Block Cipher-Modes of Operation

DR. VASUDHA ARORA

VASUDHA.ARORA@GDGU.ORG, VASUDHARORA6@GMAIL.COM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GD GOENKA UNIVERSITY, GURUGRAM

Block Ciphers

- In block Ciphers rather than encrypting one bit at a time, a block of bits is encrypted at a time and during decryption each block would be translated back to the original form.
- Problem with block ciphers is repeating text.
- For repeating text patterns same cipher text is generated, the cryptanalyst can look for repeating strings and could guess the message.
- To deal with this problem block ciphers are used in chaining modes, i.e. , previous block of cipher is mixed with the current block in order to obscure the cipher text to avoid the repeated patterns of the blocks.
- Blocks used in block ciphers generally contain 64 bits or more.

Block Ciphers-Modes

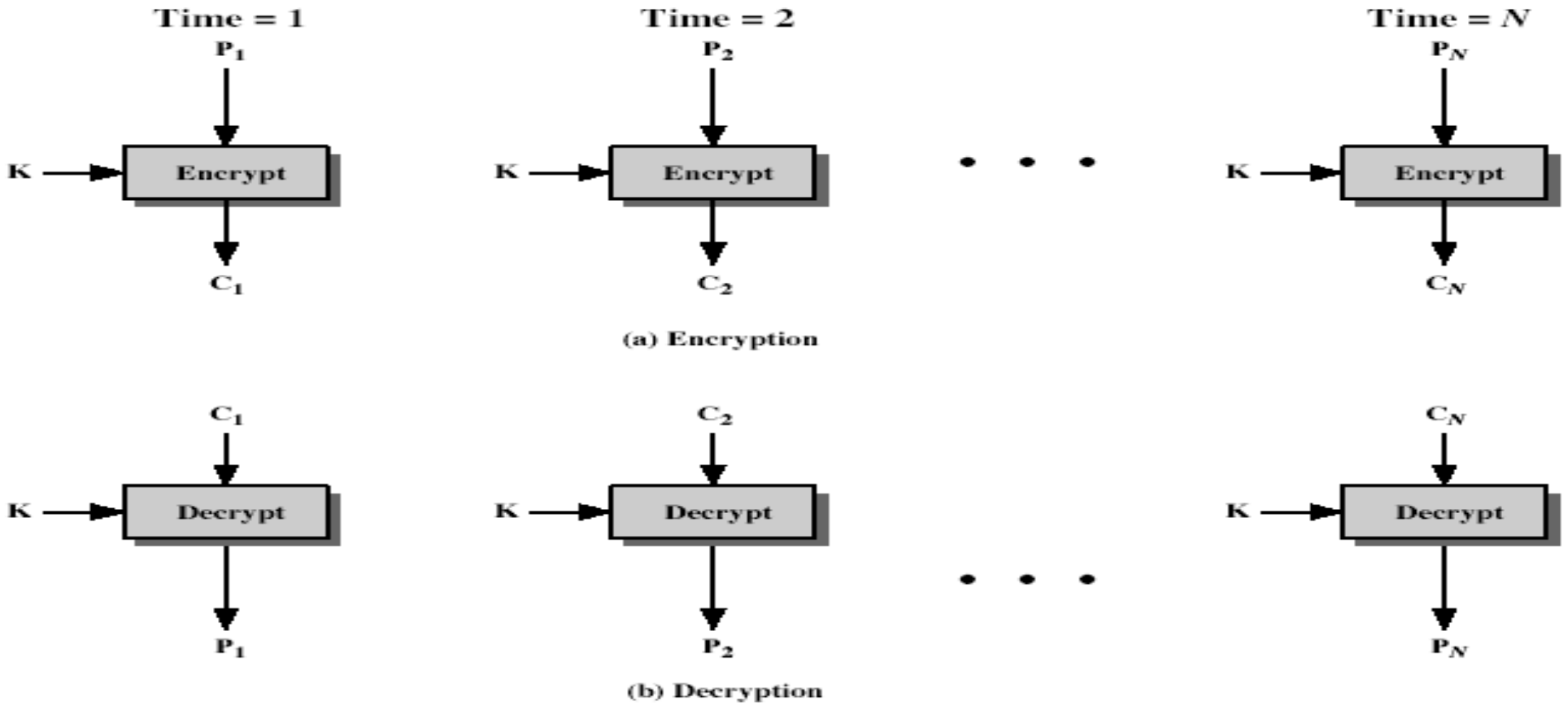
There are five modes of operation for block ciphers-----

