Chapter 20

Symmetric Encryption and Message Confidentiality

Symmetric Encryption

- Also referred to as:
  - Conventional encryption
  - Secret-key or single-key encryption
- Only alternative before public-key encryption in 1970's
  - Still most widely used alternative
- Has five ingredients:
  - Plaintext
  - Encryption algorithm
  - Secret key
  - Ciphertext
  - Decryption algorithm

Cryptography

- classified along three independent dimensions:
  - The type of operations used for transforming plaintext to ciphertext
    - Substitution – each element in the plaintext is mapped into another element
    - Transposition – elements in plaintext are rearranged
  - The number of keys used
    - Sender and receiver use same key – symmetric
    - Sender and receiver each use a different key - asymmetric
  - The way in which the plaintext is processed
    - Block cipher – processes input one block of elements at a time
    - Stream cipher – processes the input elements continuously

Substitution Ciphers

- Replaces one symbol with another. If the symbols in the plaintext are alphabetic characters, we replace one character with another.

A substitution cipher replaces one symbol with another.

The simplest substitution cipher is a shift cipher (additive cipher)

Substitution Cipher

Example Shift Cipher

- Use the additive cipher with key = 15 to encrypt the message "hello".

Solution

- We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h	→	Shift 15 characters down	→	Ciphertext: w
Plaintext: e	→	Shift 15 characters down	→	Ciphertext: t
Plaintext: l	→	Shift 15 characters down	→	Ciphertext: a
Plaintext: l	→	Shift 15 characters down	→	Ciphertext: a
Plaintext: o	→	Shift 15 characters down	→	Ciphertext: d

- The ciphertext is therefore "wtaad".

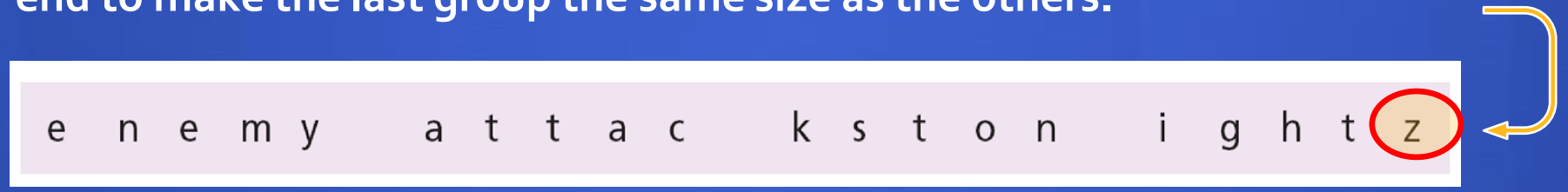
Transposition Ciphers

- A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols.
- A symbol in the 1st position of the *plaintext* may appear in the 10th position of the *ciphertext*, while a symbol in the 8th position in the *plaintext* may appear in the 1st position of the *ciphertext*.
- In other words, a transposition cipher reorders (transposes) the symbols.

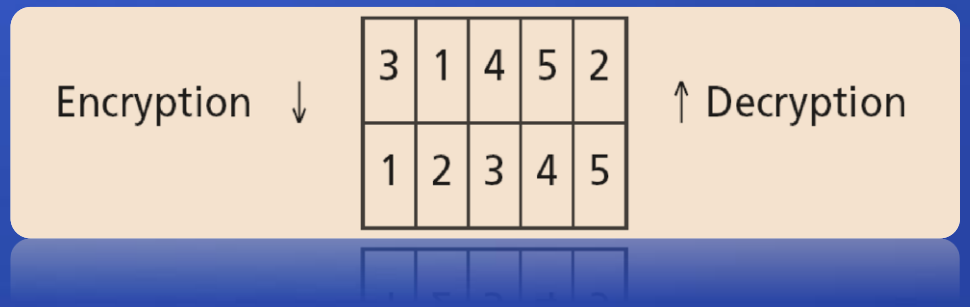
A transposition cipher reorders symbols.

Transposition Cipher

- Alice needs to send the message "*Enemy attacks tonight*" to Bob.
- Alice and Bob have agreed to divide the text into groups of five characters and then permute the characters in each group.
- The following shows the grouping after adding a bogus character (z) at the end to make the last group the same size as the others.

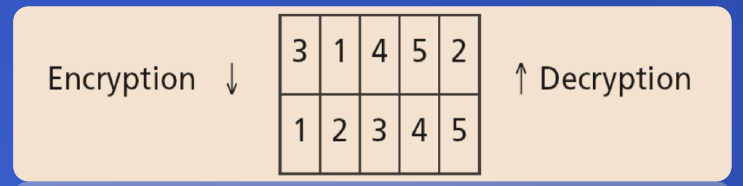


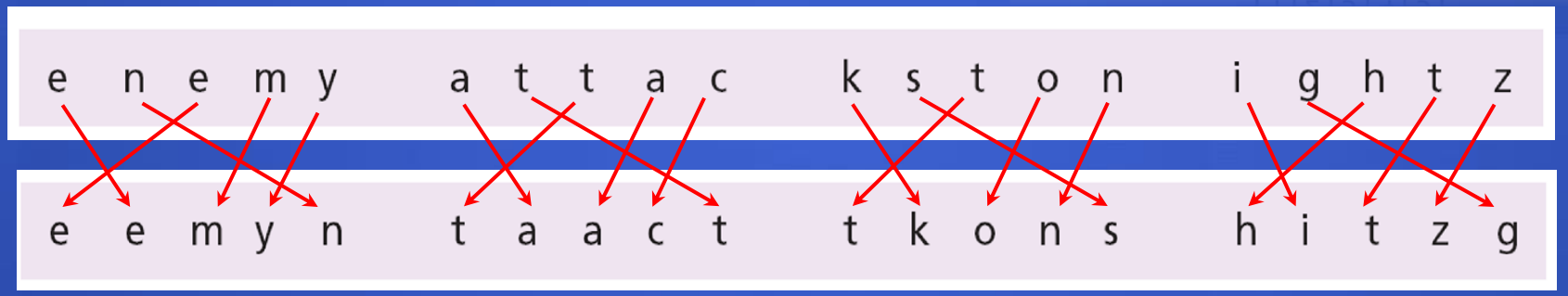
The key used for encryption and decryption is a permutation key, which shows how the character are permuted.
For this message, assume that Alice and Bob used the following key:



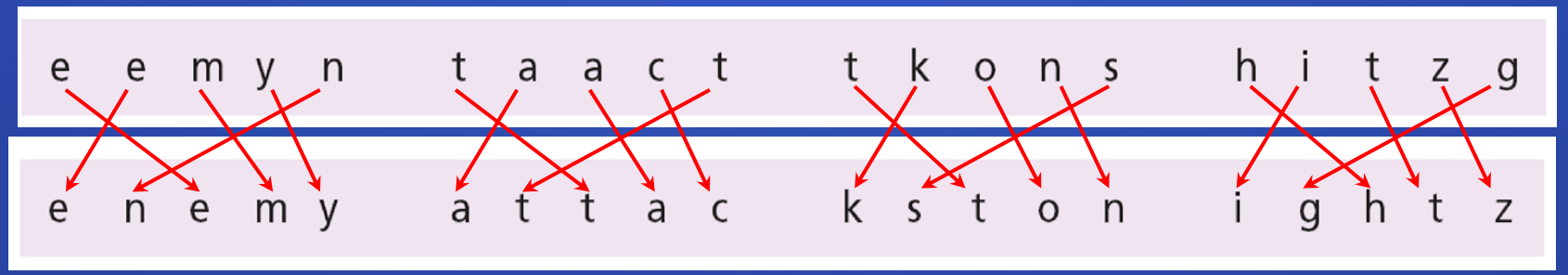
Transposition Cipher

- The 3rd character in the *plaintext* block becomes the 1st character in the *ciphertext* block. Etc….
- The permutation yields:





- Alice sends the *ciphertext* "eemyntaacttkonshitzg" to Bob.
- Bob divides the *ciphertext* into five-character groups and, using the key in the reverse order, finds the *plaintext*.



Symmetric (Secret Key) Encryption

- Since these traditional ciphers are no longer secure due to PC processing power, modern symmetric-key ciphers have been developed over the last few decades.
- Modern ciphers normally use a combination of substitution, transposition and some other complex transformations to create a *ciphertext* from a *plaintext*.
- Modern ciphers are bit-oriented (instead of character-oriented). The *plaintext*, *ciphertext* and the *key* are strings of bits.
- Some examples of modern symmetric-key ciphers are DES, AES and IDEA.

type of attack	known to cryptanalyst
Ciphertext only	•Encryption algorithm •Ciphertext to be decoded
Known plaintext	•Encryption algorithm •Ciphertext to be decoded •One or more plaintext-ciphertext pairs formed with the secret key
Chosen plaintext	•Encryption algorithm •Ciphertext to be decoded •Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen ciphertext	•Encryption algorithm •Ciphertext to be decoded •Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen text	•Encryption algorithm •Ciphertext to be decoded •Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key •Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

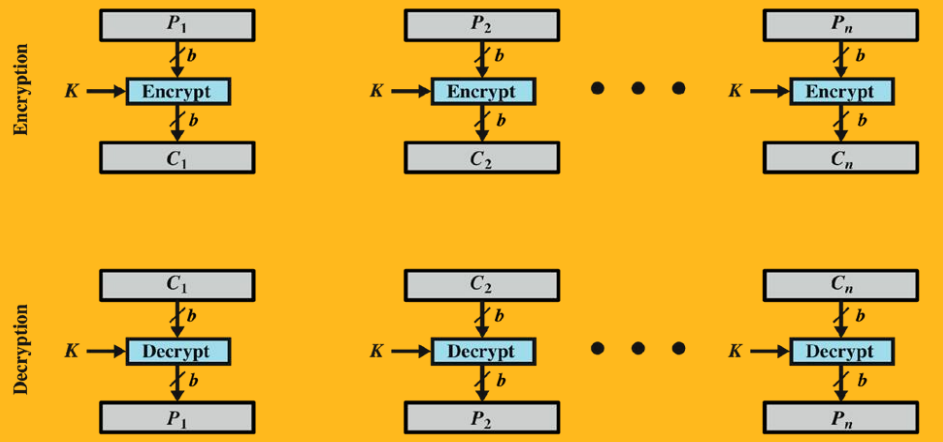
Cryptan
alysis

Computationally Secure Encryption Schemes

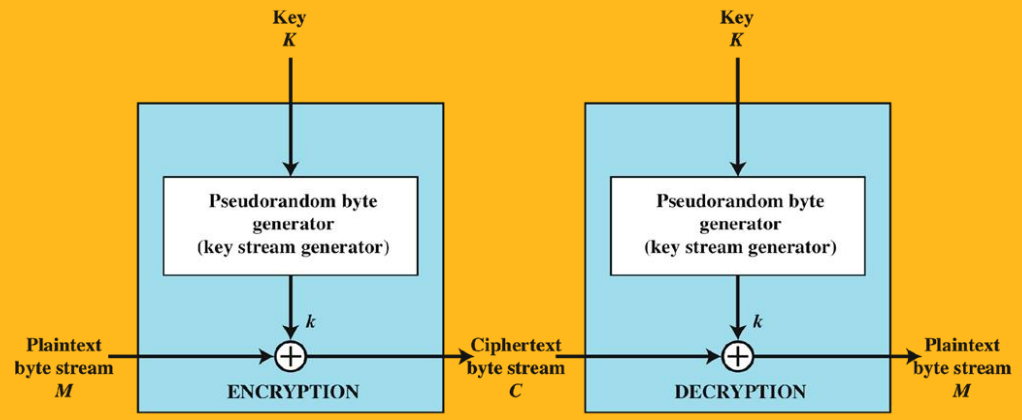
- Encryption is computationally secure if:
  - Cost of breaking cipher exceeds value of information
  - Time required to break cipher exceeds the useful lifetime of the information
- Usually very difficult to estimate the amount of effort required to break
- Can estimate time/cost of a brute-force attack

Block Cipher Encryption

Stream Encryption



(a) Block cipher encryption (electronic codebook mode)



(b) Stream encryption

Figure 2.3 Types of Symmetric Encryption

Block & Stream Ciphers

Block Cipher

- Processes the input one block of elements at a time
- Produces an output block for each input block
- Can reuse keys
- More common

Stream Cipher

- Processes the input elements continuously
- Produces output one element at a time
- Primary advantage is that they are almost always faster and use far less code
- Encrypts plaintext one byte at a time
- Pseudorandom stream is one that is unpredictable without knowledge of the input key
- Stream ciphers are vulnerable to attack if the same key is used twice or more.

Feistel Cipher Structure

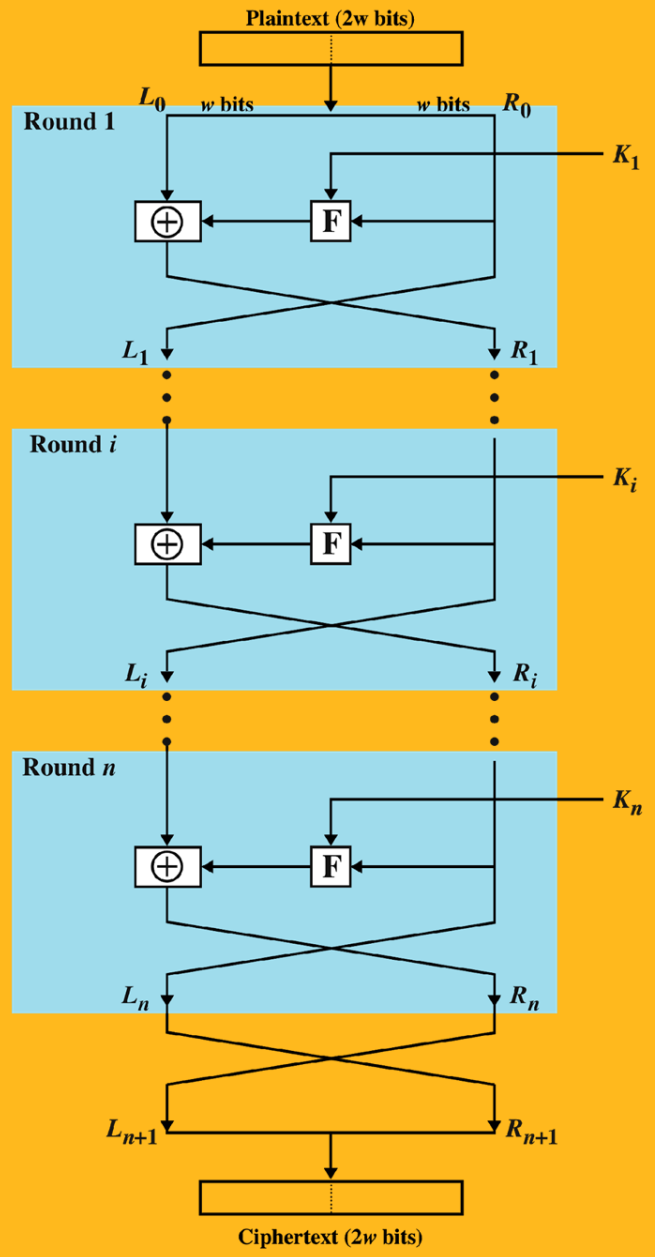
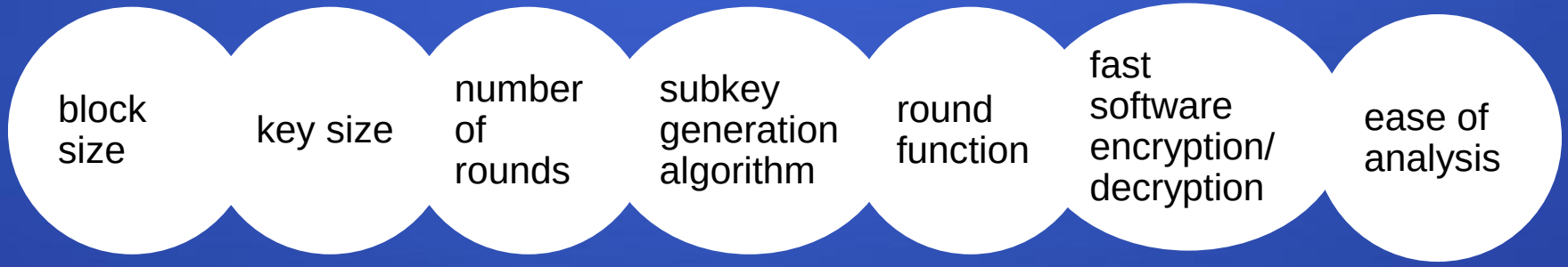


