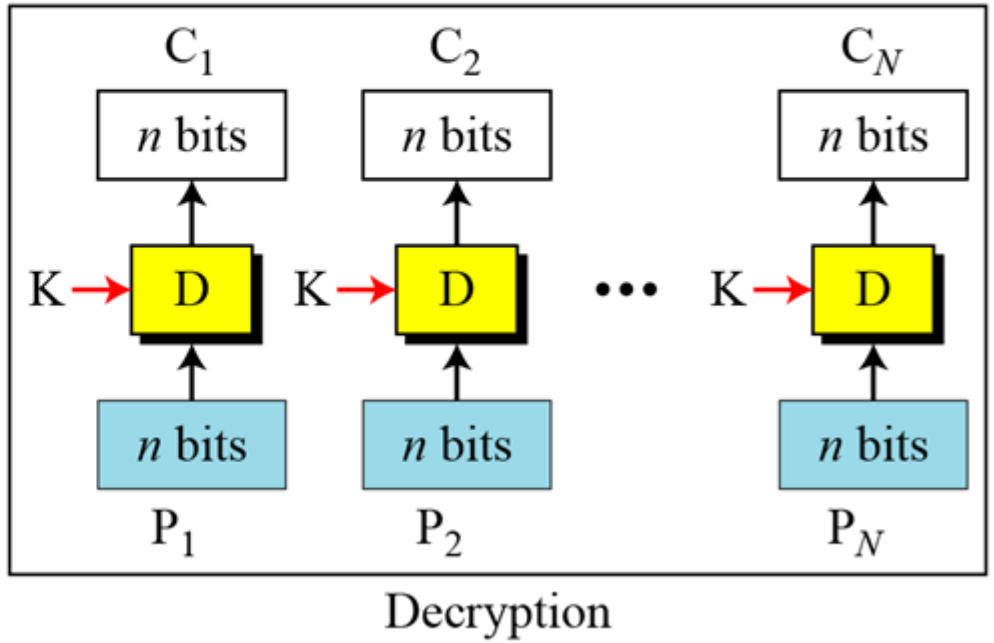
  yang dalam hal ini, P_i dan C_i masing-masing blok plainteks dan cipherteks ke-*i*.

Mode ECB

E: Encryption D: Decryption
P_i: Plaintext block i C_i: Ciphertext block i
K: Secret key



Encryption



Decryption

- Contoh:

  Plainteks: 10100010000111010101001

  Bagi plainteks menjadi blok-blok 4-bit:

  1010 0010 0011 1010 1001

  (dalam notasi HEX :A23A9)

- Kunci (juga 4-bit): 1011

- Misalkan fungsi enkripsi *E* yang sederhana algoritmanya sbb:
  1. XOR-kan blok plainteks P_i dengan *K*
  2. geser secara *wrapping* bit-bit dari hasil langkah 1 satu posisi ke kiri

$$E_K(P) = (P \oplus K) << 1$$

$$E_K(P) = (P \oplus K) << 1$$

Enkripsi:

	1010	0010	0011	1010	1001	
	1011	1011	1011	1011	1011	⊕
Hasil *XOR*:	0001	1001	1000	0001	0010	
Geser 1 bit ke kiri:	0010	0011	0001	0010	0100	
Dalam notasi HEX:	2	3	1	2	4	

Jadi, hasil enkripsi plainteks

`10100010001110101001` (`A23A9` dalam notasi HEX)

adalah

`00100011000100100100` (`23124` dalam notasi HEX)

- Pada mode ECB, blok plainteks yang sama selalu dienkripsi menjadi blok cipherteks yang sama.

	1010	0010	0011	1010	1001	
	1011	1011	1011	1011	1011	⊕
Hasil *XOR*:	0001	1001	1000	0001	0010	
Geser 1 bit ke kiri:	0010	0011	0001	0010	0100	
Dalam notasi HEX:	2	3	1	2	4	

- Pada contoh di atas, blok `1010` muncul dua kali dan selalu dienkripsi menjadi `0010`.

Cipher Block Chaining(CBC)

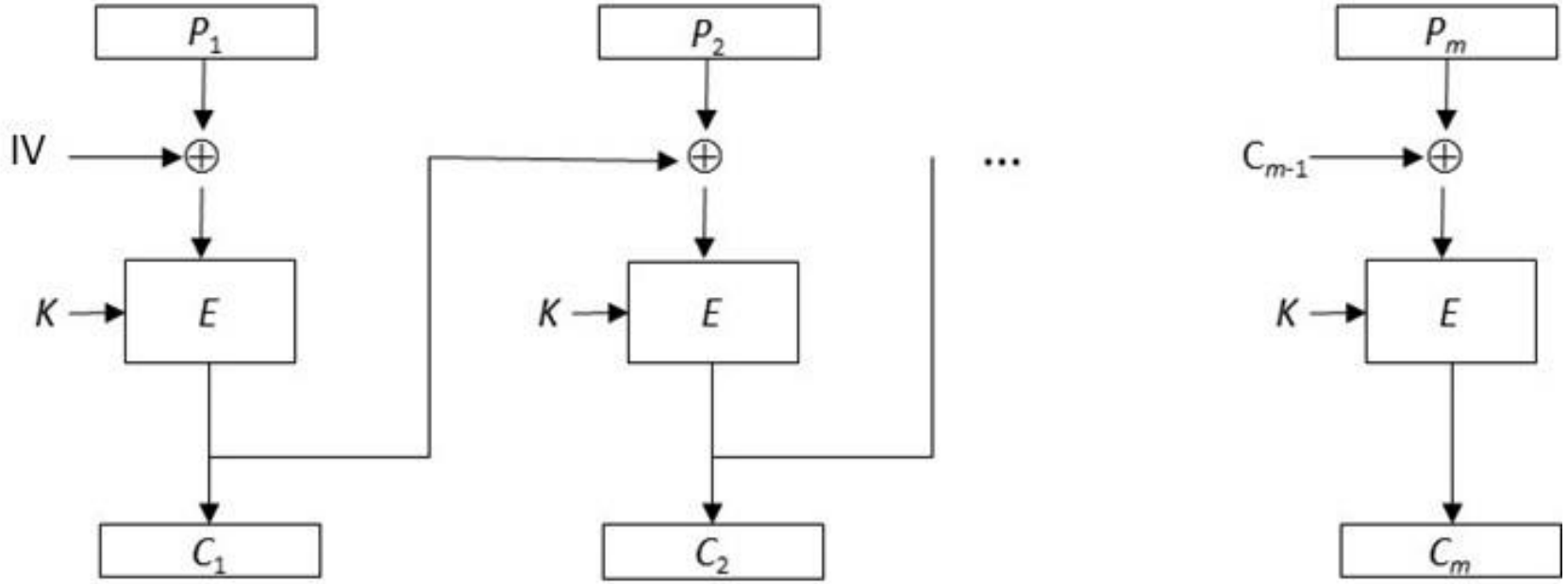
- Tujuan: membuat ketergantungan antar blok (mengatasi kelemahan mode ECB).
- Setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya tetapi juga pada seluruh blok plainteks sebelumnya.
- Hasil enkripsi blok sebelumnya di-umpan-balikkan ke dalam enkripsi blok yang *current*.
- Enkripsi blok pertama memerlukan blok semu (C_0) yang disebut *IV* (*initialization vector*).
- *IV* dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program.
- Pada dekripsi, blok plainteks diperoleh dengan cara meng-*XOR*-kan *IV* dengan hasil dekripsi terhadap blok cipherteks pertama.

(a) Skema enkripsi mode CBC

Encryption:

$C_0 = IV$

$C_i = E_K (P_i \oplus C_{i-1})$

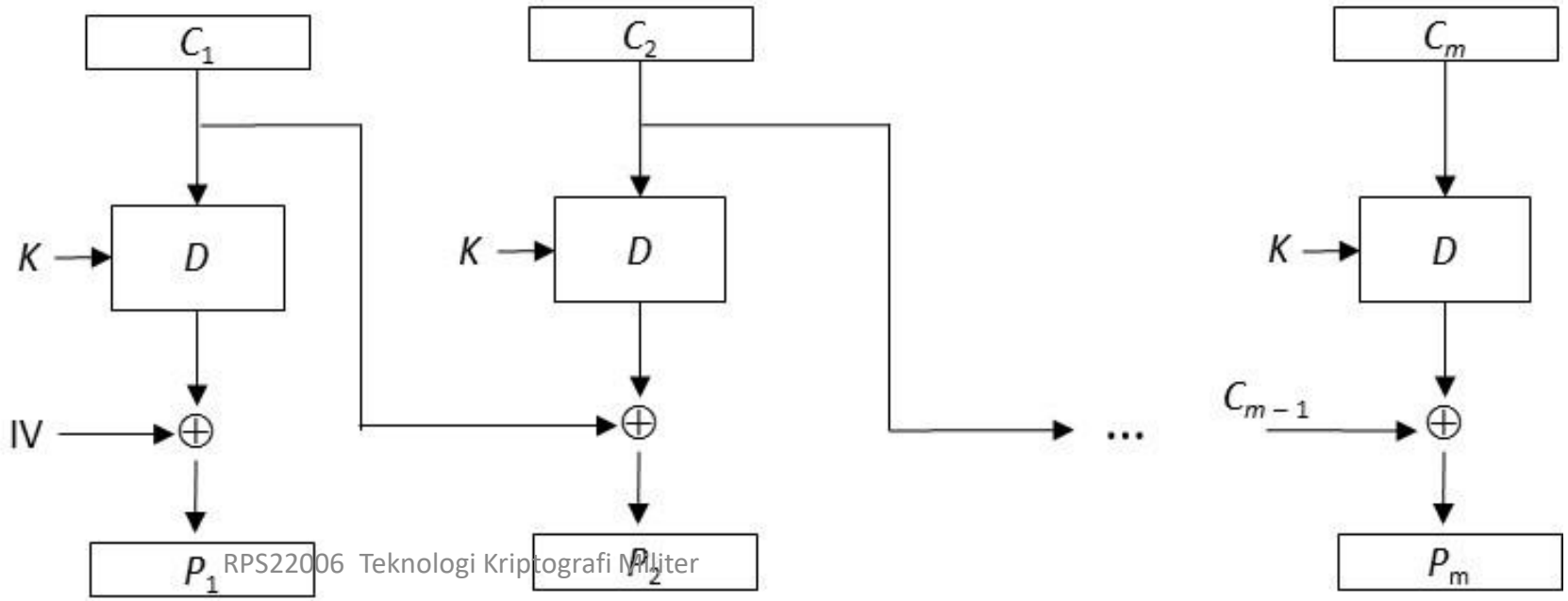


(b) Skema dekripsi mode CBC

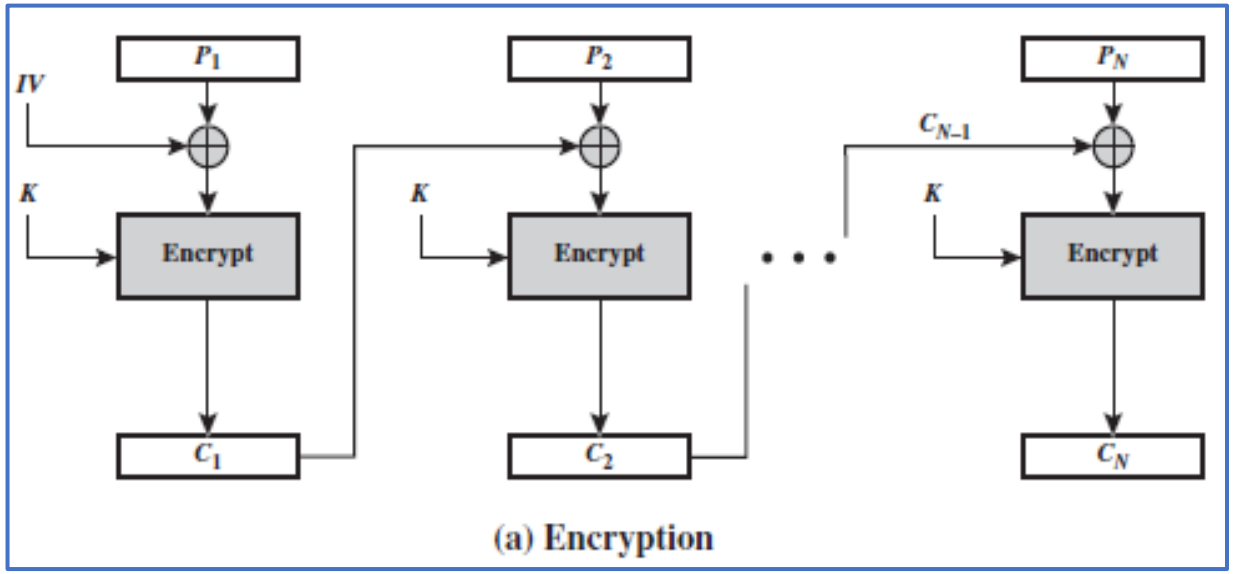
Decryption:

$C_0 = IV$

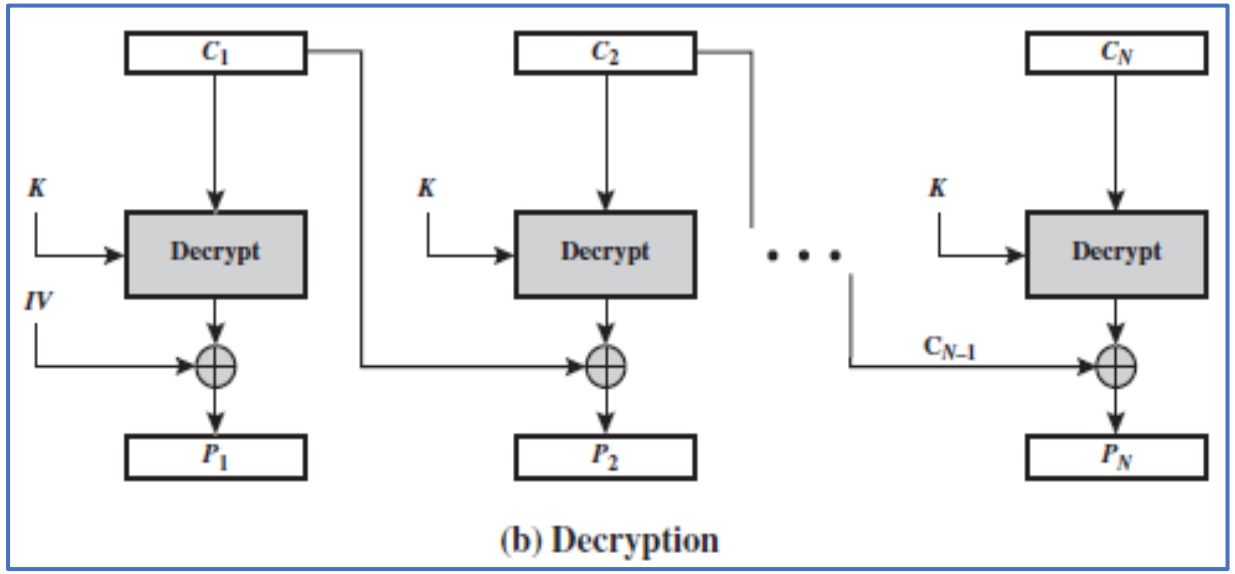
$P_i = D_K (C_i) \oplus C_{i-1}$



Mode CBC



(a) Encryption



(b) Decryption

CBC	$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}])$ $C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}])\ j = 2, \ldots, N$	$P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$ $P_j = \mathrm{D}(K, C_j) \oplus C_{j-1}\ j = 2, \ldots, N$

Contoh 9.8. Tinjau kembali plainteks dari Contoh 9.6:

```
10100010001110101001
```

Bagi plainteks menjadi blok-blok yang berukuran 4 bit:

```
1010  0010  0011  1010  1001
```

atau dalam notasi HEX adalah `A23A9`.