- Electronic Code Book Mode (ECB)
- Cipher Block Chaining Mode (CBC)
- Cipher FeedBack Mode (CFB)
- Output FeedBack Mode (OFB)
- Counter Mode (CTR)

Electronic Codebook Book (ECB)

- Simplest mode of Execution
- 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
  - $C_i = DES_{K1}(P_i)$
- For all blocks in a message the same key is used for encryption
- uses: secure transmission of single values

Electronic Codebook Book (ECB)



Advantages and Limitations of ECB

Advantages:

- main use is sending a few blocks of data

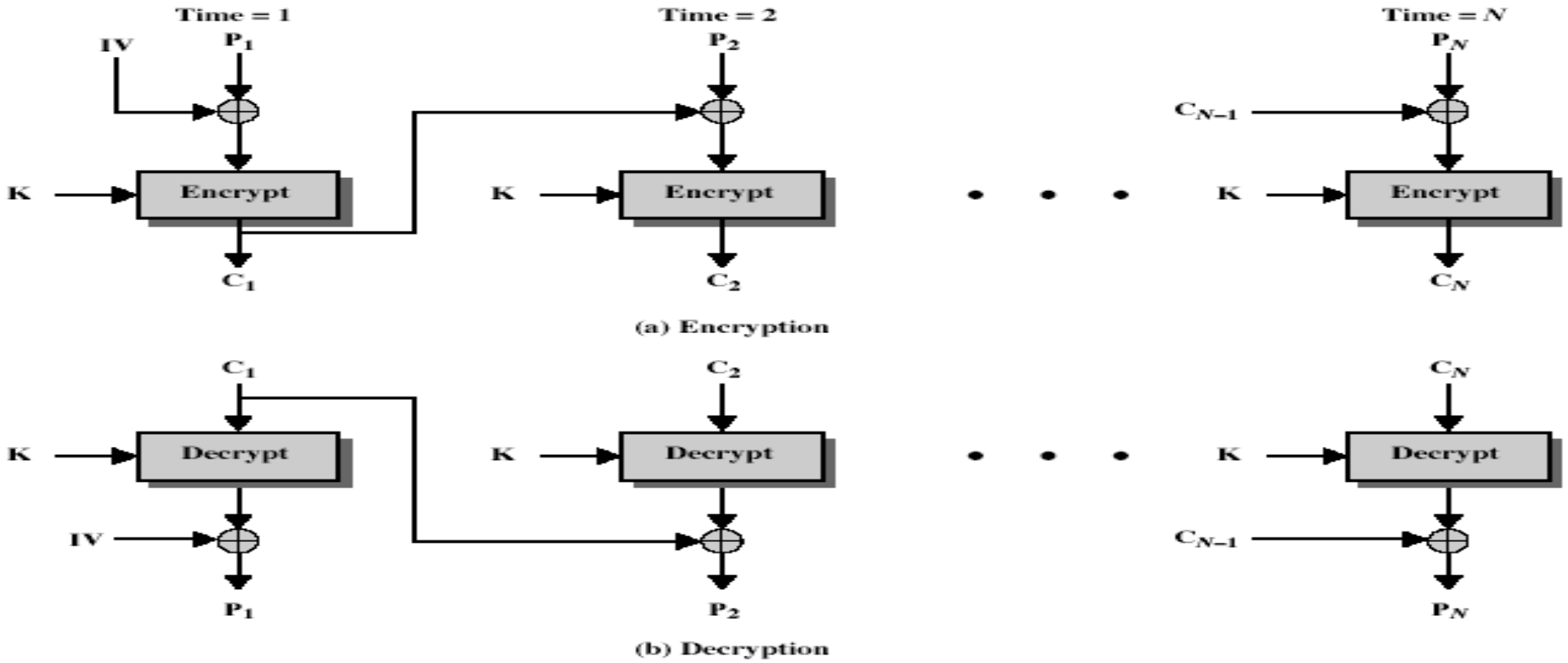
Limitations:

- message repetitions may show in ciphertext, or with messages that change very little, which become a code-book analysis problem
- weakness is due to the encrypted message blocks being independent

Cipher Block Chaining (CBC)

- Ensures even two identical plain text blocks yield totally different cipher texts in the output.
- Uses a chaining mechanism
- Chaining adds a feedback mechanism to the block cipher
- results of encryption of previous block are fed into the encryption of next block.

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC)

- The first step receives two inputs:
  - The first block of plain text
  - a random block of text known as Initialization vector (IV)
- IV has no special meaning, it is simply used to make each message unique.
- IV is generated randomly using a random number generator
- the first block of IV and Plaintext are combined using XOR and then encrypted using a key to produce first ciphertext block
- this ciphertext block is then provided as a feedback to the next plaintext block as so on.

