Algoritma Kriptografi Modern (Bagian 2)

Bahan Kuliah

IF4020 Kriptografi

Pendahuluan

- *Block cipher* yang diberikan di dalam kuliah:
  1. DES
  2. 3DES
  3. GOST
  4. RC5
  6. AES

- *Block cipher* lainnya (tidak diajarkan, dapat dibaca di dalam referensi):

1. Blowfish
2. IDEA
3. LOKI
4. RC2
5. FEAL
6. Lucifer
7. CAST
8. CRAB
9. SAFER
10. Twofish
12. Serpent
13. RC6
14. MARS
15. Camellia
16. 3-WAY
17. MMB, SkipJack, dll

- *Stream cipher* yang diberikan di dalam kuliah:
  1. RC4
  2. A5

- *Stream cipher* lainnya (tidak diajarkan, dapat dibaca di dalam referensi):
  1. A2
  2. SEAL
  3. WAKE
  4. Crypt(1)
  5. Cellular Automaton

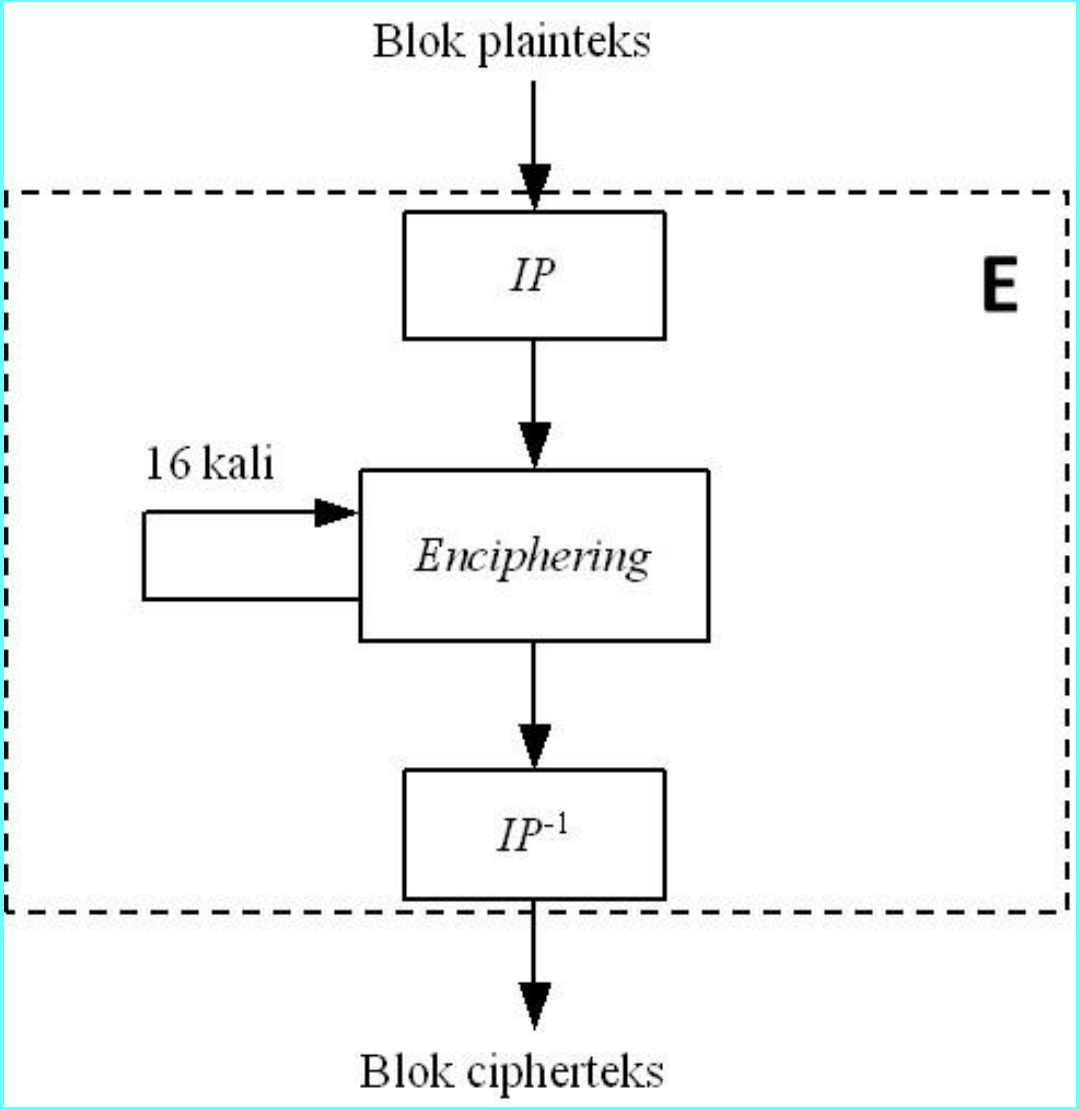
1. Data Encryption Standard (DES)

Tinjauan Umum DES

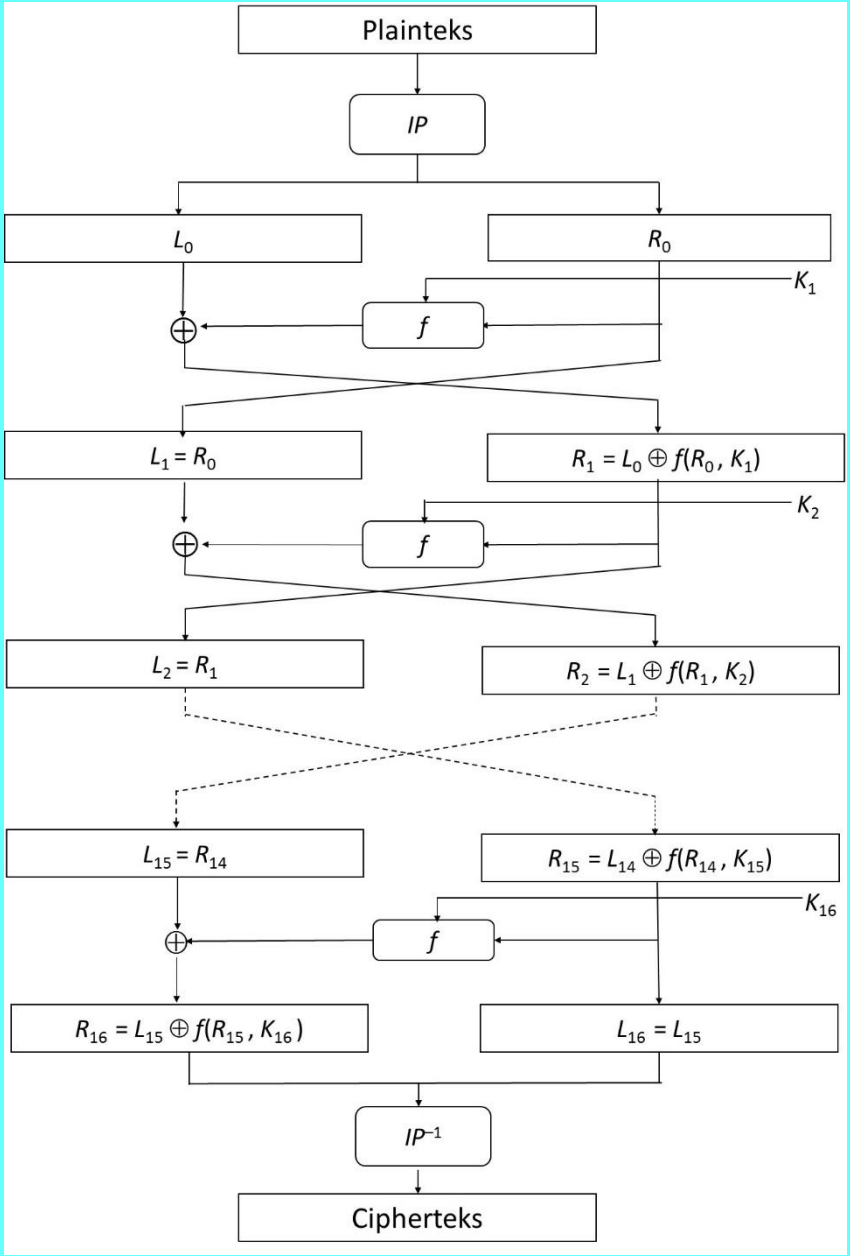
- Dikembangkan di IBM pada tahun 1972.
- Berdasarkan pada algoritma *Lucifer* yang dibuat oleh Horst Feistel.
- Disetujui oleh *National Bureau of Standard* (*NBS*) setelah penilaian kekuatannya oleh *National Security Agency* (*NSA*) Amerika Serikat.

