

DATA ENCRYPTION STANDARD (DES) STANDAR ENKRIPSI DATA

Algoritma Kriptografi Modern

SEJARAH DES

- 1960-1971; Proyek Lucifer IBM dipimpin Horst Feistel untuk kriptografi modern. Lucifer dikenal sbg blok kode pada operasi blok 64 bit dengan kunci ukuran 128 bit. Proyek Lucifer komersial dikenal dengan DES (Data Encryption Standard), dipimpin oleh Walter Tuchman.

SEJARAH DES

- 1973; NIST (National Institute of Standard and Technology) merilis proposal untuk kode standar nasional. Kriteria standar algoritma kriptografi:
 - Algoritma harus memberikan level keamanan yang tinggi;
 - Algoritma harus lengkap dan mudah diengerti;
 - Keamanan algoritma harus memiliki kunci (tidak tergantung dari algoritma yg ada);
 - Algoritma harus available untuk semua user;
 - Algoritma harus dapat beradaptasi dengan berbagai aplikasi;
 - Algoritma harus ekonomis (tidak perlu alat canggih);
 - Algoritma harus efisien dalam penggunaannya;
 - Algoritma harus valid;
 - Algoritma harus exportable.

SEJARAH DES

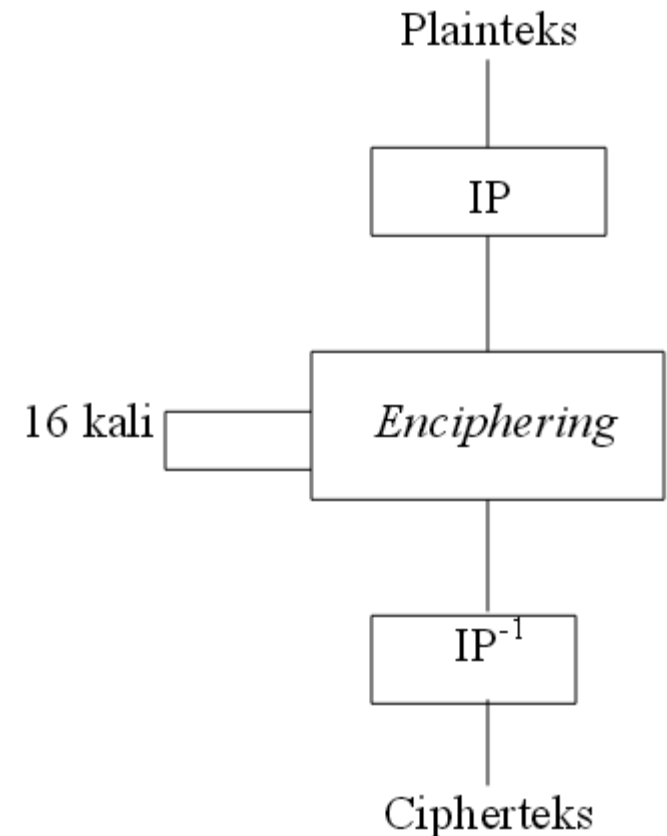
- Algoritma ini telah disetujui oleh National Bureau of Standard (NBS) setelah penilaian kekuatannya oleh National Security Agency (NSA) Amerika Serikat.
- 1976; Algoritma DES dipilih sebagai standar yang dipakai pada pemerintahan.
- 1977; UU penggunaan DES diterbitkan, FIPS PUB 46.
- 1987; Evaluasi ke-1;
- 1992; Evaluasi ke-2;

TINJAUAN UMUM

- DES adalah standard, sedangkan algoritmanya adalah DEA (Data Encryption Algorithm). Kedua nama ini sering dikacaukan.
- DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis cipher blok.
- DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal (internal key) atau subkey. Kunci internal dibangkitkan dari kunci eksternal (external key) yang panjangnya 64 bit, tetapi hanya 56 bit yang dipakai (8 bit paritas tidak digunakan).

Skema DES

- Skema global dari algoritma DES adalah sebagai berikut :
 1. Blok plainteks dipermutasi dengan matriks permutasi awal (initial permutation atau IP).
 2. Hasil permutasi awal kemudian di - enciphering- sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
 3. Hasil enciphering kemudian dipermutasi dengan matriks permutasi balikan (invers initial permutation atau IP^{-1}) menjadi blok cipherteks.

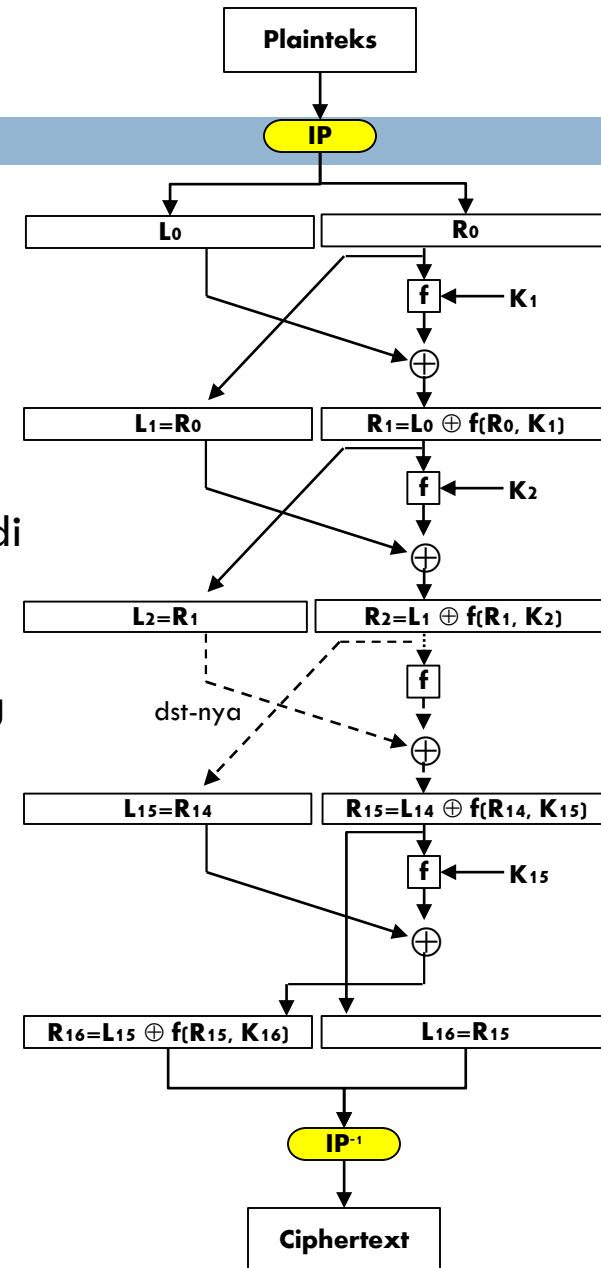


Skema DES

- Di dalam proses enciphering, blok plainteks terbagi menjadi dua bagian, kiri (L) dan kanan (R), yang masing-masing panjangnya 32 bit. Kedua bagian ini masuk ke dalam 16 putaran DES.
- Pada setiap putaran i , blok R merupakan masukan untuk fungsi transformasi yang disebut f . Pada fungsi f , blok R dikombinasikan dengan kunci internal K_i . Keluaran dari fungsi f di -XOR- kan dengan blok L untuk mendapatkan blok R yang baru. Sedangkan blok L yang baru langsung diambil dari blok R sebelumnya. Ini adalah satu putaran DES.
- Secara matematis satu putaran DES dapat dinyatakan sebagai:

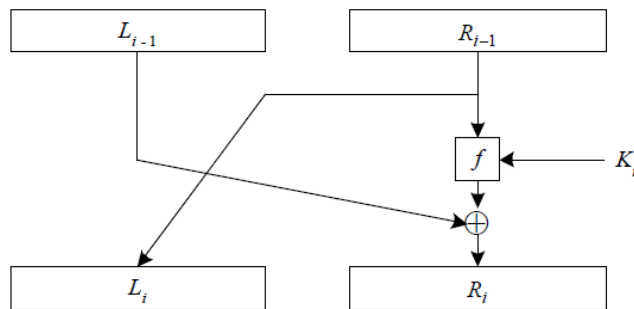
$$L_i = R_{i-1}$$

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



Skema DES

- Satu putaran DES membentuk jaringan Feistel.



- Catatan: Jika $(\mathbf{L16}, \mathbf{R16})$ merupakan keluaran dari putaran ke-16, aka $(\mathbf{R16}, \mathbf{L16})$ merupakan pra-cipherteks (*pre-ciphertext*) dari enciphering ini.
- Cipherteks yang sebenarnya diperoleh dengan melakukan permutasi awal balikan, IP^{-1} , terhadap blok pra-cipherteks.

Permutasi Awal (Initial Permutation)

- Sebelum putaran pertama, terhadap blok plainteks dilakukan permutasi awal (*initial permutation* atau *IP*). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan bit-bit di dalamnya berubah. Pengacakan dilakukan dengan menggunakan matriks permutasi awal berikut.

Permutasi Awal (Initial Permutation)

- Misal, M adalah masukan bilangan binary 64 bit.