Message Padding

- at end of message must handle a possible last short block
  - which is not as large as block size of cipher
  - pad either with known non-data value (eg nulls)
  - or pad last block along with count of pad size
    - eg. [b1 b2 b3 0 0 0 0 5]
    - means have 3 data bytes, then 5 bytes pad+count
  - this may require an extra entire block over those in message
- there are other modes, which avoid the need for an extra block

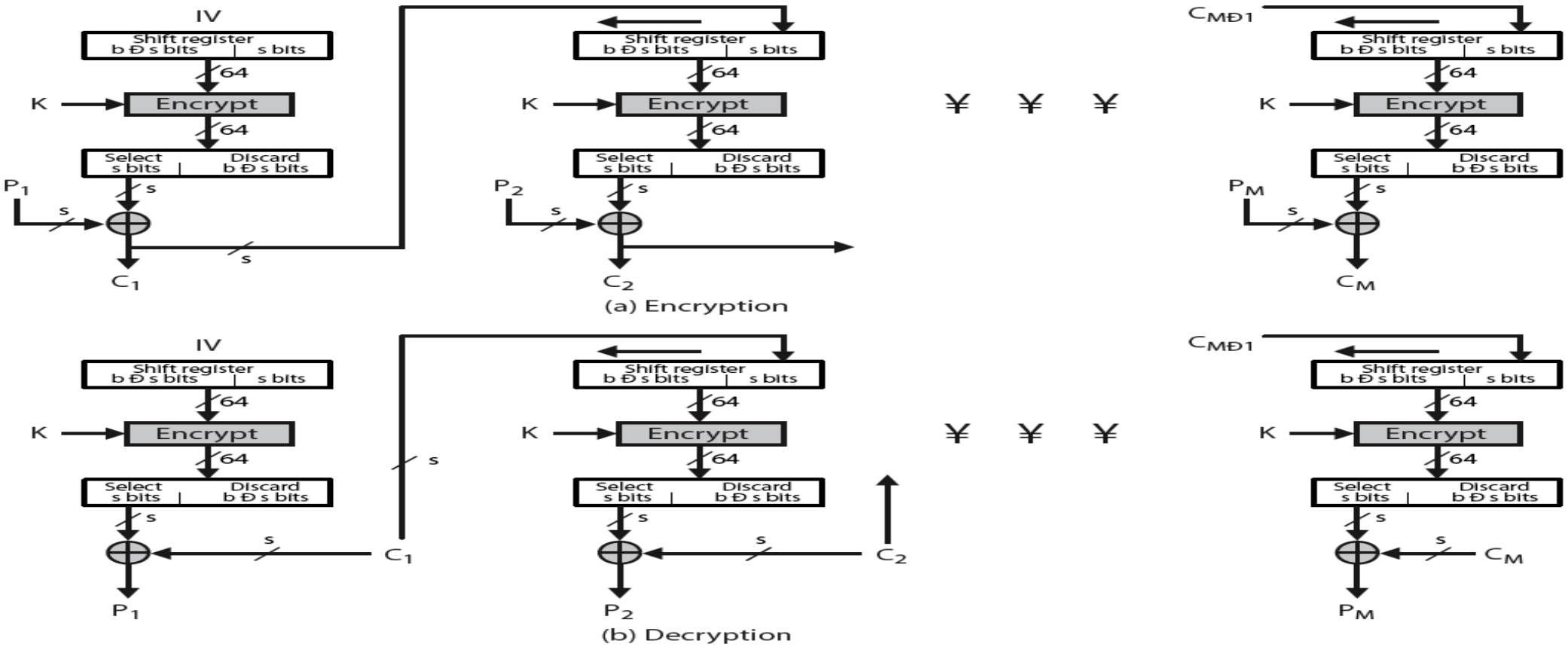
Advantages and Limitations of CBC

- a ciphertext block depends on **all** blocks before it
- any change to a block affects all following ciphertext blocks
- need **Initialization Vector** (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, and change IV to compensate
  - hence IV must either be a fixed value
  - or must be sent encrypted in ECB mode before rest of message

Cipher FeedBack (CFB)

- Like CBC a 64 bit IV is used.
- this IV is kept in a shift register, It is encrypted to produce a 64-bit encrypted IV.
- the leftmost s bits (i.e the MSB) of encrypted IV are XORed with first s bits of plaintext.
- This produces the first portion of cipher text, say c1, which is then transmitted to the receiver.
- now the bits of IV in shift register are shifted left by s positions.
- the rightmost s bits in shift register are now replaced with c1.

