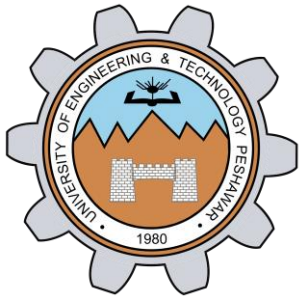
Computer Security

Lecture 6: Modern Ciphers & Data Encryption Standard (DES)

Prof. Dr. Sadeeq Jan

Department of Computer Systems Engineering

University of Engineering and Technology Peshawar



Lecture Outline

- Modern Block Ciphers
- Block Vs Stream Ciphers
- Confusion and Diffusion
- Feistal Cipher Structure
- Data Encryption Standard (DES)
- Avalanche Effect
- Strength of DES – Timing Attacks
- Mode of Operations (ECB, CBC)

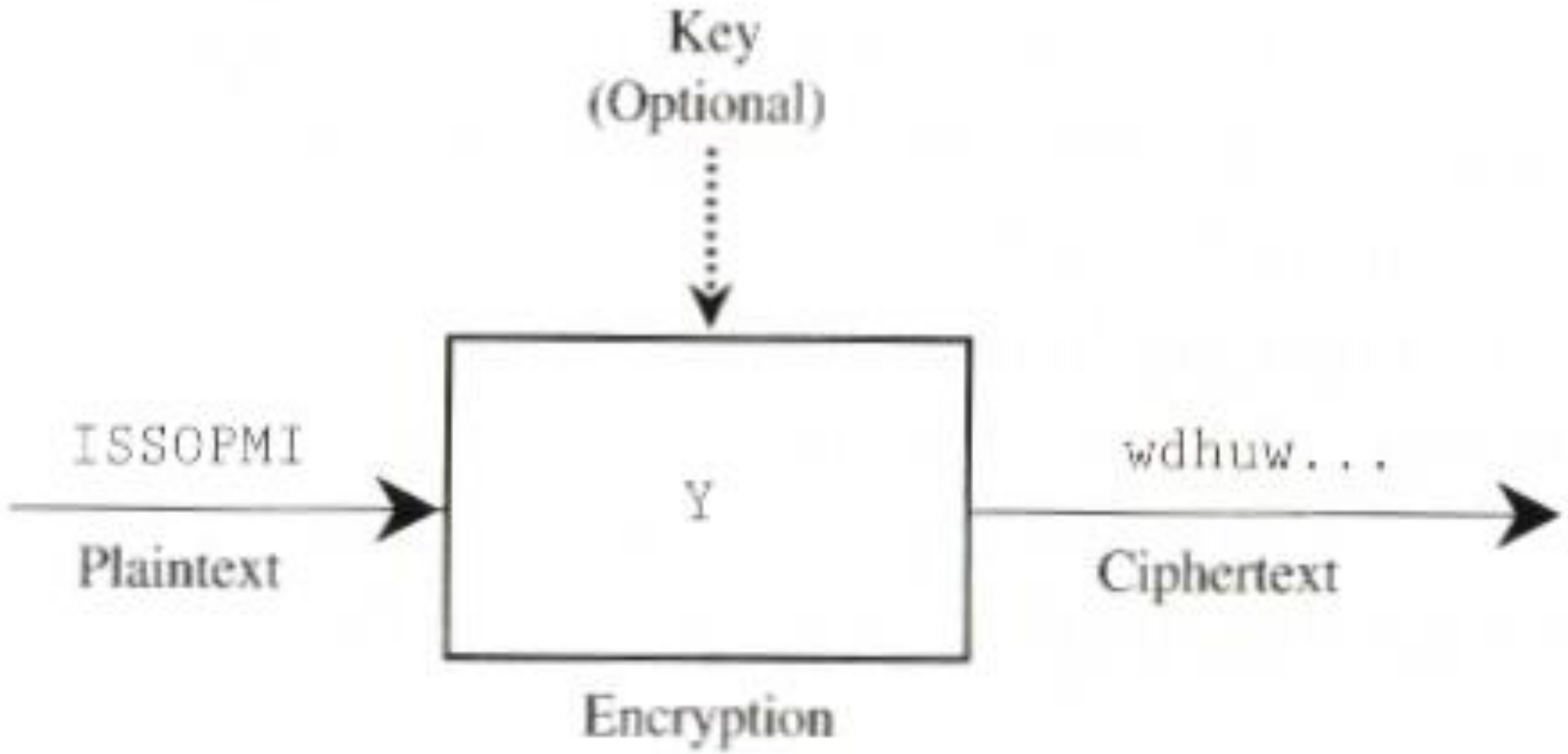
Modern Block Ciphers

- will now look at modern block ciphers
- one of the most widely used types of cryptographic algorithms
- provide secrecy and/or authentication services
- in particular will introduce DES (Data Encryption Standard)

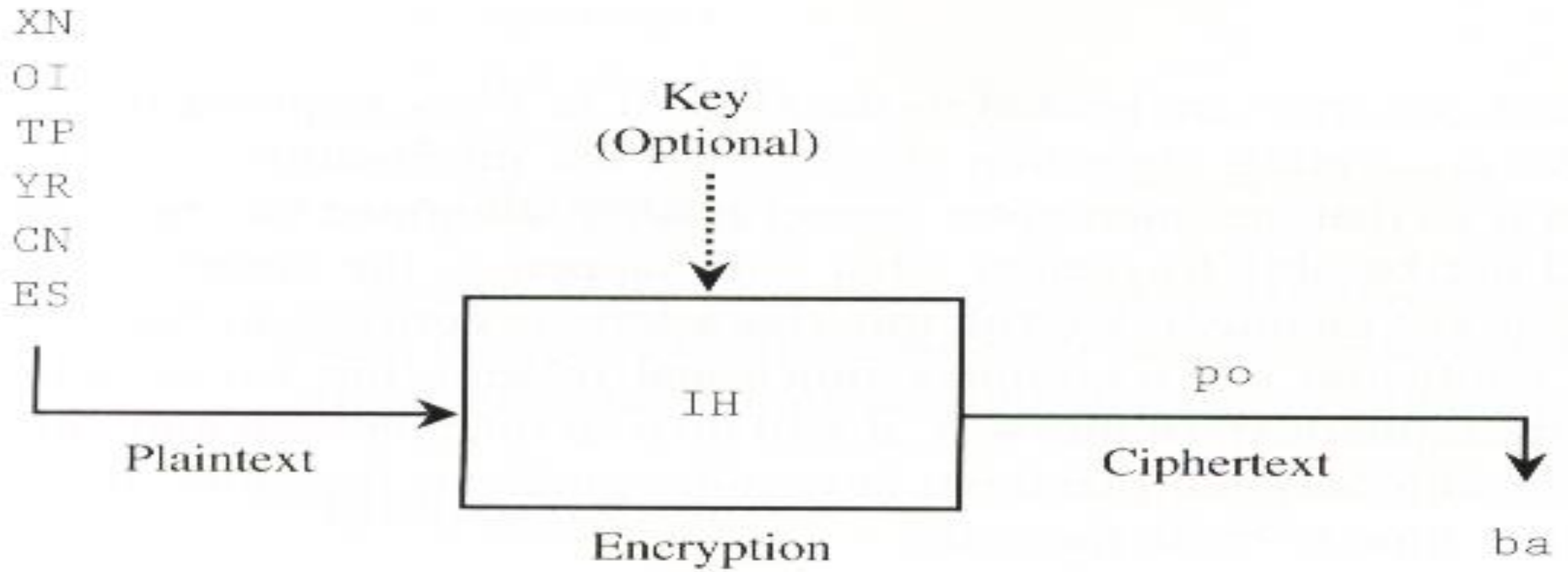
Block vs Stream Ciphers

- Block Ciphers process messages in into 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
- hence are focus of course

