

*Bahan Kuliah ke-12*

**IF5054 Kriptografi**

***Data Encryption Standard (DES)***

**Disusun oleh:**

**Ir. Rinaldi Munir, M.T.**

**Departemen Teknik Informatika  
Institut Teknologi Bandung  
2004**

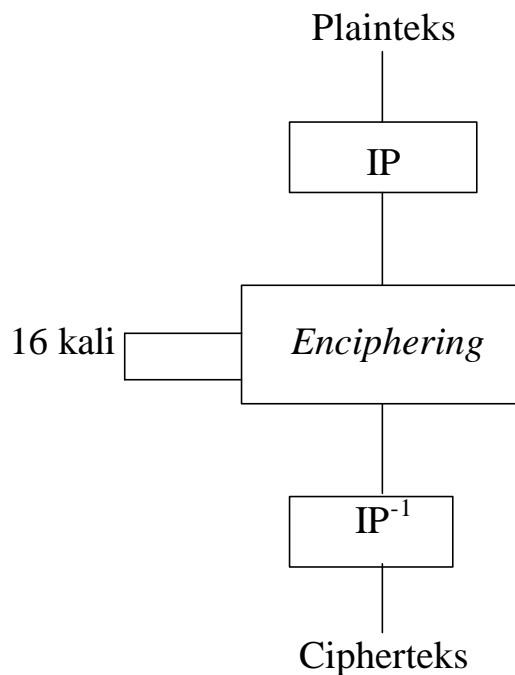
## 12. *Data Encryption Standard (DES)*

### 12.1 Sejarah DES

- Algoritma DES dikembangkan di IBM dibawah kepemimpinan W.L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma *Lucifer* yang dibuat oleh Horst Feistel.
- Algoritma ini telah disetujui oleh *National Bureau of Standard (NBS)* setelah penilaian kekuatannya oleh *National Security Agency (NSA)* Amerika Serikat.

### 12.2 Tinjauan Umum

- 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 upa-kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) yang panjangnya 64 bit.
- Skema global dari algoritma DES adalah sebagai berikut (lihat Gambar 12.1):
  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.