Cipher FeedBack (CFB)



Advantages and Limitations of CFB

appropriate when data arrives in bits/bytes

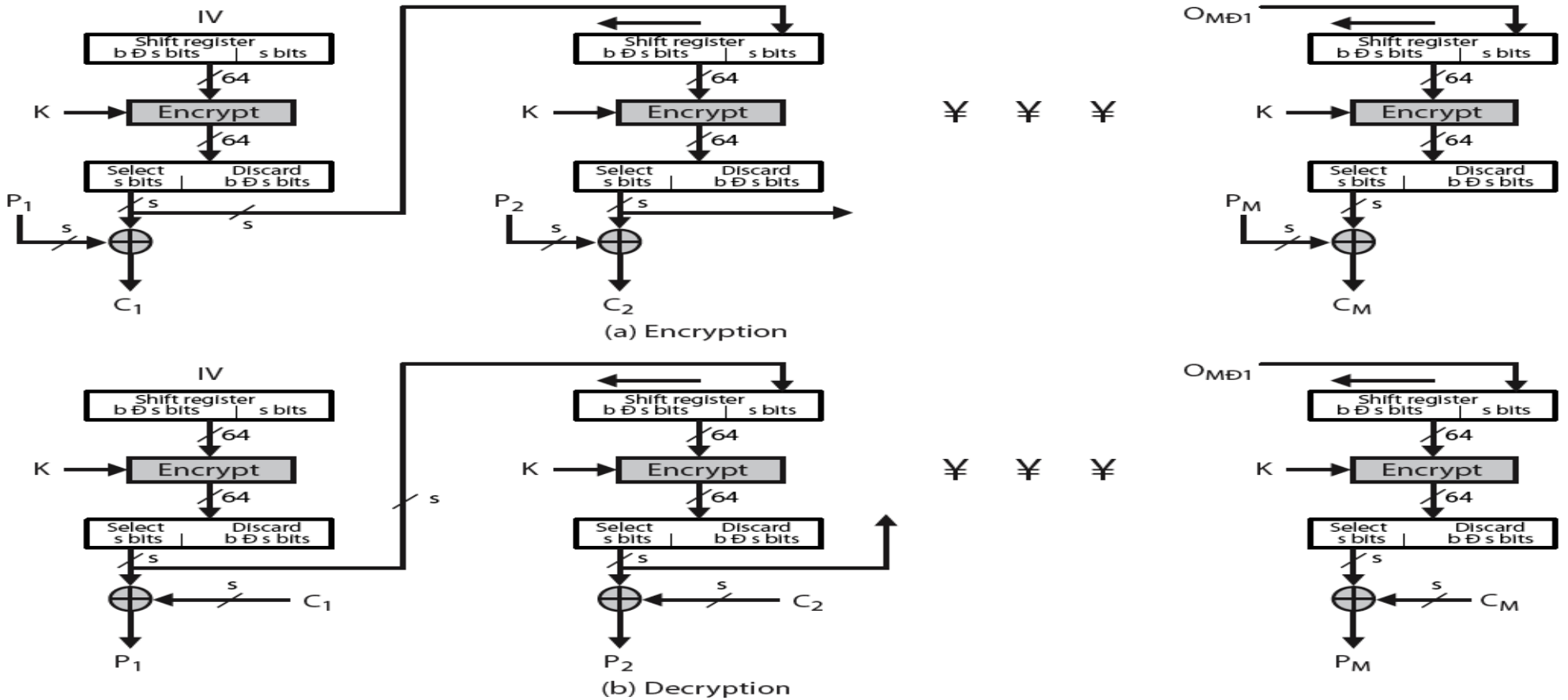
most common stream mode

limitation is error propagation for several blocks

Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance
  - $C_i = P_i \text{ XOR } O_i$
- uses: stream encryption on noisy channels

Output FeedBack (OFB)



Advantages and Limitations of OFB

bit errors do not propagate

a variation of a Vernam cipher

- hence must **never** reuse the same sequence (key+IV)

sender & receiver must remain in sync

originally specified with m-bit feedback

subsequent research has shown that only **full block feedback** (ie CFB-64 or CFB-128) should ever be used