- DES adalah standard, sedangkan algoritmanya adalah DEA (*Data EncryptionAlgorithm*). Kedua nama ini sering dikacaukan.
- DES termasuk ke dalam kriptografi kunci-simetri dan tergolong jenis *cipher* blok.
- DES beroperasi pada ukuran blok 64 bit.
- Panjang kunci ekternal = 64 bit (sesuai ukuran blok), tetapi hanya 56 bit yang dipakai (8 bit paritas tidak digunakan)

- Setiap blok (plainteks atau cipherteks) dienkripsi dalam 16 putaran.
- Setiap putaran menggunakan kunci internal berbeda.
- Kunci internal (56-bit) dibangkitkan dari kunci eksternal
- Setiap blok mengalami permutasi awal (*IP*), 16 putaran *enciphering*, dan inversi permutasi awal (IP^{-1}). (lihat Gambar 9.1)



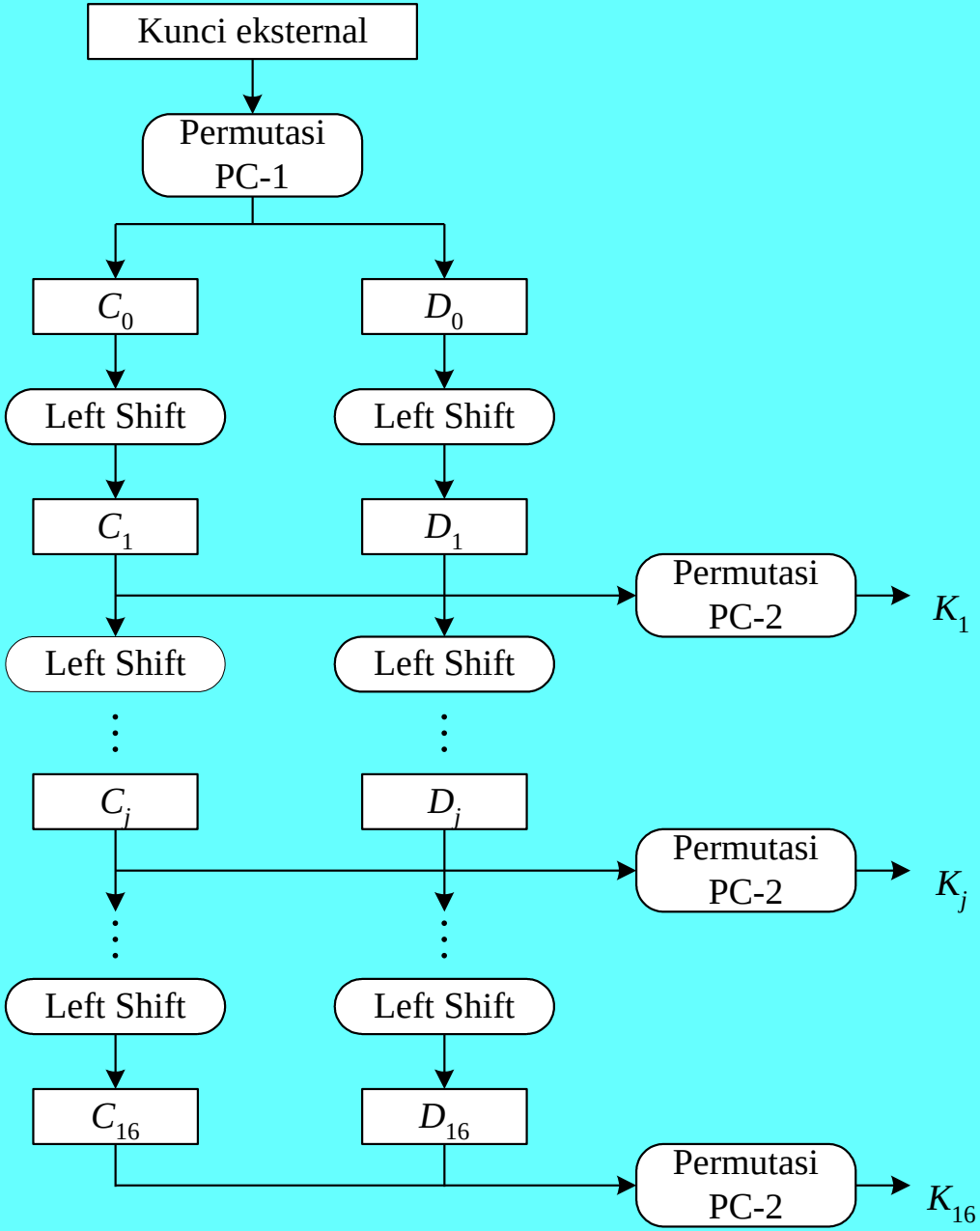
Gambar Skema global algoritma DES



Gambar Algoritma Enkripsi dengan DES

Pembangkitan Kunci Internal

- Kunci internal = kunci setiap putaran
- Ada 16 putaran, jadi ada 16 kunci internal: $K_1, K_2, \ldots, K_{16}$
- Dibangkitkan dari kunci eksternal (64 bit) yang diberikan oleh pengguna.
- Gambar 9.2 memperlihatkan proses pembangkitan kunci internal.



Gambar Proses pembangkitan kunci-kunci internal DES

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*):

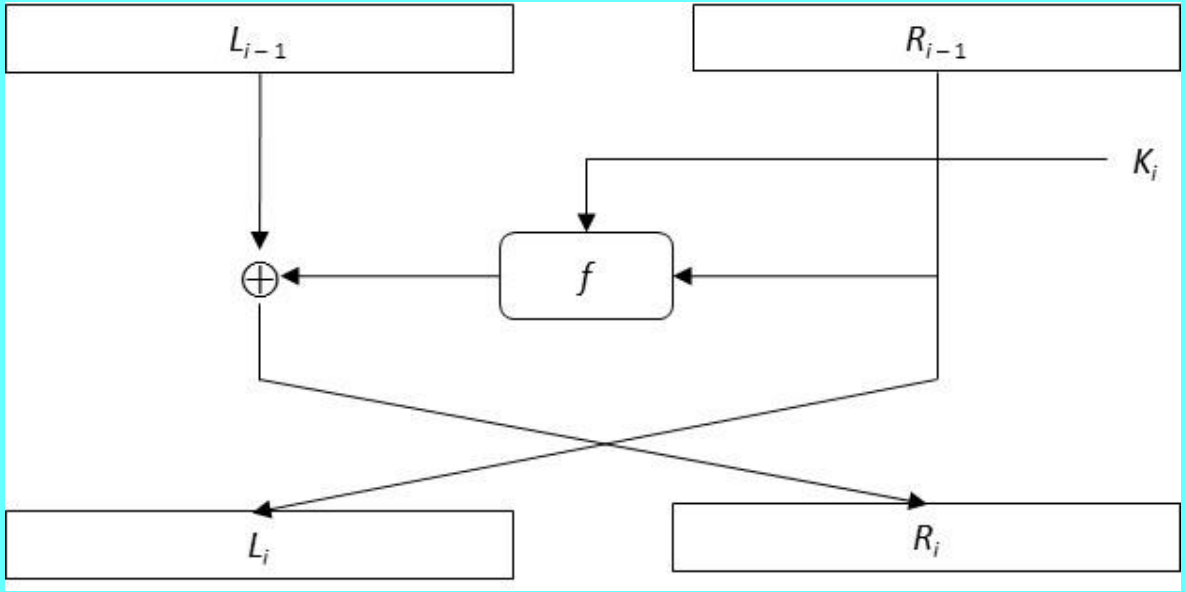
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Enciphering

