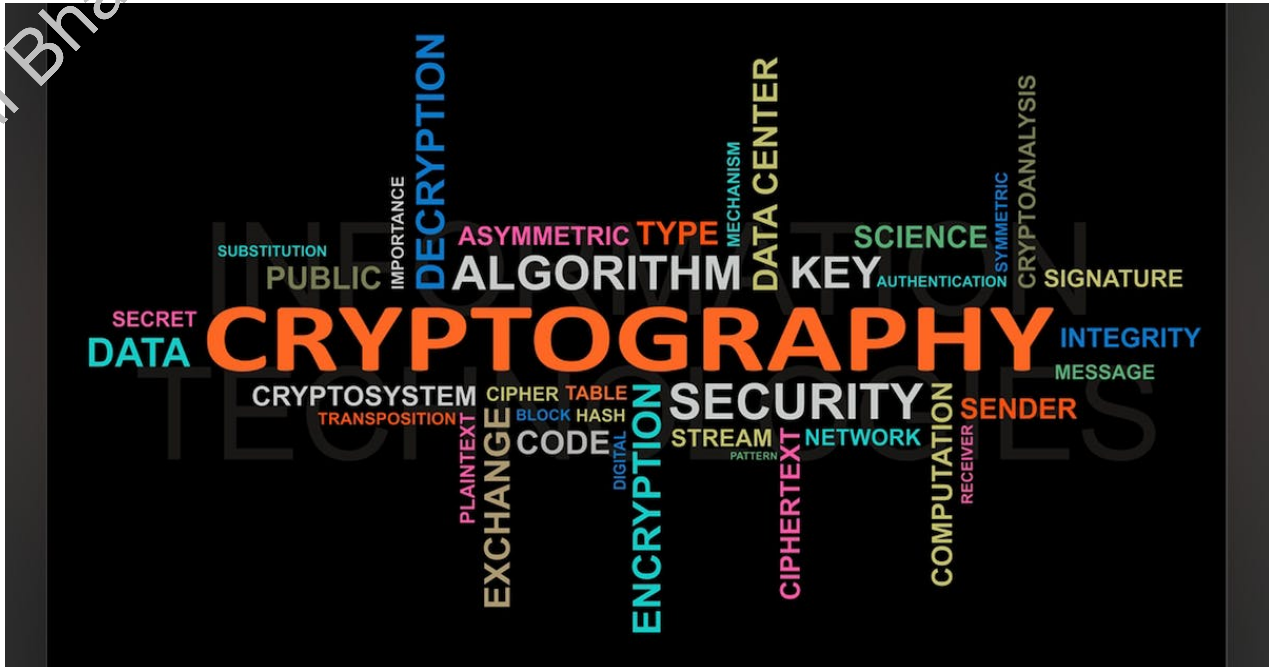
Data Communications and Networking Fourth Edition

Forouzan

Chapter 2

Cryptography

- By
- **Prof. Muhammad Iqbal Bhat**
- Government Degree College Beerwah



Cryptography

Cryptography is a branch of mathematics that deals with the transformation of data. Cryptographic algorithms are used in many ways in information security and network security.

Cryptography is the practice and study of techniques for securing communication and data in the presence of adversaries.

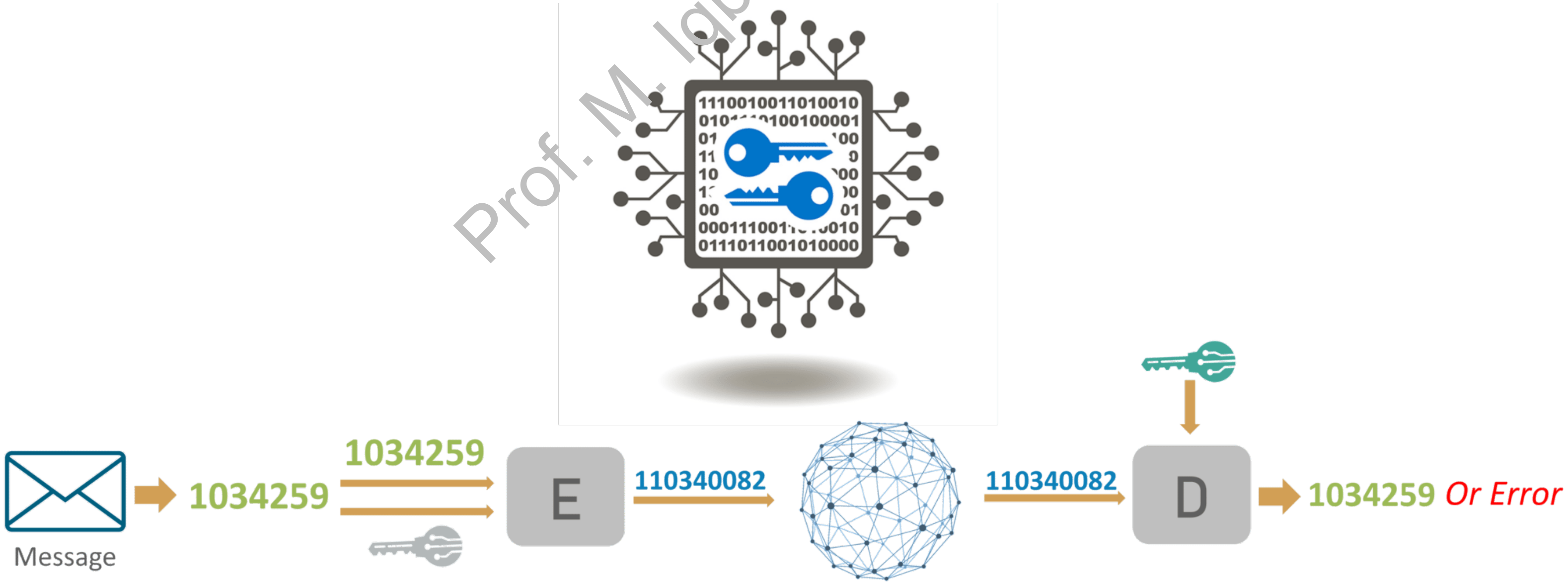


Figure 30.1 *Cryptography components*

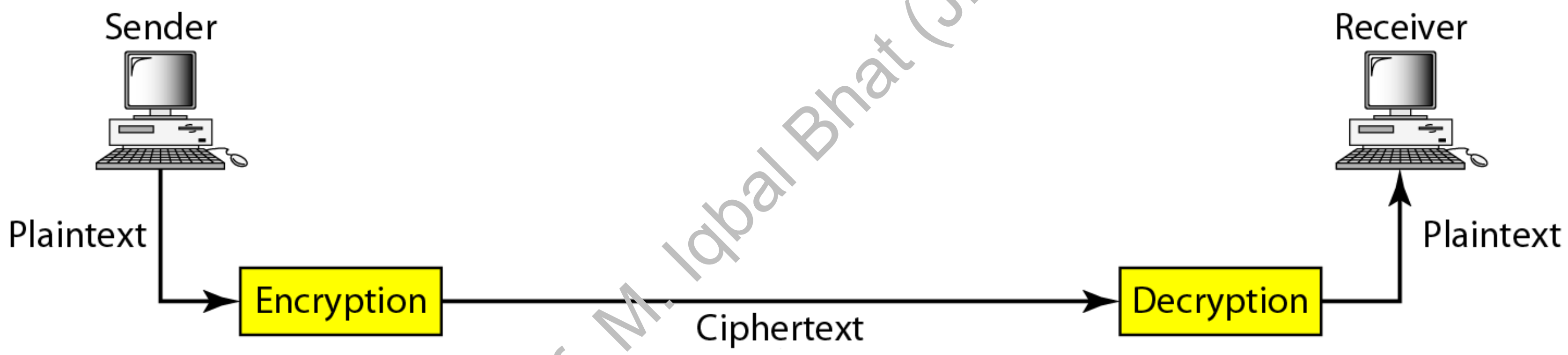


Figure 30.2 *Categories of cryptography*

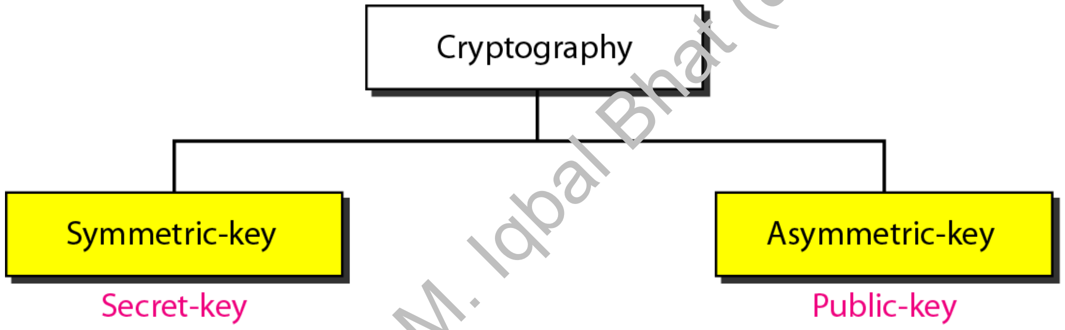
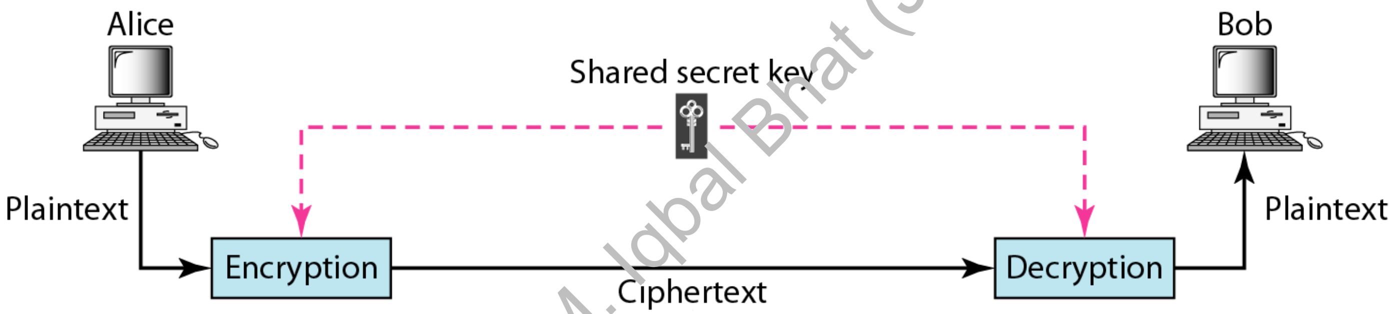


Figure 30.3 *Symmetric-key cryptography*



Note

In symmetric-key cryptography, the same key is used by the sender (for encryption)
and the receiver (for decryption).
The key is shared.

Figure 30.4 *Asymmetric-key cryptography*

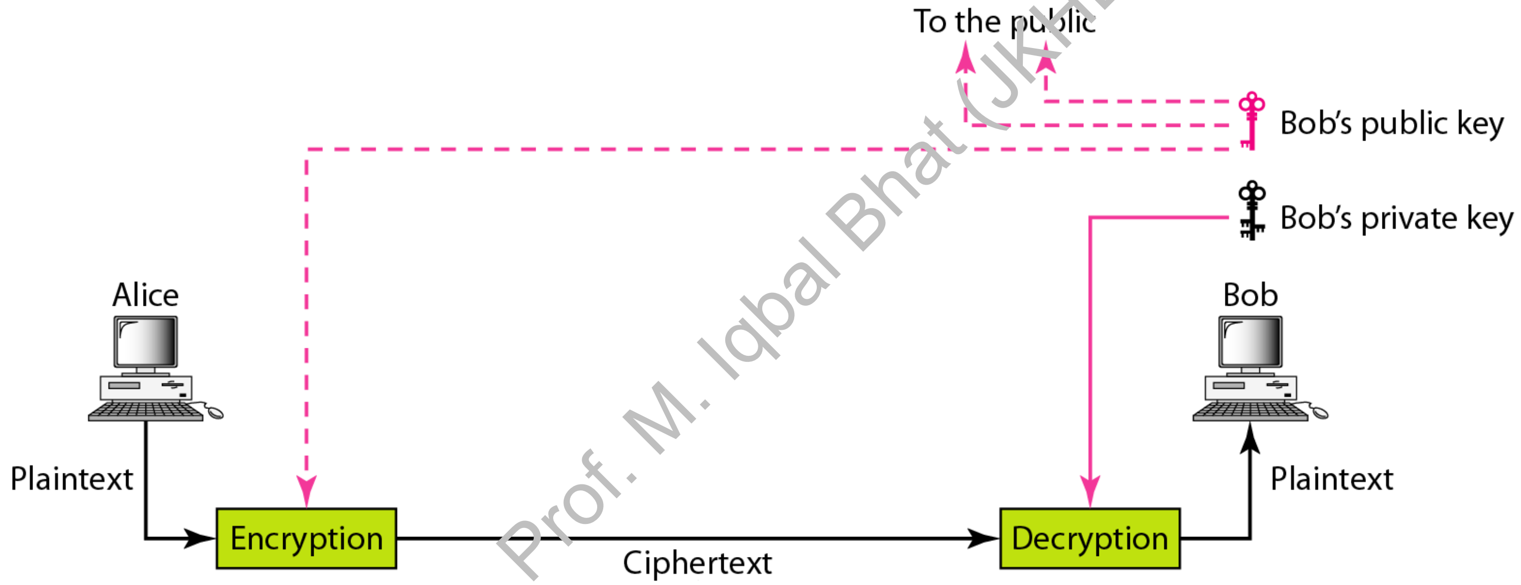


Figure 30.5 *Keys used in cryptography*

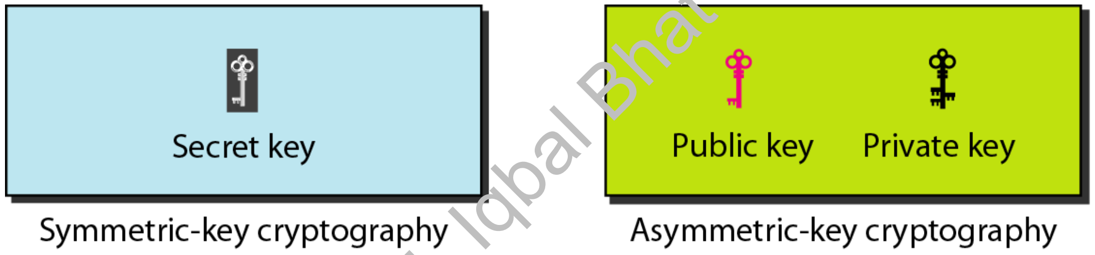
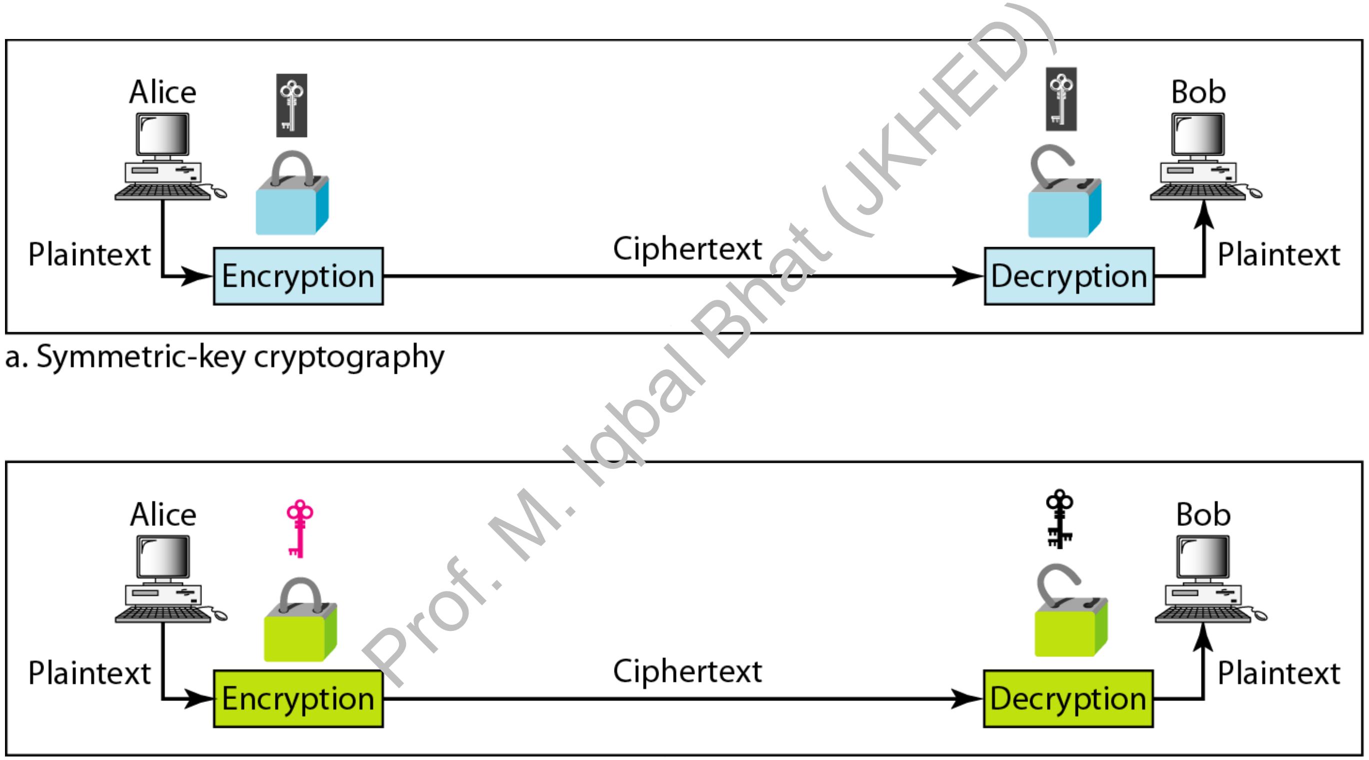


Figure 30.6 *Comparison between two categories of cryptography*



a. Symmetric-key cryptography

b. Asymmetric-key cryptography

30-2 SYMMETRIC-KEY CRYPTOGRAPHY

Symmetric-key cryptography started thousands of years ago when people needed to exchange secrets (for example, in a war). We still mainly use symmetric-key cryptography in our network security.

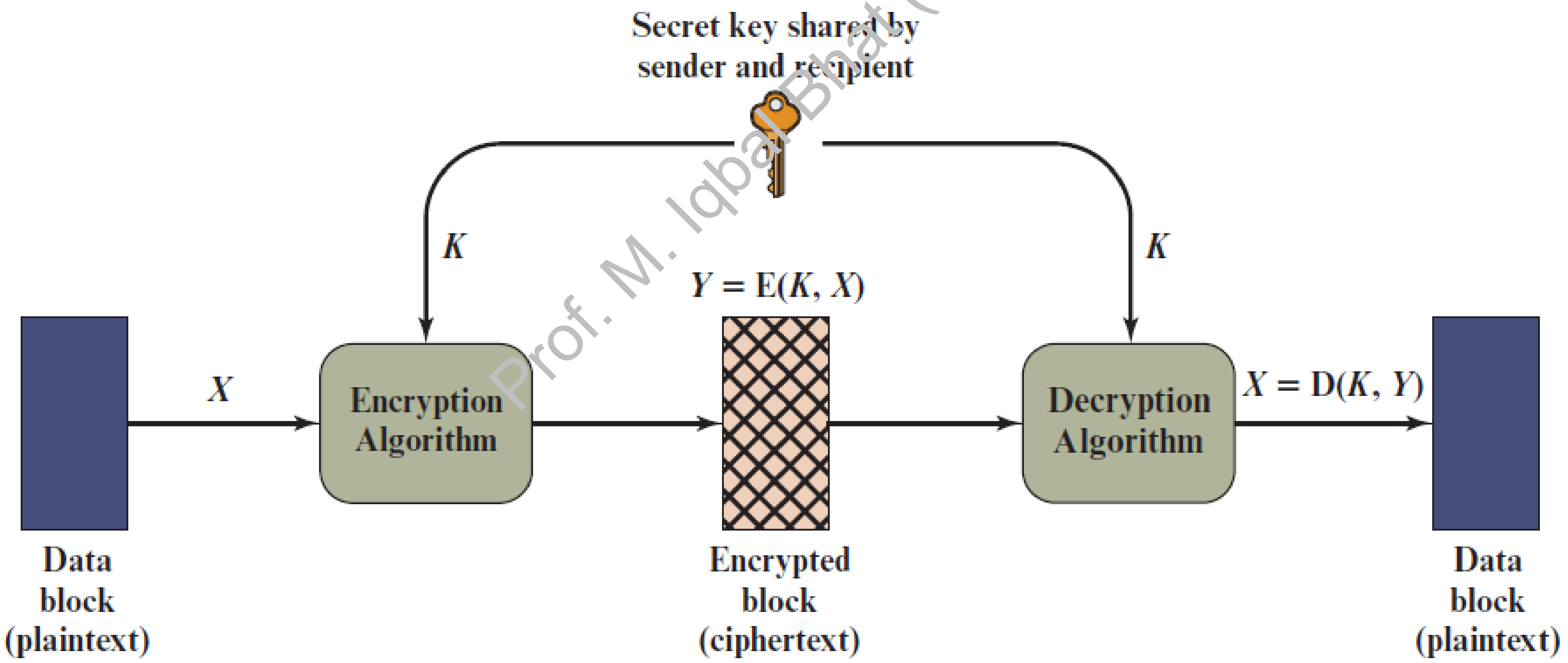
Topics discussed in this section:

Traditional Ciphers
Simple Modern Ciphers
Modern Round Ciphers
Mode of Operation

Cryptography – some notations

- $Y = E_K(X)$ denotes that Y is the encryption of the plaintext X using the key K
- $X = D_K(Y)$ denotes that X is the decryption of the cipher text Y using the key K