M1	M2	M3	M4	M5	M6	M7	M8
M9	M10	M11	M12	M13	M14	M15	M16
M17	M18	M19	M20	M21	M22	M23	M24
M25	M26	M27	M28	M29	M30	M31	M32
M33	M34	M35	M36	M37	M38	M39	M40
M41	M42	M43	M44	M45	M46	M47	M48
M49	M50	M51	M52	M53	M54	M55	M56
M57	M58	M59	M60	M61	M62	M63	M64

- Jika x adalah bilangan binary, permutasi $X=IP(M)$ adalah:



M58	M50	M42	M34	M26	M18	M10	M2
M60	M52	M44	M36	M28	M20	M12	M4
M62	M54	M46	M38	M30	M22	M14	M6
M64	M56	M48	M40	M32	M24	M16	M8
M57	M49	M41	M33	M25	M17	M9	M1
M59	M51	M43	M35	M27	M19	M11	M3
M61	M53	M45	M37	M29	M21	M13	M5
M63	M55	M47	M39	M31	M23	M15	M7

Tabel IP

1	2	3	4	5	6	7	8
58	50	42	34	26	18	10	2

9	10	11	12	13	14	15	16
60	52	44	36	28	20	12	4

17	18	19	20	21	22	23	24
62	54	46	38	30	22	14	6

25	26	27	28	29	30	31	32
64	56	48	40	32	24	16	8

33	34	35	36	37	38	39	40
57	49	41	33	25	17	9	1

41	42	43	44	45	46	47	48
59	51	43	35	27	19	11	3

49	50	51	52	53	54	55	56
61	53	45	37	29	21	13	5

57	58	59	60	61	62	63	64
63	55	47	39	31	23	15	7

Invers Permutasi Awal

□ Tabel Invers Initial Permutation (IP^{-1})

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

Pembangkitan Kunci Internal

- Kunci internal = Kunci setiap putaran
- DES mempunyai 16 putaran, sehingga memerlukan 16 kunci internal: **$K_1, K_2, K_3, \dots, K_{16}$**
- Kunci internal dibangkitkan sebelum atau bersamaan dengan proses enkripsi.
- Kunci internal dibangkitkan dari kunci eksternal (oleh pengguna) yang panjangnya 64 bit (8 karakter).

Pembangkitan Kunci Internal

PC : Permutation Choice

- Tabel yg digunakan sbg DES schedule key adalah:

Tabel Permutasi Pilihan Satu (PC-1)

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

- Dalam permutasi ini, tiap bit kedelapan (parity bit) dari delapan byte kunci diabaikan. Hasil permutasinya adalah sepanjang 56 bit sehingga panjang kunci DES=56 bit. Selanjutnya, 56 bit ini dibagi menjadi 2 bagian, kiri dan kanan, yg masing-masing panjangnya 28 bit, dan masing-masing disimpan sebagai **C₀** dan **D₀**.

Pembangkitan Kunci Internal

- **C₀** 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
- **D₀** 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
- Kemudian kedua bagian digeser ke kiri (left shift) sepanjang 1 atau 2 bit tergantung tiap putaran. Operasi pergeseran bersifat wrapping atau round shift.

Pembangkitan Kunci Internal

- Jumlah pergeseran bit setiap putaran, adalah:

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

- Misal (**C_i**, **D_i**) adalah penggabungan **C_i** dan **D_i**, maka (**C_{i+1}**, **D_{i+1}**) diperoleh dengan menggeser **C_i** dan **D_i** sepanjang satu atau 2 bit. Setelah bit bergeser, (**C_i**, **D_i**) mengalami permutasi kompresi menggunakan tabel PC-2 berikut:

Pembangkitan Kunci Internal

Tabel Permutasi Pilihan Dua (PC-2)

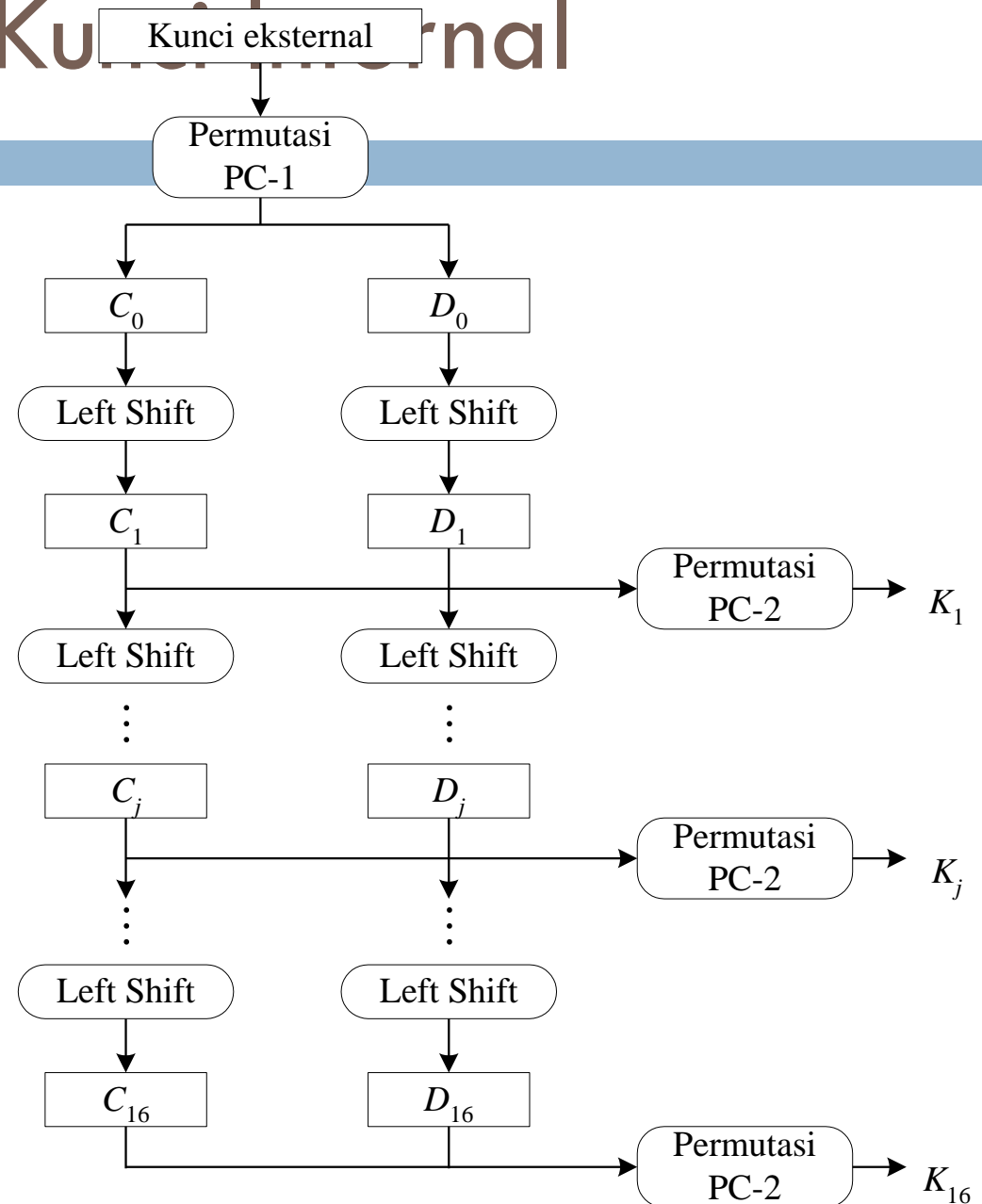
Input	1	2	3	4	5	6	7	8	9	10	11	12
Output	14	17	11	24	1	5	2	28	15	6	21	10
Input	13	14	15	16	17	18	19	20	21	22	23	24
Output	23	19	12	4	26	8	16	7	27	20	13	2
Input	25	26	27	28	29	30	31	32	33	34	35	36
Output	41	52	31	37	47	55	30	40	51	45	33	48
Input	37	38	39	40	41	42	43	44	45	46	47	48
Output	44	49	39	56	34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

Pembangkitan Kunci Internal

- Dengan permutasi tsb, maka kunci internal **K_i** diturunkan dari (**C_i**, **D_i**) yang dalam hal ini **K_i** merupakan penggabungan **C_i** dan **D_i** pada posisi-posisi berikut:
- **C_i** berisi bit-bit dari **K_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
- **D_i** berisi bit-bit dari **K_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
- Jadi setiap kunci internal **K_i** memiliki panjang 48 bit.