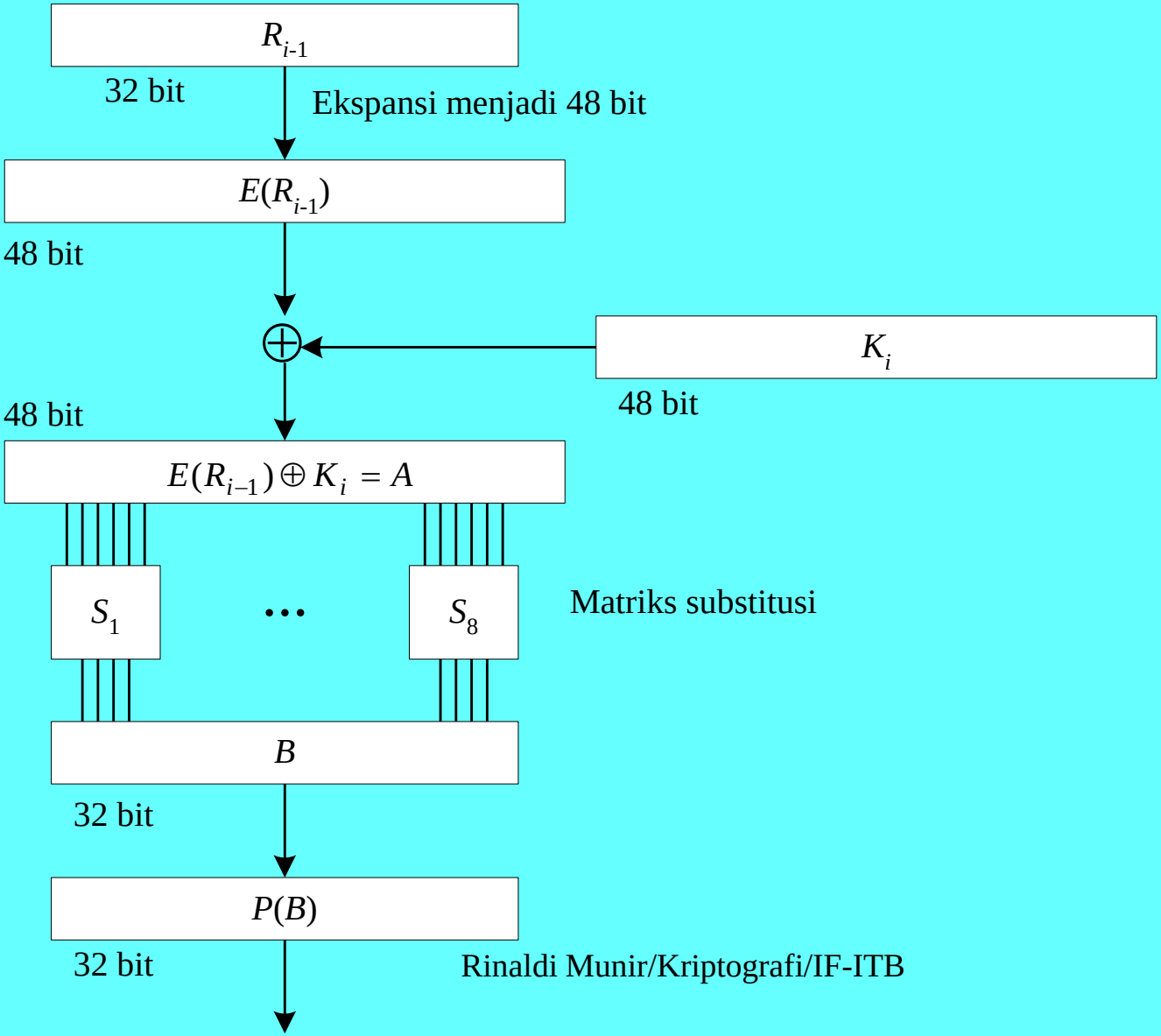
- Setiap blok plainteks mengalami 16 kali putaran *enciphering* .
- Setiap putaran *enciphering* merupakan jaringan Feistel:

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

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



- 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 ekpansi, 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 memebrikan 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

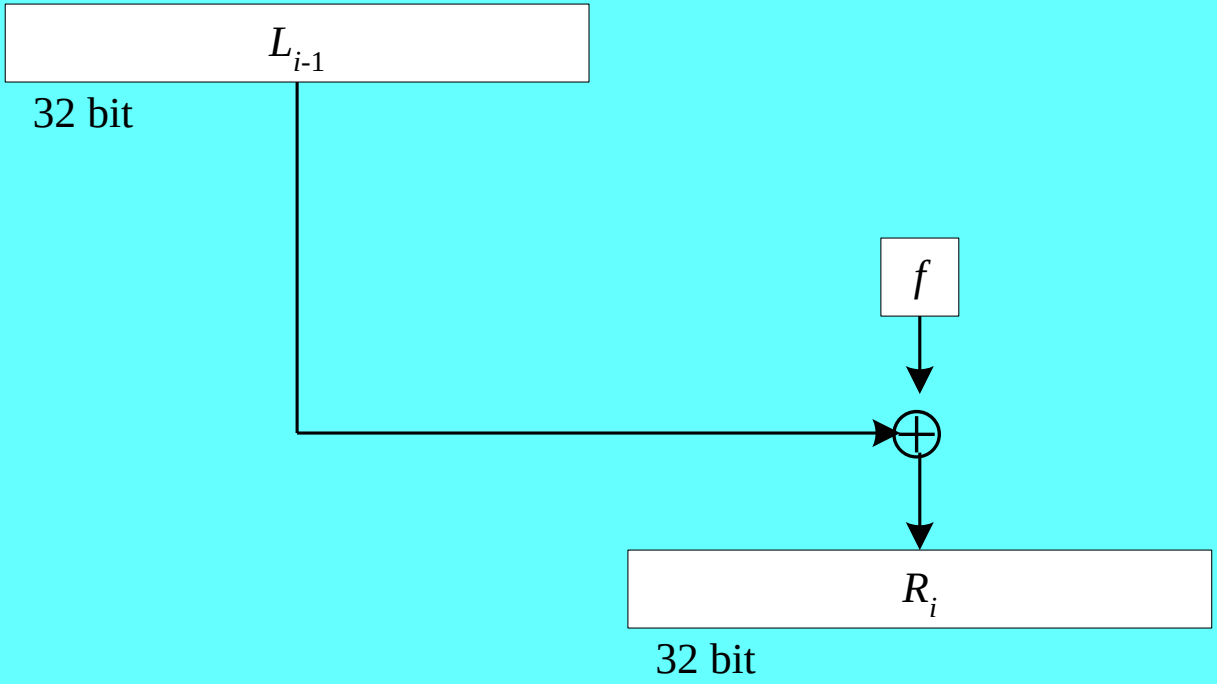
- Keluaran proses substitusi adalah vektor *B* yang panjangnya 48 bit.
- Vektor *B* menjadi masukan untuk proses permutasi.
- Tujuan permutasi adalah untuk mengacak hasil proses substitusi kotak-S.
- Permutasi dilakukan dengan menggunakan matriks permutasi *P* (*P-box*) sbb:

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 keluaran 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, keluaran 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

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, keluaran pada setiap putaran *deciphering* adalah

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

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

Mode DES

- DES dapat dioperasikan dengan mode ECB, CBC, OFB, dan CFB.

