Unit – 4

Symmetric Cipher & Asymmetric Cipher

Syllabus:

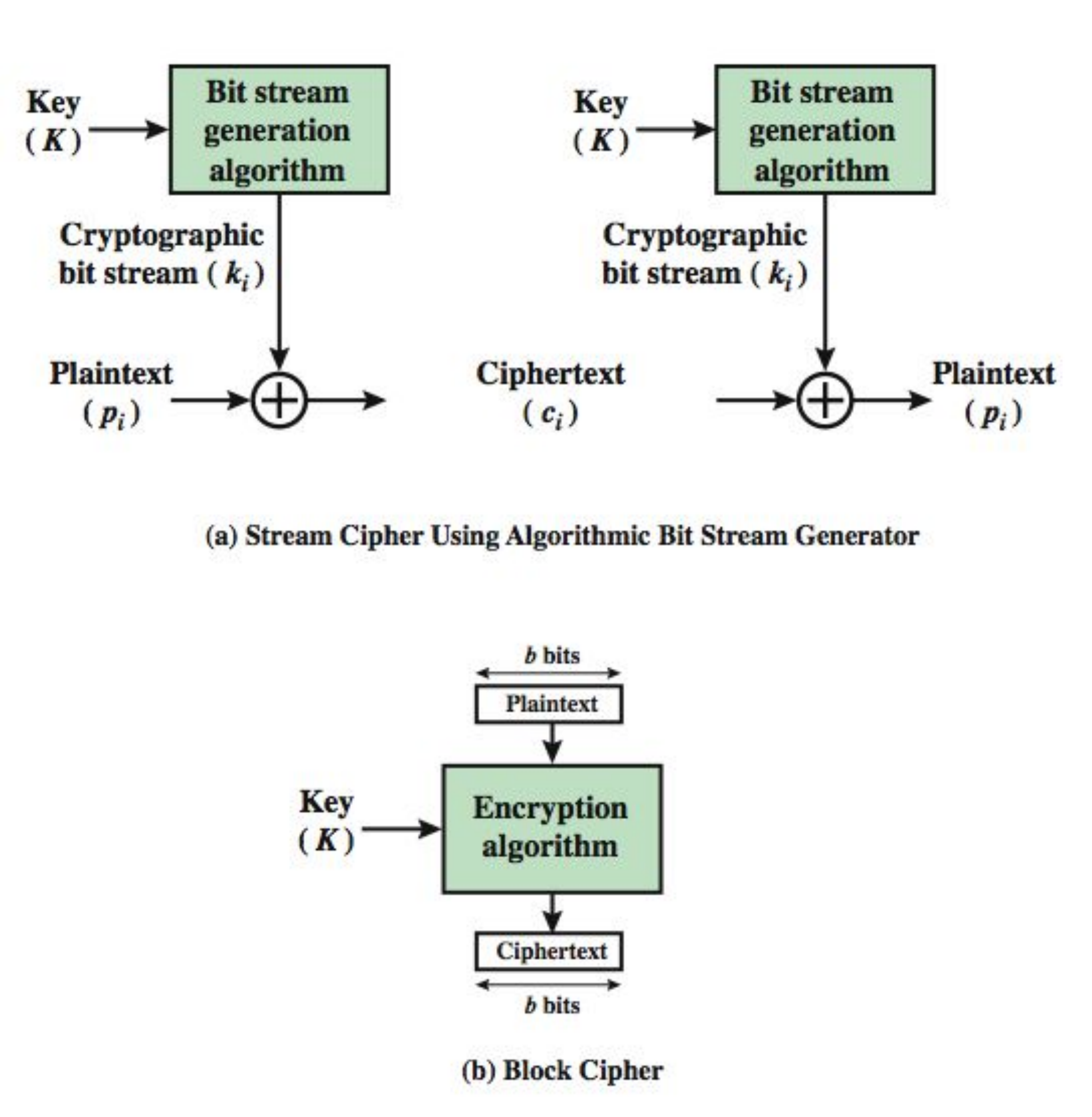
Symmetric Cipher: Stream Ciphers and Block Ciphers; Feistel Cipher Structure, Data Encryption Standard (DES): DES Encryption; DES Decryption; DES Example; Strength of DES

Asymmetric Cipher: Public-Key Cryptosystems, RSA Algorithm, Diffie-Hellman Key Exchange Algorithm

Block vs. Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on very big characters
  - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
  - better analyzed
  - broader range of applications

Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator

(b) Block Cipher

Block Cipher Principles

- most symmetric block ciphers are based on a **Feistel Cipher Structure**
- needed since must be able to **decrypt** ciphertext to recover messages efficiently
- block ciphers look like an extremely large substitution
- would need table of 2^{64} entries for a 64-bit block
- instead create from smaller building blocks
- using idea of a product cipher

Claude Shannon and Substitution-Permutation Ciphers

- Claude Shannon introduced idea of substitution-permutation (S-P) networks in 1949 paper
- form basis of modern block ciphers
- S-P nets are based on the two primitive cryptographic operations seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- provide *confusion* & *diffusion* of message & key

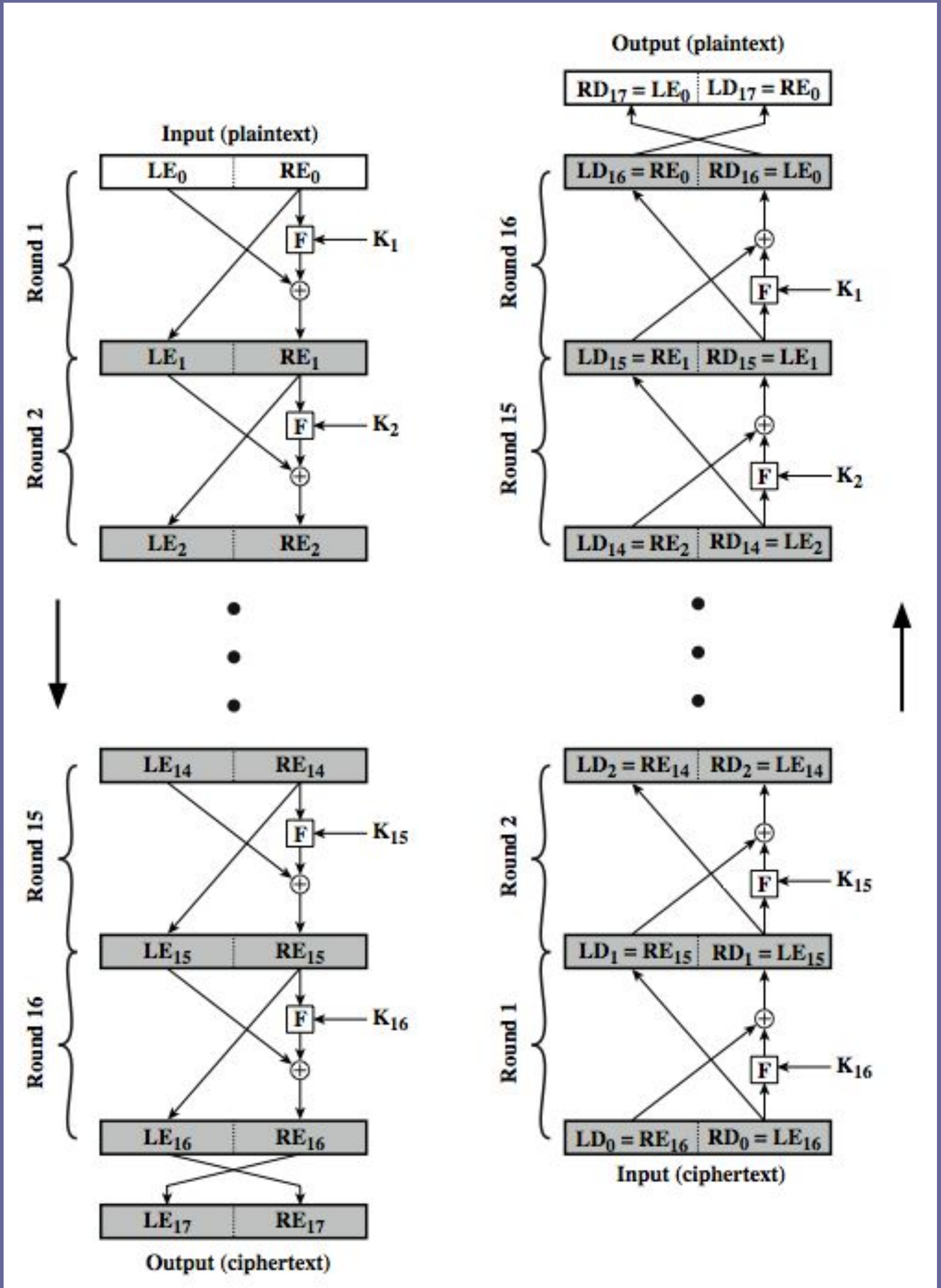
Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message
- a one-time pad does this
- more practically Shannon suggested combining S & P elements to obtain:
- **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **confusion** – makes relationship between ciphertext and key as complex as possible

Feistel Cipher Structure

- Horst Feistel devised the **feistel cipher**
  - based on concept of invertible product cipher
- partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves
- implements Shannon’s S-P net concept

Feistel Cipher Structure



Feistel Cipher Design Elements

- block size
- key size
- number of rounds
- subkey generation algorithm
- round function
- fast software en/decryption
- ease of analysis

Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security

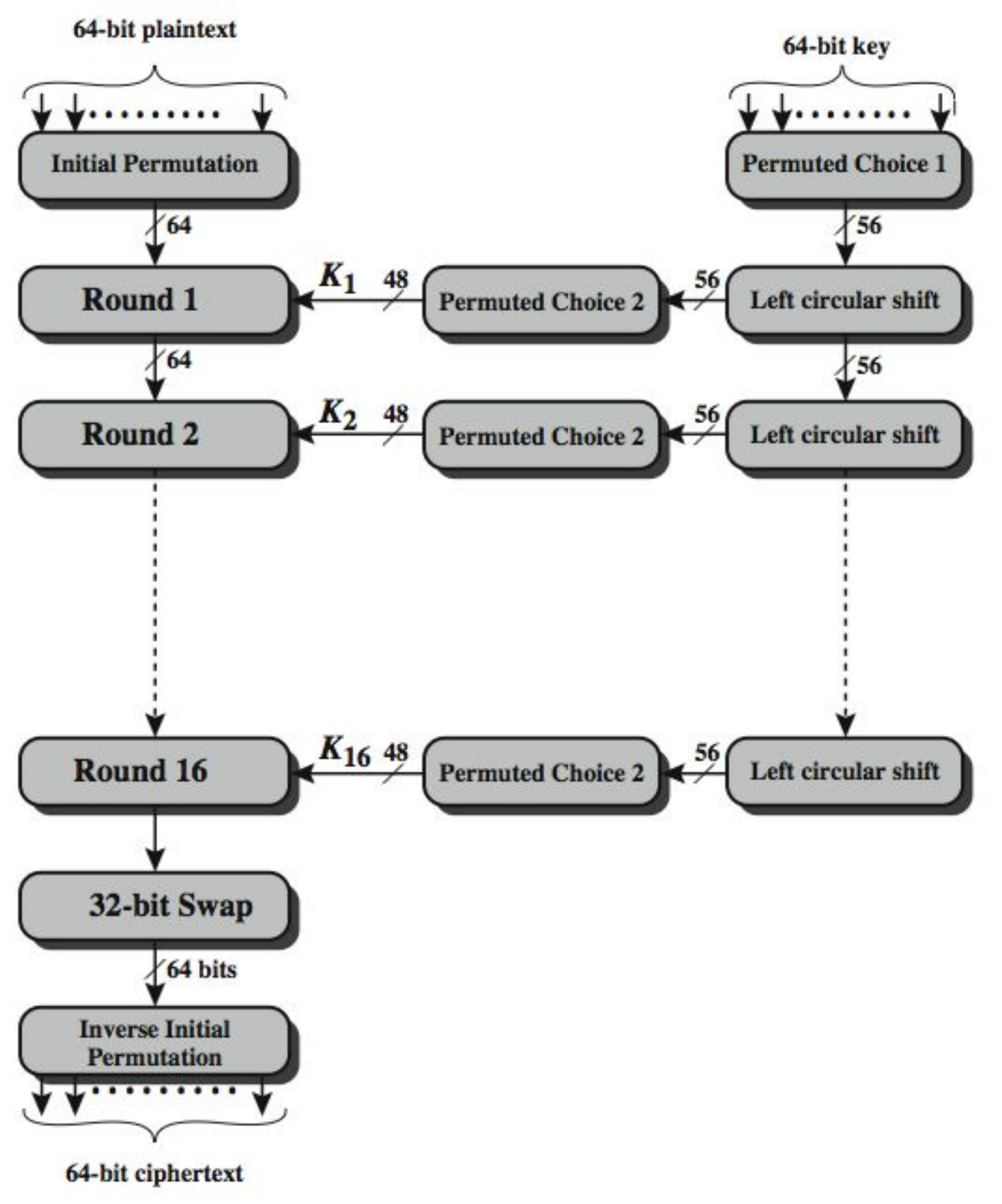
DES History

- IBM developed Lucifer cipher
  - by team led by Feistel in late 60’s
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