Misalkan kunci (K) yang digunakan adalah (panjangnya juga 4 bit)

```
1011
```

atau dalam notasi HEX adalah `B`. Sedangkan *IV* yang digunakan seluruhnya bit 0 (Jadi, C_0 = 0000)

Fungsi enkripsi E yang digunakan sama seperti sebelumnya: XOR-kan blok plainteks P_i dengan K, kemudian geser secara *wrapping* bit-bit dari $P_i \oplus K$ satu posisi ke kiri.

$$E_K(P) = (P \oplus K) << 1$$

C_1 diperoleh sebagai berikut:

$P_1 \oplus C_0 = 1010 \oplus 0000 = 1010$

Enkripsikan hasil ini dengan fungsi E sbb:

$1010 \oplus K = 1010 \oplus 1011 = 0001$

Geser (*wrapping*) hasil ini satu bit ke kiri: 0010

Jadi, $C_1 = 0010$ (atau 2 dalam HEX)

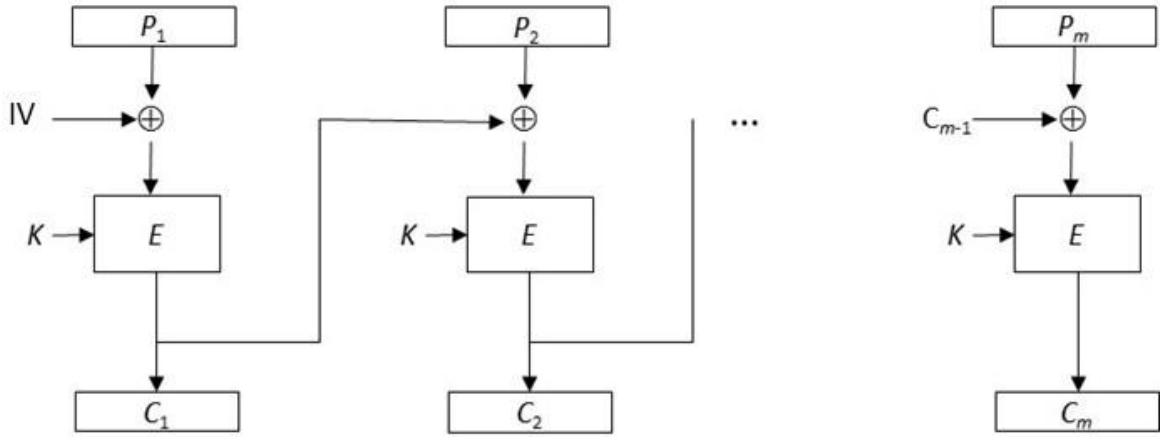
C_2 diperoleh sebagai berikut:

$P_2 \oplus C_1 = 0010 \oplus 0010 = 0000$

$0000 \oplus K = 0000 \oplus 1011 = 1011$

Geser (*wrapping*) hasil ini satu bit ke kiri: 0111

Jadi, $C_2 = 0111$ (atau 7 dalam HEX)



C_3 diperoleh sebagai berikut:

$P_3 \oplus C_2 = 0011 \oplus 0111 = 0100$

$0100 \oplus K = 0100 \oplus 1011 = 1111$

Geser (*wrapping*) hasil ini satu bit ke kiri: 1111

Jadi, $C_3 = 1111$ (atau F dalam HEX)

Demikian seterusnya, sehingga plainteks dan cipherteks hasilnya adalah:

Pesan (plainteks): `A23A9`

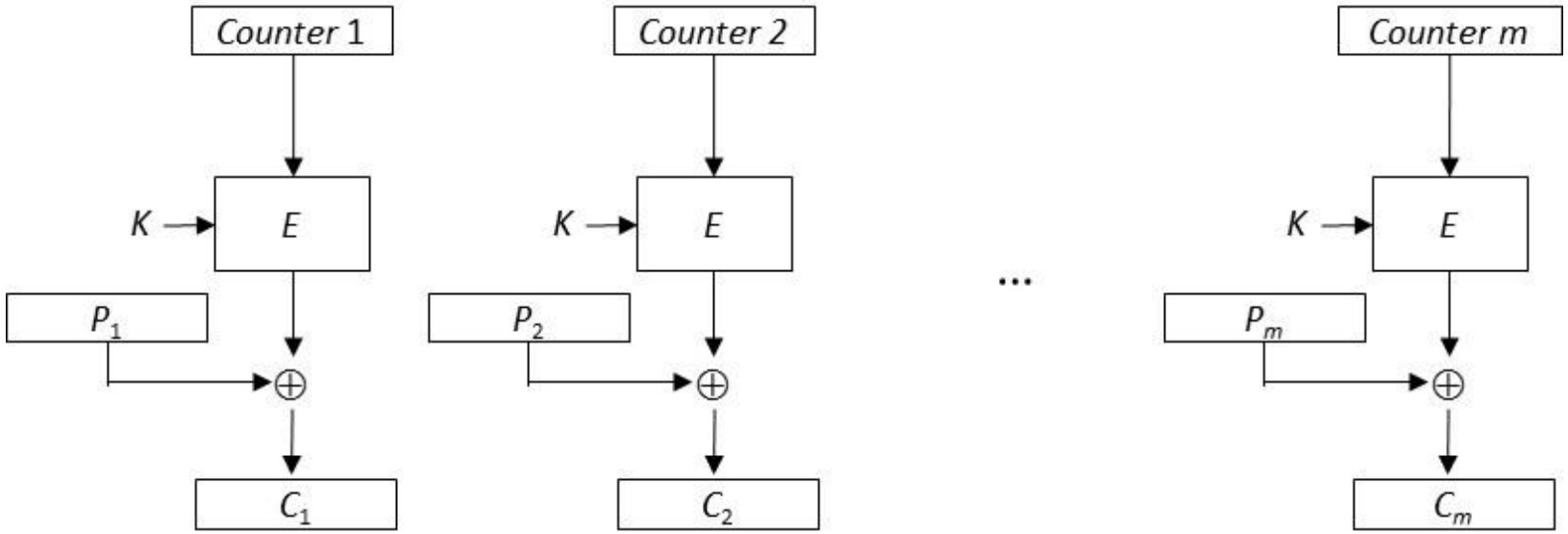
Cipherteks (mode *ECB*): `23124`

Cipherteks (mode *CBC*): `27FBF`

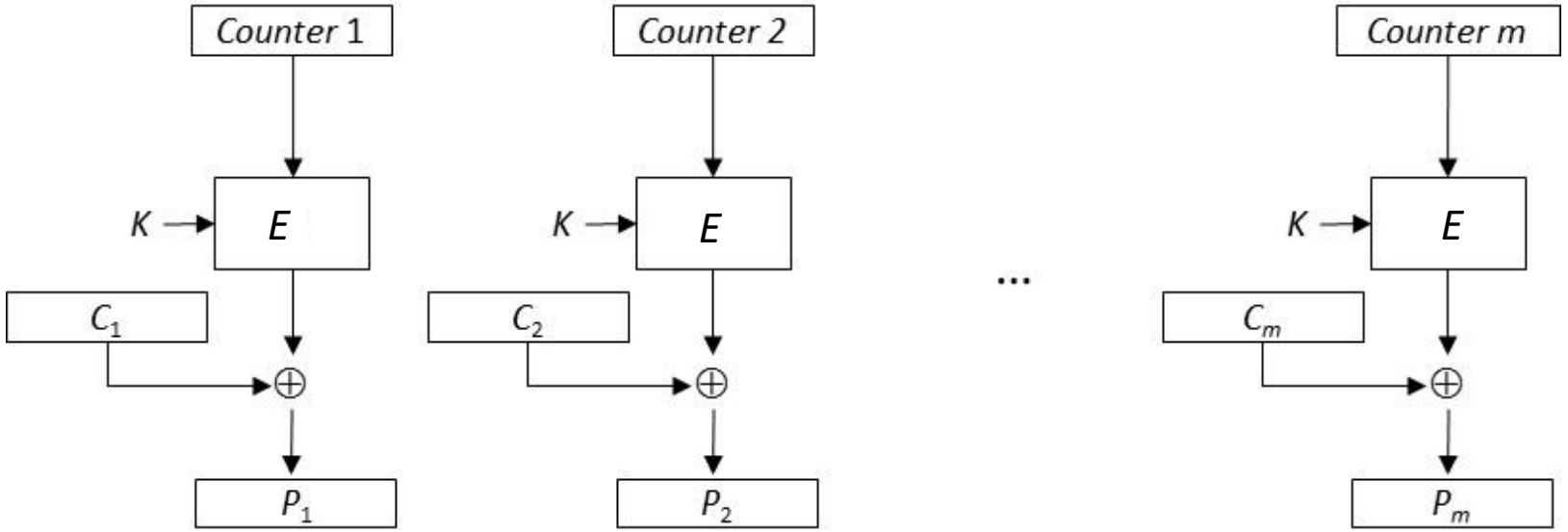
Counter Mode

- Mode *counter* tidak melakukan perantaian (*chaining*) seperti pada *CBC*.
- *Counter* adalah sebuah nilai berupa blok bit yang ukurannya sama dengan ukuran blok plainteks.
- Nilai *counter* harus berbeda dari setiap blok yang dienkripsi. Pada mulanya, untuk enkripsi blok pertama, *counter* diinisialisasi dengan sebuah nilai.
- Selanjutnya, untuk enkripsi blok-blok berikutnya *counter* dinaikkan (*increment*) nilainya satu (counter $\leftarrow$ counter + 1).

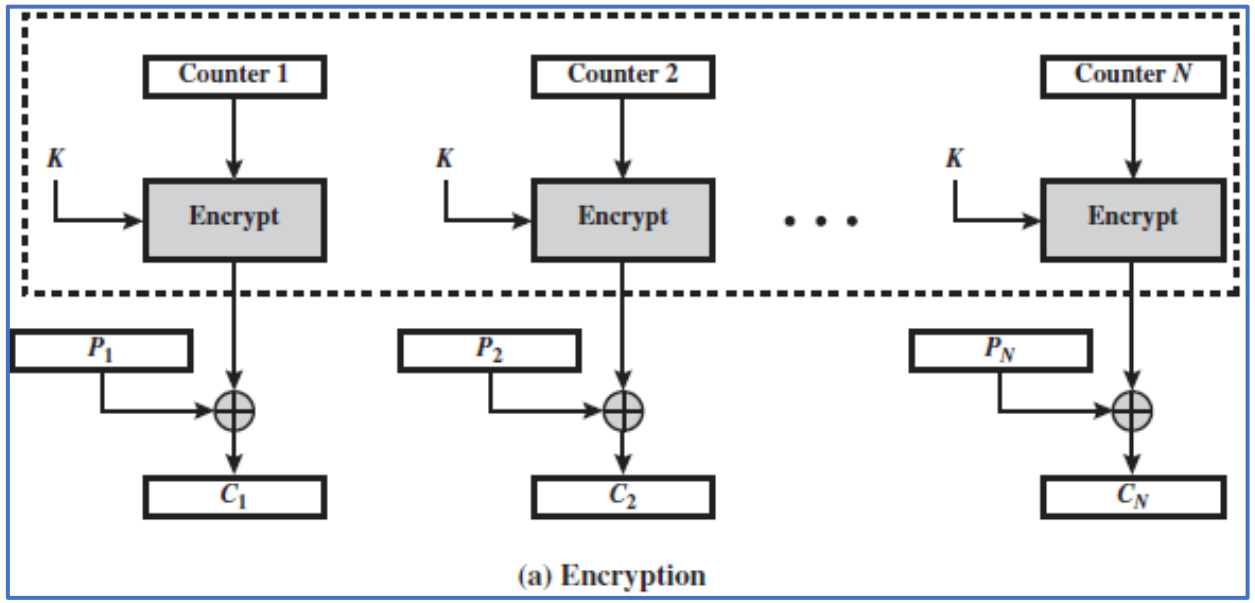
(a) Enkripsi



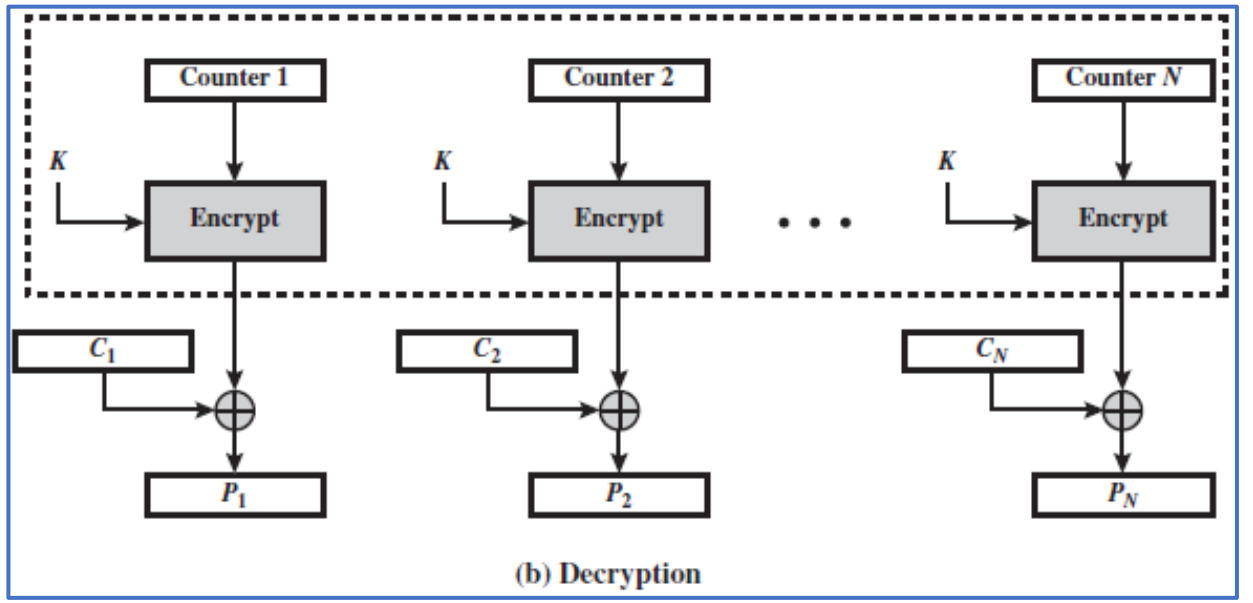
(b) Dekripsi



Mode Counter



(a) Encryption



(b) Decryption

CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$

T_j adalah counter, nilainya bertambah satu setiap kali enkripsi blok

- Contoh: tinjau kembali contoh sebelumnya

  Plainteks: 1010 0010 0011 1010 1001 (A23A9)

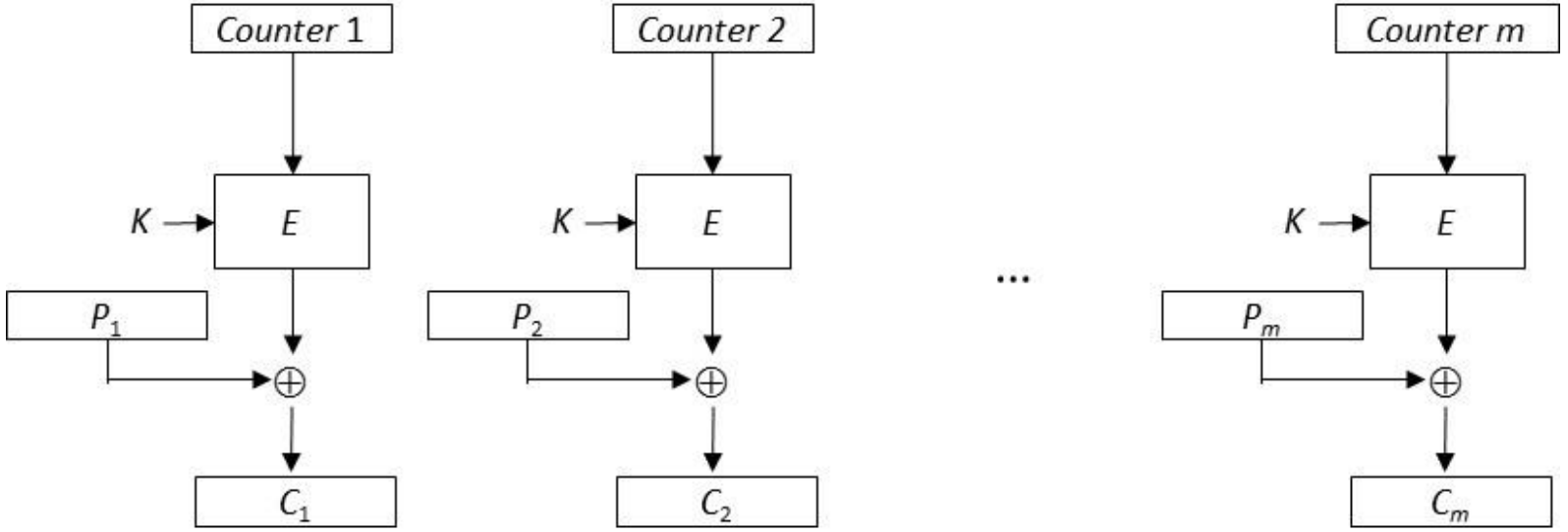
  Kunci: 1011

  Counter: 0000

  Fungsi enkripsi *E* adalah sbb:

  1. XOR-kan blok pesan T_i dengan *K*
  2. geser secara *wrapping* bit-bit dari hasil langkah 1 satu posisi ke kiri

$$E_K(T) = (T \oplus K) << 1$$



- Enkripsi: $E_K(T) = (T \oplus K) << 1$

Counter:	0000	0001	0010	0011	0100
	1011	1011	1011	1011	1011 ⊕
Hasil XOR	1011	1010	1001	1000	1111
Geser 1 bit ke kiri	0111	0101	0011	0001	1111
XOR dengan P	1010	0010	0011	1010	1001 ⊕
Cipherteks:	1101	0111	0000	1011	0111
Cipherteks (Hex):	D	7	0	B	7