Figure 20.1 Classical Feistel Network

Block Cipher Structure

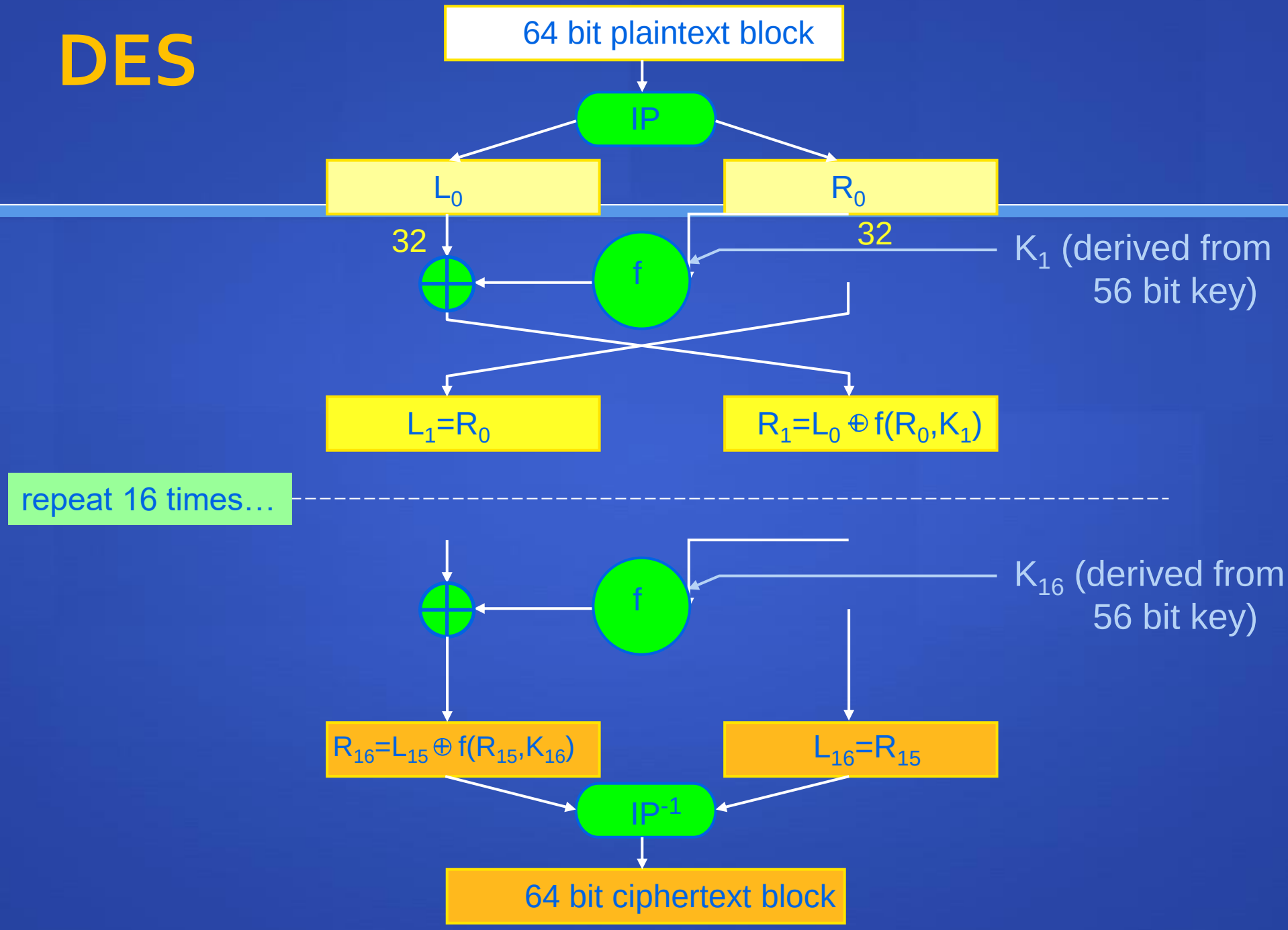
- Symmetric block cipher consists of:
  - A sequence of rounds
  - With substitutions and permutations controlled by the key
- Parameters and design features:

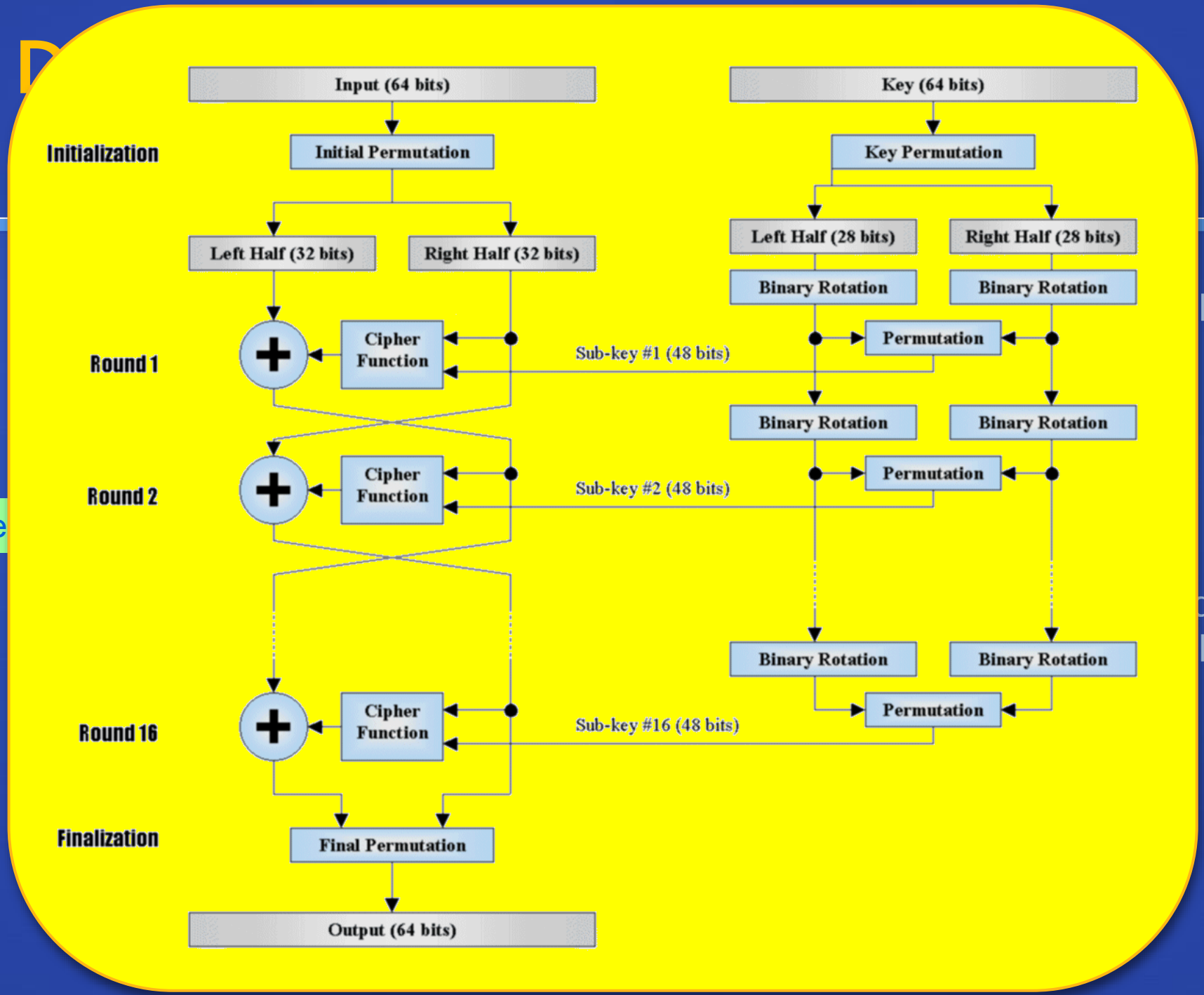


Data Encryption Standard (DES)

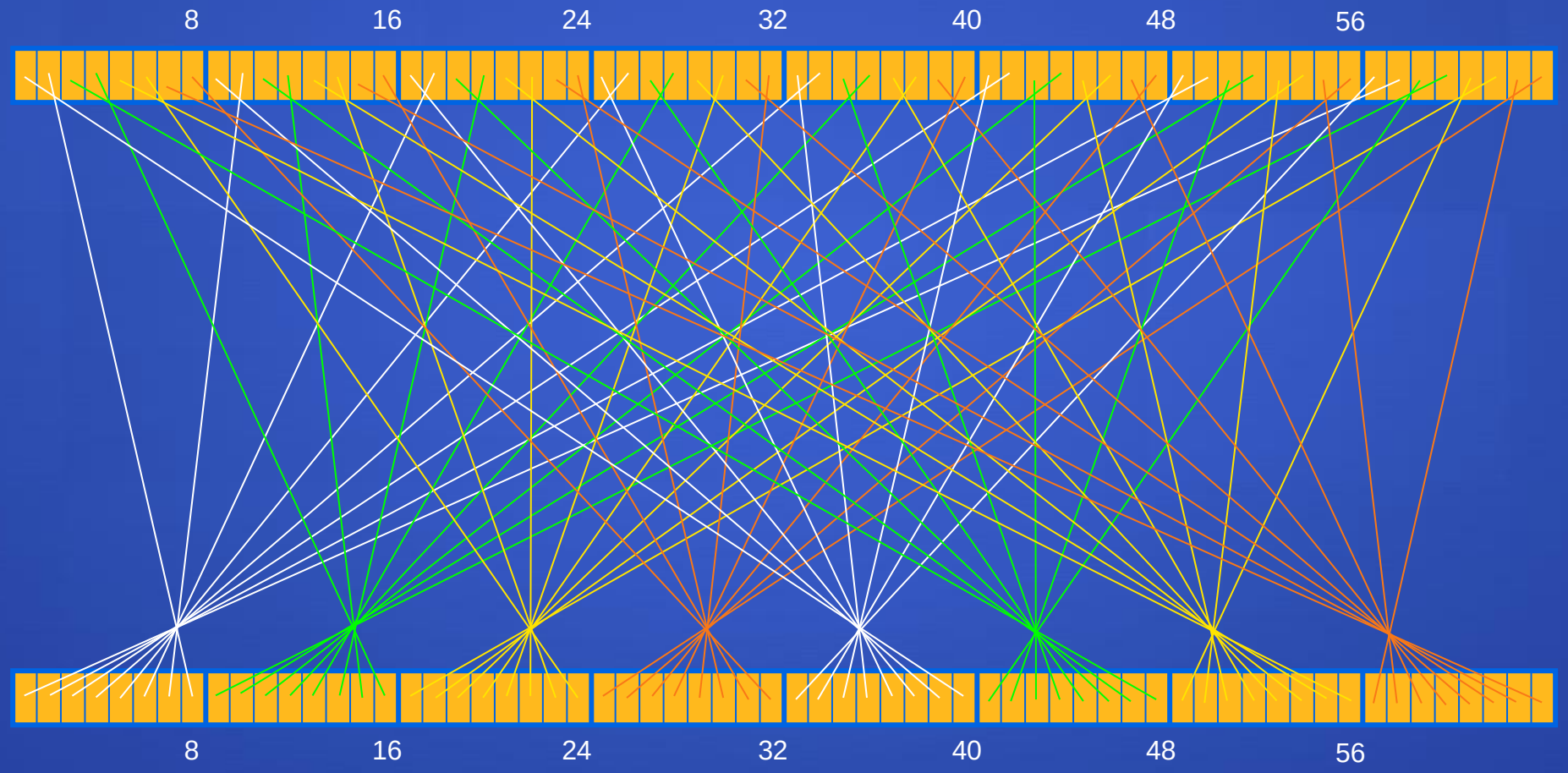
- Most widely used encryption scheme
- Adopted in 1977 by National Bureau of Standards
  - Now NIST
- FIPS PUB 46
- Algorithm is referred to as the Data Encryption Algorithm (DEA)
- Minor variation of the Feistel network

DES

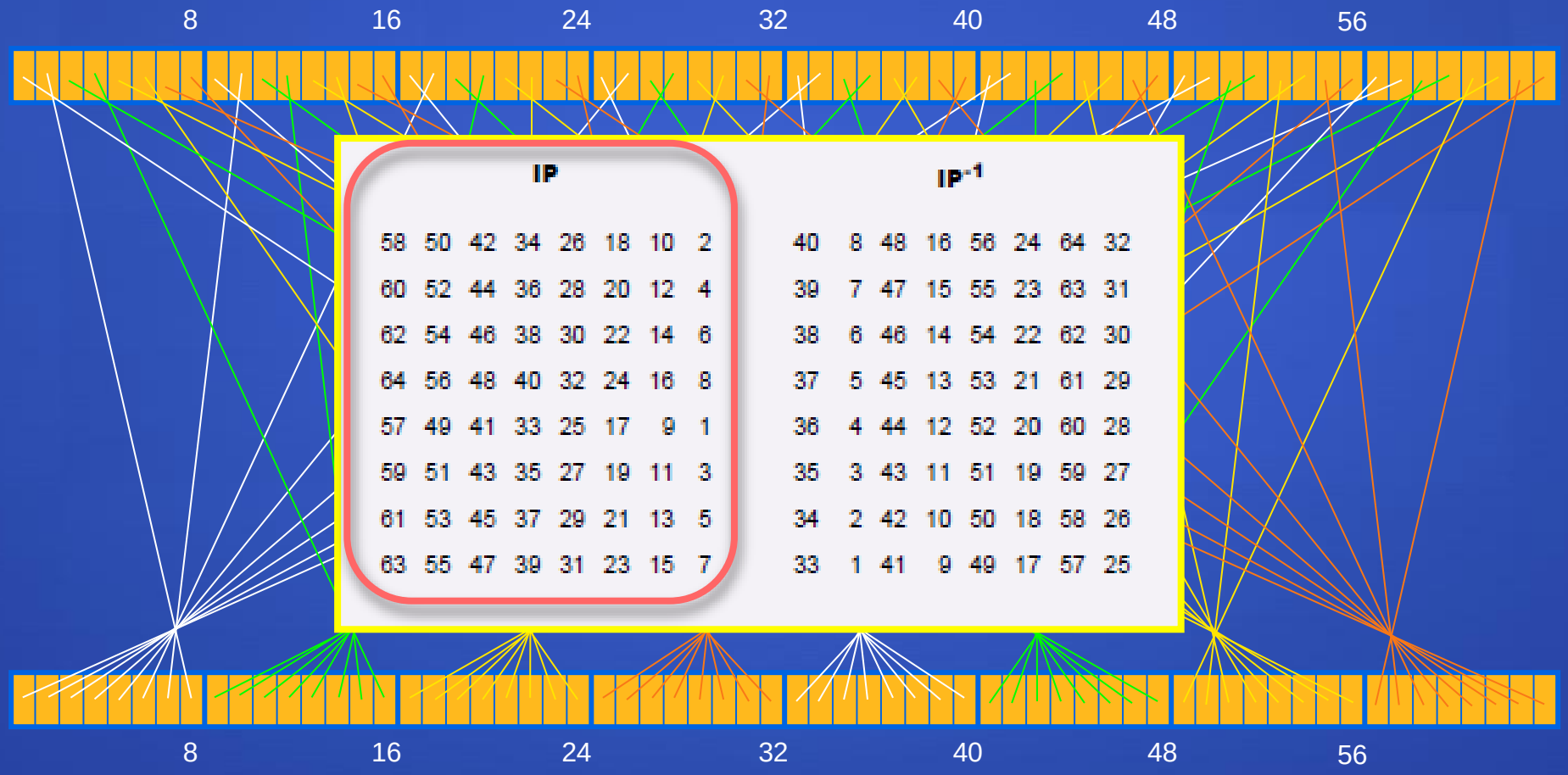

64 bit plaintext block
IP
L_0
R_0
32
32
f
K_1 (derived from 56 bit key)
$L_1=R_0$
$R_1=L_0 \oplus f(R_0,K_1)$
repeat 16 times…
f
K_{16} (derived from 56 bit key)
$R_{16}=L_{15} \oplus f(R_{15},K_{16})$
$L_{16}=R_{15}$
IP^{-1}
64 bit ciphertext block

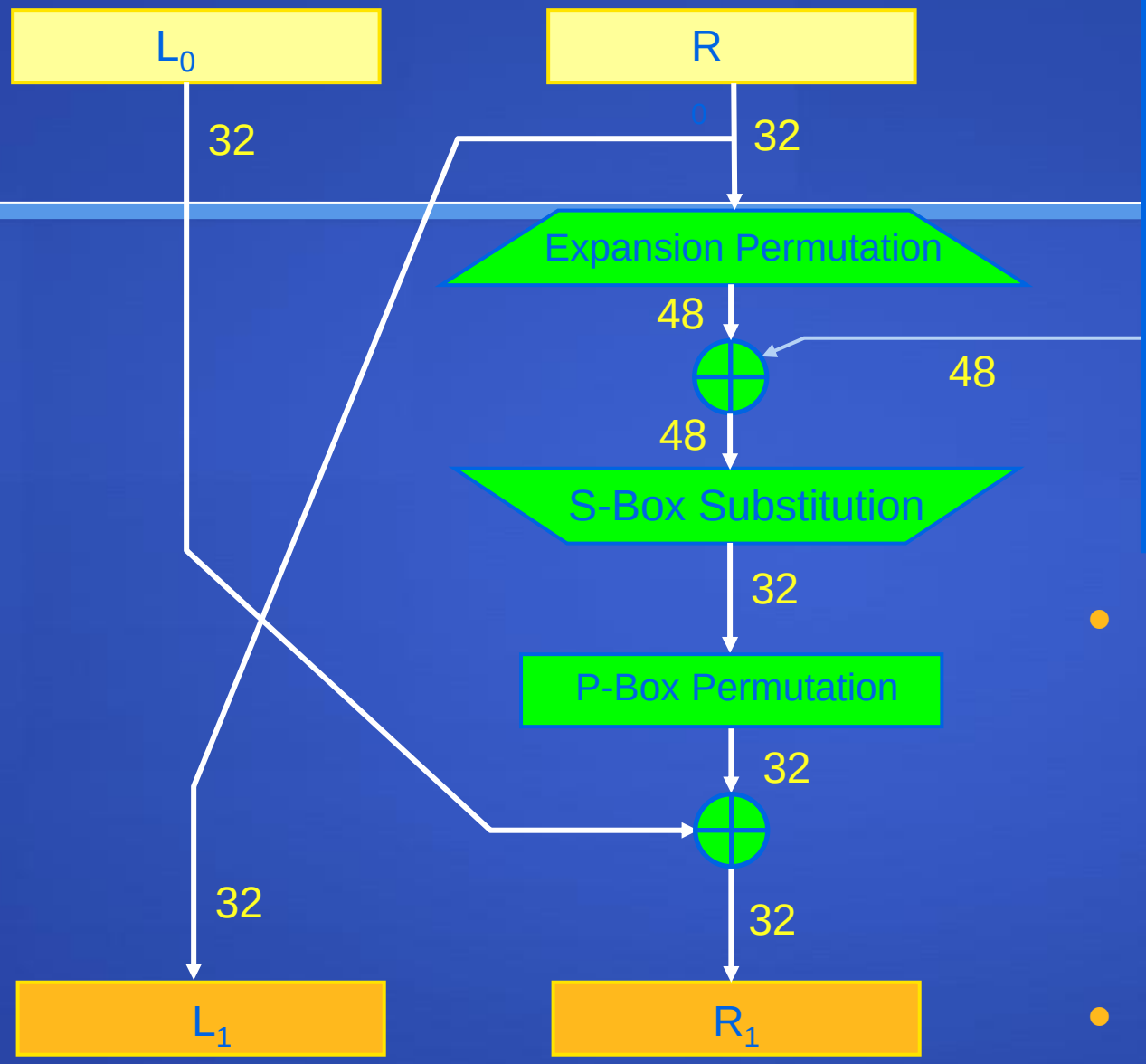

Input (64 bits)
Key (64 bits)
Initialization
Initial Permutation
Key Permutation
Left Half (32 bits)
Right Half (32 bits)
Left Half (28 bits)
Right Half (28 bits)
Binary Rotation
Binary Rotation
Round 1
Cipher Function
Permutation
Sub-key #1 (48 bits)
Binary Rotation
Binary Rotation
Round 2
Cipher Function
Permutation
Sub-key #2 (48 bits)
Binary Rotation
Binary Rotation
Round 16
Cipher Function
Permutation
Sub-key #16 (48 bits)
Finalization
Final Permutation
Output (64 bits)