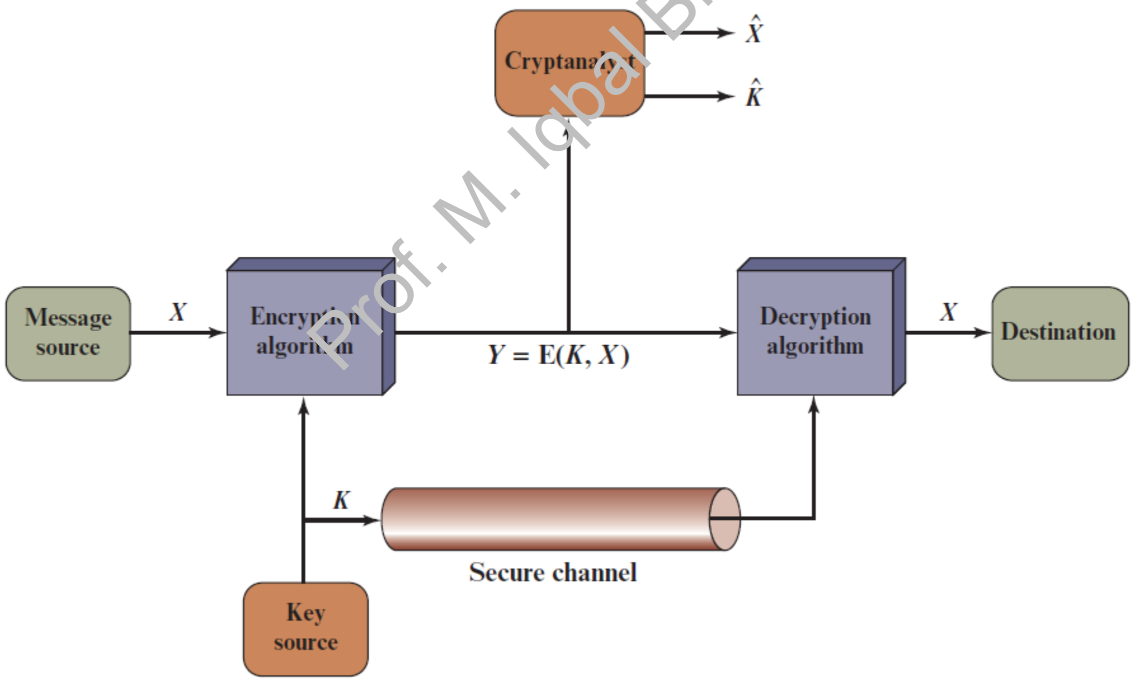
$$D_K(E_K(Y))=Y$$



Cryptanalysis and Brute-Force Attack

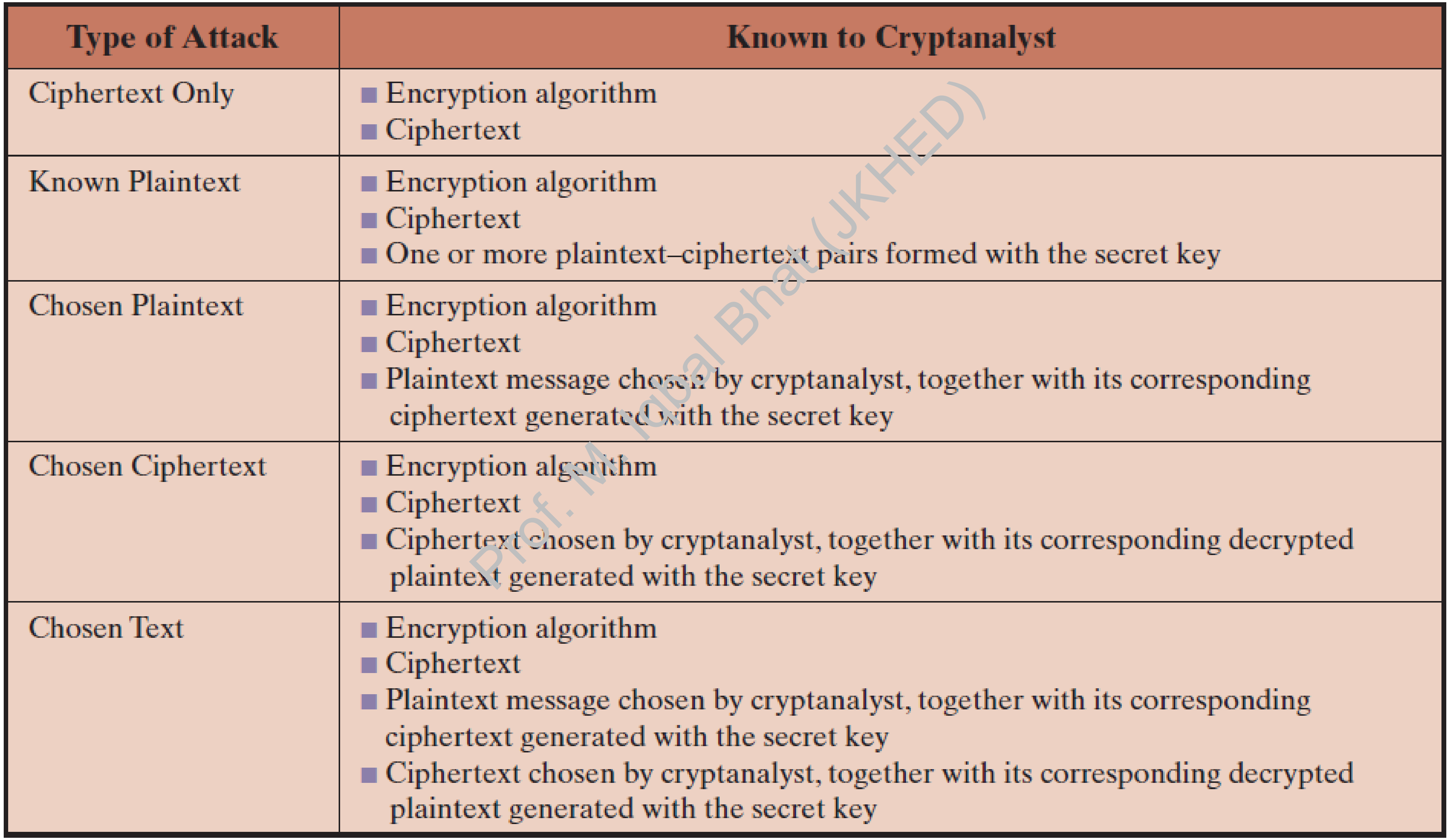
Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.



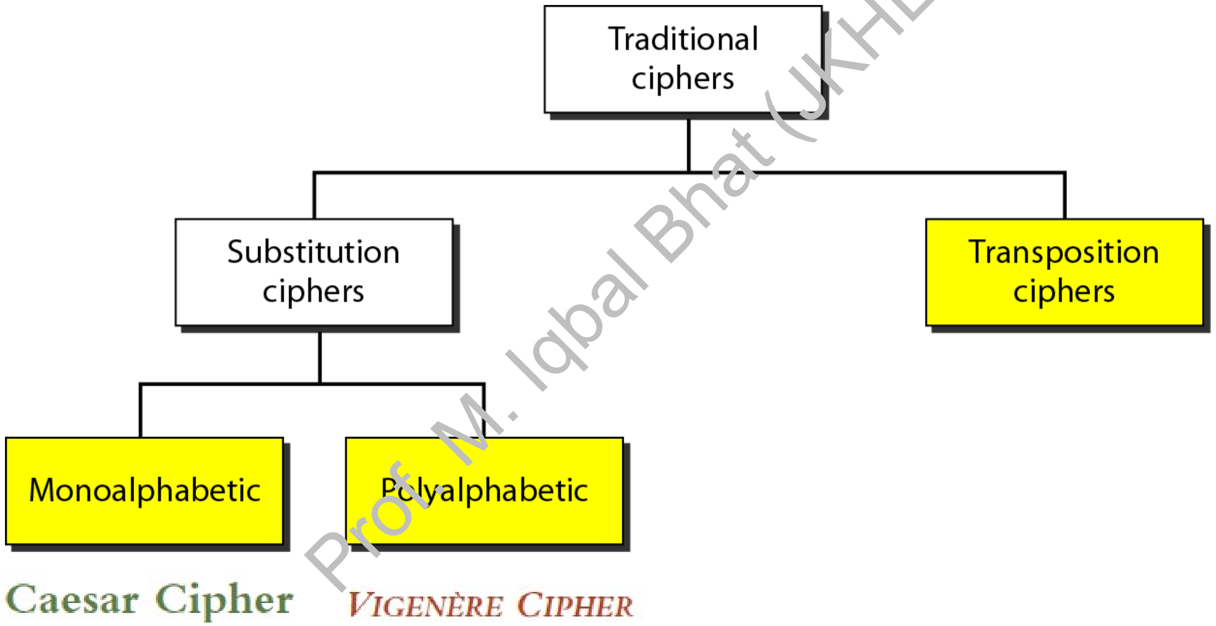
Cryptanalyst Attacks

Table 3.1 Types of Attacks on Encrypted Messages



Type of Attack	Known to Cryptanalyst
Ciphertext Only	■ Encryption algorithm ■ Ciphertext
Known Plaintext	■ Encryption algorithm ■ Ciphertext ■ One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	■ Encryption algorithm ■ Ciphertext ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	■ Encryption algorithm ■ Ciphertext ■ Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	■ Encryption algorithm ■ Ciphertext ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key ■ Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

Figure 30.7 *Traditional ciphers*



Substitution Ciphers

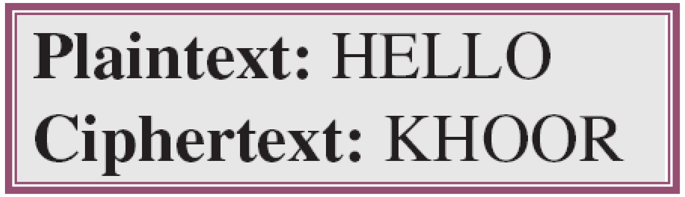
A substitution cipher replaces one symbol with another.

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns.

Example 30.1

The following shows a plaintext and its corresponding ciphertext. Is the cipher monoalphabetic?



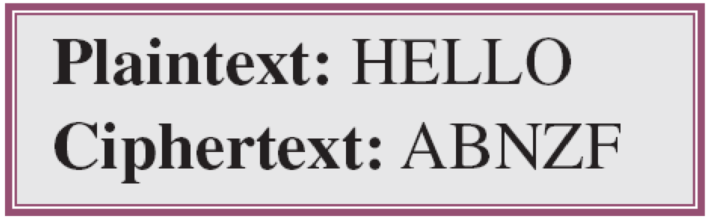
Plaintext: HELLO
Ciphertext: KHOOR

Solution

The cipher is probably monoalphabetic because both occurrences of L's are encrypted as O's.

Example 30.2

The following shows a plaintext and its corresponding ciphertext. Is the cipher monoalphabetic?



Plaintext: HELLO
Ciphertext: ABNZF

Solution

The cipher is not monoalphabetic because each occurrence of L is encrypted by a different character. The first L is encrypted as N; the second as Z.

Note

The shift cipher is sometimes referred to as the Caesar cipher.

Example 30.3

Use the shift cipher with key = 15 to encrypt the message "HELLO."

Solution

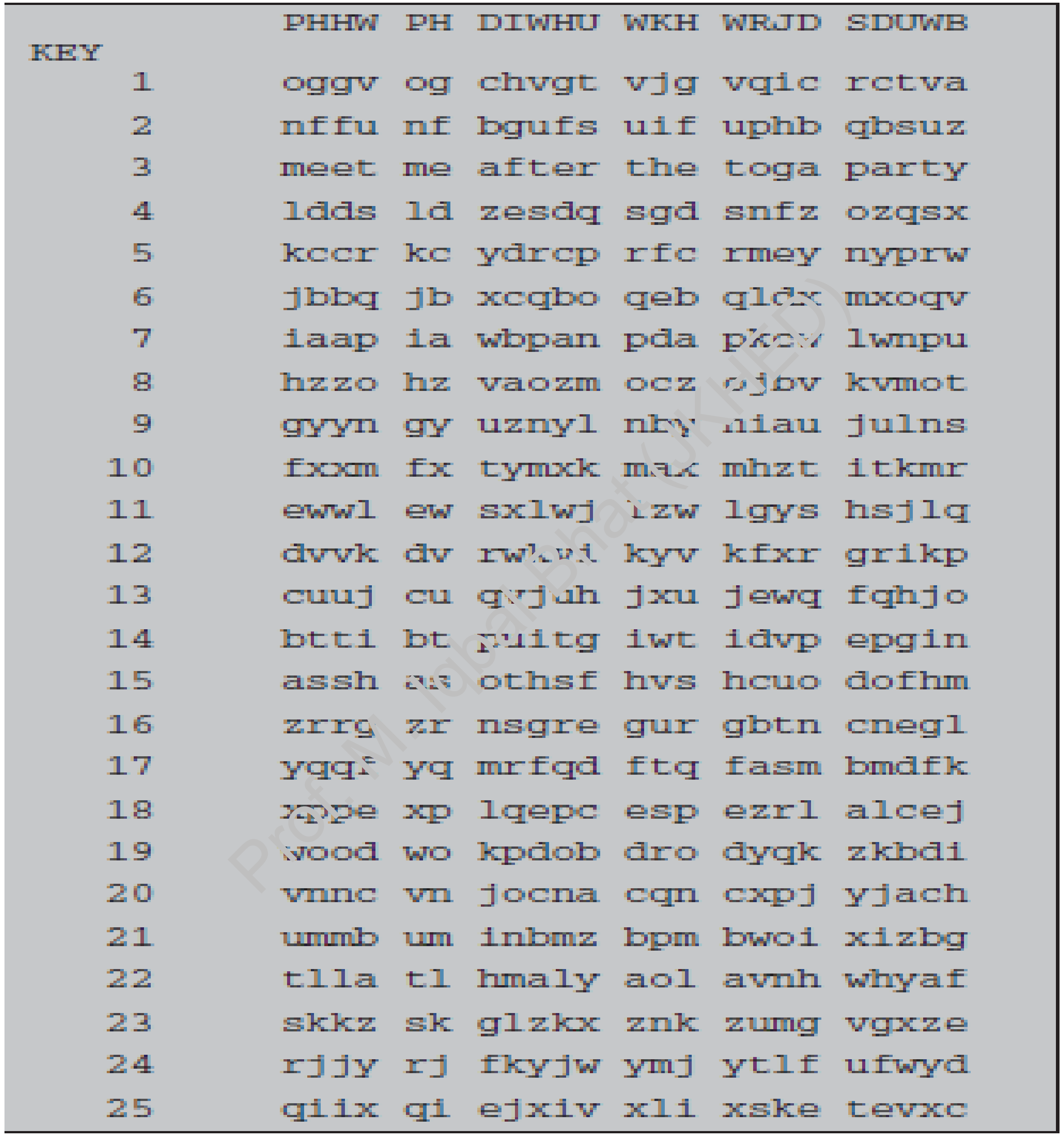
We encrypt one character at a time. Each character is shifted 15 characters down. Letter H is encrypted to W. Letter E is encrypted to T. The first L is encrypted to A. The second L is also encrypted to A. And O is encrypted to D. The cipher text is WTAAD.

Example 30.4

Use the shift cipher with key = 15 to decrypt the message "WTAAD."

Solution

We decrypt one character at a time. Each character is shifted 15 characters up. Letter W is decrypted to H. Letter T is decrypted to E. The first A is decrypted to L. The second A is decrypted to L. And, finally, D is decrypted to O. The plaintext is HELLO.



KEY	PHHW	PH	DIWHU	WKH	WRJD	SDUWB
1	oggv	og	chvgt	vjg	vqic	rctva
2	nffu	nf	bgufs	uif	uphb	qbsuz
3	meet	me	after	the	toga	party
4	ldds	ld	zesdq	sgd	snfz	ozqsx
5	kccr	kc	ydrcp	rfc	rmey	nyprw
6	jbbq	jb	xcqbo	qeb	qldx	mxoqv
7	iaap	ia	wbpan	pda	pkcw	lwnpu
8	hzzo	hz	vaozm	ocz	ojbv	kvmot
9	gyyn	gy	uznyl	nby	niau	julns
10	fxxm	fx	tymxk	max	mhzt	itkmr
11	ewwl	ew	sxlwj	lzw	lgys	hsjlq
12	dvvk	dv	rwkvi	kyv	kfxr	grikp
13	cuuj	cu	qvjuh	jxu	jewq	fqhjo
14	btti	bt	puitg	iwt	idvp	epgin
15	assh	as	othsf	hvs	hcuo	dofhm
16	zrrg	zr	nsgre	gur	gbtn	cnegl
17	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18	xppe	xp	lqepc	esp	ezrl	alcej
19	wood	wo	kpdob	dro	dyqk	zkbdi
20	vnnc	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzkx	znk	zumg	vgxze
24	rjjy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxc

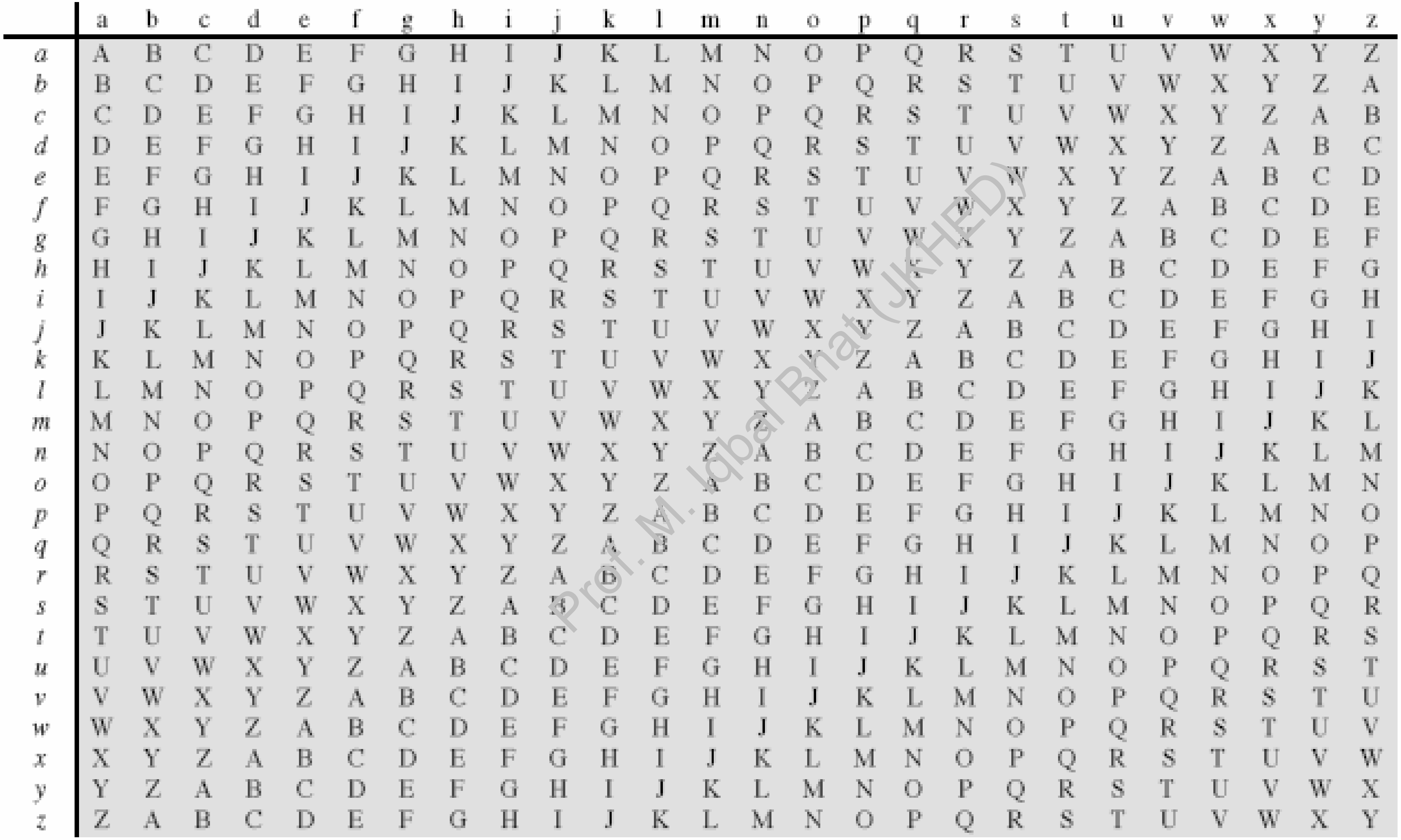
Polyalphabetic substitution ciphers

- Well, one way is to use **more than one** alphabet, switching between them systematically. This type of cipher is called a ***polyalphabetic substitution*** **cipher ("poly" is the Greek root for "many"). The difference, as you will see, is that** **frequency analysis** **no longer works the same way to break these**
- Idea: use different monoalphabetic substitutions as one proceeds through the plaintext
- Makes cryptanalysis harder with more alphabets (substitutions) to guess and flattens frequency distribution
- A key determines which particular substitution is used in each step
    - Example: the Vigenère cipher

Vigenère

- Proposed by Giovan Batista Belaso(1553) and reinvented by Blaisede Vigenère (1586), called "**le chiffreindéchiffrable"**for 300 years
- Effectively multiple Caesar ciphers
- Key is a word K = k1 k2 ... kd
- Encryption
  - Read one letter **t** from the plaintext and one letter **k** from the key
  - **t** is encrypted according to the Caesar cipher with key **k**
  - When the key word is finished, start the reading of the key from the beginning
- Decryption works in reverse
  - Example: key is "bcde"; "testing" is encrypted as "ugvxjpj"
  - Note that the two 't' are encrypted by different letters: 'u' and 'x'
  - The two 'j' in the crypto text come from different plain letters: 'i' and 'g'

Table 2.3 The Modern Vigenère Tableau



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Example Vigenère

Example

- •write the plaintext out
- •write the keyword repeated above it
- •use each key letter as a Caesar cipher key
- •encrypt the corresponding plaintext letter
- •eg using keyword *deceptive*
    - **plain: wearediscoveredsaveyourself**
    - **key: deceptivedeceptivedeceptive**
- **cipher: ZICVTWQNGRZGVTWAVZHCQYGLMGJ**

Security of Vigenère Ciphers

- Its strength lays in the fact that each plaintext letter has multiple cipher text letters
    - Letter frequencies are obscured (but not totally lost)
- Breaking Vigenère
    - If we need to decide if the text was encrypted with a monoalphabetic cipher or with Vigenère:
        - Start with letter frequencies
        - See if it "looks" monoalphabetic or not: the frequencies should be those of letters in English texts
        - If not, then it is Vigenère

One time pad

- The idea of the auto key system can be extended to create an unbreakable system: **one-time pad**
- Idea: use a (truly) random key as long as the plaintext
- It is unbreakable since the cipher text bears no statistical relationship to the plaintext
- Moreover, for **any plaintext** & **any cipher text** there exists a key mapping one to the other
- Thus, a cipher text can be decrypted to any plaintext of the same length
- The cryptanalyst is in an impossible situation

One time pad example

- THE BRITISH ARE COMING
- DKJFOISJOGIJPAPDIGN
- Step 1-
  - T H E B R I T I S H A R E C O M I N G
D K J F O I S J O G I J P A P D I G N
- Step 2 - Determine an algorithm
  - A=0
  - B=1
  - C=2
  - D=3
  - E=4
  - F=5
- It follows the formula "(plaintext + key) MOD alphabet length":

One time pad cont’d

- **Step 3 - Perform the encryption**

(T(19)+D(03)=22) MOD 26 = 22 = W
(H(07)+K(10)=17) MOD 26 = 17 = R
(E(04)+J(09)=13) MOD 26 = 13 = N
(B(01)+F(05)=06) MOD 26 = 06 = G
(R(17)+O(14)=31) MOD 26 = 05 = F
(I(08)+I(08)=16) MOD 26 = 16 = Q
(T(19)+S(18)=37) MOD 26 = 11 = L
(I(08)+J(09)=17) MOD 26 = 17 = R
(S(18)+O(14)=32) MOD 26 = 06 = G
(H(07)+G(06)=13) MOD 26 = 13 = N
(A(00)+I(08)=08) MOD 26 = 08 = I
(R(17)+J(09)=26) MOD 26 = 00 = A
(E(04)+P(15)=19) MOD 26 = 19 = T
(C(02)+A(00)=02) MOD 26 = 02 = C
(O(14)+P(15)=29) MOD 26 = 03 = D
(M(12)+D(03)=15) MOD 26 = 15 = P
(I(08)+I(08)=16) MOD 26 = 16 = Q
(N(13)+G(06)=19) MOD 26 = 19 = T
(G(06)+N(13)=19) MOD 26 = 19 = T

Pad cont’d

- We now show two different decryptions using two different keys:
- ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
- key: *pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih*
- plaintext: mr mustard with the candlestick in the hall
- ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
- key: *mfugpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt*
- plaintext: miss scarlet with the knife in the library

Pad cont’d

- Two plausible plaintexts are produced.
- How is the cryptanalyst to decide which is the correct decryption
- If the actual key were produced in a truly random fashion, then the cryptanalyst cannot say that one of these two keys is more likely than the other.

