MODUL KRIPTOGRAFI

(CTI 312)

MODUL 5

DATA ENCRYPTION STANDARD (DES)

DISUSUN OLEH

IR. NIZIRWAN ANWAR, M.T

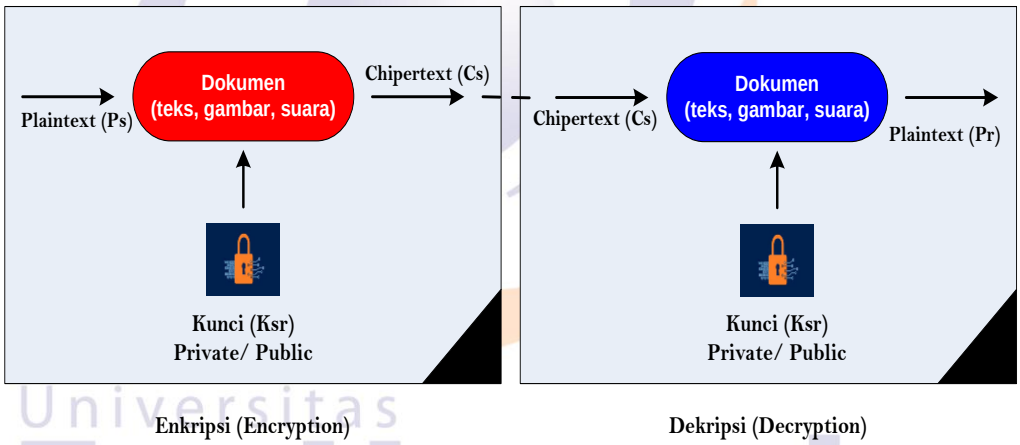
UNIVERSITAS ESA UNGGUL

2020

MODUL 5
TEKNIK KRIPTOGRAFI KLASIK, DES (*DATA ENCRYPTION STANDARD*)

1. PENDAHULUAN

Data Encryption Standard (DES) atau Standar Enkripsi Data adalah algoritma kunci-simetris (**gambar 1**) untuk enkripsi data elektronik, dimana kunci enkripsi sama dengan dekripsi. DES dan merupakan salah satu algoritma kriptografi cipher block dengan ukuran block 64 bit dan ukuran kuncinya 56 bit. Algoritma DES dibuat di IBM, dan merupakan modifikasi daripada algoritma terdahulu yang bernama **Lucifer**. Lucifer merupakan algoritma cipher block yang beroperasi pada block masukan 64 bit dan kuncinya berukuran 28 bit.



Gambar 5.1 Algoritma Kunci Simetris

DES pertama kali dipublikasikan di **Federal Register pada 17 Maret 1975**, dengan mengeluarkan rekomendasi kriteria algoritme harus memberikan tingkat keamanan yang tinggi sebagai berikut ;

1) Algoritma harus ditentukan secara lengkap dan mudah dimengerti.

2) Keamanan algoritma harus berada pada kunci keamanan seharusnya tidak bergantung pada kerahasiaan algoritma.

3) Algoritma harus tersedia untuk semua pengguna.

4) Algoritma harus dapat beradaptasi untuk digunakan dalam beragam aplikasi.

5) Algoritma harus dapat diterapkan secara elektronik secara elektronik perangkat.

6) Algoritma harus efisien untuk digunakan., dapat divalidasi dan diekspor.

Setelah melalui proses pembahasan atau diskusi yang panjang banyak diskusi, akhirnya algortima DES di-adopsi sebagai algoritma standar yang digunakan oleh **NBS (National Bureau of Standards) pada 15 Januari 1977**. Sejak saat itu, DES banyak digunakan pada dunia penyebaran informasi untuk melindungi data agar tidak bisa dibaca oleh orang lain. Salah satu hal yang harus diperhatikan dan digunakan dalam DES. S-Box merupakan bagian esensi atau penting dari DES karena merupakan bagian yang paling sulit dalam memproses enkripsi. Hal ini disebabkan karena S-Box merupakan satu – satunya bagian dari DES yang komputasinya tidak linear (non-linear) dan bersifat private.