- *Daftar block cipher:*

1. Lucifer
2. DES
3. GOST
4. 3DES (Triple-DES)
5. RC2
6. RC5
7. Blowfish
8. AES
9. IDEA
10. LOKI
11. FEAL
12. CAST-128
13. CRAB
14. SAFER
15. Twofish
16. Serpent
17. RC6
18. Camellia
19. 3-WAY
20. MMB
21. Skipjack
22. TEA
23. XTEA
24. SEED
25. Coconut
26. Cobra
27. MARS
28. BATON
29. CRYPTON
30. LEA, dll

v·t·e	Block ciphers (security summary)
Common algorithms	AES · Blowfish · DES (internal mechanics, Triple DES) · Serpent · Twofish
Less common algorithms	ARIA · Camellia · CAST-128 · GOST · IDEA · LEA · RC2 · RC5 · RC6 · SEED · Skipjack · TEA · XTEA
Other algorithms	3-Way · Akelarre · Anubis · BaseKing · BassOmatic · BATON · BEAR and LION · CAST-256 · Chiasmus · CIKS-1 · CIPHERUNICORN-A · CIPHERUNICORN-E · CLEFIA · CMEA · Cobra · COCONUT98 · Crab · Cryptomeria/C2 · CRYPTON · CS-Cipher · DEAL · DES-X · DFC · E2 · FEAL · FEA-M · FROG · G-DES · Grand Cru · Hasty Pudding cipher · Hierocrypt · ICE · IDEA NXT · Intel Cascade Cipher · Iraqi · Kalyna · KASUMI · KeeLoq · KHAZAD · Khufu and Khafre · KN-Cipher · Kuznyechik · Ladder-DES · LOKI (97, 89/91) · Lucifer · M6 · M8 · MacGuffin · Madryga · MAGENTA · MARS · Mercy · MESH · MISTY1 · MMB · MULTI2 · MultiSwap · New Data Seal · NewDES · Nimbus · NOEKEON · NUSH · PRESENT · Prince · Q · REDOC · Red Pike · S-1 · SAFER · SAVILLE · SC2000 · SHACAL · SHARK · Simon · SM4 · Speck · Spectr-H64 · Square · SXAL/MBAL · Threefish · Treyfer · UES · xmx · XXTEA · Zodiac
Design	Feistel network · Key schedule · Lai–Massey scheme · Product cipher · S-box · P-box · SPN · Confusion and diffusion · Avalanche effect · Block size · Key size · Key whitening (Whitening transformation)
Attack (cryptanalysis)	Brute-force (EFF DES cracker) · MITM (Biclique attack · 3-subset MITM attack) · Linear (Piling-up lemma) · Differential (Impossible · Truncated · Higher-order) · Differential-linear · Distinguishing (Known-key) · Integral/Square · Boomerang · Mod *n* · Related-key · Slide · Rotational · Side-channel (Timing · Power-monitoring · Electromagnetic · Acoustic · Differential-fault) · XSL · Interpolation · Partitioning · Rubber-hose · Black-bag · Davies · Rebound · Weak key · Tau · Chi-square · Time/memory/data tradeoff
Standardization	AES process · CRYPTREC · NESSIE
Utilization	Initialization vector · Mode of operation · Padding
v·t·e	Cryptography [show]

Sumber: https://en.wikipedia.org/wiki/Block_cipher

Advanced Encryption Standard (AES)

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 AES

- AES mengenkripsi blok pesan berukuran 128 bit.
- AES mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit (yaitu 128 bit, 160, 192, …, 256 bit).
- 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

- Dengan panjang kunci 128-bit, maka terdapat sebanyak

  $2^{128} = 3{,}4 \times 10^{38}$ kemungkinan kunci.

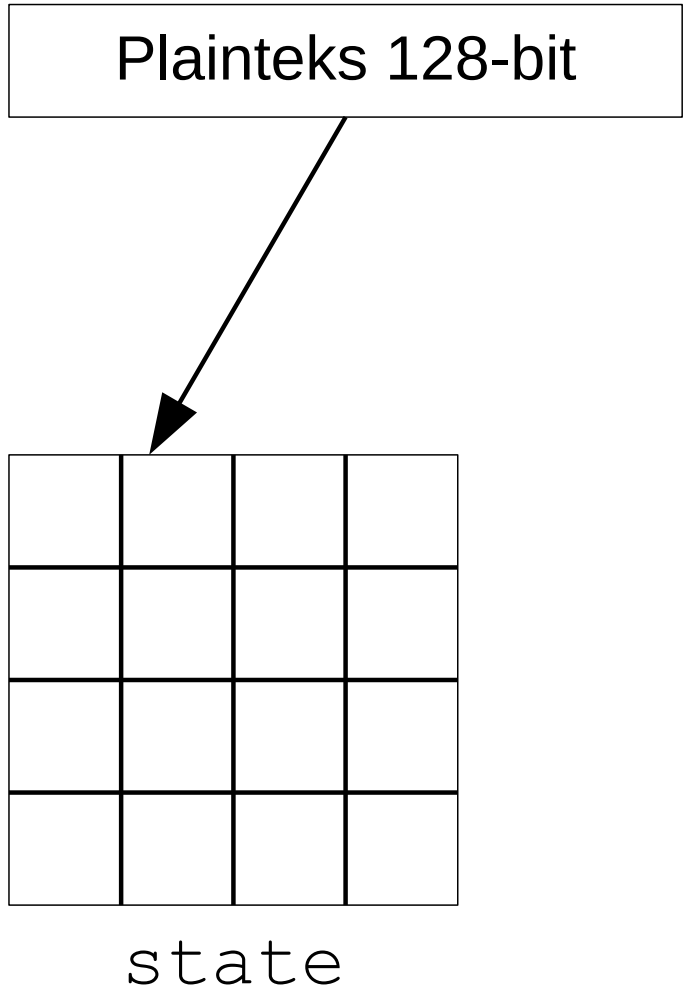
- Jika komputer tercepat dapat mencoba 1 juta kunci setiap detik, maka akan dibutuhkan waktu $5{,}4 \times 10^{24}$ tahun untuk mencoba seluruh kunci.

- Jika komputer tercepat yang dapat mencoba 1 juta kunci setiap milidetik, maka dibutuhkan waktu $5{,}4 \times 10^{18}$ tahun untuk mencoba seluruh kunci.

AES 128-bit

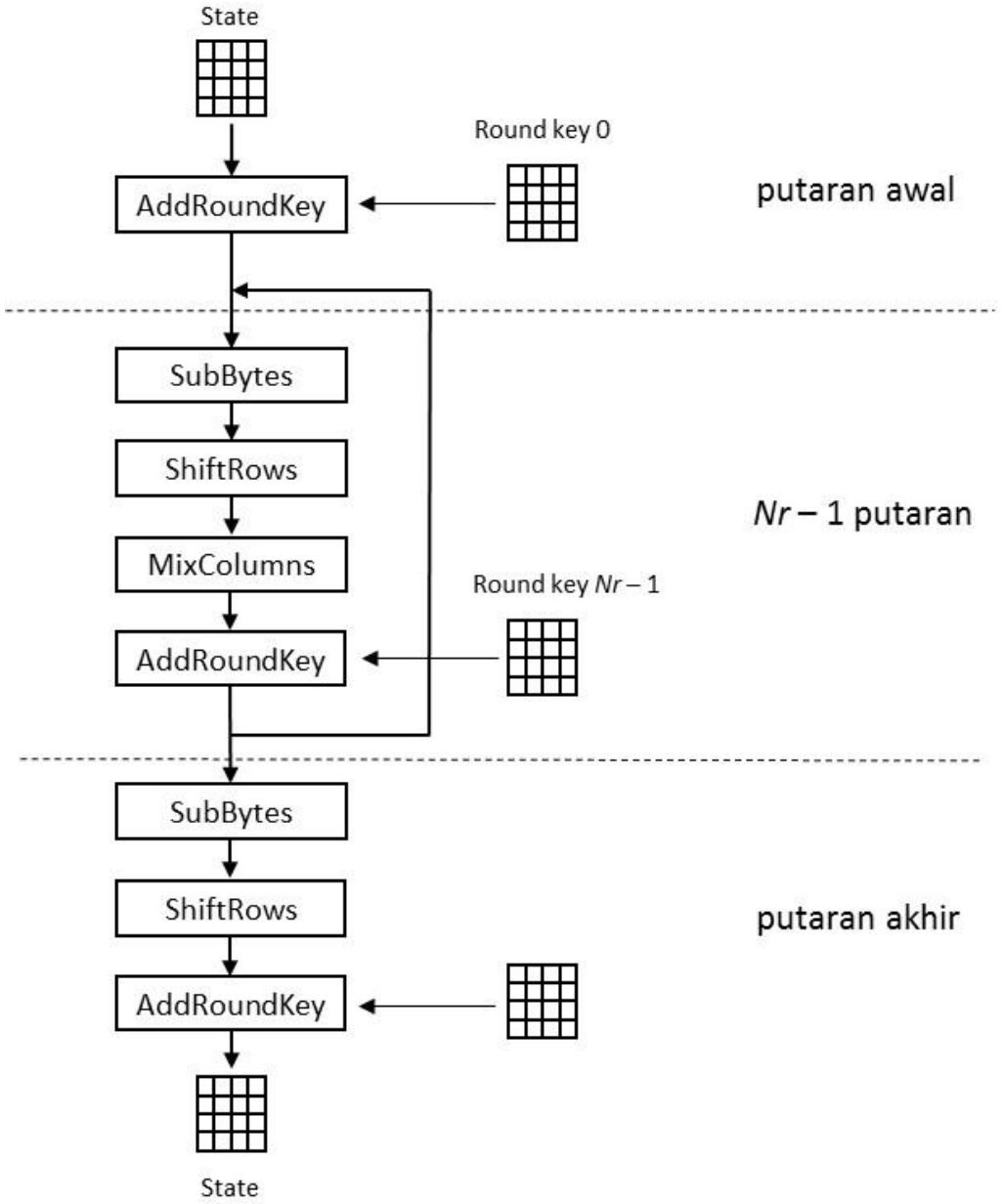
- Yang dijelaskan di sini AES 128-bit
- AES beroperasi dalam *byte*, artinya plainteks, cipherteks, dan kunci diproses dalam byte. Setiap byte direpresentasikan dalam kode Hexa.
- Setiap putaran mengunakan kunci internal yang berbeda (disebut *round key*).
- *Enciphering* melibatkan operasi substitusi dan permutasi.

- Selama kalkulasi plainteks menjadi cipherteks, status data sekarang disimpan di dalam matriks dua dimensi, `state`.
- `state` bertipe *byte* dan berukuran *Nrows* × *Ncols*
- Setiap elemen matriks diisi data sepanang 1 *byte*
- Untuk blok data 128-bit, ukuran `state` adalah 4 × 4, seperti gambar di samping.



Contoh elemen state dalam notasi HEX:

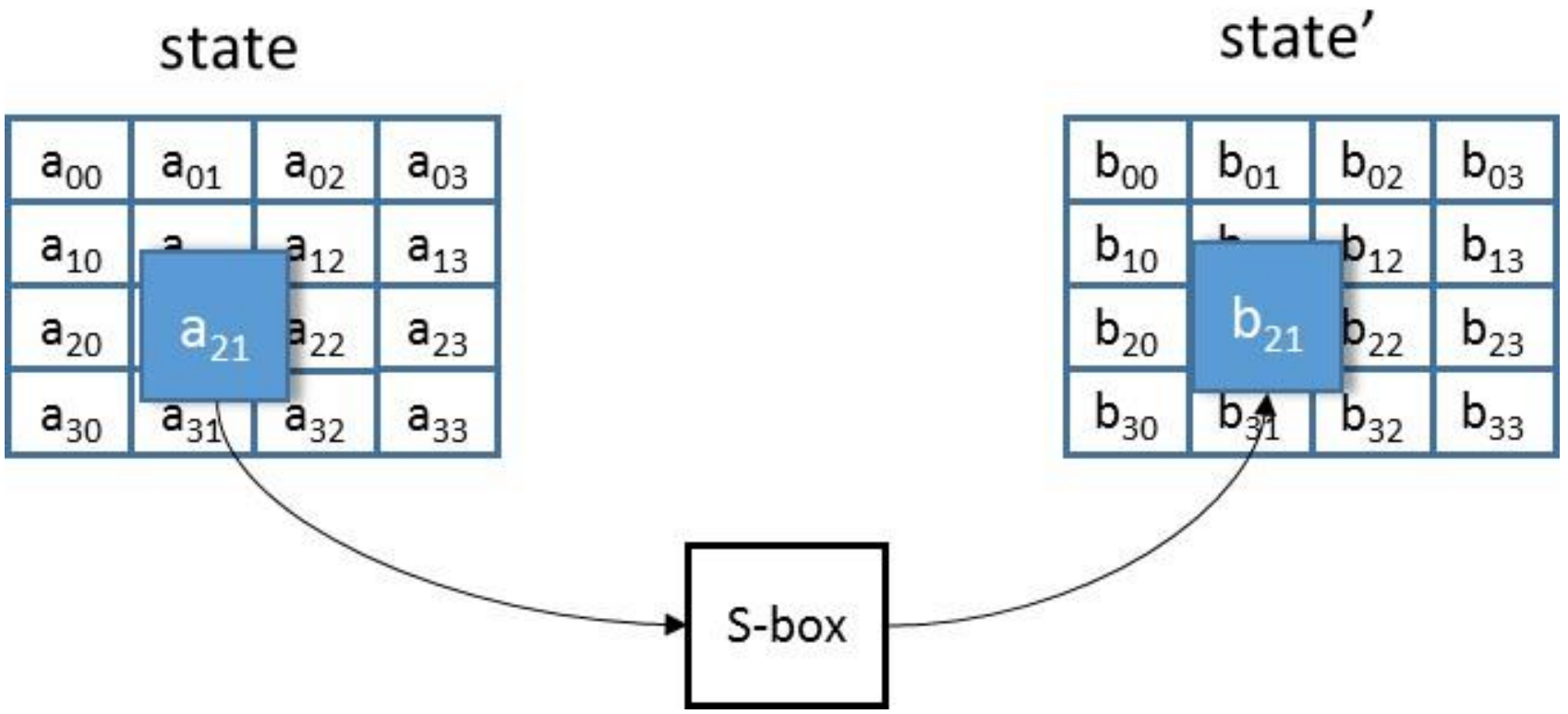
23	A2	BC	4A
D4	03	97	F3
16	48	CD	50
FF	DA	10	64


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

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



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

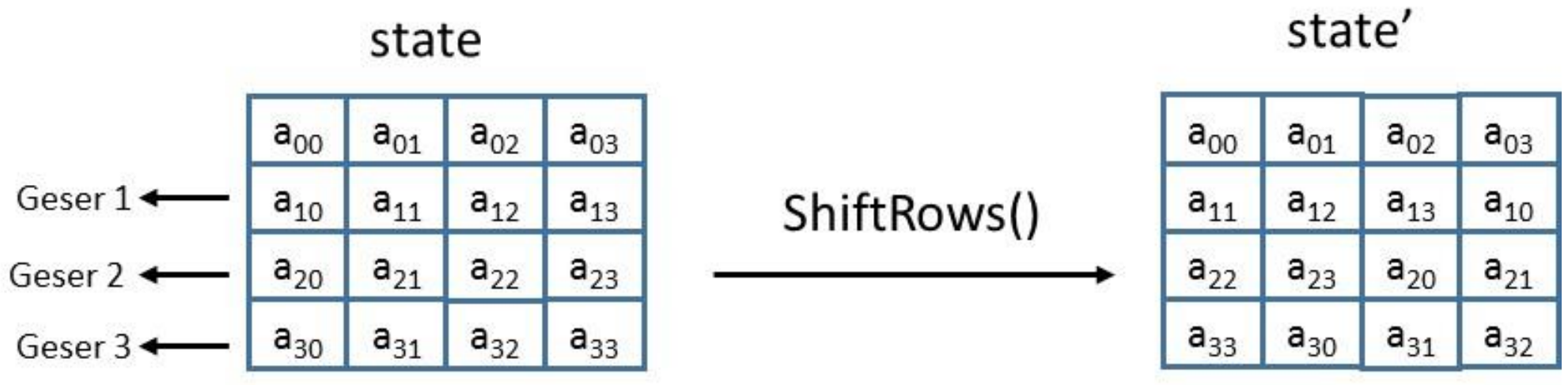
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	**26**	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20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