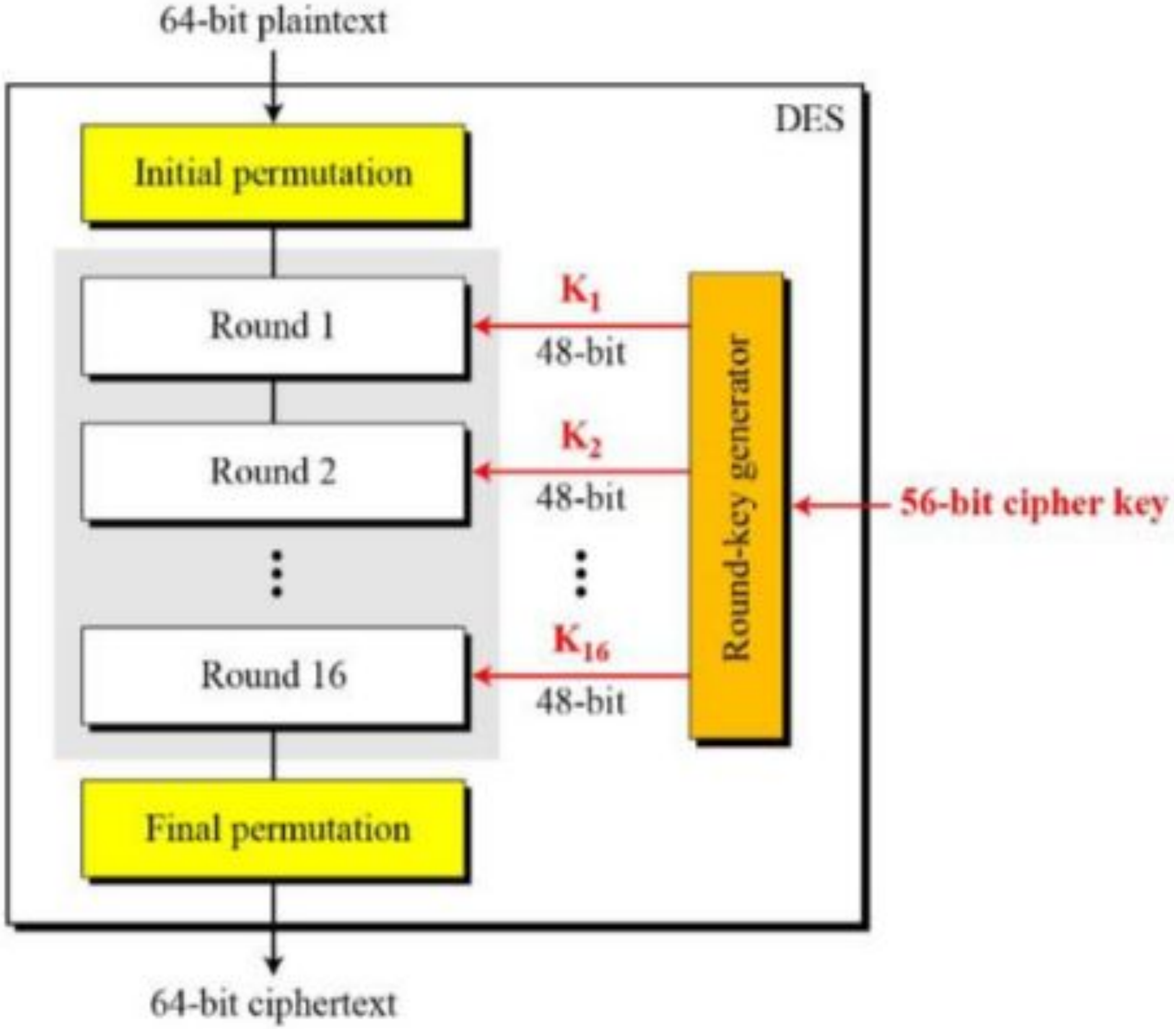
DES Design Controversy

- although DES standard is public
- was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
  - especially in financial applications
  - still standardised for legacy application use

DES Encryption Overview



General structure of DES

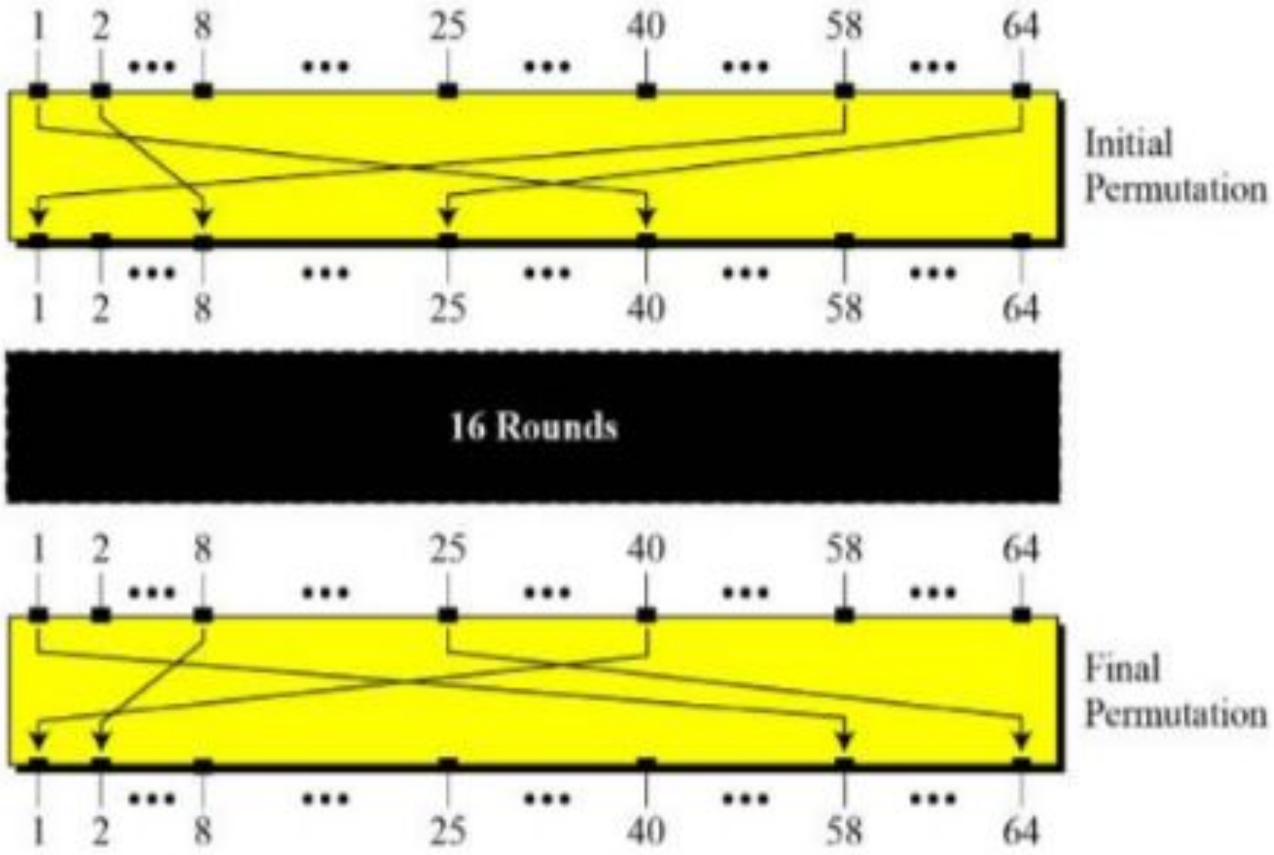


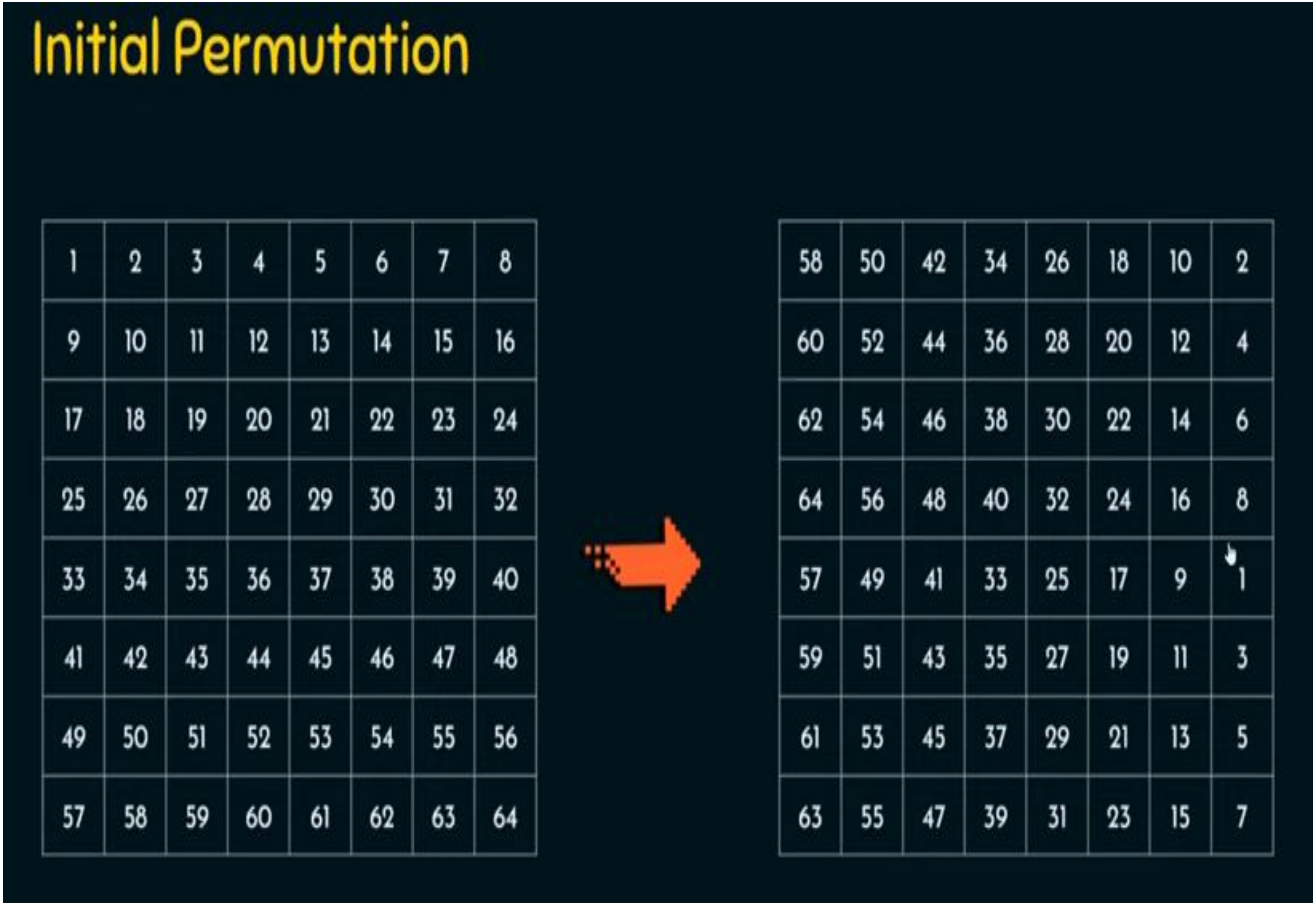
Initial Permutation (IP)

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)
- example:

IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

Initial and final permutation steps in DES





Initial Permutation

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

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

Initial and final permutation tables

Initial Permutation								*Final Permutation*							
58	50	42	34	26	18	10	02	40	08	48	16	56	24	64	32
60	52	44	36	28	20	12	04	39	07	47	15	55	23	63	31
62	54	46	38	30	22	14	06	38	06	46	14	54	22	62	30
64	56	48	40	32	24	16	08	37	05	45	13	53	21	61	29
57	49	41	33	25	17	09	01	36	04	44	12	52	20	60	28
59	51	43	35	27	19	11	03	35	03	43	11	51	19	59	27
61	53	45	37	29	21	13	05	34	02	42	10	50	18	58	26
63	55	47	39	31	23	15	07	33	01	41	09	49	17	57	25

How to read this table?

The 58th bit of input x will be the first bit of output IP(x), the 50th bit of x is the second bit of IP(x), etc.

The initial and final permutations are straight P-boxes that are inverses of each other. They have no cryptography significance in DES.