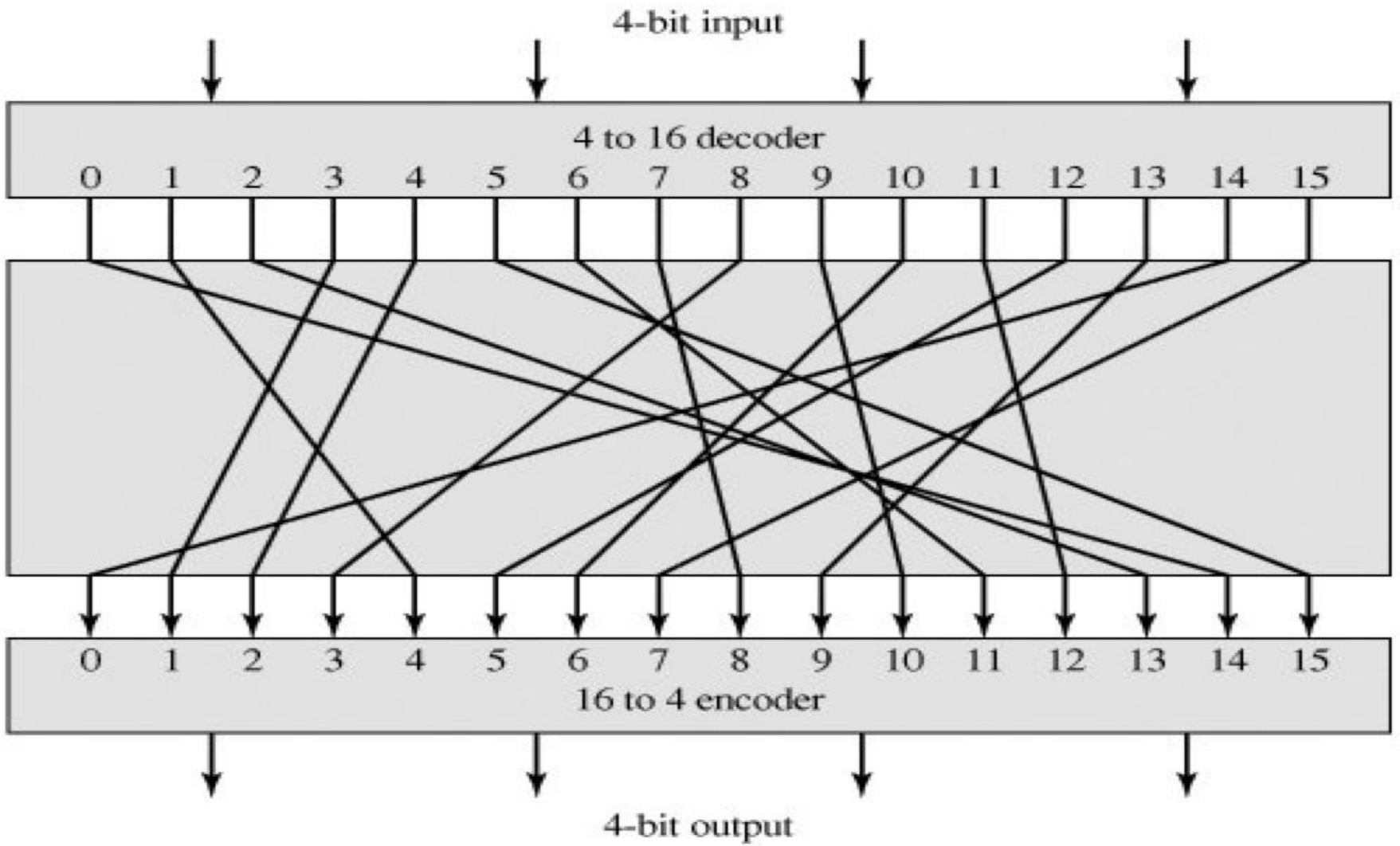
Block Cipher Systems



Block Cipher Systems



General n-bit-n-bit Block Substitution (n=4)



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

Reverse Vs. Irreversible

Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

Claude Shannon and Substitution-Permutation Ciphers

- in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks
  - modern substitution-transposition product cipher
- these form the basis of modern block ciphers
- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- provide *confusion* and *diffusion* of message

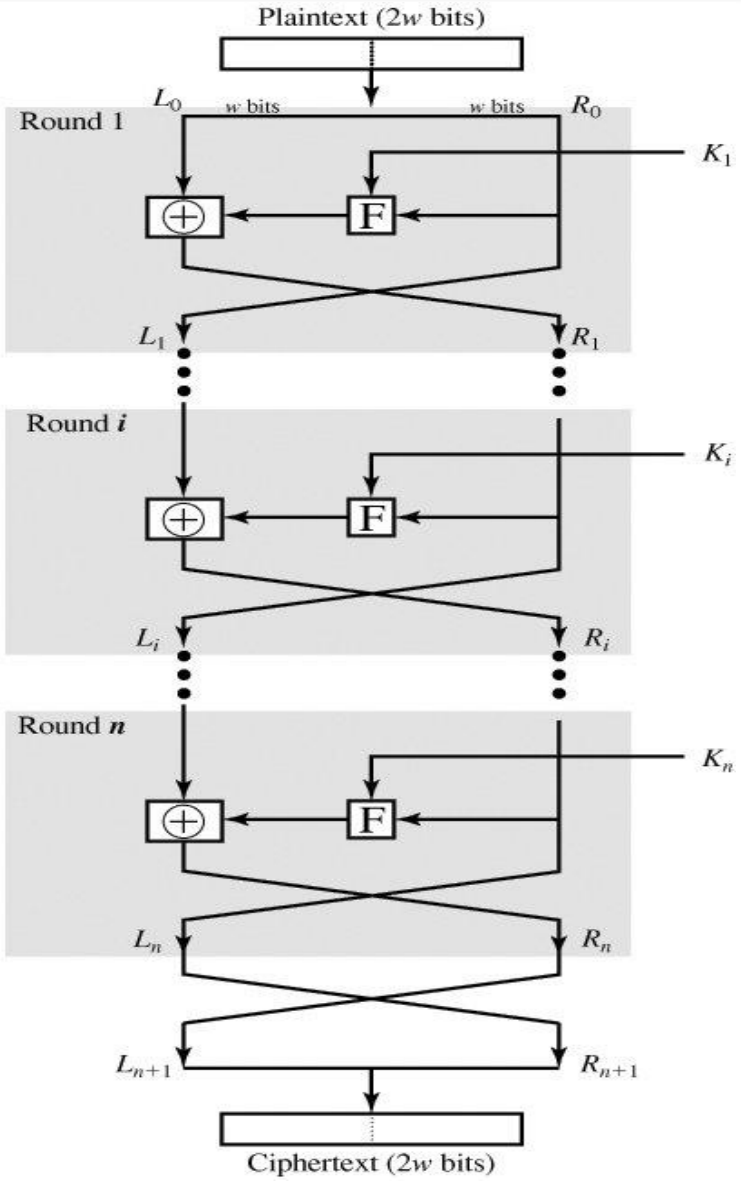
Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message
- a one-time pad does this
- more practically Shannon suggested combining 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 substitution-permutation network concept

Fiestel Cipher Structure

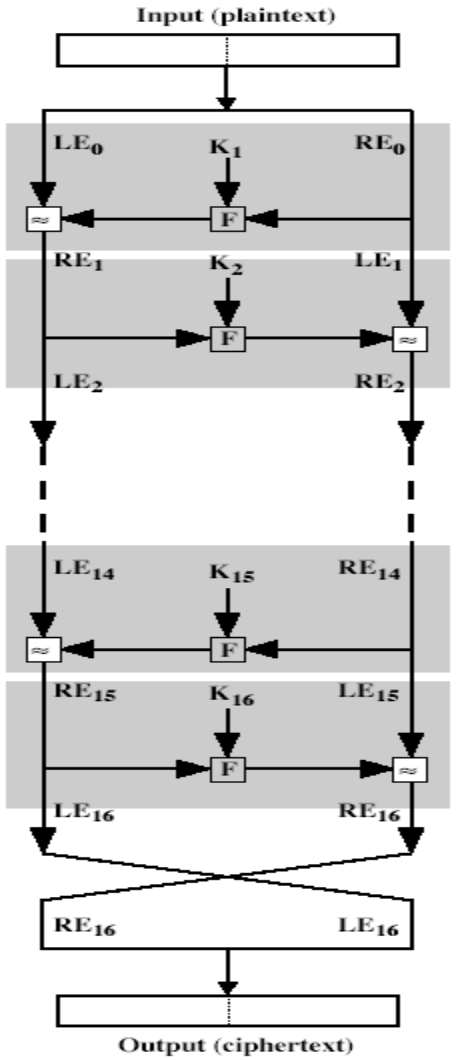


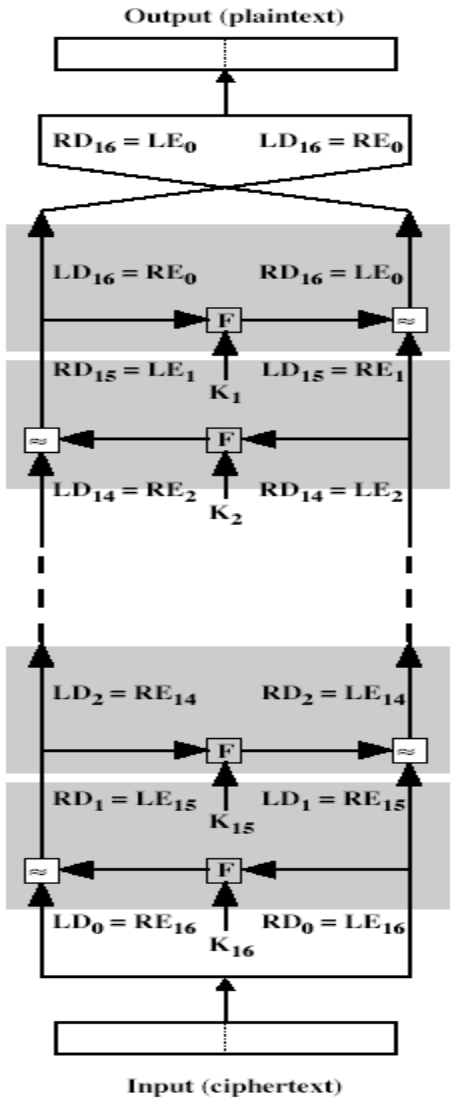
Feistel Cipher Design Principles

- **block size**
  - increasing size improves security, but slows cipher
- **key size**
  - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **number of rounds**
  - increasing number improves security, but slows cipher
- **subkey generation**
  - greater complexity can make analysis harder, but slows cipher
- **round function**
  - greater complexity can make analysis harder, but slows cipher
- **fast software en/decryption & ease of analysis**
  - are more recent concerns for practical use and testing

Data Encryption Standard (DES)

- Same process as encryption.
- Use the ciphertext as input to the algorithm, but use the subkeys *Ki* in reverse order. i.e. use *Kn* in the first round, *Kn*–1 in the second round, and so on until *K*1 is used in the last round.
- NO need to implement two different alogrithms for encryption and decryption.