- Namun karena kesederhanaannya, mode ECB lebih sering digunakan pada paket komersil.

Implementasi 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

- 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.
- Tetapi, dengan panjang kunci 56 bit akan terdapat 2^{56} atau 72.057.594.037.927.936 kemungkinan kunci.
- Jika serangan *exhaustive key search* dengan menggunakan prosesor paralel, maka dalam satu detik dapat dikerjakan satu juta serangan. Jadi seluruhnya diperlukan 1142 tahun untuk menemukan kunci yang benar.

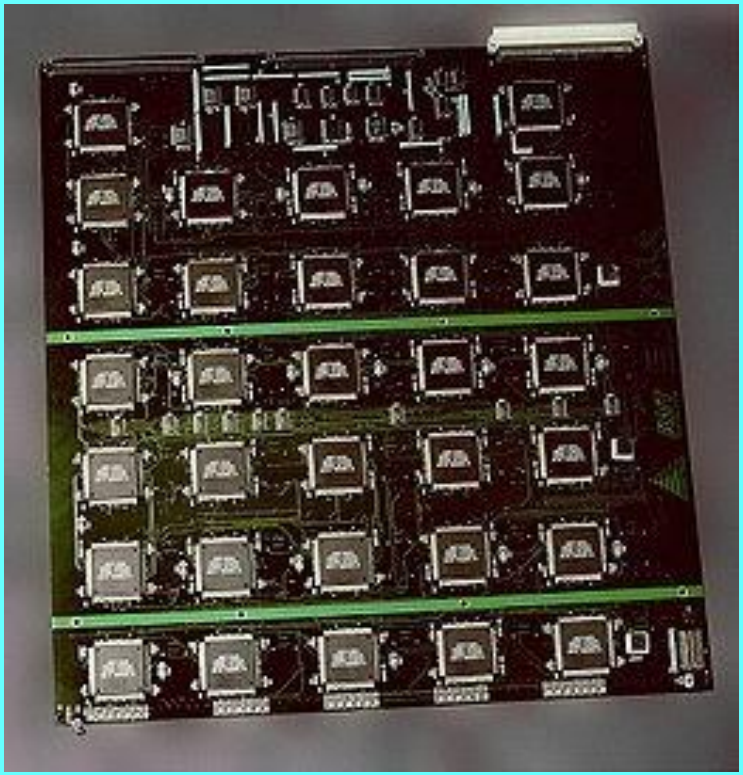
Dikutip dari Wiki:

- 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 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.



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.

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

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

GOST

Tinjauan Umum GOST

- *GOST = Gosudarstvenny Standard*, artinya standard pemerintah,
- 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*

- 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.

Tabel 6.2 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

- Pembangkitan kunci internal sangat sederhana.

- Kunci eksternal yang panjangnya 256 bit dibagi ke dalam delapan bagian yang masing-masing panjangnya 32 bit.

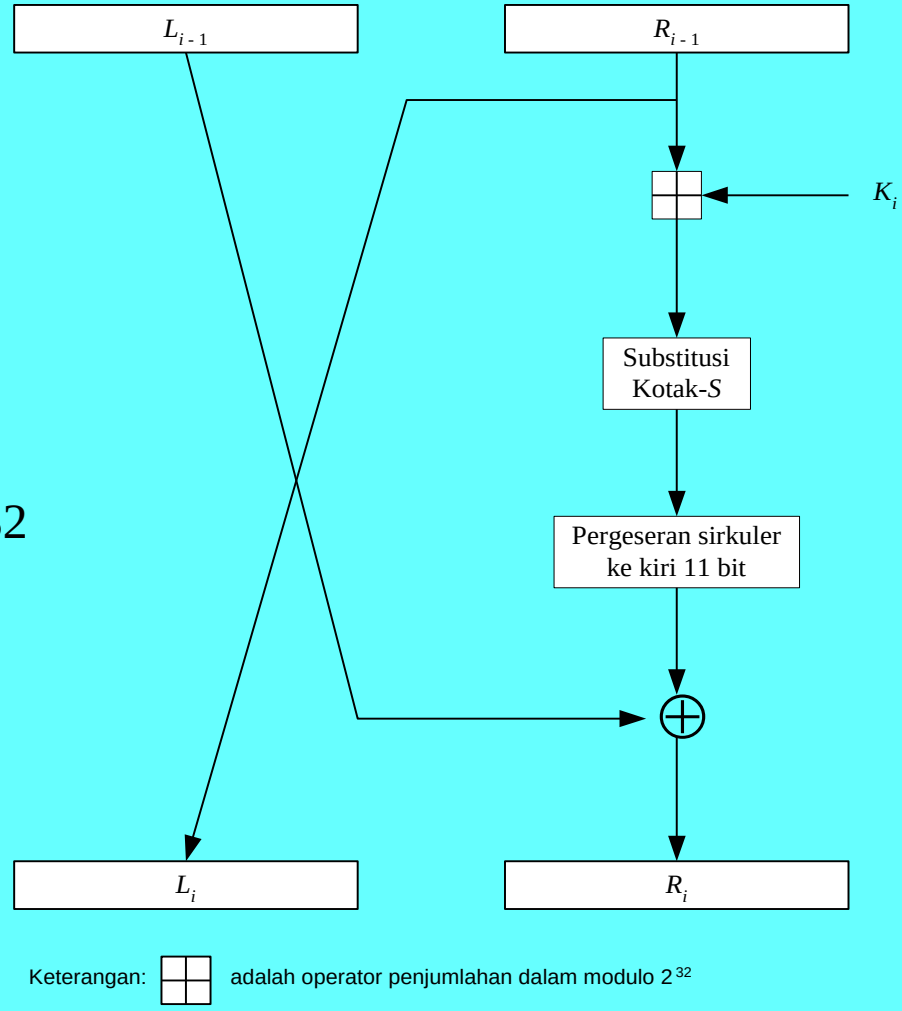
- Delapan bagian ini yang dinamakan K_1, K_2, …, K_8.

- *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



- 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

- 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.

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 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* [WIK06].

Triple DES

DES Berganda

- Karena DES mempunyai potensi kelemahan pada *brute force atack*, maka dibuat varian dari DES.

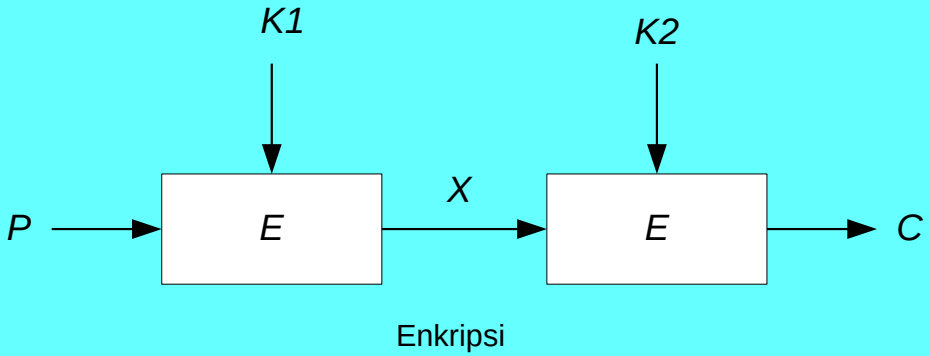
- Varian DES yang paling luas digunakan adalah DES berganda (*multiple DES*).

- DES berganda adalah enkripsi berkali-kali dengan DES dan menggunakan kunci ganda.