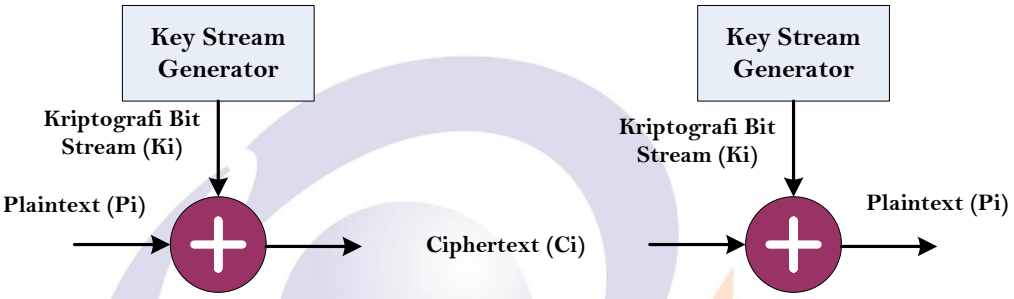
Dan kedua adalah jumlah bit pada kunci DES yang dianggap terlalu kecil, hanya 56 bit. Akibatnya DES rawan terhadap serangan **Brute Force**, adalah sebuah teknik serangan terhadap sebuah sistem keamanan komputer yang menggunakan uji coba terhadap semua kunci yang mungkin. DES mempunyai performance mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal (internal key) atau sub-kunci (sub-key). Kunci internal dibangkitkan dari kunci eksternal (external key) yang panjangnya 64 bit.

2. PRINSIP BLOCK CIPHER

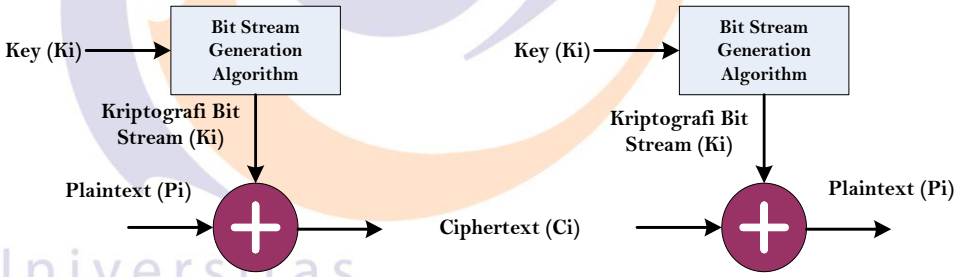
Terdapat banyak-nya algoritma enkripsi block simetris yang digunakan saat ini didasarkan pada struktur yang disebut sebagai cipher block Feistel [FEIS73]. Untuk alasan itu, penting untuk melakukannya memeriksa prinsip-prinsip desain cipher Feistel. Kami

akan mendefinsikan secara singkat saja perbandingan **Stream Cipher** dan **Block Cipher.**

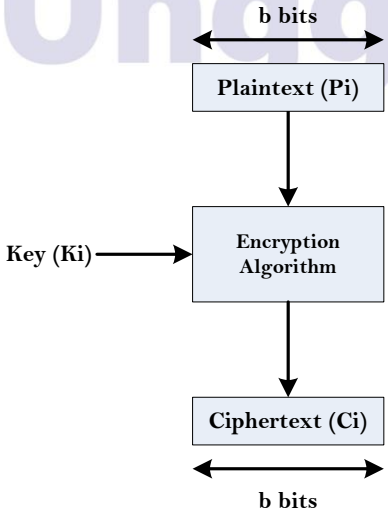
Stream cipher adalah adalah algoritma deterministik yang beroperasi pada kelompok bit dengan panjang tetap, yang disebut blok.yang mengenkripsi aliran data digital satu bit atau satu byte pada satu waktu. Contoh dari stream cipher klasik adalah cipher Vigenère yang diautoksi dan Sandi Vernam.



Gambar 5.2 Cipher Vernam



Gambar 5.3 Stream Cipher



Gambar 5.4 Block Cipher

Berikut contoh dalam menggambarkan transformasi non-singular dan singular untuk. n = 2

Reversible Mapping

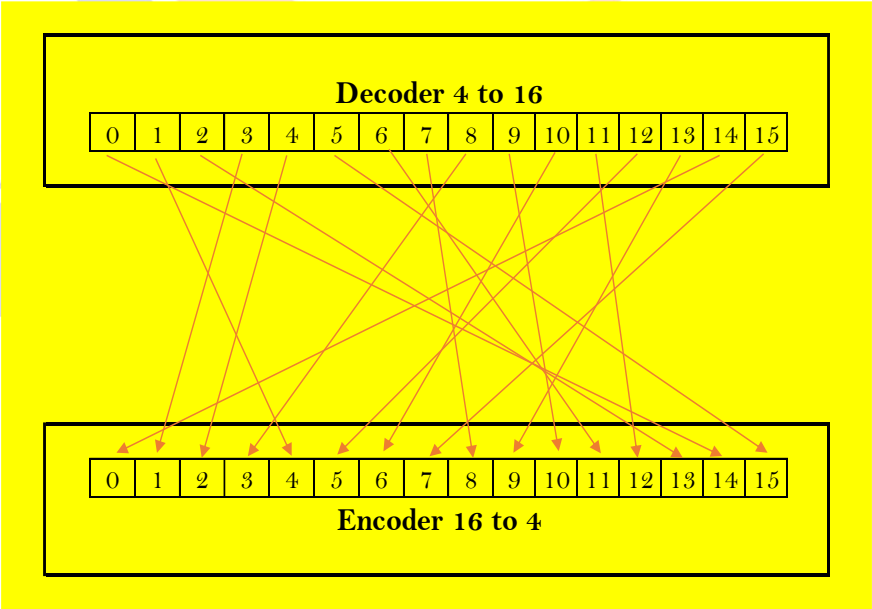
Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