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

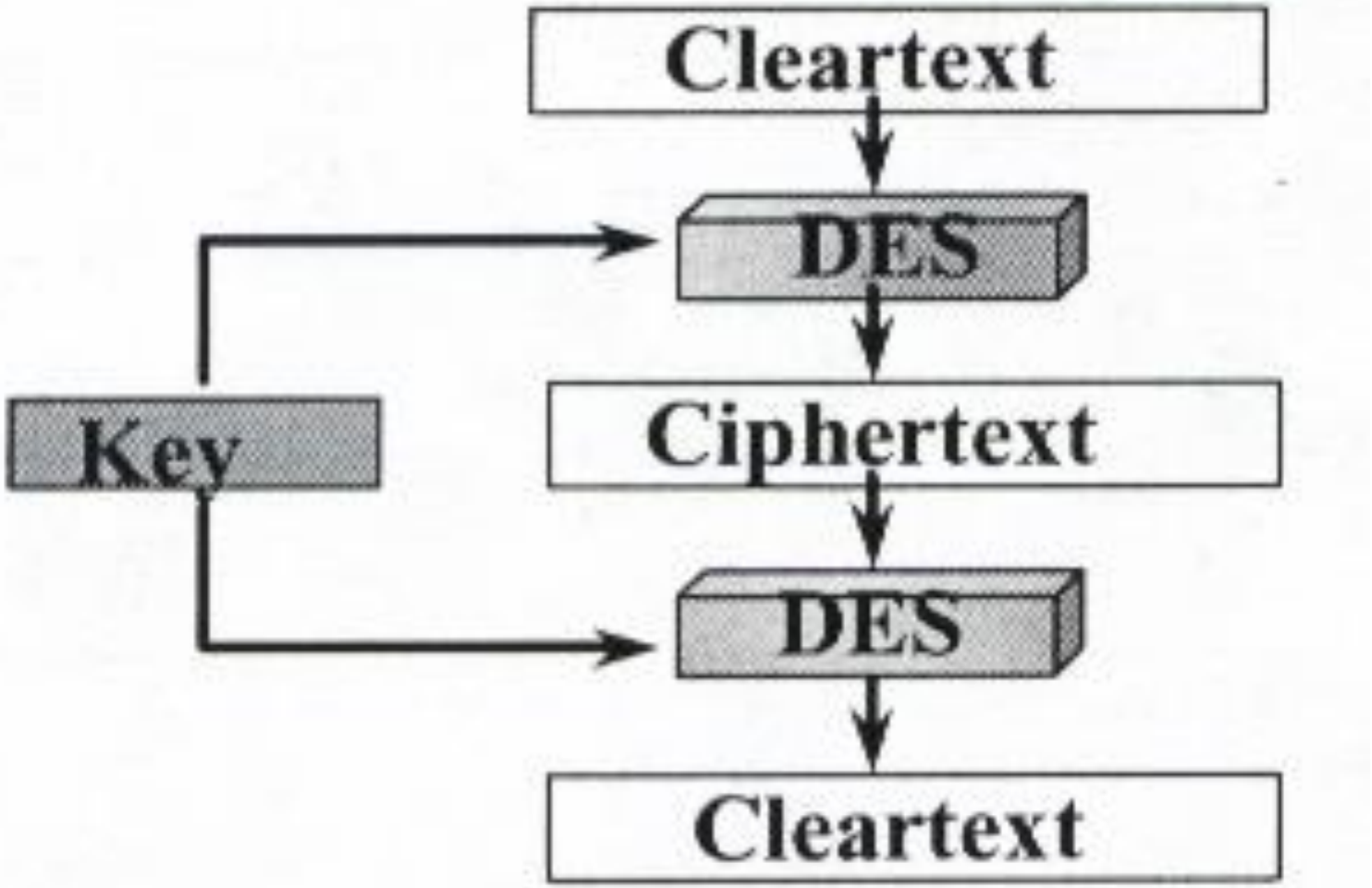
DES History

- IBM developed Lucifer cipher
  - by team led by Feistel
  - 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

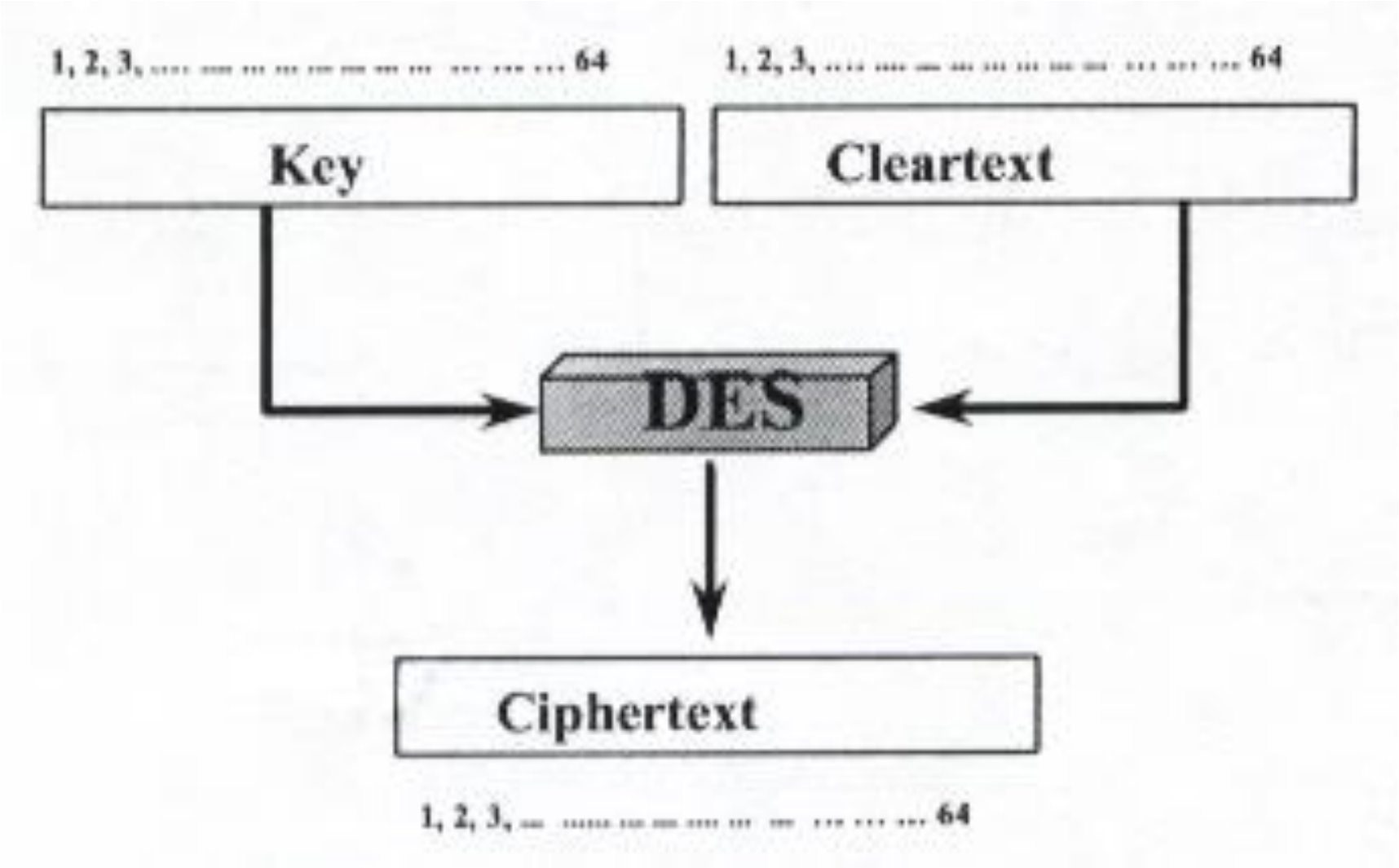
DES Design Controversy

- 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
- DES has become widely used, esp in financial applications
- 1977-1998 US Standard
- Best studied cipher in the world