IP (Initial Permutation)



IP (Initial Permutation)





48 bit subkey Generator
$K_{48} = g(i,K_{56})$

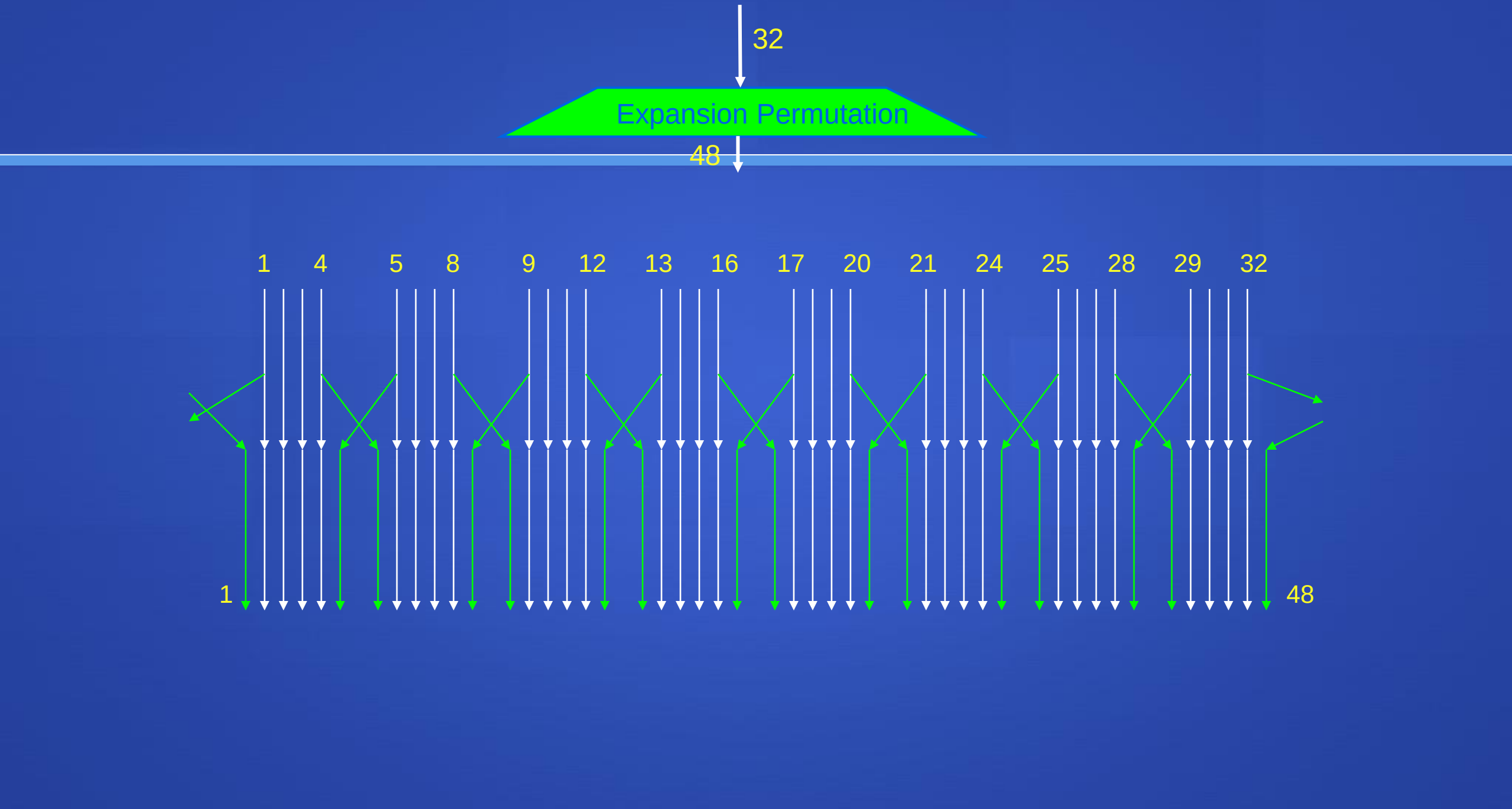
(The key for each round is deterministically found from the input 56 bit key).

- **This can be described functionally as**

$$L(i) = R(i-1)$$

$$R(i) = L(i-1) \oplus P(S(E(R(i-1)) \oplus K(i)))$$

- **This forms one round in an S-P network**


32
Expansion Permutation
48
1 4 5 8 9 12 13 16 17 20 21 24 25 28 29 32
1
48

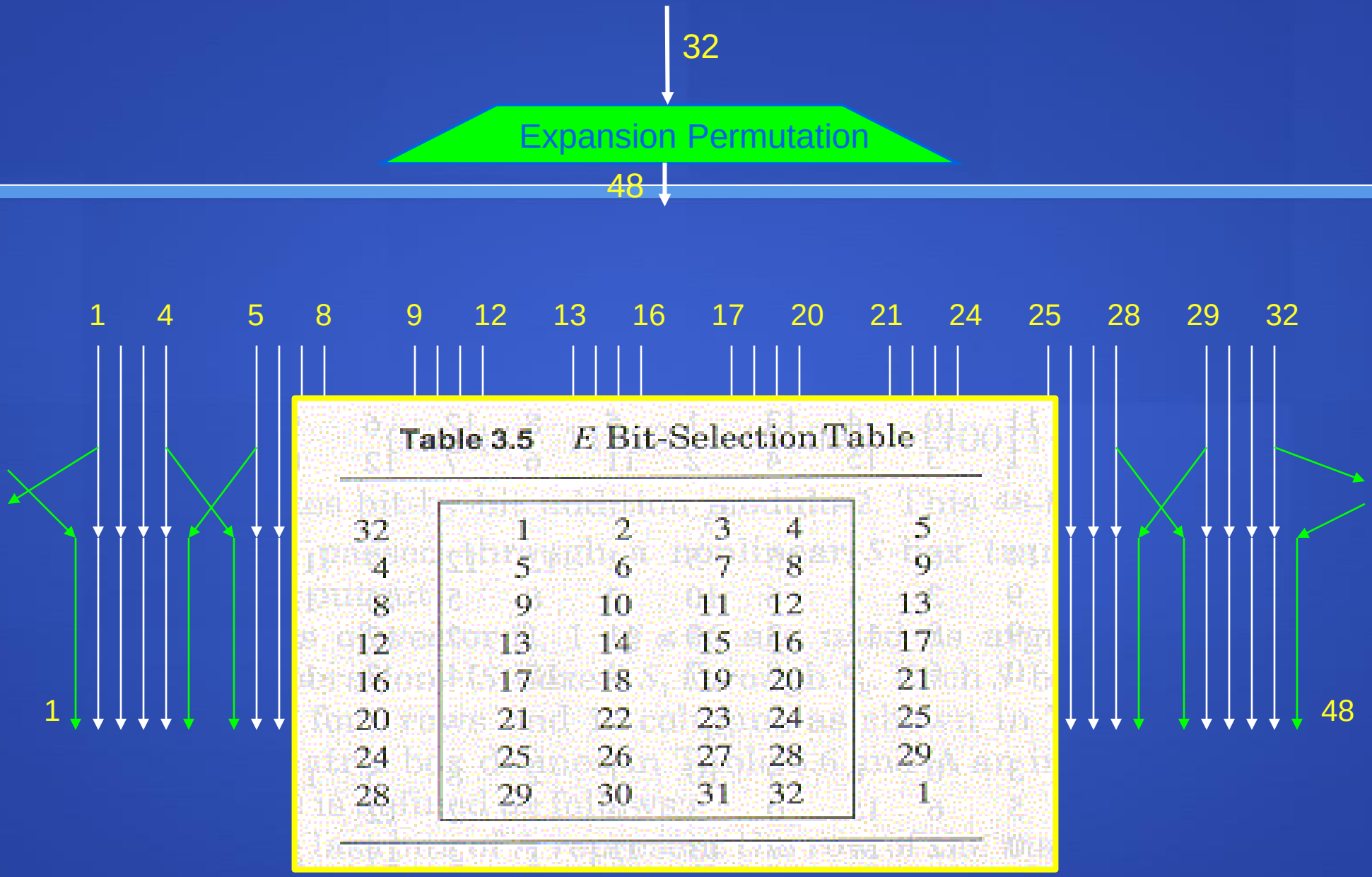
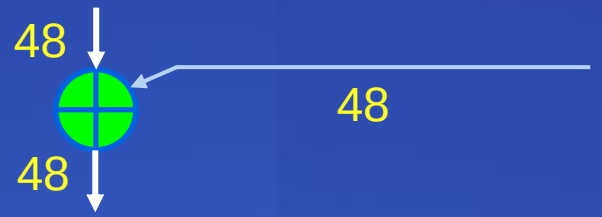
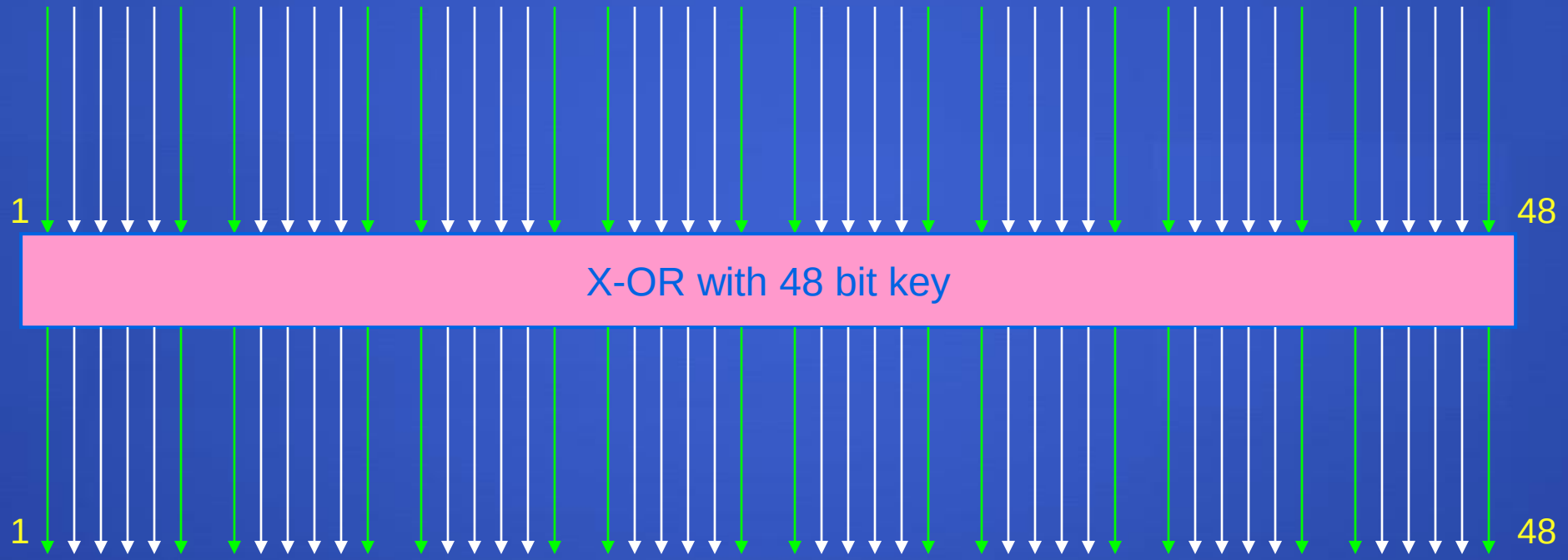
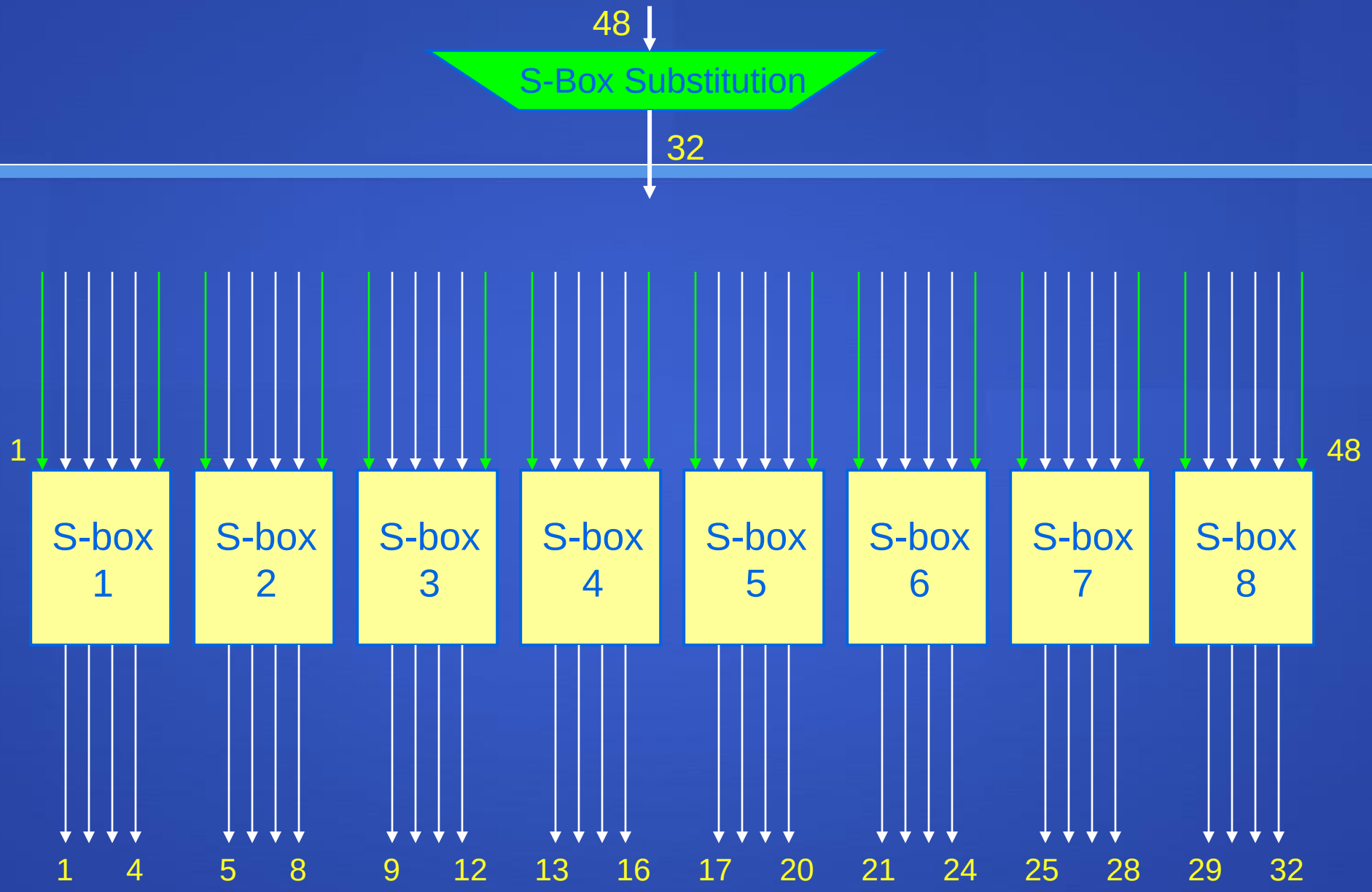


