Network Security

Unit - 2

Lecture slides by

Dr.Capt. Ravindra Babu Kallam

UNIT-2 Part-1

Topics to be covered:

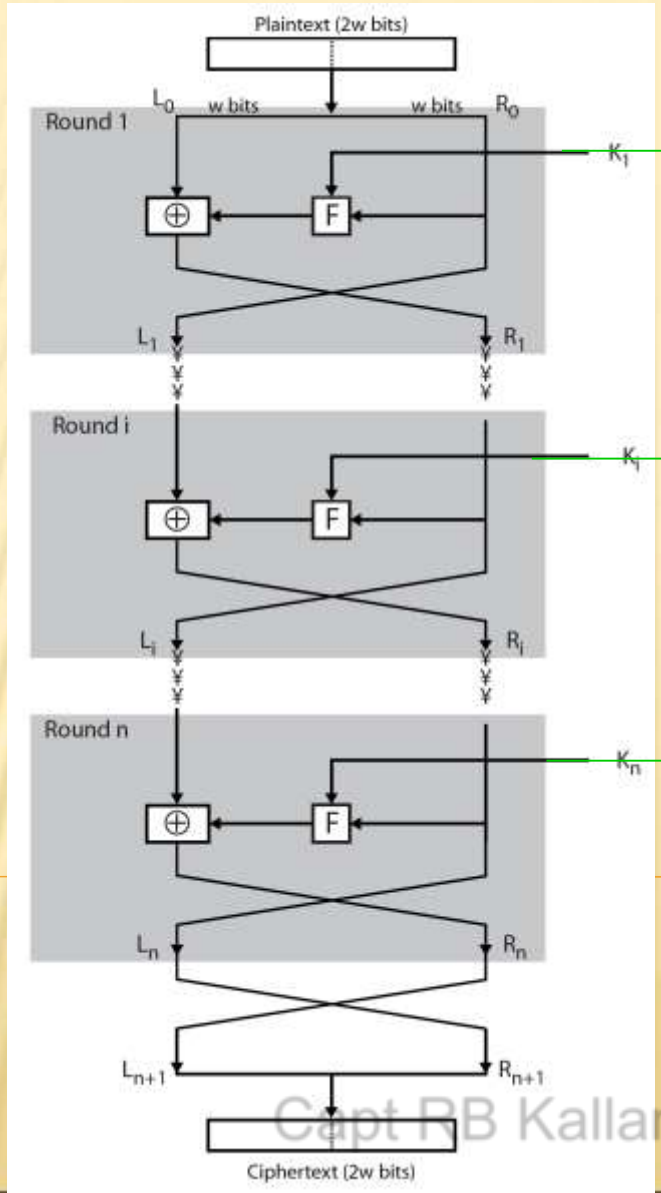
- Symmetric key Ciphers:
  - Block Cipher principles,
  - DES,
  - AES,
  - Blowfish, RC5, IDEA,
- Block cipher operation,
- Stream ciphers, RC4
- Location of Encryption Devices

Symmetric key Ciphers:

Feistel Cipher Structure

- Horst Feistel of IBM devised the **feistel cipher** Structure in 1973.
- The input to the algorithm are a plaintext of length $2w$ bits and a key K.
- The plain text block is divided in to 2 halves, L_0 and R_0.
- The two halves of the data pass through *n* rounds of processing and then combined to produce the cipher text block.
- Each round **i** has an inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as a sub key Ki, derived from the overall K.

Feistel Cipher Structure



Key

Sub key generation algorithm

Block Cipher Design Principles:

- Block size: Larger block size means the greater security but reduces the E/Decryption speed. Block size of 64 bit is reasonable.
- Key size: Larger key size means the greater security but reduces the E/Decryption speed. Most common key length is 128 bits.
- Number of rounds: A single round offer inadequate security but multiple rounds offer increasing security. A typical size is 16 rounds.
- Sub key generation algorithm: Greater complexity in this algorithm should lead greater difficulty of cryptanalysis.

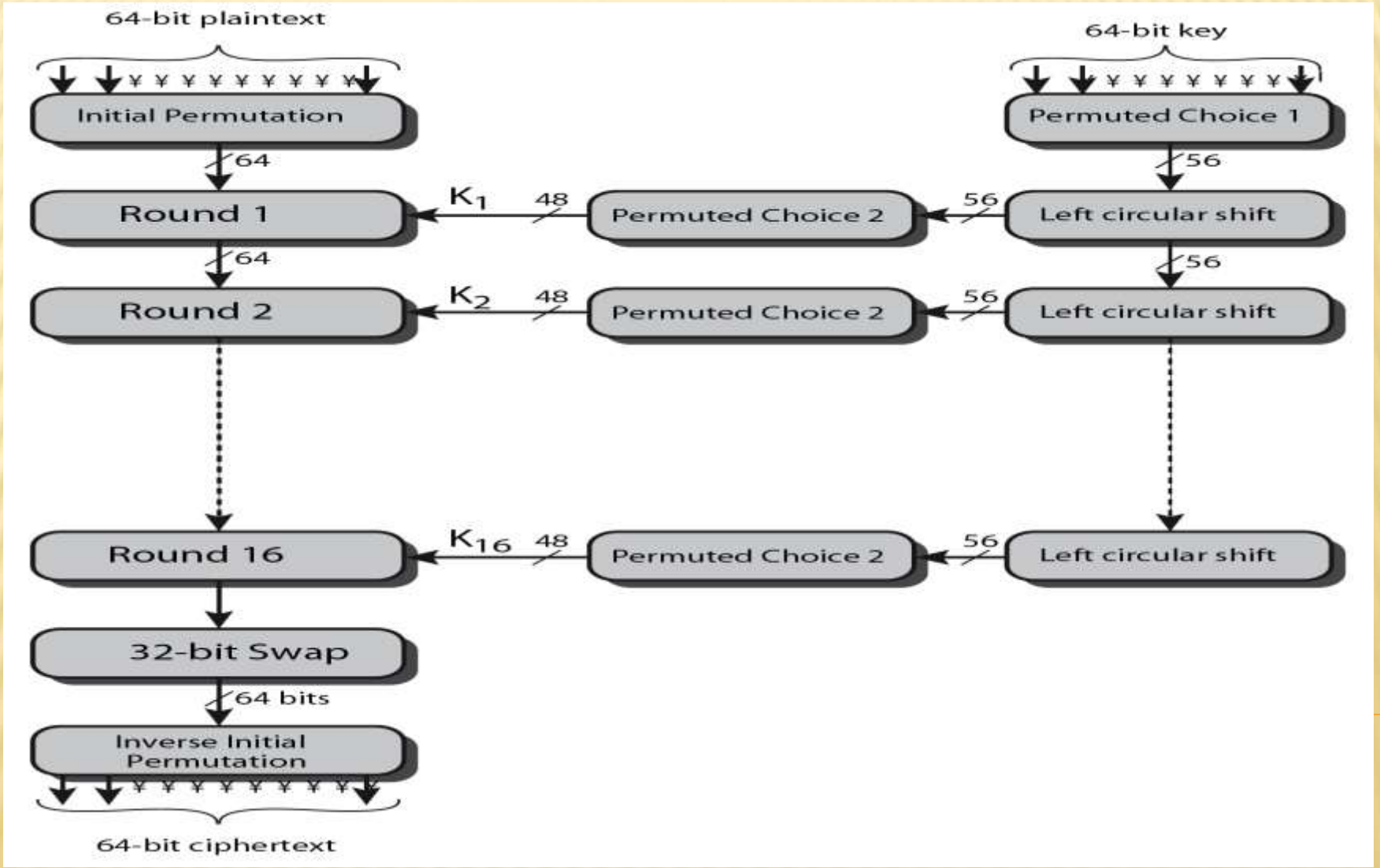
- Round function: Greater complexity means greater resistance to cryptanalysis.
- Fast software en/decryption: In many cases encryption is embedded in applications or utility functions in such a way as to preclude a hard ware implementation. Accordingly the speed of the execution of the algorithm becomes a concern.
- Ease of analysis: although we would like to make our algorithm as difficult as possible to crypt analyze, there is a great benefit in making the algorithm easy to analyze.

Conventional Encryption Algorithms:

Data Encryption Standard (DES)

- Most widely used block cipher in world
- Adopted in 1977 by NBS (now NIST)
- Encrypts 64-bit data using 56-bit key
- Has widespread use
- The algorithm it self is referred to as a data encryption algorithm (DEA).

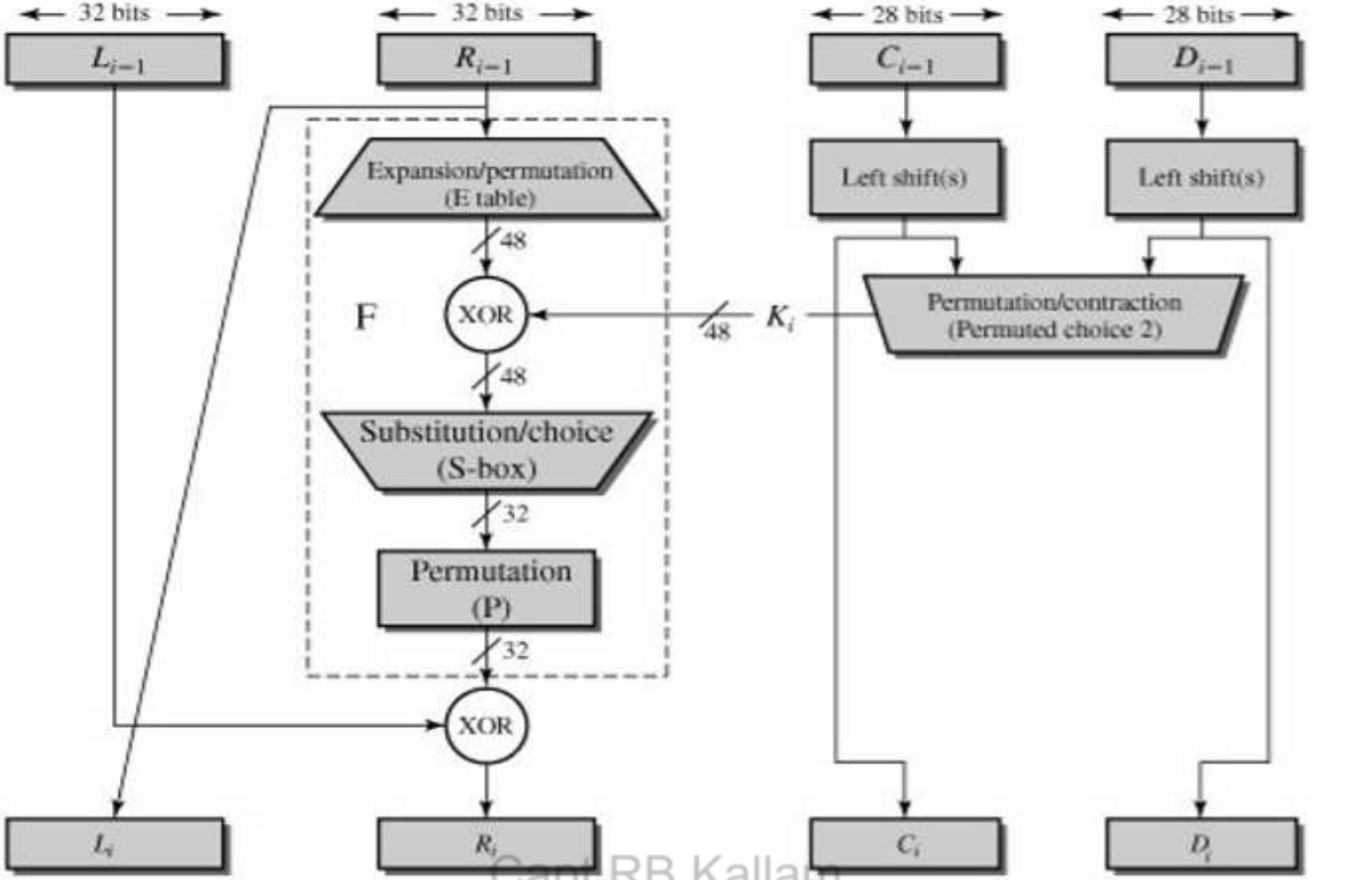
DES Encryption Overview



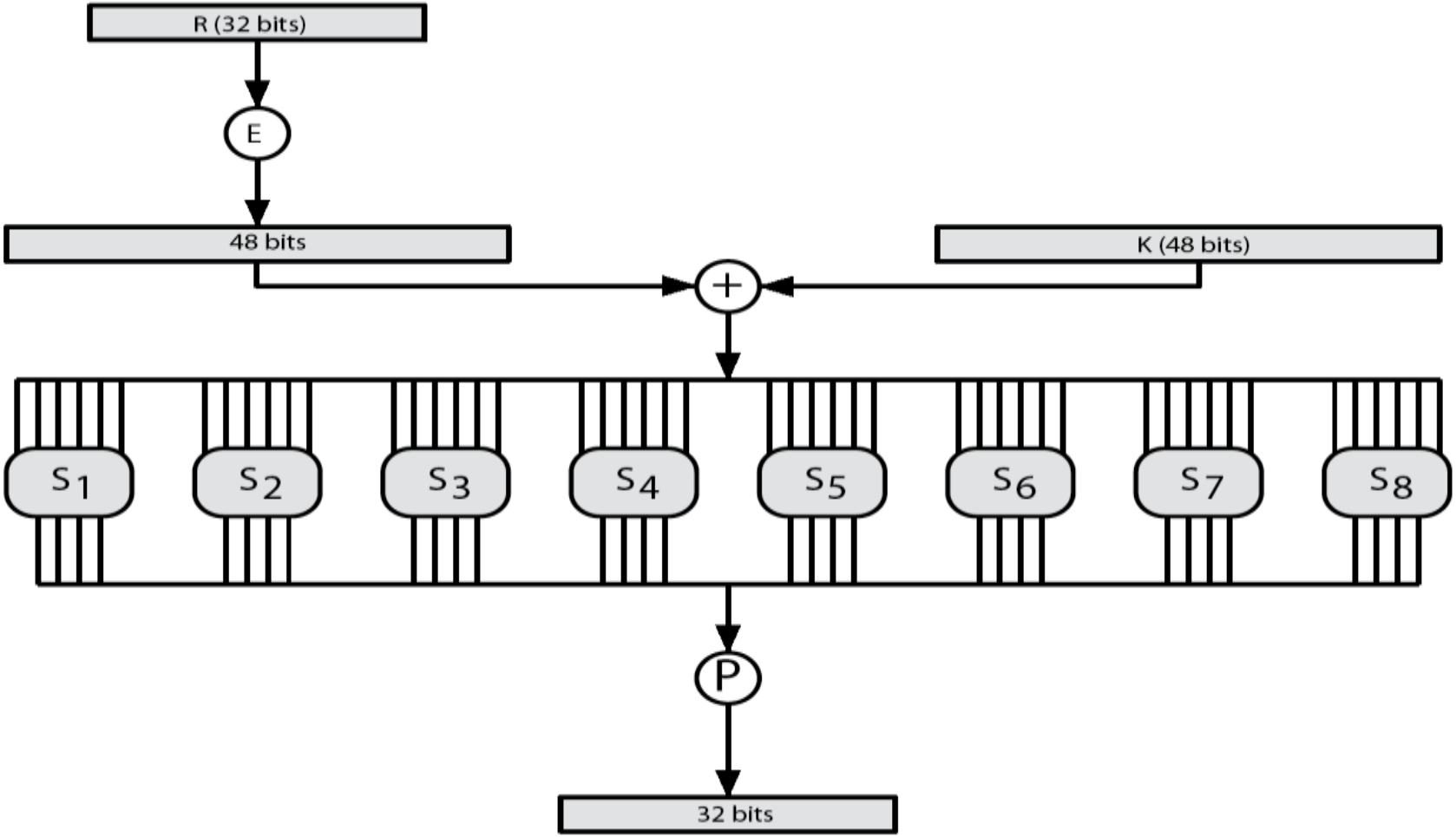
- The left hand side of the figure shows that the processing of the plain text proceeds in three phases.
  - The 64 bit plaintext passes through an IP that rearranges the bits to produce the permuted input.
  - This is followed by a phase consisting of 16 iterations of the same function.