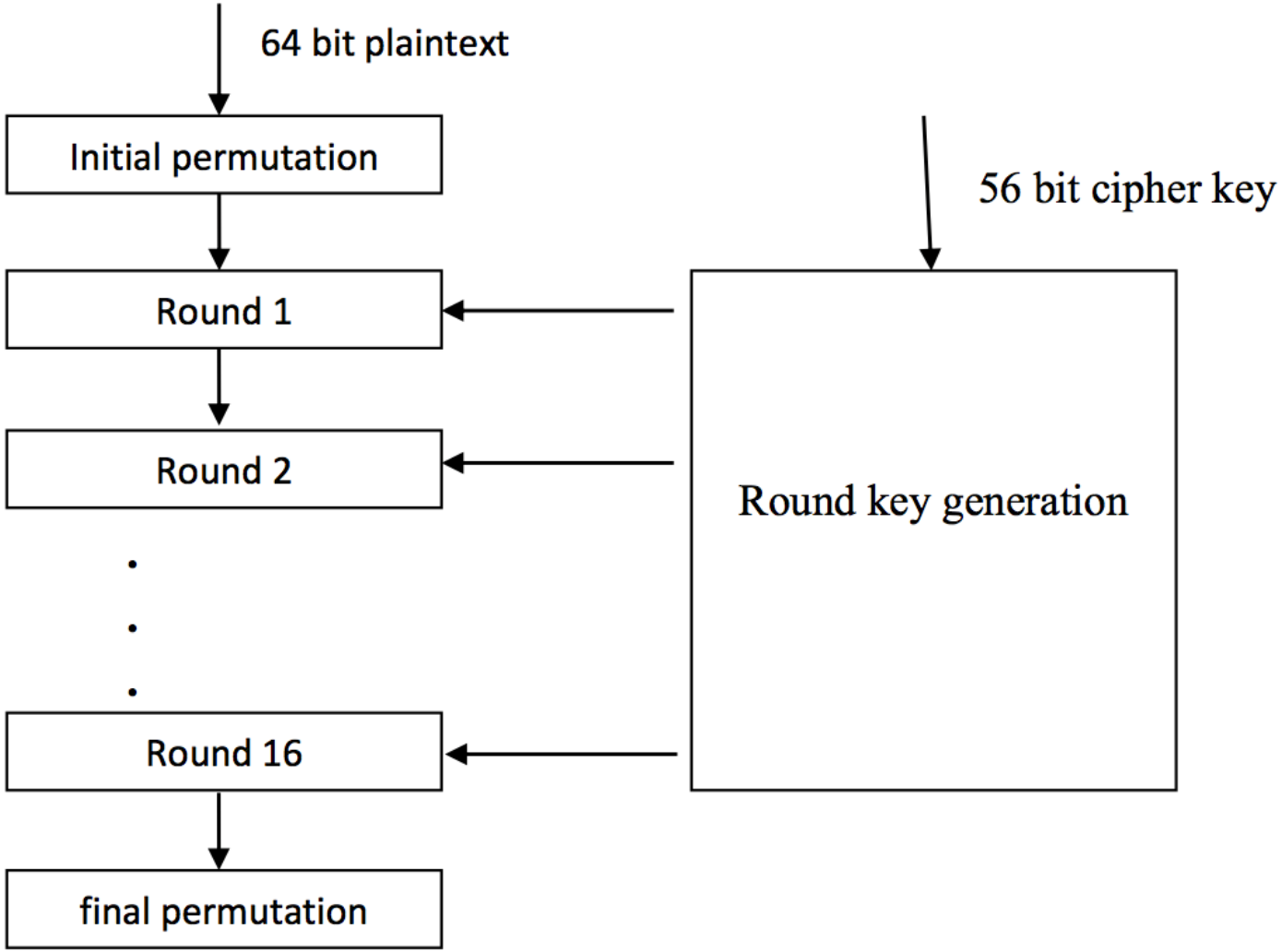
DES Algorithm



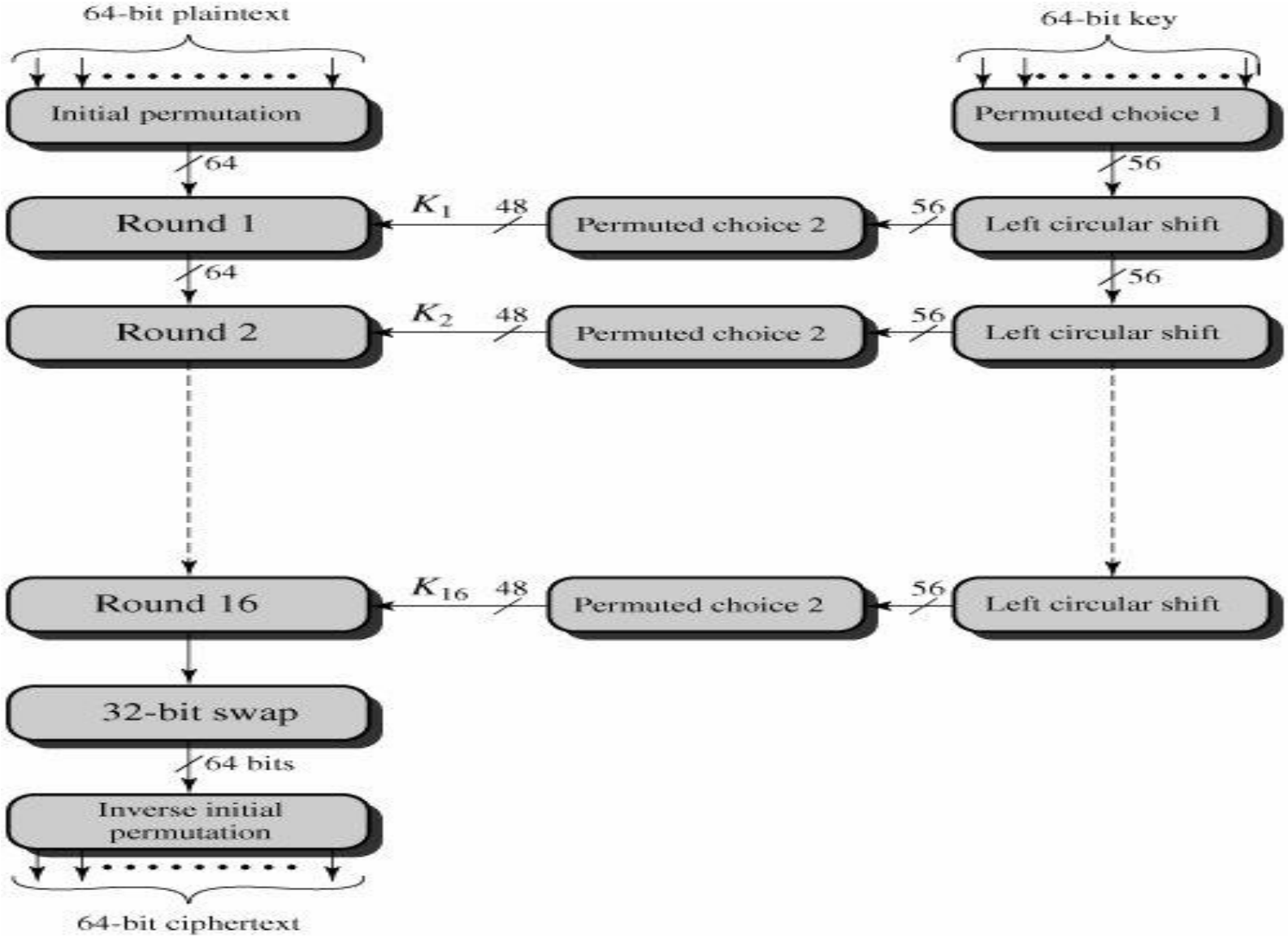
DES Properties



DES - Working Principle



DES Encryption



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)

Initial Permutation IP& Inverse Initial Permutation (IP^1)

(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

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

Expansion Permutation & Permutation Function Box

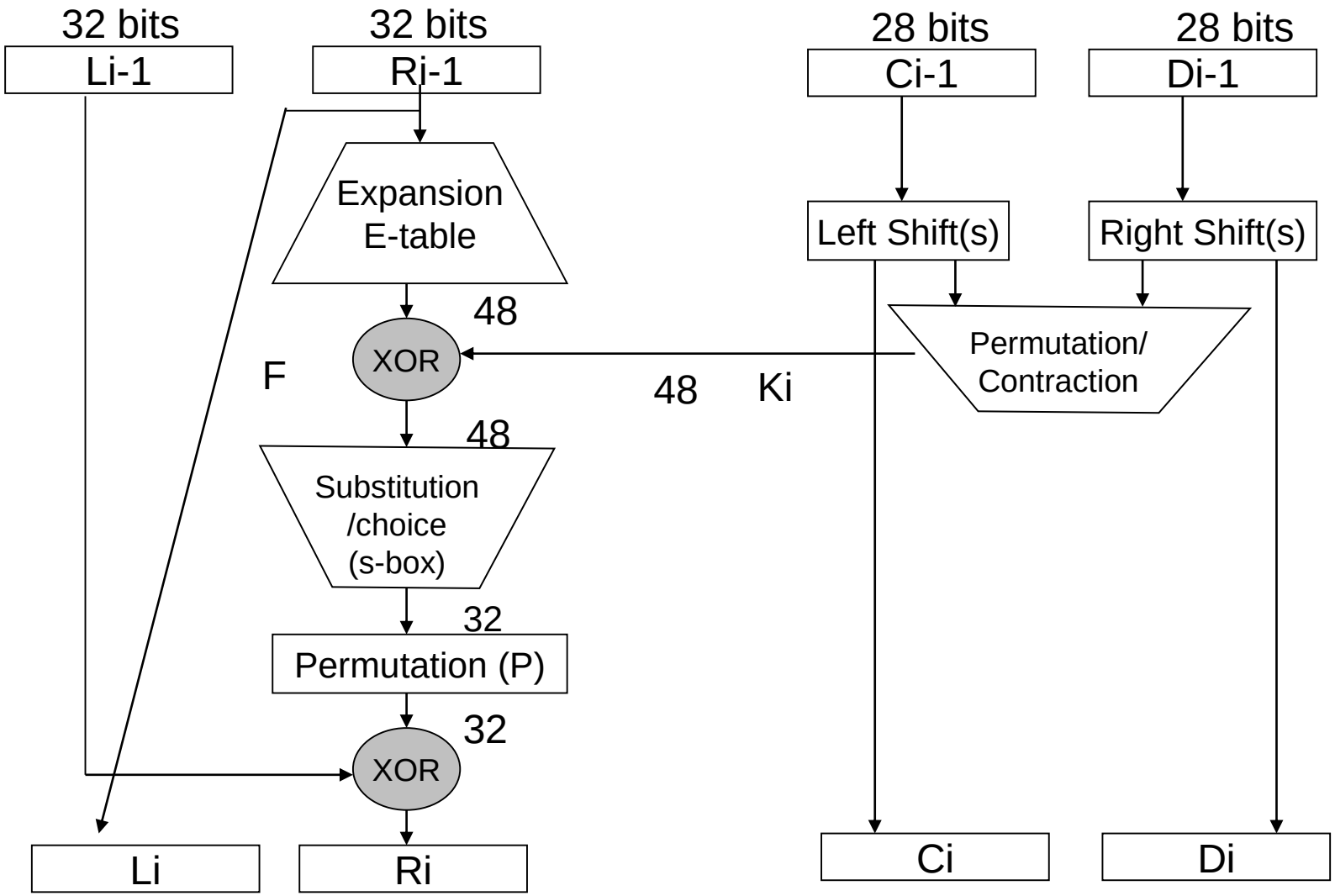
(c) Expansion Permutation (E)