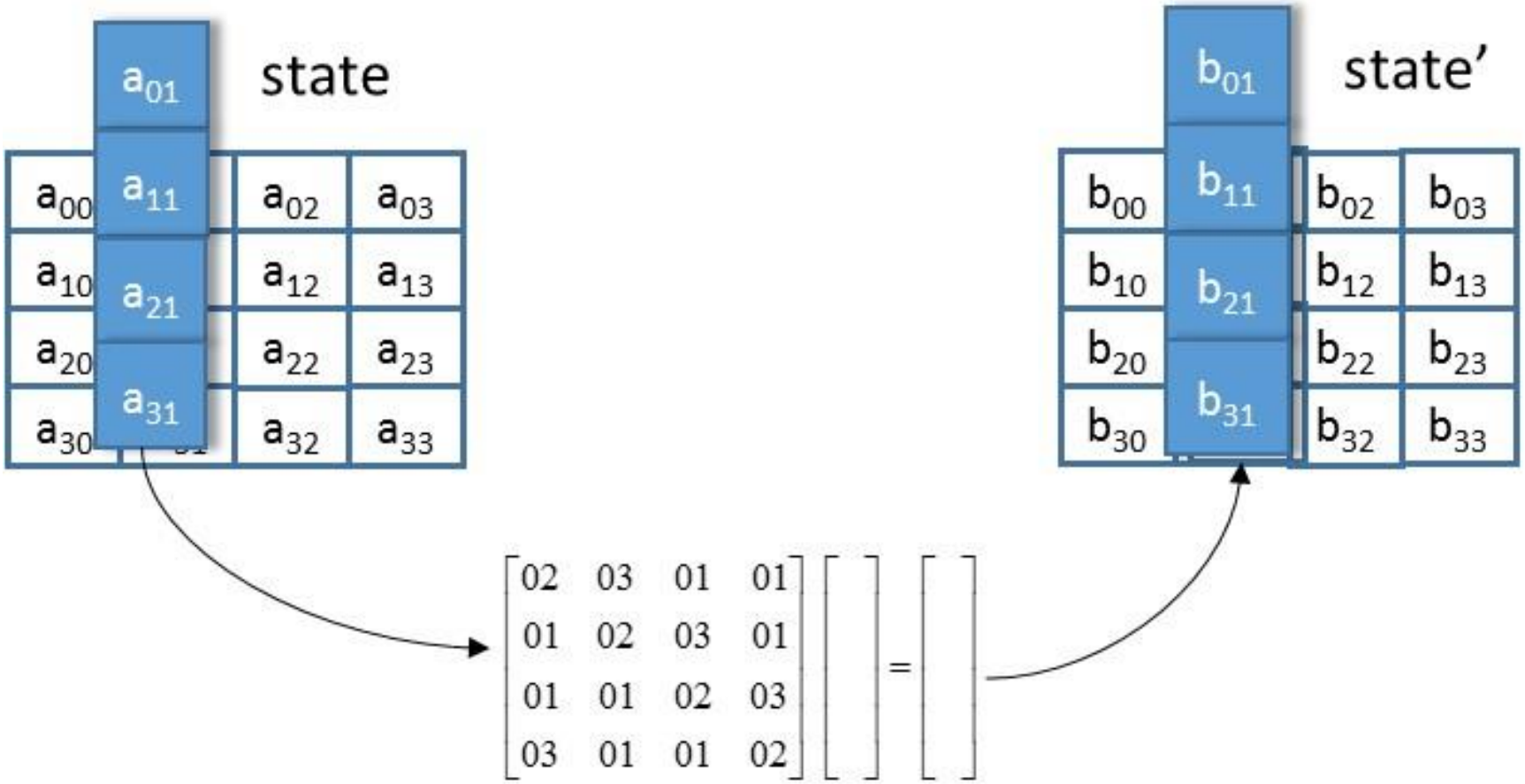
26	3A	65	D6
48	7B	88	0D
47	52	BD	53
16	57	CA	43

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:

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


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

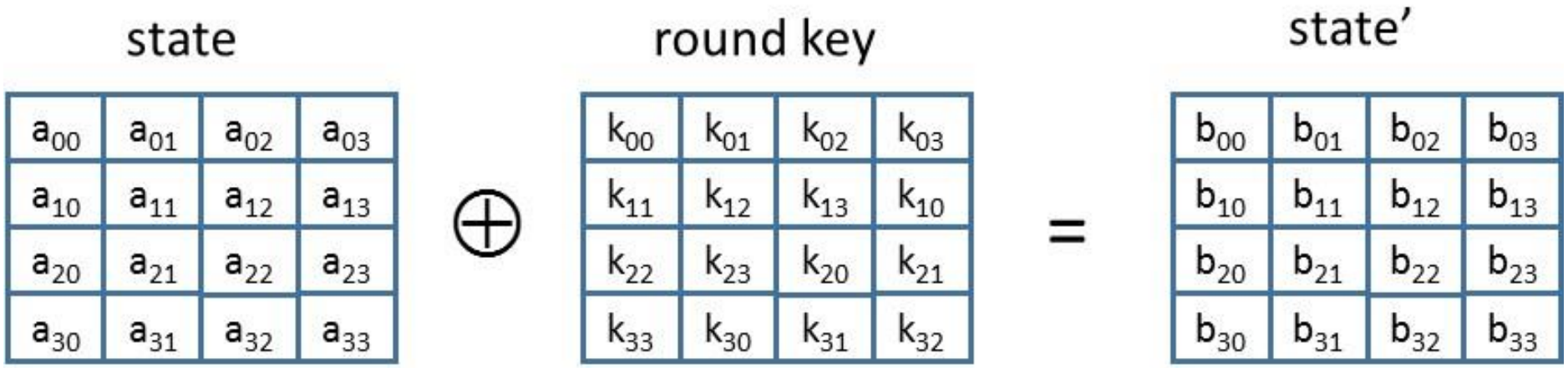
Contoh:

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

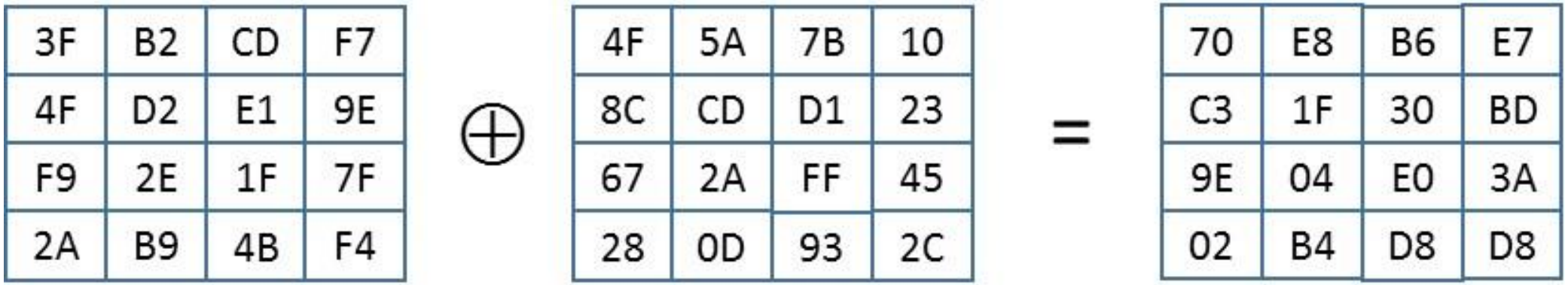
$(02 \bullet 26) \oplus (03 \bullet 7B) \oplus (01 \bullet BD) \oplus (01 \bullet 43) = 3F$
$(01 \bullet 26) \oplus (02 \bullet 7B) \oplus (03 \bullet BD) \oplus (01 \bullet 43) = 4F$
$(01 \bullet 26) \oplus (01 \bullet 7B) \oplus (02 \bullet BD) \oplus (03 \bullet 43) = F9$
$(03 \bullet 26) \oplus (01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

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

$\oplus$

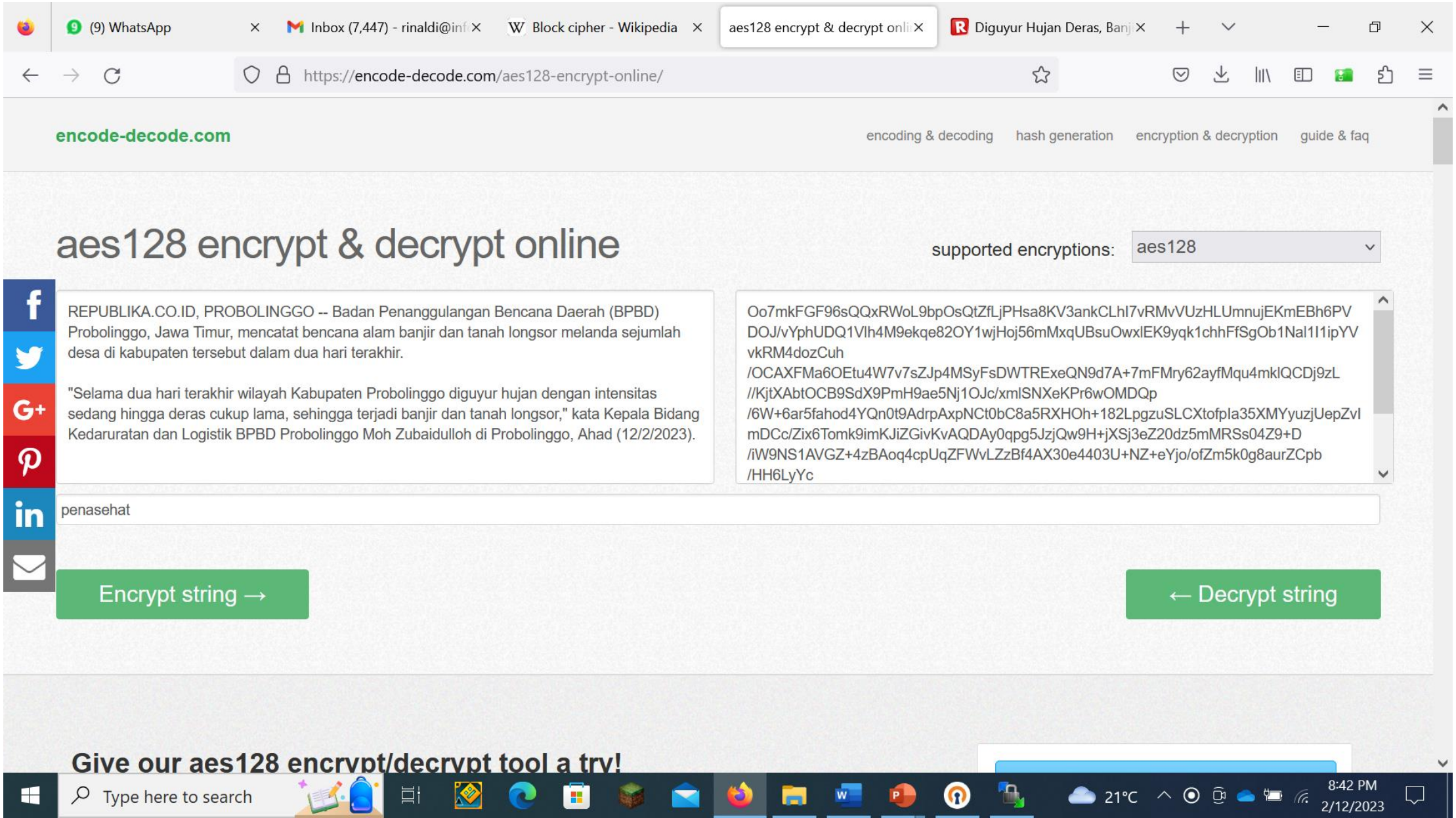
4F	5A	7B	10
8C	CD	D1	23
67	2A	FF	45
28	0D	93	2C

$=$

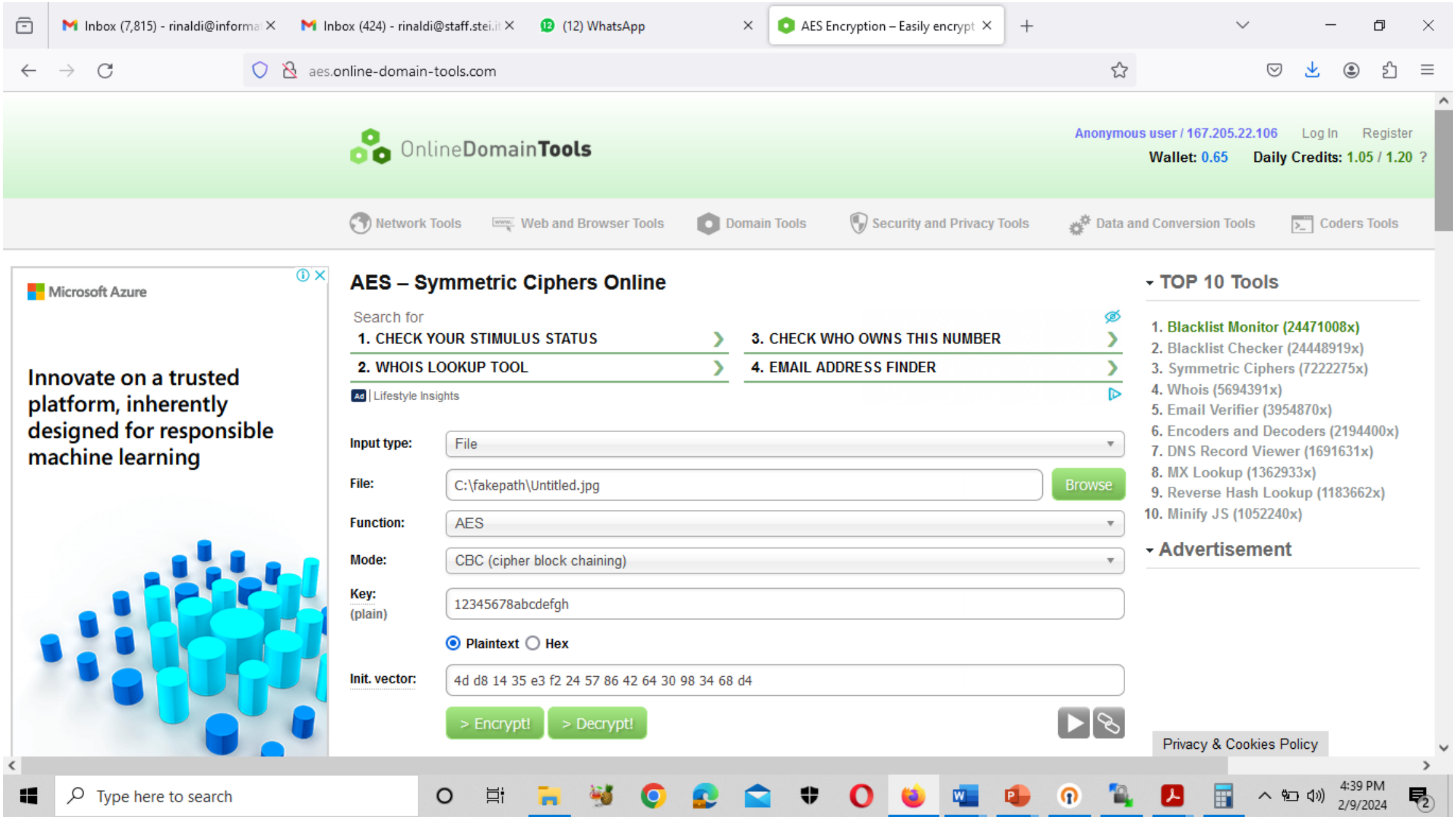
70	E8	B6	E7
C3	1F	30	BD
9E	04	E0	3A
02	B4	D8	D8

Demo online AES (1)

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



Demo online AES (2) http://aes.online-domain-tools.com/



Program enkripsi-dekripsi AES dengan Python

```
#import library crypto dan base64
from Crypto.Cipher import AES
import base64
```

```
def enkripsi (pesan):
    print("Plainteks: \n", pesan)
    print("Ketikkan kunci (16 karakter):")
    kunci = input()

    cipher = AES.new (kunci,AES.MODE_ECB)    #defenisikan AES mode

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

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

```
def dekripsi(pesan):

    print("Cipherteks: \n", pesan)

    print("Ketikkan kunci (16 karakter):")
    kunci = input()

    cipher = AES.new(kunci, AES.MODE_ECB)

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

    print("\n")

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

```
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()
       print("Dekripsi pesan...")
       dekripsi(pesan)
    else:
       print("Pilihan salah")
```

Run program:

```
In [15]: ProgramEnkripsiDekripsiAES()
```

```
-- Enkripsi dan dekripsi pesan dengan AES --
1: Enkripsi pesan
2: Dekripsi pesan
1
Ketikkan pesan yang akan dienkripsi:
Saya besok akan terbang ke Makassar, harap dijemput di bandara ya
Enkripsi pesan...
Plainteks:
 Saya besok akan terbang ke Makassar, harap dijemput di bandara ya
Ketikkan kunci (16 karakter):
abcdefghij123456


Cipherteks:
 b'RdeE0YUnm6x/pbaciLm2aT1X4r1gm0HgsLTXRn/NrGrhuC9RksRtxAs7/HRKD8z2hjIGlQezTQ6/hMU7Mdjf2573FLkbjoTTzQ
xro9ywj6g='
```