Counter (CTR)

a “new” mode, though proposed early on

similar to OFB but encrypts counter value rather than any feedback value

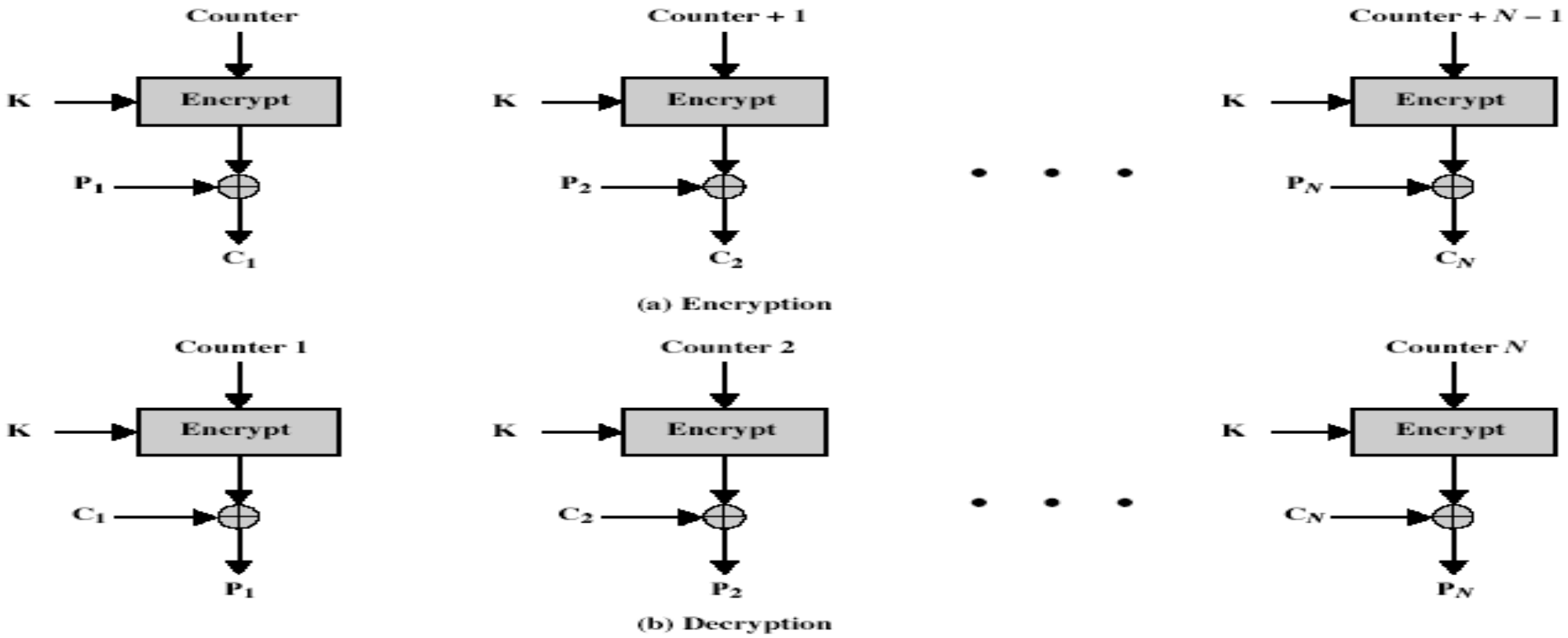
must have a different key & counter value for every plaintext block (never reused)

$C_i = P_i \text{ XOR } O_i$

$O_i = DES_{K1}(i)$

uses: high-speed network encryptions

Counter (CTR)


Counter
Encrypt
K
P1
C1
Counter + 1
Encrypt
K
P2
C2
Counter + N – 1
Encrypt
K
PN
CN
(a) Encryption
Counter 1
Encrypt
K
C1
P1
Counter 2
Encrypt
K
C2
P2
Counter N
Encrypt
K
CN
PN
(b) Decryption

Advantages and Limitations of CTR

efficiency

- can do parallel encryptions in h/w or s/w
- can preprocess in advance of need
- good for bursty high speed links