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

Single Round of DES



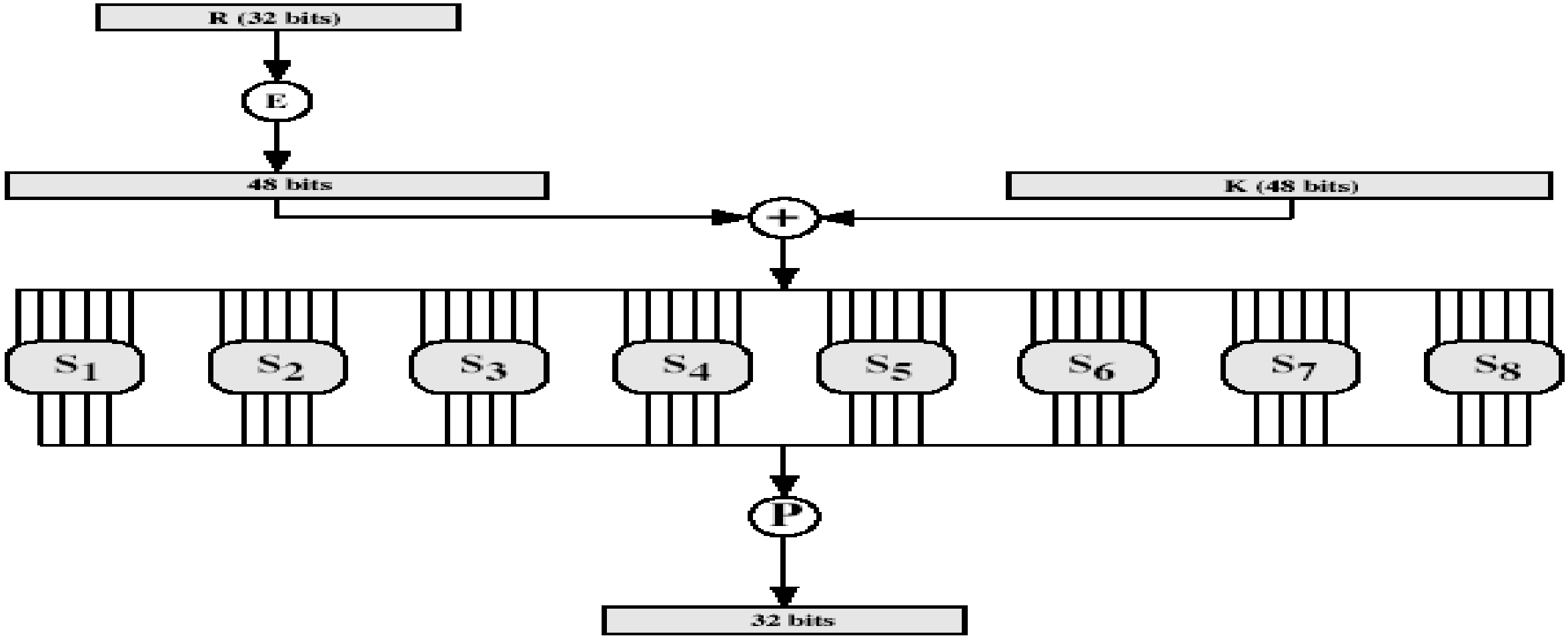
DES Round Structure

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

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \text{ xor } \mathrm{F}(R_{i-1}, K_i)$$

- takes 32-bit R half and 48-bit subkey and:
  - expands R to 48-bits using perm E
  - adds to subkey
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes this using 32-bit perm P

DES Round Structure



DES S-Boxes

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

Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
  - outer bits 1 & 6 (**row** bits) select the row number
  - inner bits 2-5 (**col** bits) selects the column number in the table (from 0-15)
  - The value at that place is a decimal number which is translated into binary (4 bit) to obtain the output of that S-box
    - For example, in S1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

DES Key Generation

- a 64-bit key is used as input to the algorithm.
- The bits of the key are numbered from 1 through 64; every eighth bit is ignored, so giving 56-bit key
- The key is first subjected to a permutation governed by a table labeled Permuted Choice One
- The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0.
- At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift, or rotation, of 1 or 2 bits. These shifted values serve as input to the next round.
- They also serve as input to Permuted Choice Two, which produces a 48-bit output that serves as input to the function $F(R_{i-1}, K_i)$.

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (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

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using subkeys in reverse order (SK16 … SK1)
- note that 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

Avalanche Effect

- key desirable property of encryption algorithm
- where a change of **one** input or key bit results in changing approx **half** output bits
- 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
- now considering alternatives to DES

DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker" machine that was built for less than $250,000. The attack took less than three days. TheEFF has published a detailed description of the machine, enabling others to build their own cracker [EFF98].

Strength of DES – Timing Attacks

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

DES Modes of Operations
&
Double/Triple DES

Modes of Operation

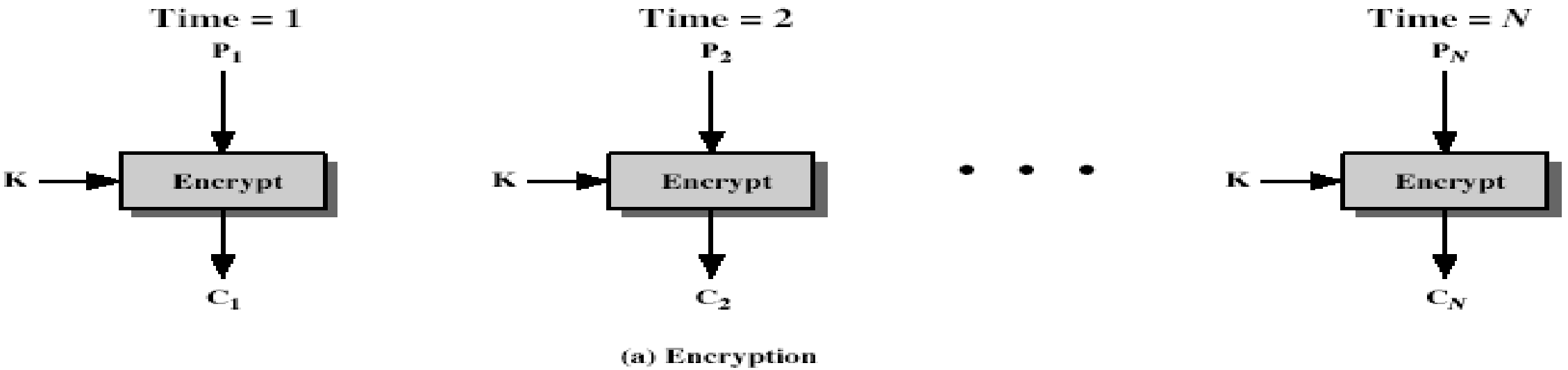
- block ciphers encrypt fixed size blocks
- eg. DES encrypts 64-bit blocks, with 56-bit key
- need way to use in practise, given usually have arbitrary amount of information to encrypt
  - 1. ECB (Electronic code book)
  - 2. CBC (Cipher block chaining)
  - 3. CFB (Cipher feedback)
  - 4. OFB (Output feedback)
  - 5. CTR (Counter method)

Self study

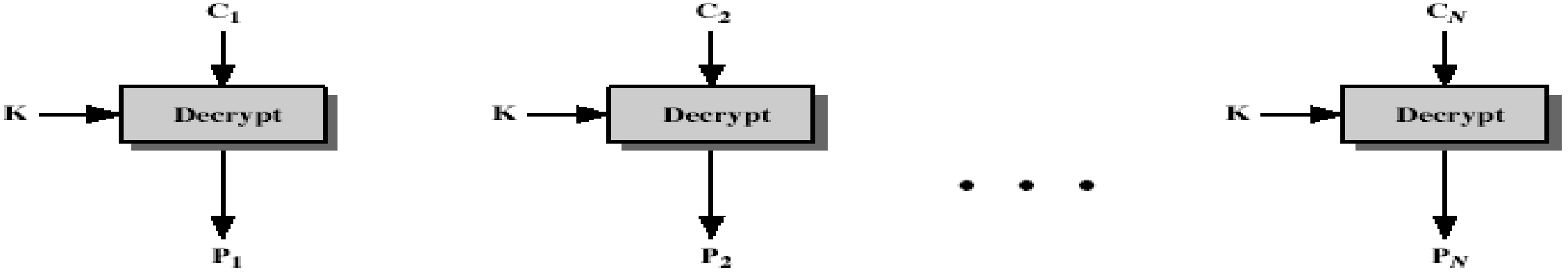
Electronic Codebook Book (ECB)

- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks
  - $C_i = DES_{K1}(P_i)$
- uses: secure transmission of single values