DES Round Structure

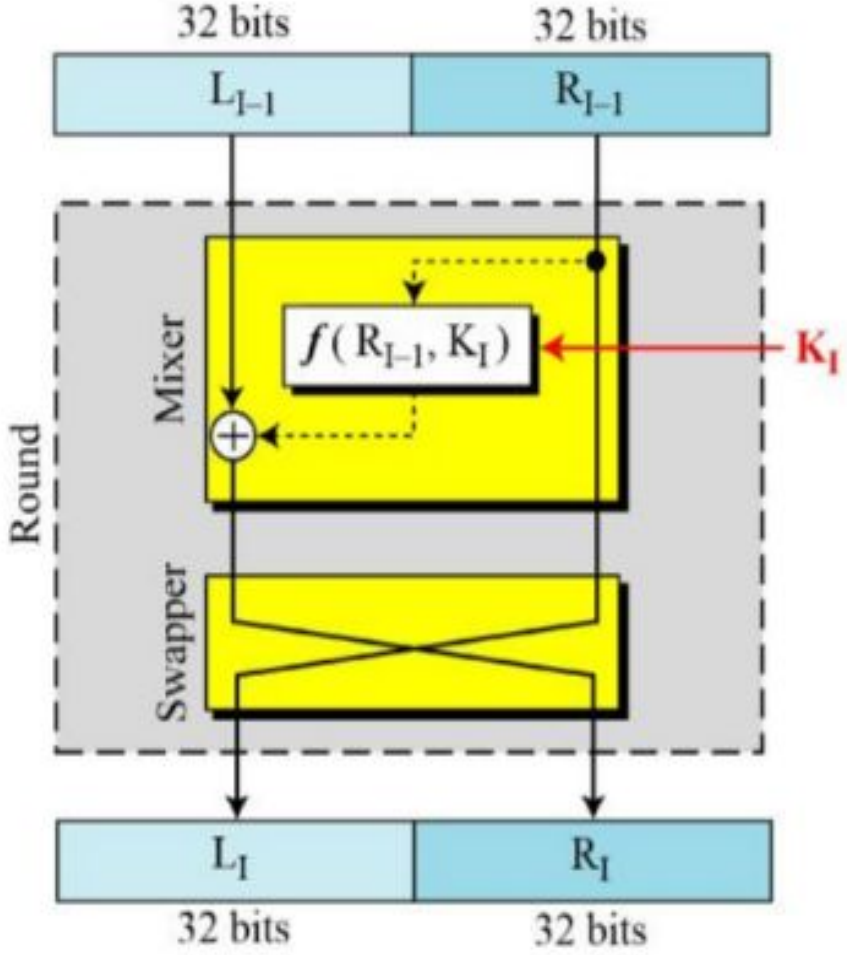
- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

  $L_i = R_{i-1}$

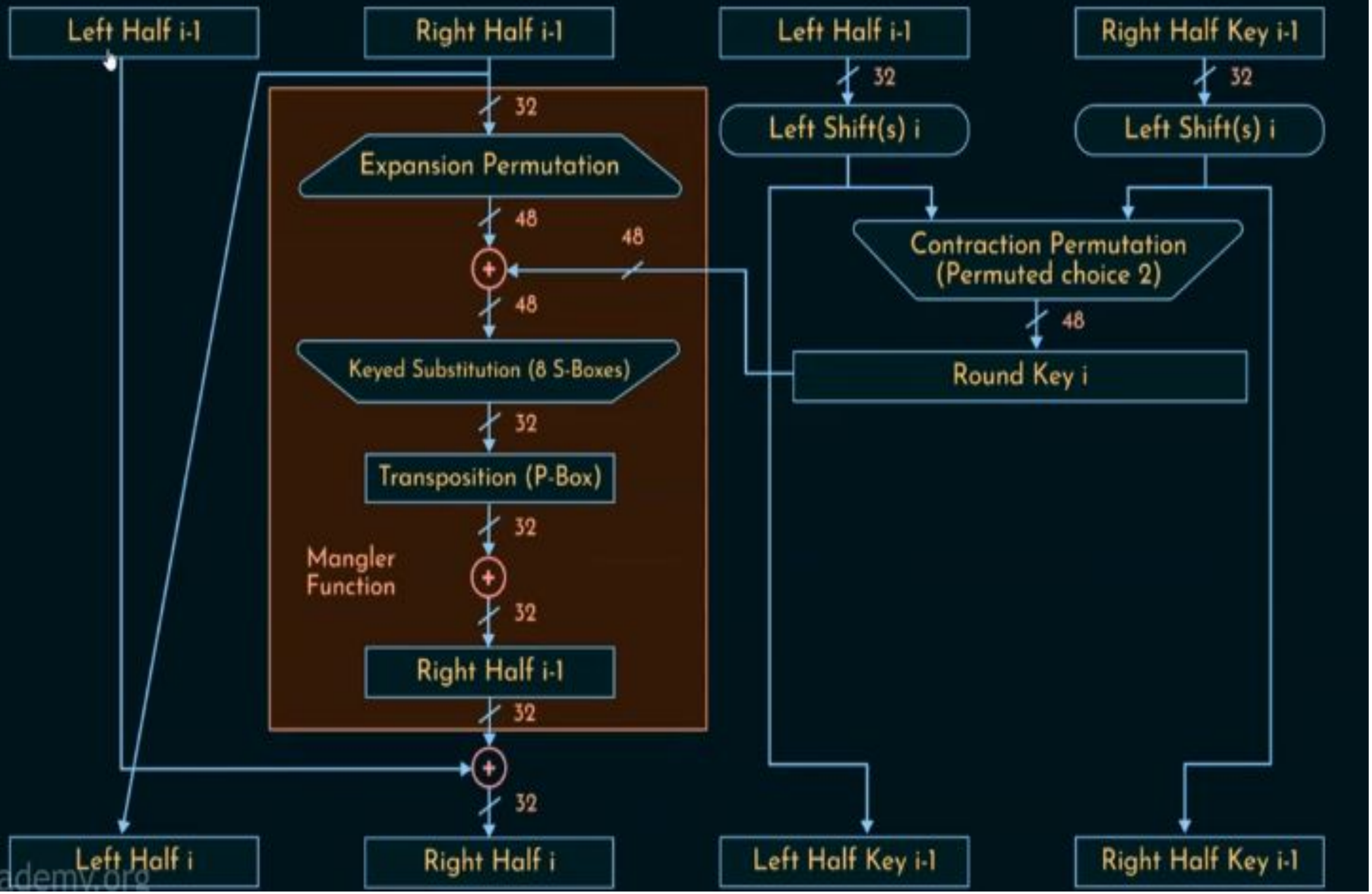
  $R_i = L_{i-1} \square \mathrm{F}(R_{i-1}, K_i)$
- F takes 32-bit R half and 48-bit subkey:
  - expands R to 48-bits using perm E
  - adds to subkey using XOR
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes using 32-bit perm P

DES uses 16 rounds. Each round of DES is a Feistel cipher.

A round in DES (encryption site)



Single Round of DES Algorithm



Left Half i-1

Right Half i-1

Left Half i-1

Right Half Key i-1

32

32

32

Left Shift(s) i

Left Shift(s) i

Expansion Permutation

48

48

Contraction Permutation
(Permuted choice 2)

48

48

Round Key i

Keyed Substitution (8 S-Boxes)

32

Transposition (P-Box)