    The output of the last iteration consists of 64 bits that are a function of the input plaintext and the key.
  - The left and right halves of the output are swapped to produce the pre output.
  - Finally, the pre output is passed through a permutation IP^{-1} that is inverse of the initial permutation function, to produce 64 bit cipher text.

SINGLE ROUND OF DES ALGORITHM



DES S-BOXES



- There are 8 S- Boxes, each of them accepting a 6 bit input and producing 4 bit output
- The S-boxes are 4X16 tables and are used as follows:
  - The first and last bit of the input to the S box form a 2 bit binary number that selects the row of the S box
  - The middle four bits select the column of the S box
  - The decimal value in the selected entry of the S box is converted to its 4 bit binary representation to produce the output.

DES S BOX VALUES

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	6
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

(a) Initial Permutation (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

(b) Inverse Initial Permutation (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

(c) Expansion Permutation (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	

(d) Permutation Function (P)

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

SIMPLIFIED DES:

- It was developed by Prof. Edward of Santa Clara University.
- It has the similar properties and structure to DES with much smaller parameters.

Overview:

- The fig below shows the overall structure of DES, which we will refer to as S- DES.
- It takes an 8-bit block of plain text (ex: 10111101) and a 10-bit key as input and produces an 8-bit block of cipher text as output.
- The S-DES decryption algorithm takes an 8-bit block of cipher text and the same 10 –bit key is used to produce the 8_bit block of plain text.

The encryption algorithm involves 5 functions:

- Initial permutation IP
- Complex function f_k, which involves both permutation and substitution operations depends on a key input.
- A simple permutation function that switches (SW) the two halves of the data.
- The function f_k again.
- Finally a permutation function that is inverse of the initial permutation (IP^{-1})

We can express the encryption algorithm function as:

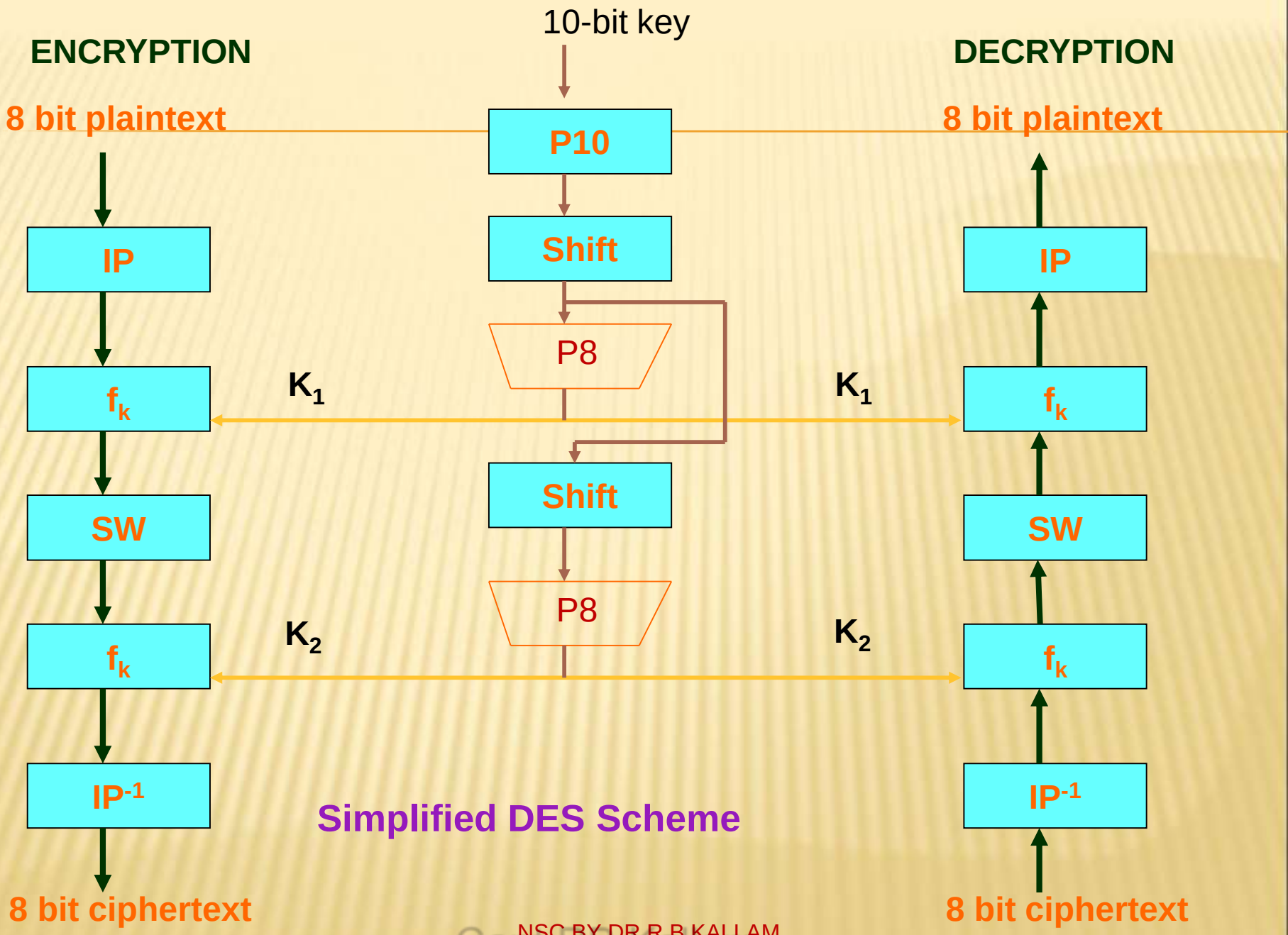
$$\text{Cipher text} = IP^{-1}(f_{k2}(SW(f_{k1}(IP(\text{plaintext})))))$$

Where $K1 = P8(Shift(P10(Key)))$

$$K2 = P8(Shift(Shift(P10(Key))))$$

Decryption algorithm is the reverse of the encryption:

$$\text{Plain text} = IP(f_{k1}(SW(f_{k2}(IP^{-1}(\text{Ciphertext})))))$$



Simplified DES Scheme

Initial Permutation:

- The input to the algorithm is an 8 bit block plaintext, which we first permute using the IP function.

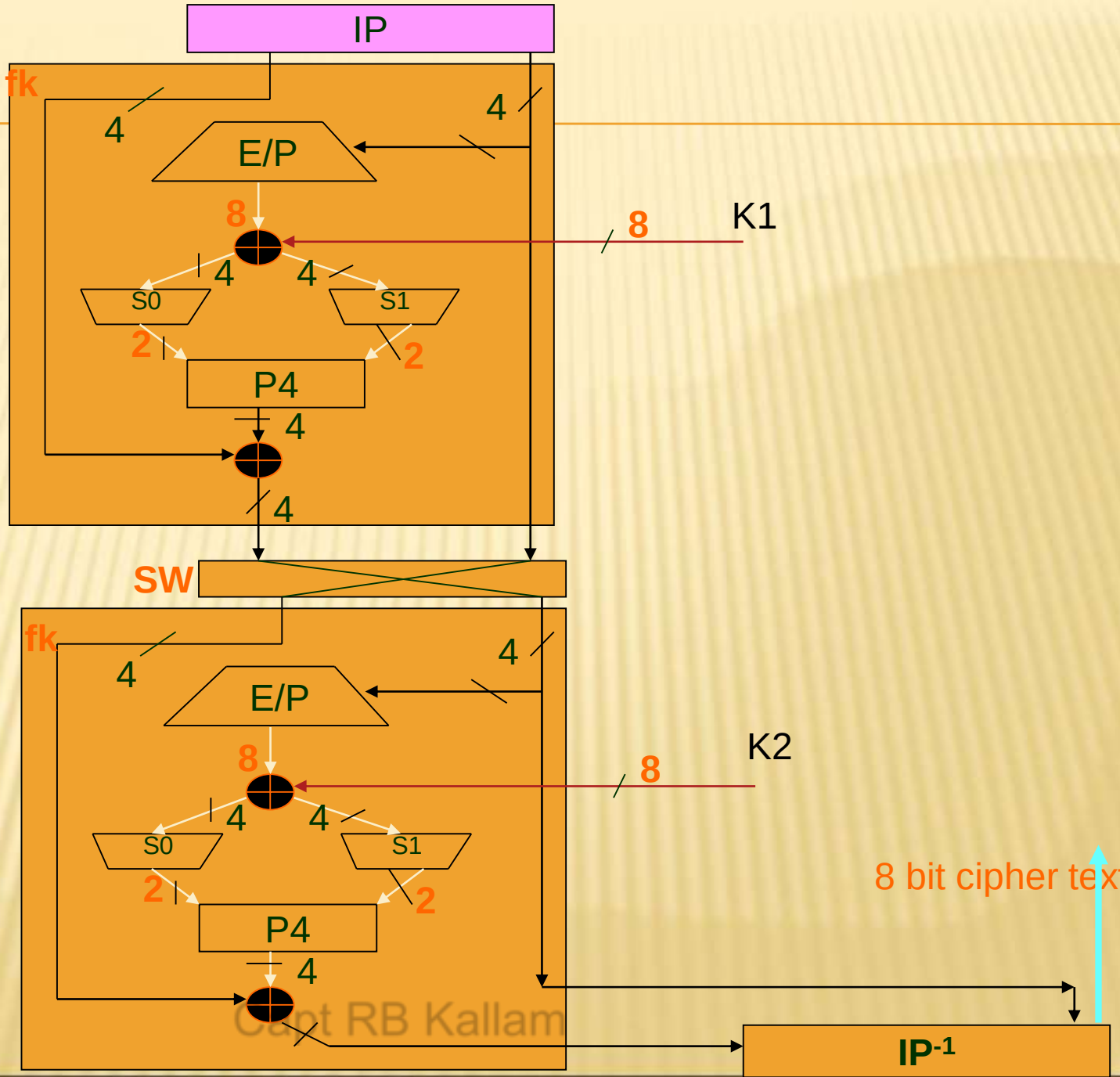
IP							
2	6	3	1	4	8	5	7

- This retain all 8 bits of the plain text but mixes them up. At the end of the algorithm, the inverse permutation is used:

IP^{-1}							
4	1	3	5	7	2	8	6

- Hence, it can be written as $IP^{-1}(IP(X)) = X$

Simplified DES scheme
Encryption Details



The Function f_k:

- The most complex component of S-DES is the function f_k, which consists of a combination of permutation and substitution.
- The function can be expressed as follows:

  - $f_k(L,R) = (L \oplus F(R,SK),R)$

where SK is a sub key and $\oplus$ is the bit by bit exclusive –OR function.

Suppose the out put of IP stage in fig., is :

(10111101) and F(1101,SK) = 1110) for some key SK.

Then f_k(10111101) = (01011101) because (1011) $\oplus$ (1110) = 0101

We now describe the mapping F.
The input is a 4 – bit number ($n_1 n_2 n_3 n_4$).
The first operation is an expansion / permutation operation:

E / P							
4	1	2	3	2	3	4	1

- The first 4 bits are fed in to the S – box S0 to produce a 2- bit output, and the remaining 4 bits are fed in to the S1 to produce another 2 – bit output.
- These 2 boxes are defined as follows:

S0 =

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

S1 =

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

- The first and fourth input bits are treated as a 2 – bit number that specify a row of the S- box, and second and third input bits specify a column of the S- box.
- The entry in the row and column in base two is the 2-bit output.
- Example, if (P_{00}, $P_{0,3}$) = (00)and (P_{01}, P_{02}) = (10) then the out put is from row0 column 2 of S0, which is 3, or binary (11).
- Similarly (P_{10}, $P_{1,3}$) and (P_{11}, P_{12}) are used to index in to a row and column of S1 to produce an additional 2 bits.
- The 4 bits produced by S0 and S1 undergo a further permutation as follows:

P4			
2	4	3	1

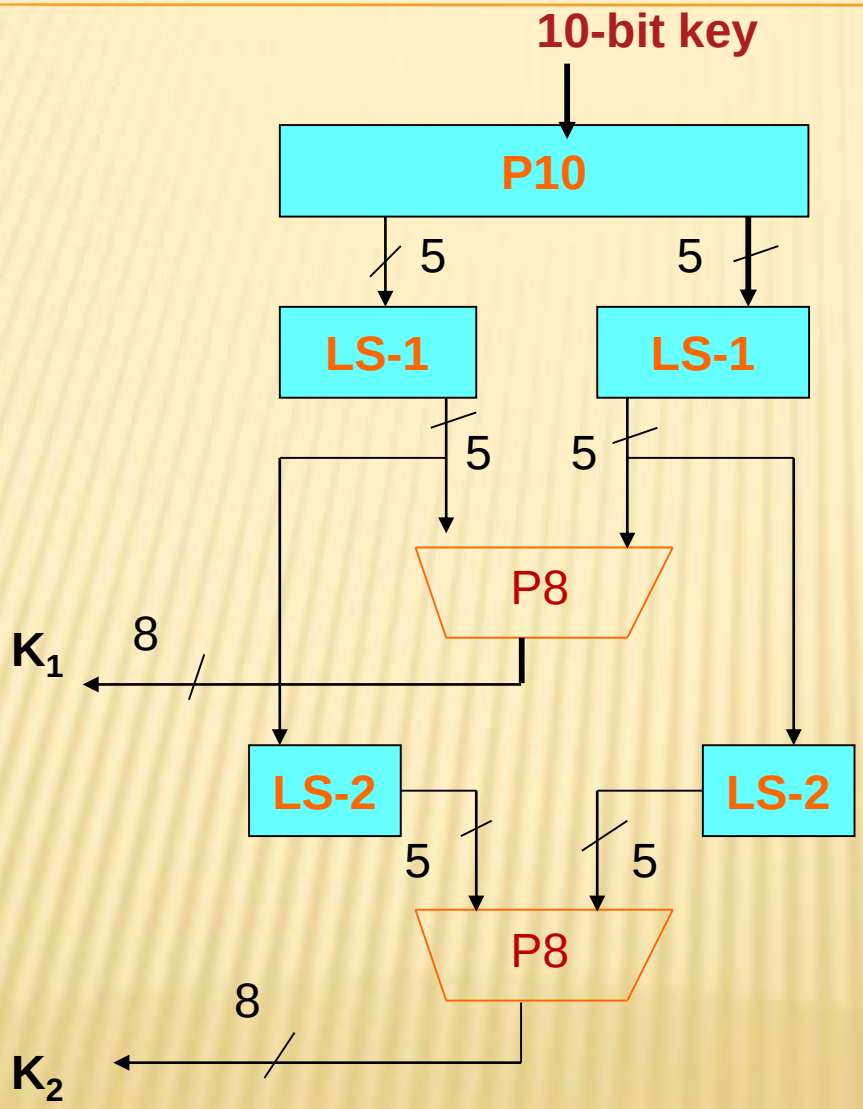
- Switch Function:

The function f_k only alters the leftmost 4 bits of the input. The switch function (SW) interchanges the left and right 4 bits so that the second instance of f_k operates on a different 4 bits.