Electronic Codebook Book (ECB)



(a) Encryption



(b) Decryption

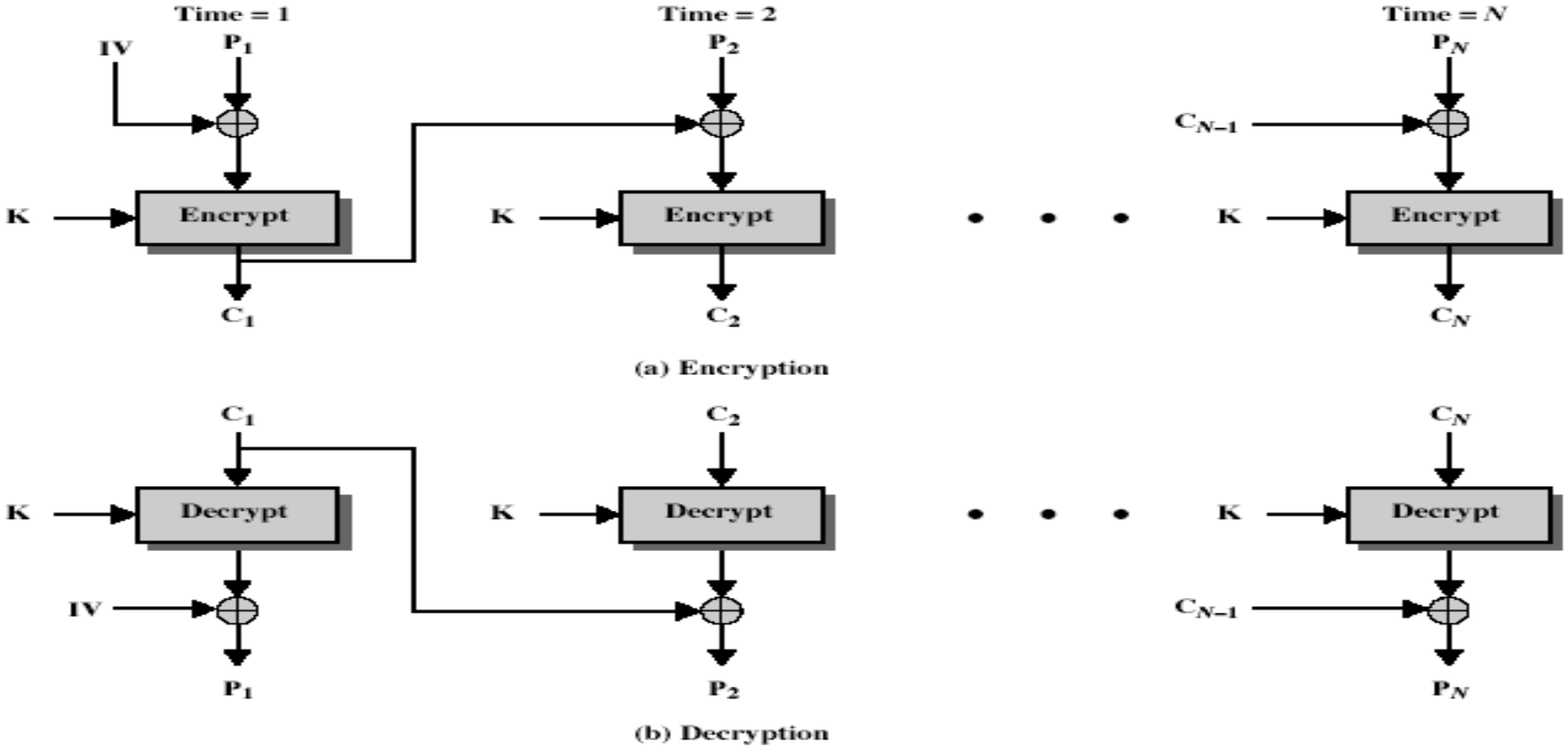
Advantages and Limitations of ECB

- repetitions in message may show in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness due to encrypted message blocks being independent
- main use is sending a few blocks of data

Cipher Block Chaining (CBC)

- message is broken into blocks
- but these are linked together in the encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process (often all 0's)
  - $C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$
  - $C_{-1} = IV$
- uses: bulk data encryption, authentication

Cipher Block Chaining (CBC)



Advantages and Limitations of CBC

- each ciphertext block depends on **all** message blocks
- thus a change in the message affects all ciphertext blocks after the change as well as the original block (avalanche affect)
- need **Initial Value** (IV) known to sender & receiver
  - however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
  - hence either IV must be a fixed value or it must be sent encrypted in ECB mode before rest of message
- Issue: at end of message, how to handle possible last block (if its not complete)
  - by padding either with known non-data value (eg nulls)
  - or pad last block with count of pad size
    - i.e. explicitly have the last byte as a count of how much padding was used (including the count)

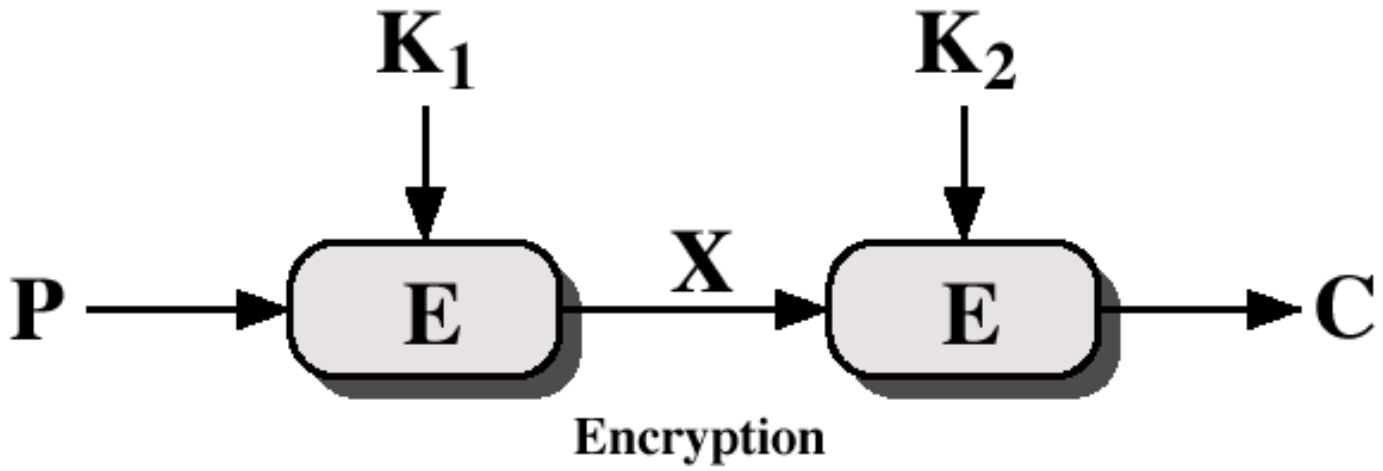
What happened after DES

- Replacement for DES was needed
  - vulnerability to cryptanalysis and practical brute-force attacks
- AES is the new standard (will see after few slides)
  - But took some time to standardize and deploy
- immediate replacement of DES that can be standardized and deployed easily
  - This was 3DES

3DES (Triple DES)

- Another method for a strong cipher
- use multiple encryption with DES with different keys
  - to preserve the investment in DES
  - for quicker deployment
- Triple DES is chosen as a standard method
  - Standardized by ANSI, ISO and NIST

Why not double DES?



Double DES

- It does twice what DES does once
- Uses two keys
  - K1 and K2
- It performs DES on original Plaintext using K1 to get the encrypted text
- It again performs DES on the encrypted text but this time with key K2

Why not double DES?

- Double DES
  - use DES two times with
    two different keys
  - Does not work due to meet-in-the-middle attack (which is a known-plaintext type of an attack)
    - $X = E_{K1}[P]$ ---- $D_{K2}[C]$
    - Try all possible K1's on P to create all possible X's and store them sorted
    - Try all possible K2's on C and match with above table
    - may create some false-alarms, so do the same attack for another plaintext-ciphertext pair
    - If the same K1-K2 pairs match for the second plaintext-ciphertext pair, then the correct keys are most probably found
    - complexity of this attack is close to the complexity of the single-DES brute-force attack, so double-DES is useless