random access to encrypted data blocks

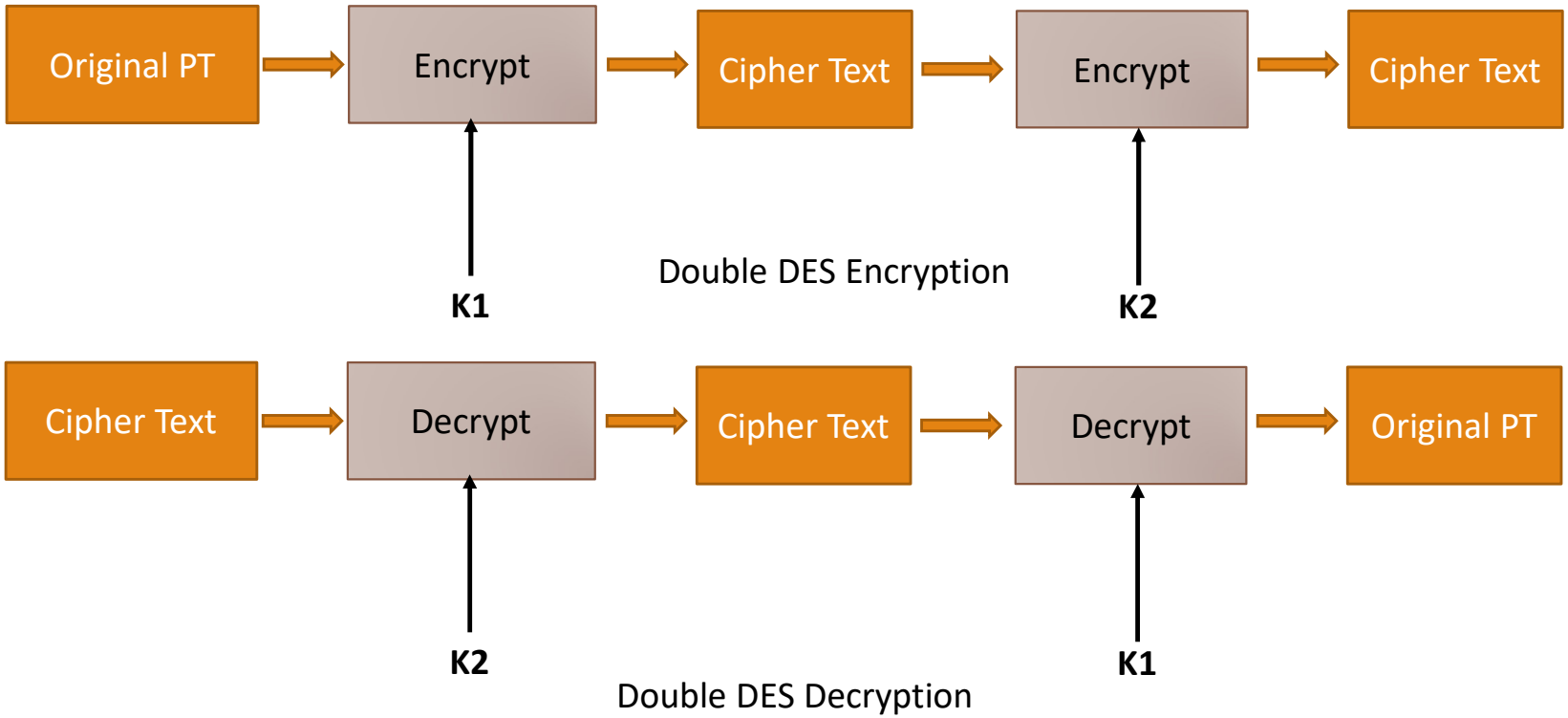
provable security (good as other modes)

but must ensure never reuse key/counter values, otherwise could break (cf OFB)

Variations of DES

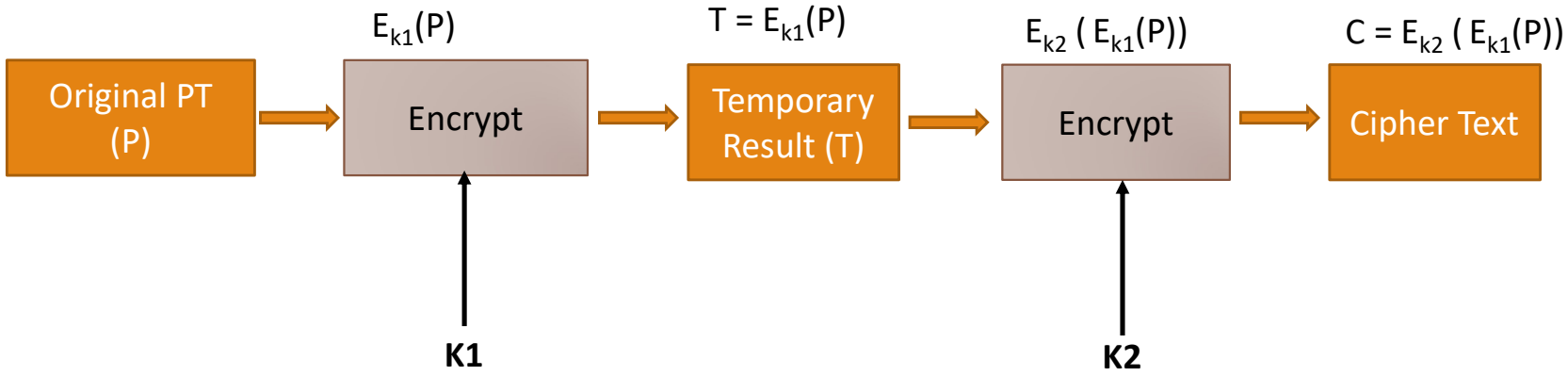
- **Double DES**
- **Triple DES**
  - **Triple DES with 3 Keys**
  - **Triple DES with 2 keys (EDE MODE)**