- In the second instance the E/P,S0,S1 and P4 functions are the same. The key input is K_2

```
enter the message:-
1
0
1
0
1
0
1
0
Enter the Key1
1
1
1
0
0
0
0
enter the scond key
0
0
0
0
1
1
1
1

After the intial permutation:--00110011
Result after expansion  xor with k1:--
01100110

swith box result:-1011
After the first round:-0100
0011
After the exor operation wih k2:---
00100111

switch box result:--0011
after the second round:--0101
0100

The transmitted cipher text is:--10000101_
```

KEY GENERATION FOR S – DES:



S- DES Key Generation

- S-DES depends on the use of a 10 bit key shared between sender and receiver.
- From this key two 8 bit sub keys are produced for use in particular stages of E/Decryption algorithms.
- First, permute the key in the following fashion:
- Let the 10 bit key designated as (k1,k2,k3,k4,k5,k6,k7,k8,k9,k10).Then the permutation P10 is defined as
- P10(k1,k2,k3,k4,k5,k6,k7,k8,k9,k10) = (k3,k5,k2,k7,k4,k10,k1,k9,k8,k6)

- P10 can be defined by the following display:

P10									
3	5	2	7	4	10	1	9	8	6

- This table is red from the left to right.
- Perform a circular left shift (LS-1) or rotation, separately on the first 5 bits and the second five bits.
- Next apply P8, which picks out and permutes 8 of the 10 bits according to the following rule: The result is K1 and apply the same to generate K2.

P8							
6	3	7	4	8	5	10	9

- Result 1:
- Enter the 10 digit integer number- 2 3 5 6 7 9 8 1 4 10
- After the Permutation P10: 5 7 3 8 6 10 2 4 1 9

- Key 1 is----------> 2 8 4 6 1 5 10 9
- Key 2 is----------> 1 5 9 7 10 3 4 2

- Result 2:
- Enter the 10 digit integer number- 5 6 7 8 9 1 2 3 4 10
- After the Permutation P10: 7 9 6 2 8 10 5 4 3 1

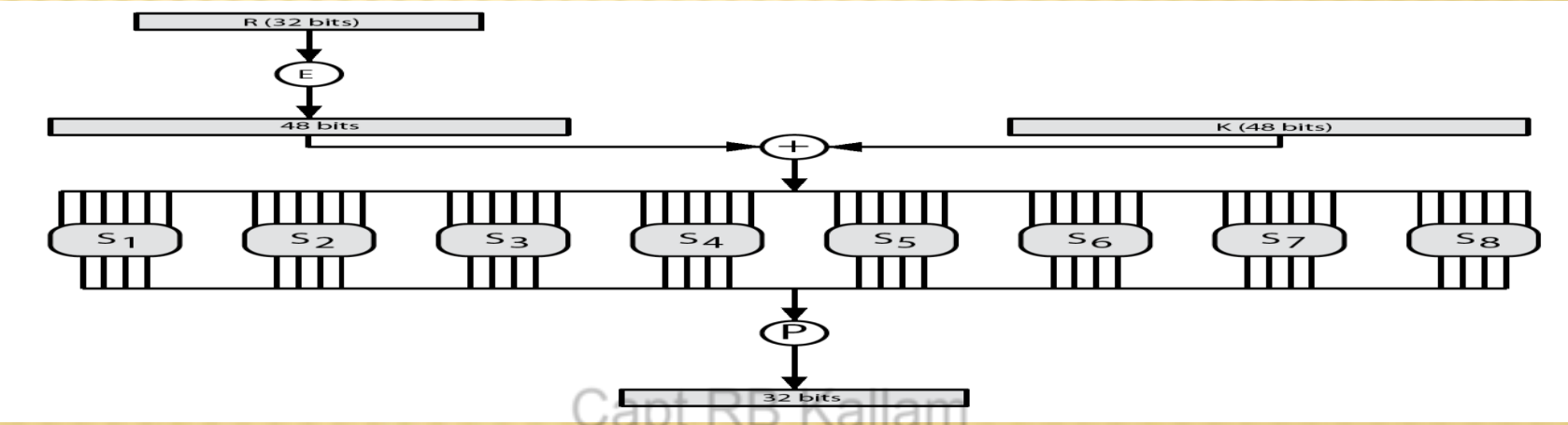
- Key 1 is----------> 5 2 4 8 3 7 10 1
- Key 2 is----------> 3 7 1 9 10 6 4 5

Strength of DES:

- It has 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values, brute force search looks hard, because to perform 1 encryption/ *us* requires 1142 years.

- In July 1998, the Electronic Frontier Foundation (EFF) announced that it had broken a new DES encryption using a special" DES cracker" machine that was built for less than $2,50,000. The attack took less than three days. Hence, for more security key length should be increased.

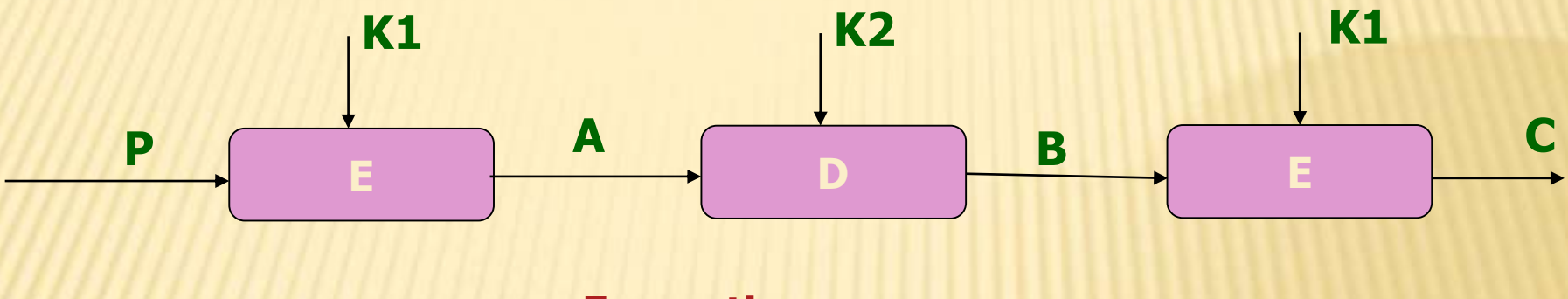
- It has 8 S-boxes, that are used in each iteration. The design criteria for these boxes were **not made public**, there is a suspension that how the boxes were constructed. Cryptanalysis is possible only when knows the weakness of S-box.



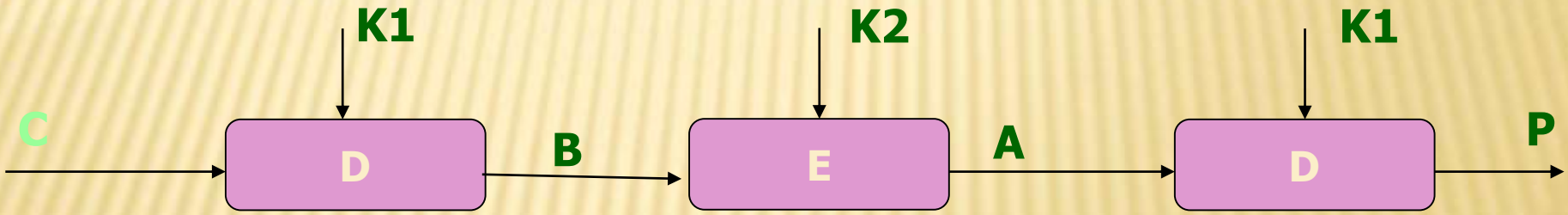
Triple-DES with Two-Keys

- Triple-DES with two keys is a popular alternative to single-DES, but suffers from being 3 times slower to run.
- It must use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks

Triple encryption with 2 keys:



Encryption



Decryption

Triple-DES with Three-Keys:

- Although, no practical attacks on two-key Triple-DES, can use Triple-DES with Three-Keys for more security
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- It has been adopted by some Internet applications, eg PGP, S/MIME, for greater security.

International Data Encryption Algorithm (IDEA):

- The IDEA is a symmetric block cipher algorithm developed in 1991
- It uses 128 bit key and 64 bit block
- It is different from DES both in round function and in the sub key generation function
- For the round function IDEA doesn't use S – boxes. Rather it relies on three different mathematical operations: *XOR, binary addition of 16 bit integers and binary multiplication of 16 bit integers.*

- These functions are combined in such a way as to produce a complex transformation that is very difficult to analyze and hence to crypt analyze.

- The sub key generation algorithm relies completely on the use of *circular shift but uses these in a complex way to generate a total of six sub keys for each of the 8 rounds of IDEA.*
- *In each round it uses 6 keys, hence 6x8=48keys, after 8th round it uses another 4 keys to generate final output.*

- IDEA is used in PGP and also in number of commercial products.

IDEA

→ 64 bit plain text
→ 128 bit key
→ 8 rounds

① $P_1 \times K_1$

② $P_2 + K_2$

③ $P_3 + K_3$

④ $P_4 \times K_4$

⑤ Step 1 ⊕ Step 3

⑥ Step 2 ⊕ Step 4

⑦ Step 5 × K_5

⑧ Step 6 + Step 7

⑨ Step 8 × K_6

⑩ Step 7 + Step 9

⑪ Step 1 ⊕ Step 9 → R_1

⑫ Step 3 ⊕ Step 9 → R_2

⑬ Step 2 ⊕ Step 10 → R_3

⑭ Step 4 ⊕ Step 10 → R_4

o/p of Round 1 → $R_1 R_3 R_2 R_4$
(up to 7 rounds)

o/p of Round 8 → $R_1 R_2 R_3 R_4$

$R_1 \times K_{49} \rightarrow C_1$

$R_2 + K_{50} \rightarrow C_2$

$R_3 + K_{51} \rightarrow C_3$

$R_4 \times K_{52} \rightarrow C_4$

Note: ① Plaintext is devided into 4 parts each is of 16 bits (64/4 = 16).

② Key 128/16 = 8 keys each is of 16 bits.

③ 6 keys are used for each round.
So total 8 × 6 = 48 keys, + 4 additional keys are used after 8th round.

IDEA KEY GENERATION

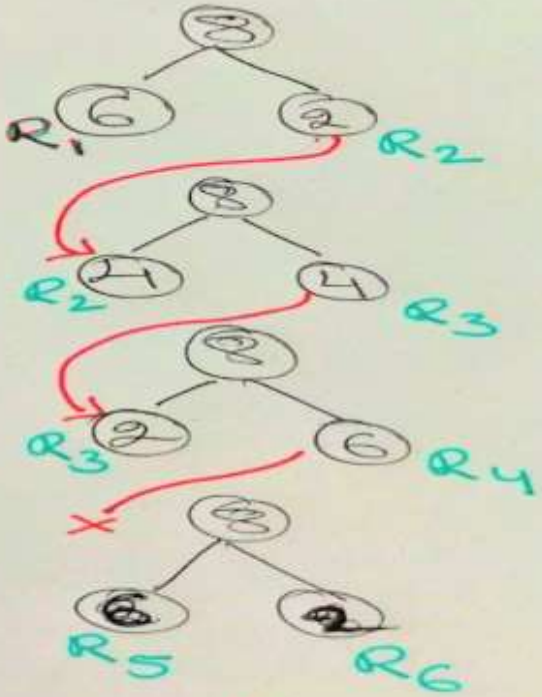
IDEA

⇒ Key = 128/16 = 8 keys

⇒ 16 bits in each key.

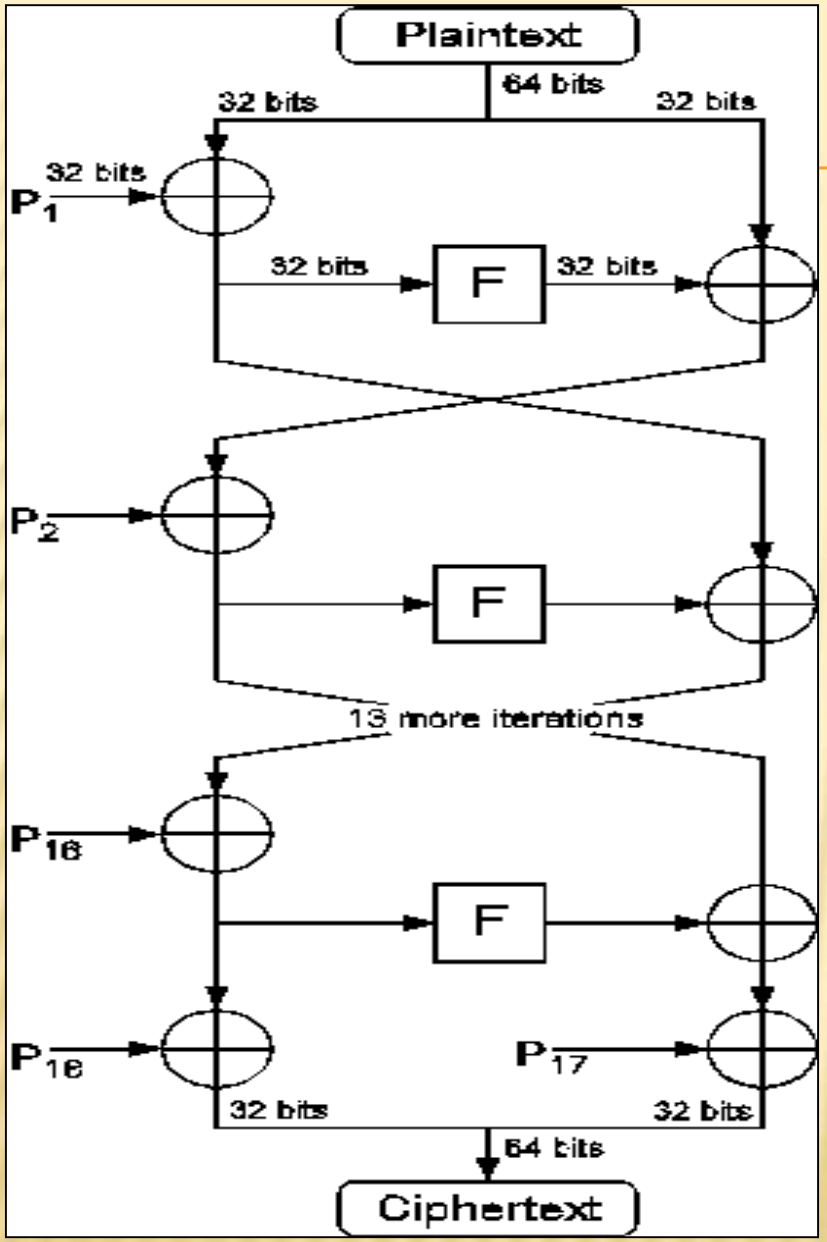
⇒ 25 bits Left Circular Shift after each round.