Gambar 5.5 Mapping, Reversible dan Irreversible

Gambar 5.6 menggambarkan logika cipher substitusi umum untuk. masukan 4-bit menghasilkan dari 16 status masukan (decoder) yang dipetakan oleh cipher substitusi menjadi salah satu yang unik dari 16 status luaran, masing-masing adalah diwakili oleh 4 bit ciphertext (encoder).



Gambar 5.6 Block Substitusi Umum untuk 4-bit

Gambar 5.6 dapat dibuatkan menjadi suatu model tabulasi di bawah ini

Table 5.1 Substitusi Ciphertext, Enkripsi dan Dekripsi

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

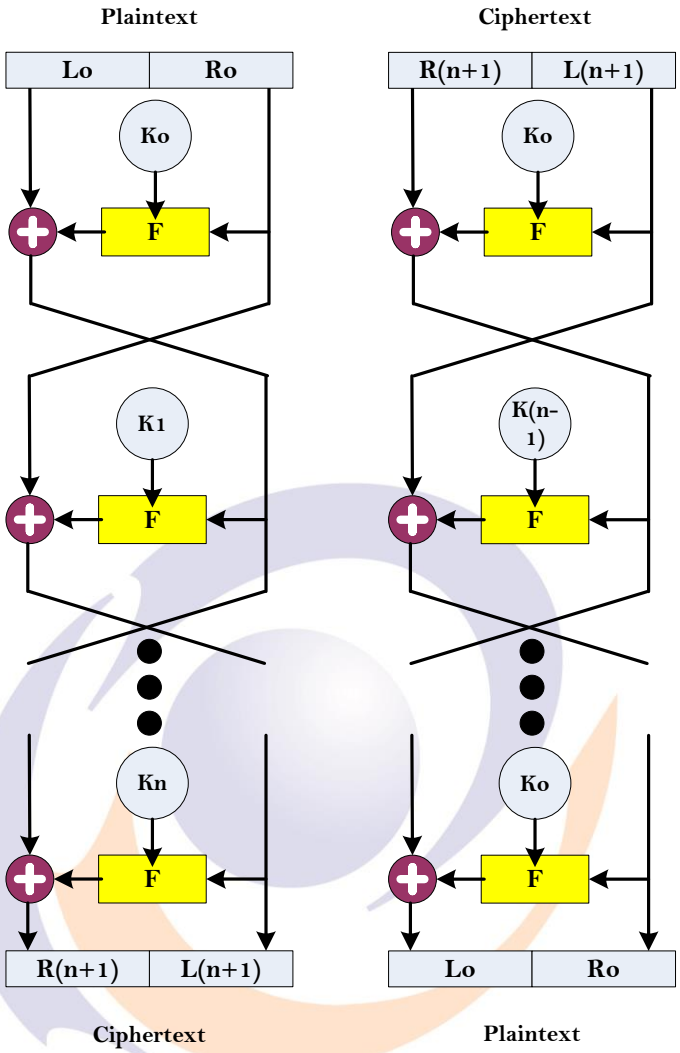
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