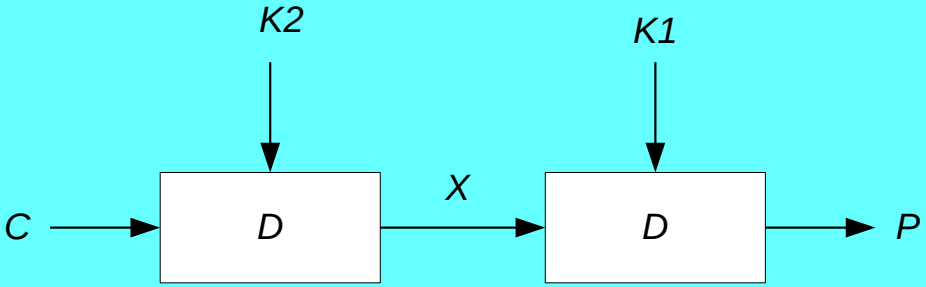
- Tinjau DES berganda:
  1. *Double DES*
  2. *Triple DES*

Double DES

- Menggunakan 2 buah kunci eksternal, K_1 dan K_2.
- Enkripsi: $C = E_{K2}(E_{K1}(P))$
- Dekripsi: $P = D_{K1}(D_{K2}(C))$



Enkripsi



Dekripsi

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

$$C = E_{K2}(E_{K1}(P))$$

maka

$$X = E_{K1}(P) = D_{K2}(C)$$

- Misalkan kriptanalis memiliki potongan C dan P yang berkorepsonden.
- Enkripsi P untuk semua kemungkinan nilai $K1$ (yaitu sebanyak 2^{56} kemungkinan kunci). Hasilnya adalah semua nilai X
- Simpan semua nilai X ini di dalam tabel

- Berikutnya, dekripsi C dengan semua semua kemungkinan nilai $K2$ (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, $K1$ dan $K2$, telah ditemukan.

- Tes kedua kunci ini dengan pasangan plainteks-cipherteks lain yang diketahui. Jika kedua kunci tersebut menghasilkan cipherteks atau plainteks yang benar, maka $K1$ dan $K2$ tersebut merupakan kunci yang benar

Triple DES (TDES)

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

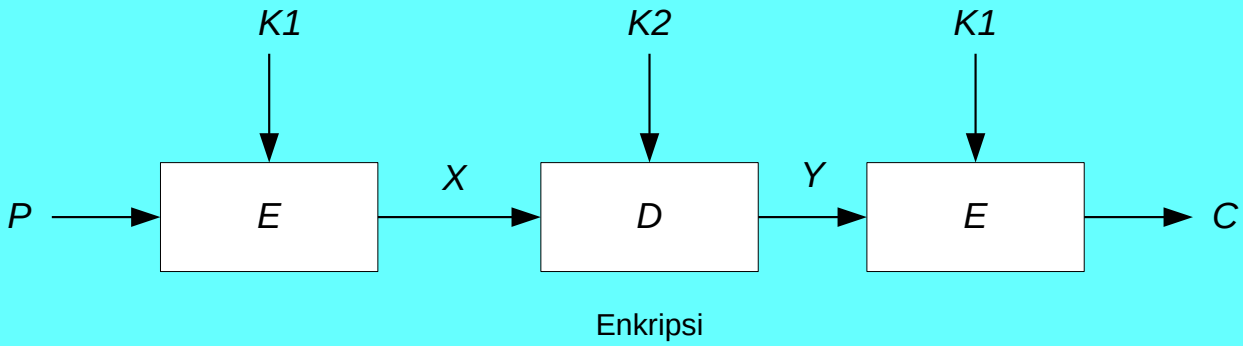
- Untuk menyederhanakan TDES, maka langkah di tengah diganti dengan D (mode EDE).
- Ada dua versi TDES dengan mode EDE:

  - Menggunakan 2 kunci

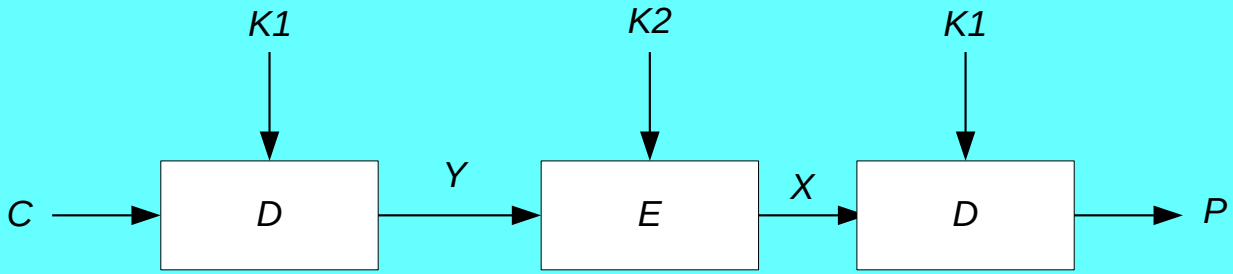
  - Menggunaakn 3 kunci

Triple DES

***Triple* DES dengan 2 kunci**

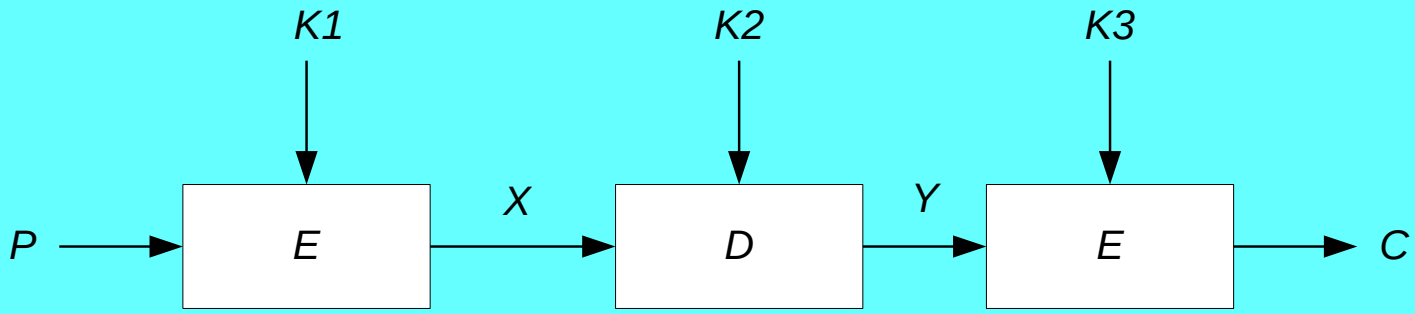


Enkripsi

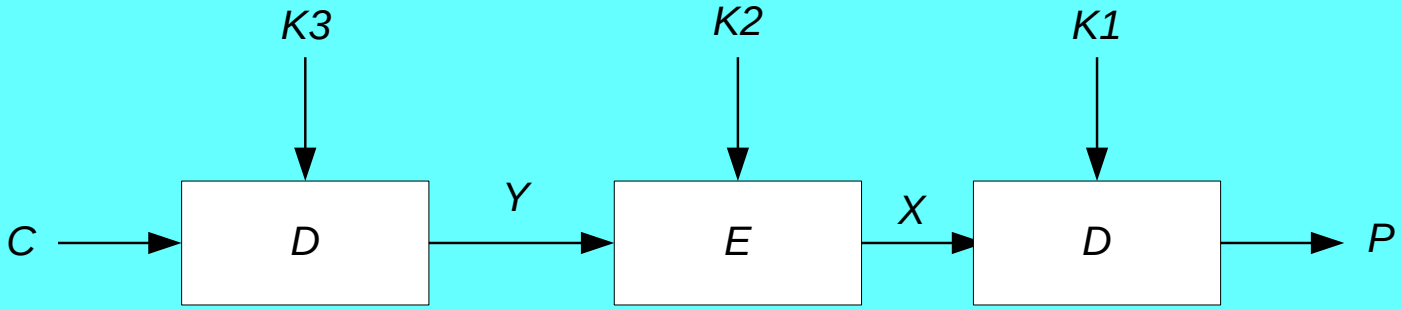


Dekripsi

Triple DES dengan 3 kunci



Enkripsi



Dekripsi

RC5

- *RC5* dibuat oleh Ron Rovest dari Laboratorium *RSA*.