⇒ 6 keys are used in each round.

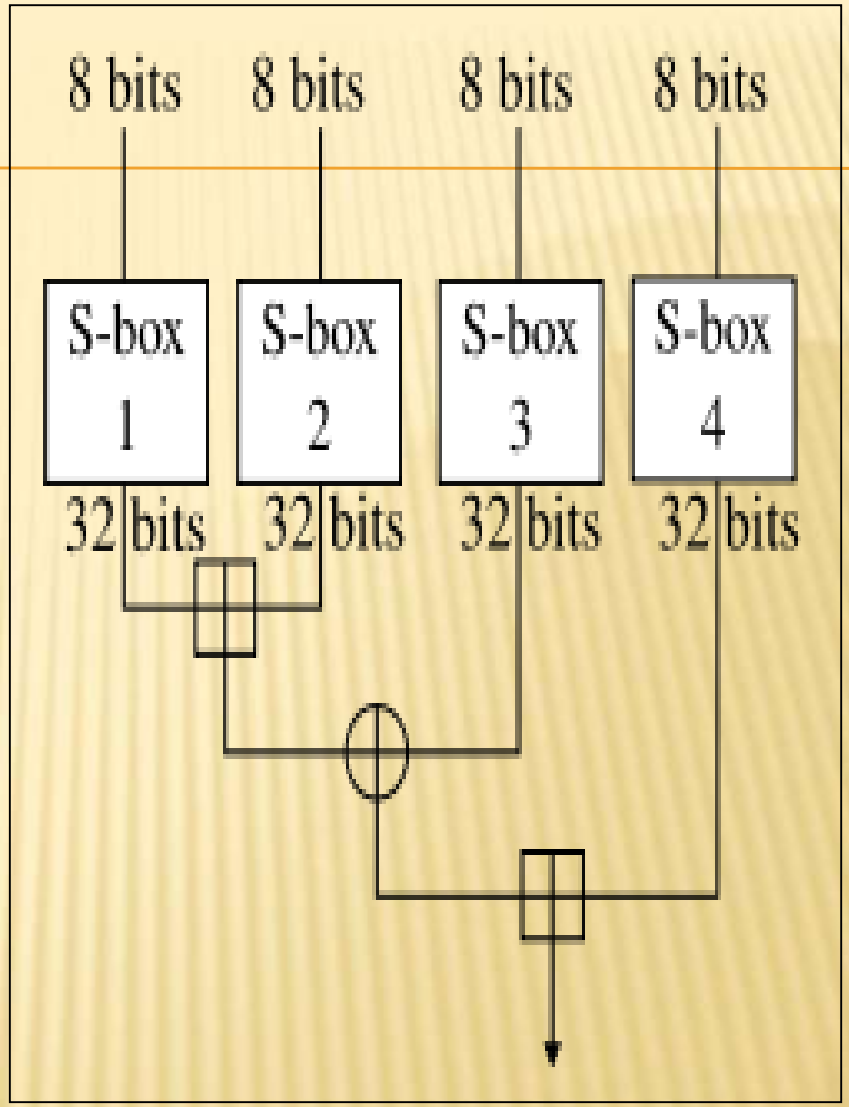


Blow fish:

- It was developed in 1993
- It is one of the most popular alternative to DES.
- It was designed to be easy to implement and to have a high execution speed.
- It is also a very compact algorithm that can run in a less then 5k of memory.
- It is a Block cipher algorithm and the Plain text Block size is 64 bits,
- It is a Symmetric key algorithm, It is a more secure then DES as Key length is variable.
- Key length can be between 32 to 448 bits, default key length is 128bits, it uses 18 sub keys and are stored in P array(P[1] to P[18]).
- For a 32 bit key; In each array element there 8 digits in hexadecimal format so that when we convert them into binary there will be 8x4=32 bits.
- It uses 16 rounds.
- It uses dynamic S-boxes that are generated as a function of the key, XOR function, and binary addition.
- This is not suitable for applications in which the secrete key changes frequently.



Structure of Blowfish algorithm



Structure of Function F

RC5:

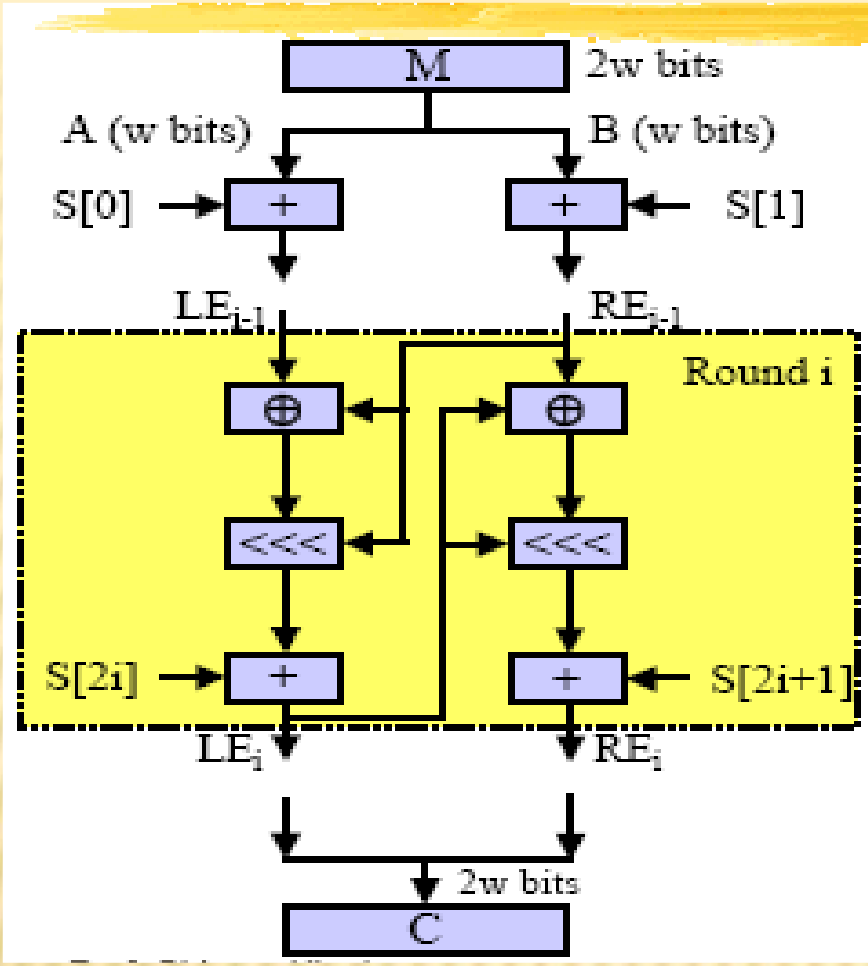
It was developed in 1994

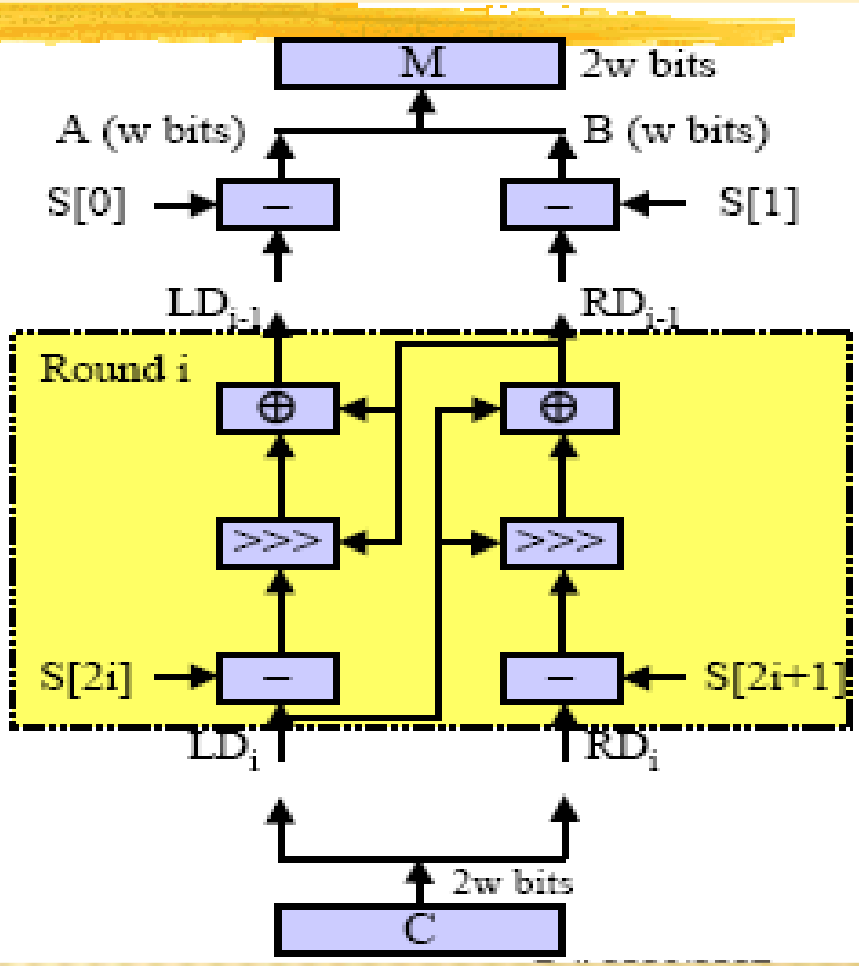
It do not follow classical Feistel Structure

RC5 (Rivest Cipher 5) is defined in RFC2040 and designed to have the following characteristics:

- It is a symmetric key and Block cipher algorithm
- Suitable for hardware and software: it uses only a primitive computational operations commonly found on microprocessors.
- Fast: it a simple algorithm and is word oriented.
- Adaptable to processors of different word length: the number of bits in a word (16,32,64,128) is a first parameter of RC5, different word length yield different algorithms.
- Variable number of rounds:: The number of rounds (0 to 255) is second parameter of RC5 this allows a trade of between speed and security.
- Variable length Key: It is third parameter of RC5. The range of keys can be 0 to 255 Bytes,

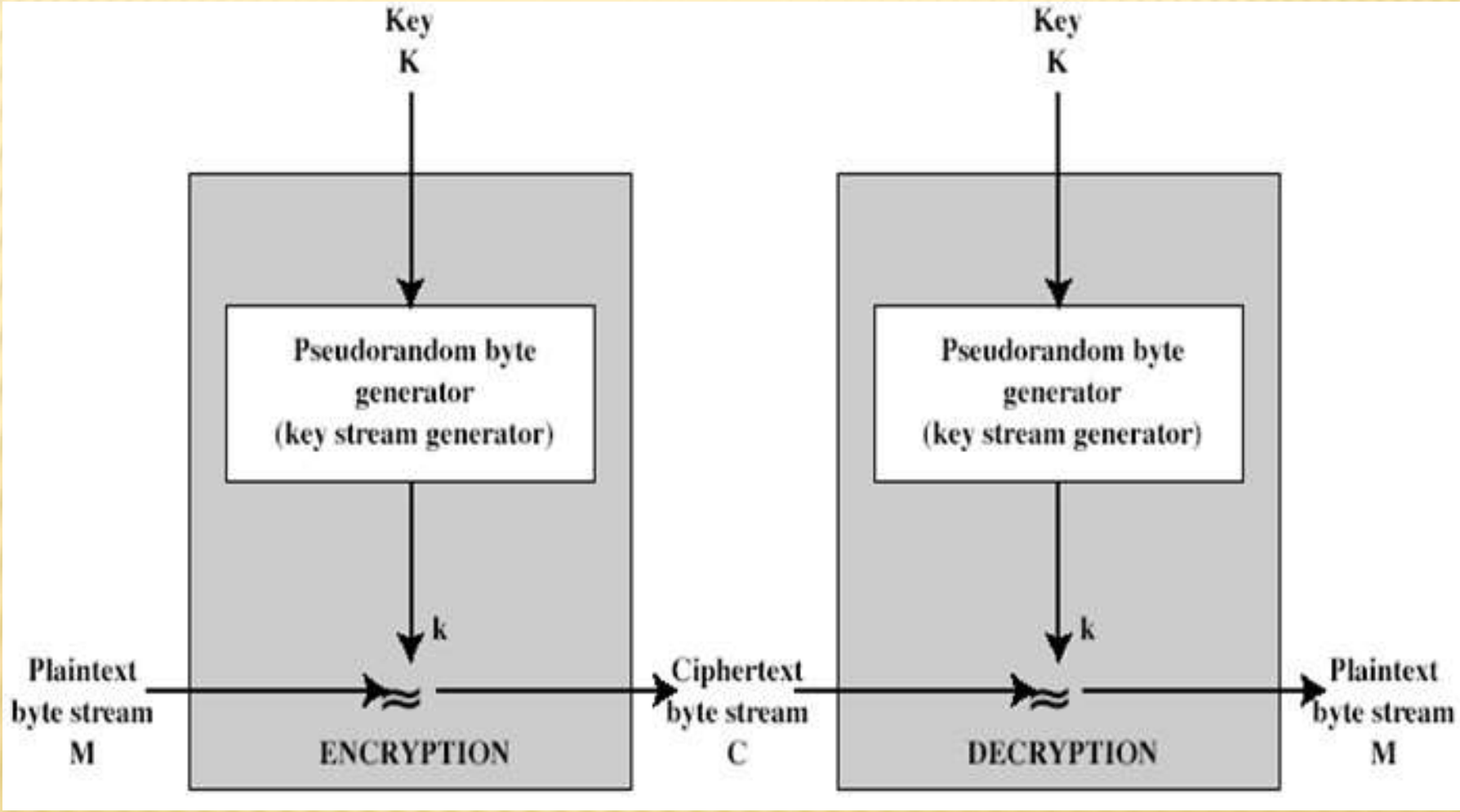
- Simple: RC5 is having simple structure and easy to implement.
- Low memory requirement: it makes RC5 suitable for smart cards and other devices with restricted memory.
- High security: RC5 is intended to provide high security with suitable parameters.
- Data dependent rotation: RC5 incorporate rotations(circular bit shifts) whose amount is data dependent. This strengthen the algorithm against cryptanalysis.
- RC5 is designated as RC5-w/r/b, Ex: RC5-32/12/16, i.e 2-32bit words(2w) i.e 64 bit block size, 12 rounds and 16bytes (128bits) key.
- In each of the r rounds consist of
  - A Substitution using both words of data
  - A Permutation using both words of data
  - A Substitution depends on the key





$LE_0 = A + S[0]$;
$RE_0 = B + S[1]$;
for $i = 1$ to r do
$LE_i = ((LE_{i-1} \oplus RE_{i-1}) <<< RE_{i-1}) + S[2i]$;
$RE_i = ((RE_{i-1} \oplus LE_i) <<< LE_i) + S[2i+1]$;

STREAM CIPHER



Stream cipher structure

- Stream cipher encrypts plaintext one byte at a time.
- As shown in the fig. a key is input to the pseudorandom bit generator that produces a stream of 8 bit numbers that are apparently random.
- A pseudorandom stream is one that is unpredictable without knowledge of the input key.
- The output of the generator is called a Key stream, is XORed one byte at a time with the one byte of plain text stream to produce one byte at a time of cipher stream.