3. CIPHER FEISTEL

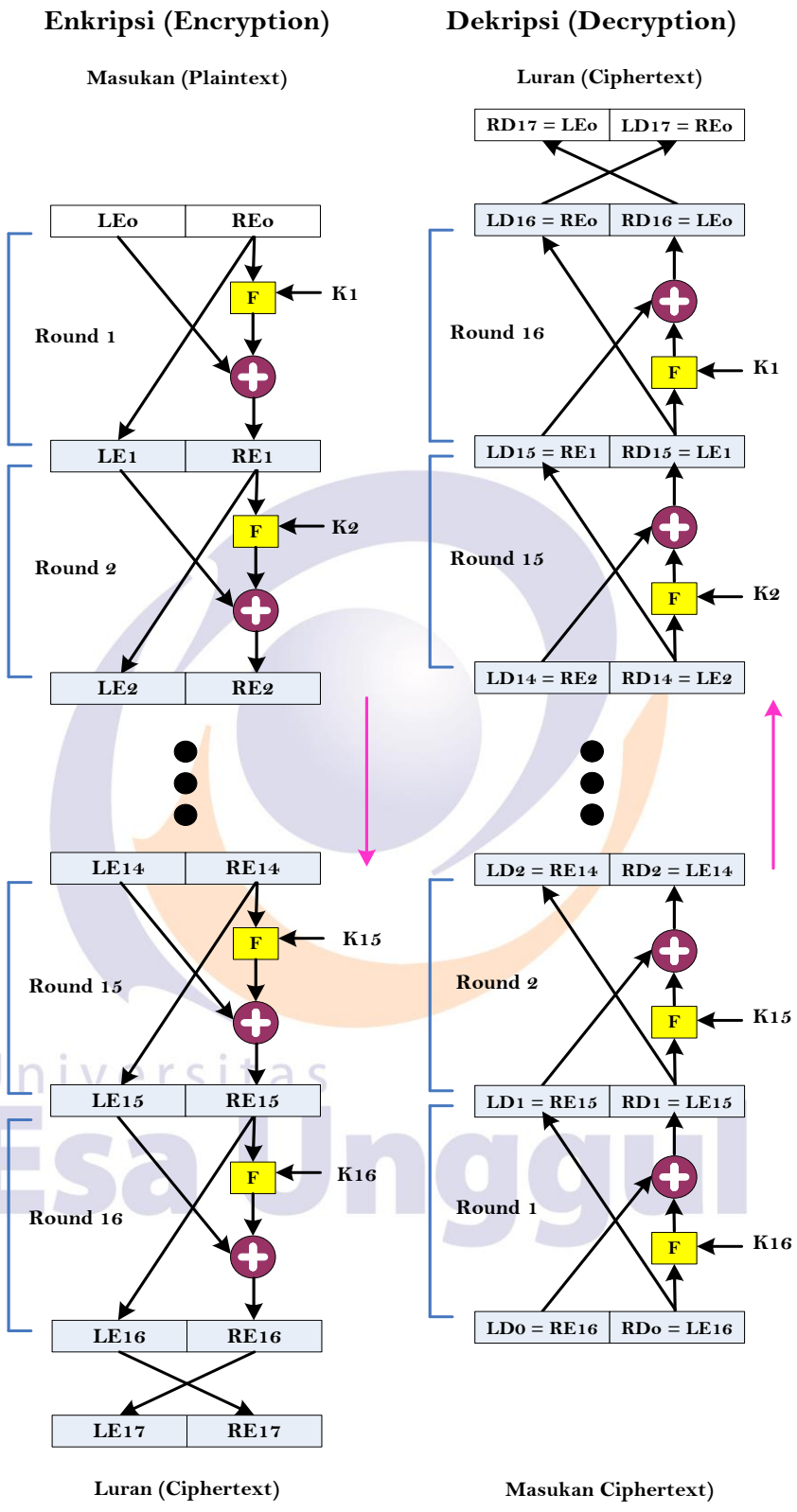
Secara khusus, Feistel mengusulkan penggunaan sandi yang berganti-ganti substitusi dan permutasi, di mana istilah tersebut didefinisikan sebagai berikut:

1) **Substitusi**, setiap elemen plaintext atau grup elemen adalah unik digantikan oleh elemen ciphertext atau grup elemen yang sesuai.

2) **Permutasi**, urutan elemen plaintext digantikan oleh urutan permutasi dan tidak ada elemen yang ditambahkan atau dihapus.



Gambar 5.7 Feistel Cipher, Proses Enkripsi dan Dekripsi



Gambar 5.8 Feistel Cipher, Enkripsi dan Dekripsi 16 Rounds

Gambar 5.8 menunjukkan proses enkripsi (kiri) dan proses dekripsi (kanan) dengan algoritma 16-putaran, dengan menggunakan symbol

atau notasi **LE_i dan RE_i** untuk proses data enkripsi dan **LD_i dan RD_i** untuk proses data dekripsi.

Proses Enkripsi

$LE_{16} = RE_{15}$

$RE_{16} = LE_{15}$ **XOR** $F(RE_{15}, K_{16})$

Proses Enkripsi

$LD_1 = RD_0 = LE_{16} = RE_{15}$

$RD_1 = LD_0$ **XOR** $F(RD_0, K_{16})$

$= RE_{16}$ **XOR** $F(RE_{15}, K_{16})$

$= [\ LE_{15}$ XOR $F(RE_{15}, K_{16})\ ]\ F(RE_{15}, K_{16})$

Sehingga kita memperoleh **$LD_1 = RD_{15}$** dan **$RD_1 = LE_{15}$**

Untuk algoritma enkripsi iterasi ke-i (secara umum)

$LE_i = RE_{i-1}$

$RE_i = LE_{i-1}$ **XOR** $F(RE_{i-1}, K_i)$

dan untuk dekripsi iterasi ke-i

$RE_{i-1} = LE_i$

$LE_{i-1} = RE_i$ **XOR** $F(RE_{i-1}, K_i)$

Dalam menerapkan secara benar Feistel berrgantung pada kita memilih parameter dan fitur desain antara lain ;

1) Ukuran blok, ukuran blok yang lebih besar akan berpengaruh pada keamanan yang lebih bagus tetapi akan mengurangi kecepatan dalam proses algoritma enkripsi / dekripsi.