32

Mangler
Function

32

Right Half i-1

32

32

Left Half i

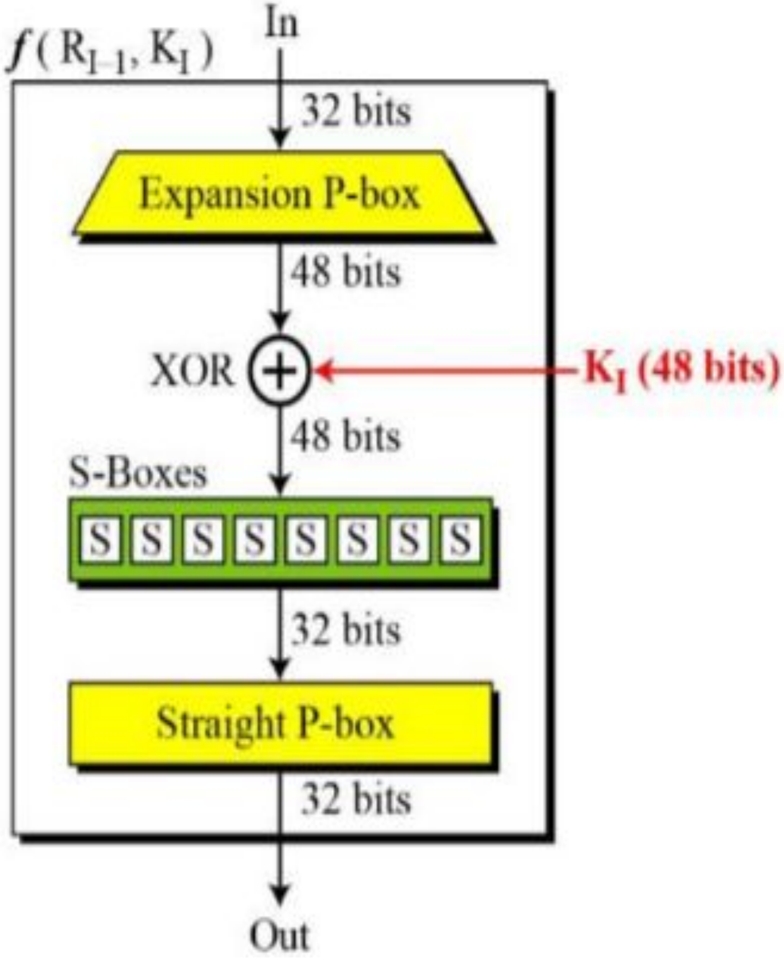
Right Half i

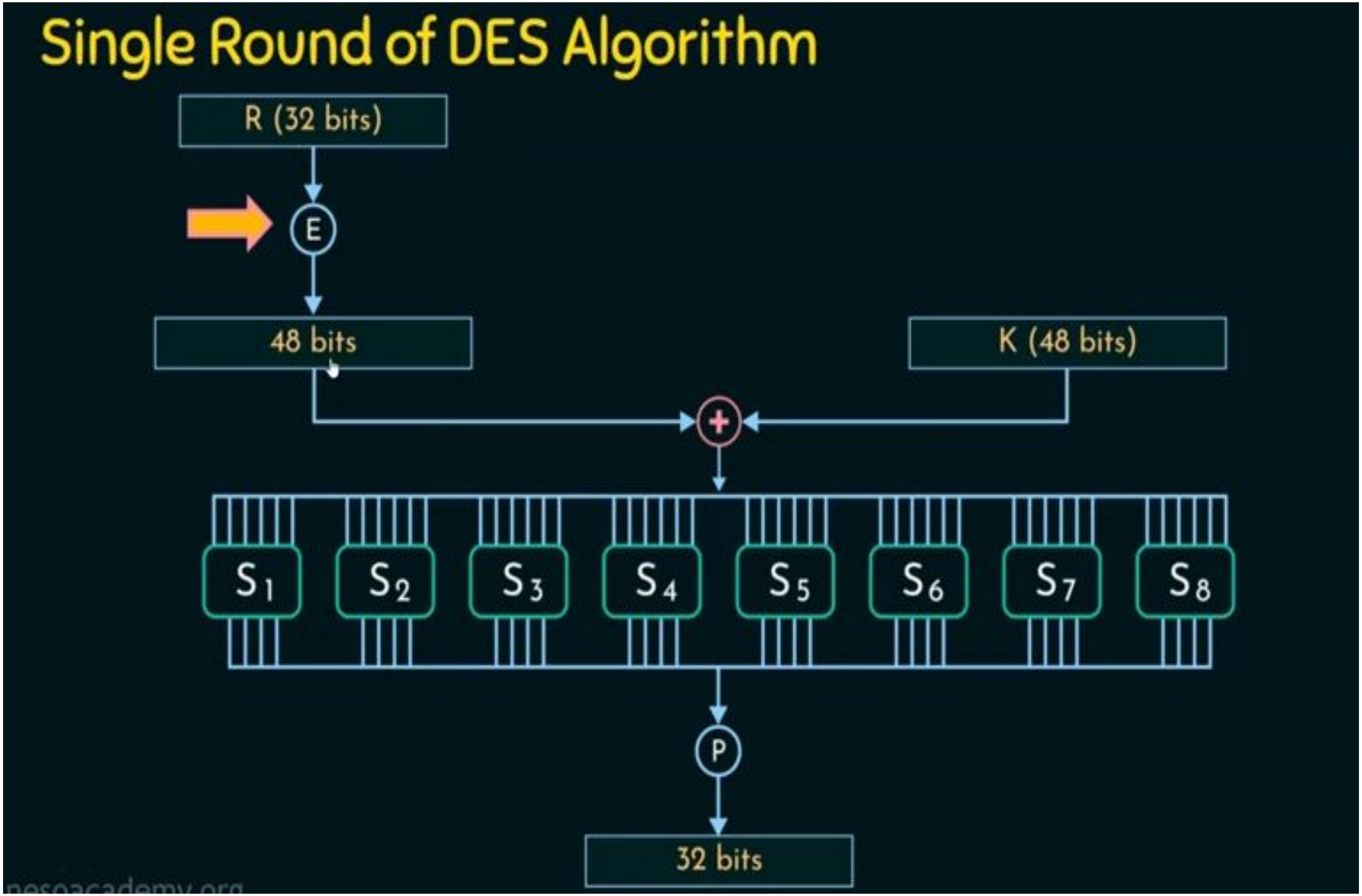
Left Half Key i-1

Right Half Key i-1

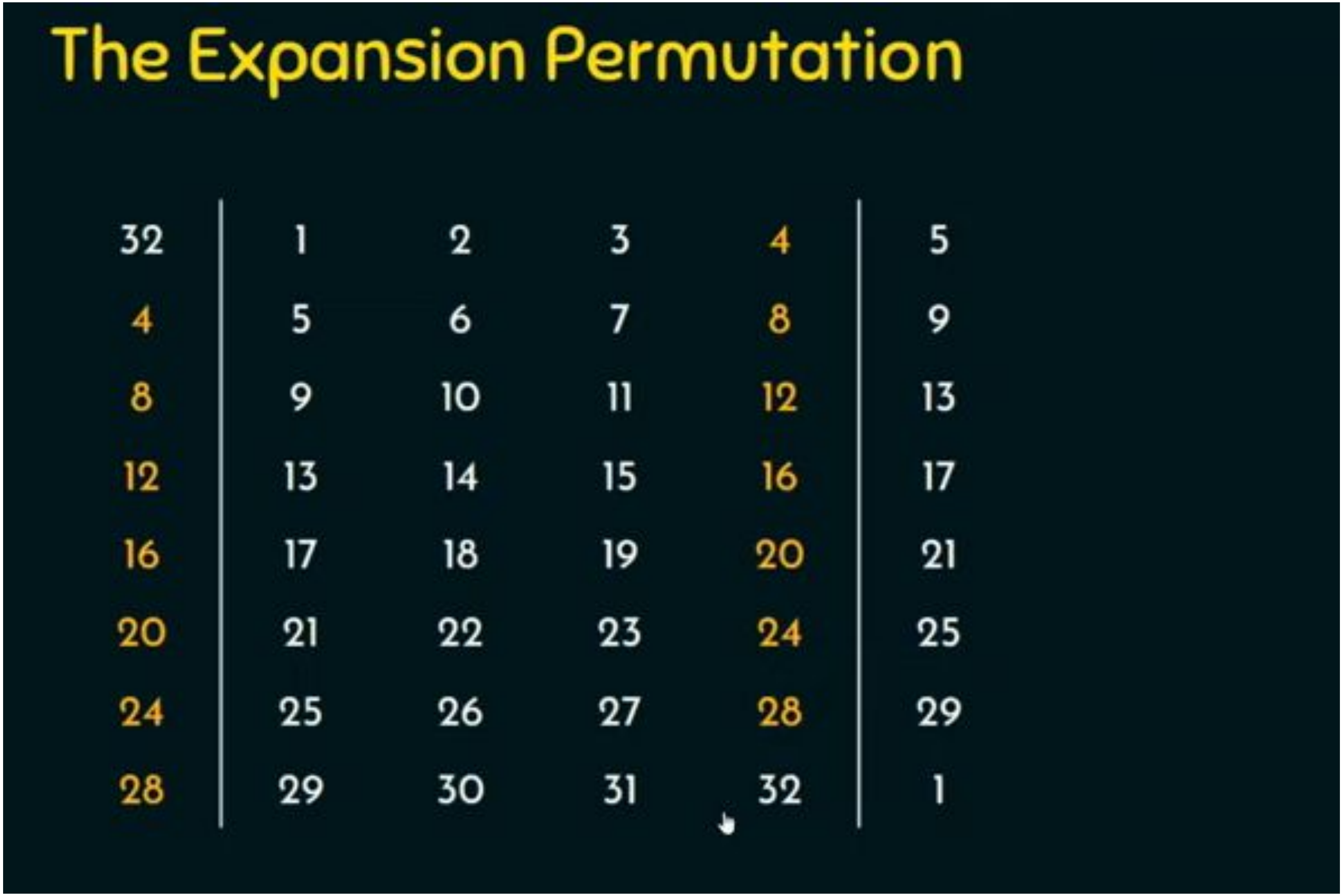
The heart of DES is the DES function. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

DES function





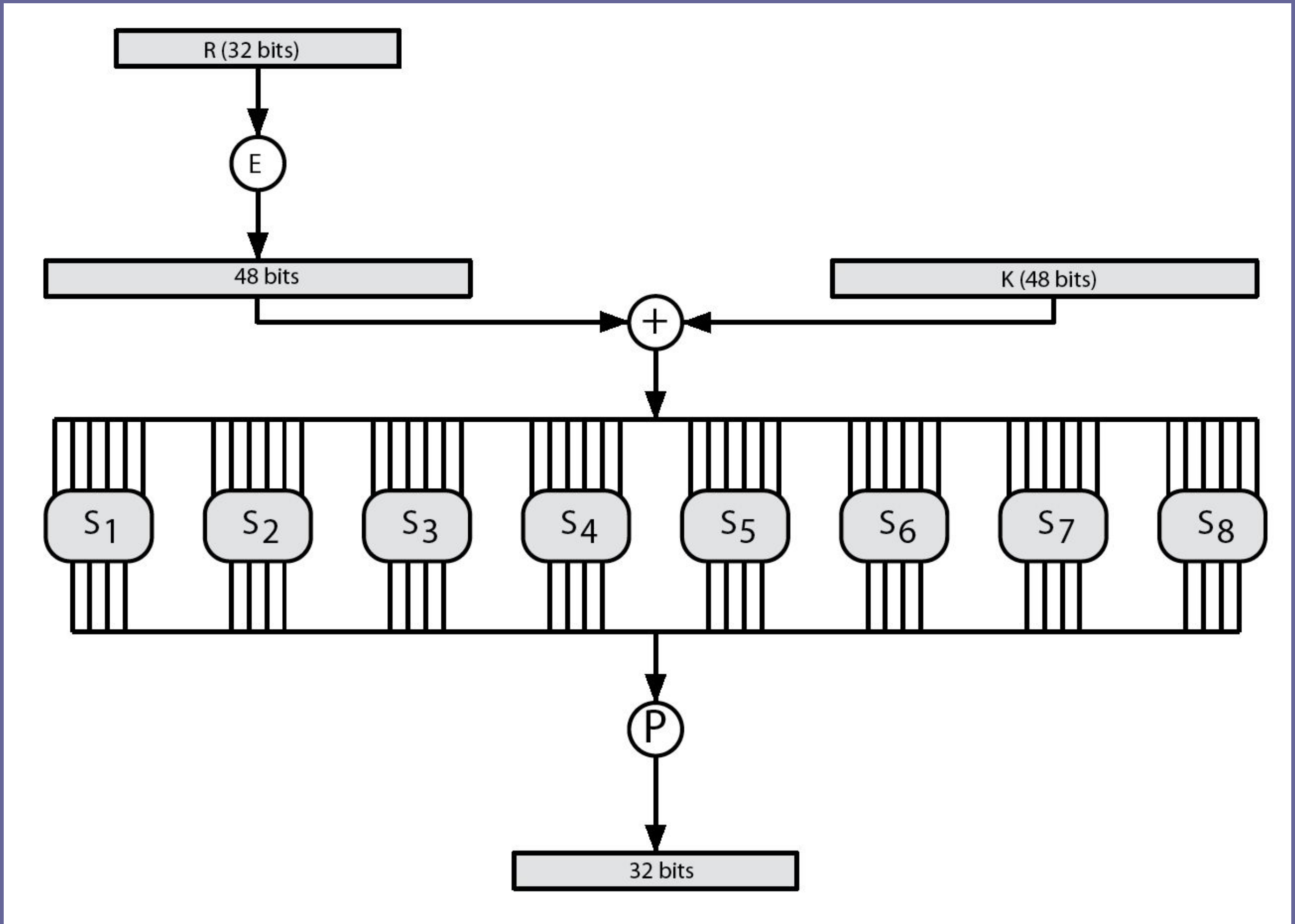
Single Round of DES Algorithm



The Expansion Permutation

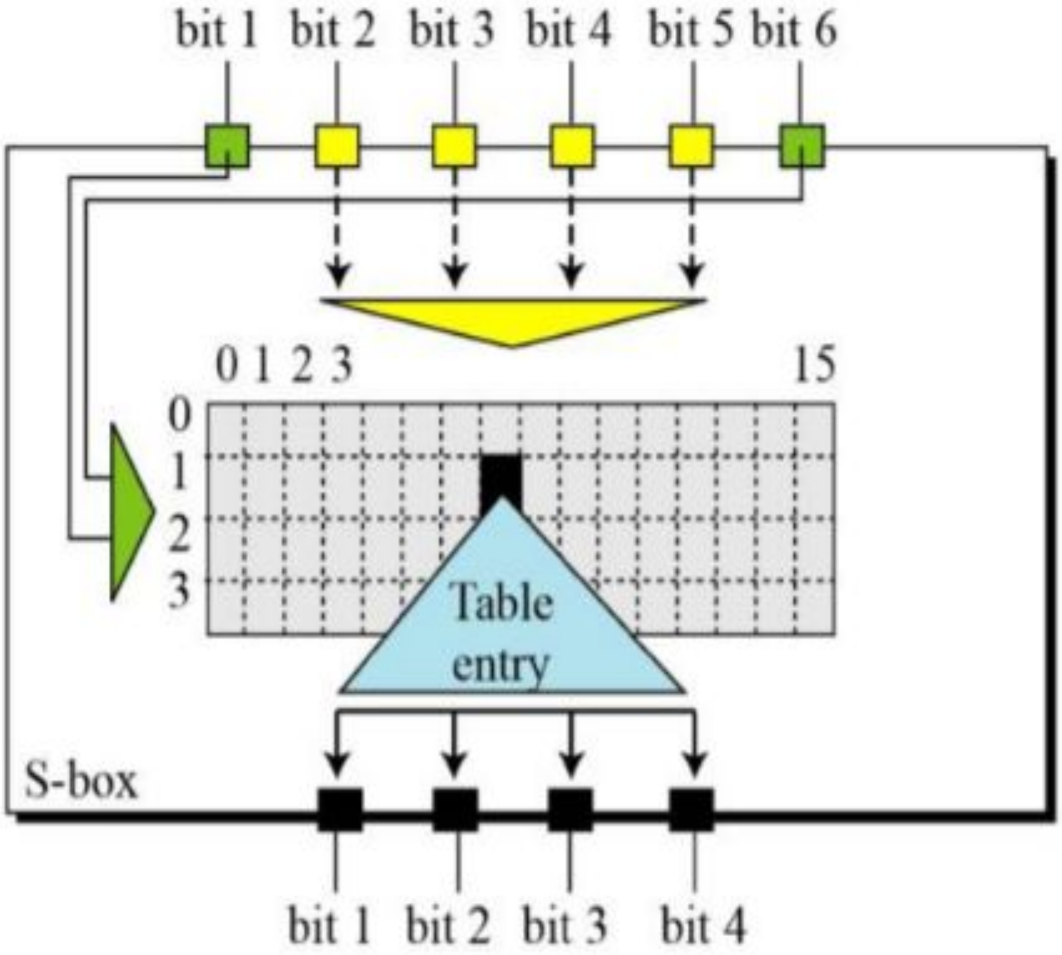
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES Round Structure