Table 3.5 *E* Bit-Selection Table

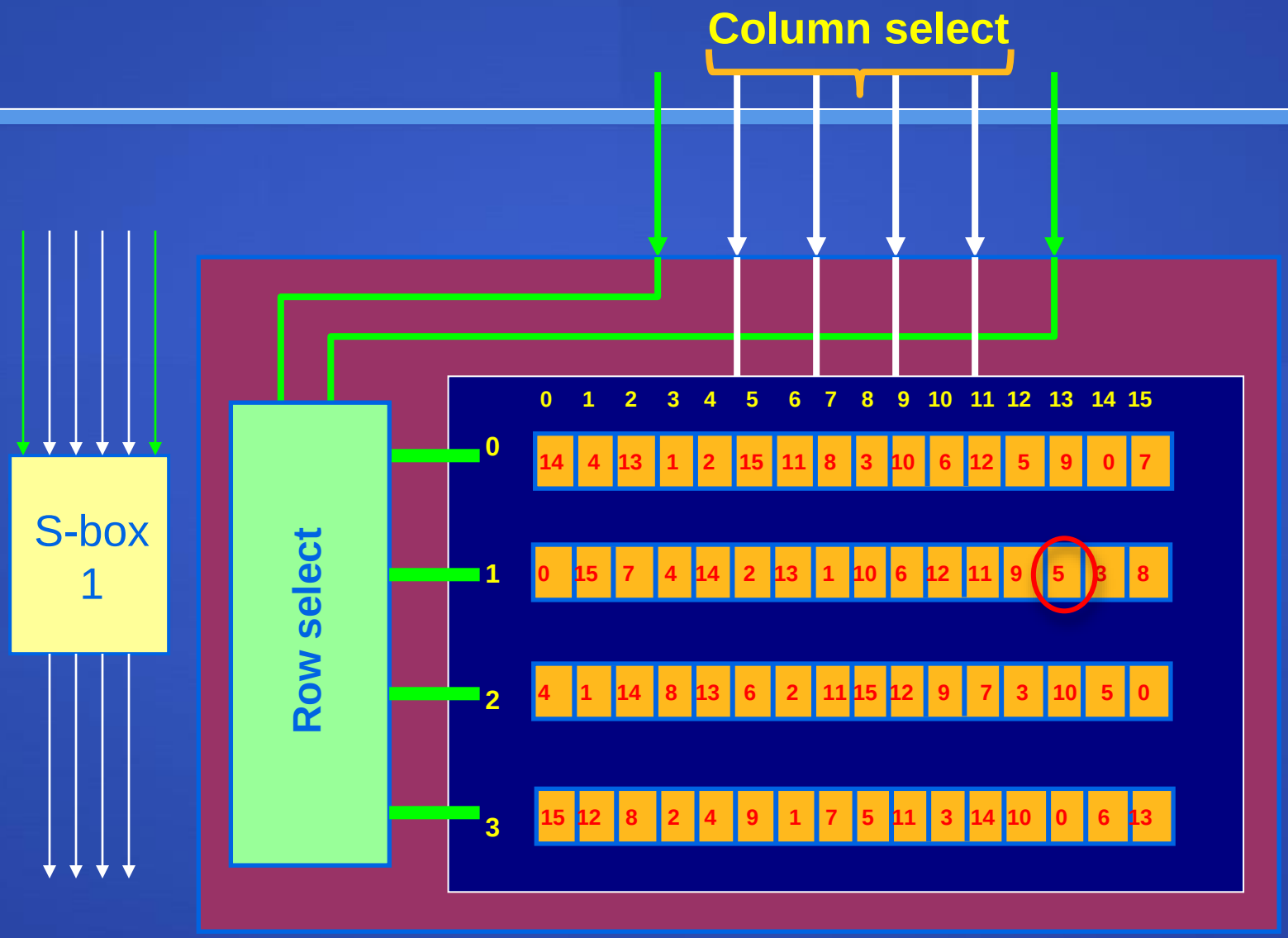
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1


48
48
48


1
48
X-OR with 48 bit key
1
48


48
S-Box Substitution
32
1
48
S-box 1
S-box 2
S-box 3
S-box 4
S-box 5
S-box 6
S-box 7
S-box 8
1 4
5 8
9 12
13 16
17 20
21 24
25 28
29 32

How an S-Box works



How an S-Box works

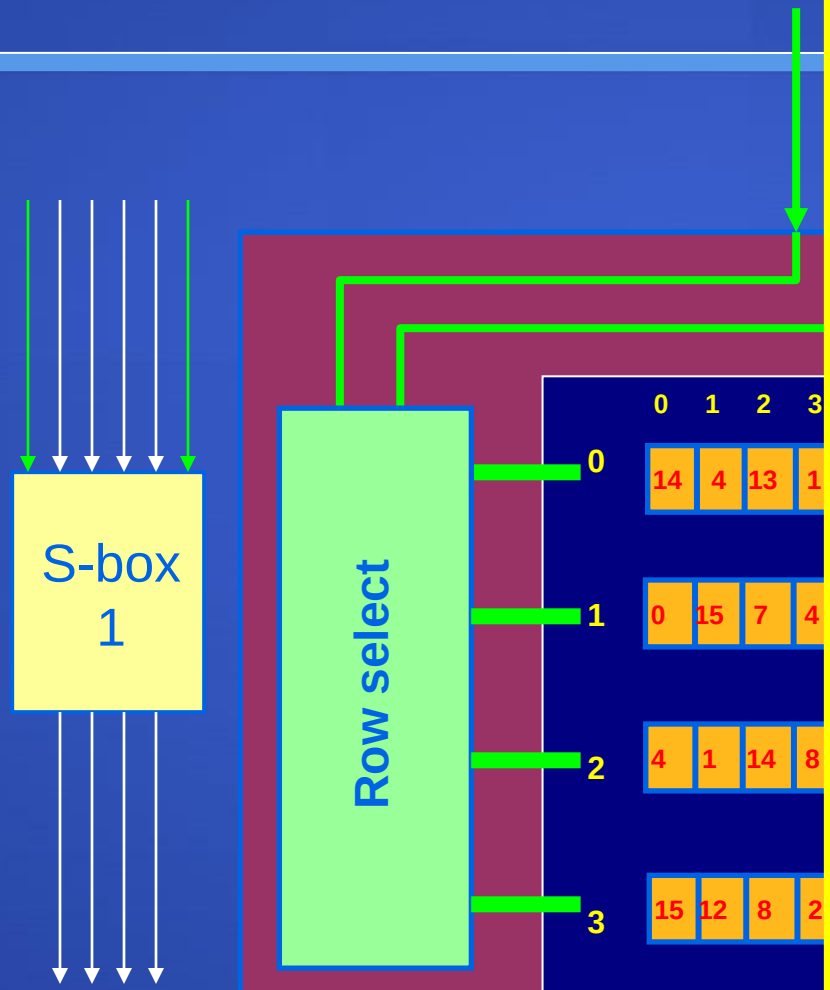


Table 3.6 Primitive S-Box Functions

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

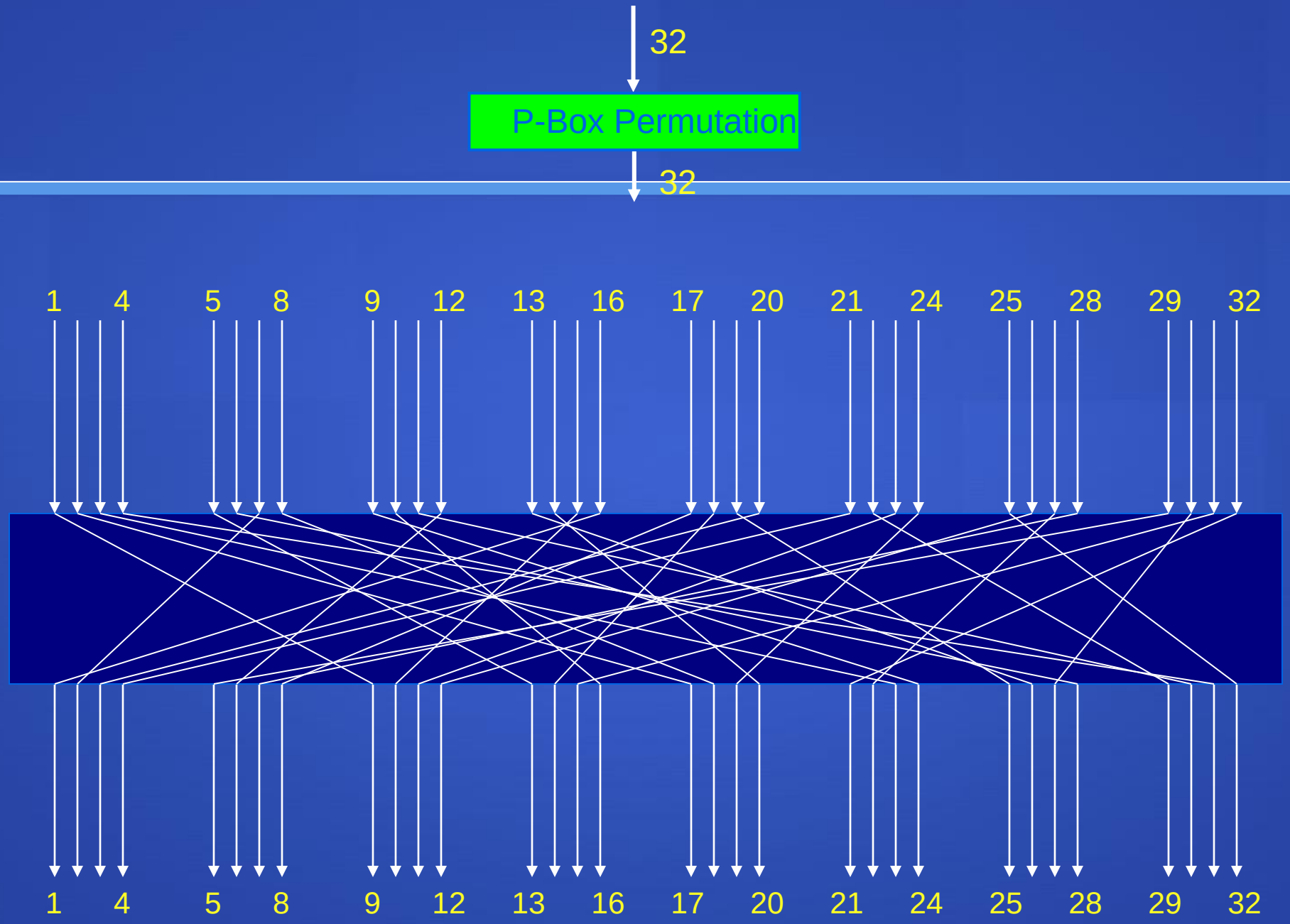
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

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


32
P-Box Permutation
32
1 4 5 8 9 12 13 16 17 20 21 24 25 28 29 32
1 4 5 8 9 12 13 16 17 20 21 24 25 28 29 32

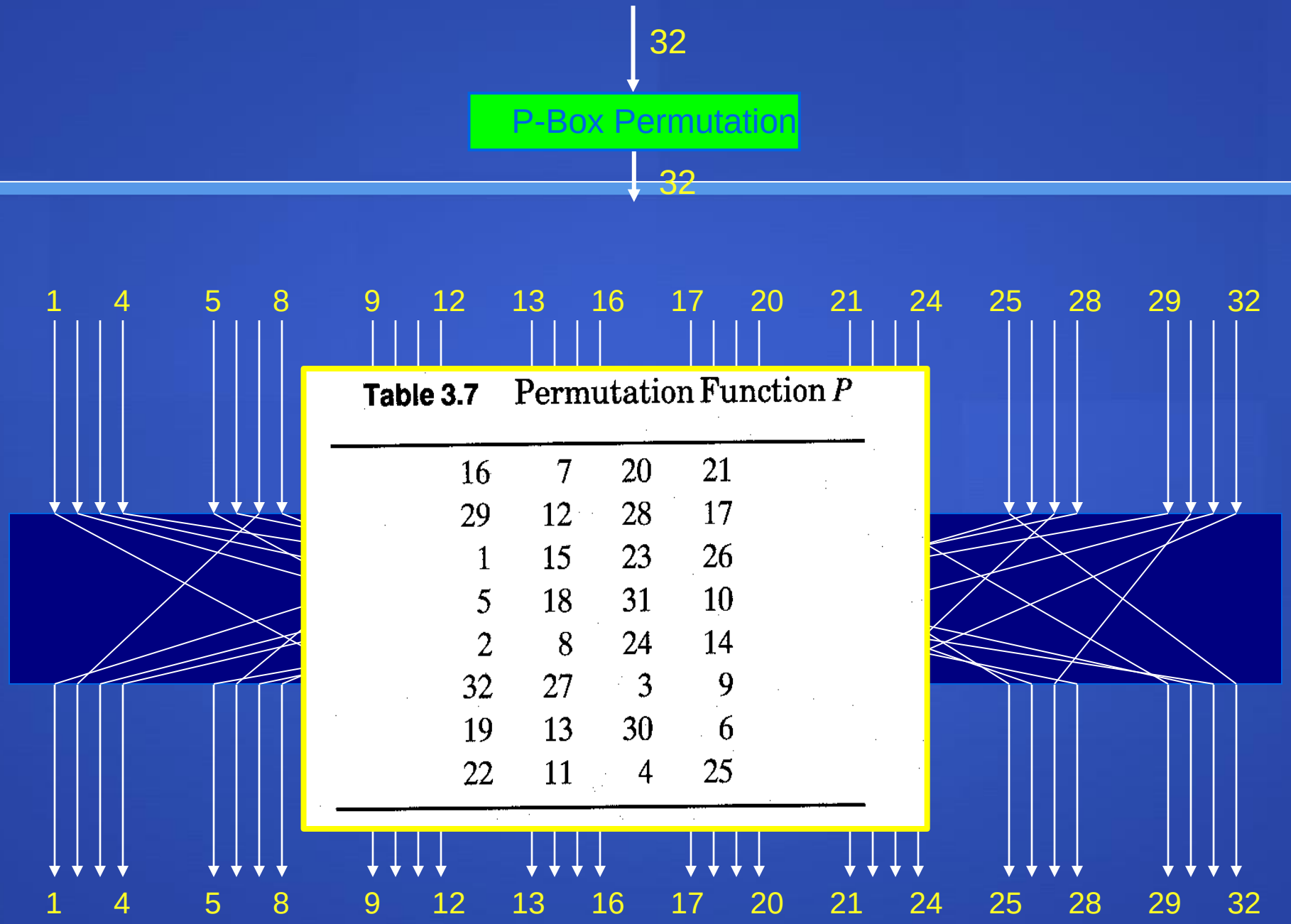
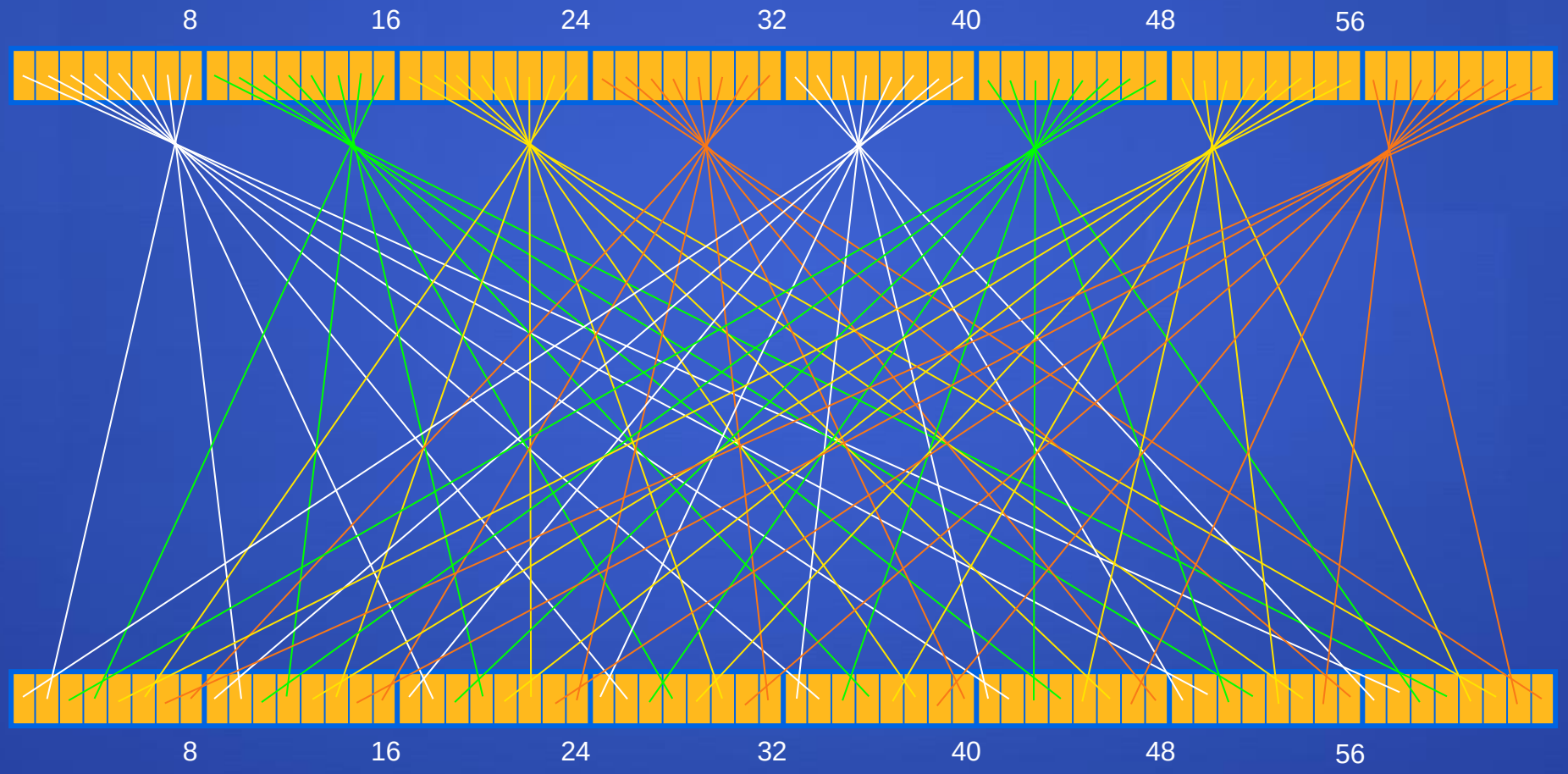