2) Ukuran kunci, ukuran kunci yang lebih besar akan berpengaruh pada keamanan yang lebih kuat tetapi dapat mengurangi kecepatan enkripsi/ dekripsi. Keamanan yang

lebih besar akan bagus dalam mempertahankan terhadap serangan brute-force.

3) Jumlah putaran, da ukuran standard Feistel Cipher adalah 16 putaran (rounds)

4) Algoritma pembangkitan sub-key, bila kompleksitas yang lebih besar dalam algoritma ini akan menyebabkan kesulitan yang lebih besar dari dalam melakukan Cryptanalysis.

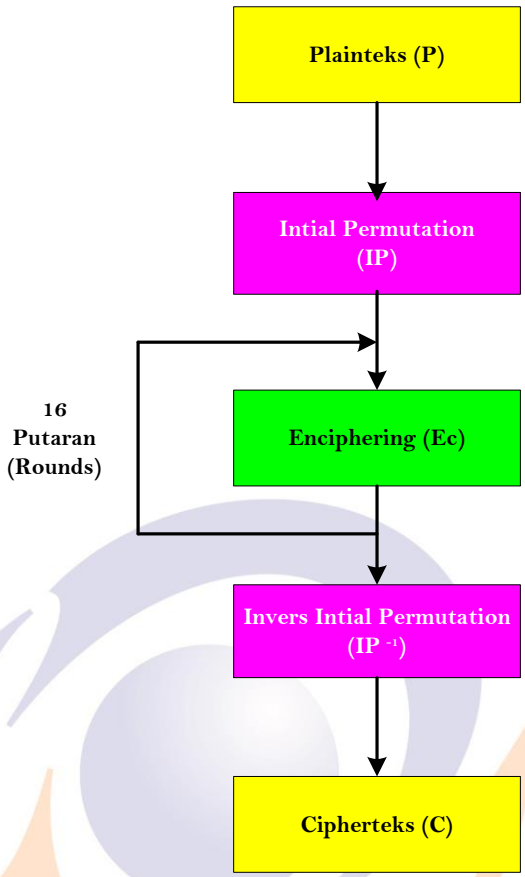
5) Fungsi putaran F, kompleksitas yang lebih besar umumnya berarti berpengaruh pada resistensi yang lebih besar

4. ALGORITMA *DATA ENCRYPTION STANDARD* (DES)

DES termasuk ke dalam sistem kriptografi simetri dan tergolong jenis cipher blok dan merupakan algoritma kriptografi simteris. Algoritma DES ini juga merupakan algoritma enkripsi block-chiper dengan panjang blok 64 bit dan dengan panjang kunci 56 bit yang bersifat rahasia yang dibagi (*shared secret*). Saat ini DES sudah hampir tidak digunakan dan cenderung ditinggalkan seiring perkembangan bidang keamanan data (*data security*) dikarenakan amat dengan mudah di-bobol dengan serangan Brute Force dengan panjang kunci yang hanya 56 bit itu.

4.1 CARA KERJA DES

Cara kerjanya adalah dengan mengubah atau me-transformasi plainteks ke bentuk cipherteks. Proses transformasi plainteks menjadi chiperteks diistilahkan dengan enkripsi, dan dekripis merupakan kebalikan dari operasi enkripsi (cipherteks menjadi plainteks).



Gambar 5.9 Hirarki Proses Algoritma DES

Contoh; bila plainteks **COMPUTER** dengan **13 34 57 79 9B BC DF F1** tentukan cipherteks dengan menggunakan algoritma DES.