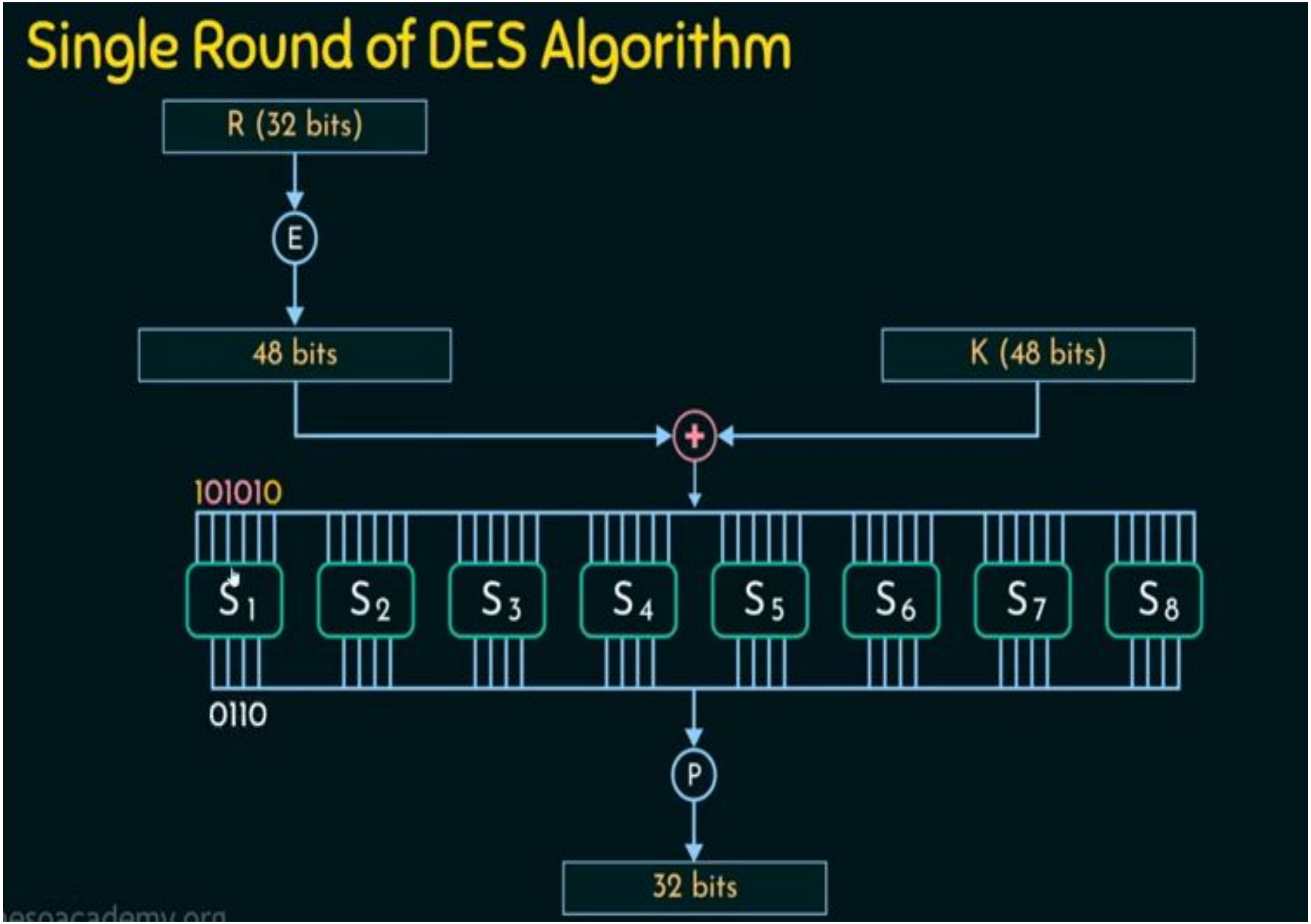


Substitution Boxes S

S-box rule



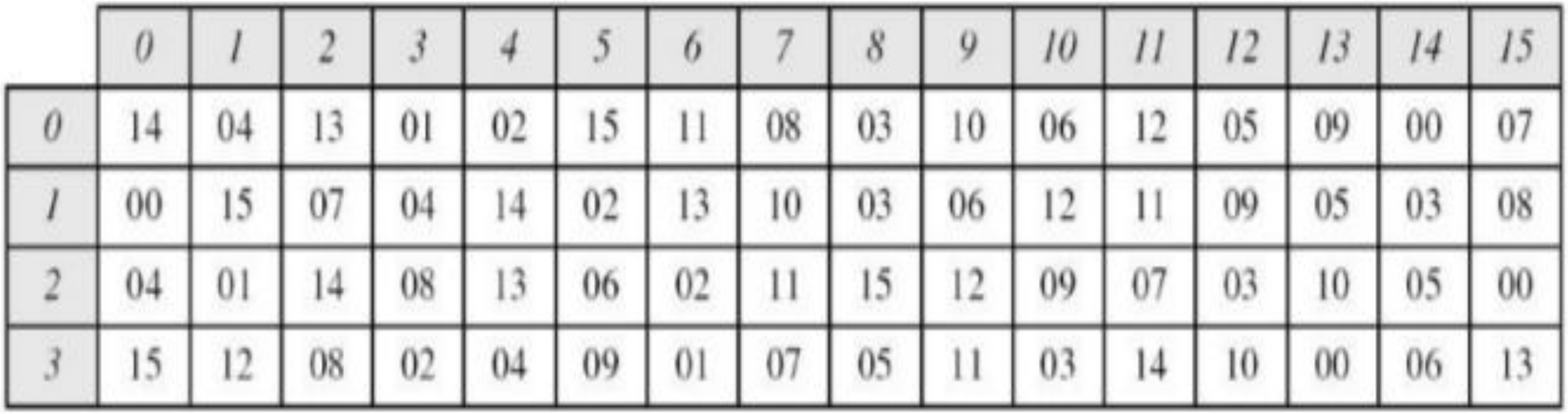
bit 1
bit 2
bit 3
bit 4
bit 5
bit 6
0 1 2 3
15
0
1
2
3
Table entry
S-box
bit 1
bit 2
bit 3
bit 4



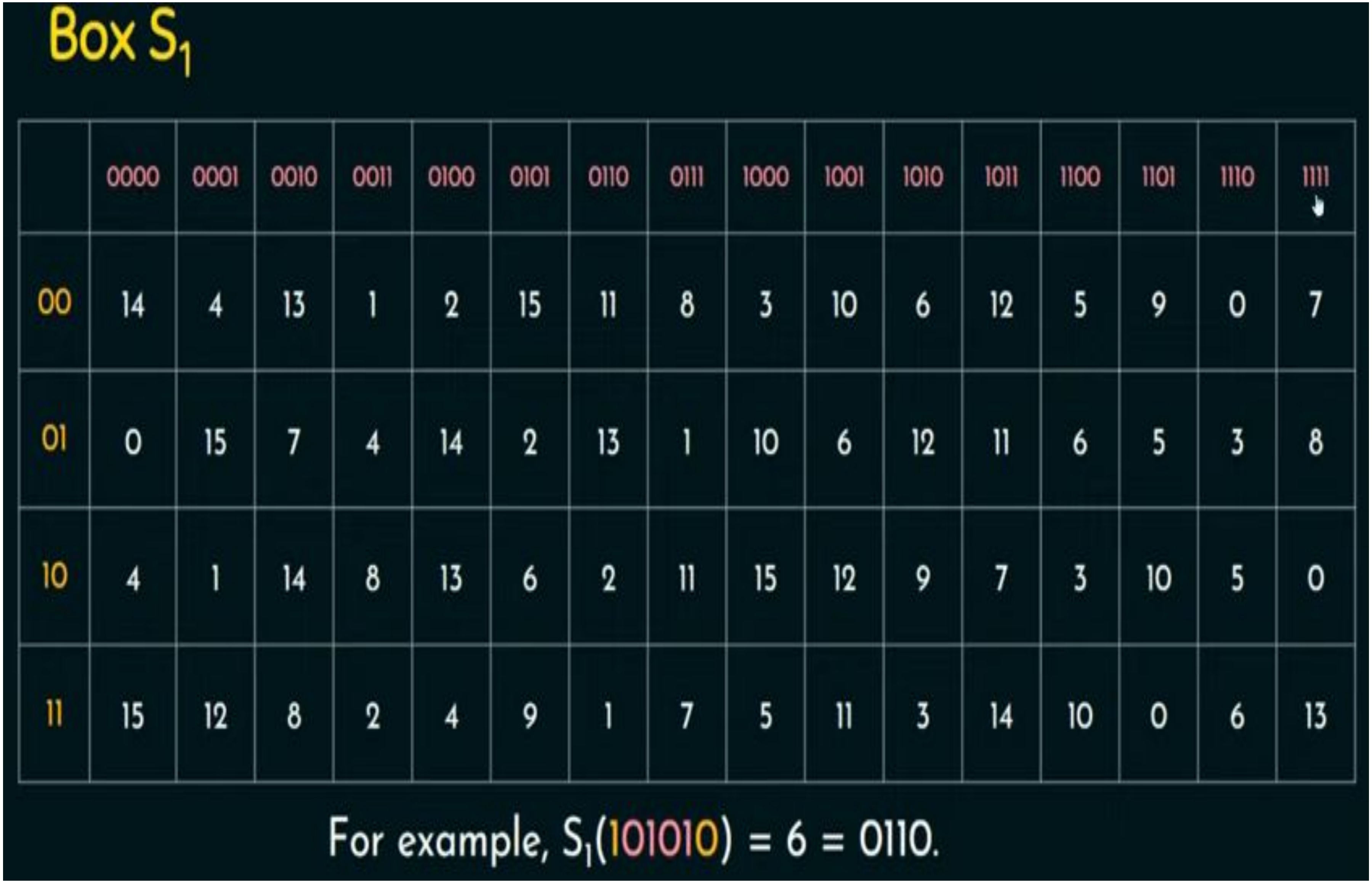
Single Round of DES Algorithm

Table shows the permutation for S-box 1. For the rest of the boxes see the textbook.

S-box 1



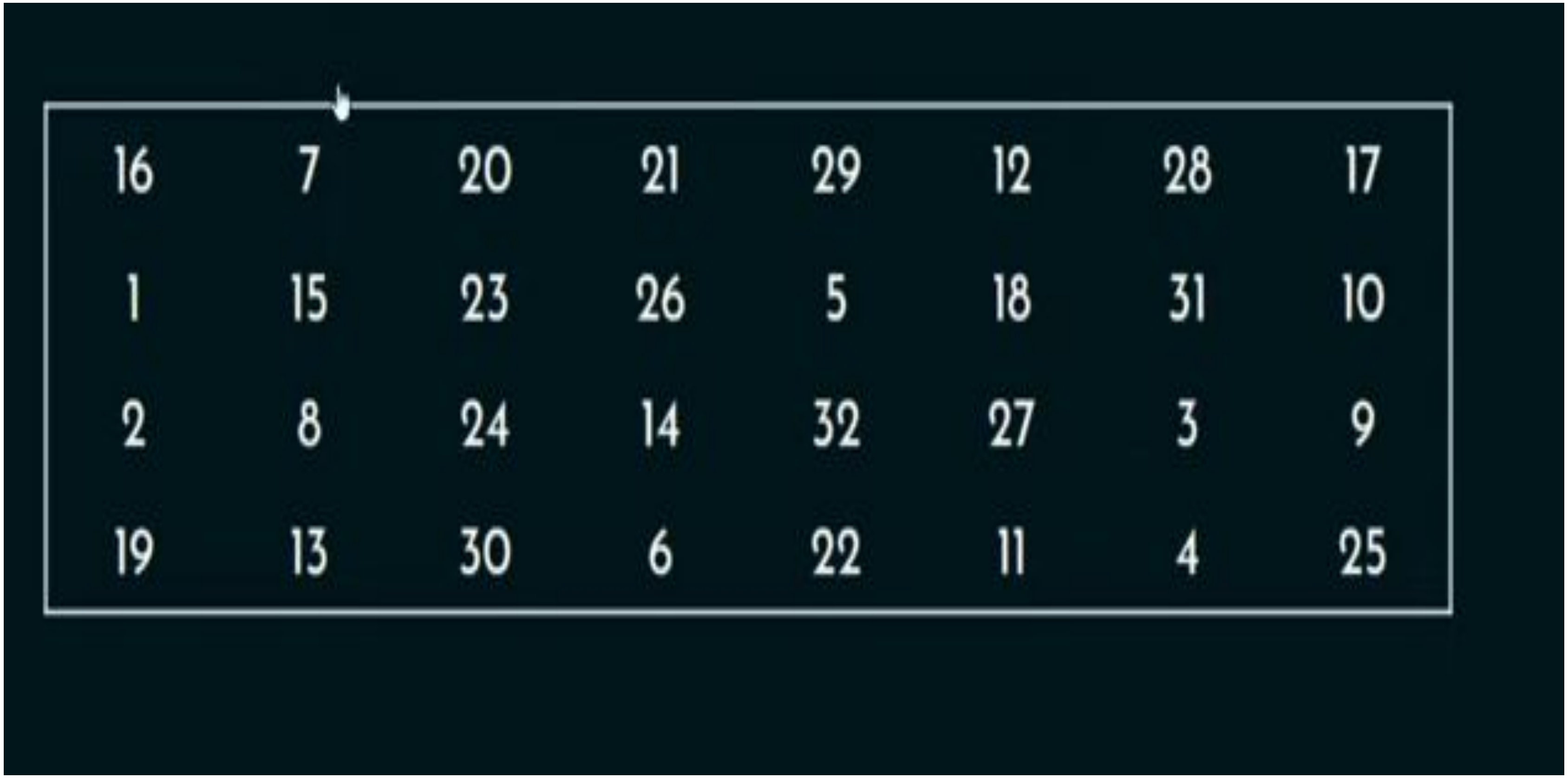
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13



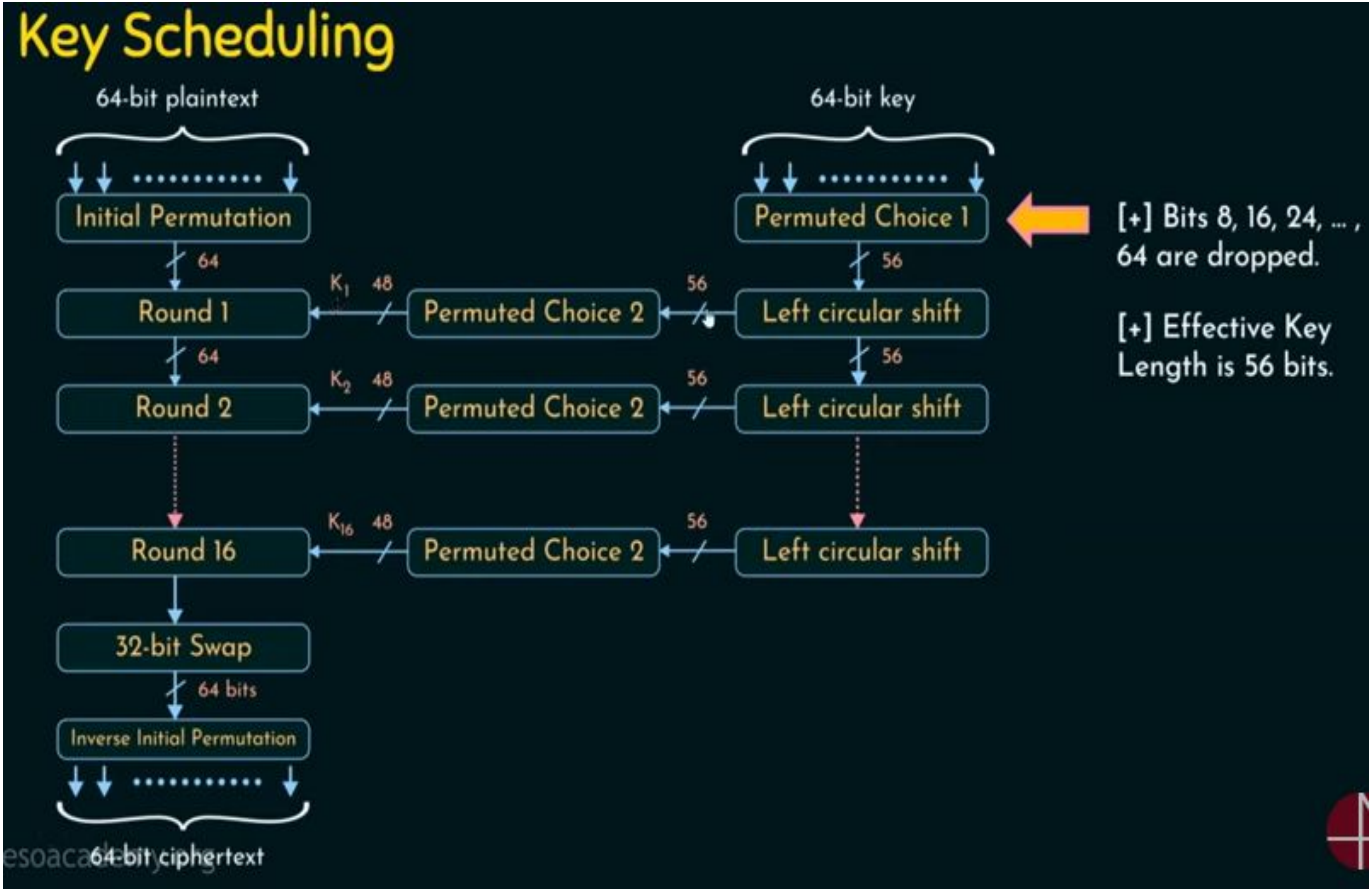
Box S_1

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	6	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

For example, $S_1(101010) = 6 = 0110$.