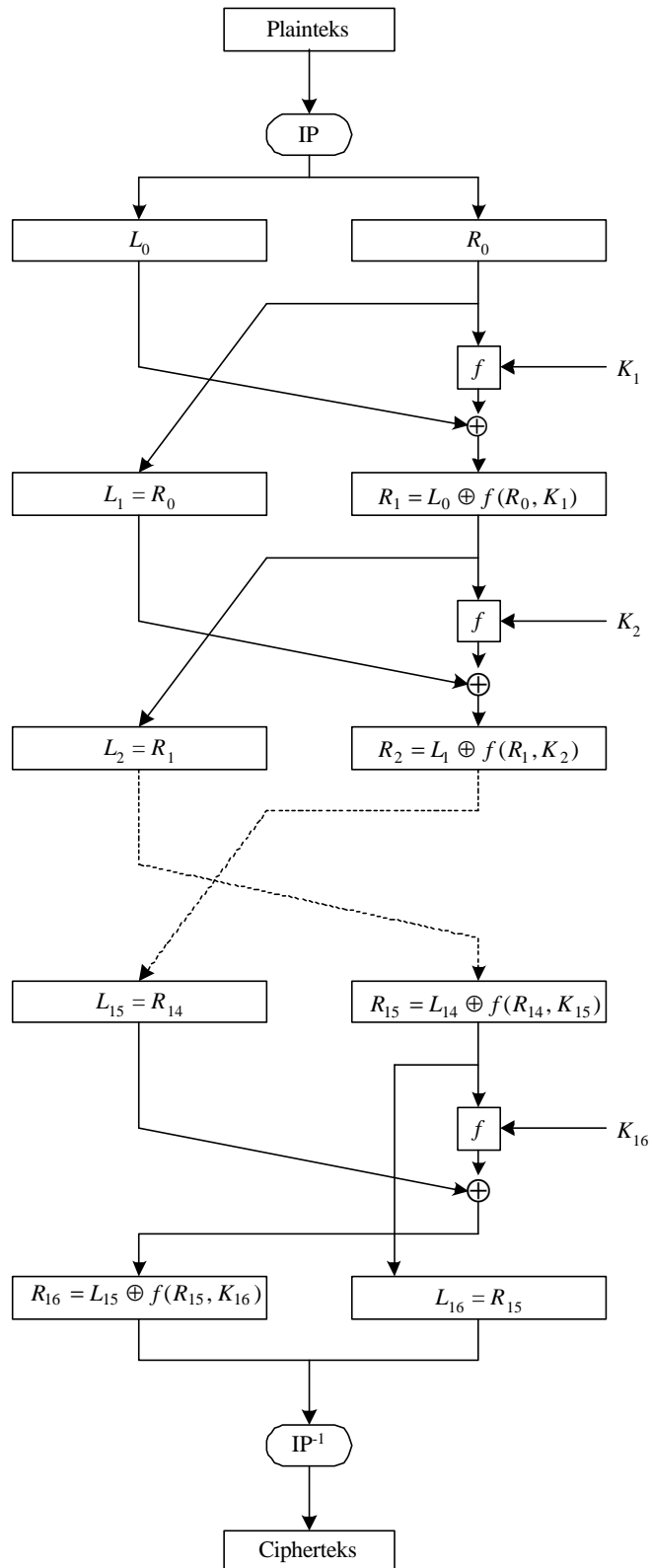


**Gambar 12.1** Skema Global Algoritma DES

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

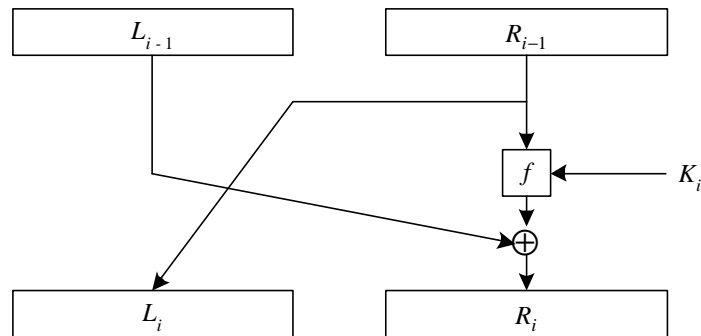
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Gambar 12.2 memperlihatkan skema algoritma DES yang lebih rinci.



**Gambar 12.2** Algoritma Enkripsi dengan DS

- Satu putaran DES merupakan model jaringan *Feistel* (lihat Gambar 12.3).



**Gambar 12.3.** Jaringan *Feistel* untuk satu putaran DES

- Perlu dicatat dari Gambar 12.2 bahwa jika  $(L_{16}, R_{16})$  merupakan keluaran dari putaran ke-16, maka  $(R_{16}, L_{16})$  merupakan pra-cipherteks (*pre-ciphertext*) dari *enciphering* ini. Cipherteks yang sebenarnya diperoleh dengan melakukan permutasi awal balikan,  $IP^{-1}$ , terhadap blok pra-cipherteks.

### 12.3 Permutasi Awal

- 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 ini:

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

Cara membaca tabel/matriks: dua *entry* ujung kiri atas (58 dan 50) berarti: “pindahkan bit ke-58 ke posisi bit 1”  
 “pindahkan bit ke-50 ke posisi bit 2”, dst

## 12.4 Pembangkitan Kunci Internal

- Karena ada 16 putaran, maka dibutuhkan kunci internal sebanyak 16 buah, yaitu  $K_1, K_2, \dots, K_{16}$ . Kunci-kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi.
- Kunci internal dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Kunci eksternal panjangnya 64 bit atau 8 karakter.
- Misalkan kunci eksternal yang tersusun dari 64 bit adalah  $K$ . Kunci eksternal ini menjadi masukan untuk permutasi dengan menggunakan matriks permutasi kompresi PC-1 sebagai berikut:

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

Dalam permutasi ini, tiap bit kedelapan (*parity bit*) dari delapan *byte* kunci diabaikan. Hasil permutasinya adalah sepanjang 56 bit, sehingga dapat dikatakan panjang kunci DES adalah 56 bit.

Selanjutnya, 56 bit ini dibagi menjadi 2 bagian, kiri dan kanan, yang masing-masing panjangnya 28 bit, yang masing-masing disimpan di dalam  $C_0$  dan  $D_0$ :

$C_0$ : 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

Selanjutnya, kedua bagian digeser ke kiri (*left shift*) sepanjang satu atau dua bit bergantung pada tiap putaran. Operasi pergeseran bersifat *wrapping* atau *round-shift*. Jumlah pergeseran pada setiap putaran ditunjukkan pada Tabel 1 sbb:

**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

Misalkan  $(C_i, D_i)$  menyatakan penggabungan  $C_i$  dan  $D_i$ .  $(C_{i+1}, D_{i+1})$  diperoleh dengan menggeser  $C_i$  dan  $D_i$  satu atau dua bit.