Double DES


Original PT
Encrypt
Cipher Text
Encrypt
Cipher Text
K1
Double DES Encryption
K2
Cipher Text
Decrypt
Cipher Text
Decrypt
Original PT
K2
Double DES Decryption
K1

MEET-In-The-Middle Attack

- Merkel and Hellman introduced the concept of meet in the middle attack.
- Involves encryption from one end and decryption from the other, and matching the results in the middle.
-



Double DES Encryption (Mathematically Explained)

MEET-In-The-Middle Attack

- for a block of known plain text and the corresponding ciphertext also known, a cryptanalyst may calculate the value of keys K1 and K2.
- for all possible values(2^{56}) of key k1, the cryptanalyst would use a large table and perform the following:
  - Encrypt the plain text block P by performing $E_{k1}(P)$. i.e. it will calculate T.
  - Will store all the results with all possible keys in the consecutive rows of the table in memory.
- Now he will perform the reverse operation, i.e Decrypt the known ciphertext with all possible values of K2
- compare the values stored in the table earlier with values calculated during reverse operation.
- To Summarize,
  - from known PT and all possible values of K1 calculate T= $E_{k1}(P)$
  - from the CT and all possible values of K2 calculate T = $D_{k2}(C)$

MEET-In-The-Middle Attack

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

=> there is a chance that he gets same T in both the operations.

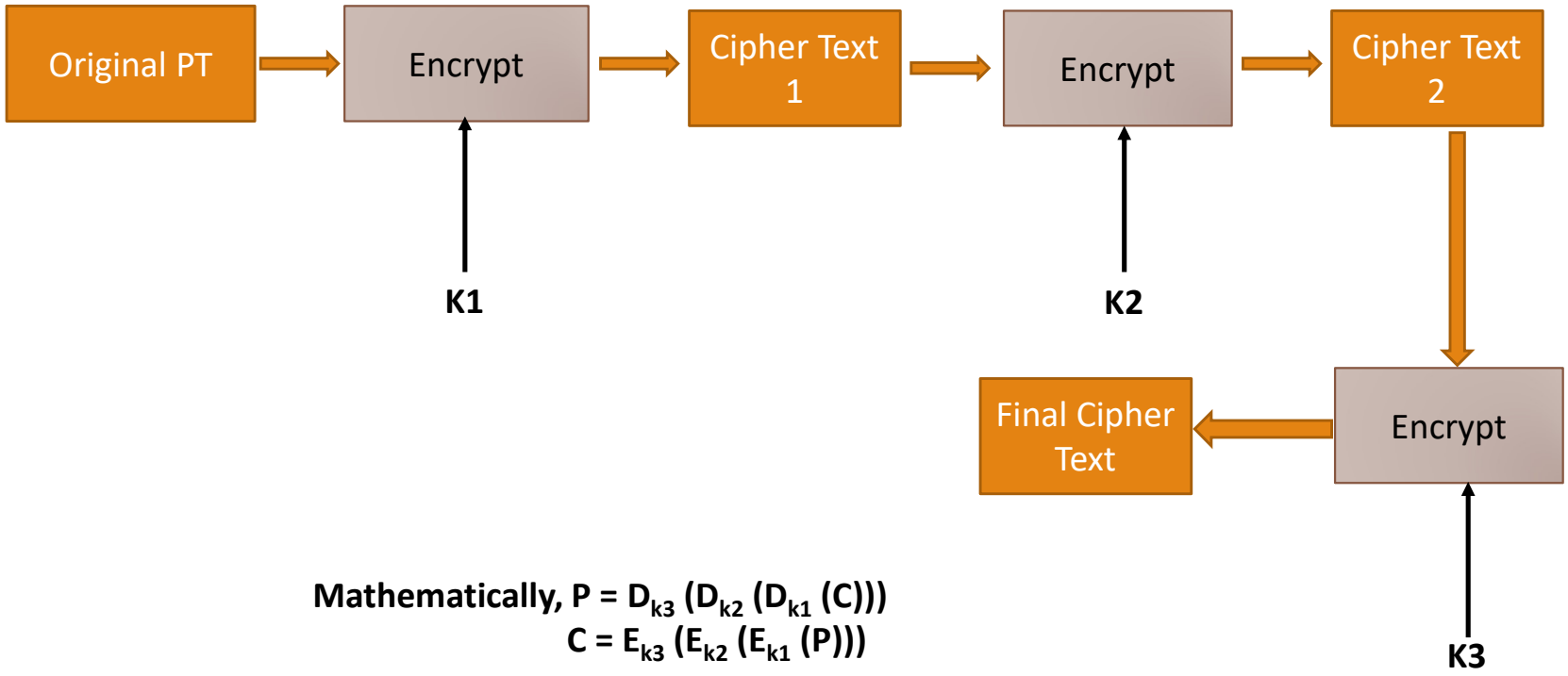
=> If Cryptanalyst is able to find same T for both , *encrypt with K1* and *Decrypt with K2* operations,