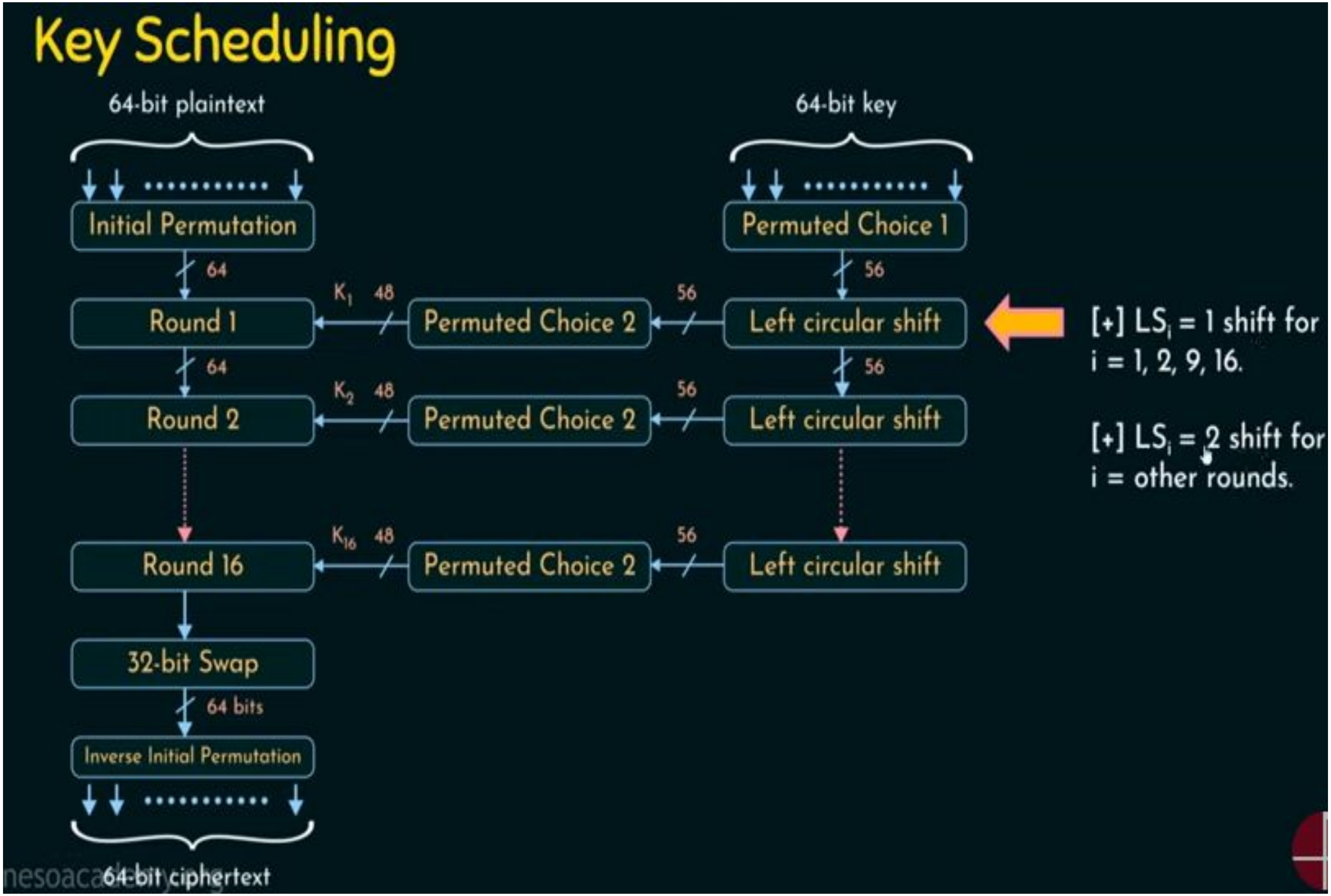
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25



Key Scheduling

[+] Bits 8, 16, 24, ... , 64 are dropped.

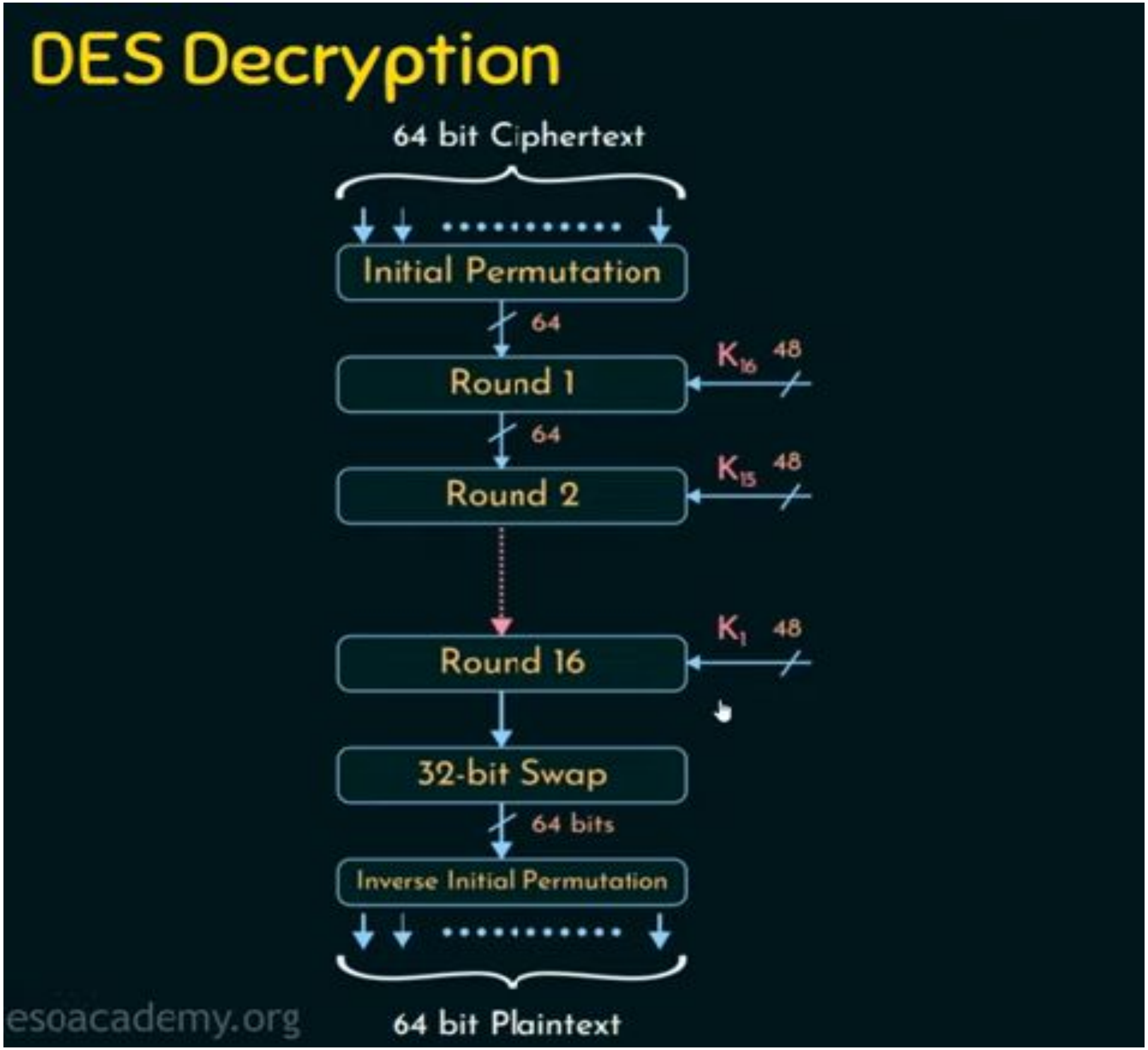
[+] Effective Key Length is 56 bits.



Key Scheduling

[+] LS_i = 1 shift for i = 1, 2, 9, 16.

[+] LS_i = 2 shift for i = other rounds.



DES Decryption

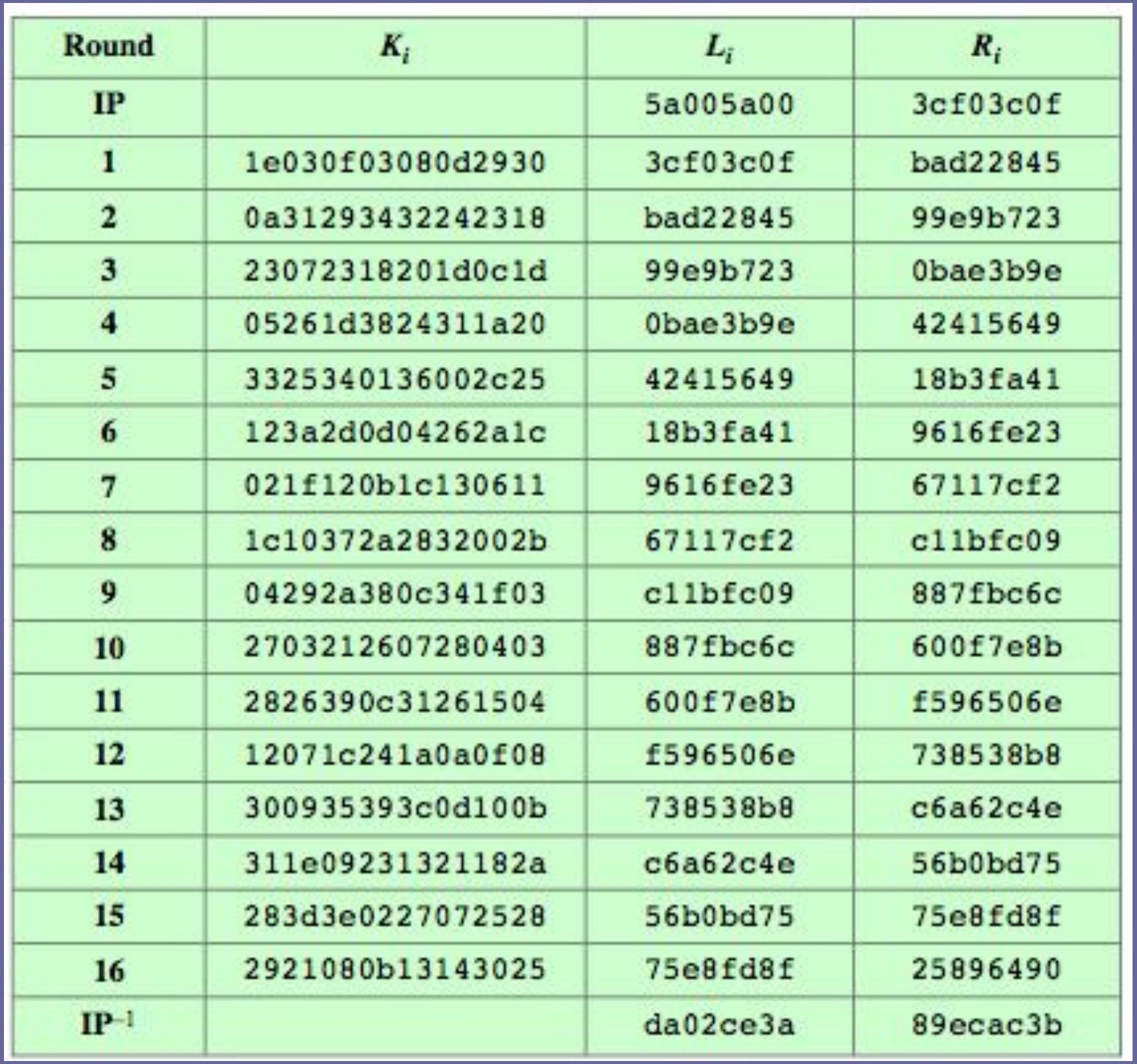
DES Key Schedule

- forms subkeys used in each round
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of:
    - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K
    - selecting 24-bits from each half & permuting them by PC2 for use in round function F
- note practical use issues in h/w vs s/w