Design considerations of the stream ciphers:

- The encryption sequence should have a larger period because the pseudorandom number generator may sometime produces the same key. The longer period of repeat, it will be more difficult for cryptanalysis.
- The Key stream should have the properties of true random number stream as close as possible. If the key stream is treated as a stream of bytes, then all of the 256 possible byte values should appear equally often.
- The more random appearing the key stream makes cryptanalysis more difficult.
- If the key is larger as in block ciphers, then it will be more difficult for the cryptanalysis. The desirable key length is 128 bits.

- Advantage of stream cipher With proper design of PRNG: stream cipher is as secure as block cipher. Stream cipher is faster than block cipher.
- Disadvantage of stream cipher never reuse the same key.

RC4

- Invented by Ron Rivest in 1987
  - "RC" is "Ron's Code" or "Rivest Cipher"
- It is a symmetric algorithm
- It is a variable key size stream cipher with byte oriented operations.
- Generate key stream byte at a step.
- Efficient in software and simple.
- Used in many applications: SSL, TLS, etc.,
- Most popular stream cipher in existence.

- A variable length key from 1 to 256 bytes (8 to 2048bits) is used to initialize a 256 byte state vector S, with elements S[0],S[1],S[2],.....S[255].
- At all time S contains a permutation of all 8-bit numbers from 0 to 255.
- For encryption and decryption , byte ‘k’ is generated from S by selecting one of the 255 entries in a systematic fashion.
- As each value of k is generated, the entries in S are once again permuted.
- For encryption, k is XORed with the next byte of plain text and for decryption k is XORed with the next byte of Cipher text.

Initialization of S:

- To begin, the entries of S are set equal to the values from 0 through 255 in ascending order; that is, S[0] = 0, S[1] = 1, c,S[255] = 255 .
- A temporary vector, T, is also created. If the length of the key K is 256 bytes, then K is transferred to T. Otherwise, for a key of length *keylen* bytes, the first *keylen* elements of T are copied from K, and then K is repeated as many times as necessary to fill out T.
- /* Initialization */
- for i = 0 to 255 do
- S[i] = i;
- T[i] = K[i mod keylen];
- Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]
- /* Initial Permutation of S */
- j = 0; for i = 0 to 255 do
- j = (j + S[i] + T[i]) mod 256;
- Swap (S[i], S[j]);
- Because the only operation on S is a swap, the only effect is a permutation. S still contains all the numbers from 0 through 255.

Stream Generation:

- Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S.
- After S[255] is reached, the process continues, starting over again at S[0]
- /* Stream Generation */
- i, j = 0;
- while (true) i = (i + 1) mod 256;
- j = (j + S[i]) mod 256;
- Swap (S[i], S[j]);
- t = (S[i] + S[j]) mod 256;
- k = S[t];
- To encrypt, XOR the value k with the next byte of plaintext. To decrypt, XOR the value k with the next byte of ciphertext.

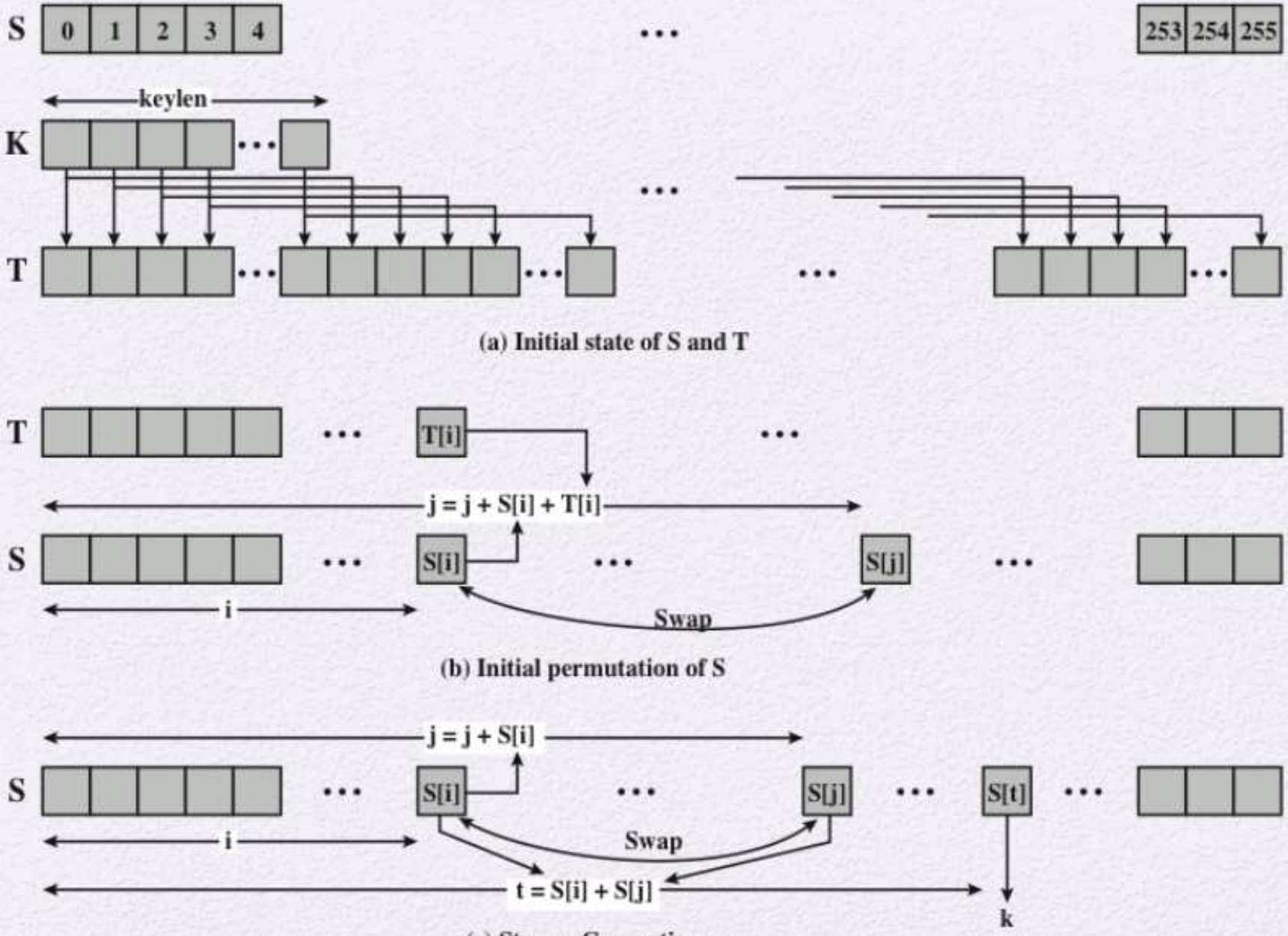


Figure 7.8 RC4

Cipher Block Modes of operations:

- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR) mode

Cipher Block Chaining (CBC)

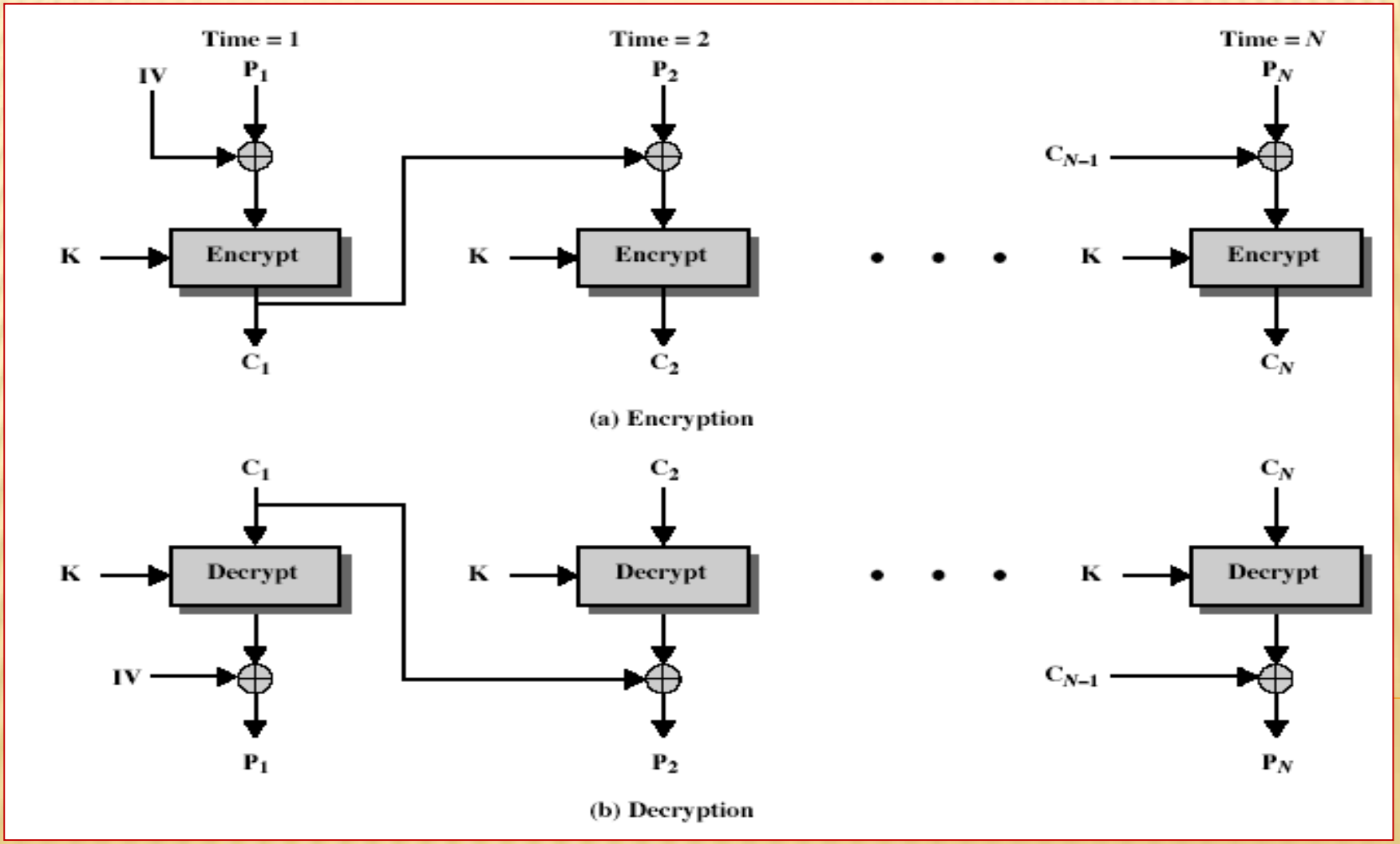
- Message is broken into blocks
- Input to the encryption algorithm is the XOR of the current plaintext block and the preceding cipher text block; the same key is used for each block.
- To produce the first block of cipher, an IV is XORed with the first block of plain text.
- On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plain text.
- Previous cipher block is chained with current plaintext block, hence name.

$$C_1 = E_K(IV \oplus P_1)$$

$$P_1 = IV \oplus D_K(C_1)$$

- CBC is widely used in security applications, specially where bulk data encryption, authentication is required.

Cipher Block Chaining (CBC)



Advantages and Limitations of CBC

- a cipher text block depends on **all** blocks before it. any change to a block affects all following cipher text blocks
- need **Initialization Vector** (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can easily attack.
  - hence IV must either be a fixed value or must be sent encrypted before the rest of the message

Cipher Feed Back (CFB)

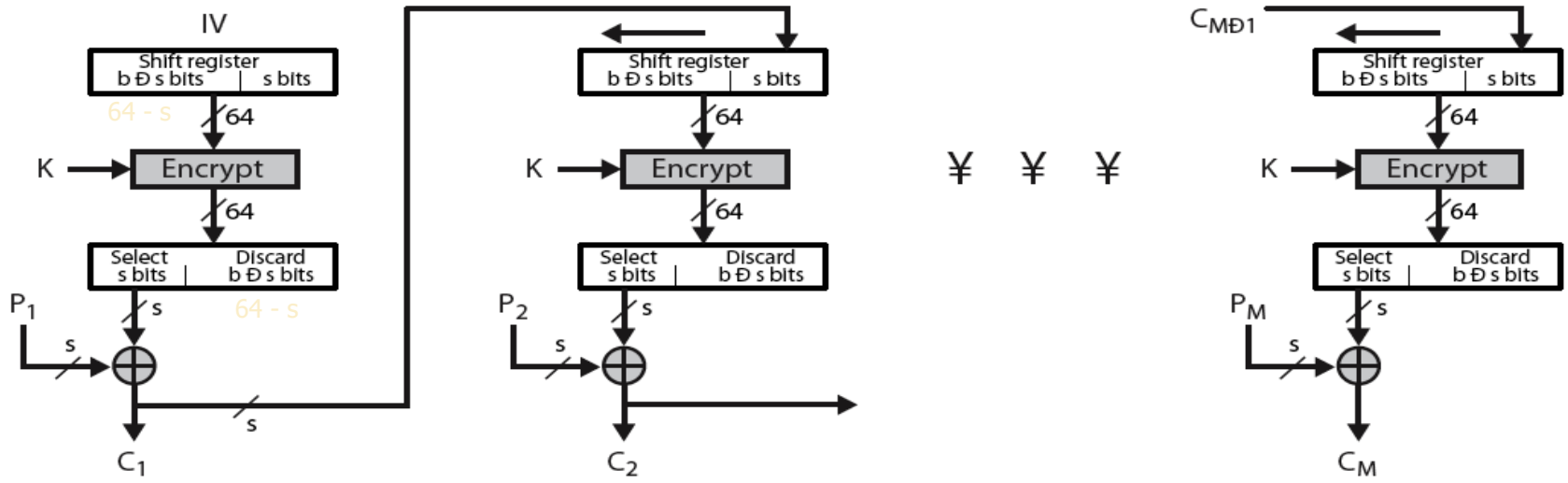
- Message is treated as a stream of bits. It also can operate in real time mode.
- In this cipher text is of the same length of the plain text
- As in the CBC the unit of plain text are chained together, so that the cipher text of any plain text is a function of all preceding plaintext.
- Encryption: the input to the encryption function is a 64 bit shift register that is initially set to some IV.
- The left most S bits of the output of the encryption function are XORed with the first segment of the plain text P1 to produce the first unit of cipher text C1, which is then transmitted.
- In addition the content of the shift register are shifted left by S bits and C1 is placed in the rightmost S bits of the shift register. And this continue till the end of the plain text.