Pembangkitan Kunci Internal

- Proses pembangkitan kunci internal dapat dilihat pada skema berikut:

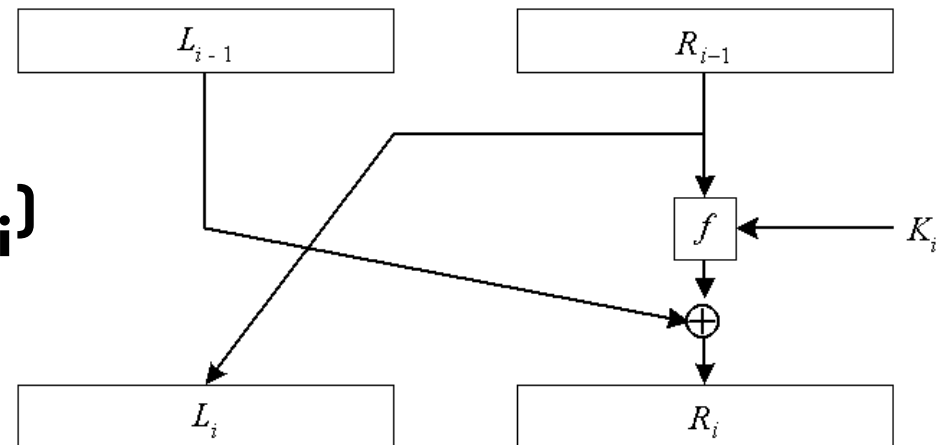


Enciphering

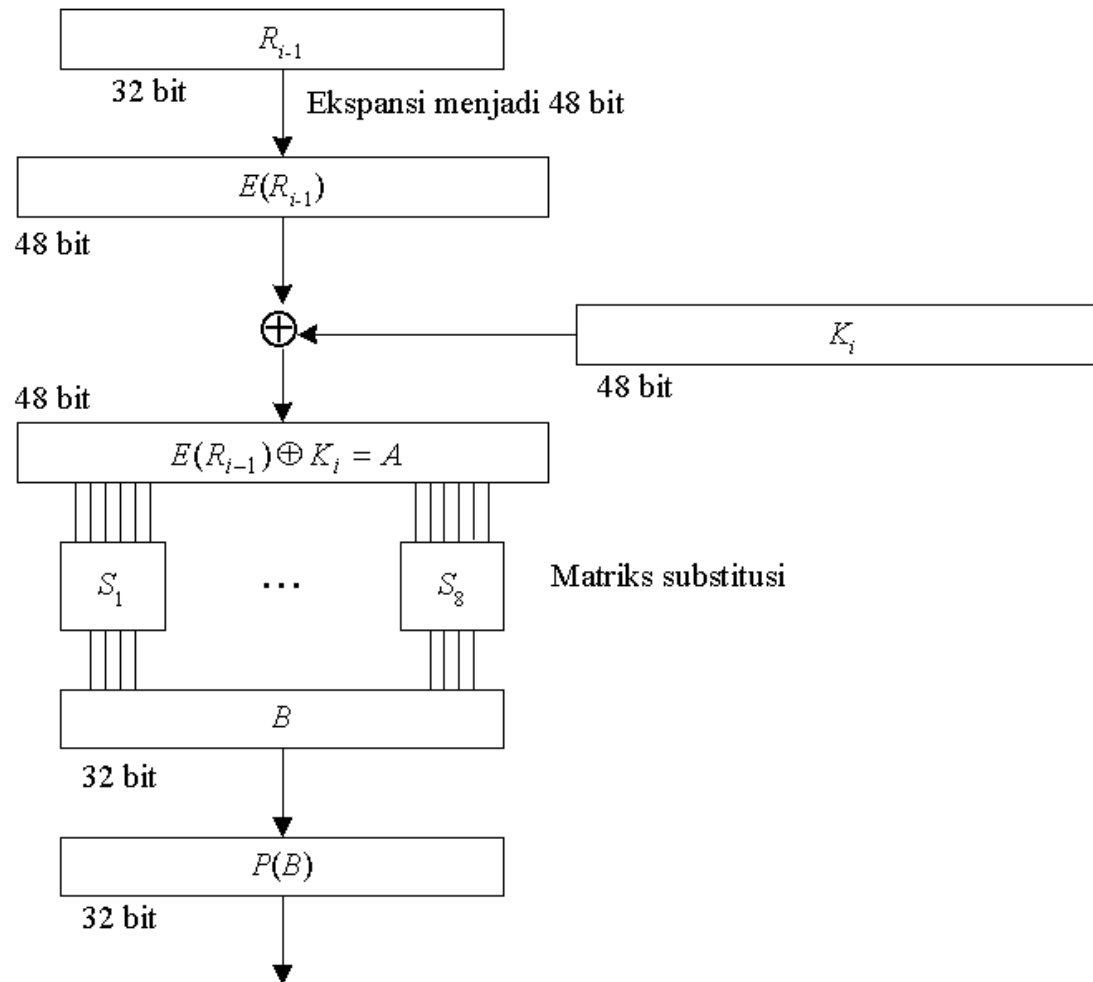
- Setiap blok plainteks mengalami 16 kali putaran *enciphering* .
- Setiap putaran *enciphering* merupakan jaringan Feistel:

$$\mathbf{L}_i = \mathbf{R}_{i-1}$$

$$\mathbf{R}_i = \mathbf{L}_{i-1} \oplus \mathbf{f}(\mathbf{R}_{i-1}, \mathbf{K}_i)$$



□ Diagram komputasi fungsi **f** :



- **E** adalah fungsi ekspansi yang memperluas blok 32-bit R_{i-1} menjadi blok 48 bit.
- Fungsi ekspansi direalisasikan dengan matriks permutasi ekspansi:
- Tabel Permutasi Ekspansi

Input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Output	32	1	2	3	4	5	4	5	6	7	8	9	8	9	10	11
Input	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Output	12	13	12	13	14	15	16	17	18	17	18	19	20	21	20	21
Input	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Output	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 vektor A 48-bit:

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

- Vektor 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 keluaran 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

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

- Keluaran proses substitusi adalah vektor **B** yang panjangnya 48 bit.
- Vektor **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:

Tabel Fungsi Permutasi

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

Contoh

- Plaintext: COMPUTER
- Kunci: 13 34 57 79 9B BC DF F1
- Lakukan enkripsi dengan DES

Langkah 1

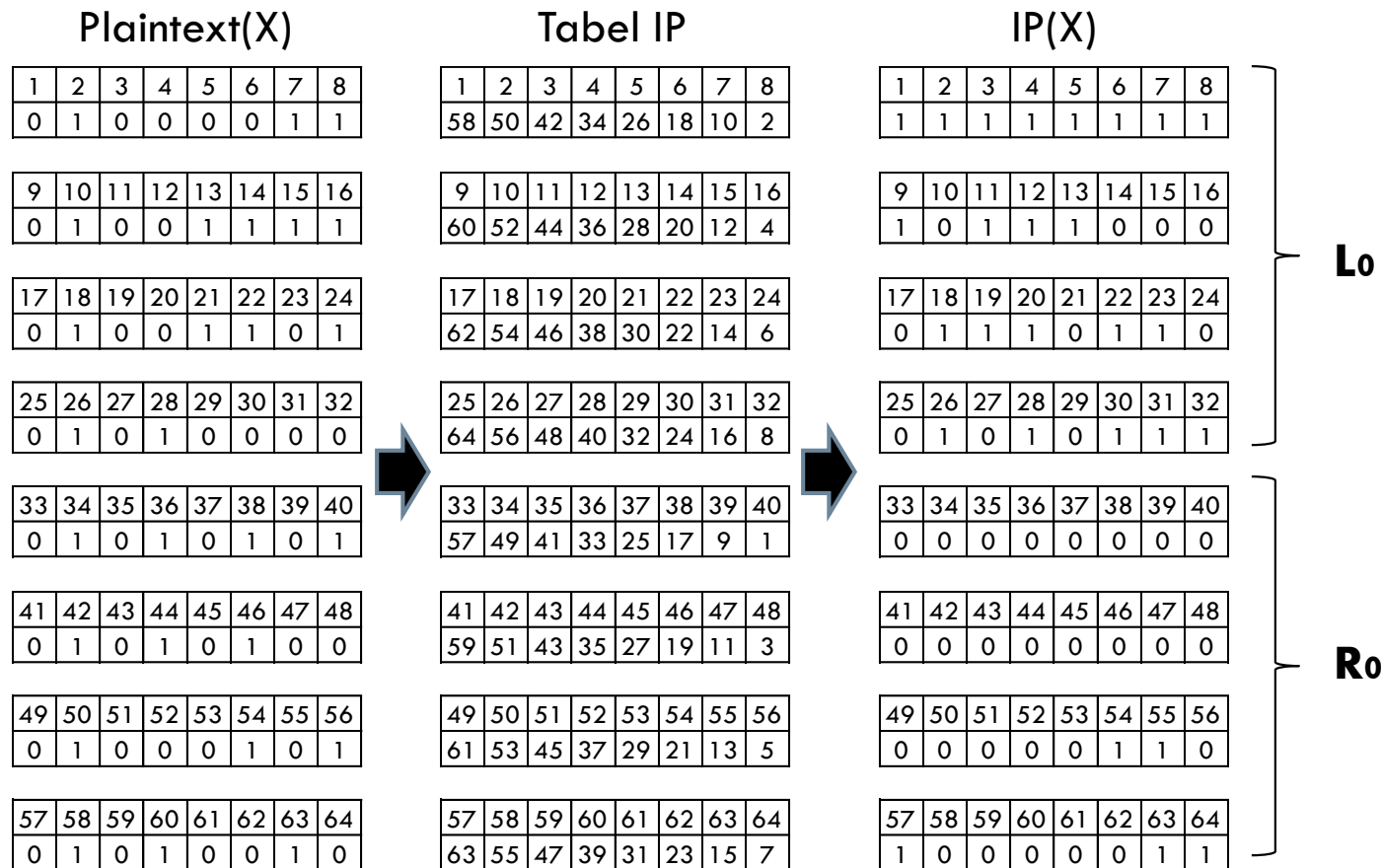
- Mengubah plaintext dan kunci menjadi bilangan biner