DES Decryption

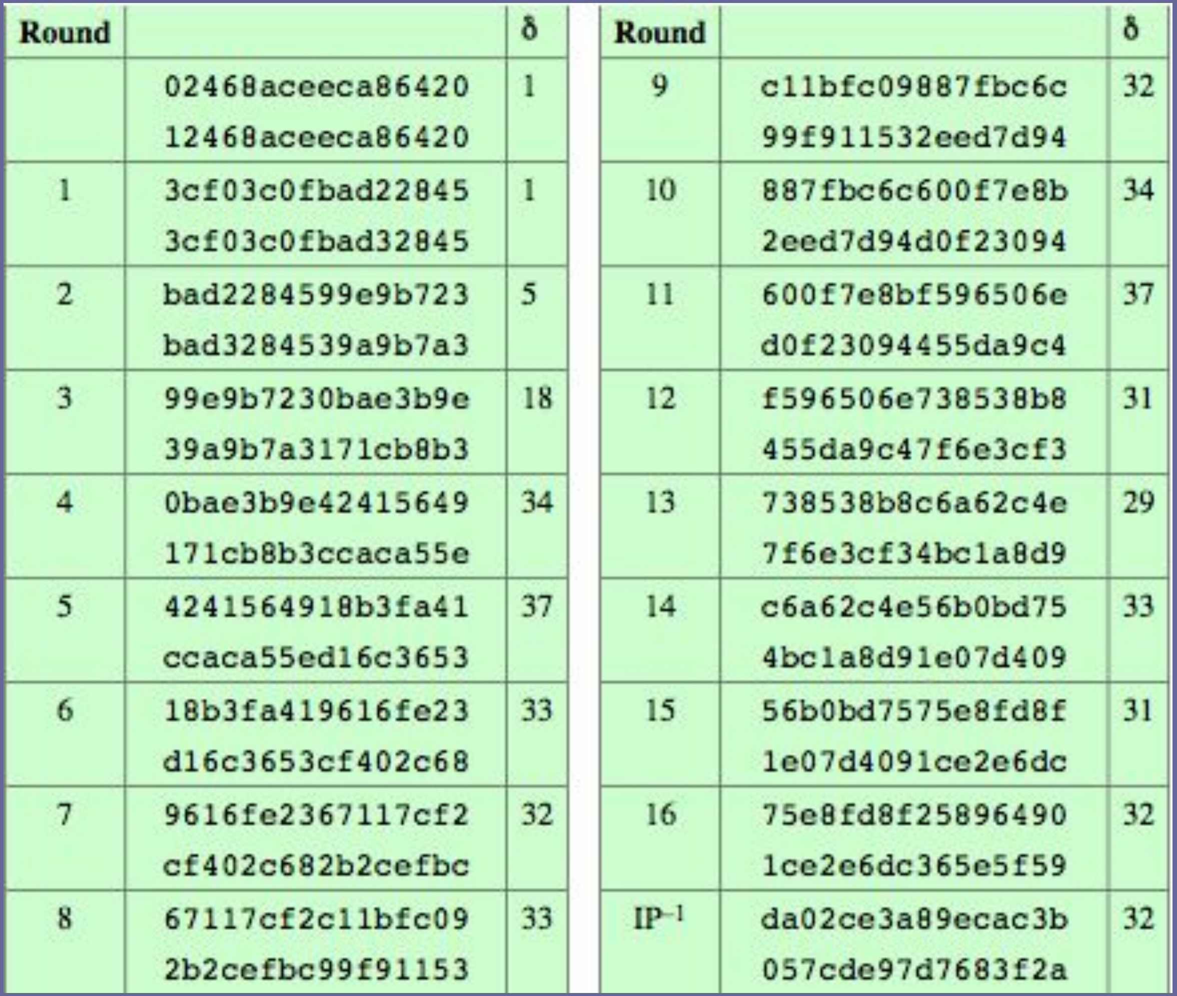
- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again using subkeys in reverse order (SK16 … SK1)
  - IP undoes final FP step of encryption
  - 1st round with SK16 undoes 16th encrypt round
  - ….
  - 16th round with SK1 undoes 1st encrypt round
  - then final FP undoes initial encryption IP
  - thus recovering original data value

DES Example



Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Avalanche in DES



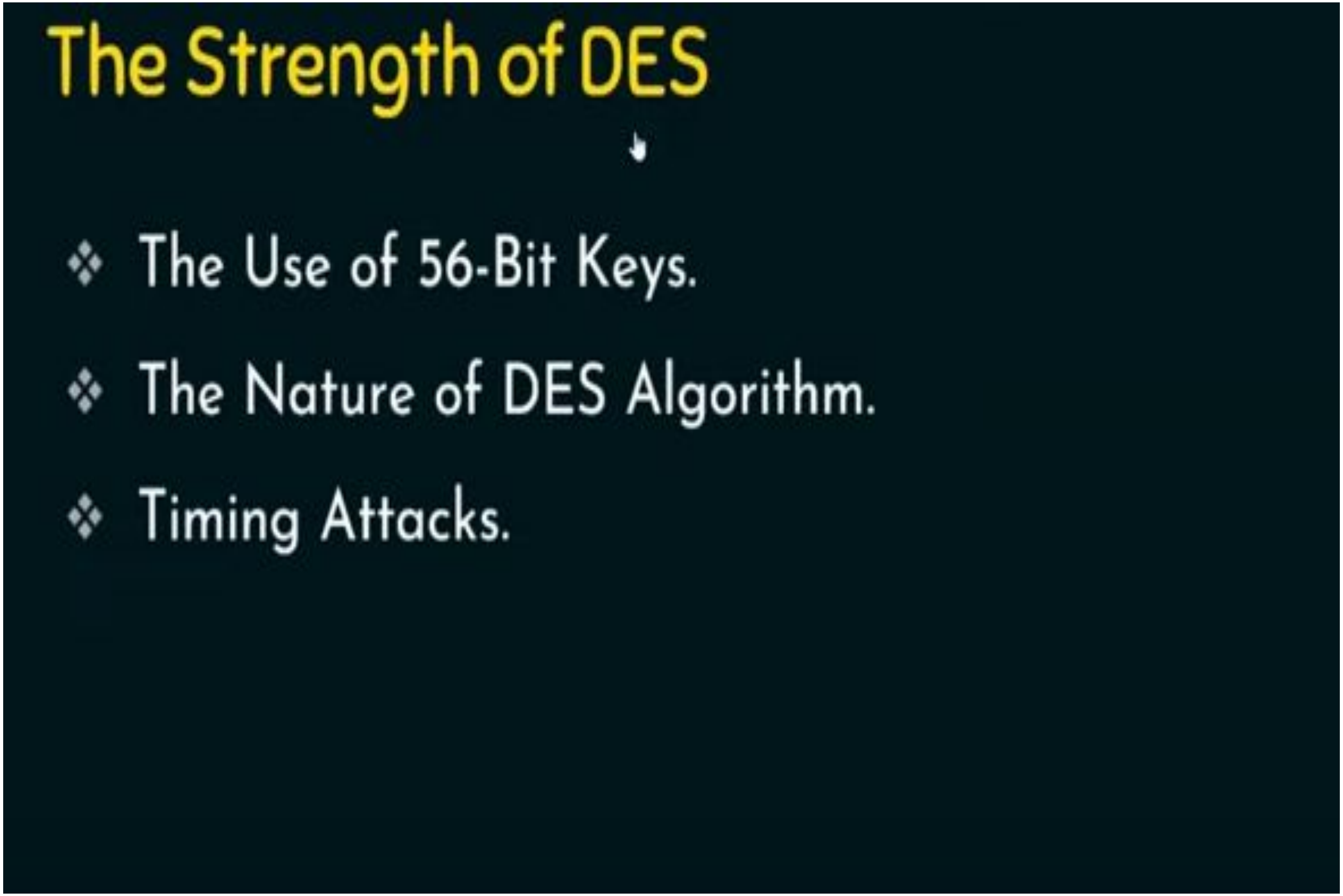
Round		δ	Round		δ
	02468aceeca86420 12468aceeca86420	1	9	c11bfc09887fbc6c 99f911532eed7d94	32
1	3cf03c0fbad22845 3cf03c0fbad32845	1	10	887fbc6c600f7e8b 2eed7d94d0f23094	34
2	bad2284599e9b723 bad3284539a9b7a3	5	11	600f7e8bf596506e d0f23094455da9c4	37
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18	12	f596506e738538b8 455da9c47f6e3cf3	31
4	0bae3b9e42415649 171cb8b3ccaca55e	34	13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
5	4241564918b3fa41 ccaca55ed16c3653	37	14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
6	18b3fa419616fe23 d16c3653cf402c68	33	15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
7	9616fe2367117cf2 cf402c682b2cefbc	32	16	75e8fd8f25896490 1ce2e6dc365e5f59	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33	IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Avalanche Effect

- key desirable property of encryption alg
- where a change of **one** input or key bit results in changing approx **half** output bits
- making attempts to “home-in” by guessing keys impossible
- DES exhibits strong avalanche

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible
  - in 1997 on Internet in a few months
  - in 1998 on dedicated h/w (EFF) in a few days
  - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- must now consider alternatives to DES



The Strength of DES

- The Use of 56-Bit Keys.
- The Nature of DES Algorithm.
- Timing Attacks.

Strength of DES – Analytic Attacks

- now have several analytic attacks on DES
- these utilise some deep structure of the cipher
    - by gathering information about encryptions
    - can eventually recover some/all of the sub-key bits
    - if necessary then exhaustively search for the rest
- generally these are statistical attacks
    - differential cryptanalysis
    - linear cryptanalysis
    - related key attacks

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive information about some/all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

DES Design Criteria

- as reported by Coppersmith in [COPP94]
- 7 criteria for S-boxes provide for
  - non-linearity
  - resistance to differential cryptanalysis
  - good confusion
- 3 criteria for permutation P provide for
  - increased diffusion

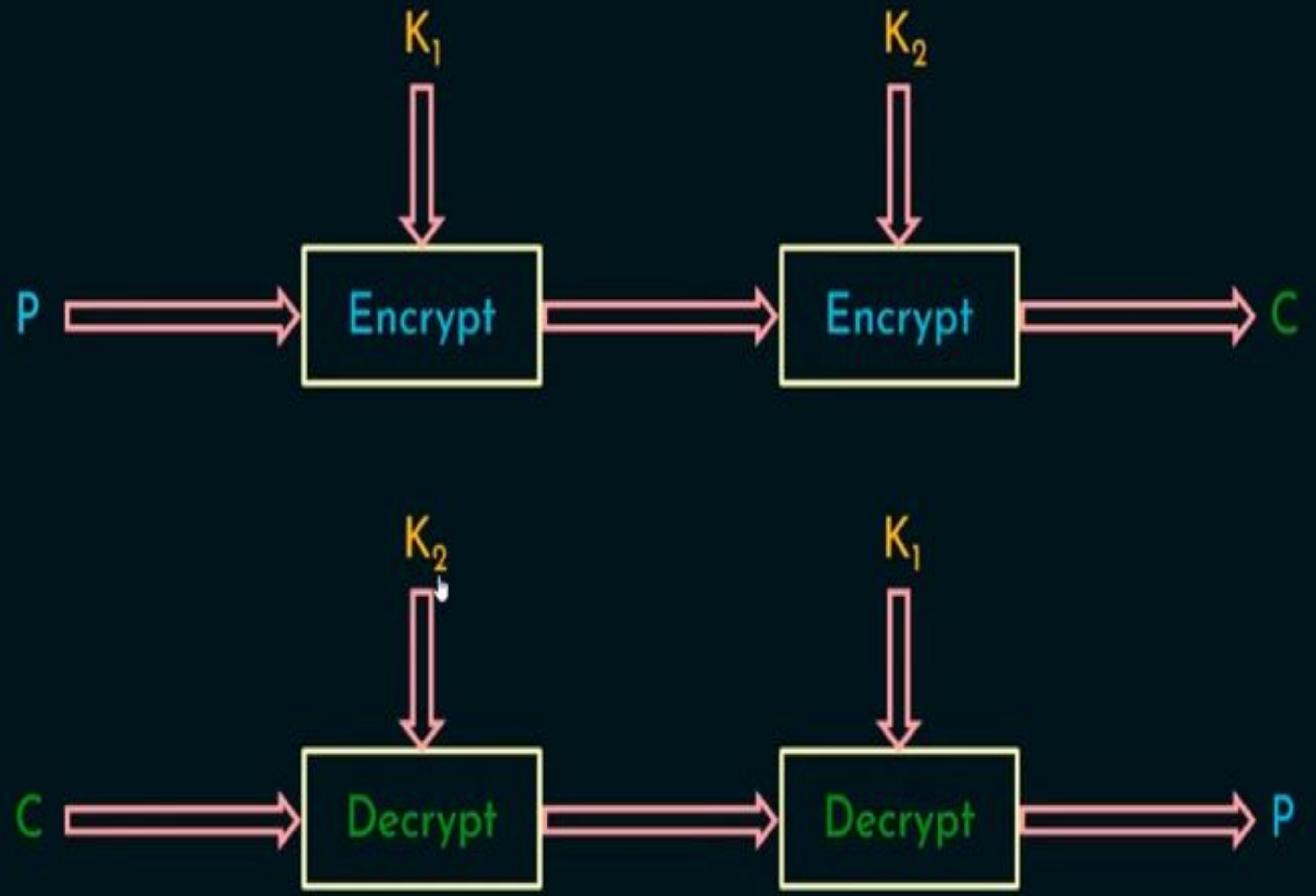
Block Cipher Design

- basic principles still like Feistel’s in 1970’s
- number of rounds
  - more is better, exhaustive search best attack
- function f:
  - provides “confusion”, is nonlinear, avalanche
  - have issues of how S-boxes are selected
- key schedule
  - complex subkey creation, key avalanche

Multiple Encryption & DES

Double-DES?