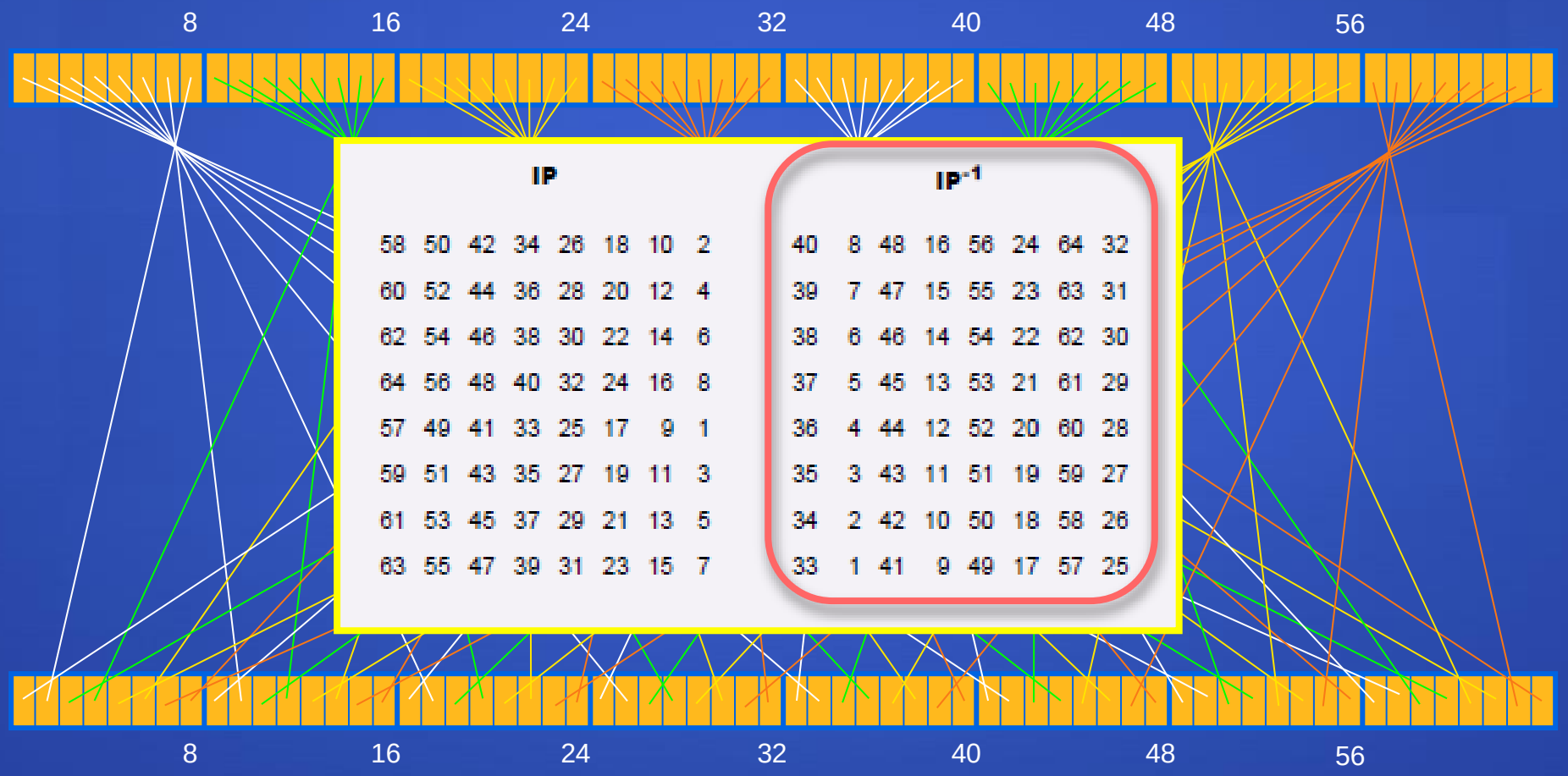
Table 3.7 Permutation Function P

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

IP^{-1} (Final Permutation):

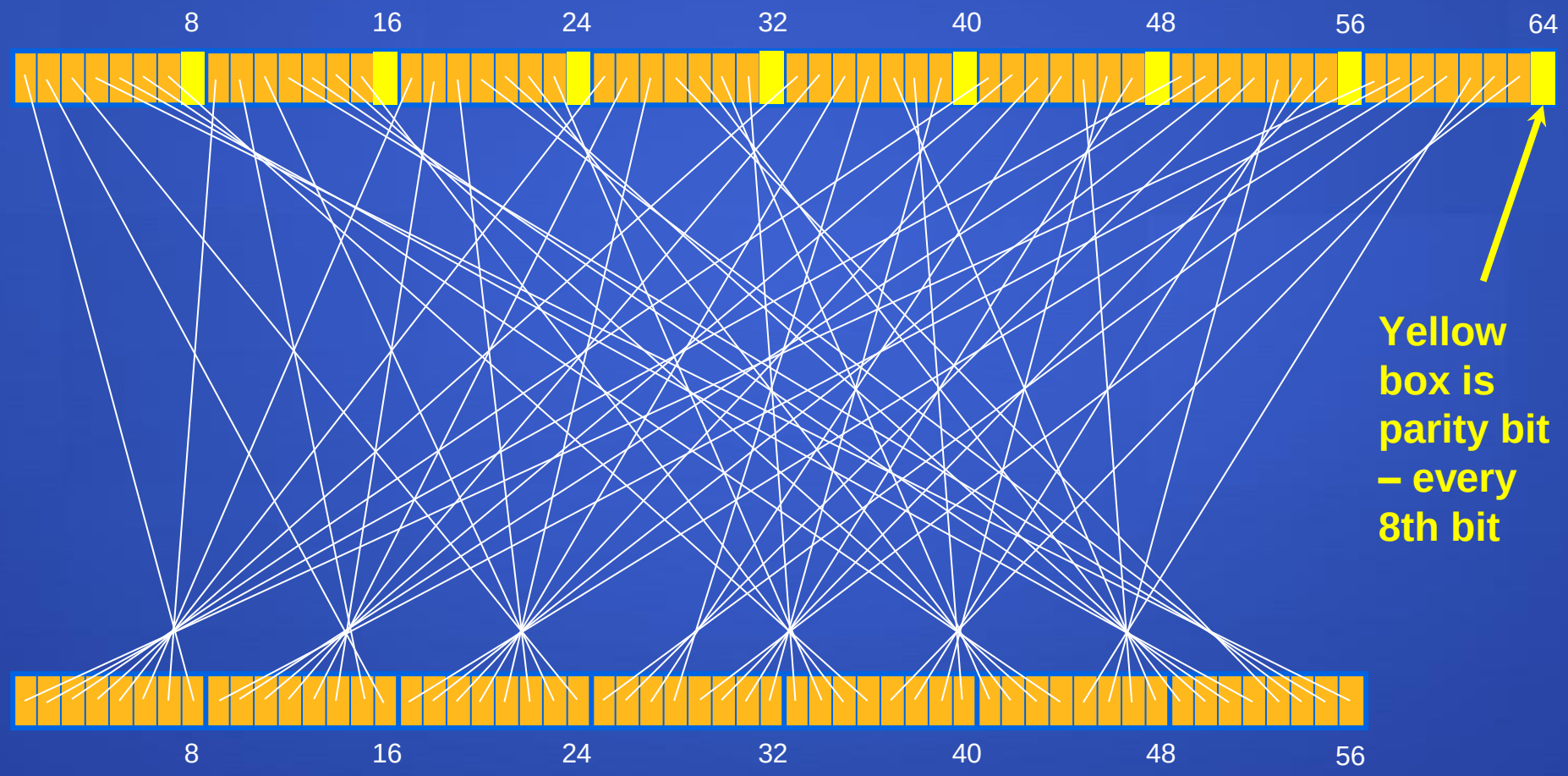


IP^{-1} (Final Permutation):