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

```
-- Enkripsi dan dekripsi pesan dengan AES --
1: Enkripsi pesan
2: Dekripsi pesan
2
Ketikkan pesan yang akan didekripsi:
RdeE0YUnm6x/pbaciLm2aT1X4r1gm0HgsLTXRn/NrGrhuC9RksRtxAs7/HRKD8z2hjIGlQezTQ6/hMU7Mdjf2573FLkbjoTTzQxro
9ywj6g=
Dekripsi pesan...
Cipherteks:
 RdeE0YUnm6x/pbaciLm2aT1X4r1gm0HgsLTXRn/NrGrhuC9RksRtxAs7/HRKD8z2hjIGlQezTQ6/hMU7Mdjf2573FLkbjoTTzQxr
o9ywj6g=
Ketikkan kunci (16 karakter):
abcdefghij123456


Pesan setelah didekripsi adalah:
 b'Saya besok akan terbang ke Makassar, harap dijemput di bandara ya                '
```

Data Encryption Standard (DES)

(Optional)

Sejarah DES

- Pada tahun 1972 NIST meminta standard enkripsi *cipher* blok.
- Pada tahun 1974 Horst Feistel di IBM mendesain cipher blok Lucifer (panjang kunci 128bit, Panjang blok 128 bit)
- Lucifer kemudian berkembang menjadi DES dengan beberapa masukan dari NSSA (*National Security Agency*):

  - panjang kunci 56 bit, ukuran blok 64 bit

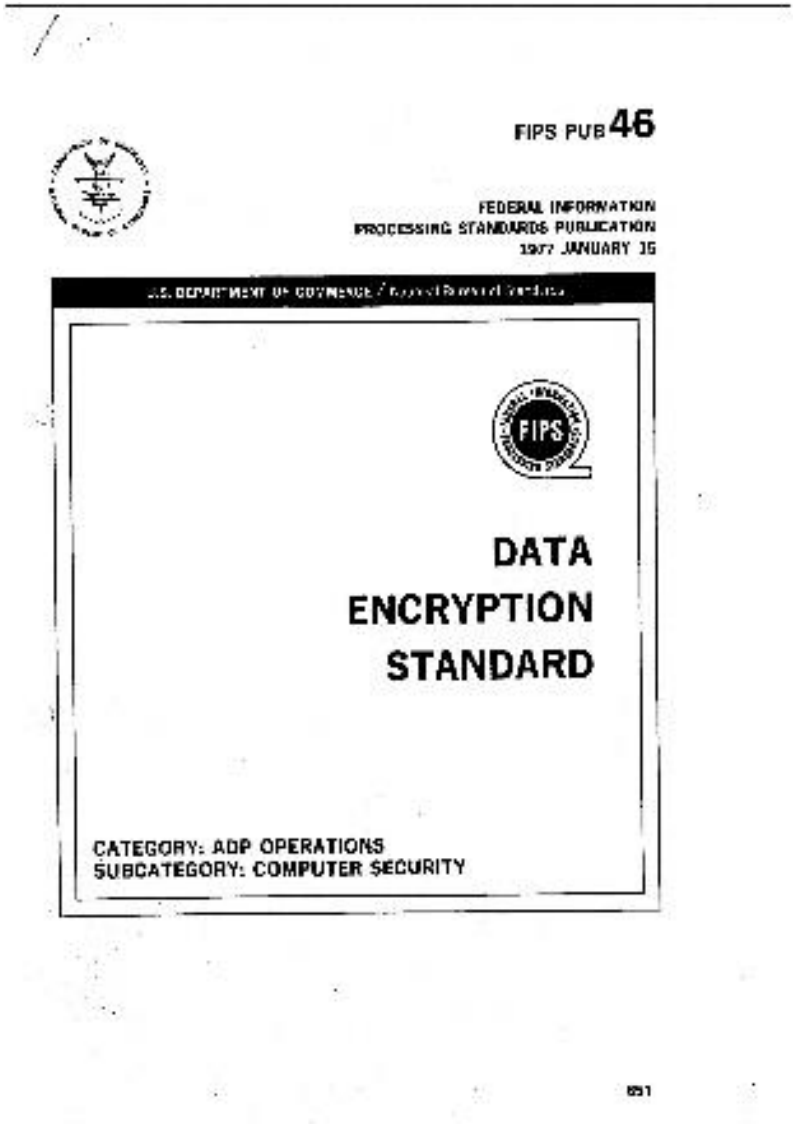
  - 16 putaran.
- Tahun 1976 DES disetujui oleh *National Bureau of Standard* (*NBS*) setelah penilaian kekuatannya oleh *NSA*.
- Sejak itu DES diimplementasikan secara luas, baik *software* maupun *hardware*.
- Tahun 1997-1998 DES berhasil dipecahkan dengan *brute force attack*
- Tahun 2001 DES diganti oleh AES

- Catat bahwa DES adalah sebah standard, sedangkan algoritmanya sendiri bernama DEA (*Data Encryption Algorithm*). Kedua nama ini sering dikacaukan.

- DES termasuk ke dalam algoritma kriptografi kunci-simetri dan tergolong ke dalam *cipher* blok.

- DES beroperasi pada ukuran blok plainteks 64 bit.

- Panjang kunci ekternal = 64 bit (sesuai ukuran blok), tetapi hanya 56 bit yang dipakai (8 bit *parity* tidak digunakan). Bit parity = bit ke-8, 16, 24, dst.


FIPS PUB 46

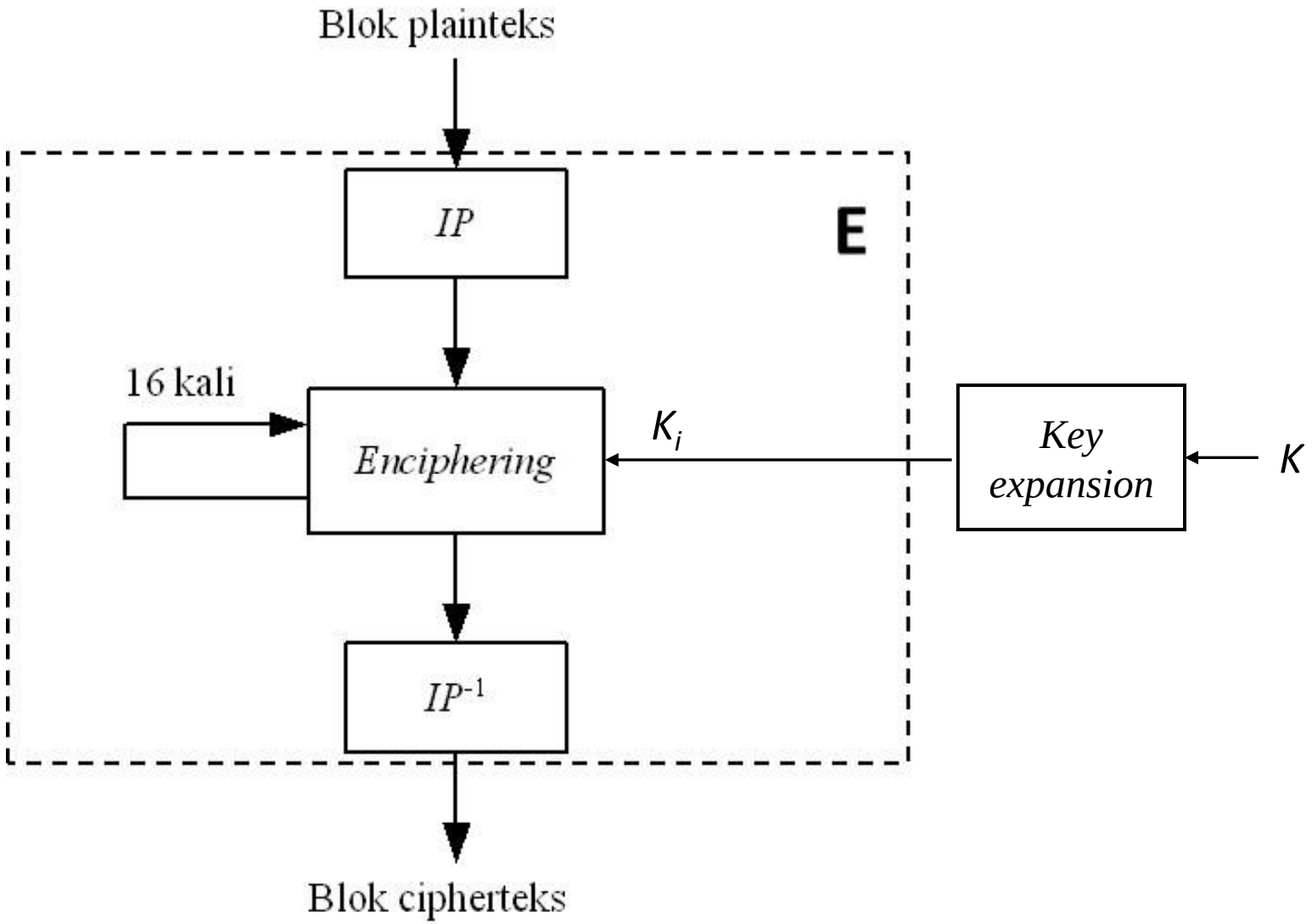
FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION 1977 JANUARY 15

FIPS

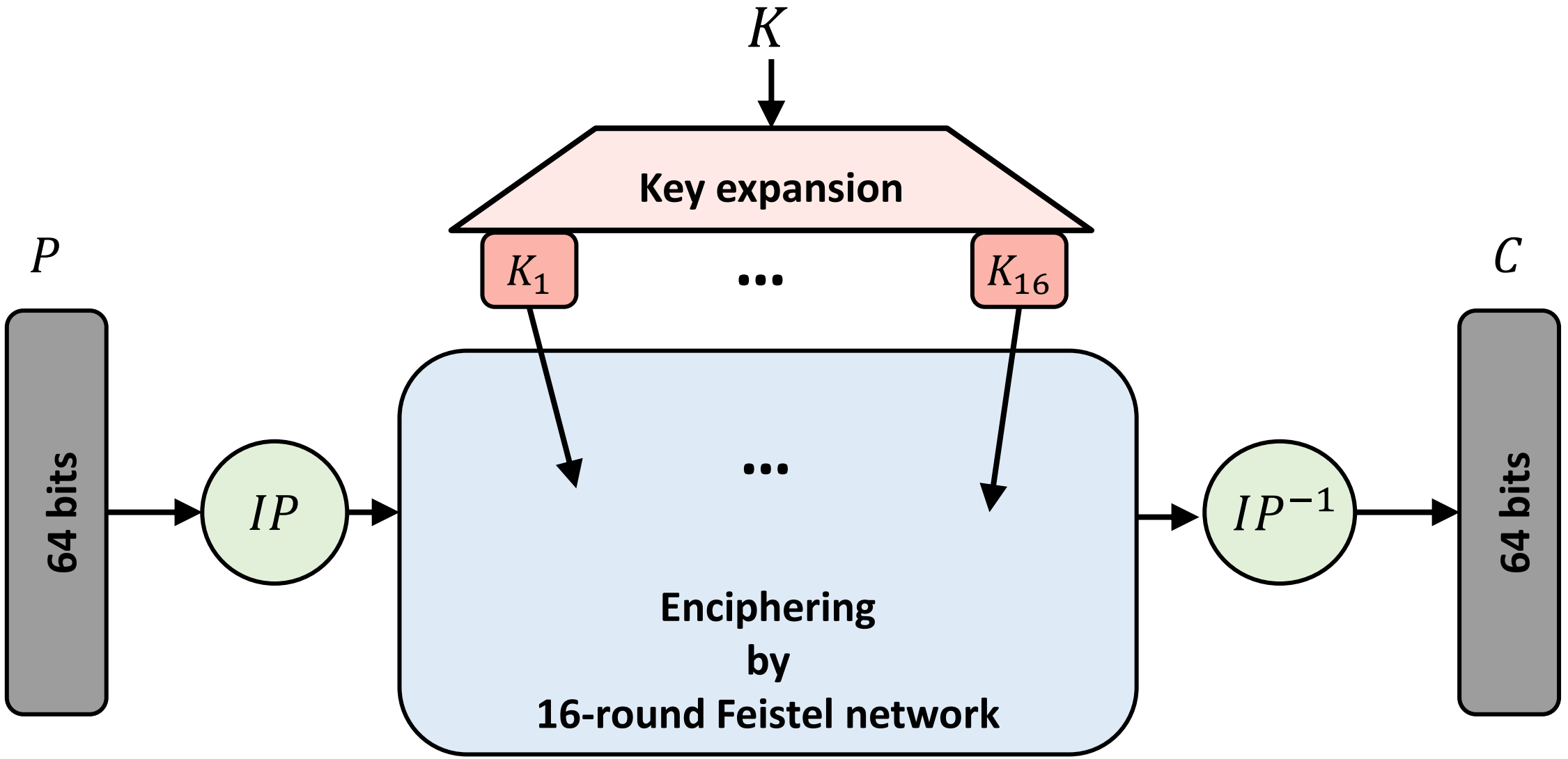
DATA ENCRYPTION STANDARD

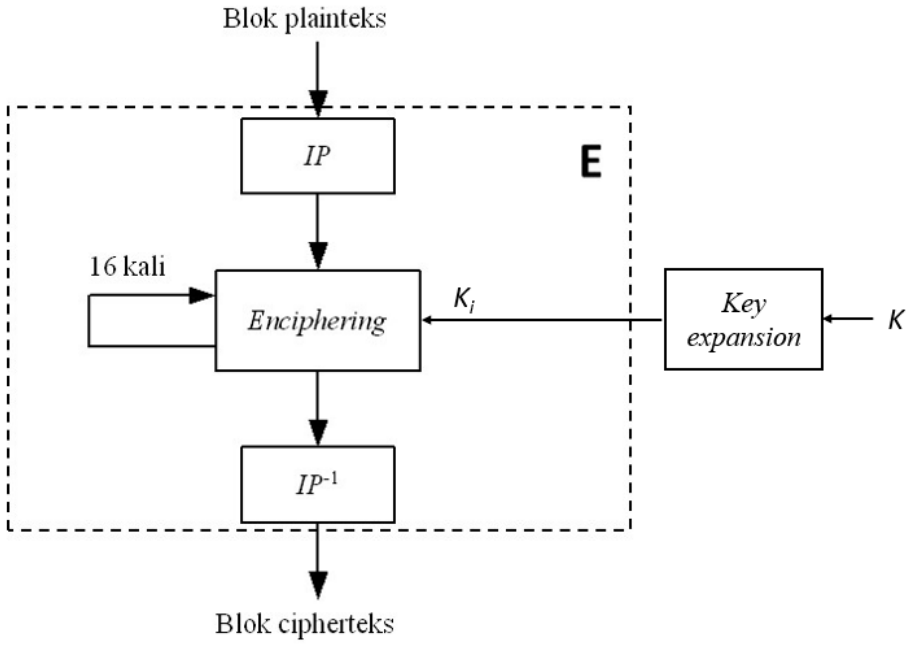
CATEGORY: ADP OPERATIONS
SUBCATEGORY: COMPUTER SECURITY

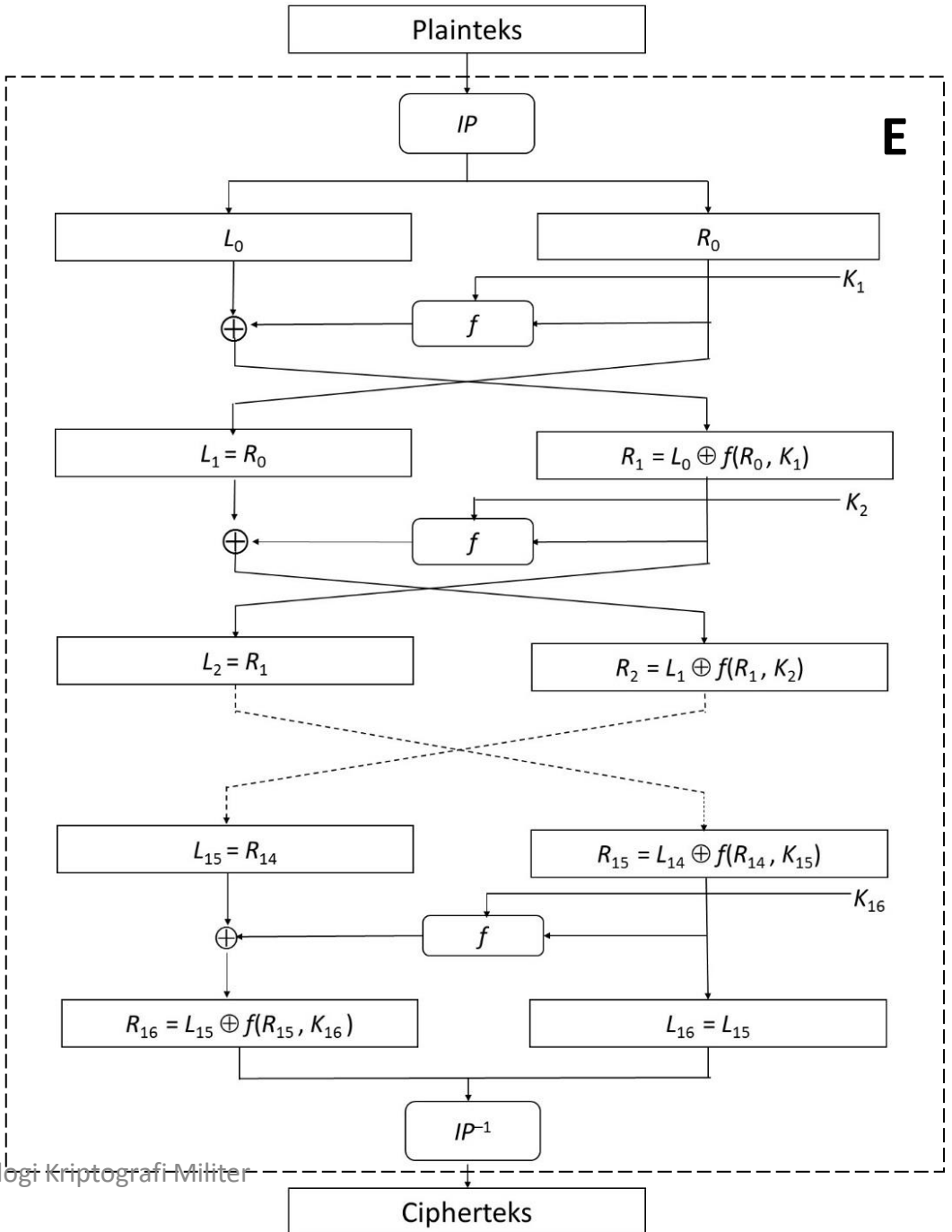
- Setiap blok plainteks dienkripsi dalam 16 putaran *enciphering*.
- Setiap putaran menggunakan kunci internal (kunci putaran) berbeda.
- Kunci internal sepanjang 48-bit dibangkitkan dari kunci eksternal
- Setiap blok plainteks mengalami permutasi awal (*IP*), 16 putaran *enciphering*, dan inversi permutasi awal (IP^{-1}). (lihat Gambar 1)