Security of the one-time pad

- The security is entirely given by the randomness of the key
    - If the key is truly random, then the ciphertext is random
    - A key can only be used **once**if the cryptanalyst is to be kept in the“dark”
- Problems with this “perfect” cryptosystem
    - Making large quantities of **truly random**characters is a significant task
    - Key distribution is enormously difficult: for any message to be sent, a key of equal length must be available to both parties

Other technique of encryption: Transpositions

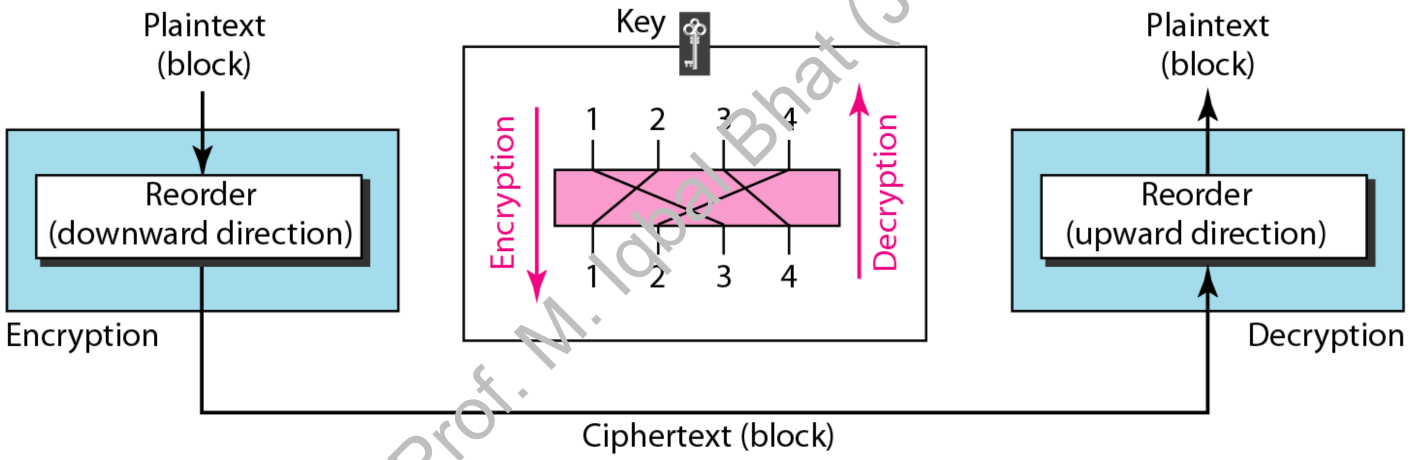
We have considered so far **substitutions** to hide the plaintext: each letter is mapped into a letter according to some substitution

- *Different idea*: perform some sort of permutation on the plaintext letters
- Hide the message by rearranging the letter order without altering the actual letters used
- The simplest such technique: *rail fence technique*

Note

A transposition cipher reorders (permutes) symbols in a block of symbols.

Figure 30.8 *Transposition cipher*



Example 30.5

Encrypt the message "HELLO MY DEAR," using the key shown in Figure 30.8.

Solution

We first remove the spaces in the message. We then divide the text into blocks of four characters. We add a bogus character Z at the end of the third block. The result is HELL OMYD EARZ. We create a three-block ciphertext ELHLMDOYAZER.

Example 30.6

Using Example 30.5, decrypt the message "ELHLMDOYAZER".

Solution

The result is HELL OMYD EARZ. After removing the bogus character and combining the characters, we get the original message "HELLO MY DEAR."

Rail Fence cipher

- **Idea**:write plaintext letters diagonally over a number of rows, then read off cipher row by row
- E.g., with a rail fence of depth 2, to encrypt the text “meet me after the toga party”, write message out as:

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

- Ciphertext is read from the above row-by-row:
  - MEMATRHTGPRYETEFETEOAAT
- **Attack**: this is easily recognized because it has the same frequency distribution as the original text

Row transposition ciphers

- More complex scheme: **row transposition**
- Write letters of message out in rows over a specified number of columns
- Reading the cryptotext column-by-column, with the columns permuted according to some key
- **Example:** “attack postponed until two am” with key 4312567:
- **Key:**
- **Plaintext:**

4	3	1	2	5	6	7
a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i	l	t
w	o	a	m	x	y	z

Row transposition ciphers

- **Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ**
- If we number the letters in the plaintext from 1 to 28, then the result of the first encryption is the following permutation of letters from plaintext:03 10 17 24 04 11 18 25 02 09 16 23 01 08 15 22 05 12 19 26 06 13 20 27 07 14 21 28
- Note the regularity of that sequence!
- Easily recognized!

Iterating the encryption makes it more secure

- Idea: use the same scheme once more to increase security

- Key: 4 3 1 2 5 6 7
- Input: T T N A A P T

  M T S U O A O

  D W C O I X K

  N L Y P E T Z

- Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
- After the second transposition we get the following sequence of letters:
  - 17 09 05 27 24 16 12 07 10 02 22 20 03 25 15 12 04 23 19 14 11 01 26 21 18 08 06 28
- This is far less structured and so, more difficult to cryptanalyze

Confusion vs Diffusion

Two Important Properties of Ciphers

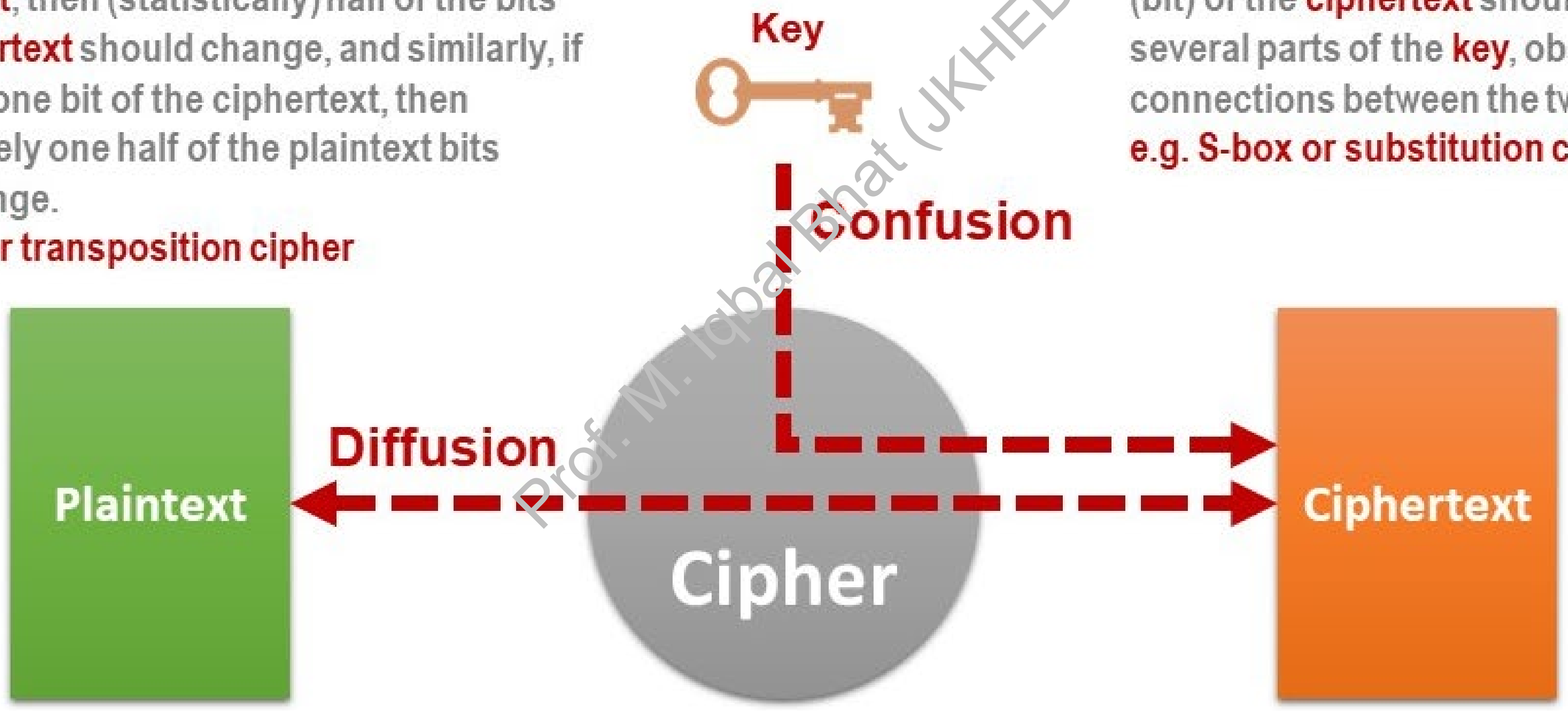
- In 1949, Claude Shannon first proposed the ideas of confusion and diffusion in the operation of a cipher.

Diffusion means that if we change a single bit of the **plaintext**, then (statistically) half of the bits in the **ciphertext** should change, and similarly, if we change one bit of the ciphertext, then approximately one half of the plaintext bits should change.

e.g. P-box or transposition cipher

Confusion means that each binary digit (bit) of the **ciphertext** should depend on several parts of the **key**, obscuring the connections between the two.

e.g. S-box or substitution cipher



Confusion vs Diffusion

Confusion

1. Confusion is the property of a cipher whereby it provides no clue regarding the relationship between the ciphertext and the key.
2. Confusion means that each binary digit (bit) of the ciphertext should depend on several parts of the key, obscuring the connections between the two.
3. This property makes it difficult to find the key from the ciphertext and if a single bit in a key is changed, most or all the bits in the ciphertext will be affected.
4. Confusion increases the ambiguity of ciphertext, and it is used by both block and stream cipher.
5. A Strong substitution (S-boxes) function enhances confusion

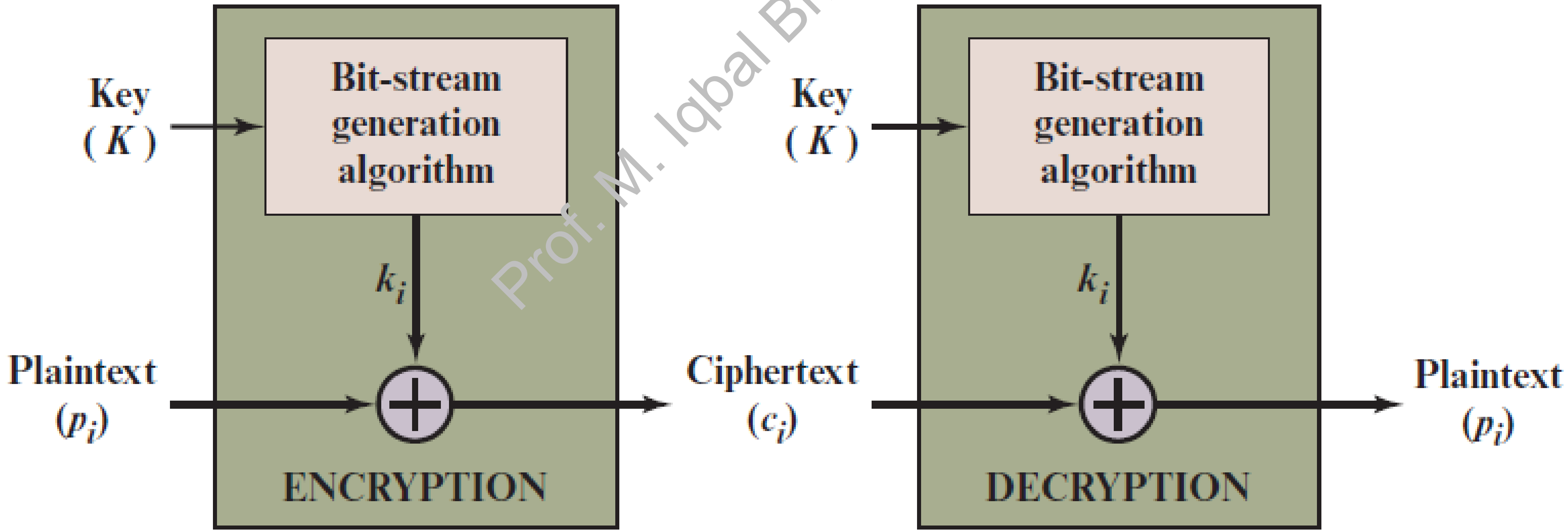
Diffusion

- Diffusion is concerned with the relationship between the plaintext and the corresponding cipher text.
- Diffusion means that if we change a single bit of the plaintext, then half of the bits in the ciphertext should change, and similarly, if we change one bit of the ciphertext, then approximately one-half of the plaintext bits should change.
- Since a bit can have only two states, when they are all re-evaluated and changed from one seemingly random position to another, half of the bits will have changed state.
- A Strong transposition (P-boxes) enhances diffusion.

Cipher Techniques

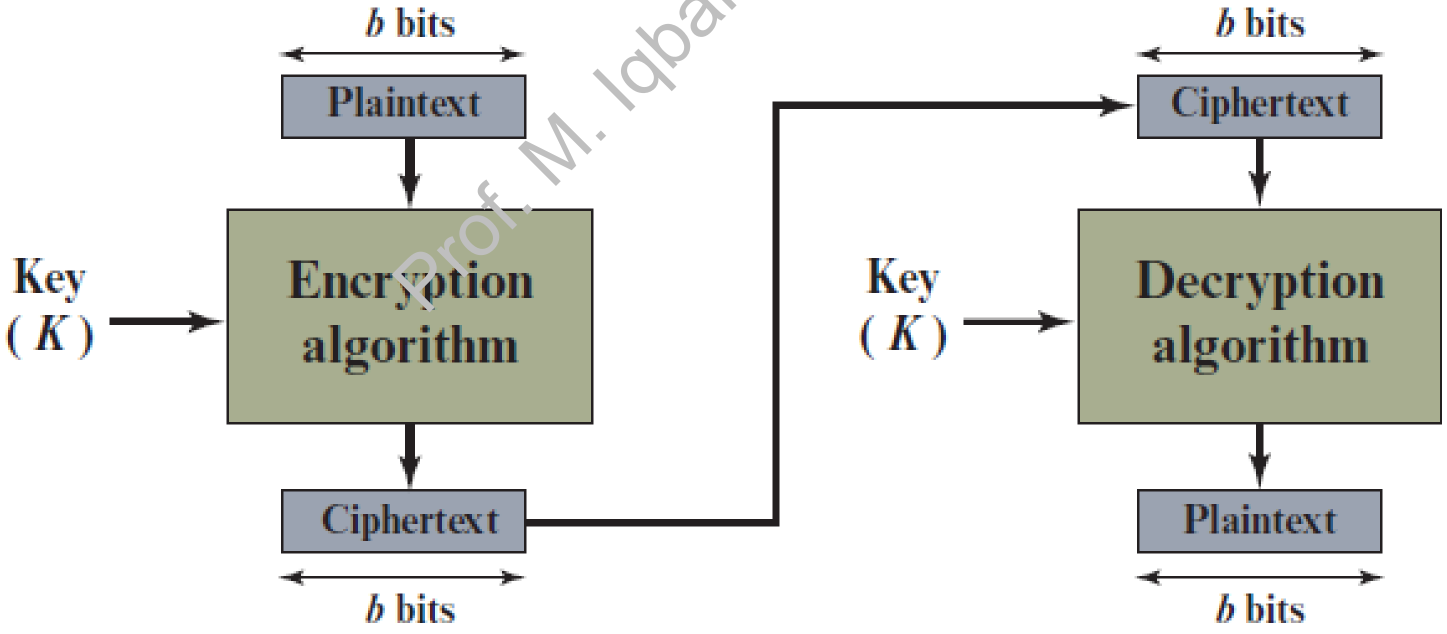
Stream Ciphers

- **Stream cipher** is one that encrypts a digital data stream one bit (or byte) at a time
  - Example: auto keyed Vigenère cipher and the Vernam cipher



Block Ciphers

- **Block cipher** is one in which the plaintext is divided into blocks and one block is encrypted at one time producing a ciphertext of equal length
  - Similar to substitution ciphers on very big characters: 64 bits or 128 bits are typical block lengths

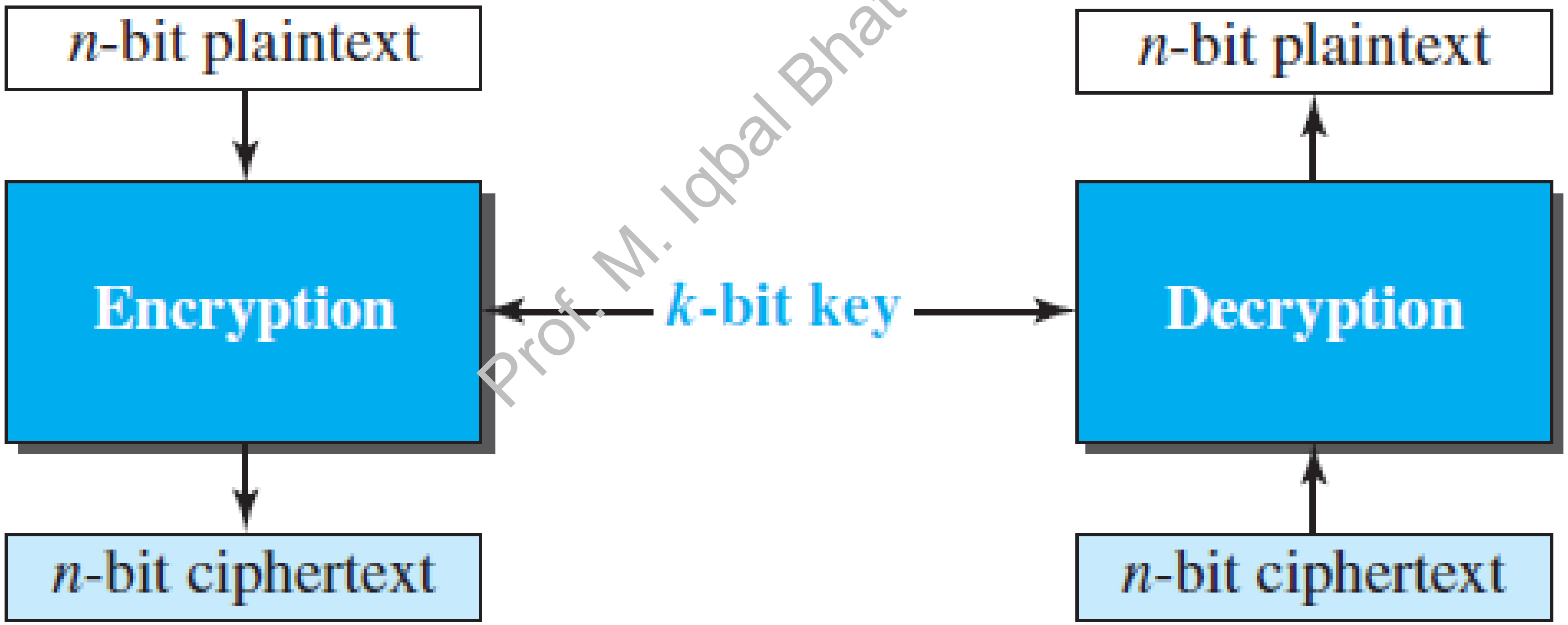


Block Cipher Vs Stream Cipher

Block Cipher	Stream Cipher
• Block Cipher Converts the plain text into cipher text by taking plain text’s block at a time.	• Stream Cipher Converts the plaint text into cipher text by taking 1 byte of plain text at a time.
• Block cipher uses either 64 bits or more than 64 bits.	• While stream cipher uses 8 bits.
• Block cipher Uses confusion as well as diffusion.	• While stream cipher uses only confusion.
• In block cipher, reverse encrypted text is hard.	• While in stream cipher, reverse encrypted text is easy.
• The algorithm modes which are used in block cipher are: ECB (Electronic Code Book) and CBC (Cipher Block Chaining).	• The algorithm modes which are used in • stream cipher are: CFB (Cipher Feedback) and OFB (Output Feedback).

Modern Symmetric-key ciphers

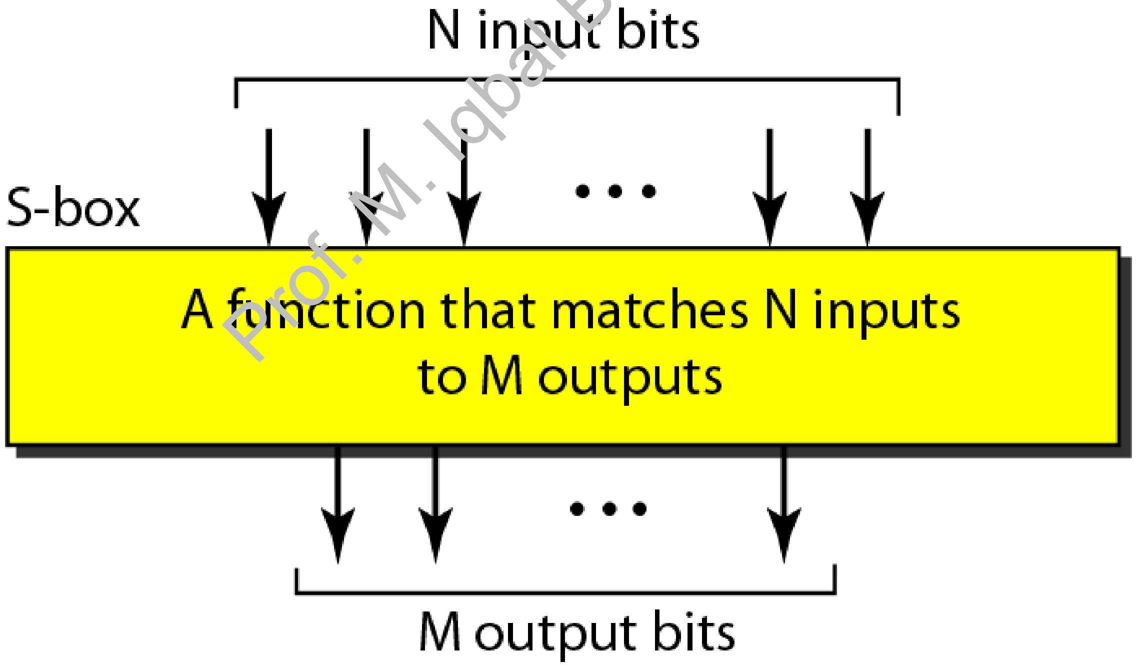
- **Modern Block ciphers:** A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key. The decryption algorithm must be the inverse of the encryption algorithm, and both operations must use the same secret key so that Bob can retrieve the message sent by Alice.