	Hexa		Biner							
	↓									
C	43		1	2	3	4	5	6	7	8
			0	1	0	0	0	0	1	1
O	4F		9	10	11	12	13	14	15	16
			0	1	0	0	1	1	1	1
M	4D		17	18	19	20	21	22	23	24
			0	1	0	0	1	1	0	1
P	50		25	26	27	28	29	30	31	32
			0	1	0	1	0	0	0	0
U	55		33	34	35	36	37	38	39	40
			0	1	0	1	0	1	0	1
T	54		41	42	43	44	45	46	47	48
			0	1	0	1	0	1	0	0
E	45		49	50	51	52	53	54	55	56
			0	1	0	0	0	1	0	1
R	52		57	58	59	60	61	62	63	64
			0	1	0	1	0	0	1	0
			Plaintext							

Hexa		Biner							
↓									
13		1	2	3	4	5	6	7	8
		0	0	0	1	0	0	1	1
34		9	10	11	12	13	14	15	16
		0	0	1	1	0	1	0	0
57		17	18	19	20	21	22	23	24
		0	1	0	1	0	1	1	1
79		25	26	27	28	29	30	31	32
		0	1	1	1	1	0	0	1
9B		33	34	35	36	37	38	39	40
		1	0	0	1	1	0	1	1
BC		41	42	43	44	45	46	47	48
		1	0	1	1	1	1	0	0
DF		49	50	51	52	53	54	55	56
		1	1	0	1	1	1	1	1
F1		57	58	59	60	61	62	63	64
		1	1	1	1	0	0	0	1
Kunci									

Langkah 2

Initial Permutation (IP) pada plaintext



Hasil: IP(X) = 11111111 10111000 01110110 01010111 00000000 00000000 00000110 10000011

Selanjutnya bit pada IP(X) dipecah menjadi 2:

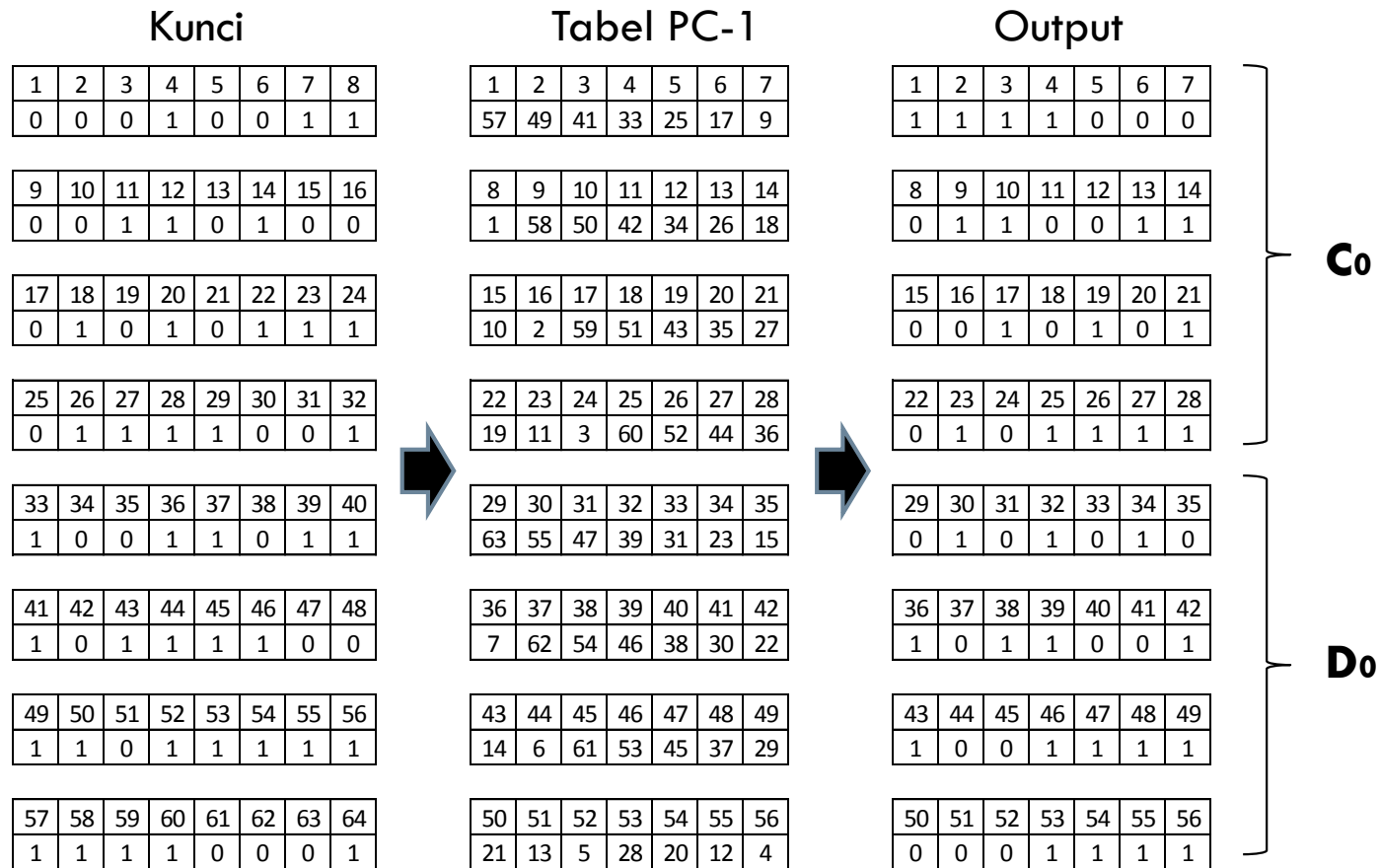
L₀ : 11111111 10111000 01110110 01010111

R₀ : 00000000 00000000 00000110 10000011

Langkah 3

Generate Kunci menggunakan tabel permutasi kompresi PC-1

Kompresi 64 bit menjadi 56 bit dengan membuang 1 bit (parity bit) tiap blok kunci



Hasil: **C₀D₀** = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Selanjutnya bit pada **C₀D₀** dipecah menjadi 2:

C₀ : 1111000 0110011 0010101 0101111

D₀ : 0101010 1011001 1001111 0001111

Langkah 4 : Left Shift Operztion

Lakukan pergeseran pada **C₀** dan **D₀** menggunakan tabel pergeseran bit 16 putaran

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

C₀D₀ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

C₀ : 1111000011001100101010101111	D₀ : 0101010101100110011110001111
C₁ : 1110000110011001010101011111	D₁ : 1010101011001100111100011110
C₂ : 1100001100110010101010111111	D₂ : 0101010110011001111000111101
C₃ : 0000110011001010101011111111	D₃ : 0101011001100111100011110101
C₄ : 0011001100101010101111111100	D₄ : 0101100110011110001111010101
C₅ : 1100110010101010111111110000	D₅ : 0110011001111000111101010101
C₆ : 0011001010101011111111000011	D₆ : 1001100111100011110101010101
C₇ : 1100101010101111111100001100	D₇ : 0110011110001111010101010110
C₈ : 0010101010111111110000110011	D₈ : 1001111000111101010101011001
C₉ : 0101010101111111100001100110	D₉ : 0011110001111010101010110011
C₁₀ : 0101010111111110000110011001	D₁₀ : 1111000111101010101011001100
C₁₁ : 0101011111111000011001100101	D₁₁ : 1100011110101010101100110011
C₁₂ : 0101111111100001100110010101	D₁₂ : 0001111010101010110011001111
C₁₃ : 0111111110000110011001010101	D₁₃ : 0111101010101011001100111100
C₁₄ : 1111111000011001100101010101	D₁₄ : 1110101010101100110011110001
C₁₅ : 1111100001100110010101010111	D₁₅ : 1010101010110011001111000111
C₁₆ : 1111000011001100101010101111	D₁₆ : 0101010101100110011110001111

Setiap hasil putaran digabungkan kembali menjadi C_iD_i dan diinput kedalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data C_iD_i 56 bit menjadi $CiDi$ 48 bit.

Tabel PC-2

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

Berikut hasil outputnya:

$C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110$
 $K_1 = 000110 110000 001011 101111 111111 000111 000001 110010$

$C_2D_2 = 1100001 1001100 1010101 0111111 0101010 1100110 0111100 0111101$
 $K_2 = 011110 011010 111011 011001 110110 111100 100111 100101$

$C_3D_3 = 0000110 0110010 1010101 1111111 0101011 0011001 1110001 1110101$
 $K_3 = 010101 011111 110010 001010 010000 101100 111110 011001$

$C_4D_4 = 0011001 1001010 1010111 1111100 0101100 1100111 1000111 1010101$
 $K_4 = 011100 101010 110111 010110 110110 110011 010100 011101$

$C_5D_5 = 1100110 0101010 1011111 1110000 0110011 0011110 0011110 1010101$
 $K_5 = 011111 001110 110000 000111 111010 110101 001110 101000$

$C_6D_6 = 0011001 0101010 1111111 1000011 1001100 1111000 1111010 1010101$
 $K_6 = 011000 111010 010100 111110 010100 000111 101100 101111$

$C_7D_7 = 1100101 0101011 1111110 0001100 0110011 1100011 1101010 1010110$
 $K_7 = 111011 001000 010010 110111 111101 100001 100010 111100$

$C_8D_8 = 0010101 0101111 1111000 0110011 1001111 0001111 0101010 1011001$
 $K_8 = 111101 111000 101000 111010 110000 010011 101111 111011$

$C_9D_9 = 0101010 1011111 1110000 1100110 0011110 0011110 1010101 0110011$
 $K_9 = 111000 001101 101111 101011 111011 011110 011110 000001$

$C_{10}D_{10} = 0101010 1111111 1000011 0011001 1111000 1111010 1010101 1001100$
 $K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111$

$C_{11}D_{11} = 0101011 1111110 0001100 1100101 1100011 1101010 1010110 0110011$
 $K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110$

$C_{12}D_{12} = 0101111 1111000 0110011 0010101 0001111 0101010 1011001 1001111$
 $K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001$

$C_{13}D_{13} = 0111111 1100001 1001100 1010101 0111101 0101010 1100110 0111100$
 $K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001$

$C_{14}D_{14} = 1111111 0000110 0110010 1010101 1110101 0101011 0011001 1110001$
 $K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010$

$C_{15}D_{15} = 1111100 0011001 1001010 1010111 1010101 0101100 1100111 1000111$
 $K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010$

$C_{16}D_{16} = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111$
 $K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101$

Note: $K_i = PC-2(CiDi)$

Langkah 5

Pada langkah ini, kita akan meng-ekspansi data R_{i-1} 32 bit menjadi R_i 48 bit sebanyak 16 kali putaran dengan nilai perputaran $1 \leq i \leq 16$ menggunakan Tabel Ekspansi (E).