Table ASCII

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Langkah Pertama

Konversikan plainteks dan kunci ke dalam bentuk biner

COMPUTER	C	O	M	P	U	T	E	R	PLAINTEKS
	43	55	53	56	61	60	45	58	
	00101011	00110111	00110101	00111000	00111101	00111100	00101101	00111010	
KEY	13	34	57	79	9B	BC	DF	F1	KUNCI
	00001101	00100010	00111001	01001111	10011011	10111100	11011111	11110001	

PLAINTEXT		KUNCI	
43	00101011	13	00001101
4F	01001111	34	00100010
4D	01001101	57	00111001
60	00111100	79	01001111
55	00110111	9B	10011011
54	00110110	BC	10111100
45	00101101	DF	11011111
52	00110100	F1	11110001

Langkah Kedua

Lakukan Initial Permutation (IP) pada bit plaintext dengan menggunakan tabel IP berikut:

Tabel Initial Permutation (IP)

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	45	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Urutan bit pada plaintext urutan ke 58 ditaruh di posisi 1,
Urutan bit pada plaintext urutan ke 50 ditaruh di posisi 2,
Urutan bit pada plaintext urutan ke 42 ditaruh di posisi 3, dst

Sehingga hasil outputnya adalah

	32 bit				32 bit			
IP (X)	11111111	10111000	01110110	01010111	'00000000	'00000001	00000110	10000011
	Lo				Ro			

IP(x) menjadi 2 bagian yaitu:

Lo	11111111	10111000	01110110	01010111
Ro	'00000000	'00000001	00000110	10000011

Langkah Ketiga

Generate kunci yang akan digunakanuntuk mengenkripsi plaintext dengan menggunakan tabel permutasi kompresi PC-1, pada langkah ini terjadi kompresi dengan membuang 1 bit masing-masing blok kunci dari 64 bit menjadi 56 bit.

Tabel Permutation Compression 1 (PC-1)

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

Dapat kita lihat pada tabel diatas, tidak terdapat urutan bit 8, 16, 24, 32, 40, 48, 56, 64 karena telah dikompress. Berikut hasil luaran nya

	28] bit				28] bit			
CD (k)	1111000	0110011	001010	0101111	0101010	1011001	1001111	0001111
	Co				Do			

k = 0	Co	1111000	0110011	001010	0101111
	Do	0101010	1011001	1001111	0001111

Langkah Ke-empat

Lakukan pergeseran kiri (Left Shift) pada C_0 dan D_0, sebanyak 1 atau 2 kali berdasarkan putaran yang ada pada tabel putaran sebagai berikut:

Tabel Left Shift

Putaran Ke-i	Jumlah Pergeseran	Putaran Ke-i	Jumlah Pergeseran
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

Untuk putaran ke 1, dilakukan pegeseran 1 bit ke kiri
Untuk putaran ke 2, dilakukan pergeseran 1 bit kekiri
Untuk putaran ke 3, dilakukan pergeseran 2 bit ke kiri, dst

Dengan menghasilkan luaran-nya sebagai berikut

	28] bit				28] bit			
CD (k)	1111000	0110011	0010101	0101111	0101010	1011001	1001111	0001111
	Co				Do			

k = 0	Co	1111000	0110011	0010101	0101111
	Do	0101010	1011001	1001111	0001111

Digeser 1 bit ke kiri

k = 1	C1	1110000	1100110	0101010	1011111
	D1	1010101	0110011	0011110	0011110

Digeser 2 bit ke kiri					
k = 2	C2	1100001	1001100	1010101	0111111
	D2	0101010	1100110	0111100	0111101
Digeser 2 bit ke kiri					
k = 3	C3	0000110	0110010	1010101	1111111
	D3	0101011	0011001	1110001	1110101
Digeser 2 bit ke kiri					
k = 4	C4	0011001	1001010	1010111	1111100
	D4	0101100	1100111	1000111	1010101
Digeser 2 bit ke kiri					
k = 5	C5	1100110	0101010	1011111	1110000
	D5	0110011	0011110	0011110	1010101
Digeser 2 bit ke kiri					
k = 6	C6	0011001	0101010	1111111	1000011
	D6	1001100	1111000	1111010	1010101
Digeser 2 bit ke kiri					
k = 7	C7	1100101	0101011	1111110	0001100
	D7	0110011	1100011	1 110101	0 101011
Digeser 2 bit ke kiri					
k = 8	C8	0010101	0101111	1111000	0110011
	D8	1001111	0001111	0101010	1011001
Digeser 1 bit ke kiri					
k = 9	C9	0101010	1011111	1110000	1100110
	D9	0011110	0011110	1010101	0110011
Digeser 2 bit ke kiri					
k = 10	C10	0101010	1111111	1000011	0011001
	D10	1111000	1111010	1010101	1001100
Digeser 2 bit ke kiri					
k = 11	C11	0101011	1111110	0001100	1100101
	D11	1100011	1101010	1010110	0110011