Gambar 1 Skema global algoritma DES


K
Key expansion
K_1
...
K_{16}
P
C
64 bits
IP
...
IP^{-1}
64 bits
Enciphering
by
16-round Feistel network

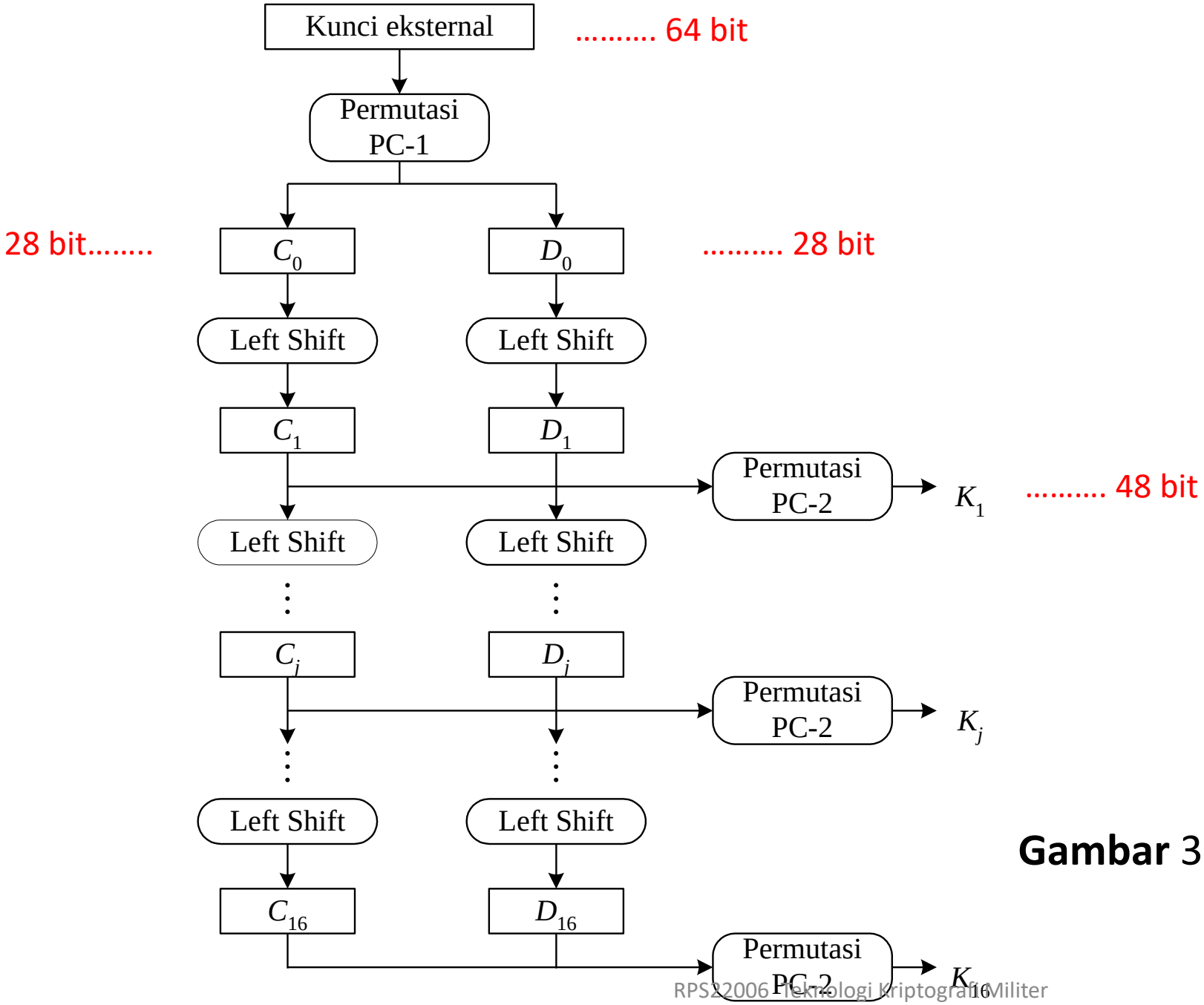




Gambar 2. Algoritma Enkripsi dengan DES

Pembangkitan Kunci Internal (*key expansion*)

- Kunci internal = kunci setiap putaran = *subkey* (*round key*)
- Ada 16 putaran, jadi ada 16 kunci internal: K_1, K_2, …, K_{16}
- Kunci internal dibangkitkan dari kunci eksternal (64 bit) yang diberikan oleh pengguna. Di dalam dokumen resmi DES pembangkitan kunci internal disebut *key schedule*.
- Panjang kunci internal adalah 48 bit
- Gambar 3 memperlihatkan proses pembangkitan kunci internal.



Gambar 3. Pembangkitan kunci internal

Perhatikan, bit-bit *parity*, yaitu bit ke-8, 16, 28, … dibuang, sehingga menjadi 56 bit

Matriks permutasi kompresi PC-1:

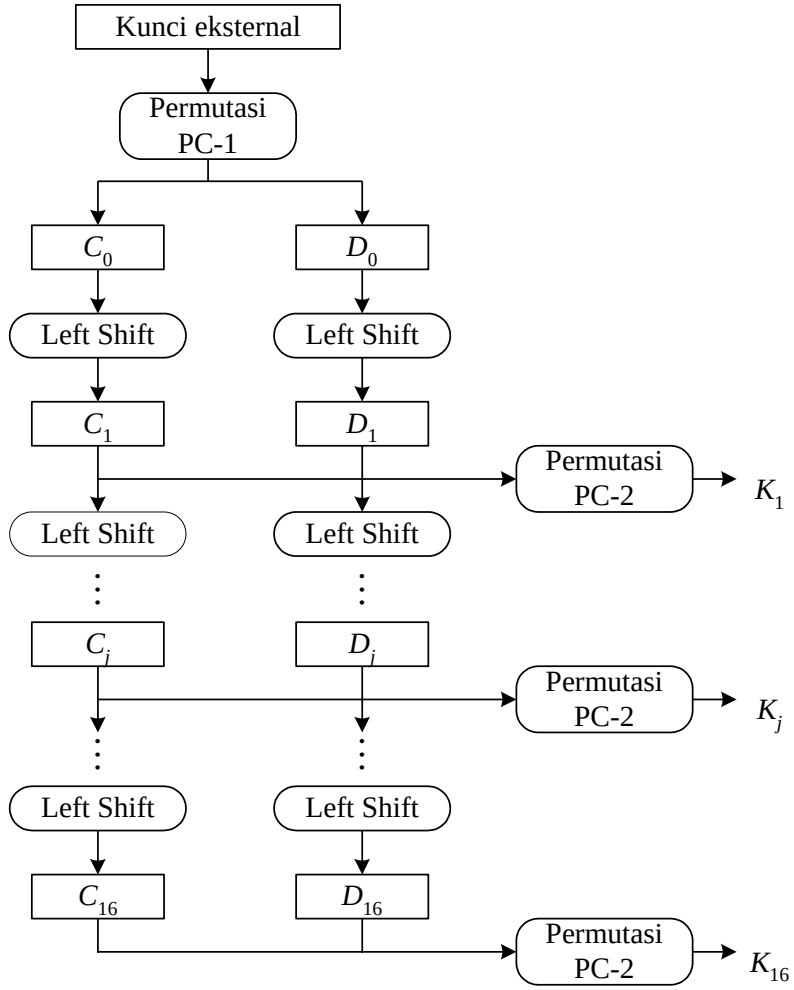
57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

C_0: berisi bit-bit dari K pada posisi

57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18
10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36

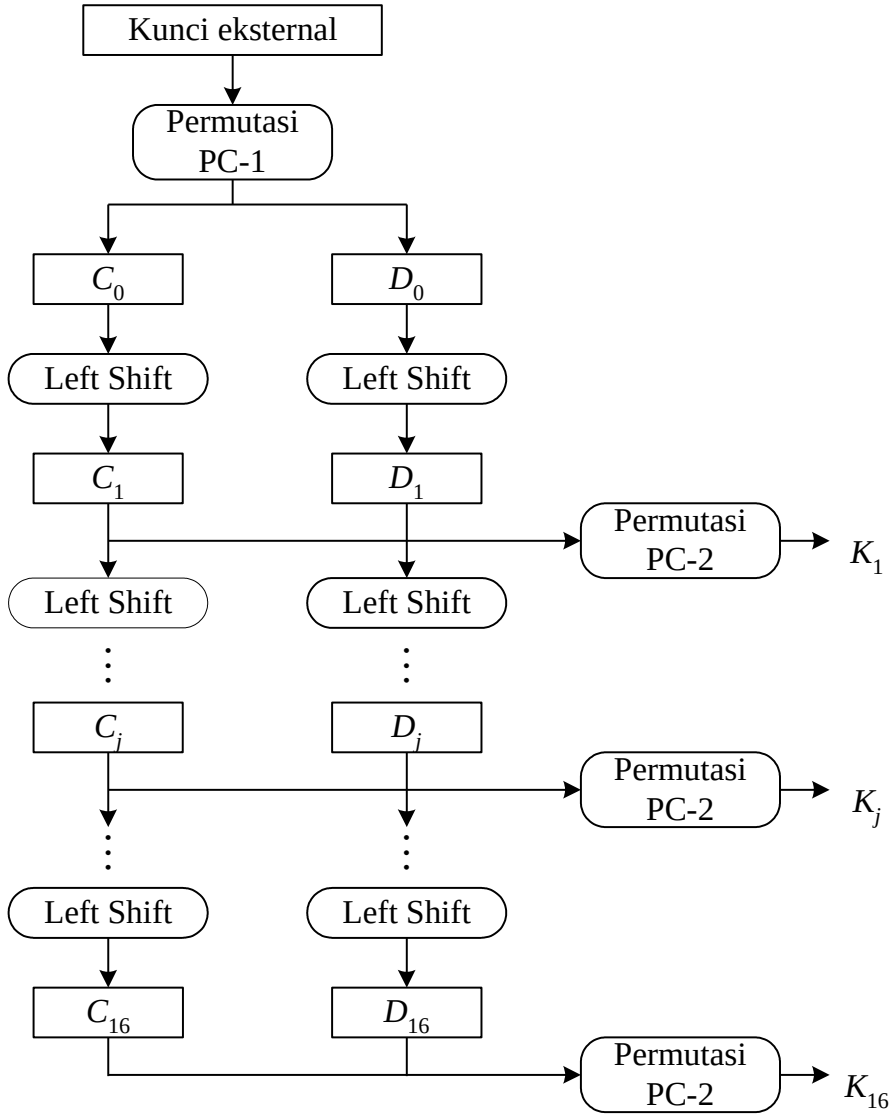
D_0: berisi bit-bit dari K pada posisi

63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22
14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4



Tabel 1. Jumlah pergeseran bit pada setiap putaran

Putaran, *i*	Jumlah pergeseran bit
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1



Matriks PC-2 berikut:

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Jadi, K_i merupakan penggabungan bit-bit C_i pada posisi:

14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10

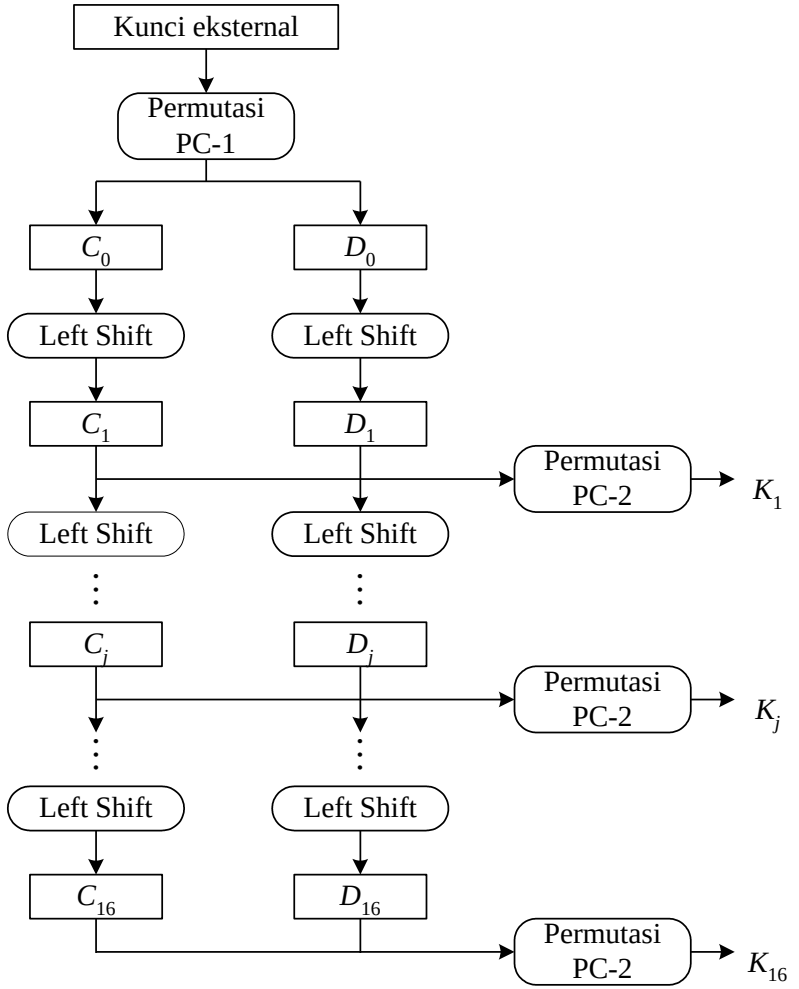
23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2

dengan bit-bit D_i pada posisi:

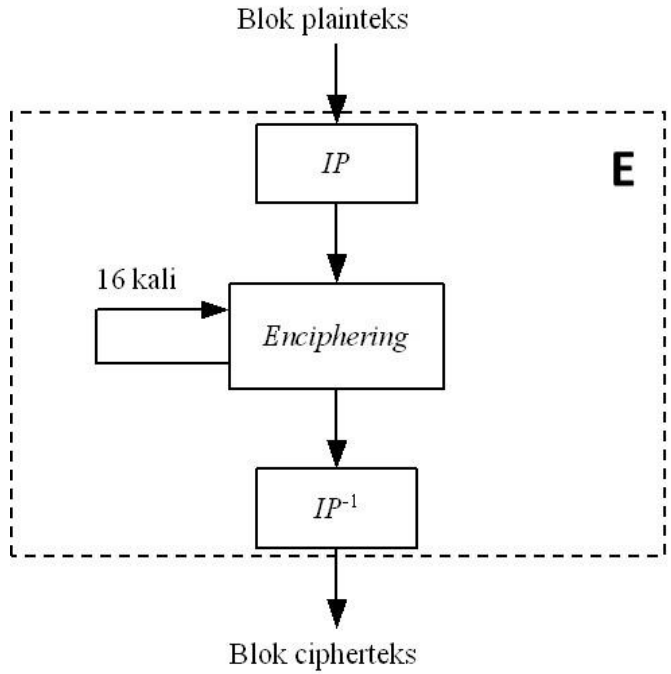
41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48

44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32

Setiap kunci internal K_i mempunyai panjang 48 bit.