- **Decryption:** the same scheme is used, except that the received cipher text unit is XORed with the output of the encryption function to produce the plain text unit.
- It allows any number of bit (1,8, 64 or 128 etc) to be feedback
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- Most efficient to use all bits in block (64 or 128)
- **Let MSBs(x) be defined as the most significant S bits of X. Then**

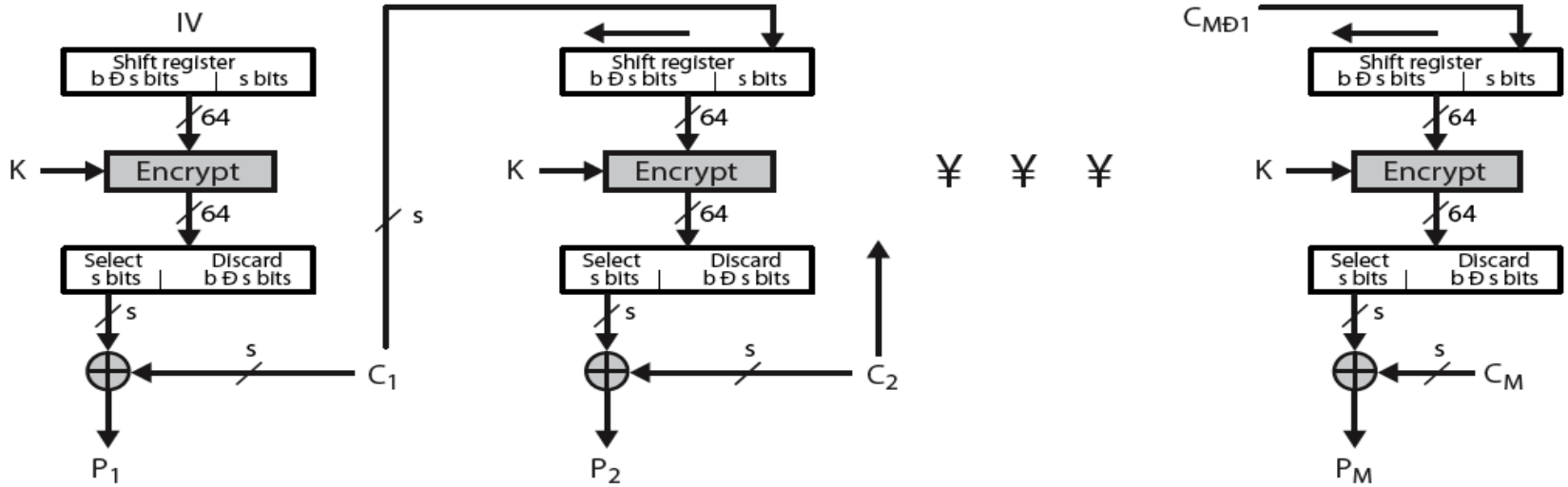
$C_1 = P_1 \oplus MSBs[E(K,IV)]$ and

$P_1 = C_1 \oplus MSBs[E(K,IV)]$

Cipher Feed Back (CFB)



(a) Encryption



(b) Decryption

Advantages and Limitations of CFB

- Appropriate when data arrives in bits/bytes
- Most common stream mode
- It also can operate in real time mode
- It eliminates the need to pad a message to be an integral number of blocks.
- Cipher text is of the same length of the plain text.
- errors propagate for several blocks after the error

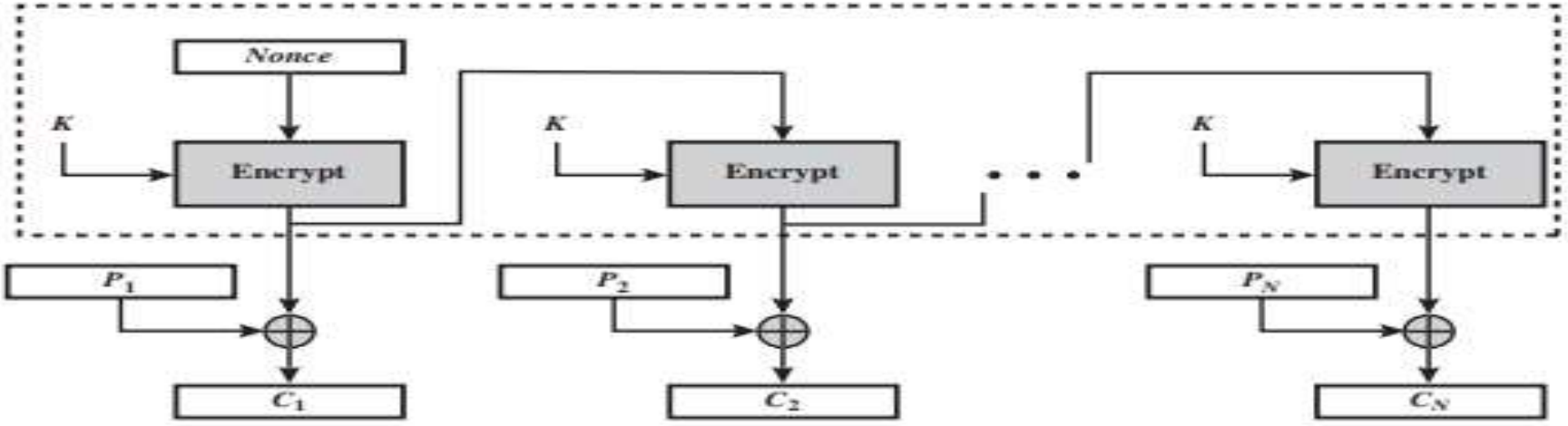
OUTPUT FEEDBACK (OFB) MODE

- The output feedback (OFB) mode is similar in structure to that of CFB.
- For OFB, the output of the encryption function is fed back to become the input for encrypting the next block of plaintext as shown in Figure.
- *In CFB, the output of the XOR unit is fed back to become input for encrypting the next block.*
- The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, whereas CFB operates on an s-bit subset.

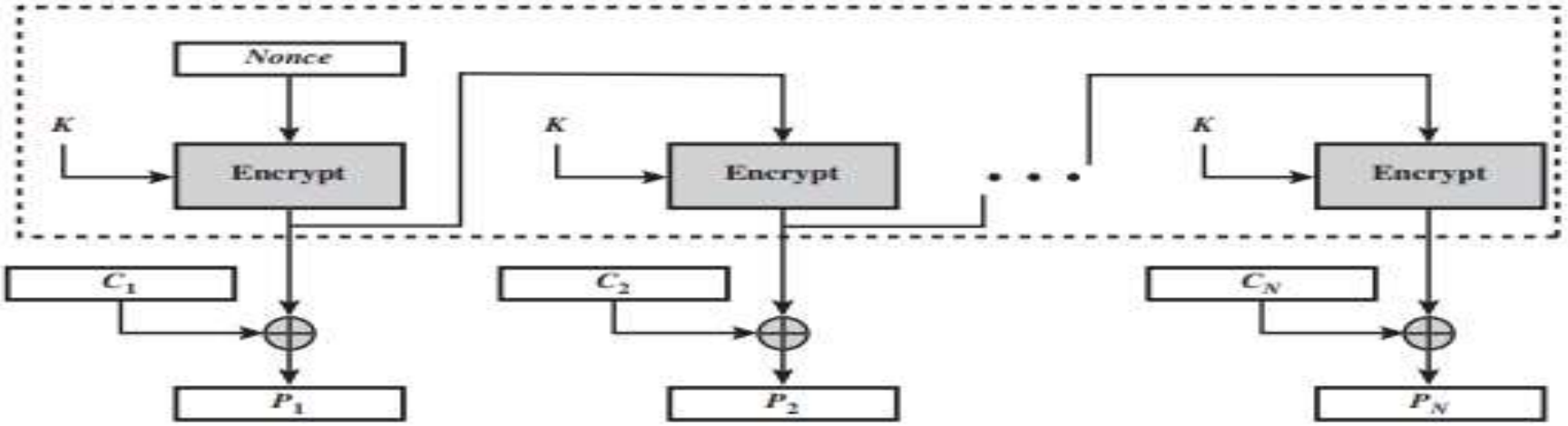
- OFB encryption can be expressed as

$$C_j = P_j \oplus E(K, O_{j-1}), \quad \text{where :} \quad O_{j-1} = E(K, O_{j-2})$$

$$P_j = C_j \oplus E(K, O_{j-1}), \quad \text{where :} \quad O_{j-1} = E(K, O_{j-2})$$



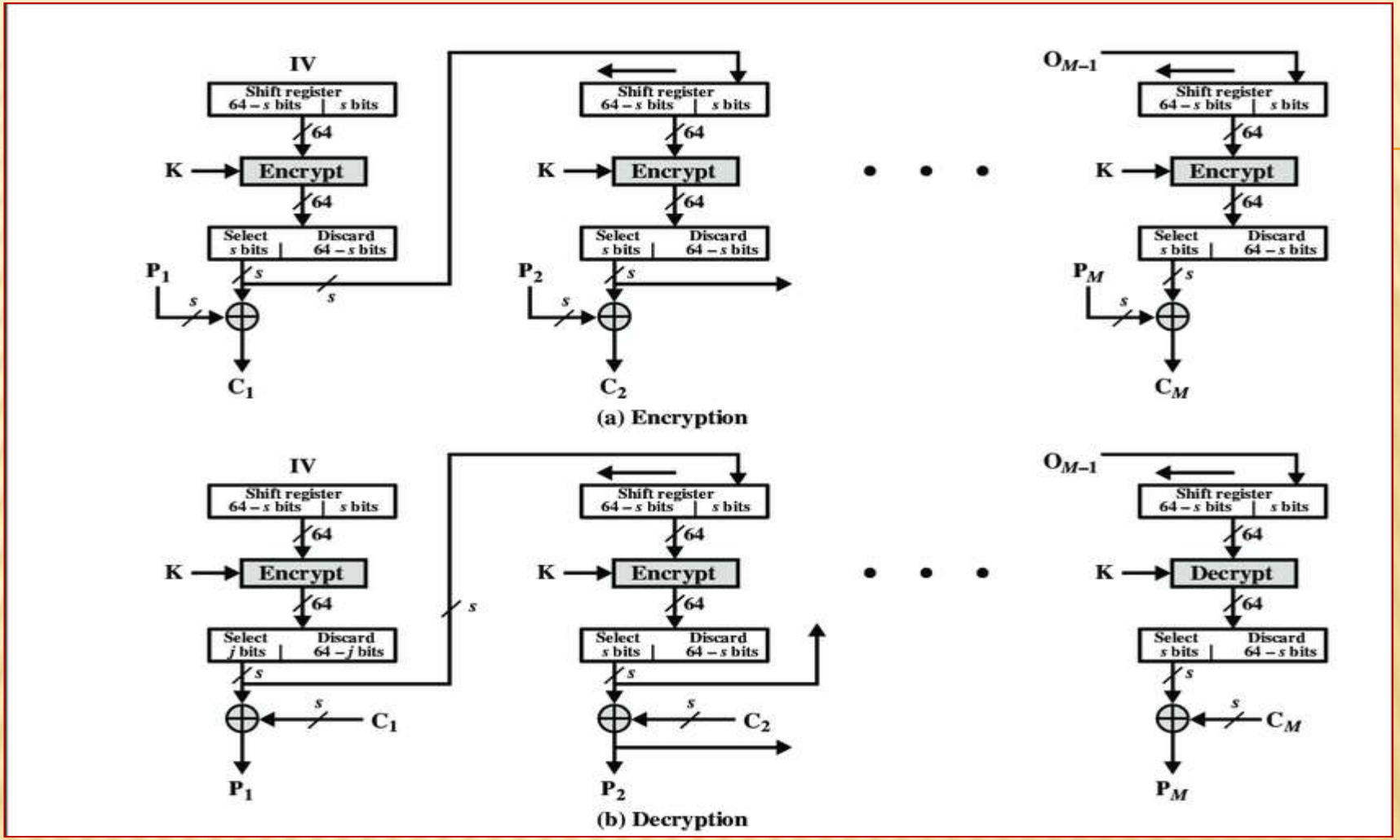
(a) Encryption



(b) Decryption

Figure 6.6 Output Feedback (OFB) Mode

Figure: W.r.f: CNS, Stallings 6th Edition



C1 = P1 ⊕ MSBs(E[K, IV])
P1 = C1 ⊕ MSBs(E[K, IV])

S bit Output Feedback Mode

W.r.f: CNS, Stallings 4th Edition

Disadvantage:

- As with CBC and CFB, the OFB mode requires an initialization vector. In the case of OFB, the IV must be a nonce; that is, the IV must be unique to each execution of the encryption operation.
- The reason for this is that the sequence of encryption output blocks, Oi, depends only on the key and the IV and does not depend on the plaintext.
- Therefore, for a given key and IV, the stream of output bits used to XOR with the stream of plaintext bits is fixed. If two different messages had an identical block of plaintext in the identical position, then an attacker would be able to determine that portion of the Oi stream.