Digeser 2 bit ke kiri					
k = 12	C12	**0101111**	**1111000**	**0110011**	**0010101**
	D12	**0001111**	**0101010**	**1011001**	**1001111**
Digeser 2 bit ke kiri					
k = 13	C13	**0111111**	**1100001**	**1001100**	**1010101**
	D13	**0111101**	**0101010**	**1100110**	**0111100**
Digeser 2 bit ke kiri					
k = 14	C14	**1111111**	**0000110**	**0110010**	**1010101**
	D14	**1110101**	**0101011**	**0011001**	**1110001**
Digeser 2 bit ke kiri					
k = 15	C15	**1111100**	**0011001**	**1001010**	**1010111**
	D15	**1010101**	**0101100**	**1100111**	**1000111**
Digeser 1 bit ke kiri					
k = 16	C16	**1111000**	**0110011**	**0010101**	**0101111**
	D16	**0101010**	**1011001**	**1001111**	**0001111**

Setiap hasil putaran (k1 – k16) digabungkan kembali menjadi C_i D_i dan di-entry ke dalam tabel Permutation Compression 2 (PC-2) dan terjadi kompresi data C_i D_i 56 bit menjadi C_i D_i 48 bit.

Tabel Permutation Compression 1 (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

C1 D1	**1110000**	**1100110**	**0101010**	**1011111**	**1010101**	**0110011**	**0011110**	**0011110**
K1	**000110**	**110000**	**001011**	**101111**	**111111**	**000111**	**000001**	**110010**

C2 D2	**1100001**	**1001100**	**1010101**	**0111111**	**0101010**	**1100110**	**0111100**	**0111101**
K2	**011110**	**011010**	**111011**	**011001**	**110110**	**111100**	**100111**	**100101**

C3 D3	0000110	0110010	1010101	1111111	0101011	0011001	1110001	1110101
K3	010101	011111	110010	001010	010000	101100	111110	011001
C4 D4	0011001	1001010	1010111	1111100	0101100	1100111	1000111	1010101
K4	011100	101010	110111	010110	110110	110011	010100	011101
C5 D5	1100110	0101010	1011111	1110000	0110011	0011110	0011110	1010101
K5	011111	001110	110000	000111	111010	110101	001110	101000
C6 D6	0011001	0101010	1111111	1000011	1001100	1111000	1111010	1010101
K6	011000	111010	010100	111110	010100	000111	101100	101111
C7 D7	1100101	0101011	1111110	0001100	0110011	1100011	1 110101	0 101011
K7	111011	001000	010010	110111	111101	100001	100010	111100
C8 D8	0010101	0101111	1111000	0110011	1001111	0001111	0101010	1011001
K8	111101	111000	101000	111010	110000	010011	101111	111011
C9 D9	0101010	1011111	1110000	1100110	0011110	0011110	1010101	0110011
K9	111000	001101	101111	101011	111011	011110	011110	000001
C10 D10	0101010	1111111	1000011	0011001	1111000	1111010	1010101	1001100
K10	101100	011111	001101	000111	101110	100100	011001	001111
C11 D11	0101011	1111110	0001100	1100101	1100011	1101010	1010110	0110011
K11	001000	010101	111111	010011	110111	101101	001110	000110
C12 D12	0101111	1111000	0110011	0010101	0001111	0101010	1011001	1001111
K12	011101	010111	000111	110101	100101	000110	011111	101001
C13 D13	0111111	1100001	1001100	1010101	0111101	0101010	1100110	0111100
K13	100101	111100	010111	010001	111110	101011	101001	000001
C14 D14	1111111	0000110	0110010	1010101	1110101	0101011	0011001	1110001
K14	010111	110100	001110	110111	111100	101110	011100	111010
C15 D15	1111100	0011001	1001010	1010111	1010101	0101100	1100111	1000111
K15	101111	111001	000110	001101	001111	010011	111100	001010
C16 D16	1111000	0110011	0010101	0101111	0101010	1011001	1001111	0001111
K16	110010	110011	110110	001011	000011	100001	011111	110101

Langkah Kelima

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).

EKPANSI						
	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 Ki dan menghasilkan Vektor Matriks Ai.

Menghasilkan luaran pada langkah ke-5 sebagai berikut ;

Iterasi - 1

E(R(1)-1)	=	100000 000000 000000 000000 000000 001101 010000 000110	
K1	=	000110 110000 001011 101111 111111 000111 000001 110010	
			XOR
A1	=	100110 110000 001011 101111 111111 001010 010001 110100	

Iterasi - 2

E(R(2)-1)	=	011010 101110 100001 010110 100110 100101 010000 001101	
K2	=	011110 011010 111011 011001 110110 111100 100111 100101	
			XOR
A2	=	000100 110100 011010 001111 010000 011001 110111 101000	

Iterasi - 3

E(R(3)-1)	=	010001 010111 111011 110011 110001 010101 010010 100001	
K3	=	010101 011111 110010 001010 010000 101100 111110 011001	
			XOR
A3	=	000100 001000 001001 111001 100001 111001 101100 111000	