- **K = 00011010101010101010101010010**
- Single Des Encryption

  $(K = K_1, K_2, K_3 \ldots\ldots K_{16})$

  Single des decryption

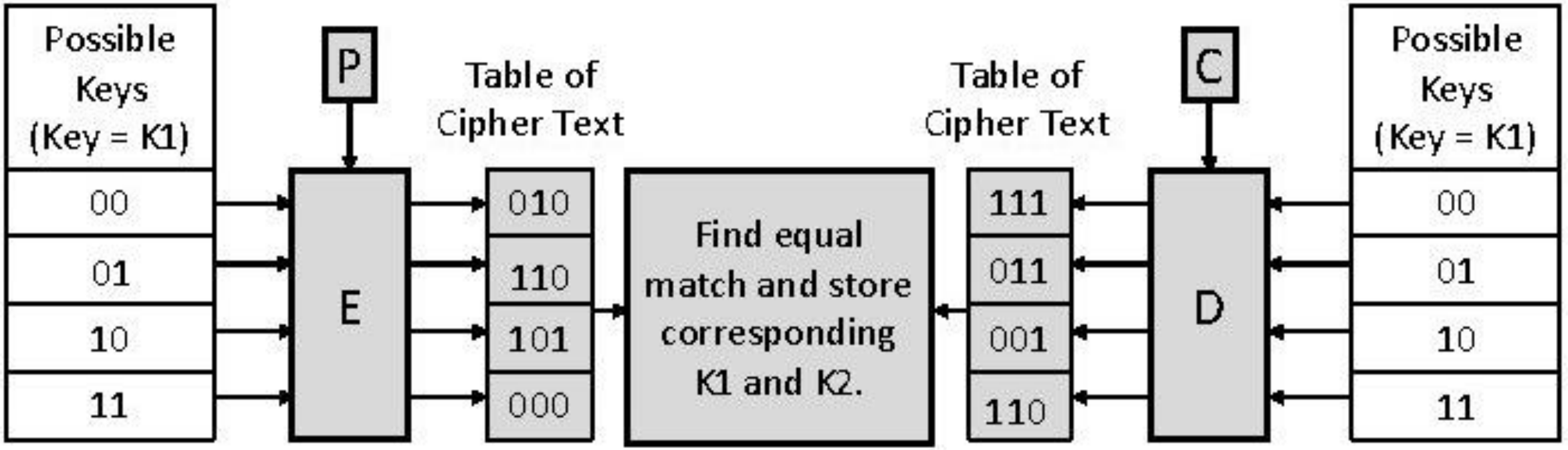
  $(K = K_{16}, K_{15}, K_{14} \ldots\ldots K_1)$

Double Des:

K1= 00011010101010101010101010010

K2= 00111111000011111111100011010

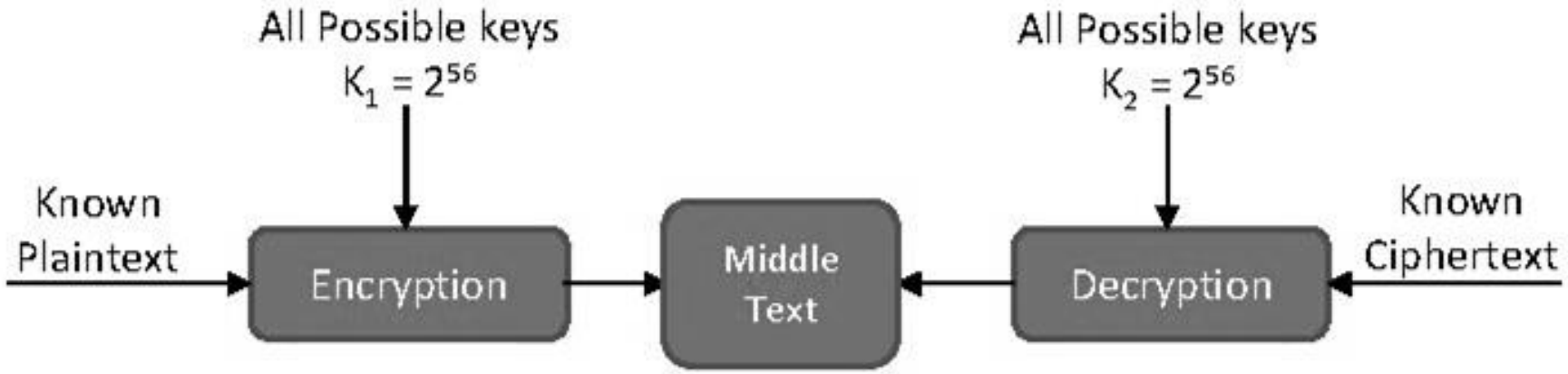
Meet in the Middle Attack



Values of K1=01 and K2=11

Meet in the Middle Attack

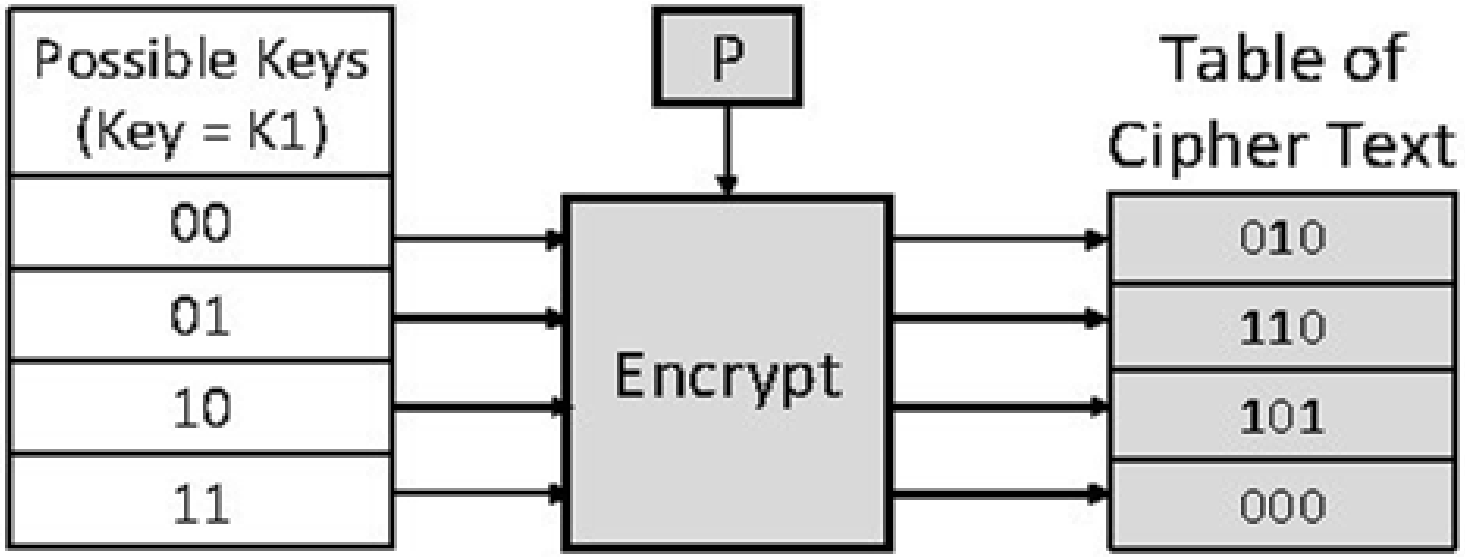
- This attack involves encryption from one end, decryption from the other and matching the results in the middle.
- Suppose cryptanalyst knows P_i and corresponding C
- I



- No. of Encryptions and Decryptions: 2 x 2^56 = 2^57
- For Double DES requires 2^57 operations for brute force attack

Meet in the Middle Attack Step -1

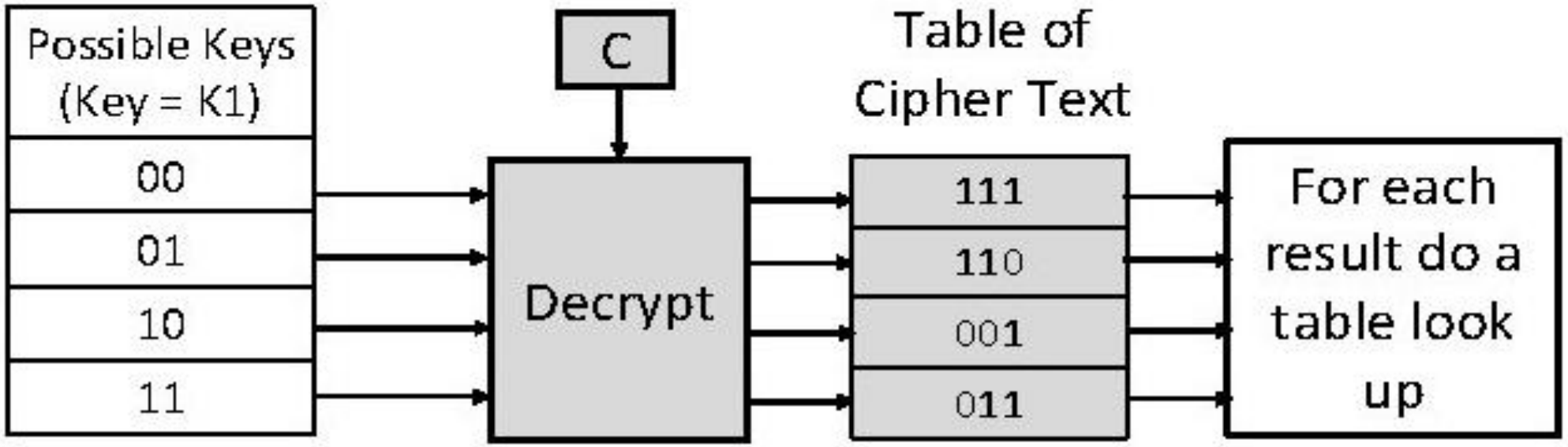
- For all possible values (2^{56}) of key K1, the cryptanalyst would encrypt the known plaintext by performing E(K1,P).
- The cryptanalyst would store output in a table.