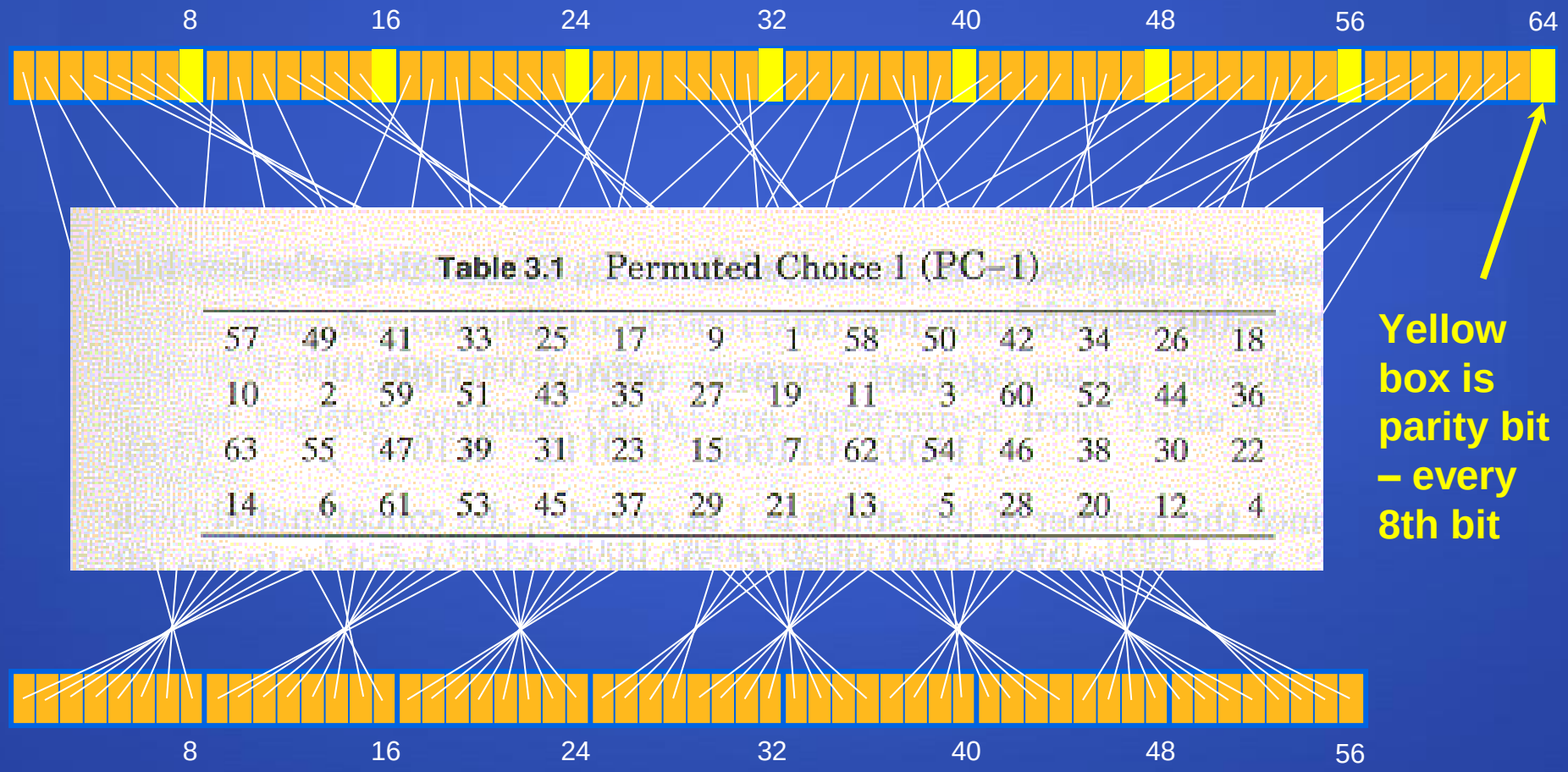
Generate Sub Keys

Initial Key Permutation



Generate Sub Keys

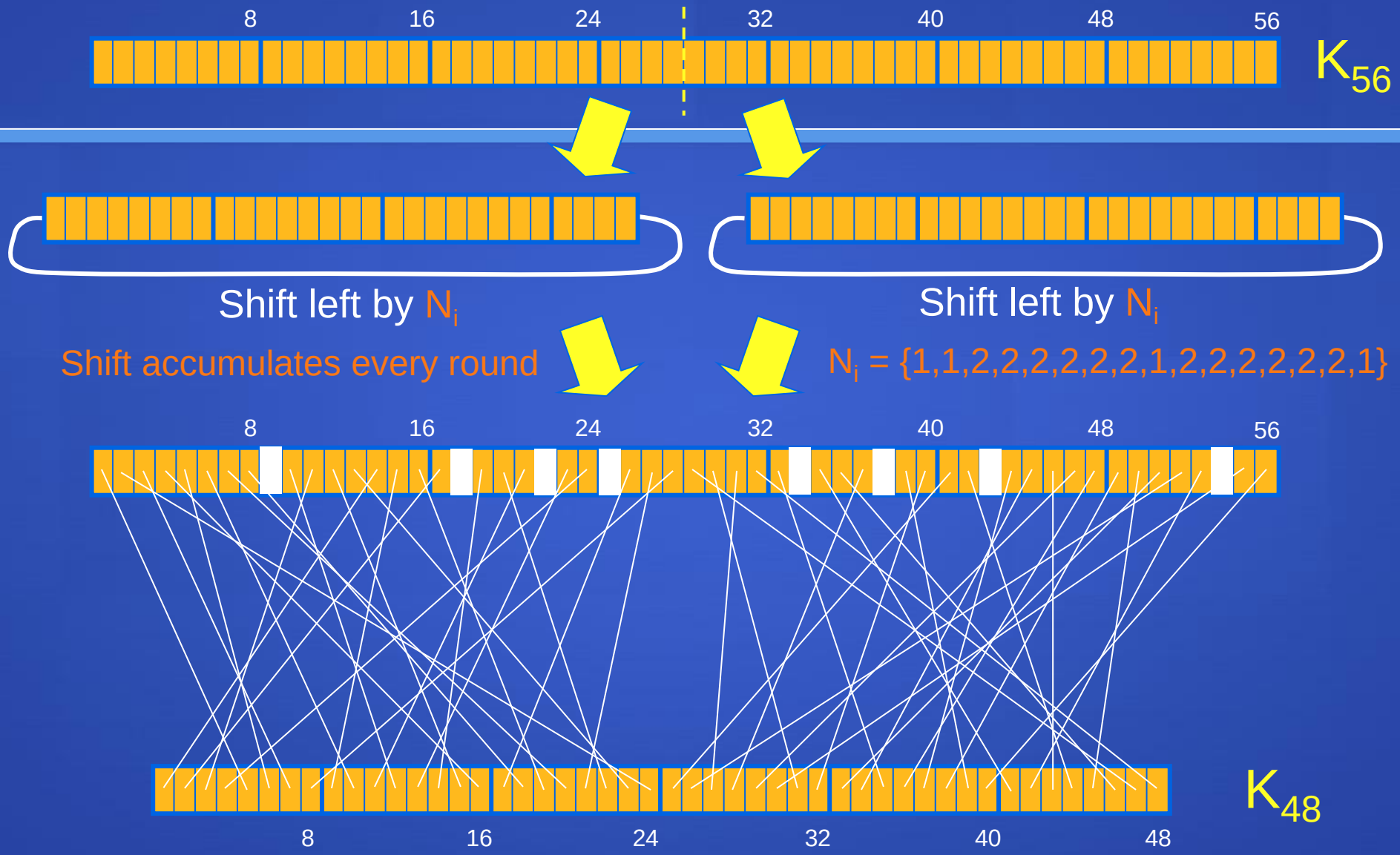
Initial Key Permutation



Yellow box is parity bit – every 8th bit

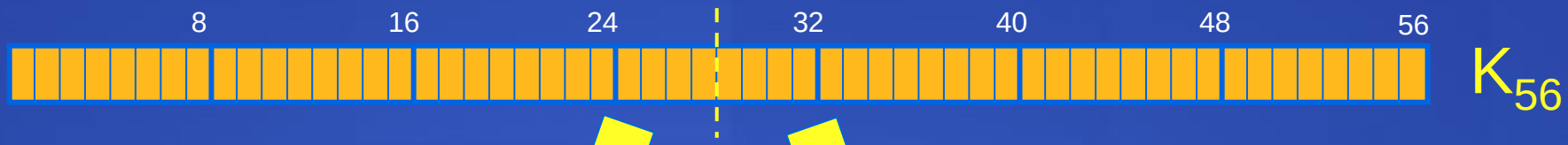
Generate Sub Keys

Key Split & Shift & Compress



Generate Sub Keys

Key Split & Shift & Compress

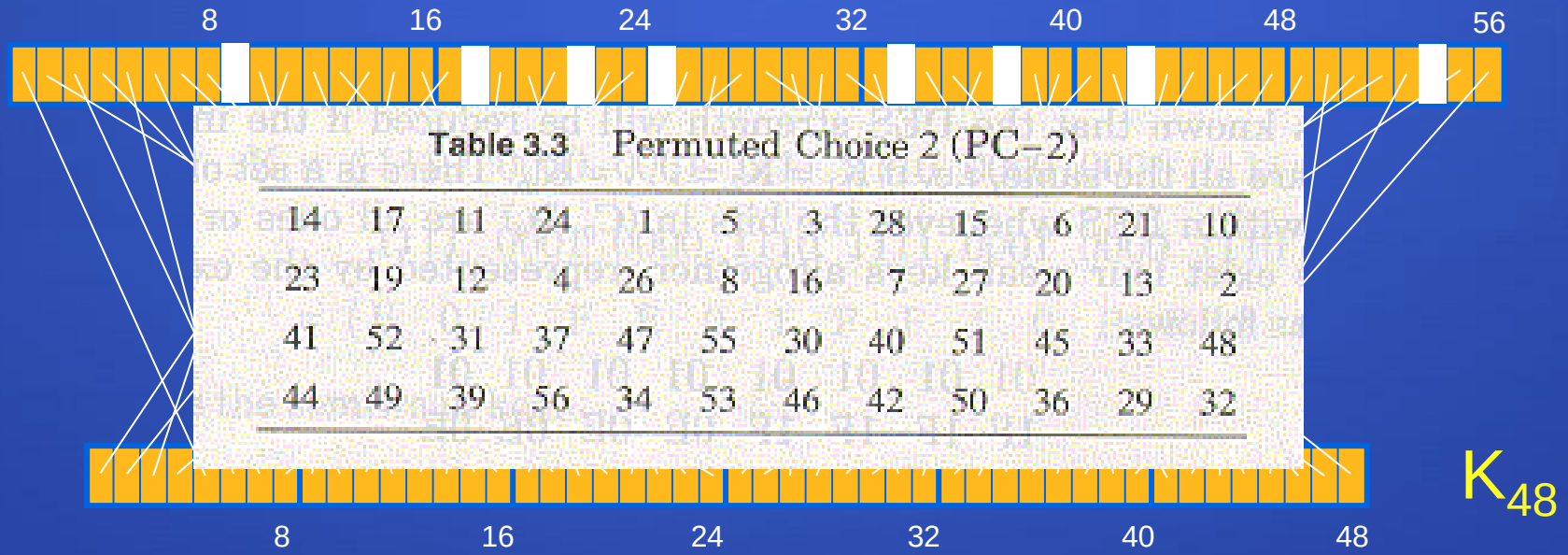


Shift left by N_i

Shift left by N_i

Shift accumulates every round

N_i = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1}



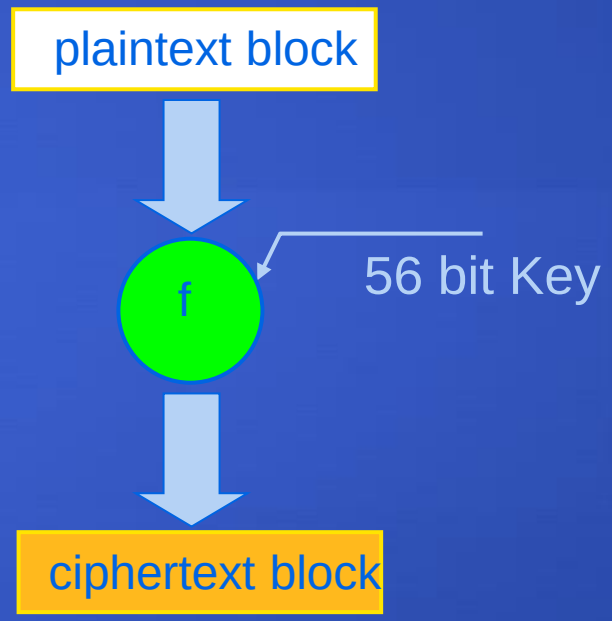
DES Advantages:

Very Fast:

Ideally suited for implementation in hardware (bit shifts, look-ups etc).

Dedicated hardware (in 1996) could run DES at 200 Mbyte/s.

Well suited for voice, video etc.



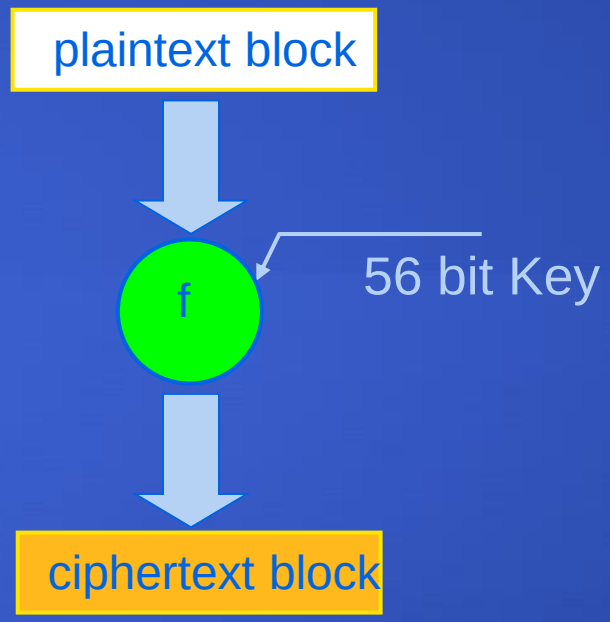
DES Security:

Not too good:

Trying all 2^{56} possible keys
is not that hard these days.

If you spend ~$25k you can build
a DES password cracker that will
succeed in a few hours.

Back in 1975 this would have cost
a few billion $$. It is widely believed
that the NSA did this.



Similar algorithms with longer keys are available today (AES,IDEA).

DES Example:

Let **M** be the plain text message **M** = 0123456789ABCDEF where **M** is in hexadecimal (base 16) format.

Rewriting **M** in binary format, we get the 64-bit block of text:
M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

Left and Right Hand sides are:
L = 0000 0001 0010 0011 0100 0101 0110 0111
R = 1000 1001 1010 1011 1100 1101 1110 1111

K is the hexadecimal key **K** = 133457799BBCDFF1 – in binary this is:-

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

DES Example:

Step 1: Create 16 subkeys, each of which is 48-bits long

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation from PC-1
K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, [illegible] half has 28 bi[illegible]

Exam
C_0 =
D_0 =

Table 3.1 Permuted Choice 1 (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

DES Example:

Step 1: Create 16 subkeys, each of which is 48-bits long

From original pair pair C_0 and D_0 we obtain:

C_0 = 111100001100110010101010111
D_0 = 010101010110011001111001111
C_1 = 111000011001100101010111111
D_1 = 101010101100110011110011110
C_2 = 110000110011001010101111111
D_2 = 010101011001100111100111101
C_3 = 000011001100101010101111111
D_3 = 010101100110011110001110101
C_4 = 001100110010101010111111100
D_4 = 010110011001111000111010101
C_5 = 110011001010101011111110000
D_5 = 011001100111100011101010101
C_6 = 001100101010101111111000011
D_6 = 100110011110001111010101010
C_7 = 110010101010111111100001100
D_7 = 011001111000111101010101110
C_8 = 001010101011111111000110011
D_8 = 100111100011110101010011001
C_9 = 010101010111111100001100110
D_9 = 001111000111101010101100111
C_{10} = 010101011111111000011001100
D_{10} = 111100011110101010011001100
C_{11} = 010101111111100001100110010
D_{11} = 110001111010101010110011001
C_{12} = 010111111110000110011001010
D_{12} = 000111101010101100110011111
C_{13} = 011111111000011001100101010
D_{13} = 011110101010101100110011110
C_{14} = 111111100001100110010101010
D_{14} = 111010101010110011001111000
C_{15} = 111110000110011001010101011
D_{15} = 101010101011001100111100011
C_{16} = 111100001100110010101010111
D_{16} = 010101010110011001111000111

DES Example:

Step 1: Create 16 subkeys, each of which is 48-bits long

We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but **PC-2** only uses 48 of these.

Table 3.3 Permuted Choice 2 (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

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

DES Example:

Step 1: Create 16 subkeys, each of which is 48-bits long

For the first key we have:

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

which, after we apply the permutation **PC-2**, becomes

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

Table 3.3 Permuted Choice 2 (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

DES Example:

Step 1: Create 16 subkeys, each of which is 48-bits long

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

DES Example:

Step 2: Encode each 64-bit block of data

Applying the initial permutation (IP) to the block of text **M**, given previously, we get:

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

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

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

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

DES Example:

Step 2: Encode each 64-bit block of data

Next divide the permuted block IP into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

Example: From IP, we get L_0 and R_0

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

We now proceed through 16 iterations, for $1 <= n <= 16$, using a function ***f*** which operates on two blocks--a data block of 32 bits and a key K_n of 48 bits--to produce a block of 32 bits.

Let + denote XOR addition, (bit-by-bit addition modulo 2). Then for **n** going from 1 to 16 we calculate

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

DES Example:

Step 2: Encode each 64-bit block of data

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

For n = 1, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0 + f(R_0, K_1)$

It remains to explain how the function f works. To calculate f, we first expand each block R_{n-1} from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in R_{n-1} . We'll call the use of this selection table the function **E**. Thus **E**(R_{n-1}) has a 32 bit input block, and a 48 bit output block.

DES Example:

Step 2: Encode each 64-bit block of data

Thus **E**(R_{n-1}) has a 32 bit input block, and a 48 bit output block.

Table 3.5 *E* Bit-Selection Table

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

We calculate **E**(R_0) from R_0 as follows:

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010
E(R_0) = 011110 100001 010101 010101 011110 100001 010101 010101

DES Example:

Step 2: Encode each 64-bit block of data

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

Next in the ***f*** calculation, we XOR the output $\mathbf{E}(R_{n-1})$ with the key K_n:

$K_n + \mathbf{E}(R_{n-1})$.

Example: For K_1 , $\mathbf{E}(R_0)$, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

$K_1 + \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111

DES Example:

Step 2: Encode each 64-bit block of data

K_1+E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111

Write the previous result, which is 48 bits, in the form: $K_n + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$, where each B_i is a group of six bits.

We now calculate

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

where $S_i(B_i)$ referres to the output of the ***i***-th **S** box.

Table 3.6 Primitive S-Box Functions

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

DES Example:

Step 2: Encode each 64-bit block of data

For the first round, we obtain as the output of the eight **S** boxes:

$K_1 + \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$
$$=$$
0101 1100 1000 0010 1011 0101 1001 0111

The final stage in the calculation of ***f*** is to do a permutation **P** of the **S**-box output to obtain the final value of ***f***:

$f = \mathbf{P}(S_1(B_1)S_2(B_2)\ldots S_8(B_8))$

Table 3.7 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

DES Example:

Step 2: Encode each 64-bit block of data

From the output of the eight **S** boxes:

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

=

0101 1100 1000 0010 1011 0101 1001 0111

we get

f = 0010 0011 0100 1010 1010 1001 1011 1011

$R_1 = L_0 + f(R_0 , K_1) =$

1100 1100 0000 0000 1100 1100 1111 1111

+

0010 0011 0100 1010 1010 1001 1011 1011

=

1110 1111 0100 1010 0110 0101 0100 0100

DES Example:

Step 2: Encode each 64-bit block of data

In the next round, we will have $L_2 = R_1$, which is the block we just calculated, and then we must calculate $R_2 = L_1 + f(R_1, K_2)$, and so on for 16 rounds.

At the end of the sixteenth round we have the blocks L_{16} and R_{16}. We then ***reverse*** the order of the two blocks into the 64-bit block $R_{16}L_{16}$ and apply a final permutation IP^{-1} as defined by the following table:

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

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

DES Example:

Finally we get

If we process all 16 blocks using the method defined previously, we get, on the 16th round,

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101

We reverse the order of these two blocks and apply the final permutation to
$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

which in hexadecimal format is 85E813540F0AB405.
This is the encrypted form of **M** = 0123456789ABCDEF:
namely, **C** = 85E813540F0AB405.

Triple DES (3DES)

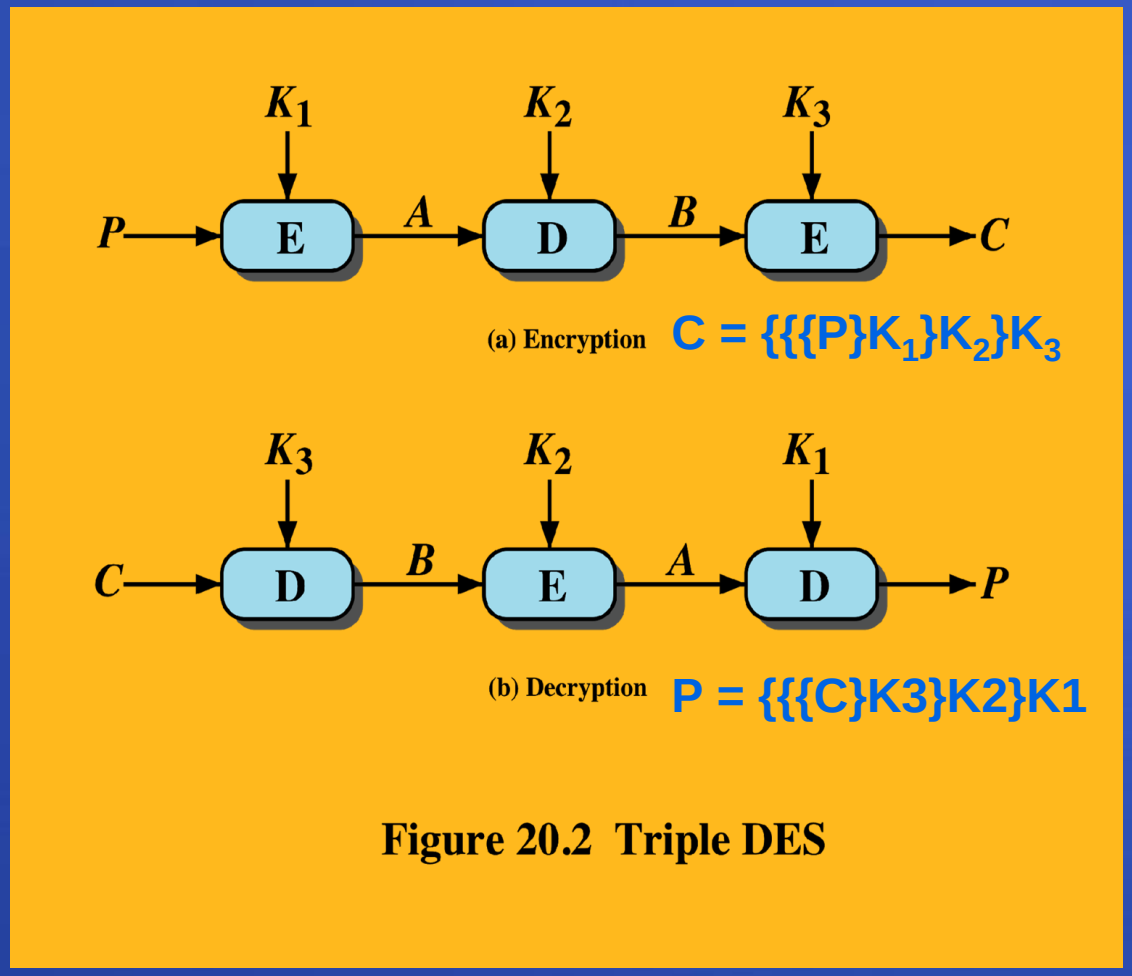


Figure 20.2 Triple DES

- First used in financial applications
- In DES FIPS PUB 46-3 standard of 1999
- Uses three keys and three DES executions:

$$C = E(K_3, D(K_2, E(K_1, P)))$$

- Decryption same with keys reversed
- Use of decryption in second stage gives compatibility with original DES users
- Effective 168-bit key length, slow, secure
- AES will eventually replace 3DES

Advanced Encryption Standard (AES)