- Tidak seperti algoritma *cipher* blok lainnya, *RC5* mempunyai:
  - ukuran blok yang variabel (32, 64, atau 128 bit)
  - panjang kunci yang variabel (0 sampai 2040 bit)
  - dan jumlah putaran yang variabel (0 sampai 255).

Parameter	Simbol	Nilai yang dibolehkan
Ukuran blok (dalam bit)	*w*	16, 32, 64
Jumlah putaran	*r*	0, 1, …, 255
Panjang kunci eksternal *K* (dalam *byte*, 1 *byte* = 8 bit)	*b*	0, 1, …, 255

Pembentukan Kunci Internal

- Kunci internal ada sebanyak $2r + 2$ buah yang masing-masing disimpan di dalam elemen-elemen larik yang dilabeli sebagai $S[0], S[1], \ldots, S[t-1]$ dengan $t = 2r + 2$.

- Setiap elemen larik panjangnya satu *word* (1 *word* = w bit)

- Mula-mula, semua *byte* dari kunci eksternal, $K[0..b-1]$, disalin ke dalam larik L yang berukuran c *word*, $L[0..c-1]$
- lalu *padding* dengan sejumlah 0 jika perlu (*padding* terjadi jika b bukan kelipatan w).

- Kemudian inisialisasi larik S sebagai berikut:

$S[0] \leftarrow P_w$
for i $\leftarrow$ 1 **to** t **-** 1 **do**
    $S[i] \leftarrow S[i-1] + Q_w$
endfor

yang dalam hal ini nilai P_w dan Q_w (dalam heksadesimal) berbeda-beda bergantung pada w sebagai berikut [STA98]:

w	16	32	64
P_w	B7E1	B7E15163	B7E151628AED2A6B
Q_w	9E37	9E3779B9	9E3779B97F4A7C15

Konstanta P_w dan Q_w didasarkan pada representasi bilangan alam e dan ϕ dalam biner,

$$P_w = \text{Odd}[(e - 2)2^w]$$
$$Q_w = \text{Odd}[(\phi - 1)2^w]$$

yang dalam hal ini,

$$e = 2.718281828459\ldots$$

$$\phi = 1.618033988749\ldots = \frac{1+\sqrt{5}}{2}$$

Akhirnya, campurkan *L* dan *S* sebagai berikut:

```
i ← 0
j ← 0
X ← 0
Y ← 0
n ← 3 * max(r, c)
for k ← 1 to n do
    S[i] ← (S[i] + X + Y) <<< 3
    X ← S[i]
    i ← (1 + 1) mod (t)
    L[j] ← (L[j] + X + Y) <<< ( X + Y)
    Y ← L[j]
    j ← (j + 1) mod (c)
endfor
```

Enkripsi

- Tinjau *RC5* dengan ukuran blok 64 bit dan jumlah putaran *r*.
- Enkripsi menggunakan kunci internal S_0, S_1, …, S_{2r+2} yang masing-masing panjangnya 32-bit.

- Dua kunci internal digunakan untuk setiap putaran i = 1, 2, .. r dan dua buah kunci internal tambahan sebelum putaran pertama jadi seluruhnya ada $2r$ + 2 buah kunci internal).
- Untuk melakukan enkripsi, mula-mula blok plainteks dibagi menjadi 2 bagian, *A* dan *B*, yang masing-masing panjangnya 32 bit. Kemudian masing-masing bagian dijumlahkan (dalam modulo 2^{32}) dengan S_0 dan S_1:

$A \leftarrow A + S[0]$

$B \leftarrow B + S[1]$

- Selanjutnya untuk setiap putaran dari 1 sampai *r* dilakukan operasi *XOR*, pergeseran ke kiri secara sirkuler, dan penjulahan dalam modulo 2^{32} dengan kunci internal sebagai berikut:

for i $\leftarrow$ 1 **to** r **do**

$A \leftarrow ((A \oplus B) <<< B) + S[2i]$

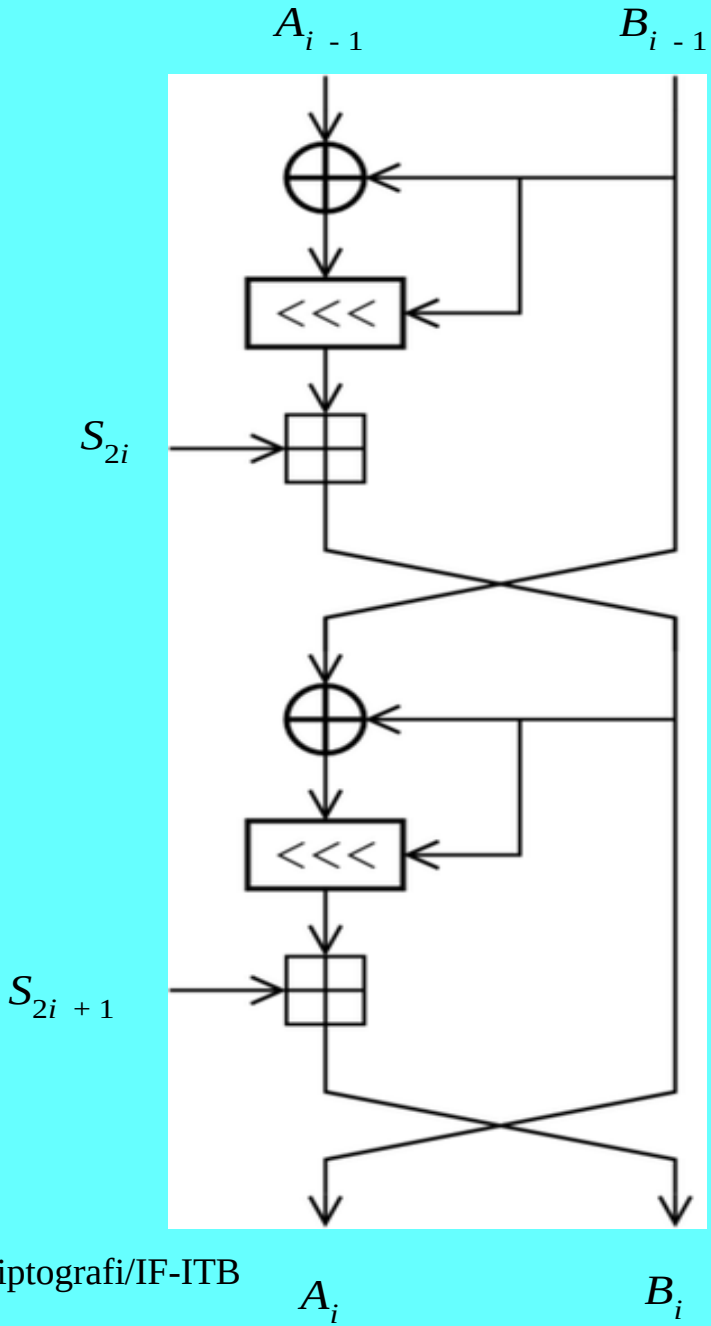
$B \leftarrow ((B \oplus A) <<< A) + S[2i+1]$

endfor

Cipherteks pada putaran terakhir disimpan di dalam *A* dan *B*.

Gabungan keduanya adalah blok plainteks yang berukuran 64 bit.

Proses enkripsi satu putaran:



RC4

RC4

- Termasuk ke dalam *cipher* aliran (*stream cipher*)
- Dibuat oleh Ron Rivest (1987) dari Laboratorium *RSA*
- *RC* adalah singkatan dari *Ron's Code*). Versi lain megatakan *Rivest Cipher* .
- Digunakan sistem keamanan seperti:

  - protokol *SSL* (*Secure Socket Layer*).