Setelah pergeseran bit,  $(C_i, D_i)$  mengalami permutasi kompresi dengan menggunakan 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

Dengan permutasi ini, kunci internal  $K_i$  diturunkan dari ( $C_i$ ,  $D_i$ ) yang dalam hal ini  $K_i$  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

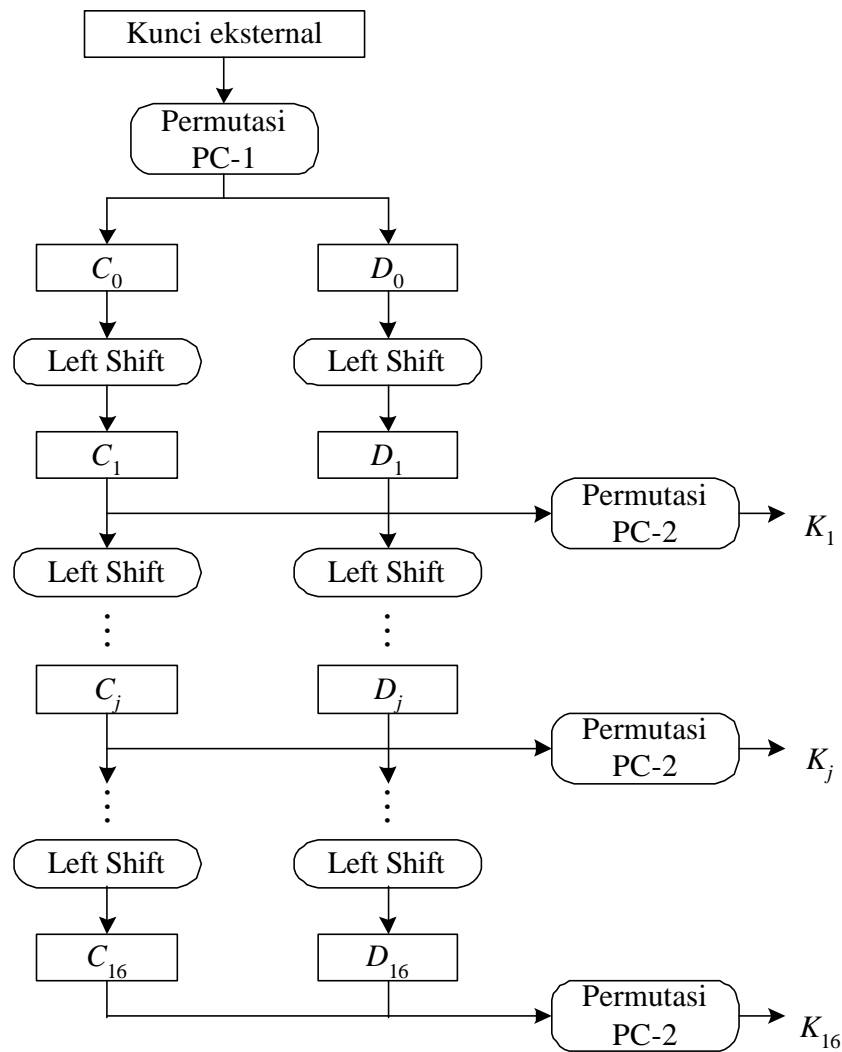
dengan bit-bit  $D_i$  pada posisi:

41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48  
 44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32

Jadi, setiap kunci internal  $K_i$  mempunyai panjang 48 bit.

Proses pembangkitan kunci-kunci internal ditunjukkan pada Gambar 12.4.

- Bila jumlah pergeseran bit-bit pada Tabel 1 dijumlahkan semuanya, maka jumlah seluruhnya sama dengan 28, yang sama dengan jumlah bit pada  $C_i$  dan  $D_i$ . Karena itu, setelah putaran ke-16 akan didapatkan kembali  $C_{16} = C_0$  dan  $D_{16} = D_0$ .