Iterasi - 4

E(R(4)-1)	=	010111 110001 010111 110011 110101 011100 001111 110001	
K4	=	011100 101010 110111 010110 110110 110011 010100 011101	
			XOR
A4	=	001011 011011 100000 100101 000011 101111 011011 101100	

Iterasi - 5

E(R(5)-1)	=	110110 101001 011100 000101 011001 011010 100110 100011	
K5	=	011111 001110 110000 000111 111010 110101 001110 101000	
			XOR
A5	=	101001 100111 101100 000010 100011 101111 101000 001011	

Iterasi - 6

E(R(6)-1)	=	100101 011011 110001 010110 101110 101100 000111 111010	
K6	=	011000 111010 010100 111110 010100 000111 101100 101111	
			XOR
A6	=	111101 100001 100101 101000 111010 101011 101011 010101	

Iterasi - 7

E(R(7)-1)	=	110010 100001 011111 110010 100111 111101 011001 010011	
K7	=	111011 001000 010010 110111 111101 100001 100010 111100	
			XOR
A7	=	001001 101001 001101 000101 011010 011100 111011 101111	

Iterasi - 8

E(R(8)-1)	=	111100 001010 101001 010101 010011 110000 001010 100011	
K8	=	111101 111000 101000 111010 110000 010011 101111 111011	
			XOR
A8	=	000001 110010 000001 101111 100011 100011 100101 011000	

Iterasi - 9

E(R(9)-1)	=	010010 101111 111000 000000 000010 101111 110101 010001	
K9	=	111000 001101 101111 101011 111011 011110 011110 000001	
			XOR
A9	=	101010 100010 010111 101011 111001 110001 101011 010000	

Iterasi - 10

E(R(10)-1)	=	100111 111000 001110 100010 100111 110111 111000 001010	
K10	=	101100 011111 001101 000111 101110 100100 011001 001111	
			XOR
A10	=	001011 100111 000011 100101 001001 010011 100001 000101	

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 Keenam :

Setiap Vektor Ai disubstitusikan kedelapan buah S-Box(Substitution Box), dimana blok pertama disubstitusikan dengan S1, blok kedua dengan S2 dan seterusnya dan menghasilkan output vektor Bi 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	15
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

Cara menggunakan S-Box, ambil contoh S1, kemudian konversi setiap angka di dalam tabel S1 yang berwarna putih menjadi biner, sehingga menjadi bentuk :

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 A1 yaitu 100110, kemudian pecah kan blok menjadi 2 yaitu:

1) Bit pertama dan terakhir yaitu 1 dan 0 digabungkan menjadi 10

2) 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 ke delapan kita bandingkan dengan S2 hingga S8. Berdasarkan cara diatas diperoleh hasil sebagai berikut:.

B1	=	1000 0101 0100 1000 0011 0010 1110 1010
B2	=	1101 1100 0100 0011 1000 0000 1111 1001
B3	=	1101 0110 0011 1100 1011 0110 0111 1111
B4	=	0010 1001 1101 0000 1011 1010 1111 1110
B5	=	0100 0001 0011 1101 1000 1010 1100 0011
B6	=	0110 1101 1101 1100 0011 0101 0100 0110
B7	=	1110 0011 0110 1011 0000 0101 0010 1101
B8	=	0000 1000 1101 1000 1000 0011 1101 0101
B9	=	0110 1110 1110 0001 1010 1011 0100 1010
B10	=	0010 0001 0111 0000 0100 0001 0110 1101
B11	=	0101 1110 0000 1100 1101 1011 1100 0010
B12	=	0110 1000 0000 1011 0011 0110 1010 1101
B13	=	1111 1001 1101 1011 0010 0100 1011 0011
B14	=	1011 1000 0111 1110 1100 0101 1100 0001
B15	=	0100 0001 0011 1001 1111 0111 0010 0111
B16	=	1000 0001 0110 1010 1111 0111 0100 1011

Langkah Ketujuh:

Setelah didapatkan nilai vektor Bi, langkah selanjutnya adalah memutasikan bit vektor Bi menggunakan tabel P-Box, kemudian

dikelompokkan menjadi 4 blok dimana tiap-tiap blok memiliki 32 bit data.

Tabel P-Box

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
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(Bi) kemudian di XOR kan dengan Li-1 untuk mendapatkan nilai Ri. sedangkan nilai Li sendiri diperoleh dari Nilai Ri-1 untuk nilai 1 <= i <= 16.

Langkah Kedelapan:

Langkah terakhir adalah menggabungkan R16 dengan L16 kemudian dipermutasikan untuk terakhir kali dengan tabel Invers Initial Permutasi(IP-1).