  - WEP (Wired Equivalent Privacy)

  - WPA (Wi-fi Protect Access) untuk nirkabel

- RC4 awalnya rahasia
- Pada September 194, RC4 dikirim secara anonim ke milis *Cypherpunks*
- Lalu dikirim ke *newsgroup* `sci.crypt` dan menyebar di internet
- Karena telah diketahui orang, RC4 bukan lagi rahasia dagang
- Status sekarang, implementasi tidak resmi adalah legal, tapi tidak boleh menggunakan nama RC4. Maka digunakan nama ARCFOUR untuk menghindari masalah *trademark*.

- *RC4* membangkitkan aliran-kunci (*keystream*) yang kemudian di-XOR-kan dengan plainteks

- *RC4* memproses data dalam ukuran *byte*, bukan dalam bit.

- Untuk membangkitkan aliran-kunci, *cipher* menggunakan status internal yang terdiri dari:
  - Permutasi angka 0 sampai 255 di dalam larik S_0, S_1, …, S_{255}. Permutasi merupakan fungsi dari kunci K dengan panjang variabel.
  - Dua buah pencacah indeks, i dan j

Algoritma RC4:

1. Inisialisasi larik S: $S_0 = 0$, $S_1 = 1$, …, $S_{255} = 255$

```
for i ← 0 to 255 do
  S[i] ← i
endfor
```

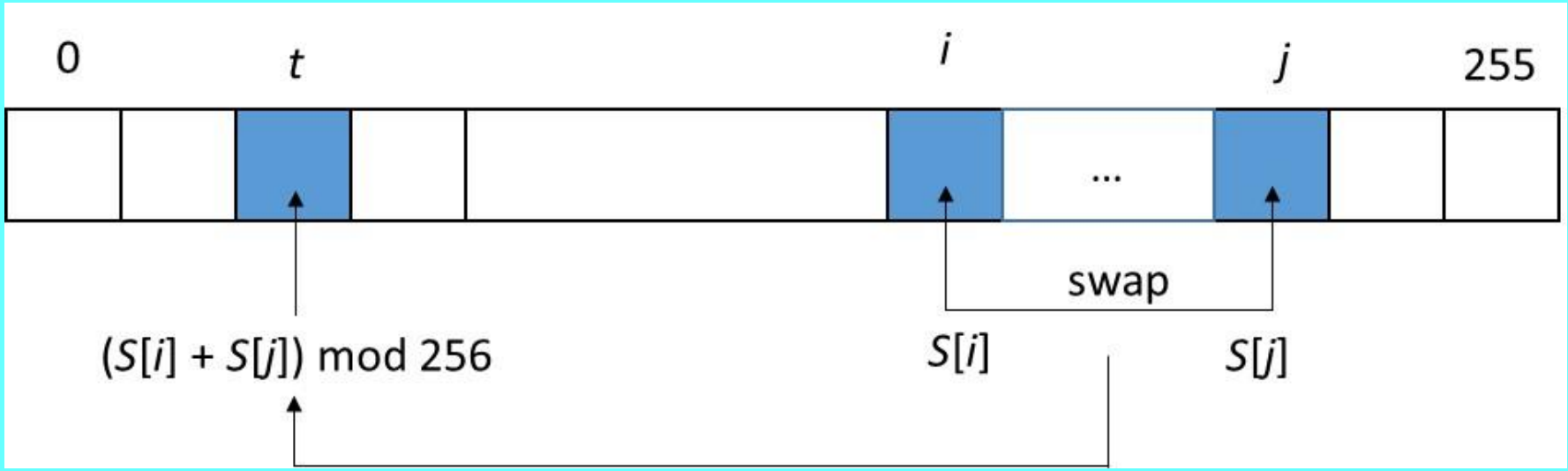
0	1	2	3	4	5	6	7	…	…	…	252	253	254	255
0	1	2	3	4	5	6	7	…	…	…	252	253	254	255

3. Lakukan pengacakan (permutasi) nilai-nilai di dalam larik *S* :

```
j ← 0
for i ← 0 to 255 do
  j ← (j + S[i] + K[i  mod Length(K)]) mod 256
  swap(S[i], S[j]) {* Pertukarkan S[i] & S[j] *}
end
```

4. Bangkitkan aliran-kunci dan lakukan enkripsi:

```
i ← 0
j ← 0
for idx ← 0 to PanjangPlainteks - 1 do
   i ← (i + 1) mod 256
   j ← (j + S[i]) mod 256
   swap(S[i], S[j])
   t ← (S[i] + S[j]) mod 256
   u ← S[t]              (* keystream *)
   c ← u ⊕ P[idx]
endfor
```


0
t
i
j
255
...
swap
(S[i] + S[j]) mod 256
S[i]
S[j]

- Sampai saat ini tidak ada yang dapat memecahkan RC4 sehinggat dapat dikatakan sangat kuat.

- Terdapat laporan versi kunci 40 bit dapat dipecahkan secara *brute force*.

- *RC4* juga mudah diserang dengan *known-plaintext attack*, dengan cara meng-XOR-kan dua set *byte* cipherteks (kelemahan umum pada *cipher*-aliran)

Aplikasi:

1. Cocok untuk enkripsi berkas citra (derajat keabuan 0 – 255)

   ➔ Demo TA Anil Dhawan (IF 2000).

2. Cocok untuk enkripsi *record* atau *field* basis data (karena ukuran cipherteks = plainteks)

   ➔ TA Dicky Ecklesia (IF 2001)

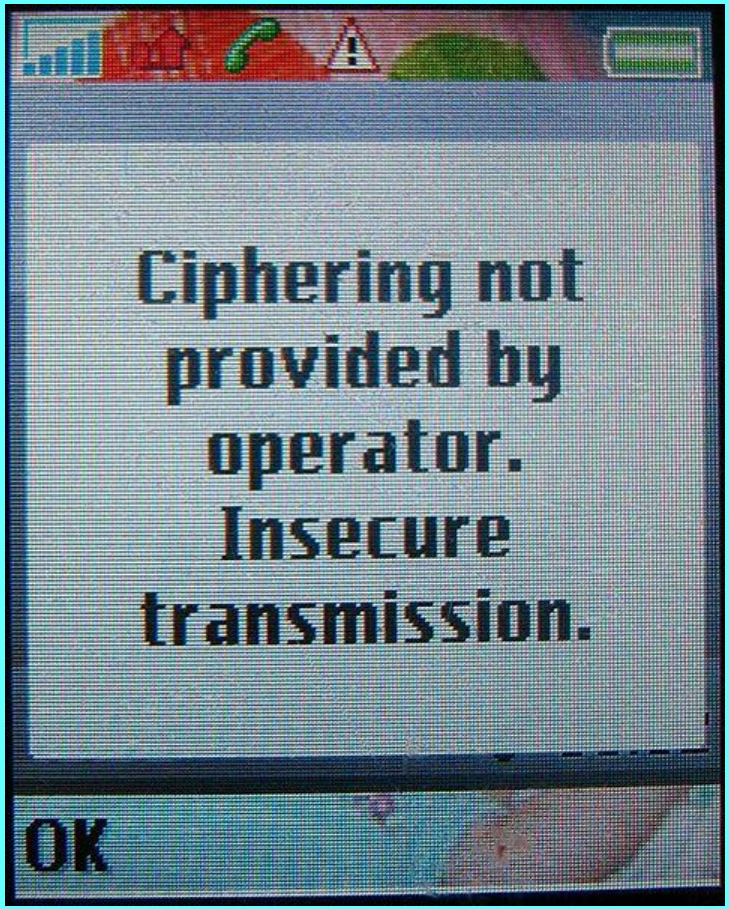
A5

- *A5*: *cipher* aliran yang digunakan untuk mengenkripsi transmisi sinyal percakapan dari standard telepon seluler *GSM* (*Group Special Mobile*).