Tabel Ekspansi(E)

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 $E(R_{i-1})$ kemudian di XOR dengan K_i dan menghasilkan Vektor Matriks A_i .

Berikut hasil outputnya:

Iterasi 1

$$\begin{array}{l} E(R(1)-1) = 100000\ 000000\ 000000\ 000000\ 000000\ 001101\ 010000\ 000110 \\ K1 = 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010 \\ \hline A1 = 100110\ 110000\ 001011\ 101111\ 111111\ 001010\ 010001\ 110100 \end{array}$$

Iterasi – 2

$$\begin{array}{l} E(R(2)-1) = 011010\ 101110\ 100001\ 010110\ 100110\ 100101\ 010000\ 001101 \\ K2 = 011110\ 011010\ 111011\ 011001\ 110110\ 111100\ 100111\ 100101 \\ \hline A2 = 000100\ 110100\ 011010\ 001111\ 010000\ 011001\ 110111\ 101000 \end{array}$$

Iterasi – 3

$$\begin{array}{l} E(R(3)-1) = 010001\ 010111\ 111011\ 110011\ 110001\ 010101\ 010010\ 100001 \\ K3 = 010101\ 011111\ 110010\ 001010\ 010000\ 101100\ 111110\ 011001 \\ \hline A3 = 000100\ 001000\ 001001\ 111001\ 100001\ 111001\ 101100\ 111000 \end{array}$$

Iterasi – 4

$$\begin{array}{l} E(R(4)-1) = 010111\ 110001\ 010111\ 110011\ 110101\ 011100\ 001111\ 110001 \\ K4 = 011100\ 101010\ 110111\ 010110\ 110110\ 110011\ 010100\ 011101 \\ \hline A4 = 001011\ 011011\ 100000\ 100101\ 000011\ 101111\ 011011\ 101100 \end{array}$$

Iterasi – 5

$$\begin{array}{l} E(R(5)-1) = 110110\ 101001\ 011100\ 000101\ 011001\ 011010\ 100110\ 100011 \\ K5 = 011111\ 001110\ 110000\ 000111\ 111010\ 110101\ 001110\ 101000 \\ \hline A5 = 101001\ 100111\ 101100\ 000010\ 100011\ 101111\ 101000\ 001011 \end{array}$$

Iterasi – 6

$$\begin{array}{l} E(R(6)-1) = 100101\ 011011\ 110001\ 010110\ 101110\ 101100\ 000111\ 111010 \\ K6 = 011000\ 111010\ 010100\ 111110\ 010100\ 000111\ 101100\ 101111 \\ \hline A6 = 111101\ 100001\ 100101\ 101000\ 111010\ 101011\ 101011\ 010101 \end{array}$$

Iterasi – 7

$$\begin{array}{l} E(R(7)-1) = 110010\ 100001\ 011111\ 110010\ 100111\ 111101\ 011001\ 010011 \\ K7 = 111011\ 001000\ 010010\ 110111\ 111101\ 100001\ 100010\ 111100 \\ \hline A7 = 001001\ 101001\ 001101\ 000101\ 011010\ 011100\ 111011\ 101111 \end{array}$$

Iterasi – 9

$$\begin{array}{l} E(R(9)-1) = 010010\ 101111\ 111000\ 000000\ 000010\ 101111\ 110101\ 010001 \\ K9 = 111000\ 001101\ 101111\ 101011\ 111011\ 011110\ 011110\ 000001 \\ \hline A9 = 101010\ 100010\ 010111\ 101011\ 111001\ 110001\ 101011\ 010000 \end{array}$$

Iterasi – 10

$$\begin{array}{l} E(R(10)-1) = 100111\ 111000\ 001110\ 100010\ 100111\ 110111\ 111000\ 001010 \\ K10 = 101100\ 011111\ 001101\ 000111\ 101110\ 100100\ 011001\ 001111 \\ \hline A10 = 001011\ 100111\ 000011\ 100101\ 001001\ 010011\ 100001\ 000101 \end{array}$$

Iterasi – 11
E(R(11)-1)= 010011 110111 111010 101010 101111 110011 110001 011001
K11 = 001000 010101 111111 010011 110111 101101 001110 000110
----- XOR
A11 = 011011 100010 000101 111001 011000 011110 111111 011111

Iterasi – 12
E(R(12)-1)= 001001 011010 101001 011111 110001 010111 110010 101100
K12 = 011101 010111 000111 110101 100101 000110 011111 101001
----- XOR
A12 = 010100 001101 101110 101010 010100 010001 101101 000101

Iterasi – 13
E(R(13)-1)= 100110 100111 110111 111011 111110 101110 101100 001010
K13 = 100101 111100 010111 010001 111110 101011 101001 000001
----- XOR
A13 = 000011 011011 100000 101010 000000 000101 000101 001011

Iterasi – 14
E(R(14)-1)= 111001 010111 110000 001000 001000 001000 001011 111011
K14 = 010111 110100 001110 110111 111100 101110 011100 111010
----- XOR
A14 = 101110 100011 111110 111111 110100 100110 010111 000001

Iterasi – 15
E(R(15)-1)= 000110 101100 001100 000001 011001 011010 100101 010100
K15 = 101111 111001 000110 001101 001111 010011 111100 001010
----- XOR
A15 = 101001 010101 001010 001100 010110 001001 011001 011110

Iterasi – 16
E(R(16)-1)= 101101 011101 010100 000101 010101 010001 010110 100010
K16 = 110010 110011 110110 001011 000011 100001 011111 110101
----- XOR
A16 = 011111 101110 100010 001110 010110 110000 001001 010111

Langkah 6

Setiap vektor **A_i** disubstitusikan ke 8 buah **S-box** (substitution box), dimana blok ke 1 disubstitusikan ke **S₁**, blok ke-2 disubstitusikan ke **S₂**, dst-nya, menghasilkan output vektor **B_i** 32 bit

S1 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
01	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
10	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
11	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
01	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
10	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
11	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
01	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
11	3	15	0	6	10	1	13	18	9	4	5	11	12	7	2	14

S5 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
01	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	16
10	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
01	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
10	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
11	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
01	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
10	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
11	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
01	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
10	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
11	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Kita ambil contoh S1, kemudian konversi setiap angka didalam tabel S1 yang berwarna putih menjadi biner, sehingga menjadi bentuk seperti dibawah:

S1 :

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Kemudian kita ambil sampel blok bit pertama dari A_1 yaitu **100110**

Kita pisahkan blok menjadi 2 yaitu:

- Bit pertama dan terakhir yaitu 1 dan 0 digabungkan menjadi 10
- Bit kedua hingga ke lima 0011

Kemudian dibandingkan dengan memeriksa perpotongan antara keduanya didapatkan nilai 1000(warna merah) dan seterusnya untuk blok kedua hingga blok kedelapan kita bandingkan dengan S2 hingga S8.