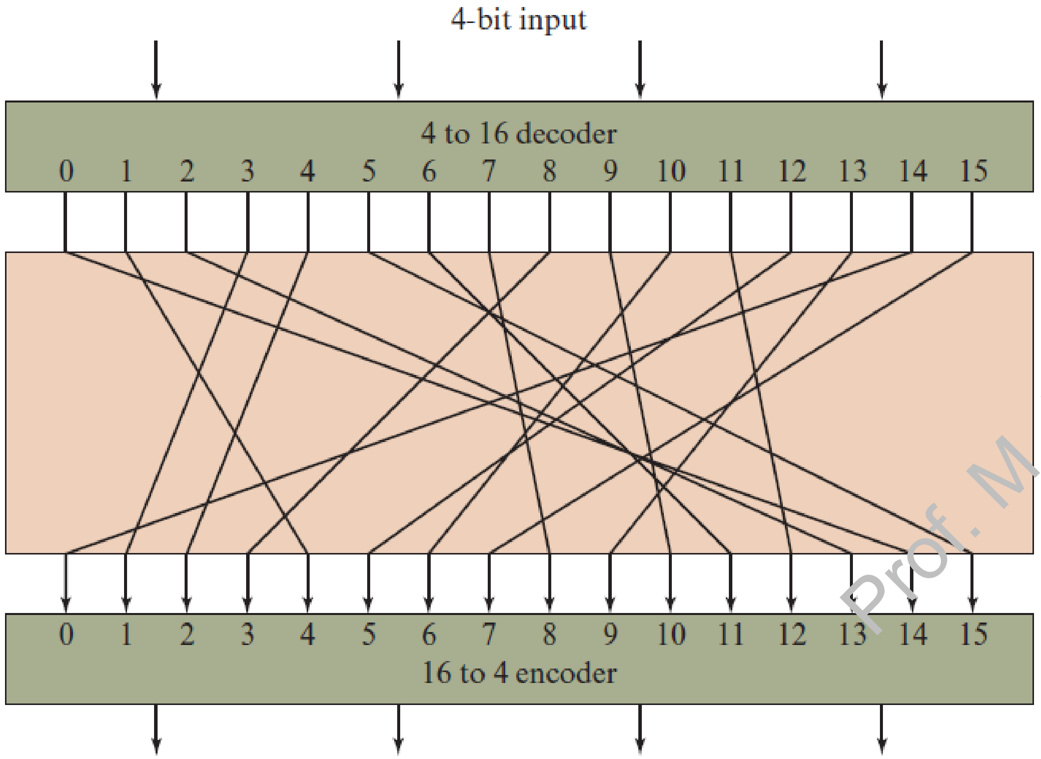
Components of Modern Block Ciphers

S-Box

An **S-box** (substitution box) can be thought of as a miniature substitution cipher, but it substitutes bits. Unlike the traditional substitution cipher, an S-box can have a different number of inputs and outputs.



S-Box

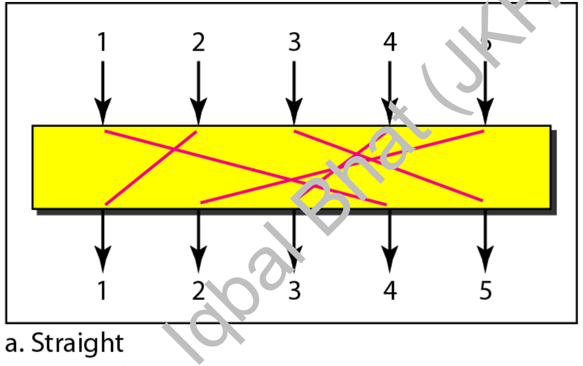


Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

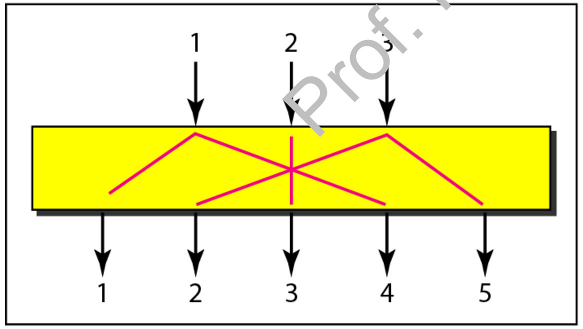
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

P-Box

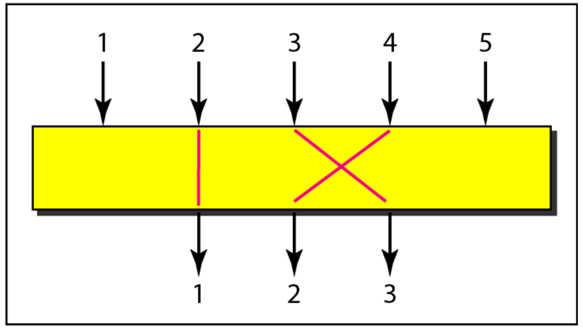
A **P-box (permutation box)** parallels the traditional transposition cipher for characters, but it transposes bits



a. Straight



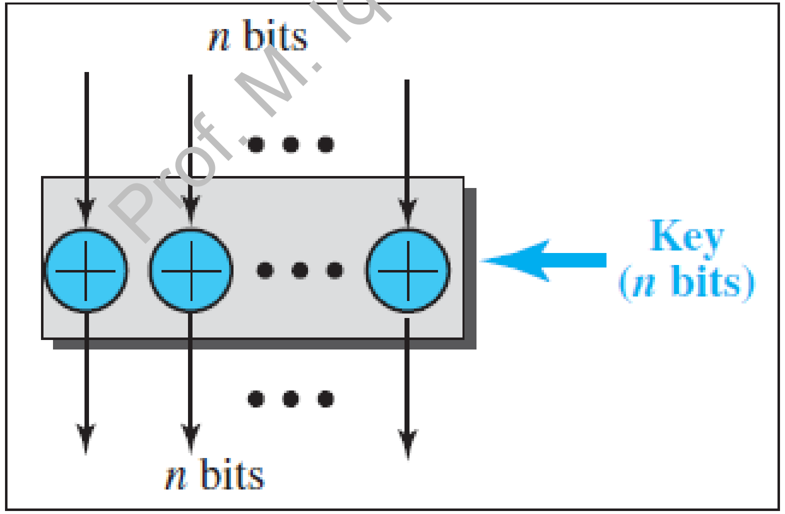
b. Expansion



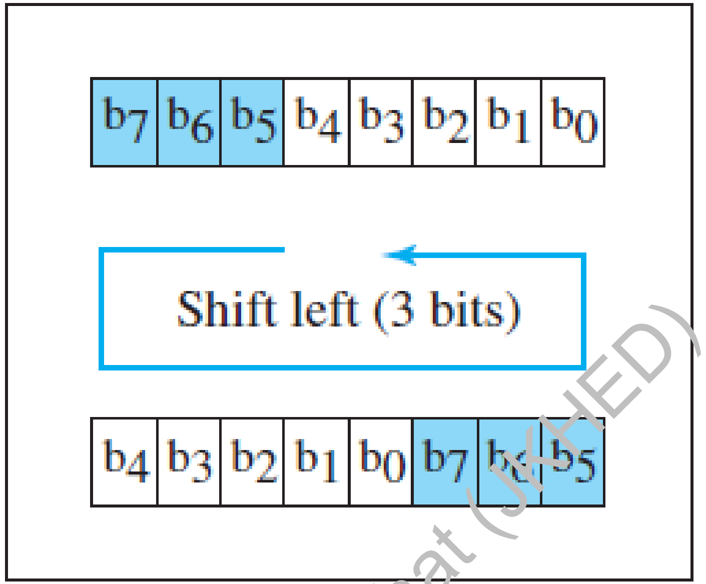
c. Compression

Exclusive-OR operation (XOR)

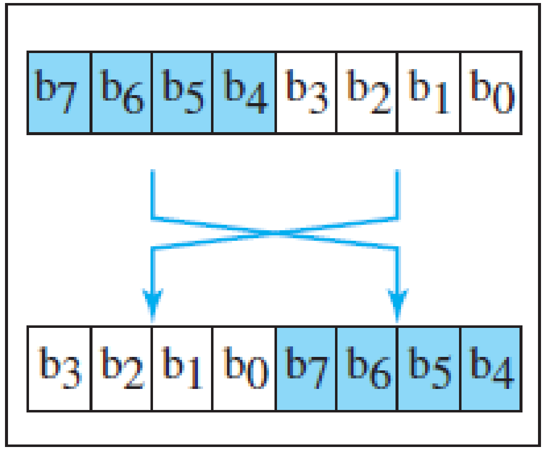
An important component in most block ciphers is the exclusive-OR operation, in which the output is 0 if the two inputs are the same, and the output is 1 if the two inputs are different. In modern block ciphers, we use n exclusive-OR operations to combine an n-bit data piece with an n-bit key. An exclusive-OR operation is normally the only unit where the key is applied. The other components are normally based on predefined functions.



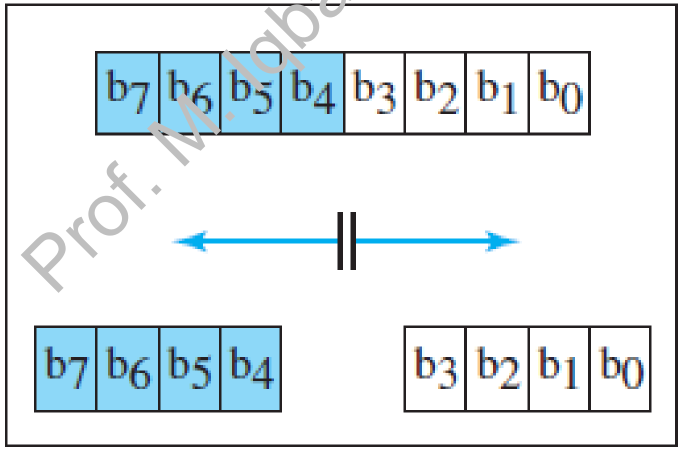
Exclusive-OR



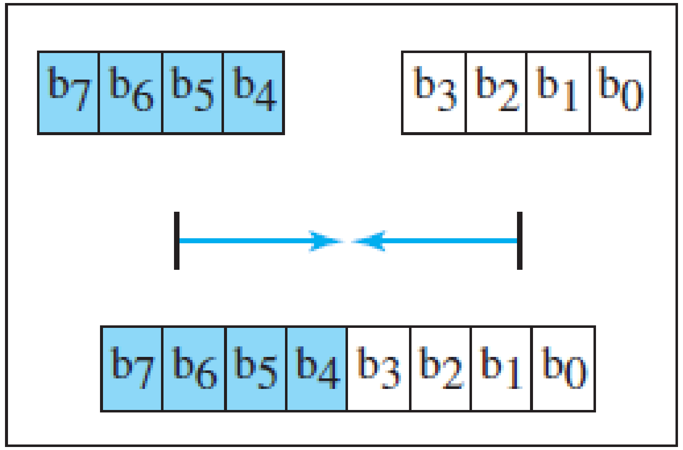
Shift



Swap



Split

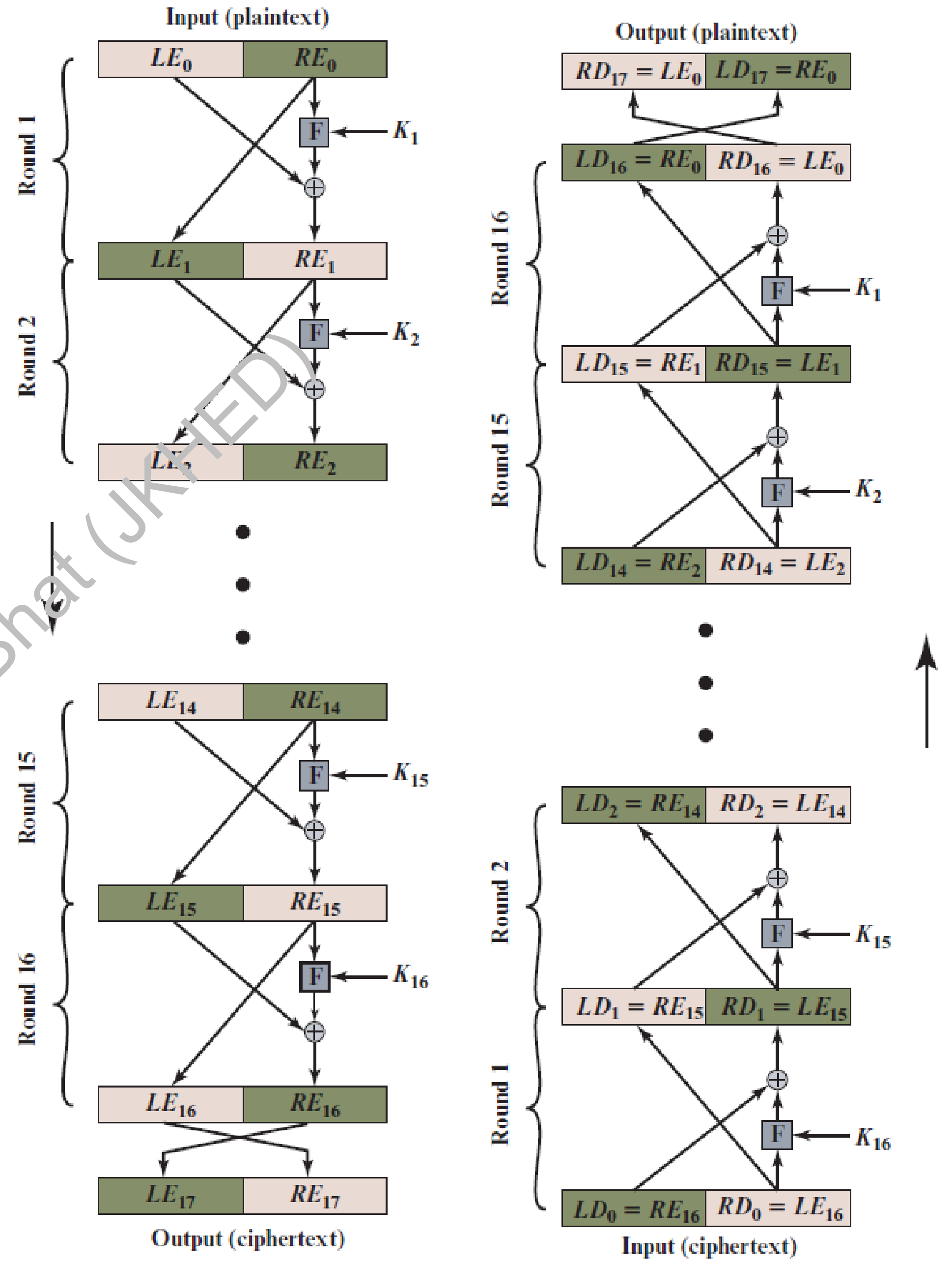


Combine

The Feistel Structure

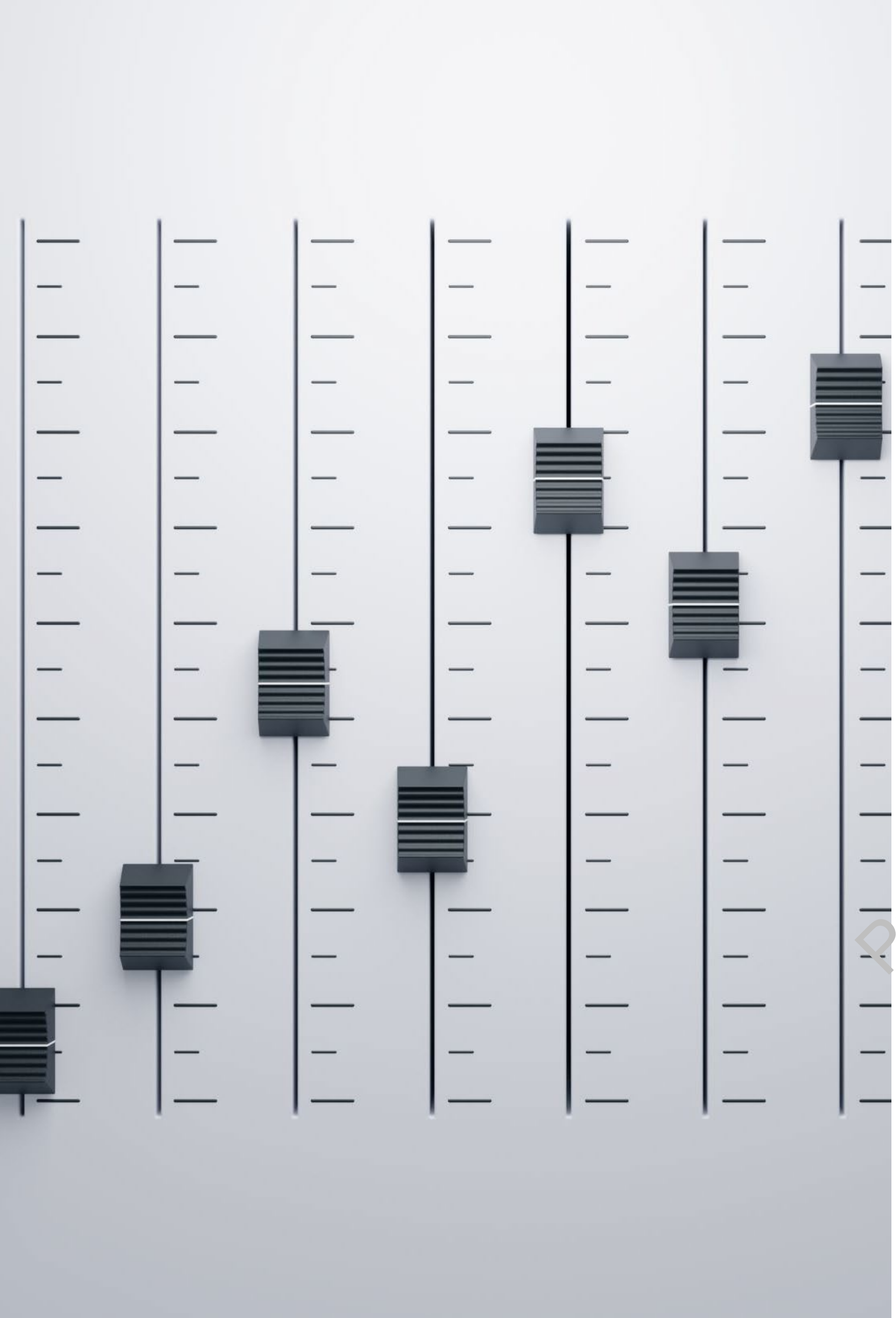
The Feistel Cipher

- Feistel proposed [FEIS73] that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.
- This design model can have invertible, non-invertible, and self-invertible components. Additionally, the Feistel block cipher uses the same encryption and decryption algorithms.



Feistel Cipher Structure

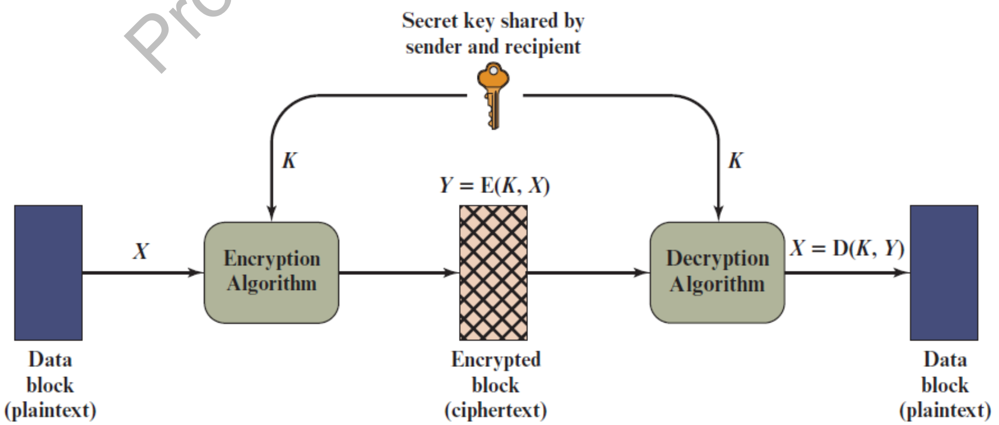
- **Block size:** larger block sizes mean greater security
- **Key Size:** larger key size means greater security
- **Number of rounds:** multiple rounds offer increasing security
- **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
- **Fast software encryption/decryption:** the speed of execution of the algorithm becomes a concern



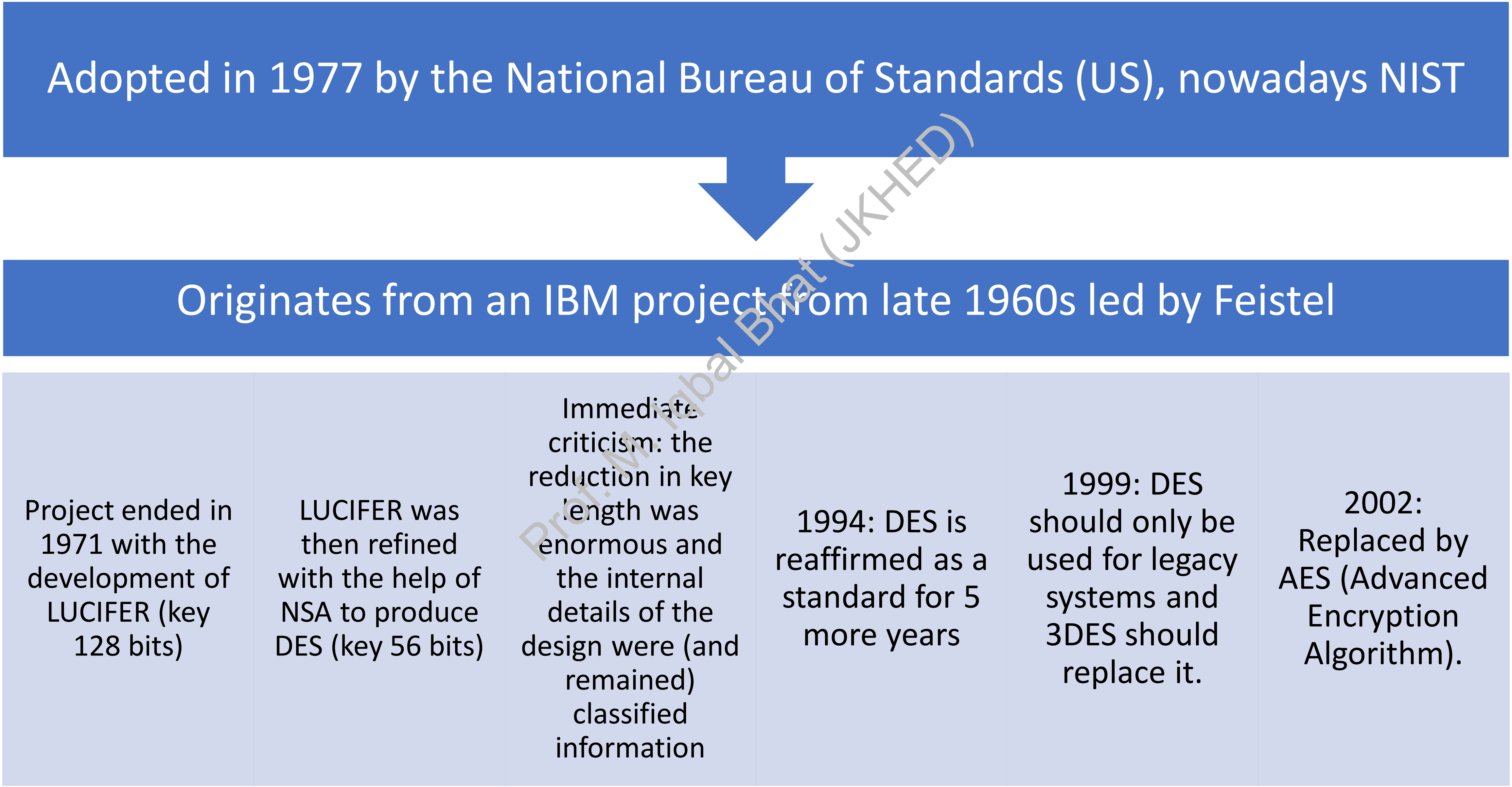
Sub key

- Sub keys are created from the original key by a key expansion algorithm designed for multiple-round ciphers called a **key schedule**. A popular method of combining a sub key with data is bitwise XOR. In each round, after the key mixing, the data is scrambled further using substitution and permutation functions.