- Sinyal *GSM* dikirim sebagai barisan *frame*. Satu *frame* panjangnya 228 bit dan dikirim setiap 4,6 milidetik.

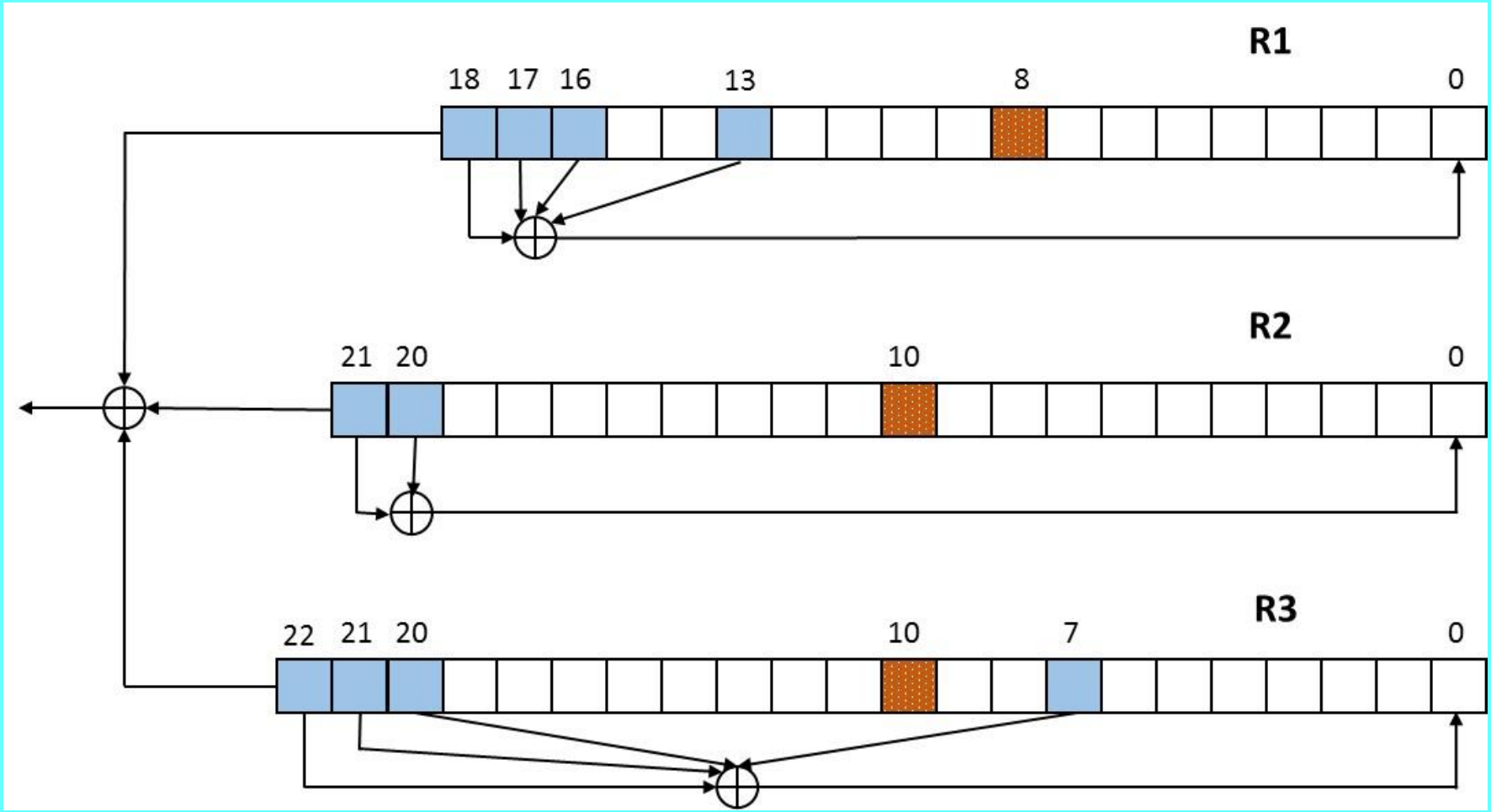
- *A5* digunakan untuk untuk menghasilkan aliran-kunci (*keystream*) 228-bit yang kemudian di-*XOR*-kan dengan *frame*. Kunci eksternal (*session key*) panjangnya 64 bit.

- GSM merupakan standard telepon seluler Eropa
- A5 Dibuat oleh Perancis
- Tidak semua operator GSM mengimplementasikan enkripsi, bergantung regulasi (seperti di Indonesia)
- A5 ada dua versi:
    1. A5/1 : versi kuat A5, digunakan di Eropa
    2. A5/2 (Kasumi) : versi ekspor, lebih lemah
- Algoritma A5/1 pada awalnya rahasia, tetapi pada tahun 1994 melalui *reverse engineering*, algoritmanya terbongkar.



Kasus dimana operator GSM tidak mengenkripsi transmisi percakapan (Sumber: Wikipedia)

- *A5* terdiri dari 3 buah *LFSR* , masing-masing panjangnya 19, 22, dan 23 bit (total = 19 + 22 + 23 = 64).
- Bit-bit di dalam register diindeks dimana bit paling tidak penting (*LSB*) diindeks dengan 0 (elemen paling kanan).
- Luaran (*output*) dari *A5* adalah hasil *XOR* dari ketiga buah *LFSR* ini.
- *A5* menggunakan tiga buah kendali detak (*clock*) yang variabel: - bit ke-8 pada register 1
  - bit ke-10 pada register 2
  - bit ke-10 pada register 3


R1
18
17
16
13
8
0
R2
21
20
10
0
R3
22
21
20
10
7
0

- Register mula-mula diinisialisasi dengan 0.
- Kunci rahasia (*session key*) *K* sepanjang 64 bit dicampur menurut aturan berikut:

  Ada 64 putaran (why?). Untuk putaran *ke-i*, bit ke-*i* ditambah dengan bit *LSB* dari setiap register dengan operasi XOR:

  $$R[0] \leftarrow R[0] + K[i]$$

- Tiap register kemudian didetak (*clocked*).

- Tiap register didetak (*clock*) berdasarkan bit pertengahannya (masing-masing: 8, 10, dan 10).

- Setiap register mempunyai bit pendetakan (*tapped bit)* yang berbeda-beda (Contoh: bit ke-18, 17, 16, 13 pada register 1).

- Pada setiap siklus (i=1..64), pendetakan menggunakan aturan mayoritas: Bit-bit tengah diperiksa dan bit mayoritasnya ditentukan. Jika bit tengah sebuah register sama dengan bit mayoritas, maka register tersebut didetak (dipompa ke kiri)

- Pada setiap siklus, paling sedikit dua atau register yang didetak. Peluang sebuah register didetak pada setiap siklus adalah 3/4.

- *Cipher* menghasilkan *keytream* yang panjangnya 228 bit untuk kemudian dienkripsi dengan meng-XOR-kan nya dengan setiap *frame*.

- Laporan mengenai serangan pada A5/1 dapat dibaca di sini: http://en.wikipedia.org/wiki/A5/1

"*A number of attacks on A5/1 have been published. Some require an expensive preprocessing stage after which the cipher can be attacked in minutes or seconds. Until recently, the weaknesses have been passive attacks using the known plaintext assumption. In 2003, more serious weaknesses were identified which can be exploited in the ciphertext-only scenario, or by an active attacker. In 2006 Elad Barkan, Eli Biham and Nathan Keller demonstrated attacks against A5/1, A5/3, or even GPRS that allow attackers to tap GSM mobile phone conversations and decrypt them either in real-time, or at any later time.*"

(Sumber: Wikipedia)