$B_1 = 1000\ 0101\ 0100\ 1000\ 0011\ 0010\ 1110\ 1010$
 $B_2 = 1101\ 1100\ 0100\ 0011\ 1000\ 0000\ 1111\ 1001$
 $B_3 = 1101\ 0110\ 0011\ 1100\ 1011\ 0110\ 0111\ 1111$
 $B_4 = 0010\ 1001\ 1101\ 0000\ 1011\ 1010\ 1111\ 1110$
 $B_5 = 0100\ 0001\ 0011\ 1101\ 1000\ 1010\ 1100\ 0011$
 $B_6 = 0110\ 1101\ 1101\ 1100\ 0011\ 0101\ 0100\ 0110$
 $B_7 = 1110\ 0011\ 0110\ 1011\ 0000\ 0101\ 0010\ 1101$
 $B_8 = 0000\ 1000\ 1101\ 1000\ 1000\ 0011\ 1101\ 0101$

$B_9 = 0110\ 1110\ 1110\ 0001\ 1010\ 1011\ 0100\ 1010$
 $B_{10} = 0010\ 0001\ 0111\ 0000\ 0100\ 0001\ 0110\ 1101$
 $B_{11} = 0101\ 1110\ 0000\ 1100\ 1101\ 1011\ 1100\ 0010$
 $B_{12} = 0110\ 1000\ 0000\ 1011\ 0011\ 0110\ 1010\ 1101$
 $B_{13} = 1111\ 1001\ 1101\ 1011\ 0010\ 0100\ 1011\ 0011$
 $B_{14} = 1011\ 1000\ 0111\ 1110\ 1100\ 0101\ 1100\ 0001$
 $B_{15} = 0100\ 0001\ 0011\ 1001\ 1111\ 0111\ 0010\ 0111$
 $B_{16} = 1000\ 0001\ 0110\ 1010\ 1111\ 0111\ 0100\ 1011$

Langkah 7

Setelah didapat vektor **Bi**, lalu permutasikan bit vektor **Bi** dengan tabel **P-Box**, lalu kelompokkan menjadi 4 blok dimana tiap-tiap blok memiliki 32 bit data

Tabel P-Box

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

Sehingga hasil yang didapat adalah sebagai berikut:

$P(B_1) = 00101000 \ 10110011 \ 01000100 \ 11010001$
 $P(B_2) = 10001011 \ 11011001 \ 10001100 \ 00010011$
 $P(B_3) = 01101111 \ 10110010 \ 10011100 \ 11111110$
 $P(B_4) = 00111111 \ 00111011 \ 01000111 \ 10100001$
 $P(B_5) = 10010101 \ 00110010 \ 11011000 \ 01000101$
 $P(B_6) = 00100100 \ 00011011 \ 11110011 \ 11111000$
 $P(B_7) = 11001000 \ 11000001 \ 11101110 \ 01101100$
 $P(B_8) = 00000111 \ 00111001 \ 00101001 \ 01100001$
 $P(B_9) = 11011001 \ 00111011 \ 10100011 \ 10010100$
 $P(B_{10}) = 00001100 \ 00010101 \ 01101110 \ 00100100$
 $P(B_{11}) = 01110001 \ 00111110 \ 10110000 \ 01010011$
 $P(B_{12}) = 10101000 \ 01101000 \ 10001110 \ 11101001$
 $P(B_{13}) = 10000110 \ 11001011 \ 11001111 \ 11001011$
 $P(B_{14}) = 00000101 \ 11011101 \ 00111010 \ 01001111$
 $P(B_{15}) = 10100101 \ 00100110 \ 11101100 \ 11101100$
 $P(B_{16}) = 00101001 \ 11110111 \ 01101000 \ 11001100$

Hasil $P(B_i)$ kemudian di XOR kan dengan L_{i-1} untuk mendapatkan nilai R_i . Sedangkan nilai L_i sendiri diperoleh dari Nilai R_{i-1} untuk nilai $1 \leq i \leq 16$.

$L_0 = 11111111 \ 10111000 \ 01110110 \ 01010111$
 $R_0 = 00000000 \ 00000000 \ 00000110 \ 10000011$

$P(B_1) = 00101000 \ 10110011 \ 01000100 \ 11010001$
 $L(1)-1 = 11111111 \ 10111000 \ 01110110 \ 01010111$

-----XOR

$R_1 = 11010111 \ 00001011 \ 00110010 \ 10000110$

$P(B_2) = 10001011 \ 11011001 \ 10001100 \ 00010011$
 $L(2)-1 = 00000000 \ 00000000 \ 00000110 \ 10000011$

-----XOR

$R_2 = 10001011 \ 11011001 \ 10001010 \ 10010000$

$P(B_3) = 01101111 \ 10110010 \ 10011100 \ 11111110$
 $L(3)-1 = 11010111 \ 00001011 \ 00110010 \ 10000110$

-----XOR

$R_3 = 10111000 \ 10111001 \ 10101110 \ 01111000$

$P(B_4) = 00111111 \ 00111011 \ 01000111 \ 10100001$
 $L(4)-1 = 10001011 \ 11011001 \ 10001010 \ 10010000$

-----XOR

$R_4 = 10110100 \ 11100010 \ 11001101 \ 00110001$

$P(B_5) = 10010101 \ 00110010 \ 11011000 \ 01000101$
 $L(5)-1 = 10111000 \ 10111001 \ 10101110 \ 01111000$

-----XOR

$R_5 = 00101101 \ 10001011 \ 01110110 \ 00111101$

P(B6) = 00100100 00011011 11110011 11111000
 L(6)-1 = 10110100 11100010 11001101 00110001
 -----XOR
 R6 = 10010000 11111001 00111110 11001001

P(B7) = 11001000 11000001 11101110 01101100
 L(7)-1 = 00101101 10001011 01110110 00111101
 -----XOR
 R7 = 11100101 01001010 10011000 01010001

P(B8) = 00000111 00111001 00101001 01100001
 L(8)-1 = 10010000 11111001 00111110 11001001
 -----XOR
 R8 = 10010111 11000000 00010111 10101000

P(B9) = 11011001 00111011 10100011 10010100
 L(9)-1 = 11100101 01001010 10011000 01010001
 -----XOR
 R9 = 00111100 01110001 00111011 11000101

P(B10) = 00001100 00010101 01101110 00100100
 L(10)-1 = 10010111 11000000 00010111 10101000
 -----XOR
 R10 = 10011011 11010101 01111001 10001100

P(B11) = 01110001 00111110 10110000 01010011
 L(11)-1 = 00111100 01110001 00111011 11000101
 -----XOR
 R11 = 01001101 01001111 10001011 10010110

P(B12) = 10101000 01101000 10001110 11101001
 L(12)-1 = 10011011 11010101 01111001 10001100
 -----XOR
 R12 = 00110011 10111101 11110111 01100101

P(B13) = 10000110 11001011 11001111 11001011
 L(13)-1 = 01001101 01001111 10001011 10010110
 -----XOR
 R13 = 11001011 10000100 01000100 01011101

P(B14) = 00000101 11011101 00111010 01001111
 L(14)-1 = 00110011 10111101 11110111 01100101
 -----XOR
 R14 = 00110110 01100000 11001101 00101010

P(B15) = 10100101 00100110 11101100 11101100
 L(15)-1 = 11001011 10000100 01000100 01011101
 -----XOR
 R15 = 01101110 10100010 10101000 10110001

P(B16) = 00101001 11110111 01101000 11001100
 L(16)-1 = 00110110 01100000 11001101 00101010
 -----XOR
R16 = 00011111 10010111 10100101 11100110

L 16 = 01101110 10100010 10101000 10110001

Langkah 8

Gabungkan **R₁₆** dengan **L₁₆** lalu permutasikan untuk terakhir kali dengan tabel Inverse Initial Permutation (**IP⁻¹**)

Tabel IP⁻¹

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

Sehingga Input :

$R_{16}L_{16} = 00011111\ 10010111\ 10100101\ 11100110\ 01101110\ 10100010\ 10101000\ 10110001$

Menghasilkan Output:

Cipher(dalam biner) = **01010110 11110001 11010101 11001000 01010010 10101111 10000001 00111111**

atau

Cipher(dalam hexa) = **56 f1 d5 c8 52 af 81 3f**

Dekripsi DES

- Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah K_1, K_2, \dots, K_{16} , maka pada proses dekripsi urutan kunci yang digunakan adalah $K_{16}, K_{15}, \dots, K_1$.
- Untuk tiap putaran 16, 15, ..., 1, keluaran pada setiap putaran *deciphering* adalah

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

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

yang dalam hal ini, (R_{16}, L_{16}) adalah blok masukan awal untuk deciphering. Blok (R_{16}, L_{16}) diperoleh dengan mempermutasikan cipherteks dengan matriks permutasi IP^{-1} . Pra-keluaran dari deciphering adalah (L_0, R_0) . Dengan permutasi awal IP akan didapatkan kembali blok plainteks semula.

Dekripsi DES

- Tinjau kembali proses pembangkitan kunci internal pada Gambar 4. Selama *deciphering*, K_{16} dihasilkan dari (C_{16}, D_{16}) dengan permutasi PC-2. Tentu saja (C_{16}, D_{16}) tidak dapat diperoleh langsung pada permulaan *deciphering*. Tetapi karena $(C_{16}, D_{16}) = (C_0, D_0)$, maka K_{16} dapat dihasilkan dari (C_0, D_0) tanpa perlu lagi melakukan pergeseran bit. Catatlah bahwa (C_0, D_0) yang merupakan bit-bit dari kunci eksternal K yang diberikan pengguna pada waktu dekripsi.
- Selanjutnya, K_{15} dihasilkan dari (C_{15}, D_{15}) yang mana (C_{15}, D_{15}) diperoleh dengan menggeser C_{16} (yang sama dengan C_0) dan D_{16} (yang sama dengan C_0) satu bit ke kanan. Sisanya, K_{14} sampai K_1 dihasilkan dari (C_{14}, D_{14}) sampai (C_1, D_1) . Catatlah bahwa (C_{i-1}, D_{i-1}) diperoleh dengan menggeser C_i dan D_i , tetapi pergeseran kiri (*left shift*) diganti menjadi pergeseran kanan (*right shift*).

Mode Operasi DES

- ❑ DES dapat dioperasikan dengan mode ECB, CBC, OFB, dan CFB. Namun karena kesederhanaannya, mode ECB lebih sering digunakan pada paket program komersil meskipun sangat rentan terhadap serangan.
- ❑ Mode CBC lebih kompleks daripada EBC namun memberikan tingkat keamanan yang lebih bagus daripada mode EBC. Mode CBC hanya kadang-kadang saja digunakan.