DES
(Data Encryption Standard)



DES



Adopted in 1977 by the National Bureau of Standards (US), nowadays NIST

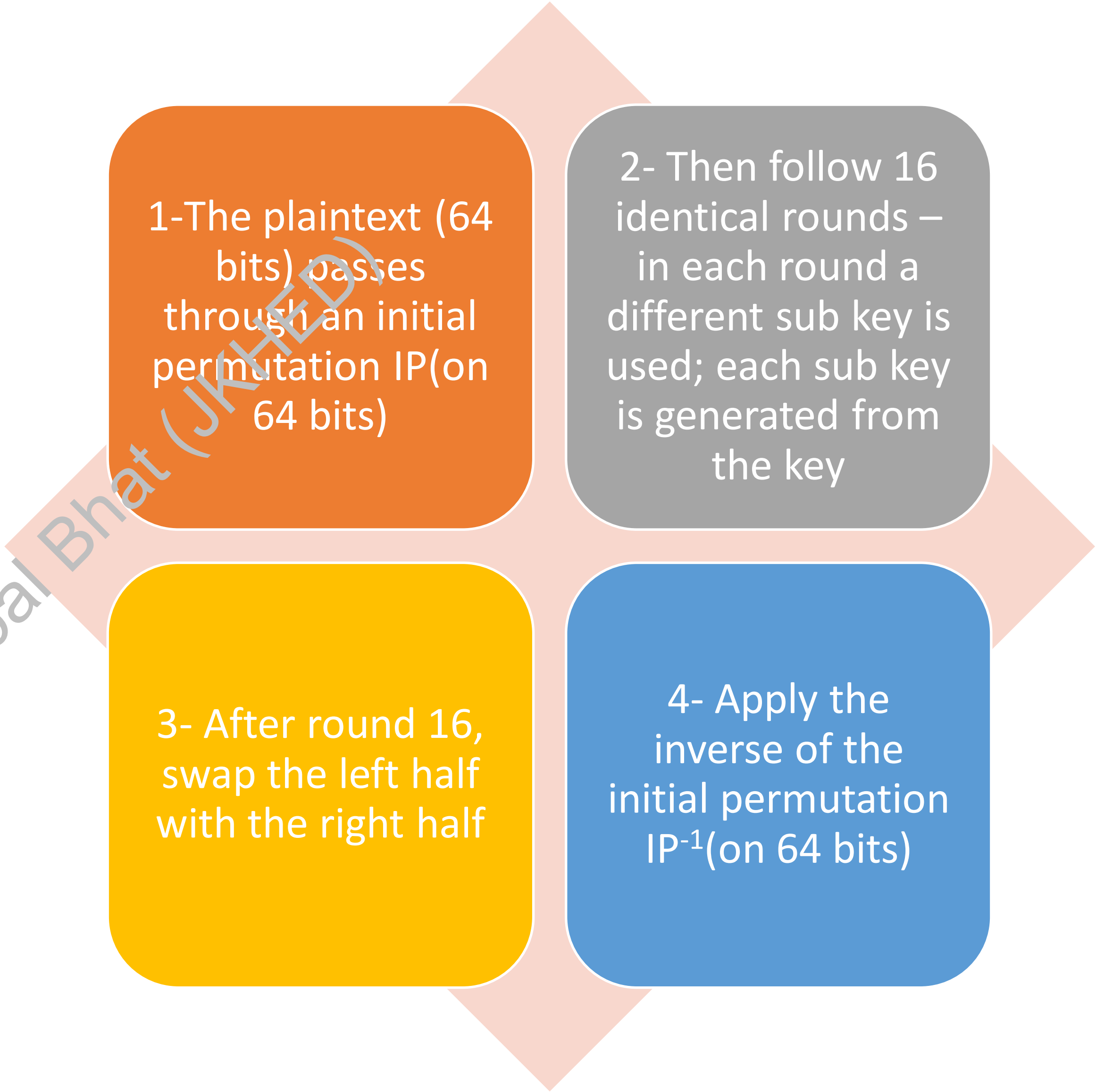
Originates from an IBM project from late 1960s led by Feistel

- Project ended in 1971 with the development of LUCIFER (key 128 bits)
- LUCIFER was then refined with the help of NSA to produce DES (key 56 bits)
- Immediate criticism: the reduction in key length was enormous and the internal details of the design were (and remained) classified information
- 1994: DES is reaffirmed as a standard for 5 more years
- 1999: DES should only be used for legacy systems and 3DES should replace it.
- 2002: Replaced by AES (Advanced Encryption Algorithm).

DES

- Data Encryption Standard (DES)
  - The most widely used encryption scheme
  - The algorithm is reffered to the Data Encryption Algorithm (DEA)
  - DES is a block cipher
  - The plaintext is processed in 64-bit blocks
  - The key is 56-bits in length

DES encryption scheme



1-The plaintext (64 bits) passes through an initial permutation IP(on 64 bits)

2- Then follow 16 identical rounds – in each round a different sub key is used; each sub key is generated from the key

3- After round 16, swap the left half with the right half

4- Apply the inverse of the initial permutation IP^{-1}(on 64 bits)

Figure 30.13 *DES*

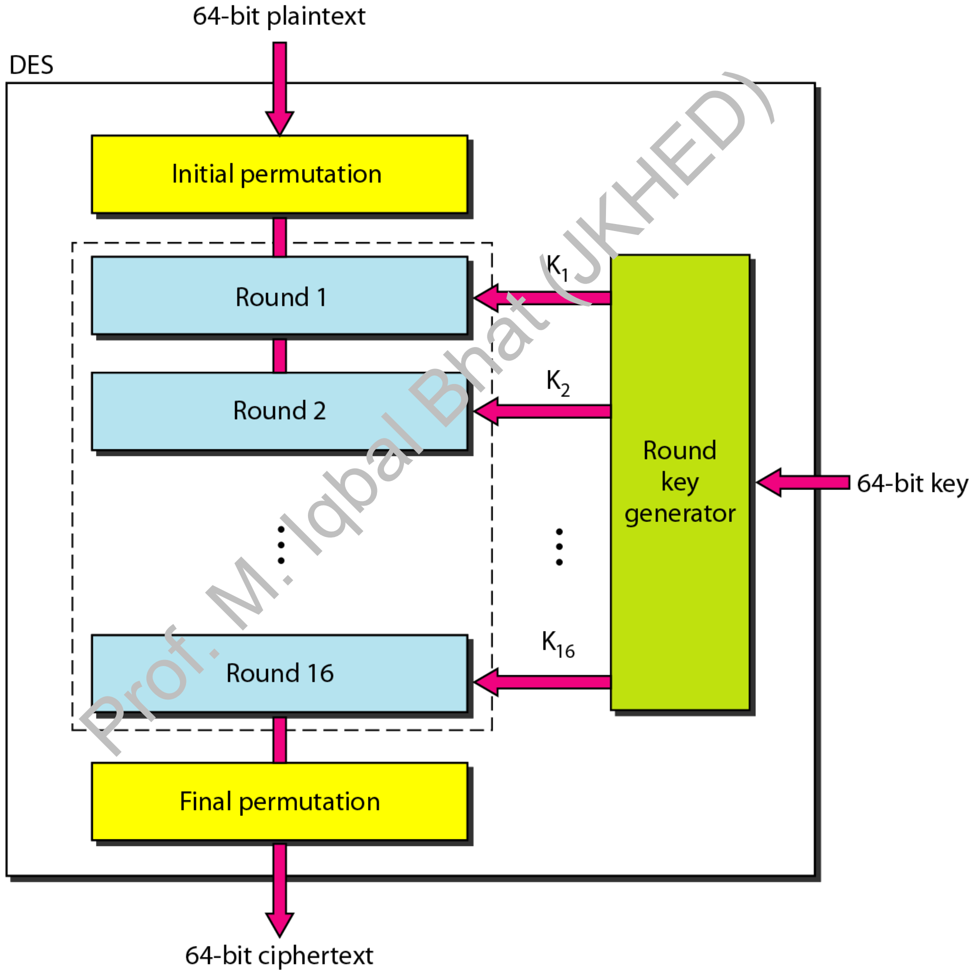


Figure 30.13 *DES*

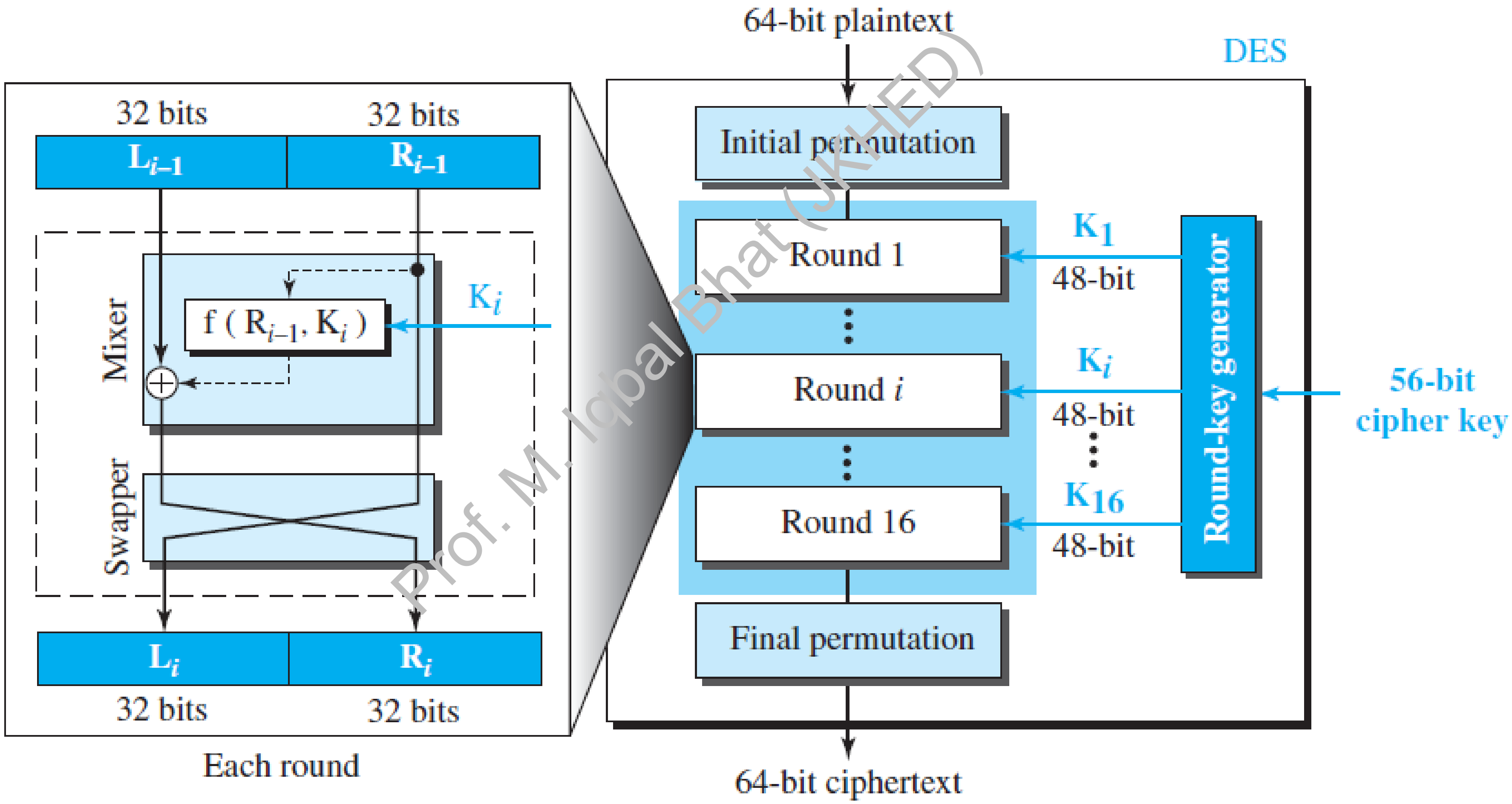
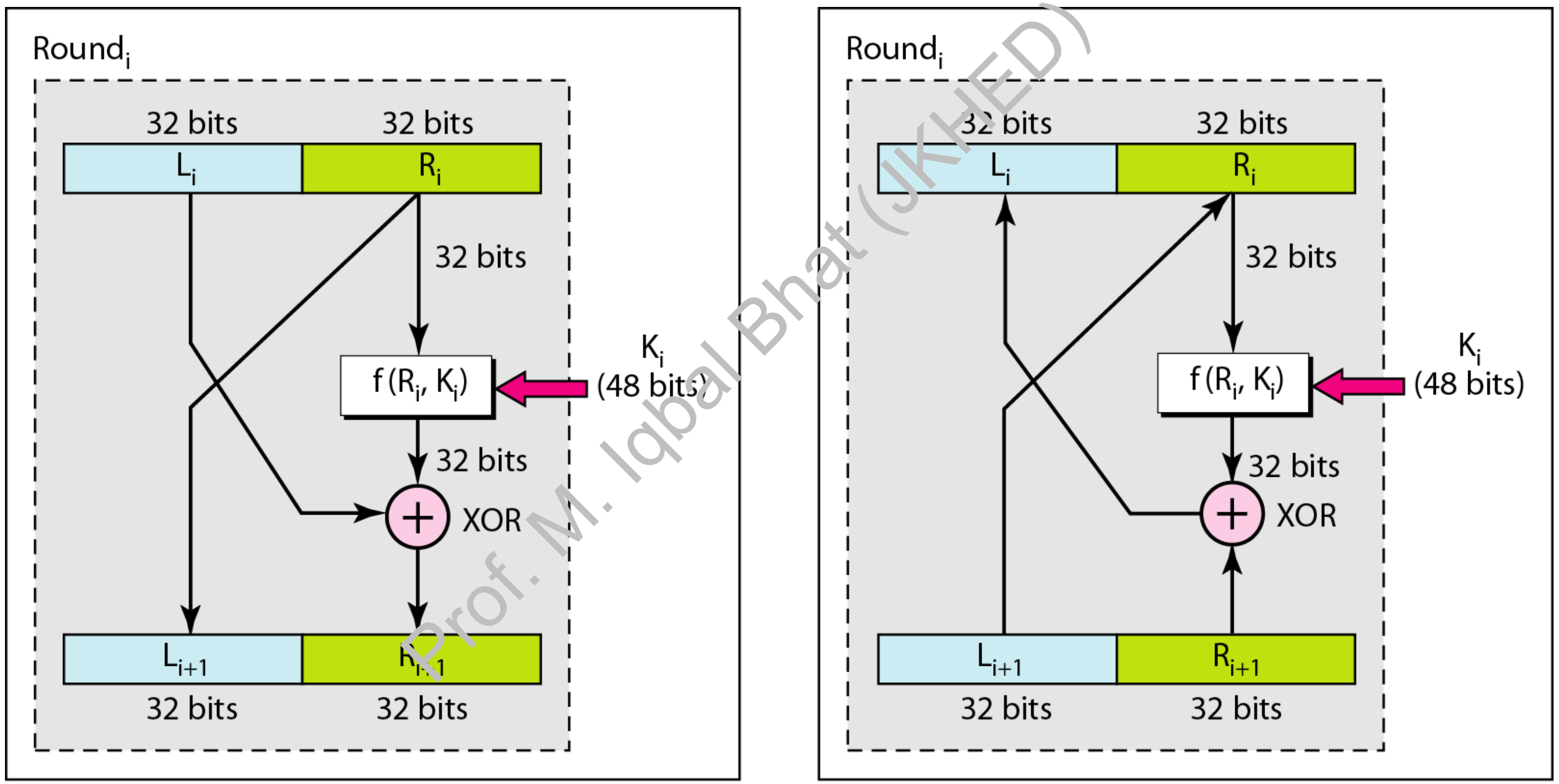


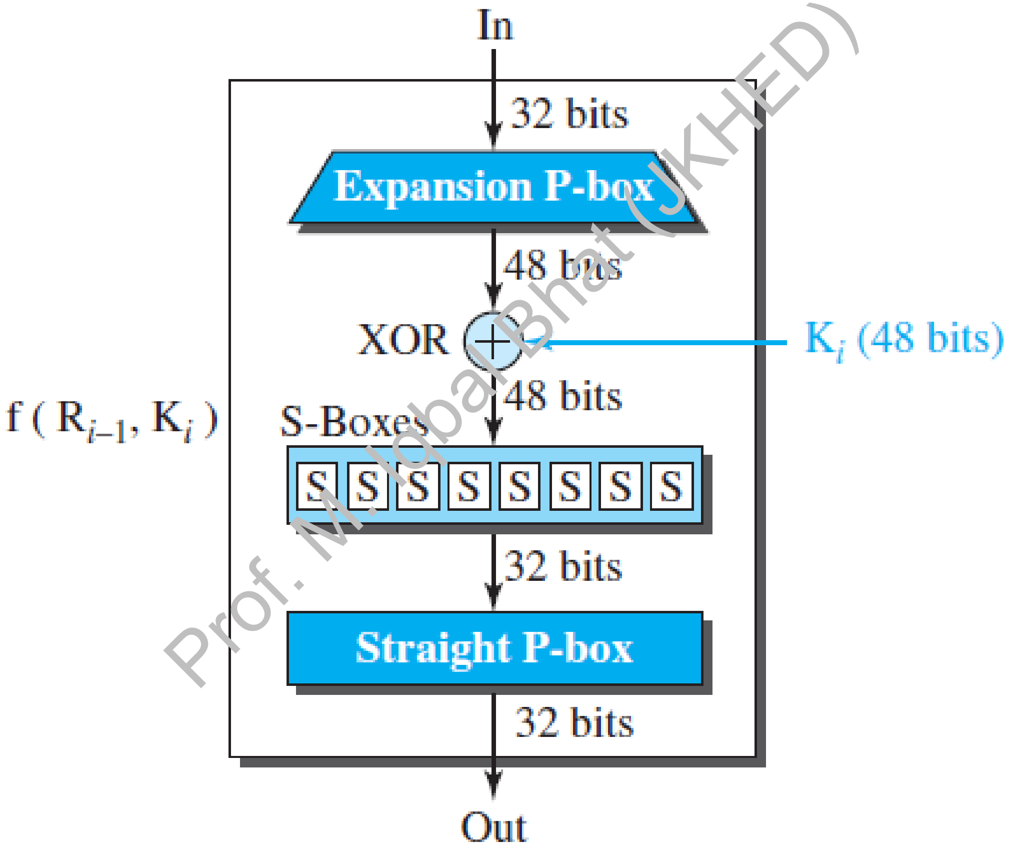
Figure 30.14 *One round in DES ciphers*



a. Encryption round

b. Decryption round

DES Function



DES function

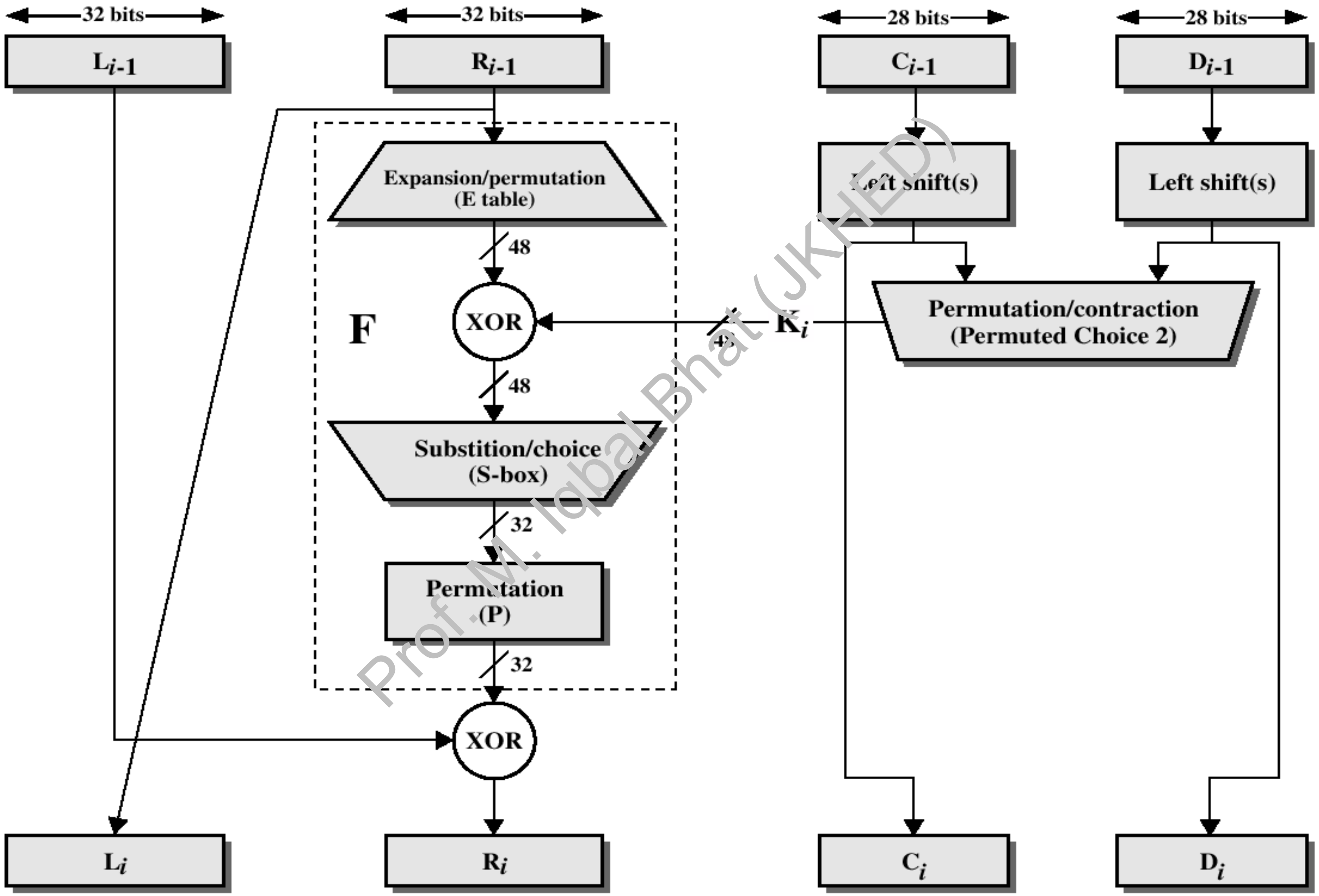


Figure 2.4 Single Round of DES Algorithm

Sub key generation

Before round 1 of DES, they key is permuted according to a table labeled Permuted Choice One –the resulting 56-bit key is split into its two 28-bit halves labeled C0and D0