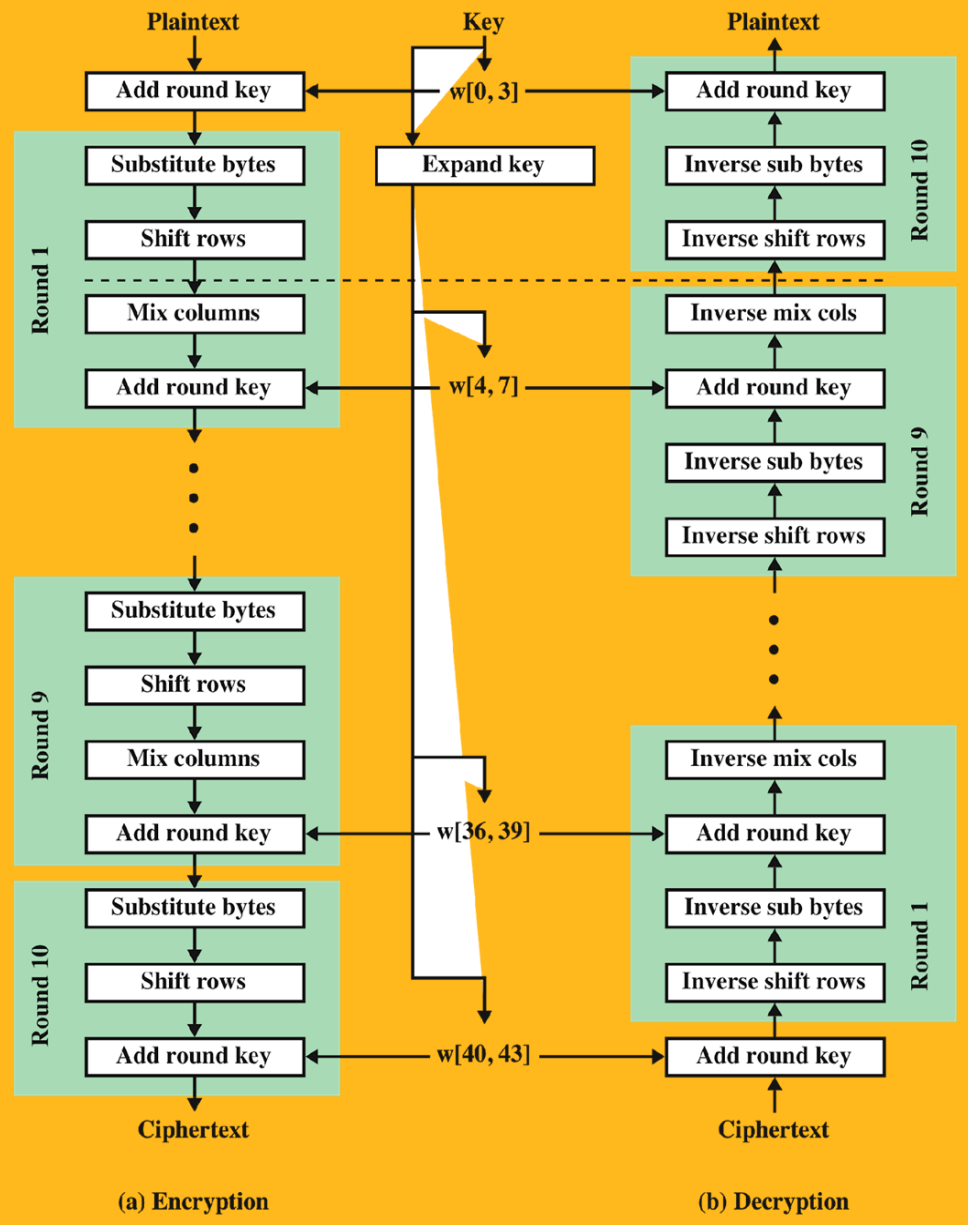


Figure 20.3 AES Encryption and Decryption

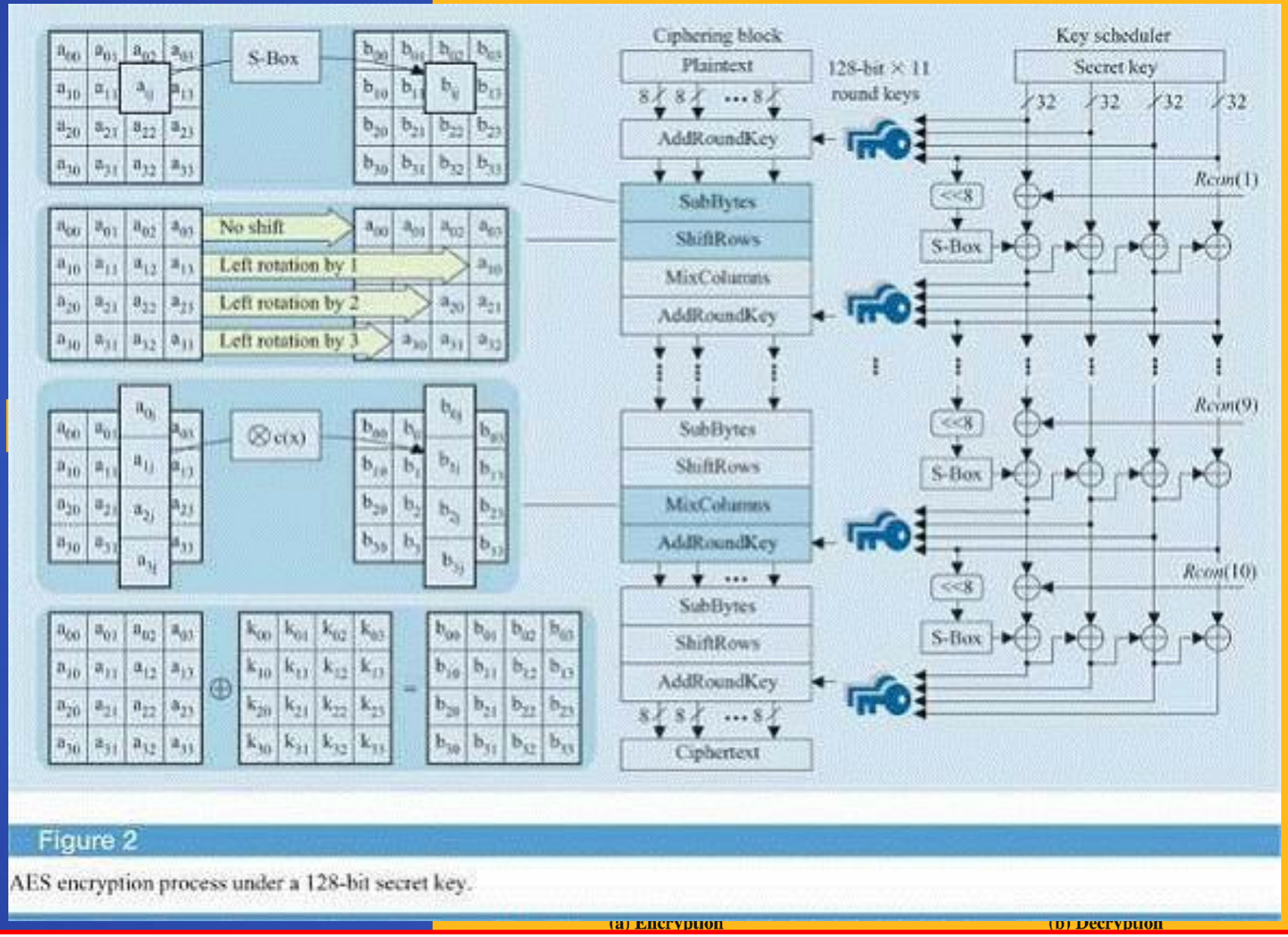


Figure 2

AES encryption process under a 128-bit secret key.

(a) Encryption (b) Decryption

Click Here for a good demo of AES.

https://www.youtube.com/watch?v=mlzxpkdXP58

Table 20.2
(a) S-box

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	[illegible]	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	[illegible]	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table 20.2
(b) Inverse S-box

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	[illegible]	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	[illegible]	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Shift Rows

- On encryption left rotate each row of State by 0,1,2,3 bytes respectively
- Decryption does reverse
- To move individual bytes from one column to another and spread bytes over columns

Mix Columns and Add Key

- Mix columns
  - Operates on each column individually
  - Mapping each byte to a new value that is a function of all four bytes in the column
  - Use of equations over finite fields
  - To provide good mixing of bytes in column
- Add round key
  - Simply XOR State with bits of expanded key
  - Security from complexity of round key expansion and other stages of AES

Stream Ciphers

- Processes input elements continuously
- Key input to a pseudorandom bit generator
  - Produces stream of random like numbers
  - Unpredictable without knowing input key
  - XOR keystream output with plaintext bytes
- Are faster and use far less code
- Design considerations:
  - Encryption sequence should have a large period
  - Keystream approximates random number properties
  - Uses a sufficiently long key

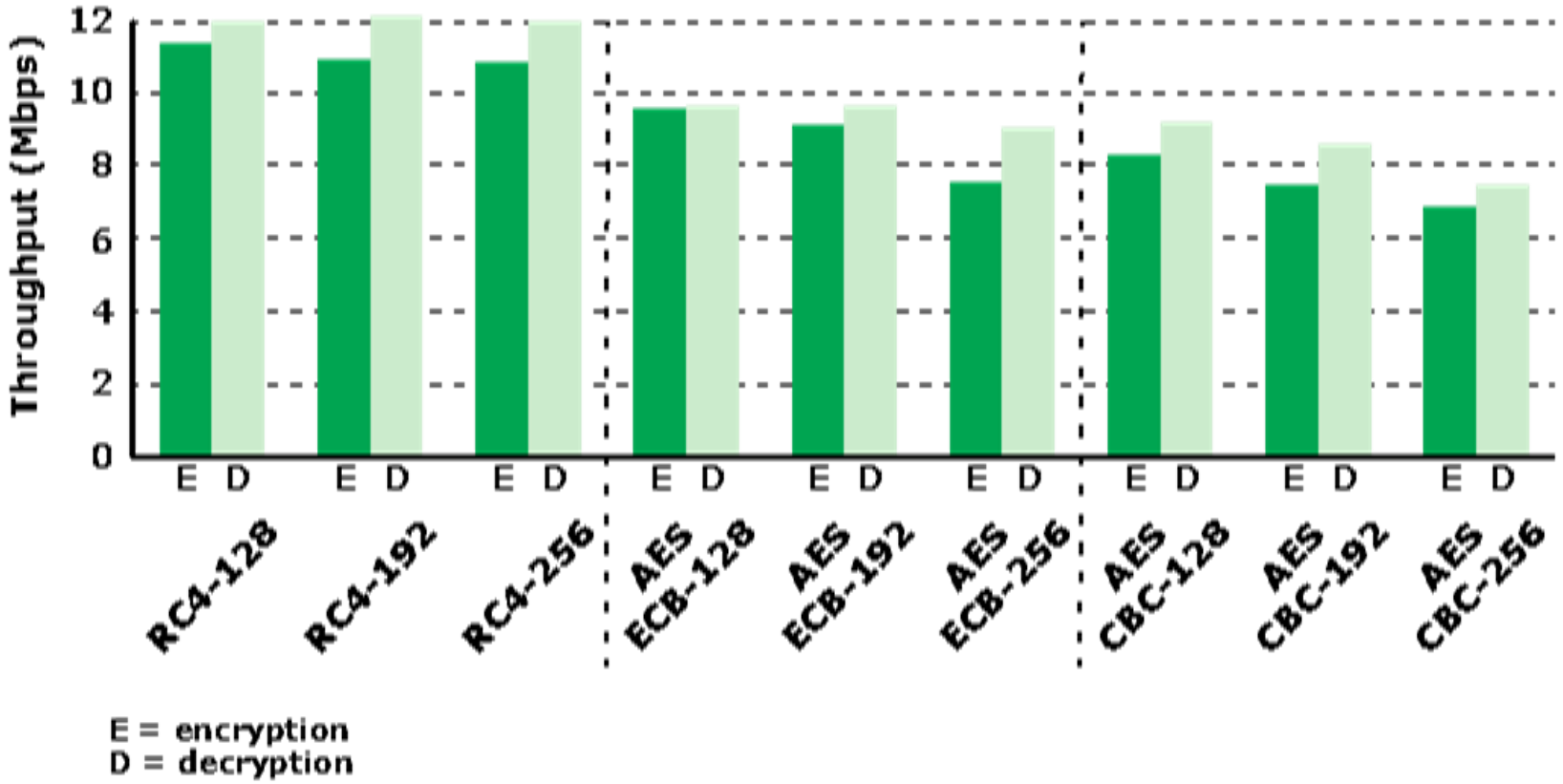


Figure 20.5 Performance Comparison of Symmetric Ciphers on a 3-GHz Processor

RC4 and WEP

WEP vs WPA vs WPA2

	WEP	WPA	WPA2
ENCRYPTION	RC4	RC4	AES
KEY ROTATION	NONE	Dynamic Session Keys	Dynamic Session Keys
KEY DISTRIBUTION	Manually typed into each device	Automatic distribution available	Automatic distribution available
AUTHENTICATION	Uses WEP key as Authentication	Can use 802.1x & EAP	Can use 802.1x & EAP

Symmetric Algorithms

Symmetric Encryption Algorithm	Key length (in bits)	Description
DES	56	Designed at IBM during the 1970s and was the NIST standard until 1997. Although considered outdated, DES remains widely in use. Designed to be implemented only in hardware, and is therefore extremely slow in software.
3DES	112 and 168	Based on using DES three times which means that the input data is encrypted three times and therefore considered much stronger than DES. However, it is rather slow compared to some new block ciphers such as AES.
AES	128, 192, and 256	Fast in both software and hardware, is relatively easy to implement, and requires little memory. As a new encryption standard, it is currently being deployed on a large scale.
Software Encryption Algorithm (SEAL)	160	SEAL is an alternative algorithm to DES, 3DES, and AES. It uses a 160-bit encryption key and has a lower impact to the CPU when compared to other software-based algorithms.
The RC series	RC2 (40 and 64) RC4 (1 to 256) RC5 (0 to 2040) RC6 (128, 192, and 256)	A set of symmetric-key encryption algorithms invented by Ron Rivest. RC1 was never published and RC3 was broken before ever being used. RC4 is the world's most widely used stream cipher. RC6, a 128-bit block cipher based heavily on RC5, was an AES finalist developed in 1997.

Selecting an Algorithm

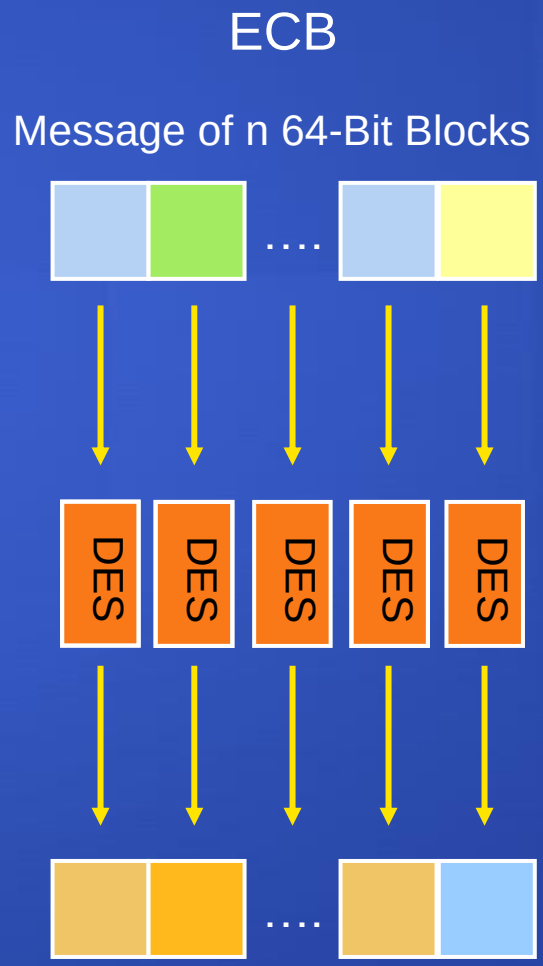
	DES	3DES	AES
The algorithm is trusted by the cryptographic community	Been replaced by 3DES	Yes	Verdict is still out
The algorithm adequately protects against brute-force attacks	No	Yes	Yes

Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

Electronic Codebook (ECB)

- Simplest mode
- Plaintext is handled *b* bits at a time and each block is encrypted using the same key
- "codebook" because you have a unique ciphertext value for each plaintext block
  - Not secure for long messages since repeated plaintext is seen in repeated ciphertext
- To overcome security deficiencies you need a technique where the same plaintext block, if repeated, produces different ciphertext blocks



Cipher Block Chaining (CBC)

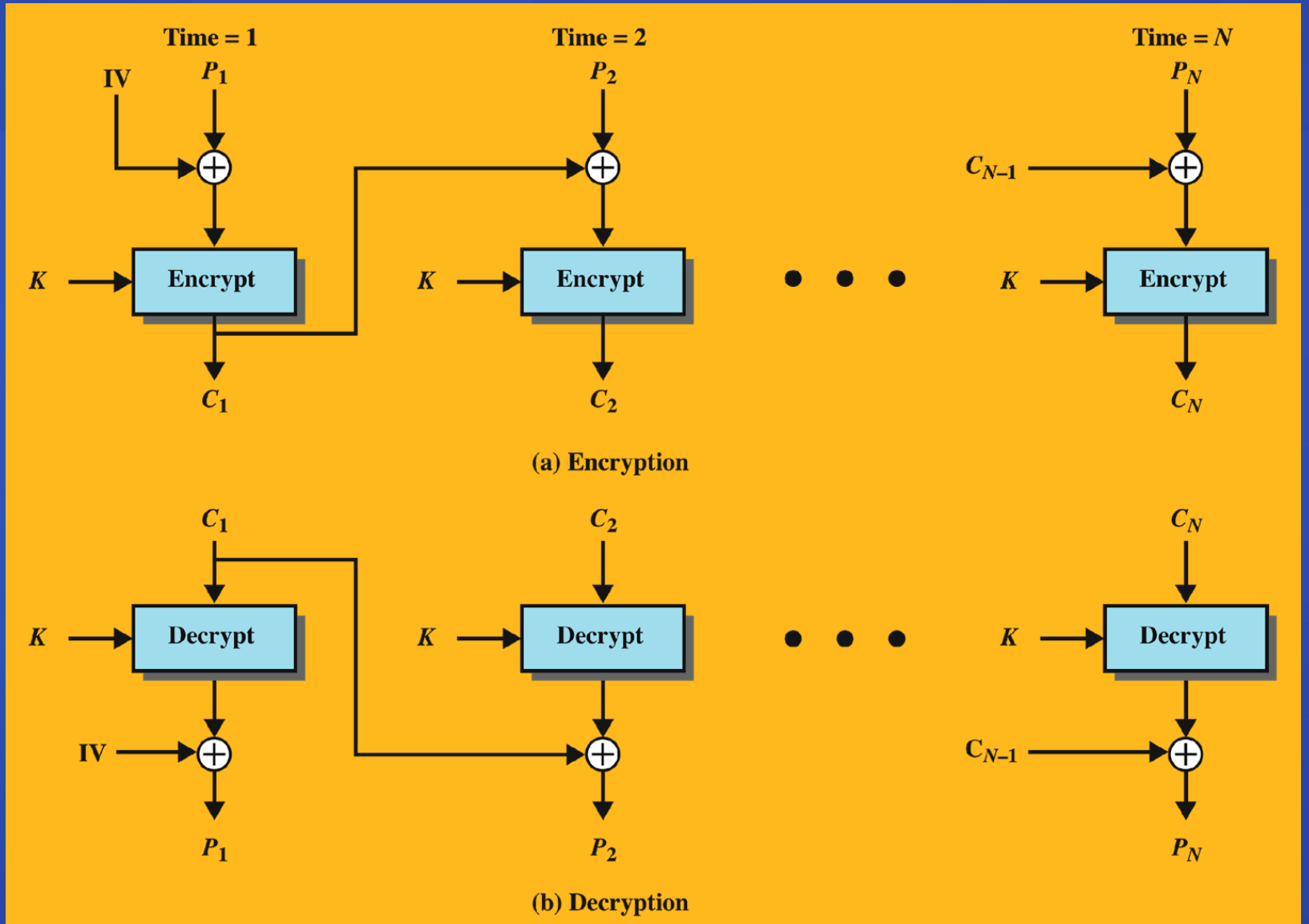


Figure 20.6 Cipher Block Chaining (CBC) Mode

Cipher Feedback (CFB)