Implementasi Hardware dan Software DES

- DES sudah diimplementasikan dalam bentuk perangkat keras.
- Dalam bentuk perangkat keras, DES diimplementasikan di dalam *chip*. Setiap detik *chip* ini dapat mengenkripsikan 16,8 juta blok (atau 1 gigabit per detik).
- Implementasi DES ke dalam perangkat lunak dapat melakukan enkripsi 32.000 blok per detik (pada komputer *mainframe* IBM 3090).

Keamanan DES

- Isu-isu yang menjadi perdebatan kontroversial menyangkut keamanan DES:
 1. Panjang kunci
 2. Jumlah putaran
 3. Kotak-S

Panjang kunci DES

- Panjang kunci eksternal DES hanya 64 bit atau 8 karakter, itupun 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. Alasan pengurangan tidak diumumkan.
- Tetapi, dengan panjang kunci 56 bit akan terdapat 2^{56} atau 72.057.594.037.927.936 kemungkinan kunci. Jika diasumsikan serangan *exhaustive key search* dengan menggunakan prosesor paralel mencoba setengah dari jumlah kemungkinan kunci itu, maka dalam satu detik dapat dikerjakan satu juta serangan. Jadi seluruhnya diperlukan 1142 tahun untuk menemukan kunci yang benar.
- Tahun 1998, *Electronic Frontier Foundation* (EFE) merancang dan membuat perangkat keras khusus untuk menemukan kunci DES secara *exhaustive search key* 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.

Jumlah putaran DES

- Sebenarnya, delapan putaran sudah cukup untuk membuat cipherteks sebagai fungsi acak dari setiap bit plainteks dan setiap bit cipherteks. Jadi, mengapa harus 16 kali putaran?
- Dari penelitian, DES dengan jumlah putaran yang kurang dari 16 ternyata dapat dipecahkan dengan *known-plaintext attack* lebih mangkus daripada dengan *brute force attack*.

Kotak-S

- Pengisian kotak-S DES masih menjadi misteri tanpa ada alasan mengapa memilih konstanta-konstanta di dalam kotak itu.

GOST

- GOST = Gosudarstvenny Standard, artinya standard pemerintah,
- GOST adalah algoritma enkripsi dari negara Uni Soviet dahulu
- Dikembangkan pada tahun 1970.
- Dibuat oleh Soviet sebagai alternatif terhadap algoritma enkripsi standard Amerika Serikat, *DES*.
- GOST secara struktural mirip dengan DES

GOST

- Ukuran blok pesan = 64 bit
- Panjang kunci = 256 bit
- Jumlah putaran = 32 putaran
- Setiap putaran menggunakan kunci internal.
- Kunci internal sebenarnya hanya ada 8 buah, K_1 sampai K_8 ,
- Karena ada 32 putaran, maka 8 buah kunci internal ini dijadwalkan penggunaannya.

GOST

Tabel Penjadwalan kunci internal GOST

Putaran	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Kunci internal	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8

Putaran	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Kunci internal	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_8	K_7	K_6	K_5	K_4	K_3	K_2	K_1

GOST

- Pembangkitan kunci internal sangat sederhana.
- Kunci eksternal yang panjangnya 256 bit dibagi ke dalam delapan bagian yang masing-masing panjangnya 32 bit.
- Delapan bagian ini yang dinamakan K_1, K_2, \dots, K_8 .

GOST

□ GOST menggunakan Jaringan *Feistel*

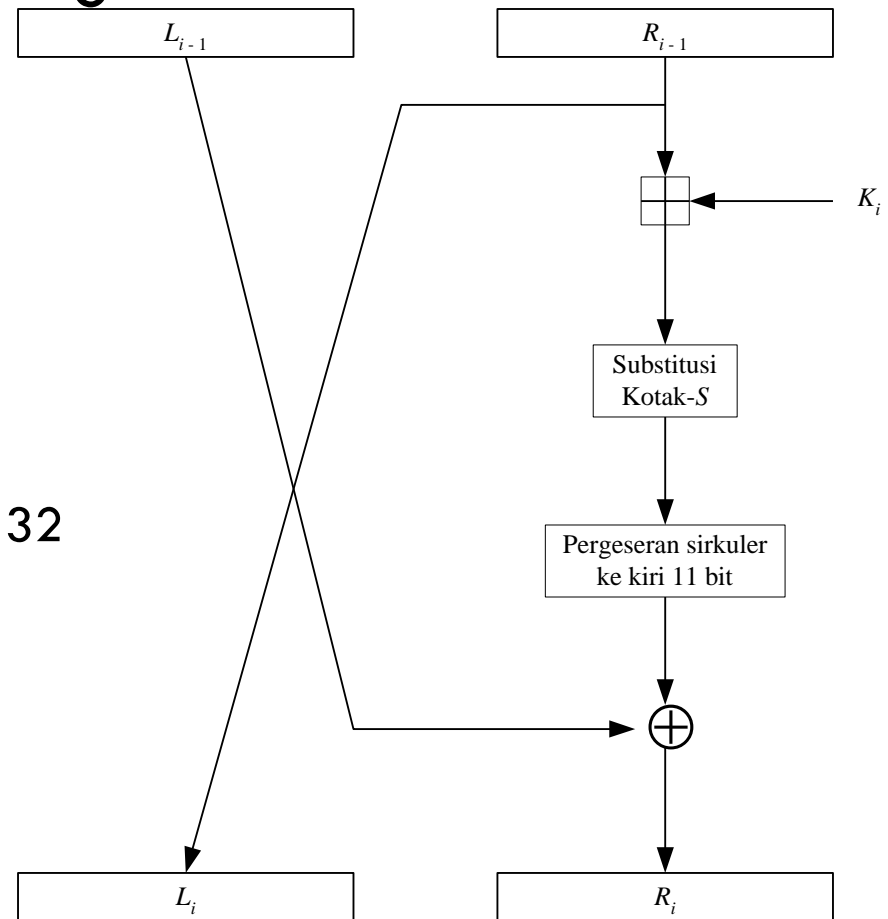
□ Satu putaran GOST:


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

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

□ Fungsi f terdiri dari

- penjumlahan modulo 2^{32}
- substitusi
- pergeseran



Keterangan:  adalah operator penjumlahan dalam modulo 2^{32}

GOST

- Hasil penjumlahan R_{i-1} dengan kunci internal ke- i menghasilkan luaran yang panjangnya 32 bit.
- Luaran ini dibagi mejadi 8 bagian yang masing-masing panjangnya 4 bit.
- Setiap 4 bit masuk ke dalam kotak S untuk proses substitusi. Empat bit pertama masuk ke dalam kotak S pertama, 4 bit kedua masuk ke dalam kotak S kedua, demikian seterusnya.
- Hasil substitusi setiap kotak S adalah 4 bit. GOST memiliki 8 buah kotak S , setiap kotak berisi 16 buah elemen nilai. Setiap kotak berisi permutasi angka 0 sampai 15.

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

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

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

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

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

Kotak-S 6:															
4	11	10	0	7	2	1	13	3	6	8	5	9	12	14	14

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

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

GOST

- Misalnya pada kotak S pertama, masukannya:

0000 (nilai desimal 0)

maka luarannya: nilai di dalam elemen ke-0:

4 atau 0100

- Hasil substitusi dari semua kotak S ini digabung menjadi pesan 32-bit, kemudian pesan 32-bit ini digeser ke kiri sejauh 11 bit secara sirkuler.
- Hasilnya kemudian di-XOR-kan dengan L_{i-1} untuk kemudian memberikan bagian cipherteks kanan yang baru, R_i . Proses ini diulang sebanyak 32 kali.

GOST vs DES

Perbedaan GOST dengan DES:

1. Kunci *DES* 56 bit, sedangkan kunci GOST lebih panjang yaitu 256 bit. Ini menyebabkan *exhaustive key search* terhadap GOST lebih sukar dibandingkan dengan *DES*.
2. Jumlah putaran DES 16 kali, sedangkan GOST lebih banyak yaitu 32 kali sehingga membuat kriptanalisis menjadi sangat sulit
3. Kotak *S* di dalam *DES* menerima masukan 6 bit dan luaran 4 bit (berukuran 6×4), sedangkan kotak *S* di dalam GOST menerima masukan 4 bit dan luaran 4 bit (berukuran 4×4)
4. Pembangkitan kunci internal *DES* rumit, sedangkan di dalam GOST pembangkitan kunci internalnya sederhana
5. DES mempunyai permutasi yang tidak teratur, sedangkan GOST hanya menggunakan pergeseran 11-bit secara sirkuler

GOST

- GOST adalah *cipher* yang sangat aman. Hal ini mungkin disebabkan jumlah putaran dan panjang kunci yang lebih banyak dari DES.
- Belum ada publikasi kriptanalisis tentang GOST.

DES Berganda (Multiple DES)

- Karena DES mempunyai potensi kelemahan pada *brute force attack*, maka dibuat varian dari DES.
- Varian DES yang paling luas digunakan adalah DES berganda (*multiple DES*).
- DES berganda adalah enkripsi berkali-kali dengan DES dan menggunakan kunci ganda.