In each round, Ci-1 and Di-1 are separately subjected to a circular left shift of one or two bits according to the table on the next slide –the shifted values will be input to next round

The shifted values serve as input to Permuted Choice Two which produces a 48-bit output: the sub key of the current round

Example of DES

M = 0123456789ABCDEF

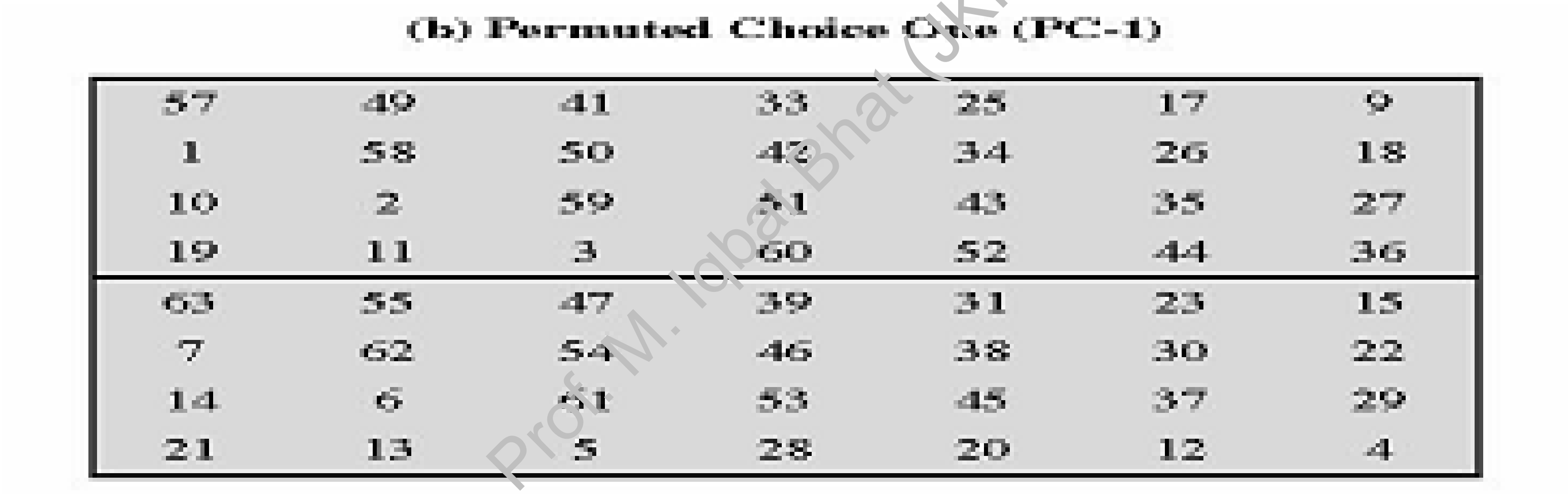
- **M** = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

K = 133457799BBCDFF1

- **K** = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Step 1: Create 16 sub keys, each of which is 48-bits long.

- In the general scheme of DES is shown that a 64-bit key is used –the bits of the key are numbered from 1 to 64.
- The algorithm ignores every 8, 16, 24, 32, 40, 48, 56, and 64 bit –thus, the key for DES is effectively 56-bit long



(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

Sub keys cont'd

Example: From the original 64-bit key

- **K** = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001
- we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, ***C0*** and ***D0***, where each half has 28 bits.

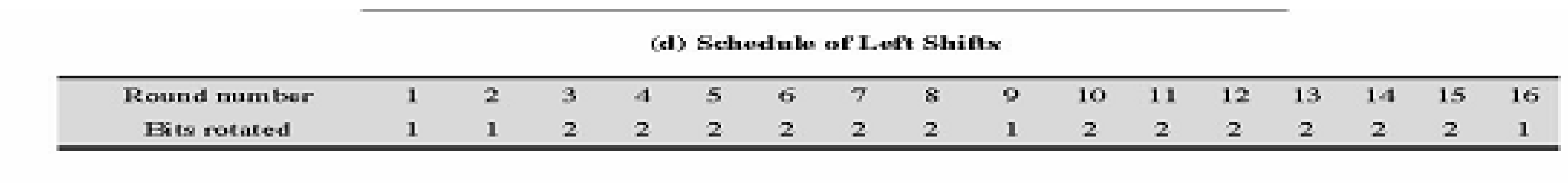
Example: From the permuted key **K+**, we get

C0 = 1111000 0110011 0010101 0101111
D0 = 0101010 1011001 1001111 0001111

C1 = 1110000110011001010101011111
D1 = 1010101011001100111100011110
C2 = 1100001100110010101010111111
D2 = 0101010110011001111000111101
C3 = 0000110011001010101011111111
D3 = 0101011001100111100011110101
C4 = 0011001100101010101111111100
D4 = 0101100110011110001111010101
C5 = 1100110010101010111111110000
D5 = 0110011001111000111101010101
C6 = 0011001010101011111111000011
D6 = 1001100111100011110101010101
C7 = 1100101010101111111100001100
D7 = 0110011110001111010101010110

…

……….

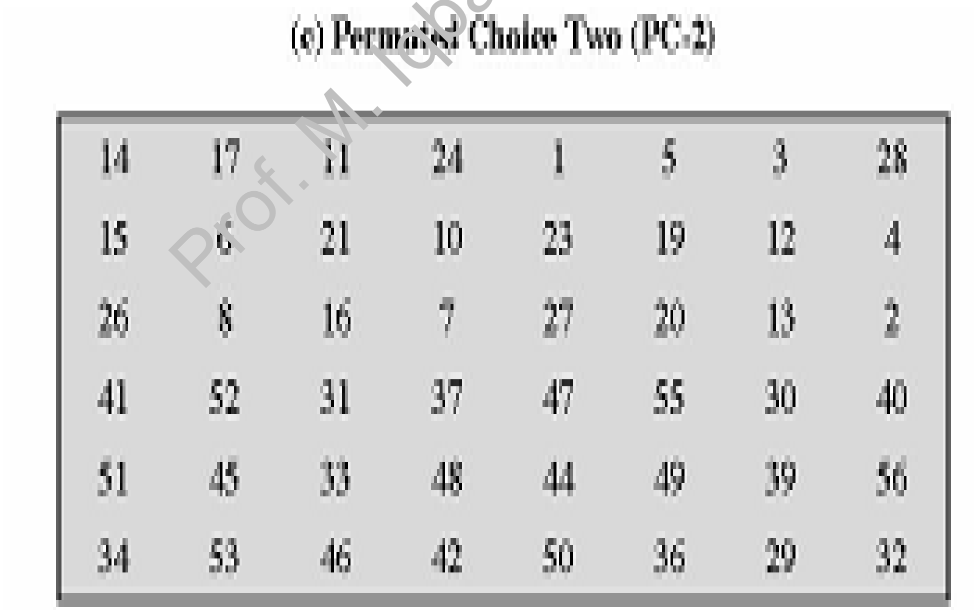


(d) Schedule of Left Shifts

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

Sub key contd

- We now form the keys ***Kn***, for 1<=***n***<=16, by applying the following permutation table to each of the concatenated pairs ***CnDn***. Each pair has 56 bits, but **PC-2** only uses 48 of these
- **Example:** For the first key we have ***C1D1*** = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110
- which, after we apply the permutation **PC-2**, becomes
- ***K1*** = 000110 110000 001011 101111 111111 000111 000001 110010



(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

Sub key gnerated

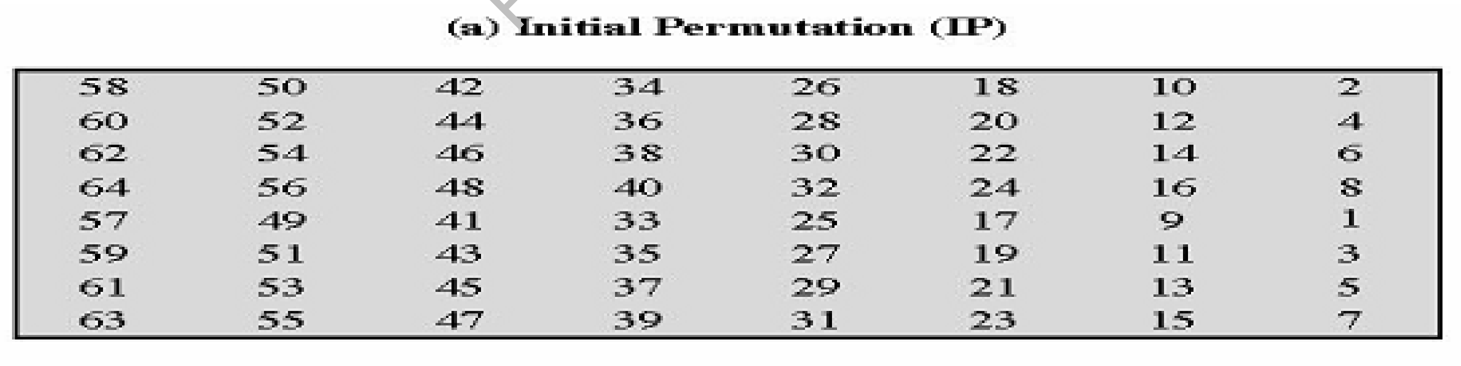
- For the other keys we have
- ***K2*** = 011110 011010 111011 011001 110110 111100 100111 100101
  K3 = 010101 011111 110010 001010 010000 101100 111110 011001
  K4 = 011100 101010 110111 010110 110110 110011 010100 011101
  K5 = 011111 001110 110000 000111 111010 110101 001110 101000
  K6 = 011000 111010 010100 111110 010100 000111 101100 101111
  K7 = 111011 001000 010010 110111 111101 100001 100010 111100
  K8 = 111101 111000 101000 111010 110000 010011 101111 111011
  K9 = 111000 001101 101111 101011 111011 011110 011110 000001
  K10 = 101100 011111 001101 000111 101110 100100 011001 001111
  K11 = 001000 010101 111111 010011 110111 101101 001110 000110
  K12 = 011101 010111 000111 110101 100101 000110 011111 101001
  K13 = 100101 111100 010111 010001 111110 101011 101001 000001
  K14 = 010111 110100 001110 110111 111100 101110 011100 111010
  K15 = 101111 111001 000110 001101 001111 010011 111100 001010
  K16 = 110010 110011 110110 001011 000011 100001 011111 110101

Step 2: Encode each 64-bit block of data

- **M** = 0123456789ABCDEF
  - **M** = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

- There is an *initial permutation* **IP** of the 64 bits of the message data **M**. This rearranges the bits according to the following table
- **Example:** Applying the initial permutation 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 = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010

  1010



(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

Step 2 contd

- Next divide the permuted block **IP** into a left half ***L0*** of 32 bits, and a right half ***R0*** of 32 bits.
- **Example:** From **IP**, we get ***L0*** and ***R0***
- ***L0*** = 1100 1100 0000 0000 1100 1100 1111 1111
  R0 = 1111 0000 1010 1010 1111 0000 1010 1010
- for 1<=***n***<=16, using a function ***f*** which operates on two blocks--a data block of 32 bits and a key ***Kn*** of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, T**hen for **n** going from 1 to 16 we calculate
- $L_n = R_n\text{-}1$
  $R_n = L_n\text{-}1 + f(R_n\text{-}1, K_n)$
- **Example:** For ***n*** = 1, we have
  - ***K1*** = 000110 110000 001011 101111 111111 000111 000001 110010
    L1 = ***R0*** = 1111 0000 1010 1010 1111 0000 1010 1010
    R1 = ***L0*** + ***f***(***R0***,***K1***)

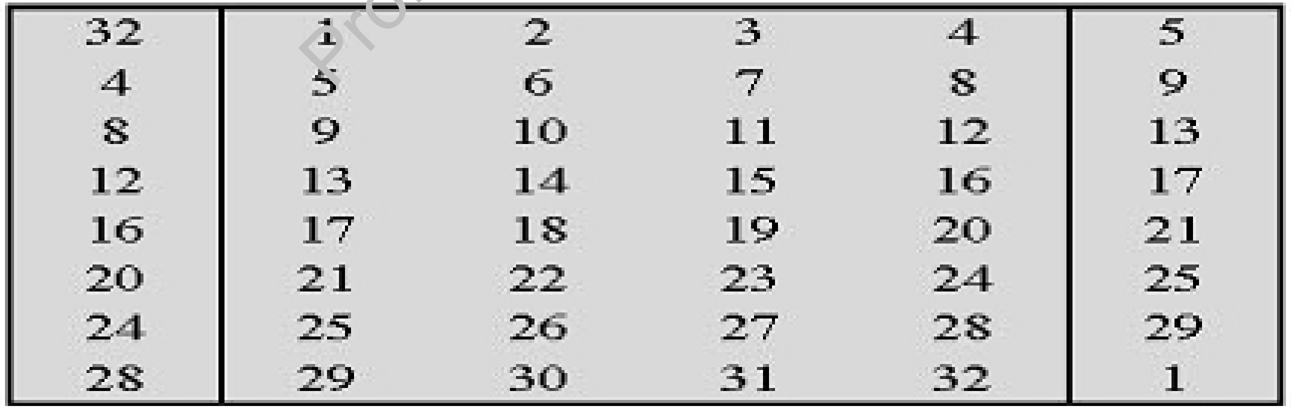
Step 2 contd

- To calculate ***f***, we first expand each block ***Rn-1*** from 32 bits to 48 bits. This is done by using a selection table
- **Example:** We calculate **E**(***R0***) from ***R0*** as follows:
- ***R0*** = 1111 0000 1010 1010 1111 0000 1010 1010

E(***R0***) = 011110 100001 010101 010101 011110 100001 010101 010101

- Note that each block of 4 original bits has been expanded to a block of 6 output bits

(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

Step 2 contd

- Next in the ***f*** calculation, we XOR the output **E(*Rn-1*)** with the key ***Kn***:
- ***Kn*** + **E(*Rn-1*)**.
- **Example:** For ***K1*** , **E(*R0*)**, we have
- ***K1*** = 000110 110000 001011 101111 111111 000111 000001 110010
  E(*R0*) = 011110 100001 010101 010101 011110 100001 010101 010101
  K1+**E(*R0*)** = 011000 010001 011110 111010 100001 100110 010100 100111
- We have not yet finished calculating the function ***f*** . To this point we have expanded ***Rn-1*** from 32 bits to 48 bits, using the selection table, and XORed the result with the key ***Kn*** . We now have 48 bits, or eight groups of six bits.

Step 2 contd

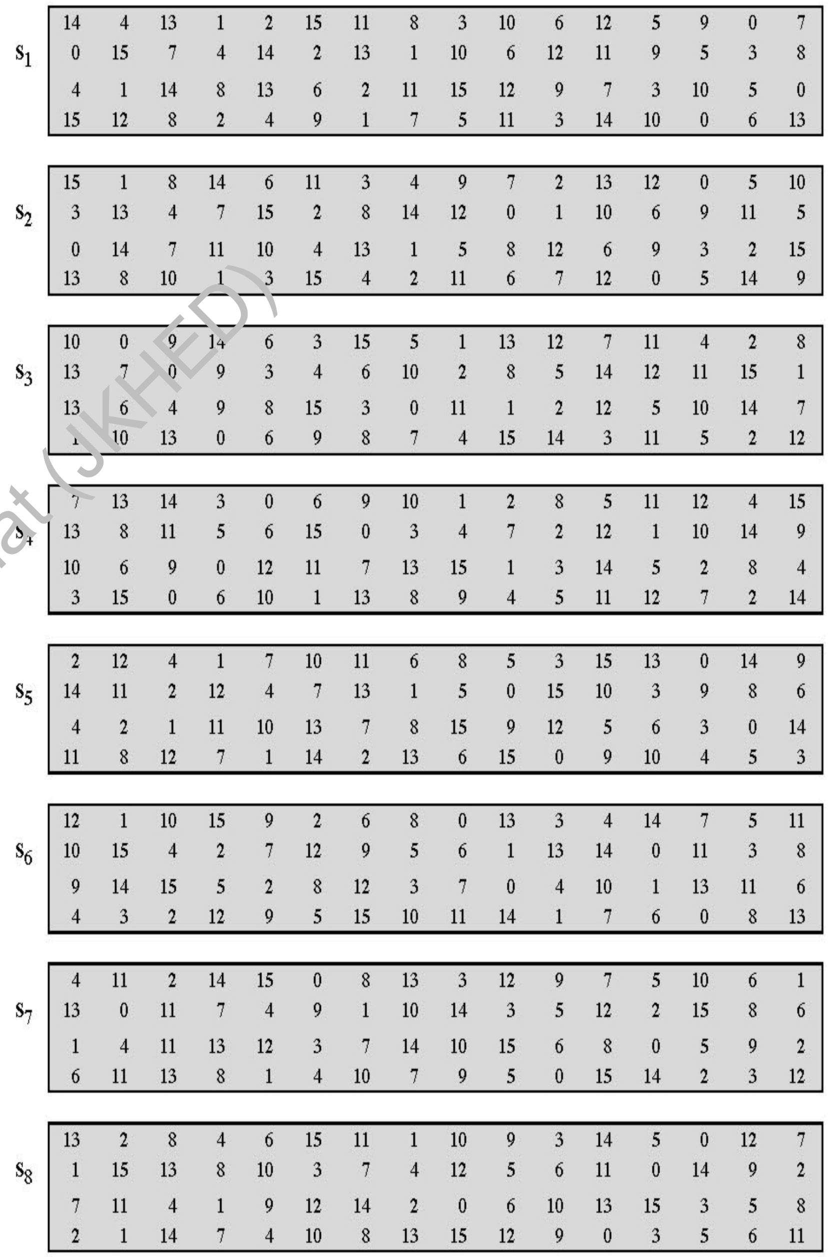
- with each group of six bits: we use them as addresses in tables called "**S boxes**"
- Write the previous result, which is 48 bits, in the form:
- ***Kn*** + **E**(***Rn-1***) =***B1B2B3B4B5B6B7B8***,
- where each ***Bi*** is a group of six bits. We now calculate
- ***S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)***

S-boxes

- Example: consider the input 011001 to S-box S1
- The row is **0**1100**1:01**(i.e. 1)
- The column is 0**1100**1: **1100** (i.e. 12)
- The value in the selected cell is 9Output is 1001
- **Example:** For the first round, we obtain as the output of the eight **S** boxes:
- ***K1*** + **E**(***R0***) = 011000 010001 011110 111010 100001 100110 010100 100111.
- ***S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)*** = 0101 1100 1000 0010 1011

  0101 1001 0111

Table 3.3 Definition of 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

Step 2 contd

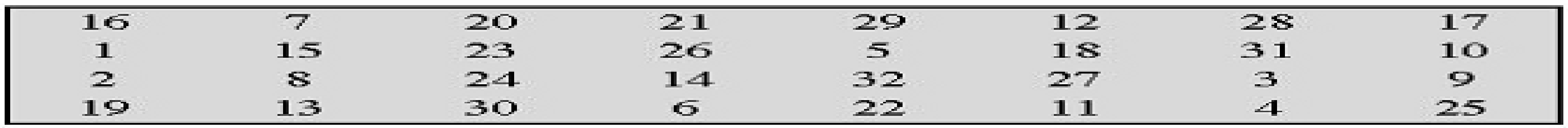
- The final stage in the calculation of ***f*** is to do a permutation **P** of the **S**-box output to obtain the final value of ***f***:
- ***f*** = **P**(***S1(B1)S2(B2)...S8(B8)***)
- **P** yields a 32-bit output from a 32-bit input by permuting the bits of the input block
- **Example:** From the output of the eight **S** boxes:
- ***S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)*** = 0101 1100 1000 0010 1011 0101 1001 0111

  we get ***f*** = 0010 0011 0100 1010 1010 1001 1011 1011

- ***R1*** = ***L0*** + ***f***(***R0*** , ***K1***)
  = 1100 1100 0000 0000 1100 1100 1111 1111
  + 0010 0011 0100 1010 1010 1001 1011 1011

  = 1110 1111 0100 1010 0110 0101 0100 0100

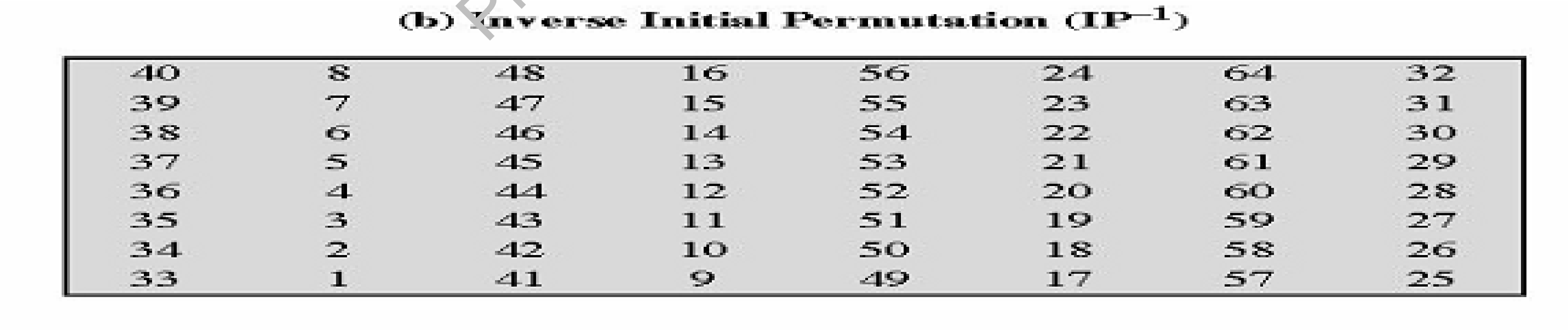
(d) Permutation Function (P)



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

Step 2 contd

- We then ***reverse*** the order of the two blocks into the 64-bit block ***R16L16***
- and apply a final permutation **IP-1** as defined by the following table:



(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

Step 2 contd

- **Example:** If we process all 16 blocks using the method defined previously, we get, on the 16th round,
- ***L16*** = 0100 0011 0100 0010 0011 0010 0011 0100
  R16 = 0000 1010 0100 1100 1101 1001 1001 0101
- We reverse the order of these two blocks and apply the final permutation to
- ***R16L16*** = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100
- ***IP-1*** = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101
- which in hexadecimal format is
- 85E813540F0AB405.