Double DES

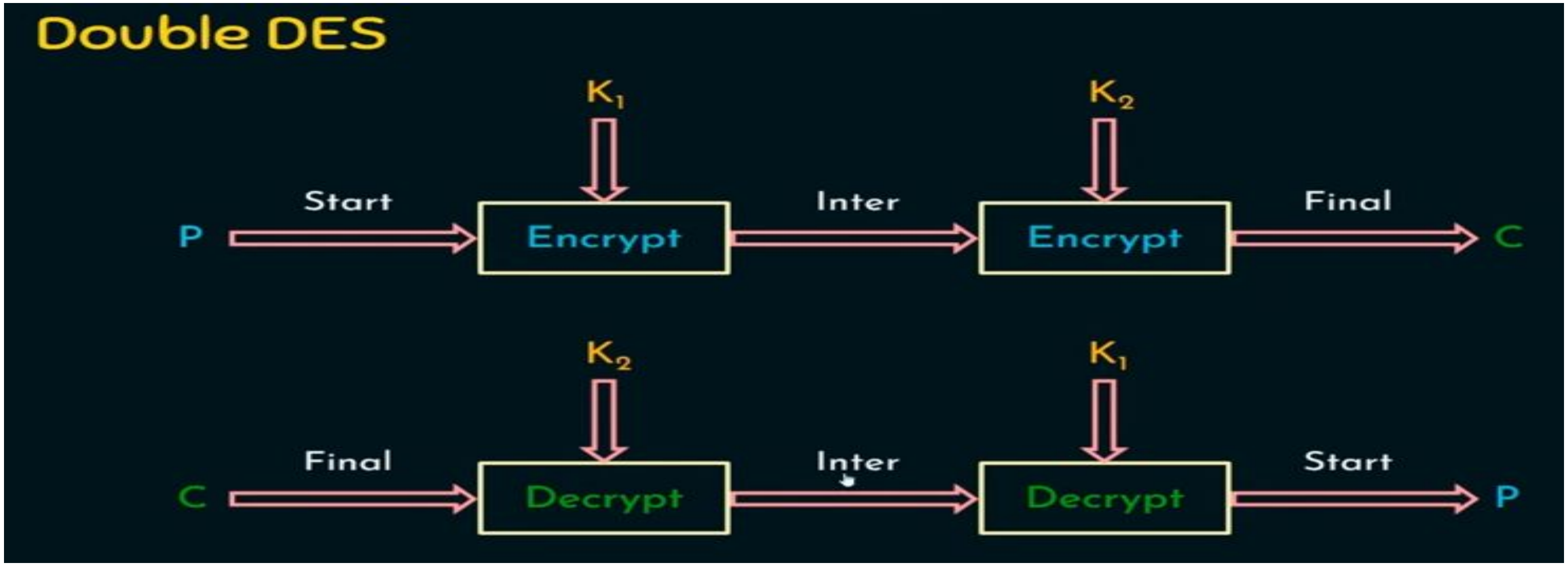


K_1 K_2

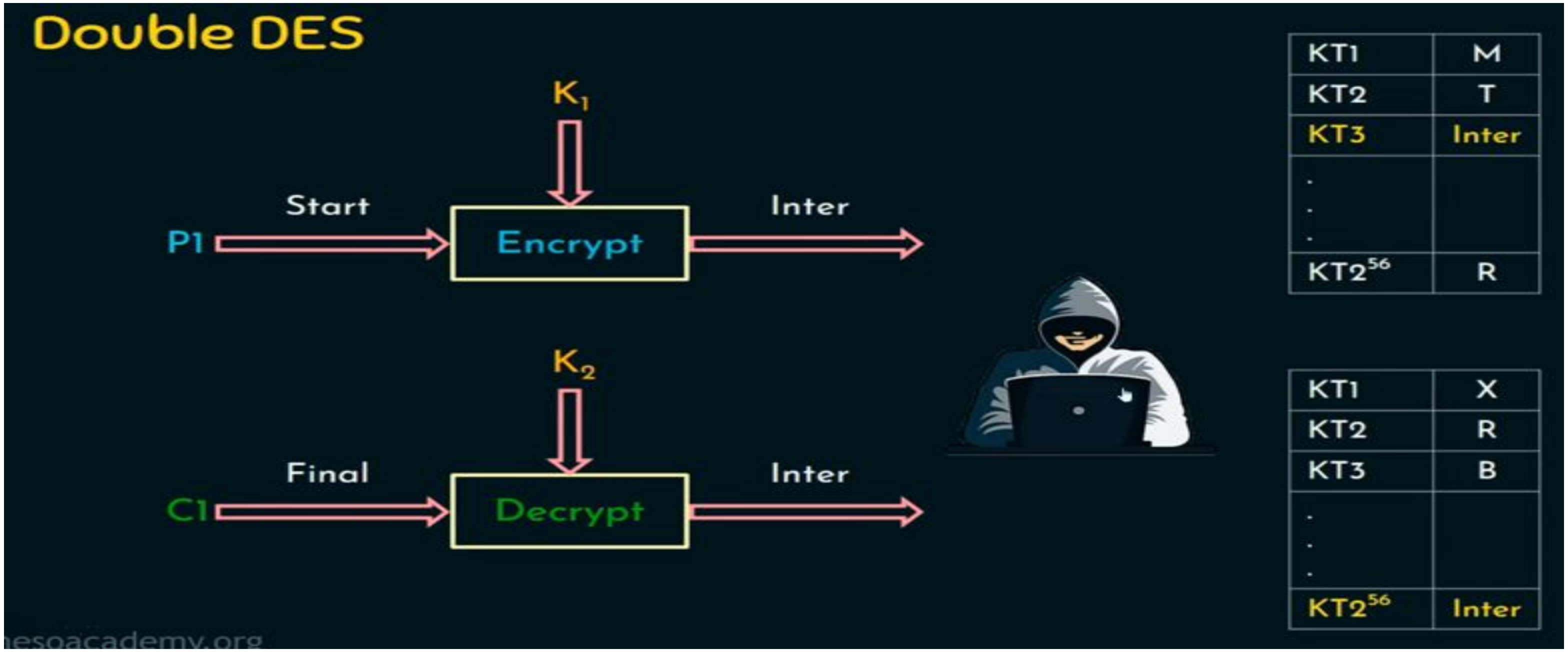
P → Encrypt → Encrypt → C

K_2 K_1

C → Decrypt → Decrypt → P



Double DES



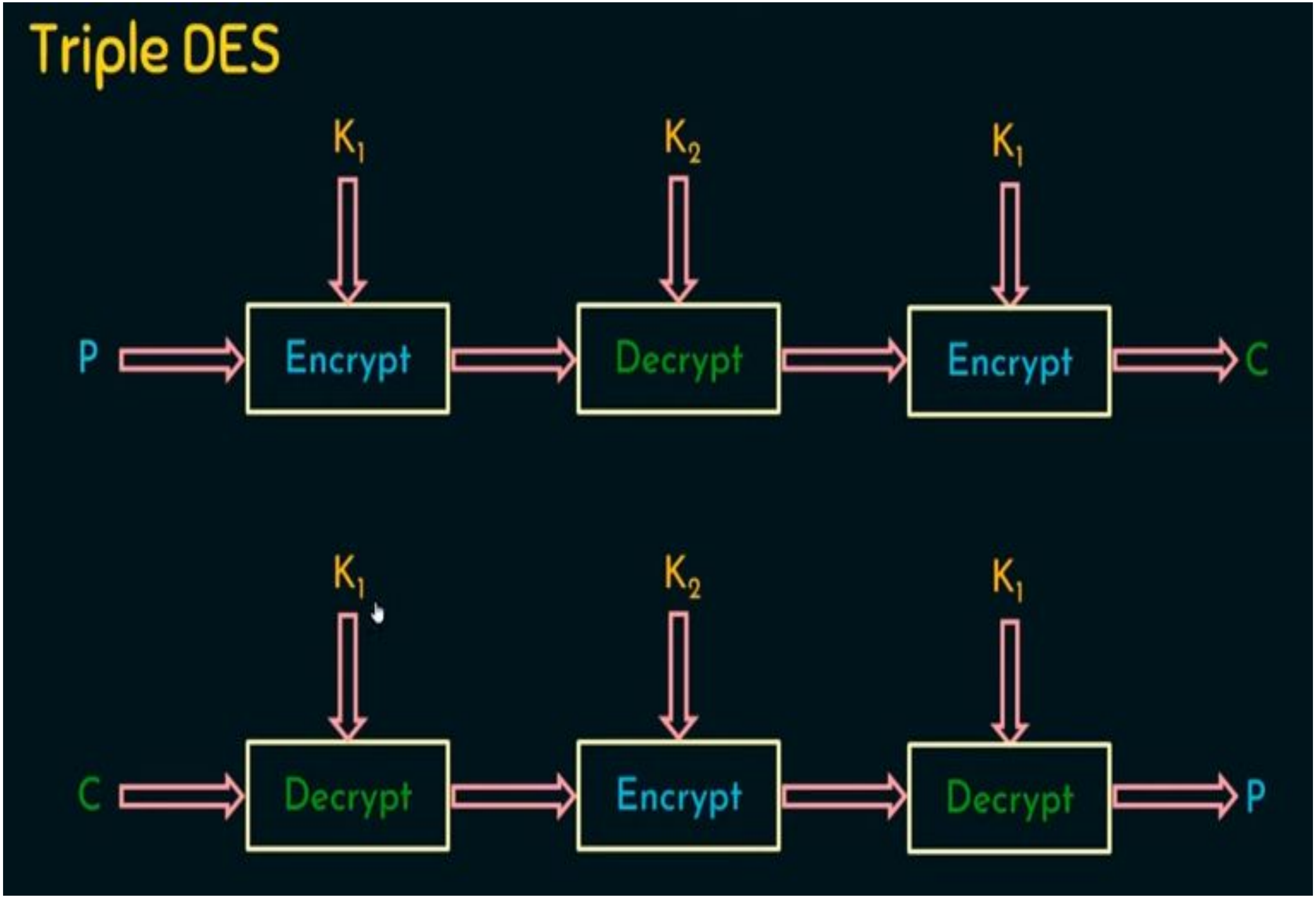
Double DES

KT1	M
KT2	T
KT3	Inter
. . .	
$KT2^{56}$	R

KT1	X
KT2	R
KT3	B
. . .	
$KT2^{56}$	Inter

Triple-DES with Two-Keys

- hence must use 3 encryptions
  - would seem to need 3 distinct keys
- but can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - nb encrypt & decrypt equivalent in security
  - if K1=K2 then can work with single DES
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - several proposed impractical attacks might become basis of future attacks



Triple DES

K1
K2
K1
P
Encrypt
Decrypt
Encrypt
C
K1
K2
K1
C
Decrypt
Encrypt
Decrypt
P

Triple-DES with Three-Keys

- although are no practical attacks on two-key Triple-DES have some indications
- can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- has been adopted by some Internet applications, eg PGP, S/MIME

RSA Algorithm:

- RSA stands for **Rivest–Shamir–Adleman** (inventors, 1977).
- A public-key cryptosystem for:
    - **Data encryption**
    - **Digital signatures**
- Based on mathematical difficulty of factoring large numbers.

RSA Key Components :

- **Public Key (e, n):** Used to encrypt messages.
- **Private Key (d, n):** Used to decrypt messages.
- Both are derived from:
  - Two large prime numbers: p and q
  - n = p * q

Key Generation Steps

1. Choose two large prime numbers: p, q

2. Compute n = p * q

3. Compute Euler's totient function: $\varphi(n) = (p-1)(q-1)$

4. Choose “e” such that $1 < e < \varphi(n)$, and $\gcd(e, \varphi(n)) = 1$

5. Compute d such that $d = e^{-1} \pmod{\varphi(n)}$

   i.e. $ed(\bmod\ \varphi(n)) = 1$

6. Public key = (e, n), Private key = (d, n)

Encryption and Decryption

- **Encryption:**
  - Ciphertext $C = M^e \bmod n$; where $M<n$
- **Decryption:**
  - Message $M = C^d \bmod n$
- Only the private key holder can decrypt.

Example:

Choose primes: $p=3$, $q=11$

$n = 3 \times 11 = 33$, $\varphi(n) = (3-1)(11-1) = 20$

Choose $e = 3 \rightarrow \gcd(3, 20) = 1$

Find d such that $(d \times 3) \bmod 20 = 1 \rightarrow d = 7$

Public key: (3, 33), Private key: (7, 33)

Ciphertext C = M^e mod n ; where M<n

Encrypt M = 7: $C = 7^3 \bmod 33 = 343 \bmod 33 = 13$

Message M = C^d mod n

Decrypt C = 13: $M = 13^7 \bmod 33 = 7$

Applications of RSA

1. Secure email (PGP)
2. HTTPS/SSL certificates
3. Digital signatures
4. Secure file transfer

Strengths of RSA

1. Strong security with large key sizes
2. Supports encryption & digital signatures
3. Well-established and widely used

Diffie-Hellman Key Exchange Algorithm

- Key exchange is fundamental for secure communication.
- The challenge: Sharing a secret key over an insecure channel.
- Solution: Use mathematical techniques to agree on a shared secret i.e. **Diffie-Hellman Algorithm**
- Proposed in 1976 by **Whitfield Diffie** and **Martin Hellman**.
- Asymetric key method for **secure key exchange**.
- Allows two parties to generate a **shared secret** over a public channel.
- Not a encryption or decryption algorithm but the foundation of many modern encryption protocols.

Key Exchange Process

1. Publicly agree on a large prime number p and base g
2. Alice picks a secret a, computes A = g^a mod p
3. Bob picks a secret b, computes B = g^b mod p
4. Alice sends A to Bob, Bob sends B to Alice
5. Shared secret:
    - Alice computes: s = B^a mod p
    - Bob computes: s = A^b mod p
    - Result: Both compute the same s

Example:

Let’s choose:

- $p = 23$, $g = 5$
- Alice picks $a = 6$, computes $A = 5^6 \bmod 23 = 8$
- Bob picks $b = 15$, computes $B = 5^{15} \bmod 23 = 2$
- Shared secret:
  - Alice: $2^6 \bmod 23 = 13$
  - Bob: $8^{15} \bmod 23 = 13$

Result: Shared secret is **13**

Applications of Diffie-Hellman:

Used in:

1. TLS/SSL (for HTTPS)
2. VPN protocols (IKE in IPsec)
3. Secure Messaging (e.g., Signal Protocol)

Often combined with digital signatures for authentication