**Gambar 12.4.** Proses pembangkitan kunci-kunci internal DES

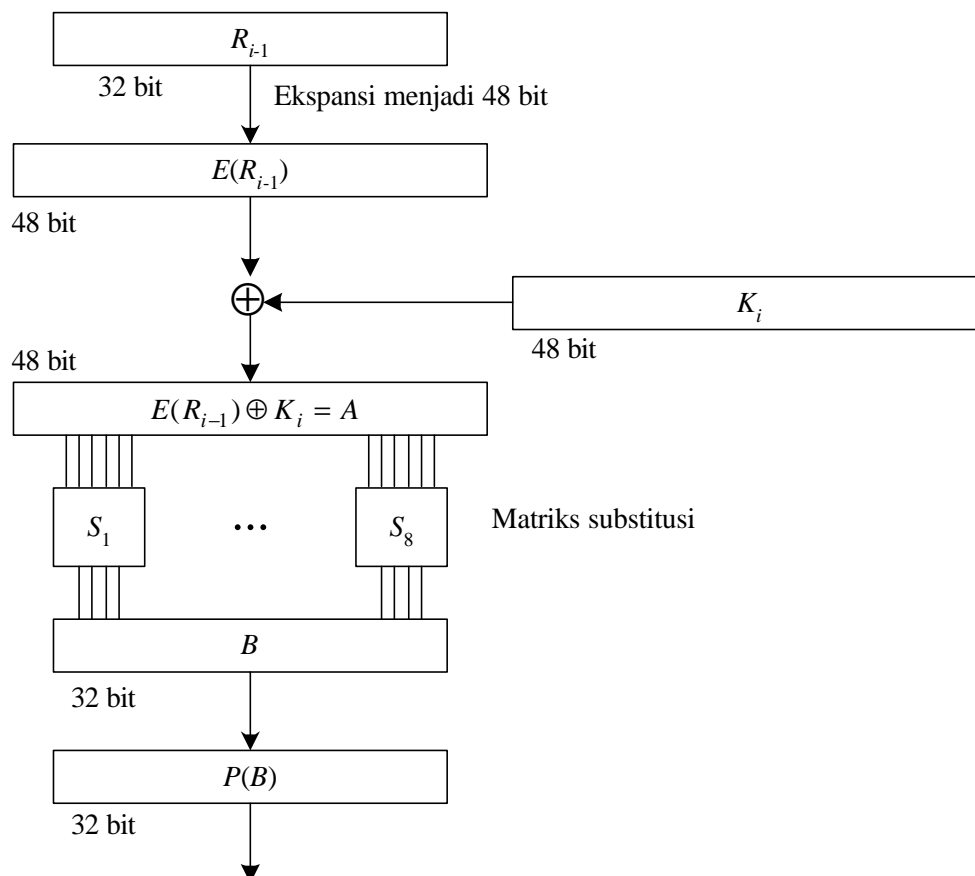
## 12.5 Enciphering

- Proses *enciphering* terhadap blok plainteks dilakukan setelah permutasi awal (lihat Gambar 12.1). Setiap blok plainteks mengalami 16 kali putaran *enciphering* (lihat Gambar 2). Setiap putaran *enciphering* merupakan jaringan Feistel yang secara matematis dinyatakan sebagai

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

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

Diagram komputasi fungsi  $f$  diperlihatkan pada Gambar 12.5.



**Gambar 12.5.** Rincian komputasi fungsi  $f$

- $E$  adalah fungsi ekspansi yang memperluas blok  $R_{i-1}$  yang panjangnya 32-bit menjadi blok 48 bit. Fungsi ekspansi direalisasikan dengan matriks permutasi ekspansi sbb:

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

- Selanjutnya, hasil ekspansi, yaitu  $E(R_{i-1})$ , yang panjangnya 48 bit di-XOR-kan dengan  $K_i$  yang panjangnya 48 bit menghasilkan vektor  $A$  yang panjangnya 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. Proses substitusi dilakukan dengan menggunakan delapan buah kotak-S (*S-box*),  $S_1$  sampai  $S_8$ . Setiap kotak-S menerima masukan 6 bit dan menghasilkan keluaran 4 bit. Kelompok 6-bit pertama menggunakan  $S_1$ , kelompok 6-bit kedua menggunakan  $S_2$ , dan seterusnya.  
(cara pensubstitusian dengan kotak-S sudah dijelaskan pada materi “Tipe dan Mode Algoritma Simetri (Bagian 3)”)

Kedelapan kotak-S tersebut adalah:

$S_1$ :

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:

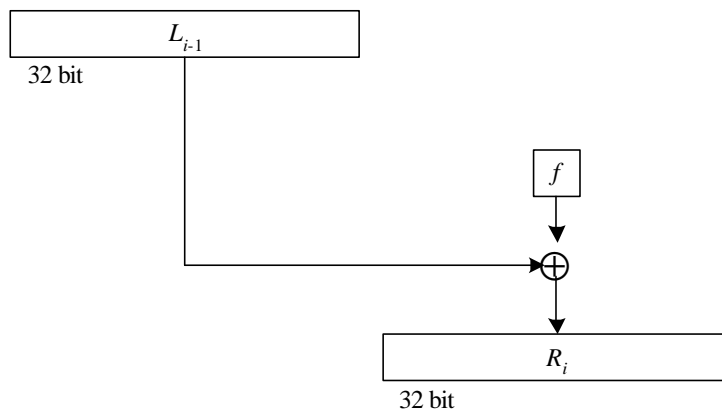
16	7	20	21	29	12	28	17	1	15	23	26	5	8	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

- Bit-bit  $P(B)$  merupakan keluaran dari fungsi  $f$ .
- Akhirnya, bit-bit  $P(B)$  di- $XOR$ -kan dengan  $L_{i-1}$  untuk mendapatkan  $R_i$  (lihat Gambar 6):

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

- Jadi, keluaran dari putaran ke- $i$  adalah

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



**Gambar 12.6** Skema perolehan  $R_i$

## Permutasi Terakhir (*Inverse Initial Permutation*)

- Permutasi terakhir dilakukan setelah 16 kali putaran terhadap gabungan blok kiri dan blok kanan.
- Proses permutasi menggunakan matriks permutasi awal balikan (*inverse initial permutation* atau  $IP^{-1}$ ) 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

## 12.7 Dekripsi

- 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.

- 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  $D_i$  dengan cara yang sama seperti pada Tabel 1, tetapi pergeseran kiri (*left shift*) diganti menjadi pergeseran kanan (*right shift*).

## 12.8 Mode 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.

## 12.9 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).

## 12.10 Keamanan DES

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

### *Panjang kunci*

- 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*

- 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.

## Kunci Lemah dan Kunci Setengah Lemah

- DES mempunyai beberapa kunci lemah (*weak key*). Kunci lemah menyebabkan kunci-kunci internal pada setiap putaran sama ( $K_1 = K_2 = \dots = K_{16}$ ). Akibatnya, enkripsi dua kali berturut-turut terhadap plainteks menghasilkan kembali plainteks semula.
- Kunci lemah terjadi bila bit-bit di dalam  $C_i$  dan  $D_i$  semuanya 0 atau 1, atau setengah dari kunci seluruh bitnya 1 dan setengah lagi seluruhnya 0.
- Kunci eksternal (dalam notasi HEX) yang menyebabkan terjadinya kunci lemah adalah (ingat bahwa setiap bit kedelapan adalah bit paritas).

Kunci lemah (dengan bit paritas)	Kunci sebenarnya
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 1F1F 1F1F	0000000 FFFFFFFF
E0E0 E0E0 F1F1 F11F	FFFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFFF FFFFFFFF

- Selain kunci lemah, DES juga mempunyai sejumlah pasangan kunci setengah-lemah (*semiweak key*). Pasangan kunci setengah-lemah mengenkripsikan plainteks menjadi cipherteks yang sama. Sehingga, satu kunci dalam pasangan itu dapat mendekripsi pesan yang dienkripsi oleh kunci yang lain di dalam pasangan itu.
- Kunci setengah-lemah terjadi bila:
  1. Register  $C$  dan  $D$  berisi bit-bit dengan pola 0101...0101 atau 1010...1010
  2. Register yang lain ( $C$  atau  $D$ ) berisi bit-bit dengan pola 0000...0000, 1111...1111, 0101...0101, atau 1010...1010

- Ada 6 pasang kunci setengah lemah (dalam notasi HEX):
  - a. 01FE 01FE 01FE 01FE dan FE01 FE01 FE01 FE01
  - b. 1FE0 1FE0 0EF1 0EF1 dan E01F E01F F10E F10E
  - c. 01E0 01E0 01F1 01F1 dan E001 E001 F101 F101
  - d. 1FFE 1FFE 0EFE 0EFE dan FE1F FE1F FE0E FE0E
  - e. 011F 011F 010E 010E dan 1F01 1F01 0E01 0E01
  - f. E0FE E0FE F1FE F1FE dan FEE0 FEE0 FEF1 FEF1