Strength of DES

Two main concerns with DES: **the length of the key** and **the nature of the algorithm**

- The key is rather short: 56 bits –
  - In average, only half of the keys have to be tried to break the system
  - In principle it should take long time to break the system
  - Things are quicker with dedicated hardware: 1998 –a special machine was built for less than 250 000 $ breaking DES in less than 3 days, 2006 –estimates are that a hardware costing around 20.000$ may break DES within a day

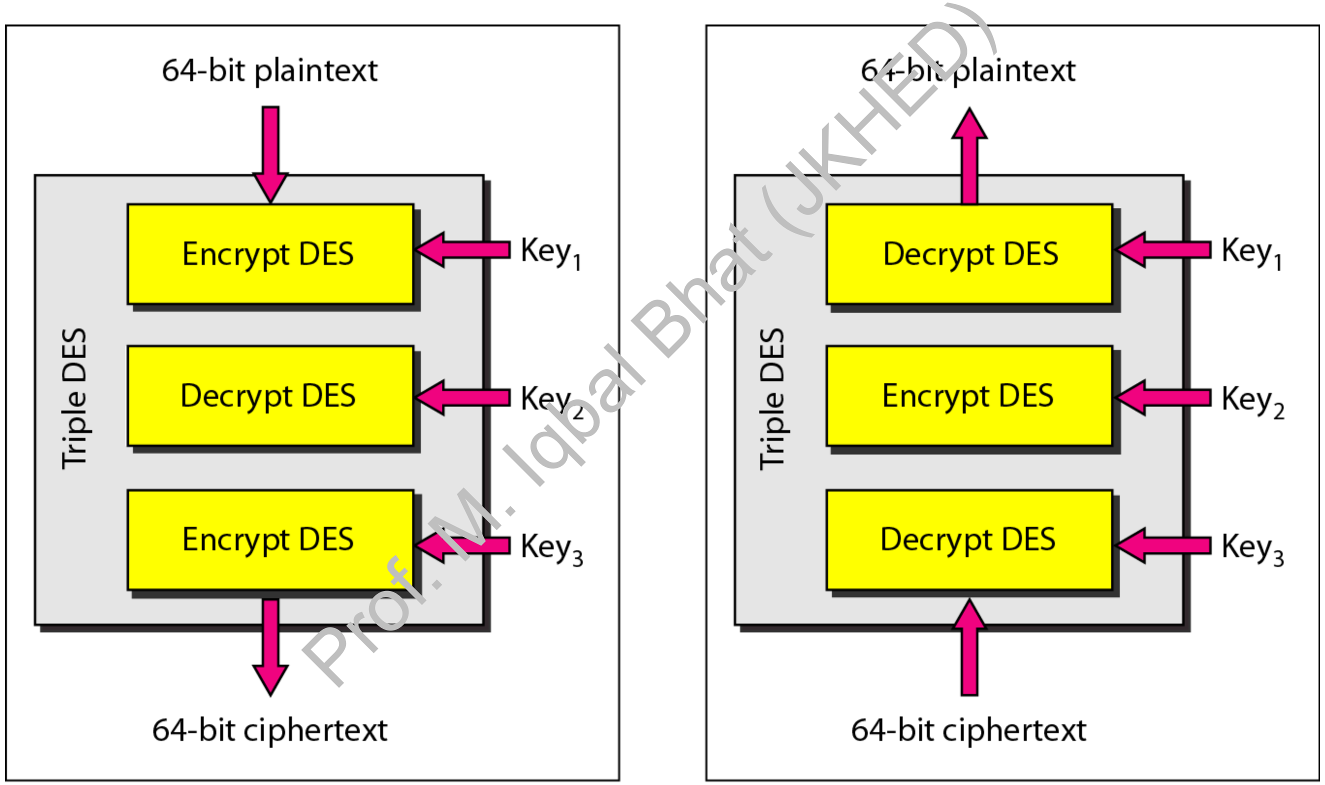
Strength of DES

- **Nature of the algorithm**
- There has always been a concern about the design of DES, especially about the design of S-boxes –perhaps they have been designed in such a way as to ensure a trapdoor to the algorithm –break it without having to search for the key
- The design criteria for the S-boxes (and for the rest of the algorithm) have been *classified information* and **NSA** was involved in the design
- Many regularities and unexpected behavior of the S-boxes have been reported
- On the other hand, changing the S-boxes slightly seems to weaken the algorithm
- No fatal weaknesses in the S-boxes have been (publicly) reported so far

Analysis of DES

- **Avalanche effect**: this is a desirable property of any encryption algorithm
- A small change (even 1 bit) in the plaintext should produce a significant change in the ciphertext
- Example: consider two blocks of 64 zeros and in the second block rewrite 1 on the first position. Encrypt them both with DES: depending on the key, the result may have 34 different bits!
- A small change (even 1 bit) in the key should produce a significant change in the ciphertext
- Example: a change of one bit in the DES key may produce 35 different bits in the encryption of the same plaintext

Figure 30.16 *Triple DES*



a. Encryption Triple DES

b. Decryption Triple DES

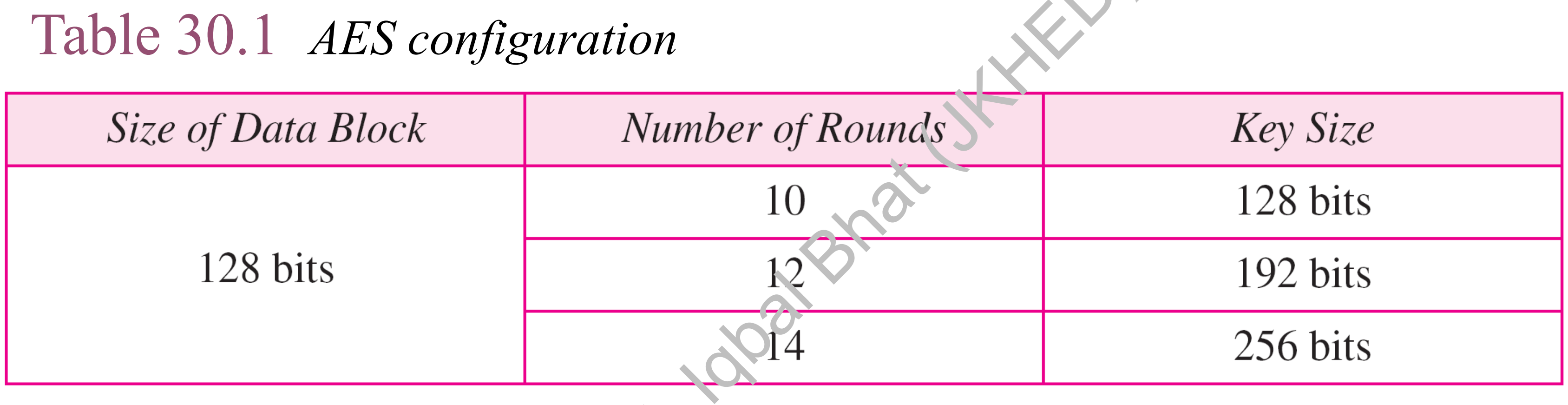


Table 30.1 *AES configuration*

Size of Data Block	*Number of Rounds*	*Key Size*
128 bits	10	128 bits
	12	192 bits
	14	256 bits

Note

AES has three different configurations with respect to the number of rounds and key size.

Figure 30.17 *AES*

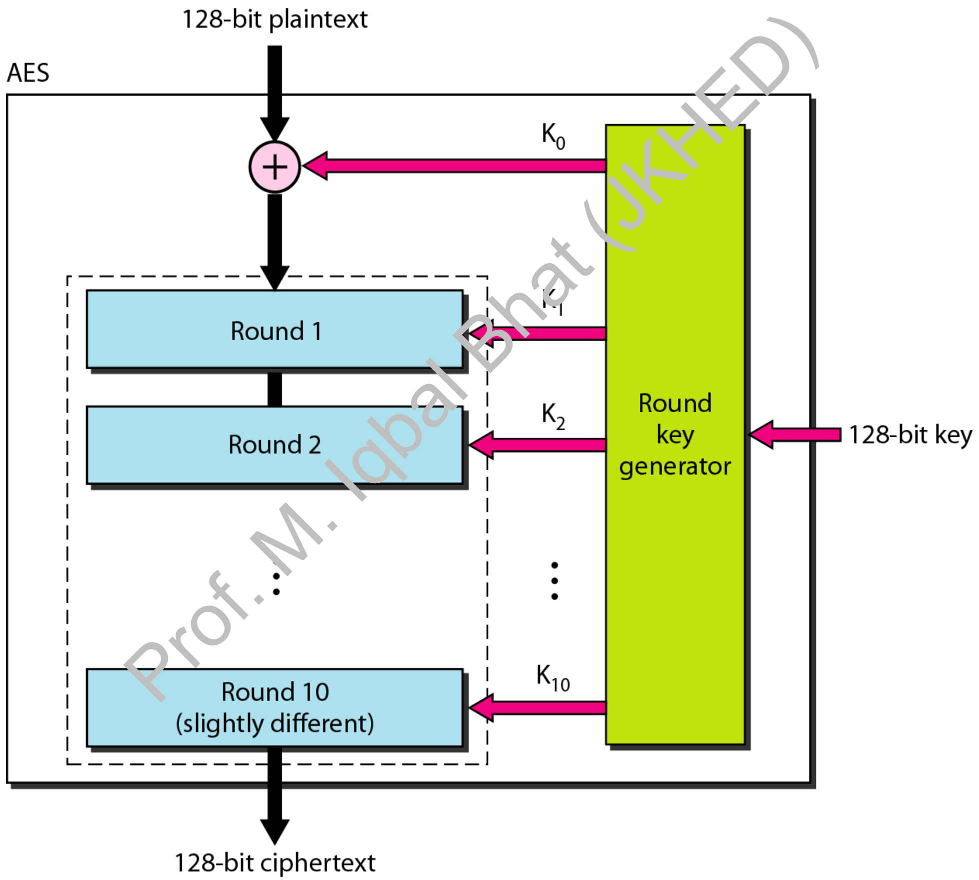


Figure 30.18 *Structure of each round*

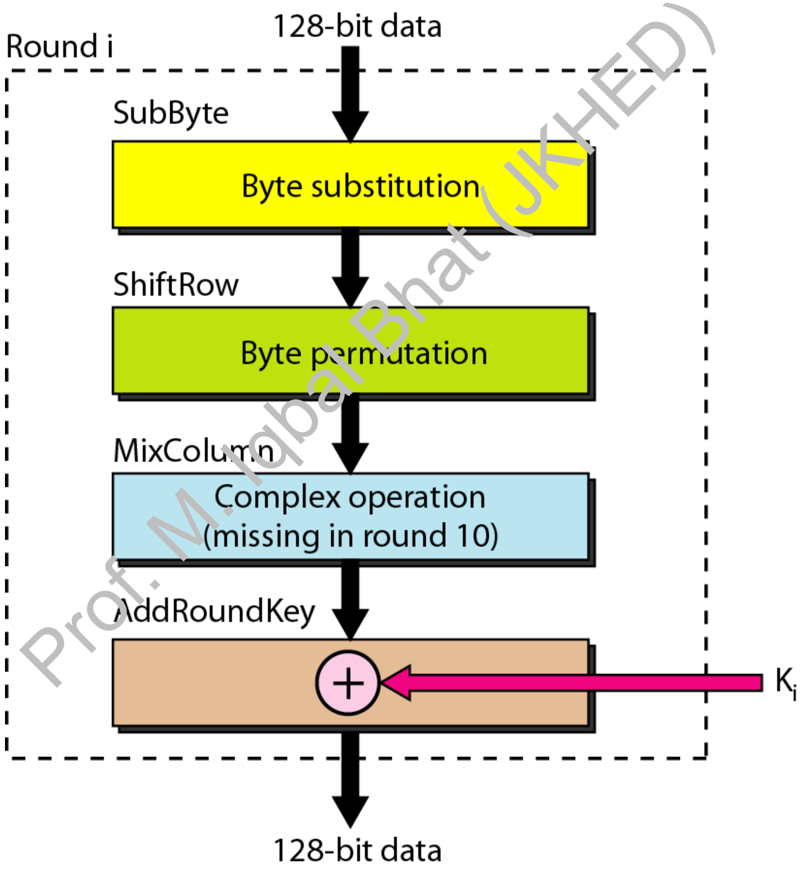


Figure 30.19 *Modes of operation for block ciphers*

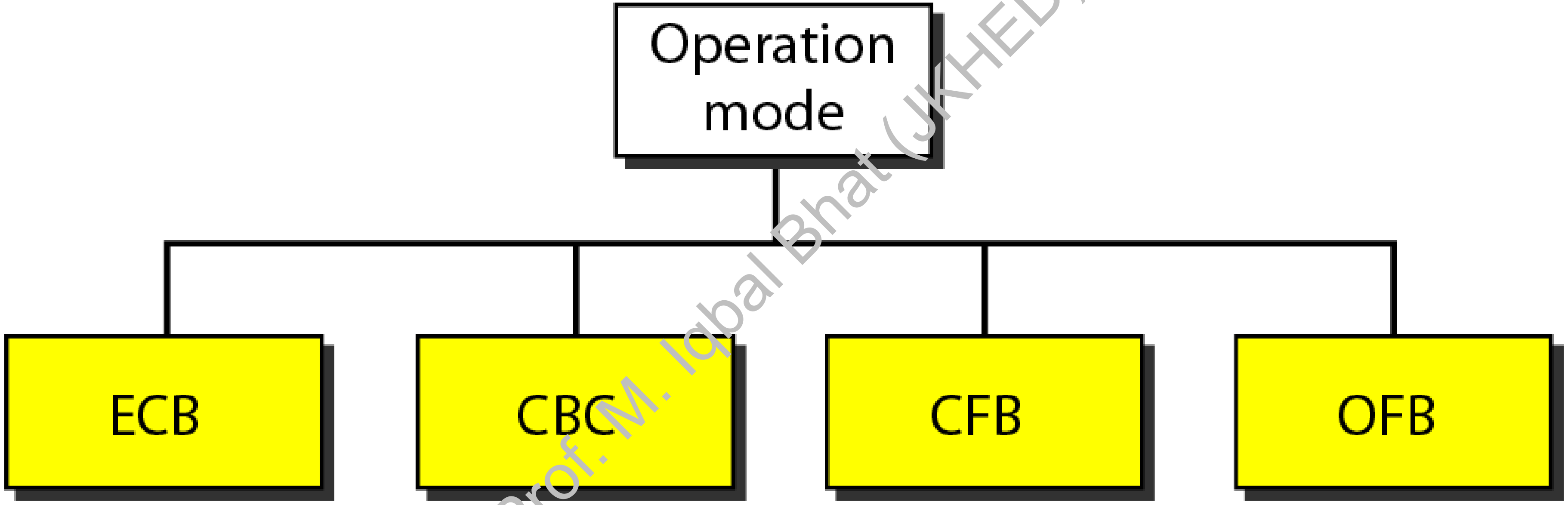


Figure 30.20 *ECB mode*

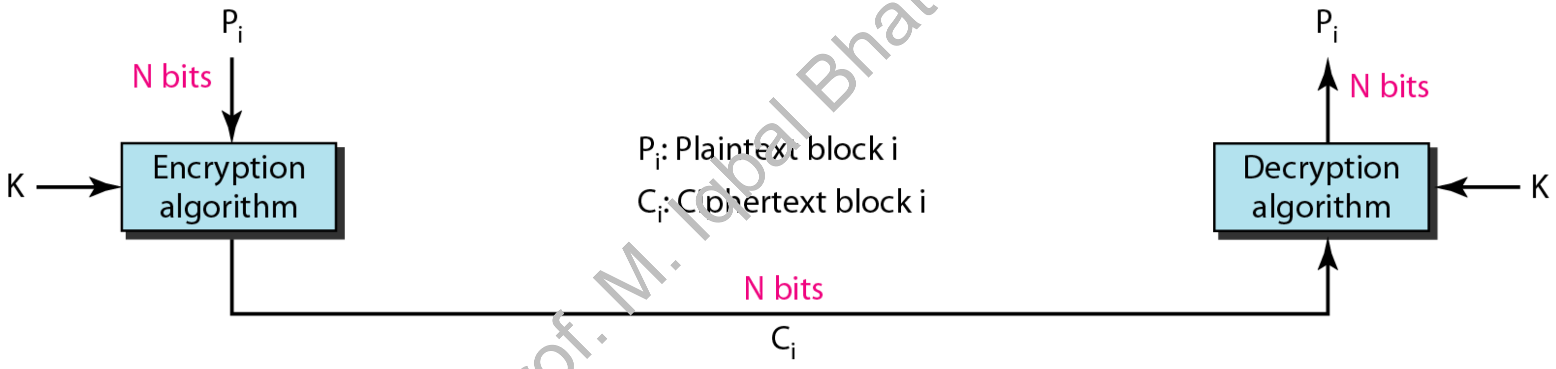


Figure 30.21 *CBC mode*

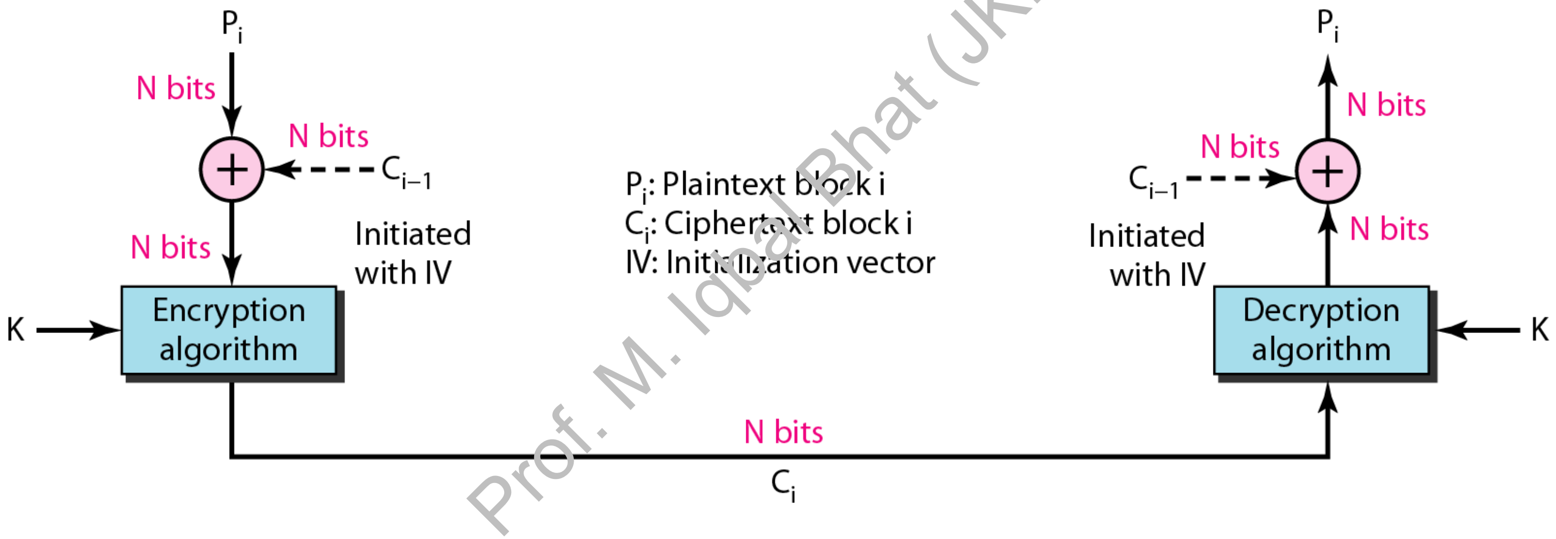
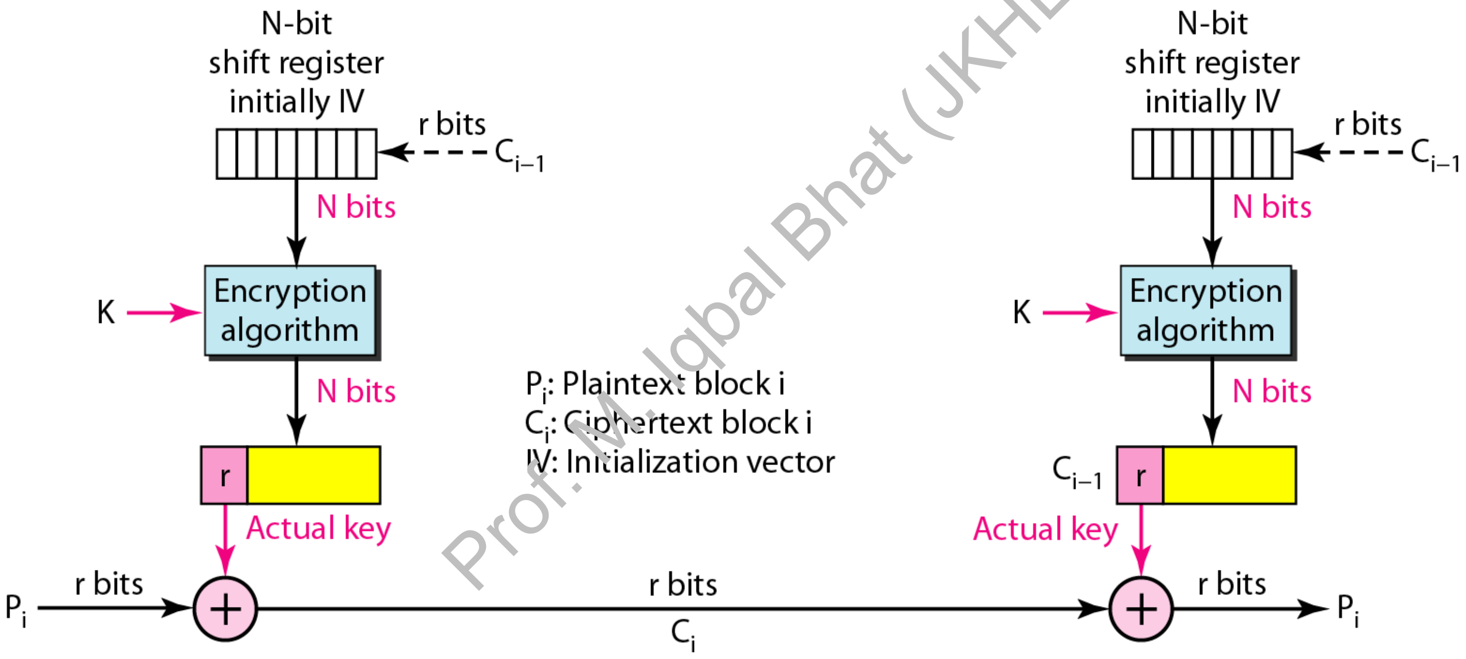
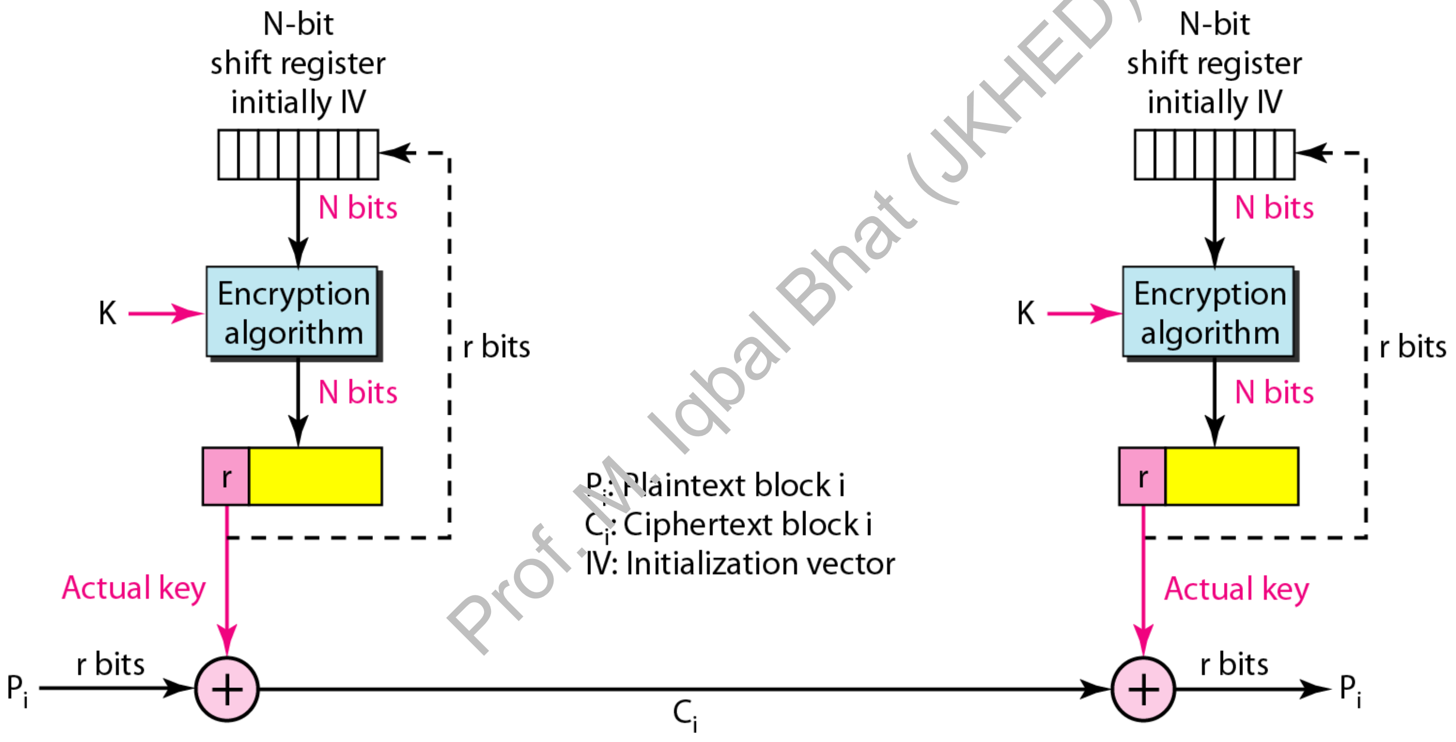