DES Berganda (Multiple DES)

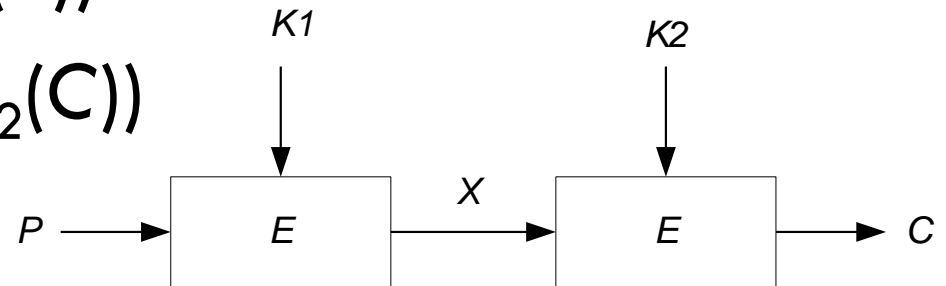
- Tinjau DES berganda:

1. *Double DES*

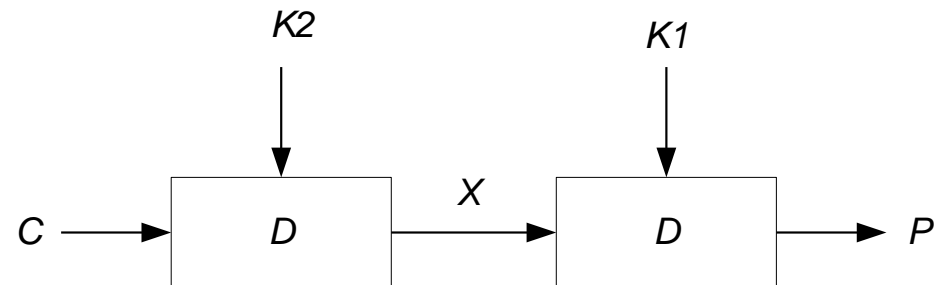
2. *Triple DES*

Double DES

- Menggunakan 2 buah kunci eksternal, K_1 dan K_2 .
- Enkripsi: $C = E_{K_2}(E_{K_1}(P))$
- Dekripsi: $P = D_{K_1}(D_{K_2}(C))$



Enkripsi



Dekripsi

Double DES

- Kelemahan *Double DES*: serangan *meet-in-the-middle attack*:

- Dari pengamatan,

$$C = E_{K_2}(E_{K_1}(P))$$

maka

$$X = E_{K_1}(P) = D_{K_2}(C)$$

- Misalkan kriptanalisis memiliki potongan C dan P yang berkoreponden.
- Enkripsi P untuk semua kemungkinan nilai K_1 (yaitu sebanyak 2^{56} kemungkinan kunci). Hasilnya adalah semua nilai X
- Simpan semua nilai X ini di dalam tabel

Double DES

- Berikutnya, dekripsi C dengan semua semua kemungkinan nilai K_2 (yaitu sebanyak 2^{56} kemungkinan kunci).
- Bandingkan semua hasil dekripsi ini dengan elemen di dalam tabel tadi. Jika ada yang sama, maka dua buah kunci, K_1 dan K_2 , telah ditemukan.
- Tes kedua kunci ini dengan pasangan plainteks-cipherteks lain yang diketahui. Jika kedua kunci tersebut menghasilkan cipherteks atau plainteks yang benar, maka K_1 dan K_2 tersebut merupakan kunci yang benar

Triple DES

- Menggunakan DES tiga kali
- Bertujuan untuk mencegah *meet-in-the-middle attack*.

- Bentuk umum TDES (mode EEE):

Enkripsi: $C = E_{K3}(E_{K2}(E_{K1}(P)))$

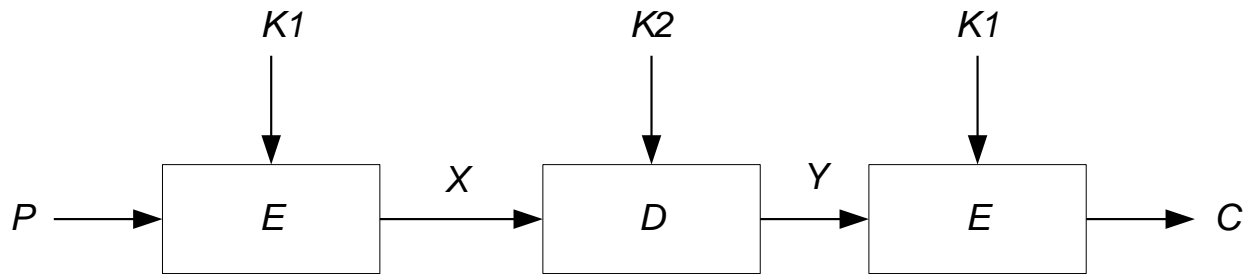
Dekripsi: $P = D_{K1}(D_{K2}(D_{K3}(C)))$

Triple DES

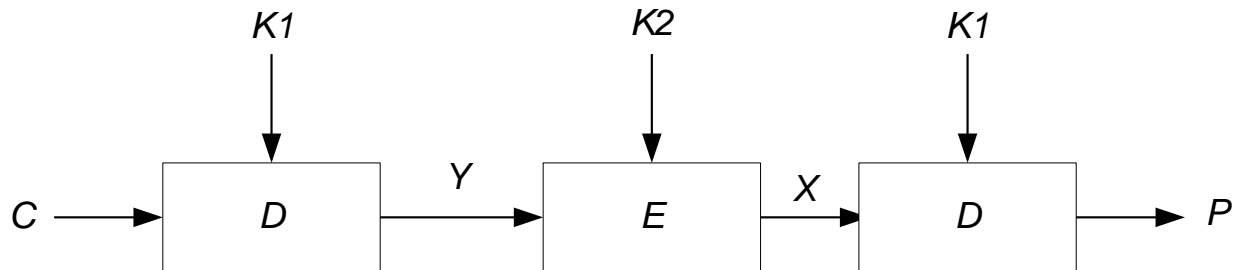
- Untuk menyederhanakan TDES, maka langkah di tengah diganti dengan D (mode EDE).
- Ada dua versi TDES dengan mode EDE:
 - Menggunakan 2 kunci
 - Menggunakan 3 kunci

Triple DES

□ Triple DES dengan 2 kunci



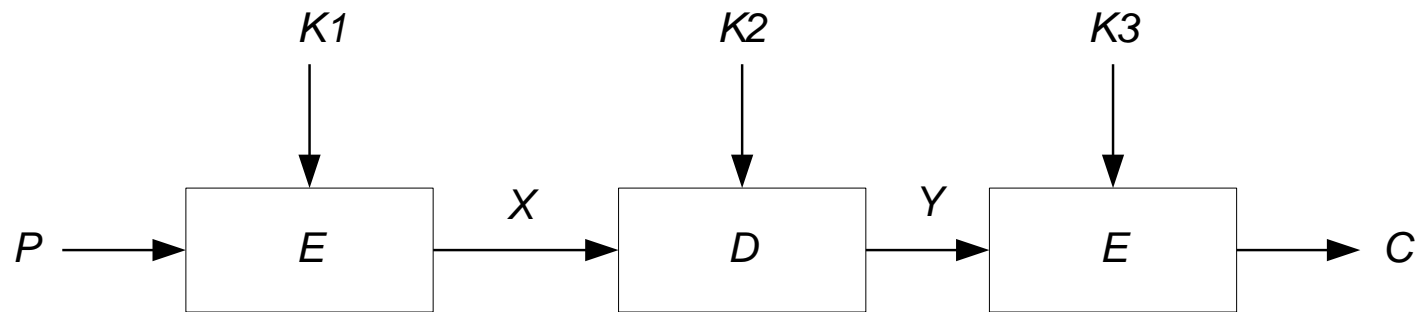
Enkripsi



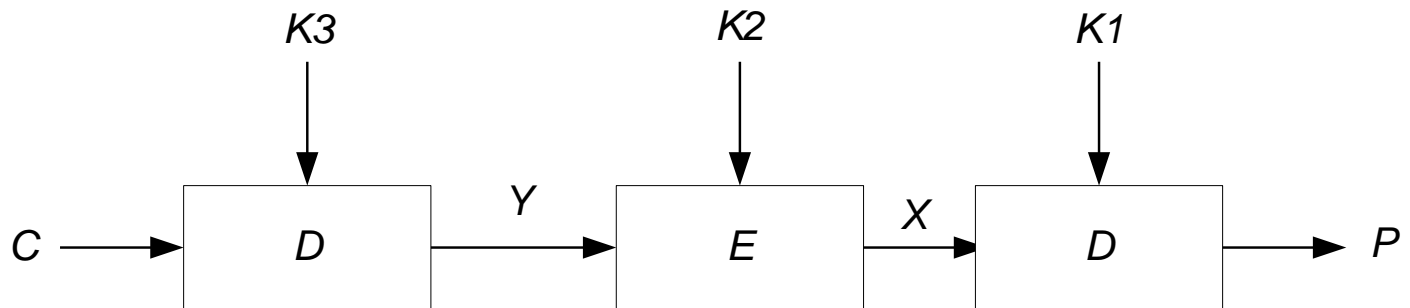
Dekripsi

Triple DES

□ Triple DES dengan 3 kunci



Enkripsi



Dekripsi