Permutasi Awal



- Tujuan: mengacak plainteks sehingga urutan bit-bit di dalamnya berubah.
- Matriks permutasi awal (*IP*):

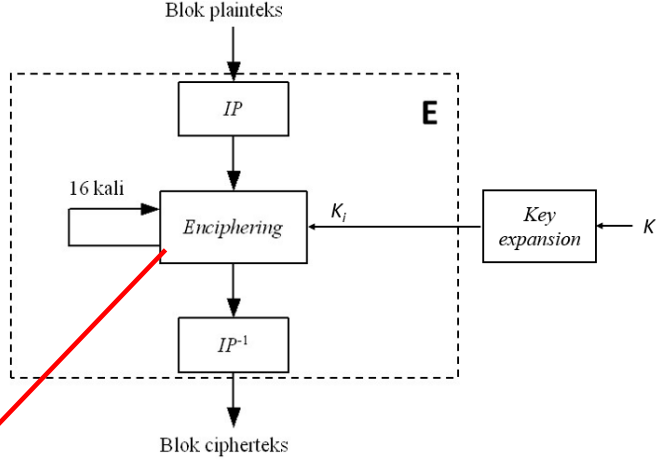
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

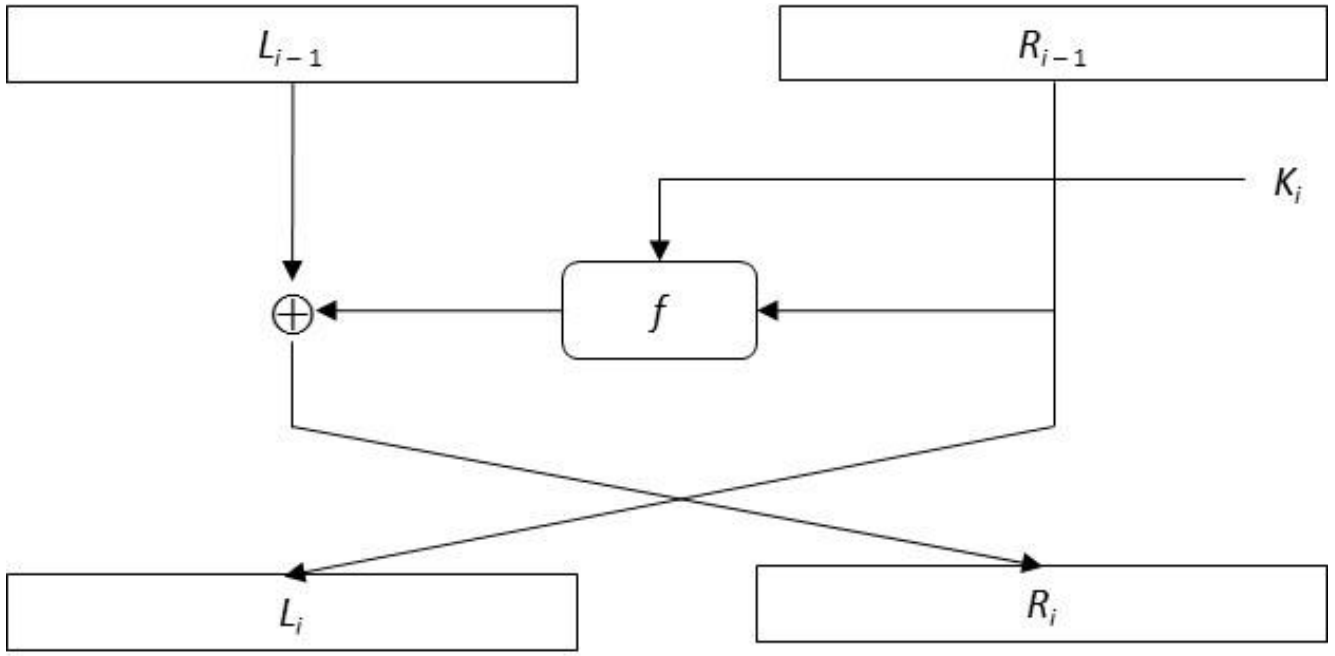
Enciphering

- Setiap blok plainteks mengalami 16 kali putaran *enciphering*
- Setiap putaran *enciphering* merupakan jaringan Feistel:

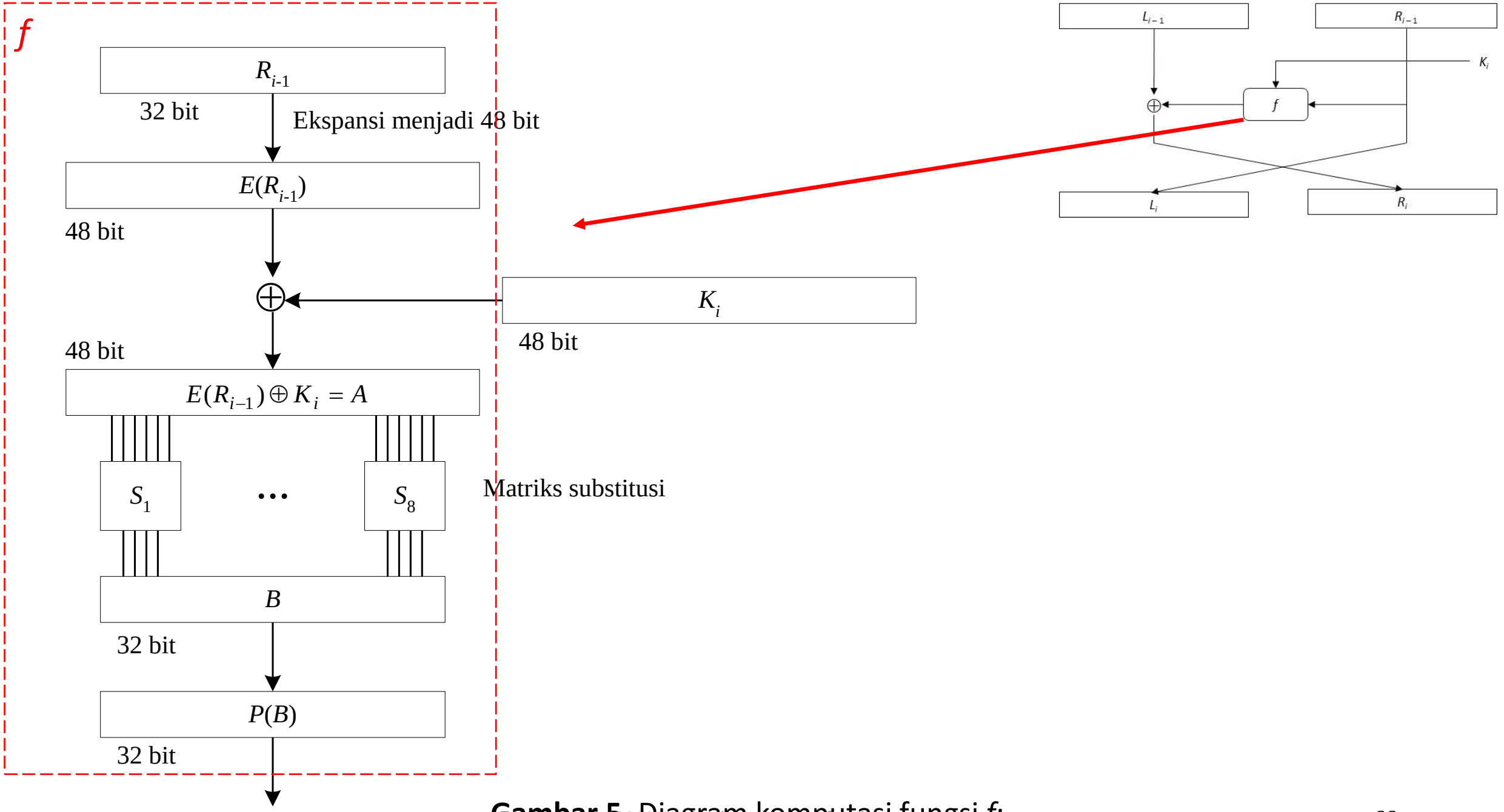
$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$





Gambar 4. Satu putaran *enciphering*

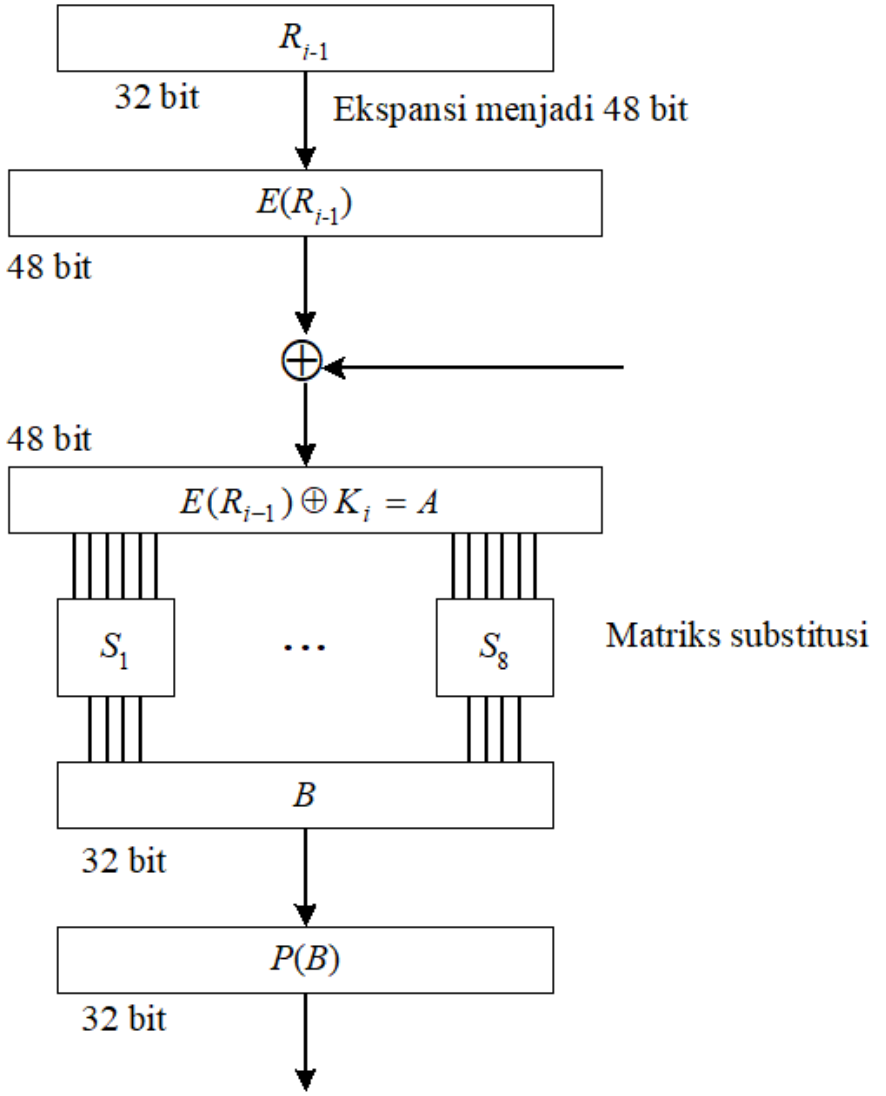


Gambar 5. Diagram komputasi fungsi *f*:

- *E* adalah fungsi ekspansi yang memperluas blok R_{i-1} 32-bit menjadi blok 48 bit.

- Fungsi ekspansi direalisasikan dengan matriks permutasi ekspansi:

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1



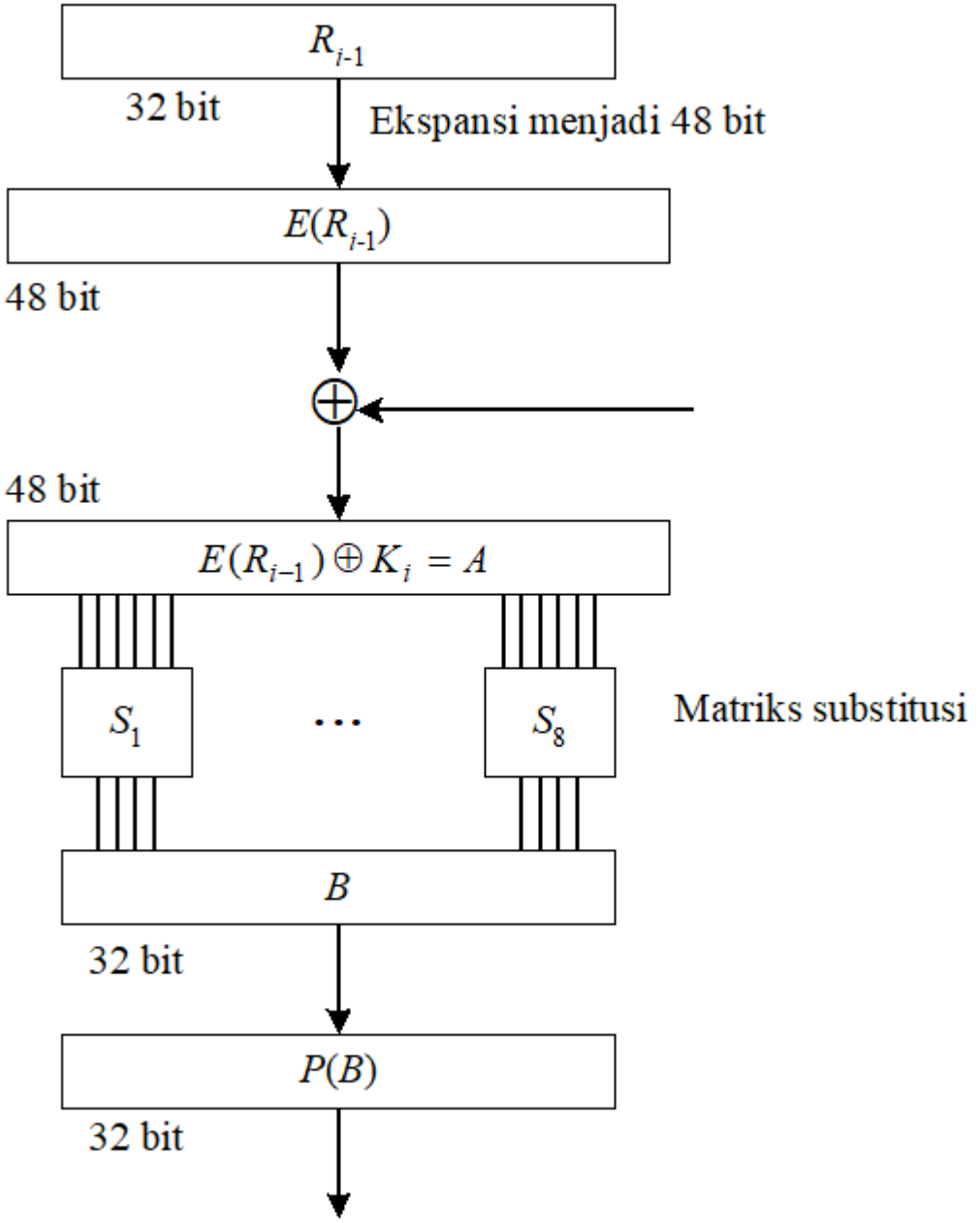
- Hasil ekspansi, yaitu $E(R_{i-1})$ di-XOR-kan dengan K_i menghasilkan blok A berukuran 48-bit:

$$E(R_{i-1}) \oplus K_i = A$$

- Blok A dikelompokkan menjadi 8 kelompok, masing-masing 6 bit, dan menjadi masukan bagi proses substitusi.

- Ada 8 matriks substitusi, masing-masing dinyatakan dengan kotak-S.

- Kotak –S menerima masukan 6 bit dan memberikan luaran 4 bit.