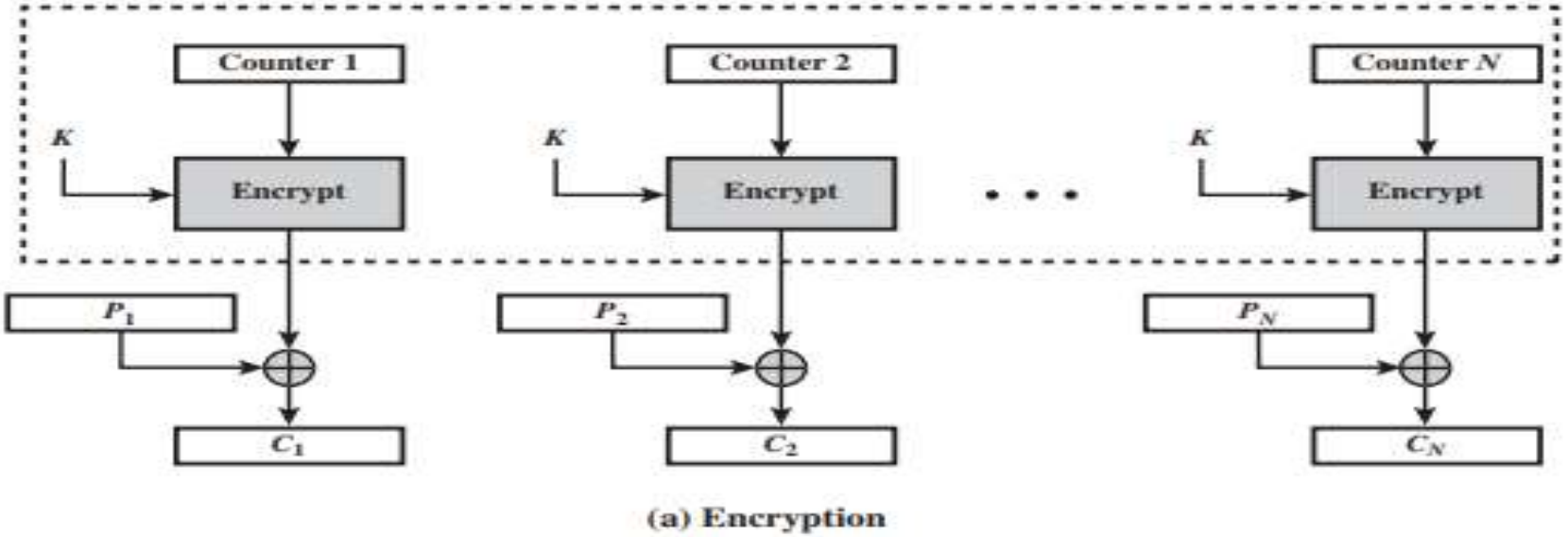
Advantage:

- One advantage of the OFB method is that bit errors in transmission do not propagate. For example, if a bit error occurs in C1, only the recovered value of P1 is affected; subsequent plaintext units are not corrupted.
- With CFB, C1 also serves as input to the shift register and therefore causes additional corruption downstream.
- The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB.

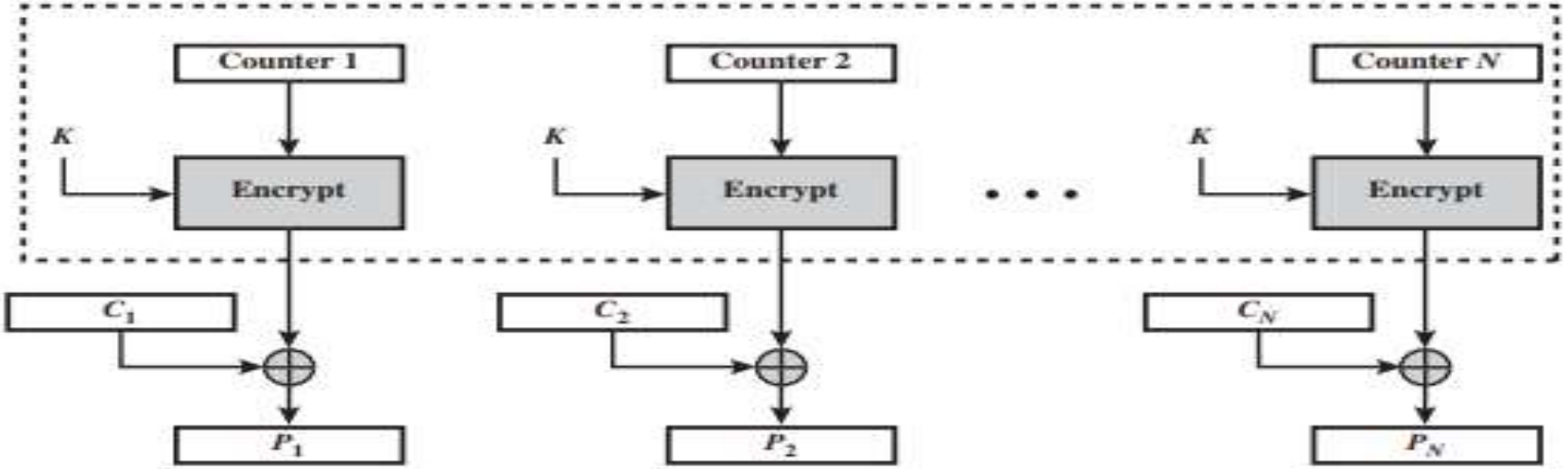
COUNTER (CTR) MODE

- A counter equal to the plaintext block size is used.
- The only requirement is that the counter value must be different for each plaintext block that is encrypted.
- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block.
- For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining.
- For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block.
- Thus, the initial counter value must be made available for decryption. Given a sequence of counters T1, T2, ..., TN, we can define CTR mode as follows.

$$C_j = P_j \oplus E(K, T_j), \quad \text{Where: } j = 1....N-1$$
$$P_j = C_j \oplus E(K, T_j), \quad \text{where: } j = 1....N-1$$



(a) Encryption



(b) Decryption

Figure 6.7 Counter (CTR) Mode

Location of Encryption Devices:

Link Encryption:

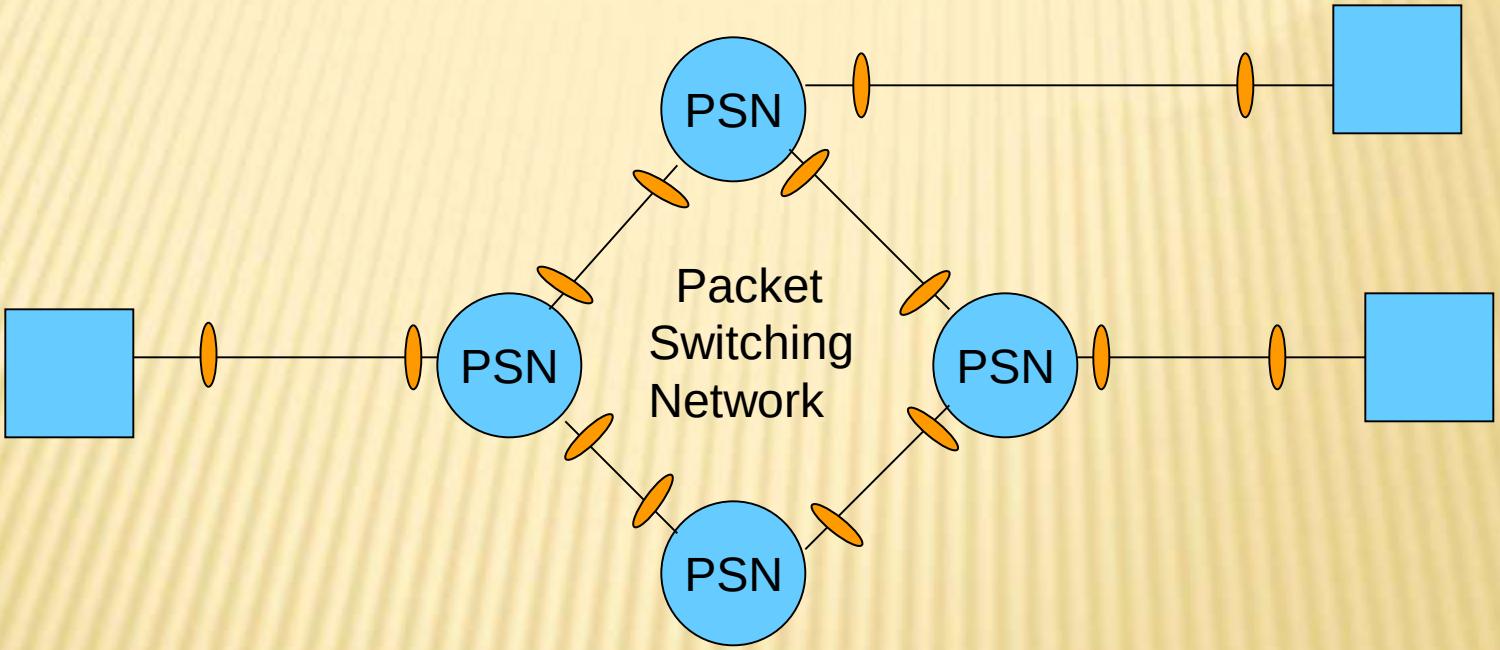
- In this each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured.
- Although this requires a lot of encryption devices in a large network, it provides a high level of security.
- One disadvantage of this approach is that the message must be decrypted each time it enters a packet switch;

  this is necessary because the switch must read the address in the packet header to route the packet.

  Thus, the message is vulnerable at each switch.

  If this is a public packet-switching network, the user has no Control over the security of the nodes.

Link Encryption

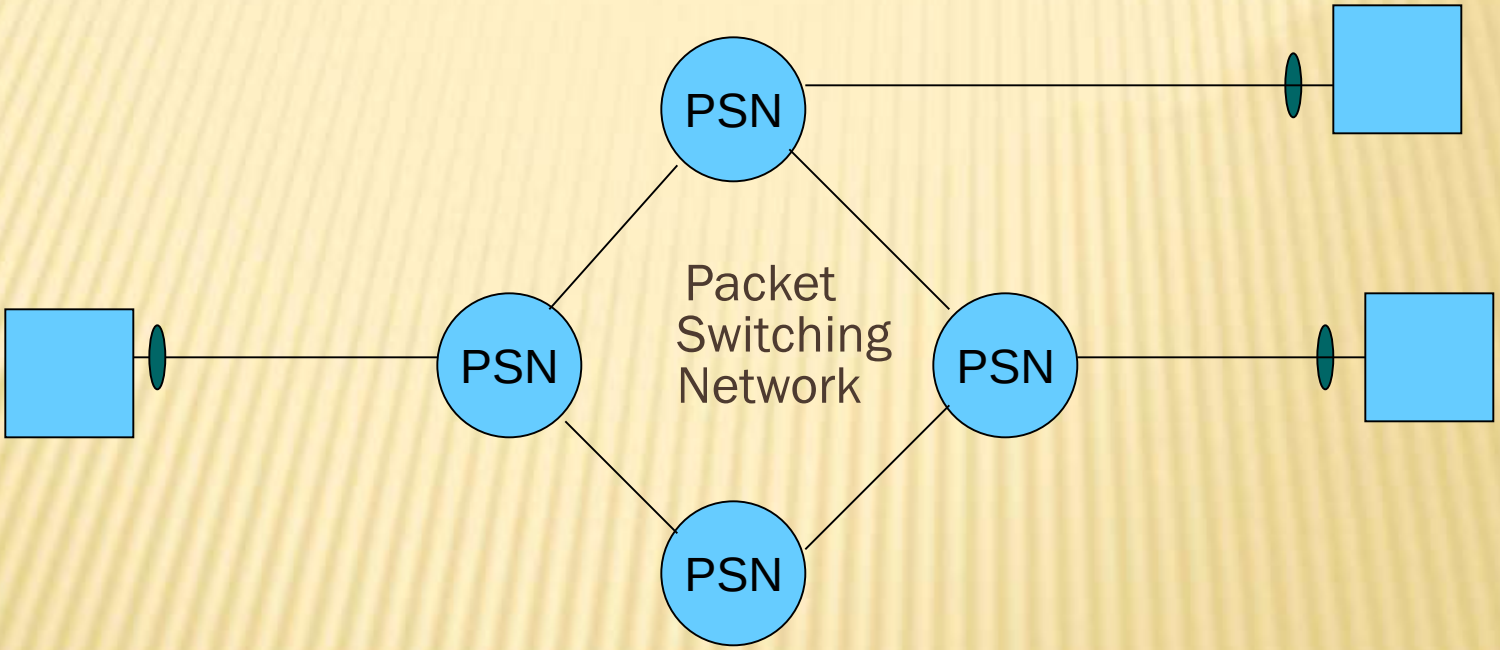


- Encryption device

End-to-End encryption:

- With end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data.
- The data, in encrypted form, are then transmitted unaltered across the network to the destination terminal or host.
- The Destination shares a key with the source and so is able to decrypt the data.
- This approach would seem to secure the transmission against attacks on the network links or switches.

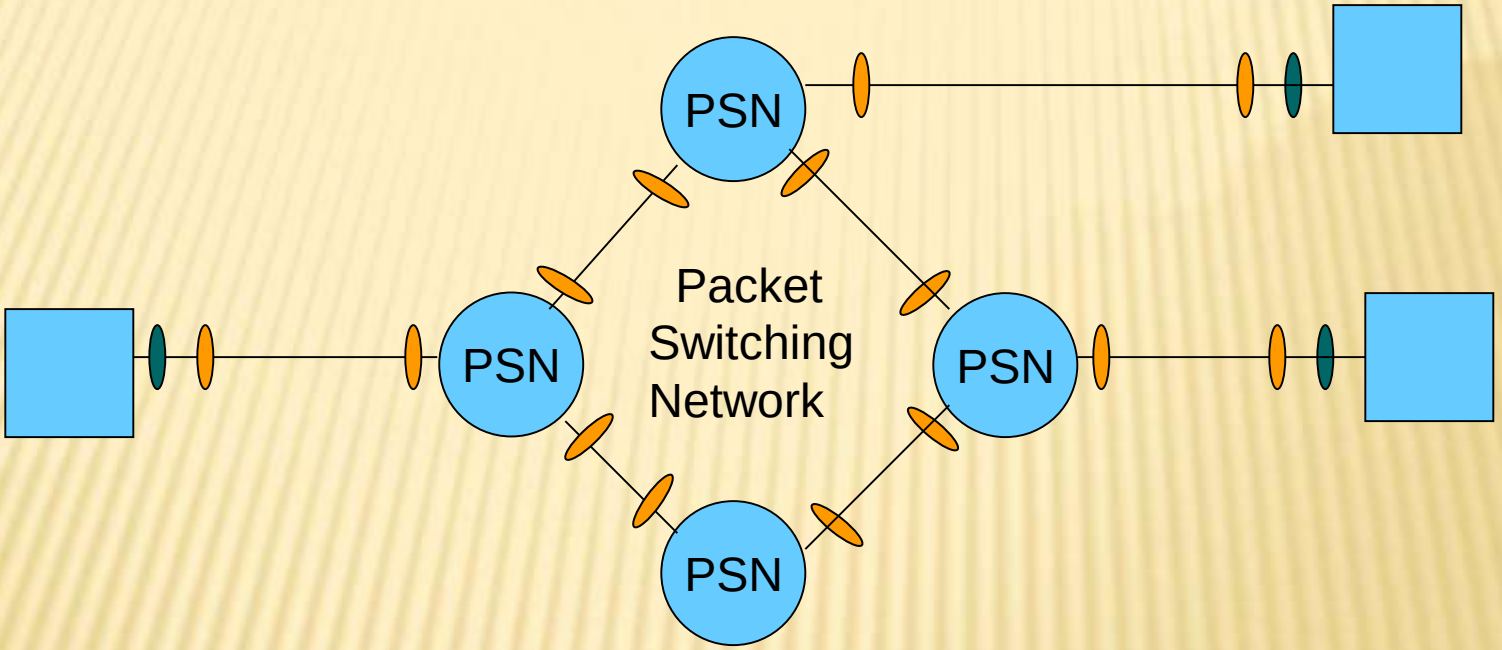
End-to-End Encryption



- Encryption device

- What part of each packet will the host encrypt?
- Suppose that the host encrypts the entire packet, including the header. This will not work because, remember, only the other host can perform the decryptions.
- The packet-switching node will receive an encrypted packet and be unable to read the header.
- *Therefore, it will not be able to route the packet.* It follows that the host may only encrypt the user data portion of the packet and must leave the header in the clear, so that it can be read by the network.
- Thus, with end-to-end encryption, the user data are secure. However, the traffic Pattern is not, because packet headers are transmitted in the clear.
- *To achieve greater security, both link and end-to-end encryptions are needed.*