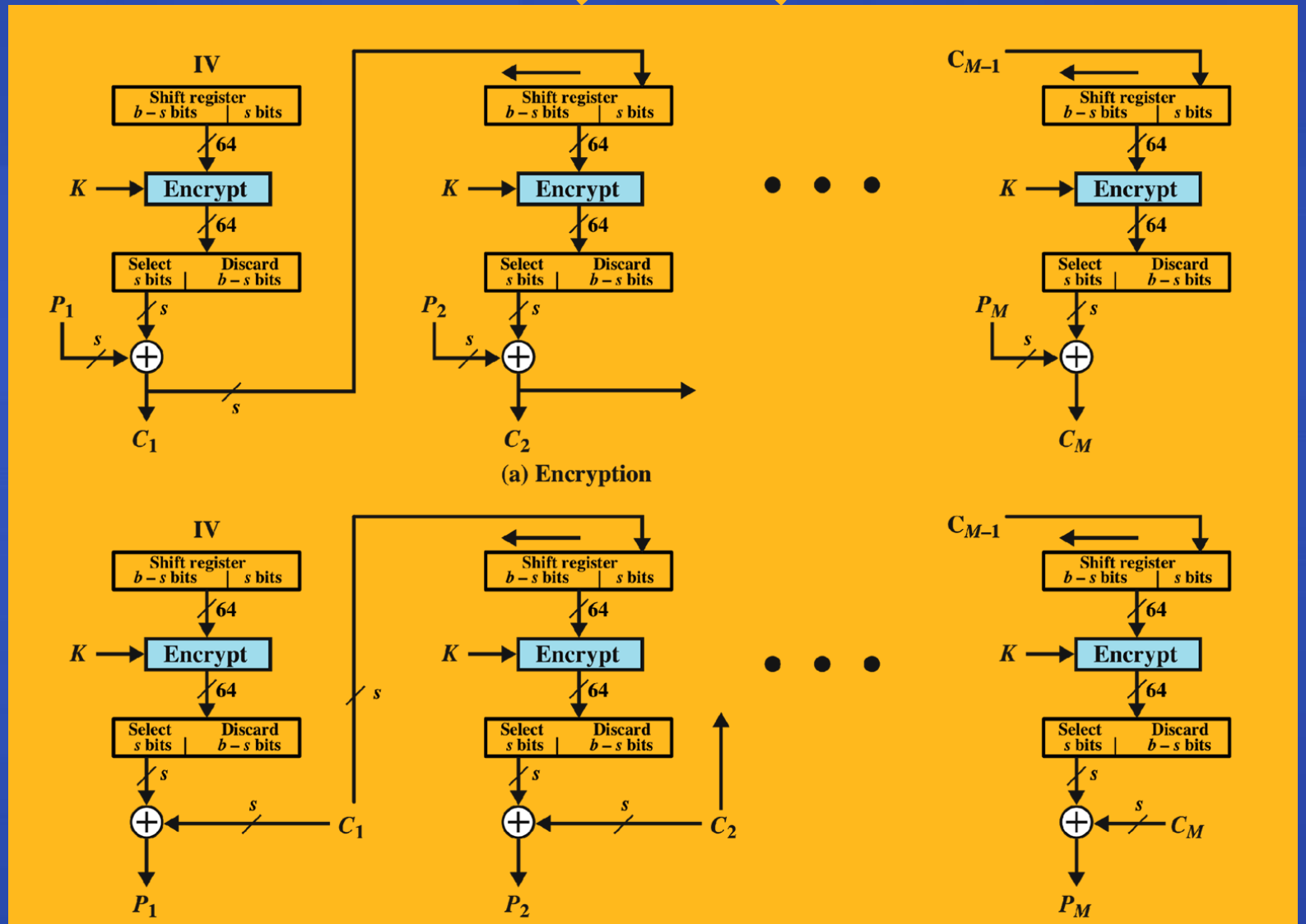


Figure 20.7 *s*-bit Cipher Feedback (CFB) Mode

Counter (CTR)

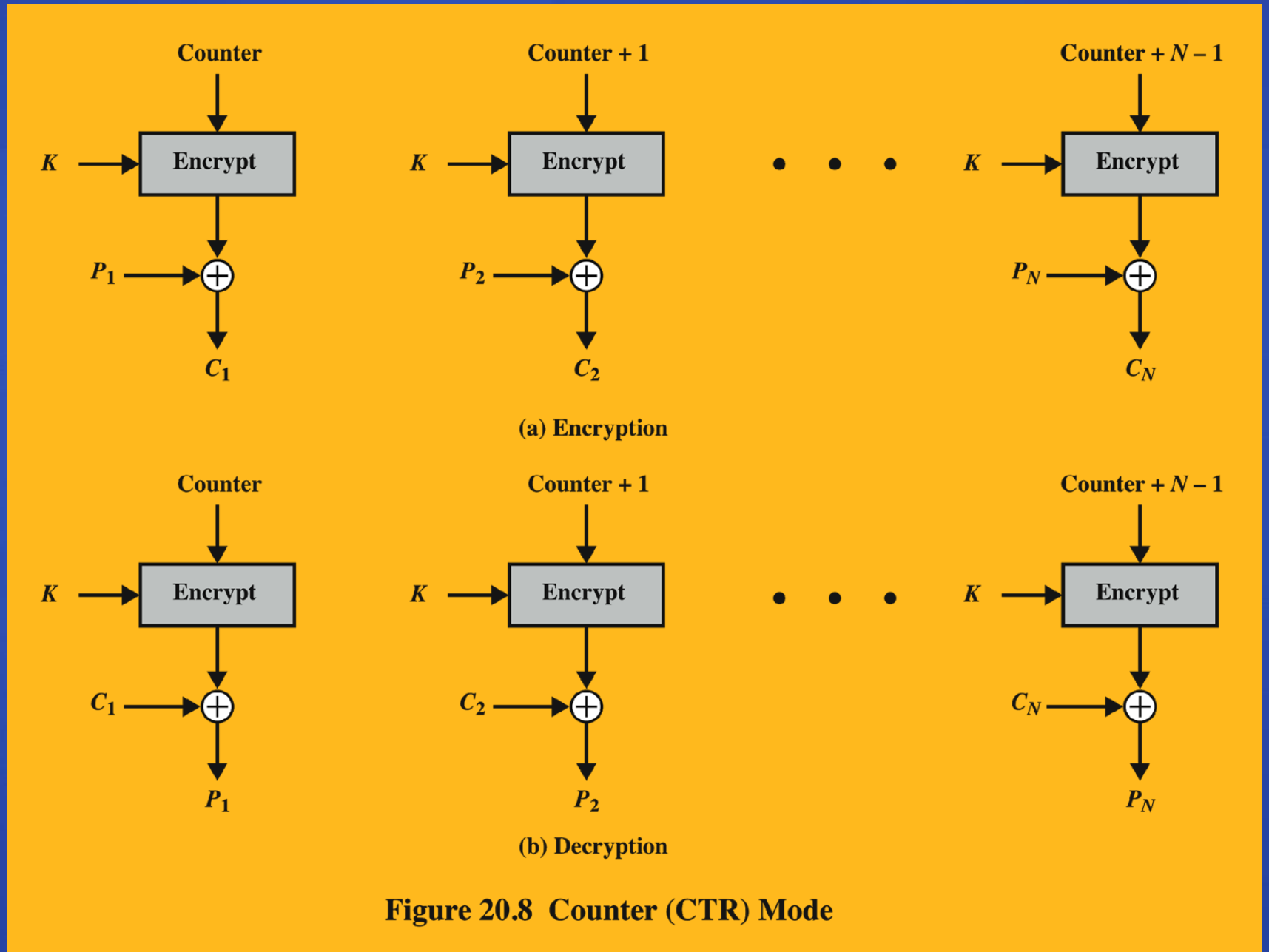


Figure 20.8 Counter (CTR) Mode

Location of Encryption

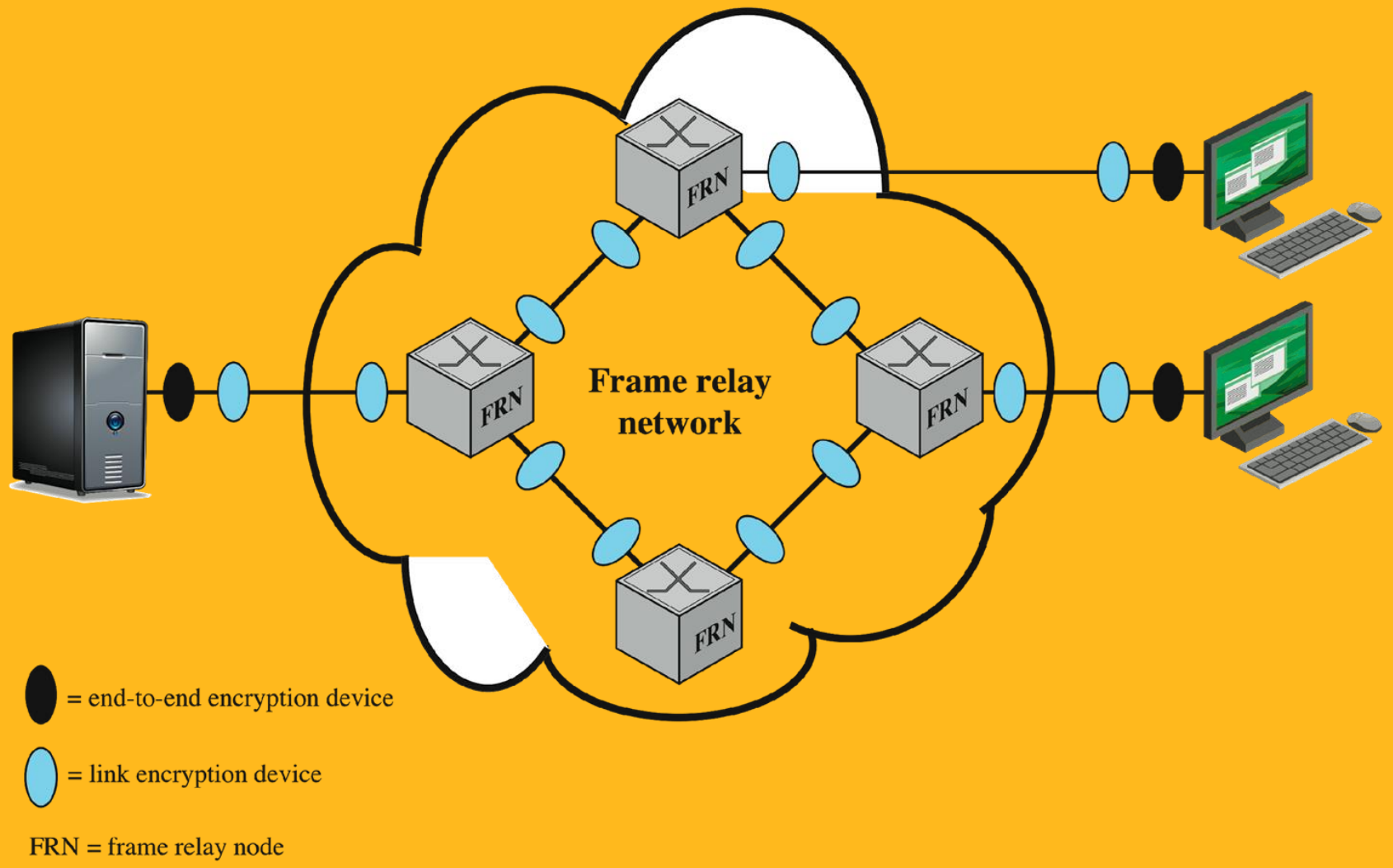


Figure 20.9 Encryption Across a Frame Relay Network

Key Distribution

- **The means of delivering a key to two parties that wish to exchange data without allowing others to see the key**
- **Two parties (A and B) can achieve this by:**
  - **1**
    - A key could be selected by A and physically delivered to B
  - **2**
    - A third party could select the key and physically deliver it to A and B
  - **3**
    - If A and B have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key
  - **4**
    - If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B

Key Distribution

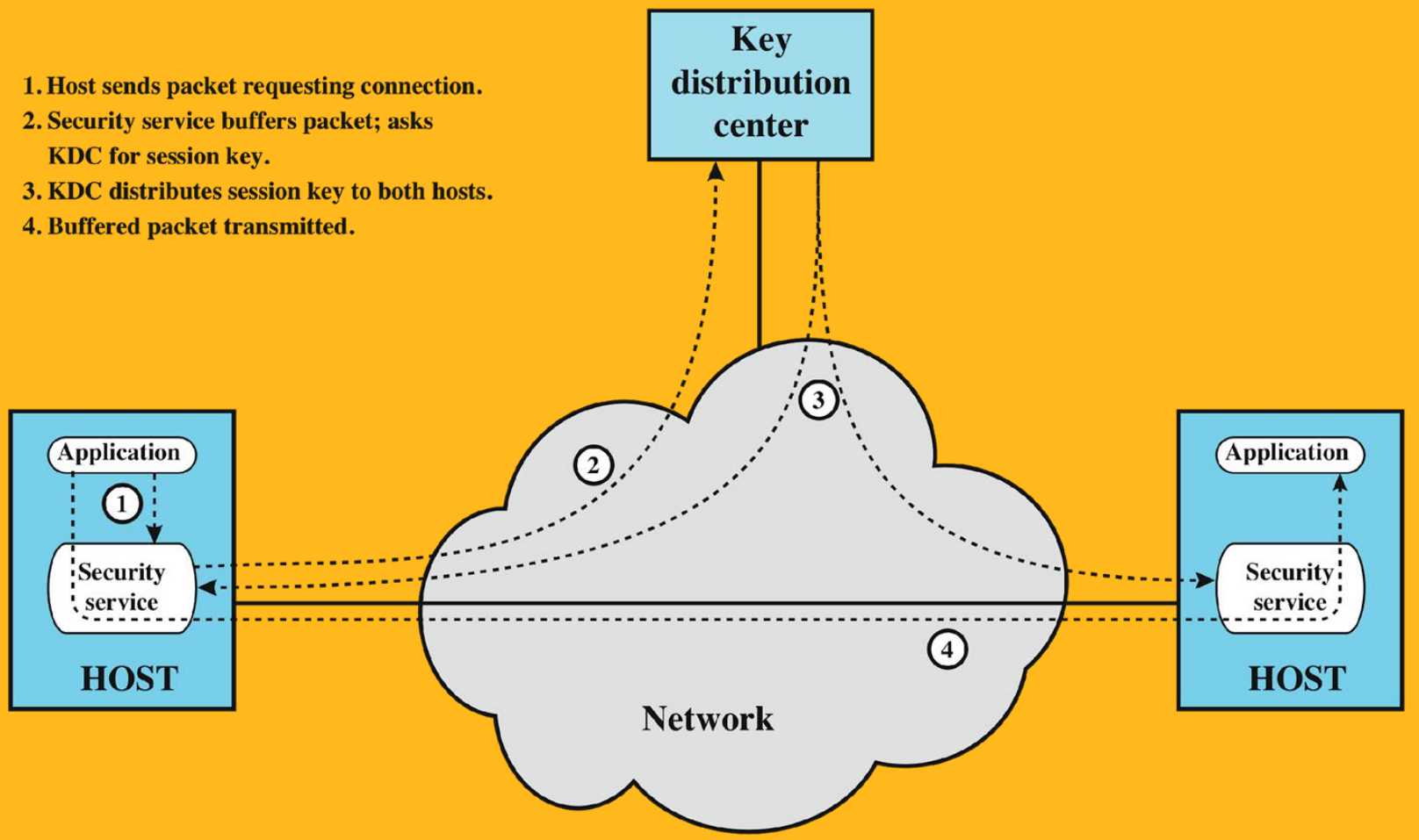


Figure 20.10 Automatic Key Distribution for Connection-Oriented Protocol

Summary

- Symmetric encryption principles
  - cryptography
  - cryptanalysis
  - Feistel cipher structure
- Data encryption standard (DES)
  - Triple DES
- Advanced encryption standard (AES)
  - Algorithm details
- Key distribution intro
- Stream ciphers
  - Stream cipher structure
  - RC4 and WEP
- Cipher block modes of operation
  - Electronic codebook mode
  - Cipher block chaining mode
  - Cipher feedback mode
  - Counter mode
- Location of symmetric encryption devices