Figure 30.22 *CFB mode*



P_i: Plaintext block i
C_i: Ciphertext block i
IV: Initialization vector

Figure 30.23 *OFB mode*



P_i: Plaintext block i
C_i: Ciphertext block i
IV: Initialization vector

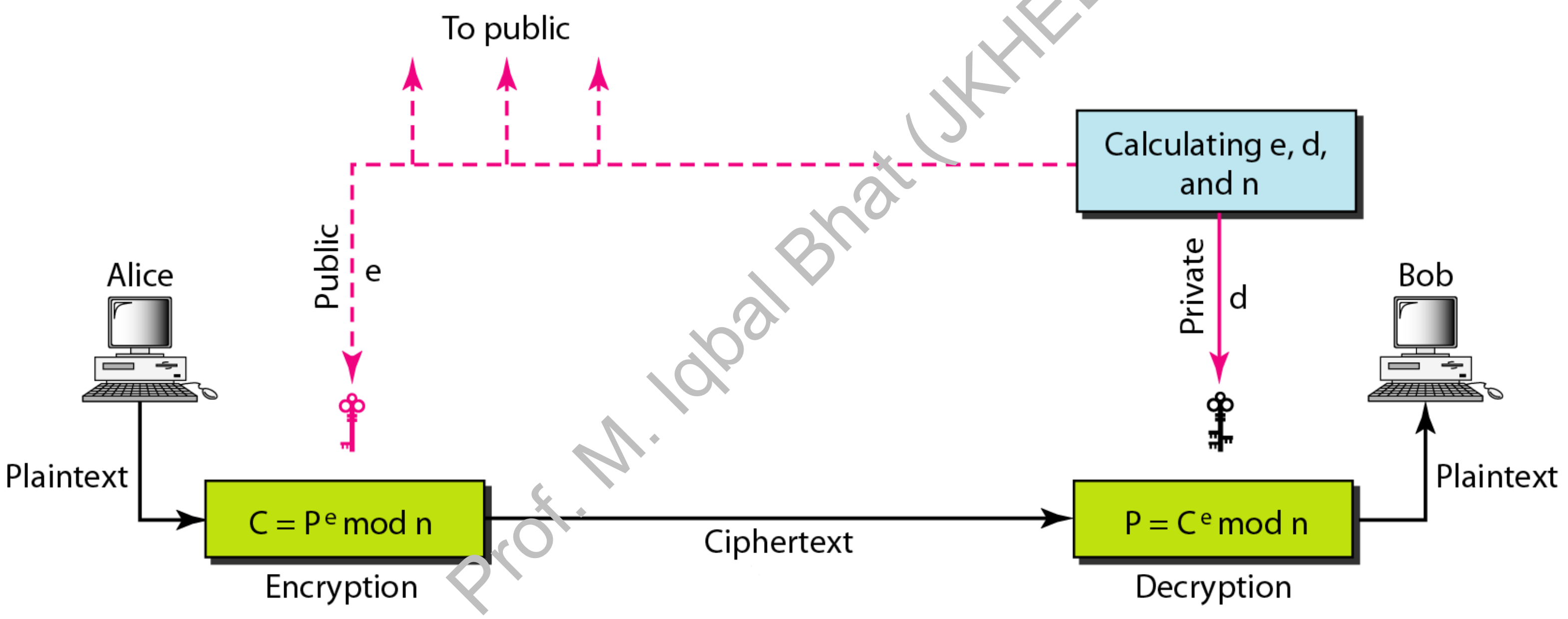
30-3 ASYMMETRIC-KEY CRYPTOGRAPHY

An asymmetric-key (or public-key) cipher uses two keys: one private and one public. We discuss two algorithms: RSA and Diffie-Hellman.

Topics discussed in this section:

RSA
Diffie-Hellman

Figure 30.24 *RSA*

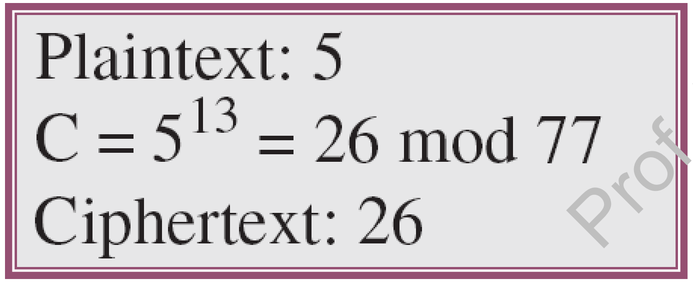


Note

In RSA, e and n are announced to the public; d and Φ are kept secret.

Example 30.7

Bob chooses 7 and 11 as p and q and calculates $n = 7 \cdot 11 = 77$. The value of $\Phi = (7 - 1)(11 - 1)$ or 60. Now he chooses two keys, e and d. If he chooses e to be 13, then d is 37. Now imagine Alice sends the plaintext 5 to Bob. She uses the public key 13 to encrypt 5.



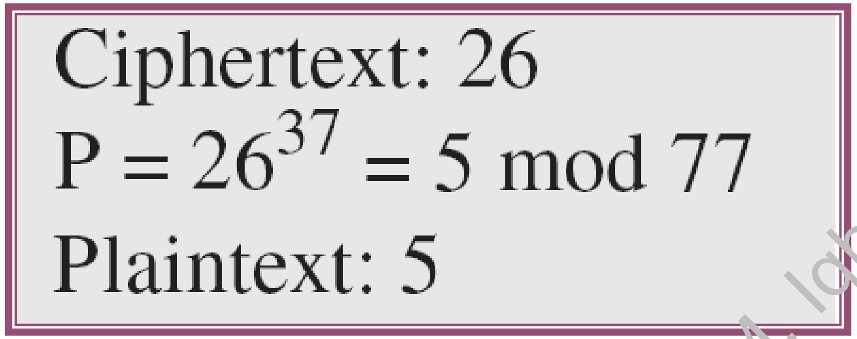
Plaintext: 5

$C = 5^{13} = 26 \bmod 77$

Ciphertext: 26

Example 30.7 (continued)

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:



Ciphertext: 26

$P = 26^{37} = 5 \bmod 77$

Plaintext: 5

The plaintext 5 sent by Alice is received as plaintext 5 by Bob.

Example 30.8

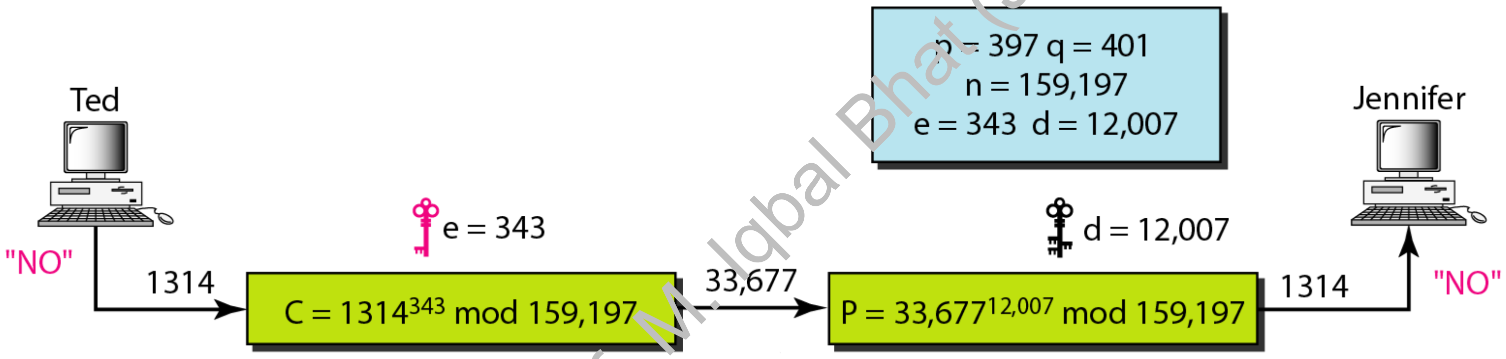
Jennifer creates a pair of keys for herself. She chooses $p = 397$ and $q = 401$. She calculates $n = 159{,}197$ and $\Phi = 396 \cdot 400 = 158{,}400$. She then chooses $e = 343$ and $d = 12{,}007$. Show how Ted can send a message to Jennifer if he knows e and n.

Example 30.8 (continued)

Solution

Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25) with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314. Ted then uses e and n to encrypt the message. The ciphertext is $1314^{343} = 33{,}677$ mod 159,197. Jennifer receives the message 33,677 and uses the decryption key d to decipher it as $33{,}677^{12{,}007} =$ 1314 mod 159,197. Jennifer then decodes 1314 as the message "NO". Figure 30.25 shows the process.

Figure 30.25 *Example 30.8*



Example 30.9

Let us give a realistic example. We randomly chose an integer of 512 bits. The integer p is a 159-digit number.

p = 96130345313583504574191581280615427909309845594996215822583150879647940
455056470638491257160180347503120986660649242019180878066742109606335 4
219926661209

The integer q is a 160-digit number.

q = 12060191957231446918276794204450896001555925054637033936061798321731482
14848376465921538945320917522527322683010712069560460251388714552496900
0359660045617

Example 30.9 (continued)

We calculate n. It has 309 digits:

n = 115935041739676149688925098646158875237714573754541447754855261376147 88
540832635081727687881596832516846884930062548576411125016241455233918 29
271625076567727274600970827141277304349605005563472745666280600999240 37
102991424472292215772798531727033839381334692684137327622000966676671 83
1831088373420823444370953

We calculate Φ. It has 309 digits:

ϕ = 115935041739676149688925098646158875237714573754541447754855261376147 88
540832635081727687881596832516846884930062548576411125016241455233918 29
271625076567510542336084929167520344826279881175547876570139234444057 16
989581728196098226361075467211864612171359107358640614008885170265377 27
7264467341066243857664128

Example 30.9 (continued)

We choose e = 35,535. We then find d.

e = 35535

d = 5800830286003776393609366128967791759466906208965096218042286611138059385282235873170628691003002171085904433840217072986908760061153062025249598844480475682409662470814858171304632406440777048331340108509473852956450719367740611973265574242372176176746207763716420760033708533328853214470885955136670294831

Alice wants to send the message "THIS IS A TEST" which can be changed to a numeric value by using the 00–26 encoding scheme (26 is the space character).

P = 1907081826081826002619041819

Example 30.9 (continued)

The ciphertext calculated by Alice is $C = P^e$, which is.

C = 4753091236462268272063655506105451809423717960704917165232392430544529
6061319932856661784341835911415119741125200568297979457173603610127821
8847892741566090480023507190715277185914975188465888632101148354103361
6578984679683867637337657774656250792805211481418440481418443081277305
9004692874248559166462108656

Bob can recover the plaintext from the ciphertext by using $P = C^d$, which is

P = 1907081826081826002619041819

The recovered plaintext is THIS IS A TEST after decoding.

Note

The symmetric (shared) key in the Diffie-Hellman protocol is $K = g^{xy} \bmod p$.

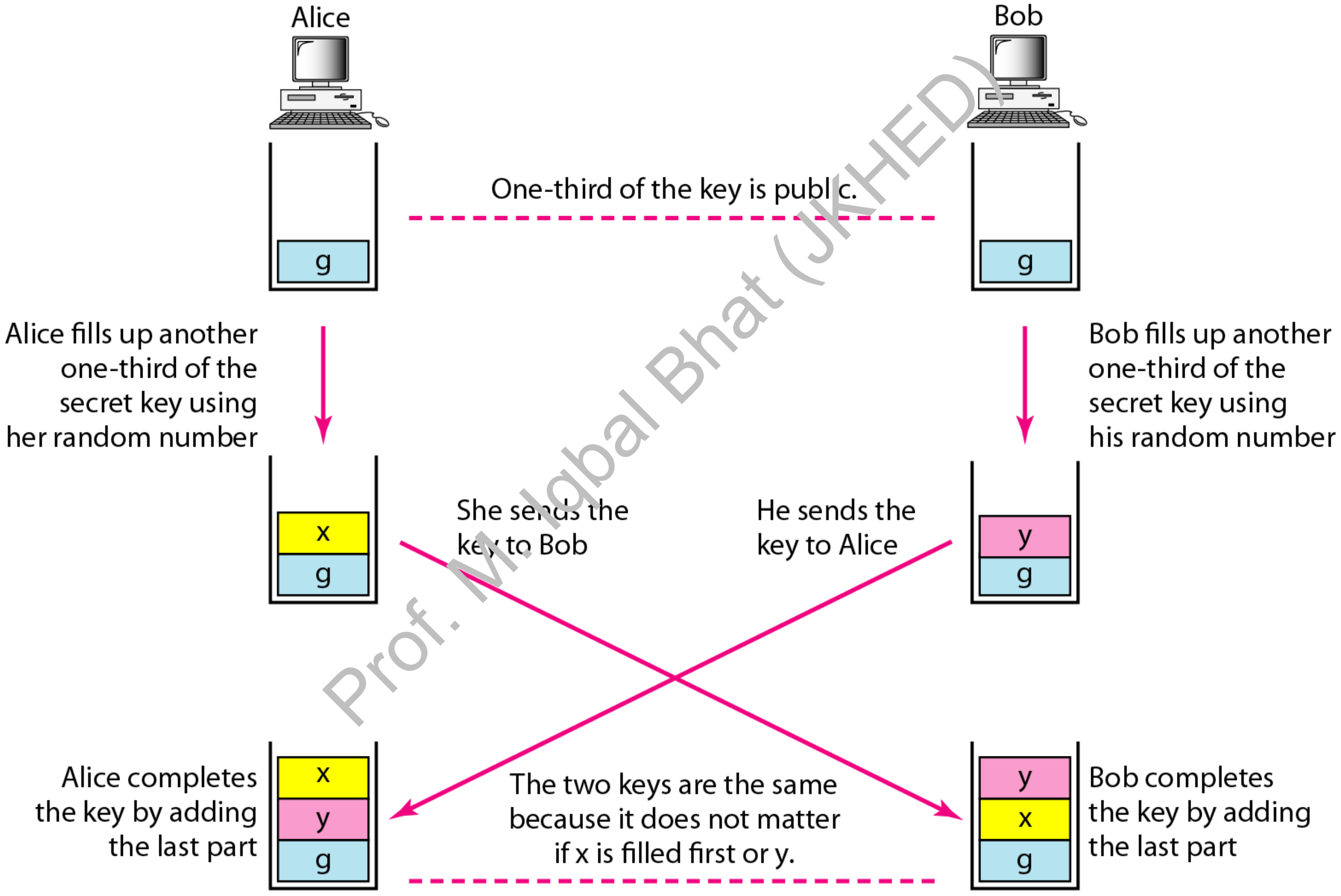
Example 30.10

Let us give a trivial example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume g = 7 and p = 23. The steps are as follows:

1. *Alice chooses $x = 3$ and calculates $R_1 = 7^3 \bmod 23 = 21$.*
2. *Bob chooses $y = 6$ and calculates $R_2 = 7^6 \bmod 23 = 4$.*
3. *Alice sends the number 21 to Bob.*
4. *Bob sends the number 4 to Alice.*
5. *Alice calculates the symmetric key $K = 4^3 \bmod 23 = 18$.*
6. *Bob calculates the symmetric key $K = 21^6 \bmod 23 = 18$.*

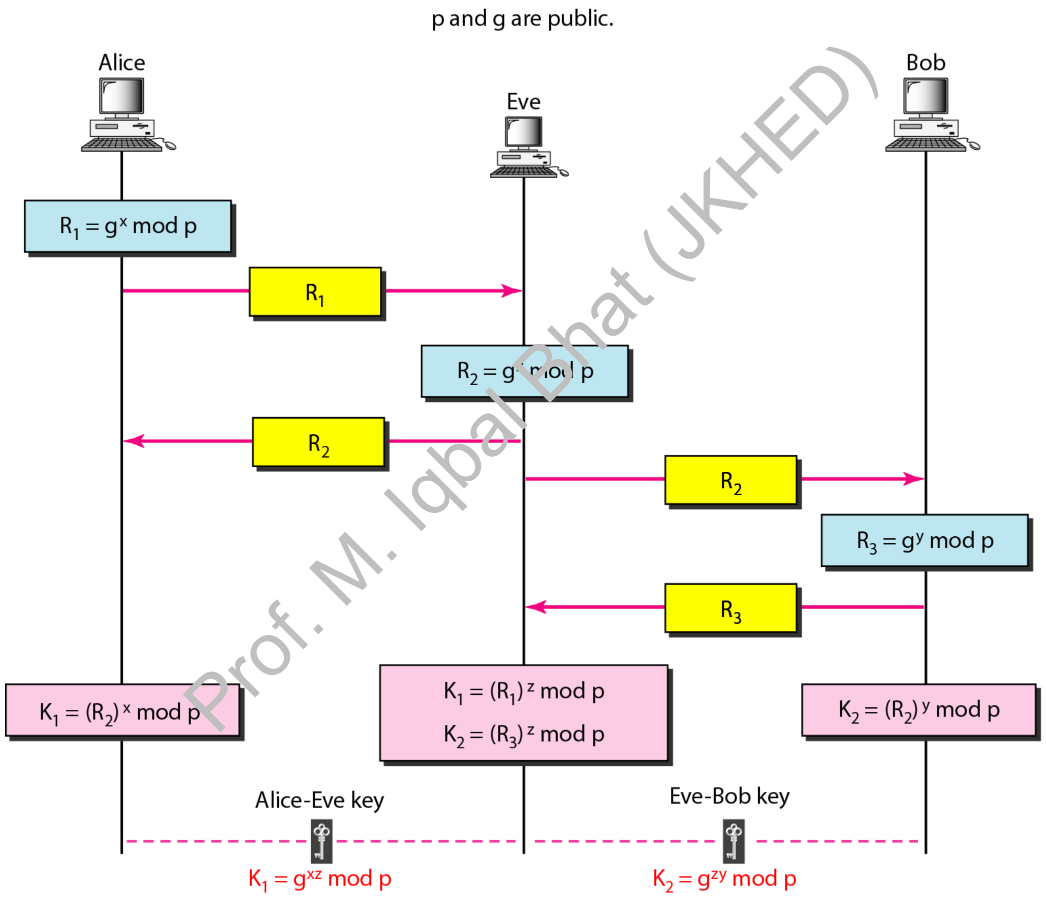
The value of K is the same for both Alice and Bob; $g^{xy} \bmod p = 7^{18} \bmod 23 = 18$.

Figure 30.27 *Diffie-Hellman idea*



Alice
Bob
One-third of the key is public.
g
g
Alice fills up another one-third of the secret key using her random number
Bob fills up another one-third of the secret key using his random number
x
g
y
g
She sends the key to Bob
He sends the key to Alice
Alice completes the key by adding the last part
x
y
g
y
x
g
Bob completes the key by adding the last part
The two keys are the same because it does not matter if x is filled first or y.

Figure 30.28 *Man-in-the-middle attack*


p and g are public.
Alice
Eve
Bob
$R_1 = g^x \bmod p$
R_1
$R_2 = g^z \bmod p$
R_2
R_2
$R_3 = g^y \bmod p$
R_3
$K_1 = (R_2)^x \bmod p$
$K_1 = (R_1)^z \bmod p$
$K_2 = (R_3)^z \bmod p$
$K_2 = (R_2)^y \bmod p$
Alice-Eve key
$K_1 = g^{xz} \bmod p$
Eve-Bob key
$K_2 = g^{zy} \bmod p$