Combined *link and end-to-end encryptions*



Note: Ideally having both at once

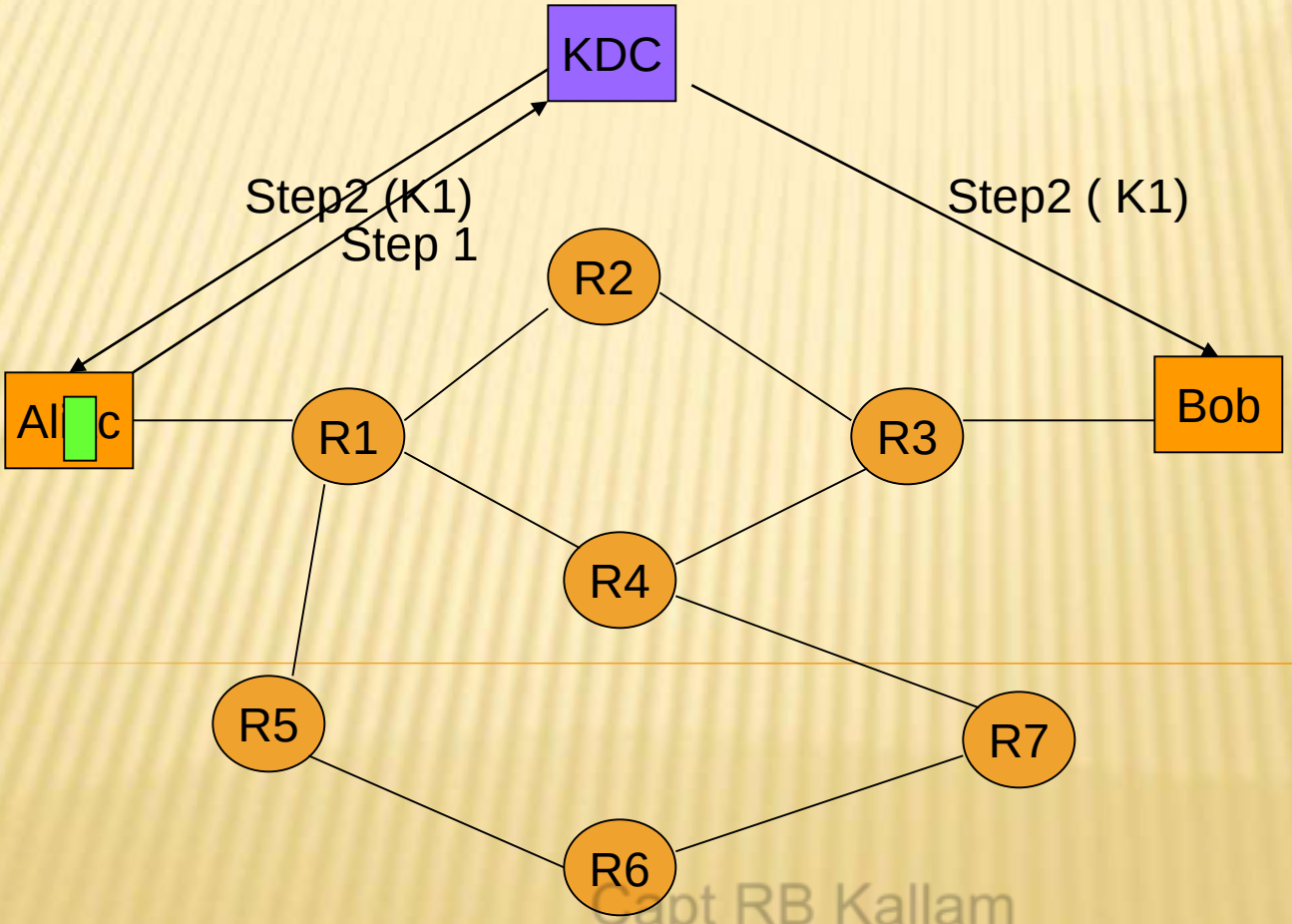
end-to-end protects data contents over entire path and provides authentication

link protects traffic flows from monitoring

Key Distribution

- symmetric schemes require both parties to share a common secret key
- issue is how to securely distribute this key
- Given parties A and B have various **key distribution** alternatives:
  1. A can select key and physically deliver to B
  2. third party can select & physically deliver key to A & B
  3. if A & B have communicated previously can use previous recent key to encrypt a new key
  4. if A & B have secure communications with a third party C, C can relay key between A & B

Demonstration of the key distribution and Secured transformation:



Key Hierarchy

- typically have a hierarchy of keys
- session key
  - temporary key
  - used for encryption of data between users
  - for one logical session then discarded
- master key
  - used to encrypt session keys
  - shared by user & key distribution center

Key Distribution Scenario

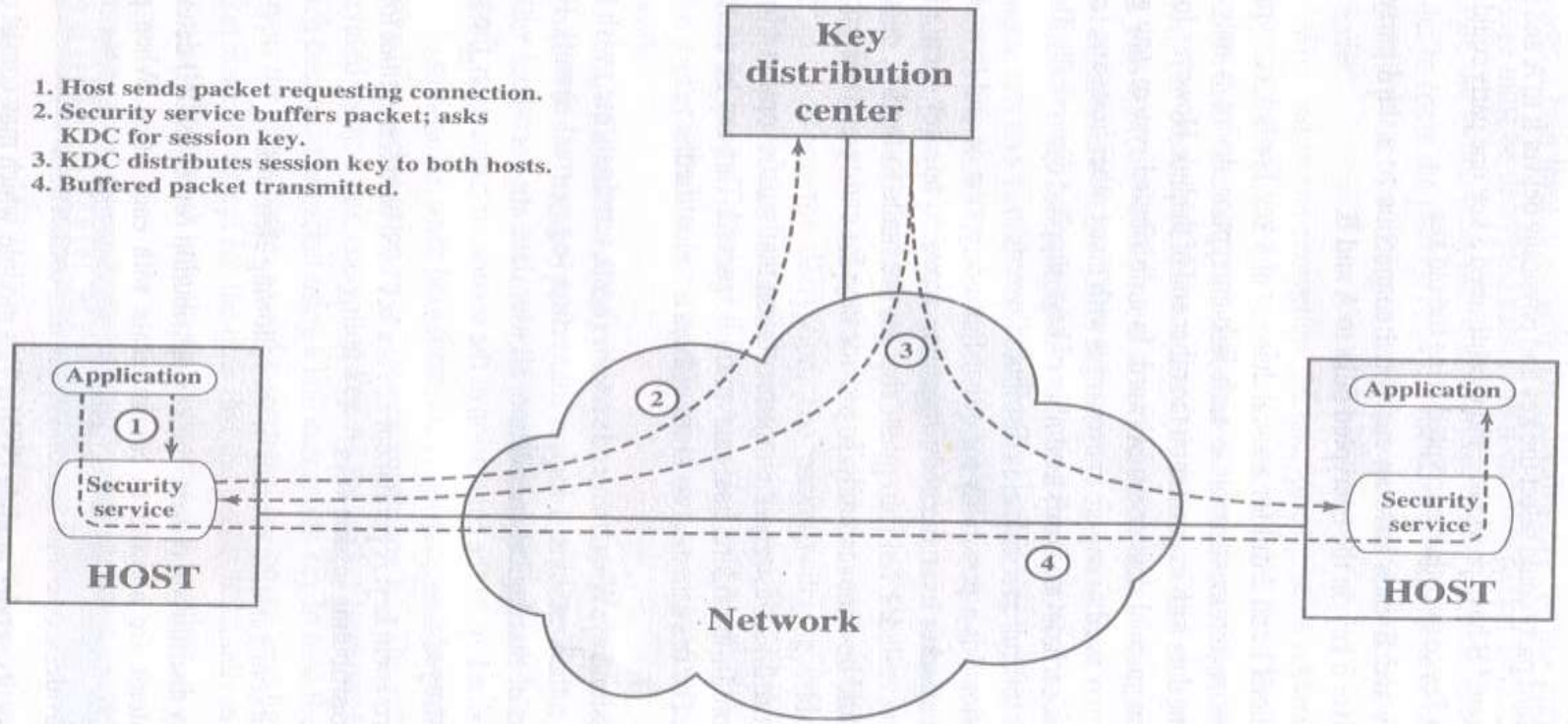
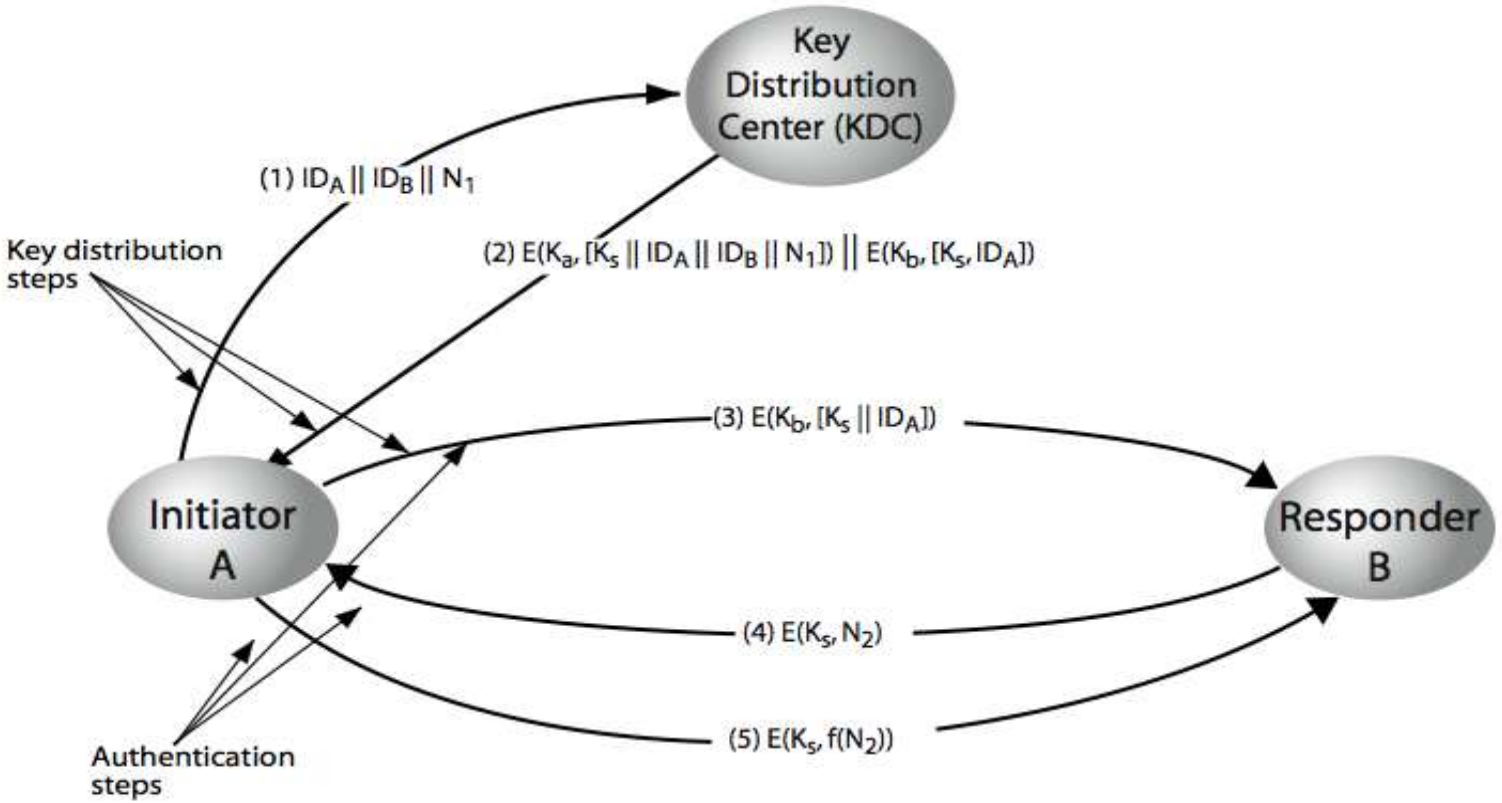


Figure 2.12 Automatic Key Distribution for Connection-Oriented Protocol

Key Distribution Scenario



Key Distribution Scenario

1. A issues a request to the KDC for a session key to protect a logical connection to B.

   The message includes ($ID_A//ID_B//N_1$)the identity of A and B and a unique identifier N_1 , for this transaction, which we refer to as a nonce or a random number.

2. The KDC responds with a message encrypted using K_a . The message includes the following:

   $E(Ka [K_s// ID_A//ID_B//N_1]) // E(K_{b.} [Ks//D_A])$

   1. Intended for A:
      - The one time session key, K_s , to be used for this session
      - The original request message, including the nonce, to enable A to match this response with the appropriate request.
   2. Intended for B:
      - The one-time session key, K_s to be used for this session
      - An identifier of A, ID_A

   The last two items are encrypted with $K_{b.}$ They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the up coming session and forwards to B the information that originated at the KDC for B, namely, $E(K_{b.} [Ks // ID_A])$

4. Using the newly mentioned session key for encryption, B sends a nonce, N_2 to A.

5. Also using *Ks*, A responds with $f(N_2)$,where f is a function that performs some transformation on N_2

Key Distribution Issues

- hierarchies of KDC's required for large networks, but must trust each other
- session key lifetimes should be limited for greater security
- use of automatic key distribution on behalf of users, but must trust system (Ref:216 v4 tb)
- use of decentralized key distribution(Ref:217 v4 tb)
- controlling key usage

End Of The 2nd Unit Part-1