S_1:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2:

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3:

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4:

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Input: 101010

Baris = 10 = 2

Kolom = 0101 = 5

Luaran = 13 = 1101

S_5:

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	16
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6:

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7:

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

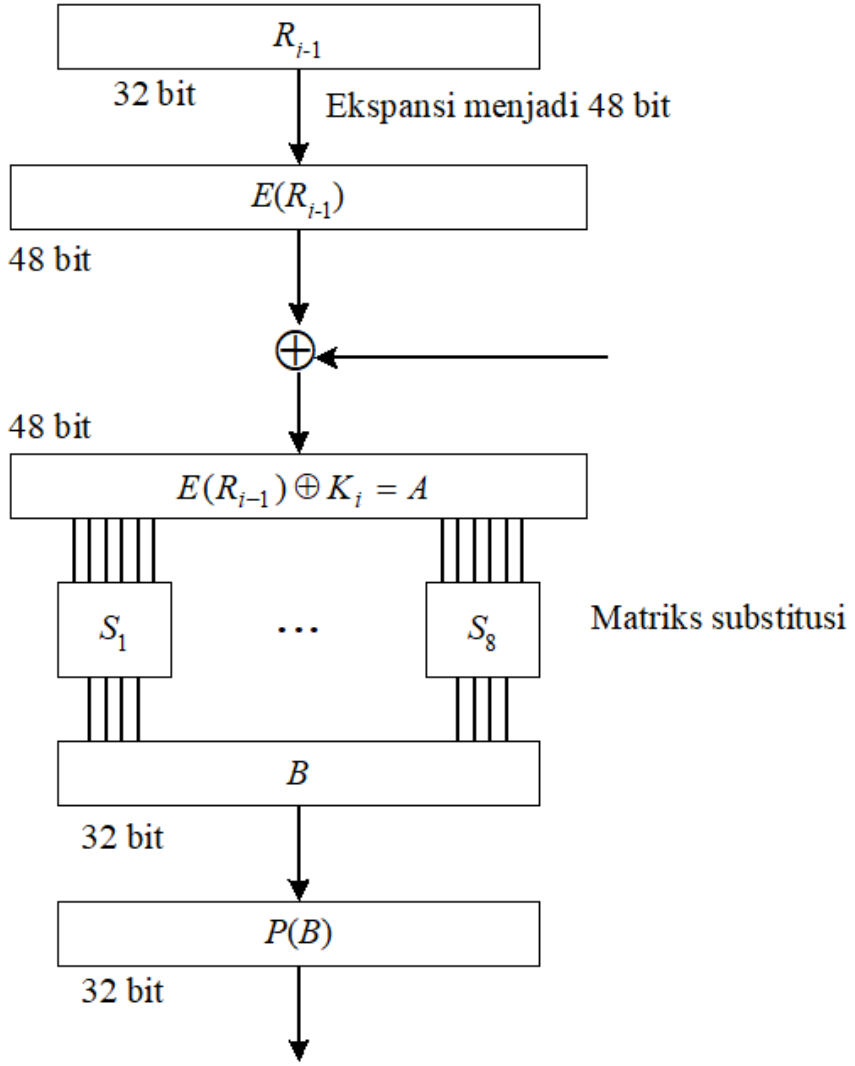
S_8:

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

- Setiap baris di dalam S-box adalah permutasi angka-angka 0, 1, 2,…, 15.
- Mengubah satu bit di dalam input S-box mengakibatkan perubahan paling sedikit 2 bit pada output S-box

→ prinsip *confussion*

- Luaran proses substitusi adalah blok *B* yang panjangnya 32 bit.
- Blok *B* menjadi masukan untuk proses permutasi.
- Tujuan permutasi adalah untuk mengacak hasil proses substitusi kotak-S.
- Permutasi dilakukan dengan menggunakan matriks permutasi *P* (*P-box*) sbb:



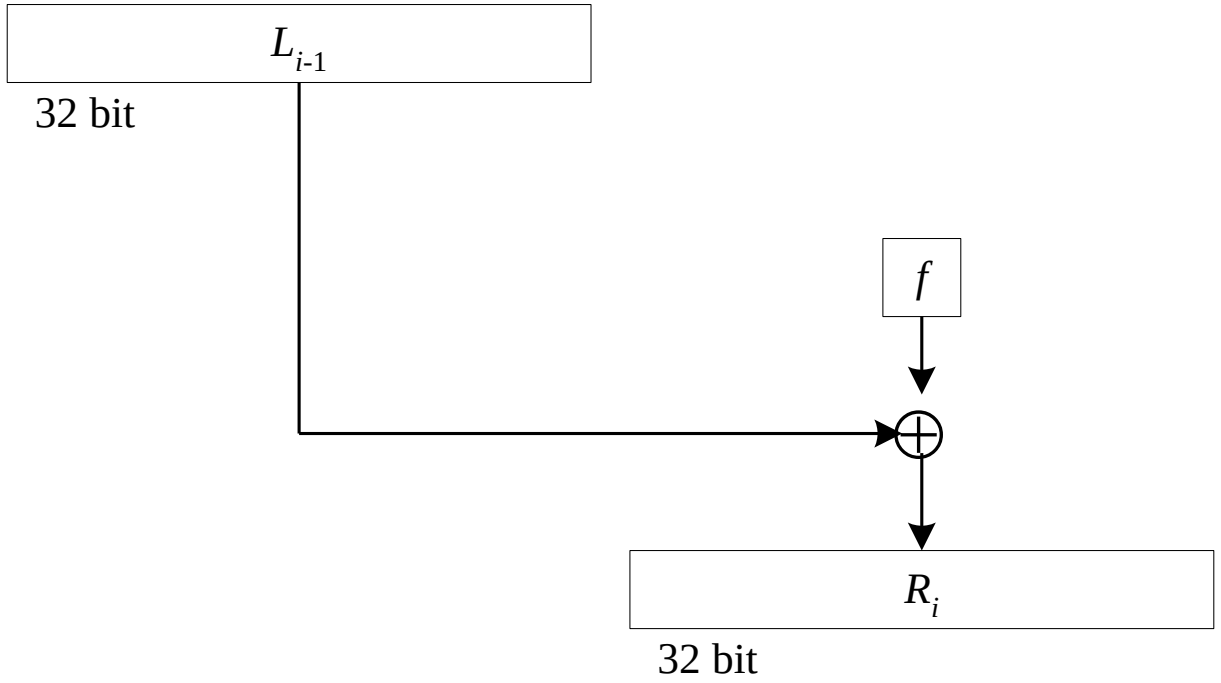
16	7	20	21	29	12	28	17	1	15	23	26	5	8	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

- $P(B)$ merupakan luaran dari fungsi f.
- Bit-bit $P(B)$ di-*XOR*-kan dengan L_{i-1} menghasilkan R_i:

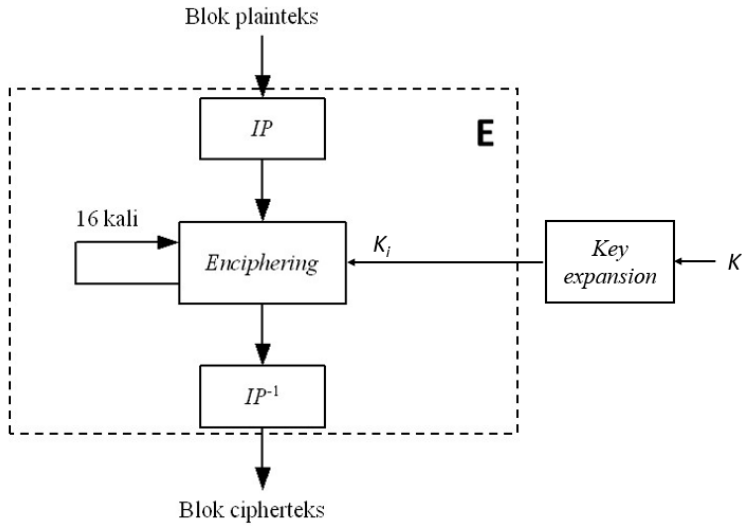
$$R_i = L_{i-1} \oplus P(B)$$

- Jadi, luaran dari putaran ke-i adalah

$$(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus P(B))$$



Inversi Permutasi (IP^{-1})



- Permutasi terakhir dilakukan setelah 16 kali putaran terhadap gabungan blok kiri dan blok kanan.

- Permutasi menggunakan matriks permutasi awal balikan (IP^{-1}) sbb:

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Contoh hasil enkripsi DES pada setiap putaran:

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Sumber: Cryptography and Network Security Chapter 3, Lecture slides by Lawrie Brown Modified by Richard Newman

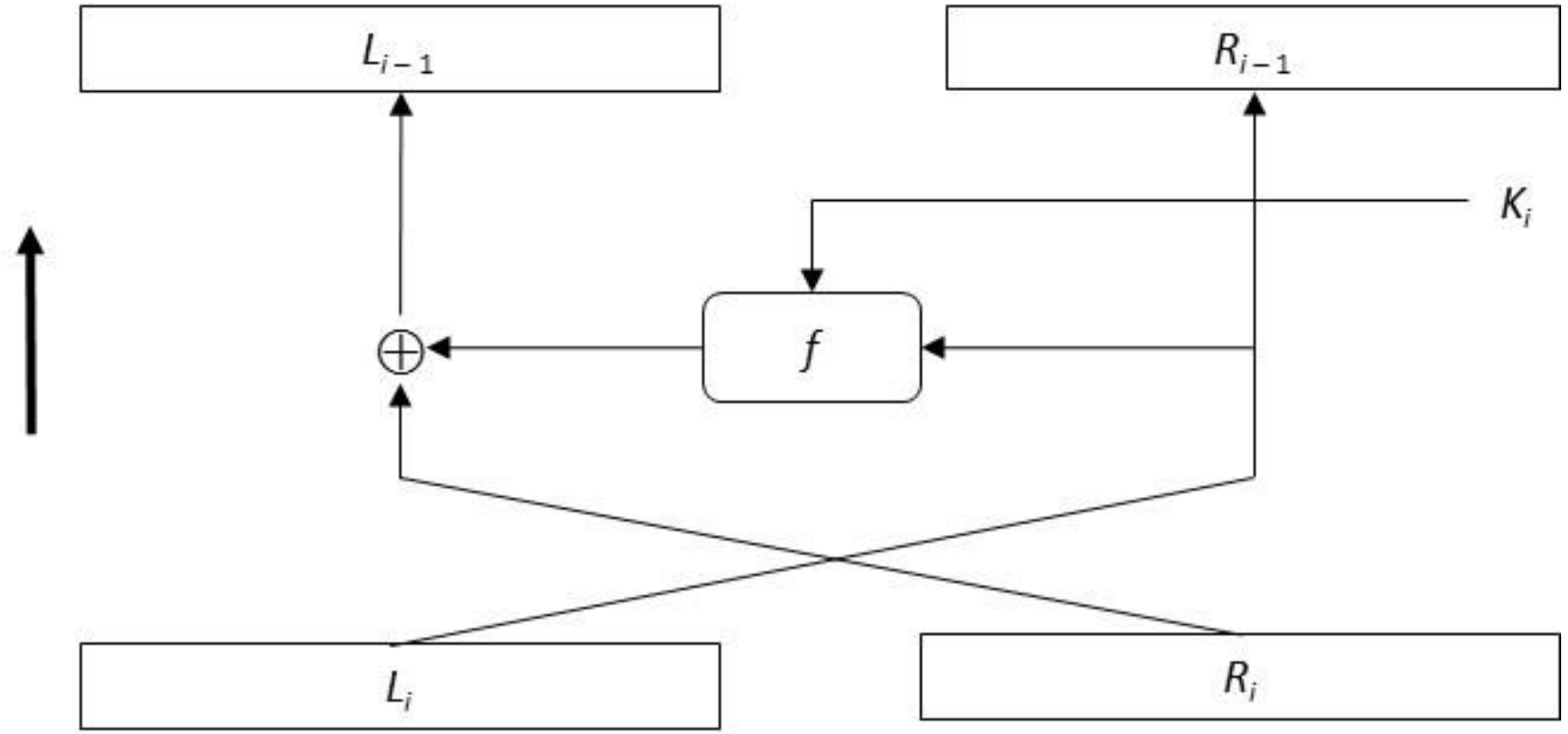
Dekripsi

- Dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi.
- DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi.
- Pada proses dekripsi urutan kunci yang digunakan adalah K_{16}, K_{15}, …, K_1.
- Untuk tiap putaran 16, 15, …, 1, luaran pada setiap putaran *deciphering* adalah

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus f(R_{i-1}, K_i) = R_i \oplus f(L_i, K_i)$$

Dekripsi:



$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus f(R_{i-1}, K_i) = R_i \oplus f(L_i, K_i)$$

Keamanan DES

- Keamanan DES ditentukan oleh kunci.
- Panjang kunci eksternal DES hanya 64 bit, tetapi yang dipakai hanya 56 bit.
- Pada rancangan awal, panjang kunci yang diusulkan IBM adalah 128 bit, tetapi atas permintaan NSA, panjang kunci diperkecil menjadi 56 bit.
- Dengan panjang kunci 56 bit akan terdapat 2^{56} atau 72.057.594.037.927.936 kemungkinan kunci.
- Ruang kunci (*key space*) DES sebanyak 2^{56} terlalu kecil, tidak aman karena kunci dapat ditemukan dengan serangan *exhaustive key search* (*brute force attack*)
- Jika serangan *exhaustive key search* (*brute force attack*) dengan menggunakan prosesor paralel, andaikan dalam satu detik dapat dikerjakan satu juta serangan. Jadi seluruhnya diperlukan 1142 tahun untuk menemukan kunci yang benar.

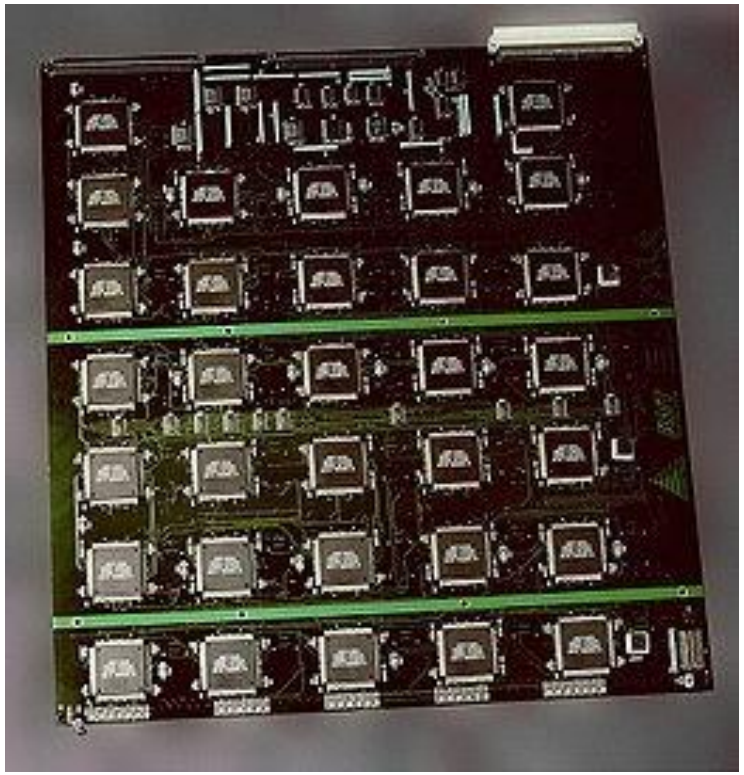
Dikutip dari Wiki:

- In 1997, RSA Security sponsored a series of contests, offering a $10,000 prize to the first team that broke a message encrypted with DES for the contest.

- That contest was won by the DESCHALL Project, led by Rocke Verser, Matt Curtin, and Justin Dolske, using idle cycles of thousands of computers across the Internet.

- Tahun 1998, *Electronic Frontier Foundation* (*EFE*) merancang dan membuat perangkat keras khusus untuk menemukan kunci DES secara *exhaustive key search* dengan biaya $250.000 dan diharapkan dapat menemukan kunci selama 5 hari.

- Tahun 1999, kombinasi perangkat keras *EFE* dengan kolaborasi internet yang melibatkan lebih dari 100.000 komputer dapat menemukan kunci DES kurang dari 1 hari.



The EFF's US$250,000 DES cracking machine contained 1,856 custom chips and could brute force a DES key in a matter of days — the photo shows a DES Cracker circuit board fitted with several Deep Crack chips (Sumber Wikipedia).

- Their motivation was to show that DES was breakable in practice as well as in theory: "*There are many people who will not believe a truth until they can see it with their own eyes. Showing them a physical machine that can crack DES in a few days is the only way to convince some people that they really cannot trust their security to DES.*"
- The machine brute-forced a key in a little more than 2 days search.

- Pengisian kotak-S DES masih menjadi misteri. Nilai-nilai awal S-box sudah diubah.

- Sebenarnya delapan putaran sudah cukup untuk membuat cipherteks sebagai fungsi acak dari setiap bit plainteks dan setiap bit kunci.

- Namun dari penelitian dengan melakukan kriptanalisis, DES dengan jumlah putaran yang kurang dari 16 ternyata dapat dipecahkan dengan *known-plaintext attack.*

Demo online DES

https://encode-decode.com/des-encrypt-online/