Cryptanalyst encryption operation

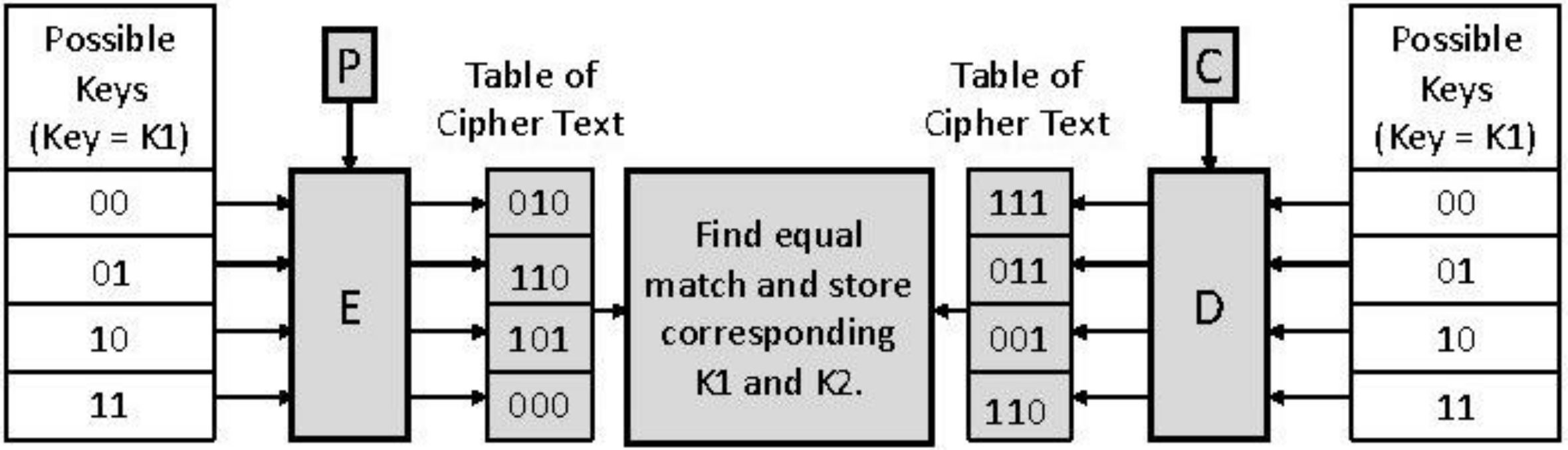
Meet in the Middle Attack Step -2

- Cryptanalyst decrypt the known ciphertext with all possible values of K2.
- In each case cryptanalyst will compare the resulting value with the all values in the table of ciphertext.



Cryptanalyst decryption operation

Meet in the Middle Attack

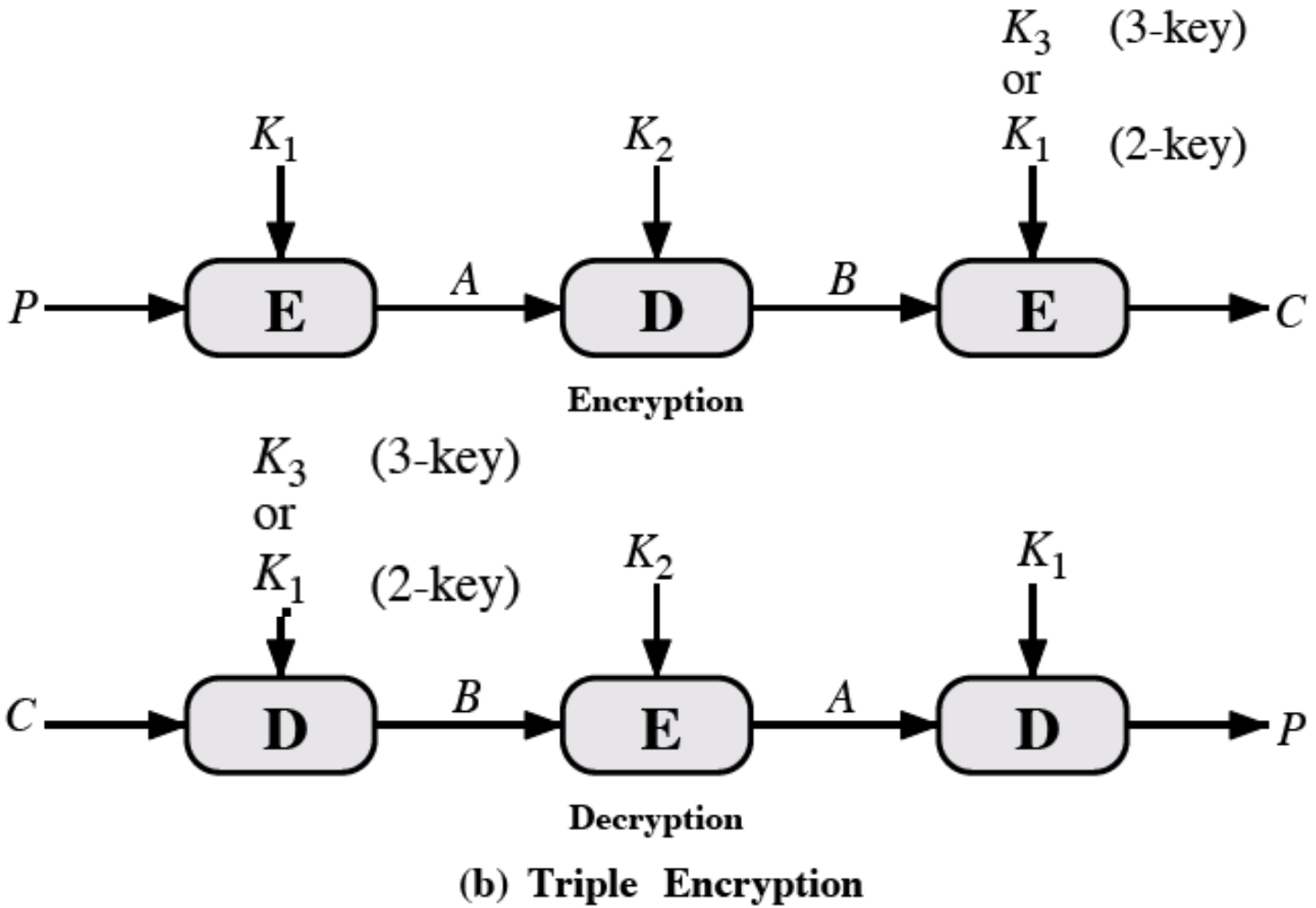


Values of K1=01 and K2=11

3DES (Triple-DES)

- Three stages of DES
  - **with two different keys**
    - some attacks are possible but impractical
    - Merkle and Hellman, 1981
      - 2^{56} trials, but requires 2^{56} plaintext-ciphertext pairs
    - Oorschot and Wiener, 1990
      - $2^{120}/n$ trials, where n is the number of plaintext-ciphertext pairs
  - **with three different keys**
    - Attack complexity increases and becomes impractical

Triple-Des with two/three keys



(b) Triple Encryption

Triple-DES with Two-Keys

Hence must use 3 encryptions

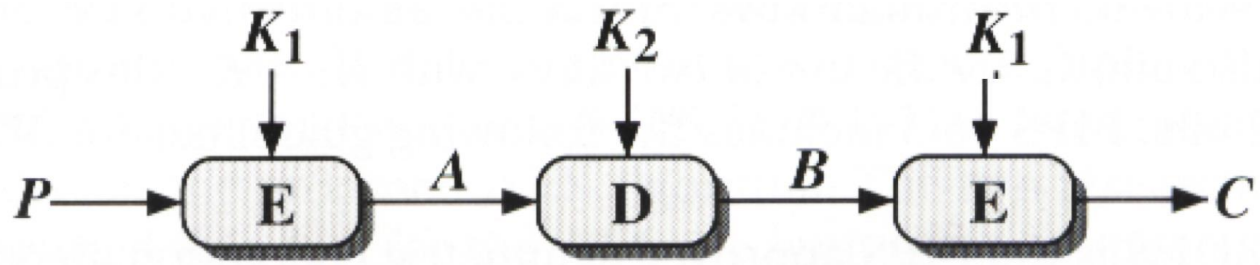
- Would seem to need 3 distinct keys

- But can use 2 keys with E-D-E sequence
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - If $K1=K2$ then can work with single DES
- Standardized in ANSI X9.17 & ISO8732
- No current known practical attacks

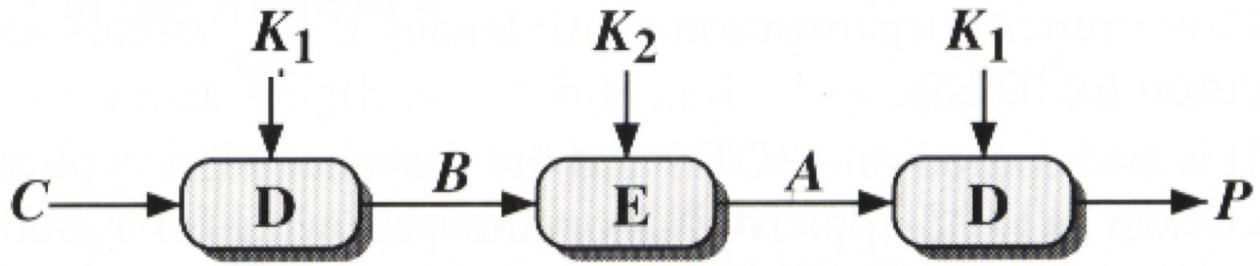
Why do we use Ede order for 3DES?

- Encrypt-decrypt-encrypt (EDE) is the preferred method because **if a single key is used for all 3 operations it is equivalent to regular 56-bit DES**. That is, a 56-bit DES implementation can decrypt that message. This makes this version of 3DES backwards compatible with DES.

Triple DES with 2 Keys



(a) Encryption



(b) Decryption

$$C = E_{k1}(D_{k2}(E_{k1}(P)))$$

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
  - $C = E_{K3}[E_{K2}[E_{K1}[P]]]$
- Has been adopted by some Internet applications, e.g. PGP, S/MIME

END