=> he is able to find possible values of K1 and K2

Triple DES

- To overcome drawbacks of Double DES and making CT more secure paved way for Triple DES – DES three times.
- It comes in two flavours:
  - Triple DES with 3 keys
  - Triple DES with two keys.
- Triple DES with 3 keys is used extensively in many products including PGP, S/MIME

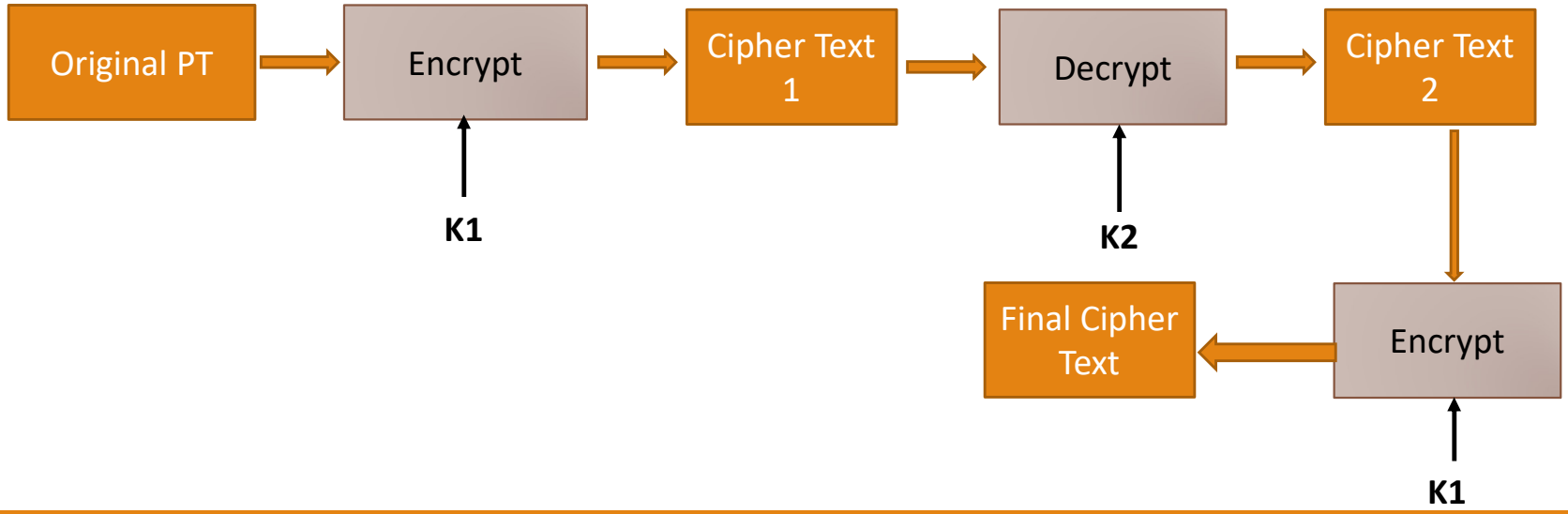
Triple DES with 3 keys



Mathematically, $P = D_{k3} (D_{k2} (D_{k1} (C)))$
$C = E_{k3} (E_{k2} (E_{k1} (P)))$

Triple DES with 2 keys – EDE MODE

- Encrypt the PT with key K1. Thus we have **(E_{k1} (P))**
- Decrypt the output **(E_{k1} (P))** with key K2. Thus we have (D_{k2} (E_{k1} (P)))
- Finally Encrypt the output again with key K1. Thus we have (E_{k1} (D_{k2} (E_{k1} (P))))



Avalanche Effect

- DES exhibits a strong Avalanche Effect.
  - A change in one bit of PT or key produces a change in large number bits of Cipher texts
- e.g. 1000000000000000000000000000000000000000000000000000000000000000….64 bits (original PT)

  0000000000000000000000000000000000000000000000000000000000000000……64 bits (changed first bit)

Produces a change in 21 bits after 3 rounds and more than 34 bits after completion of 16 rounds.

Data Encryption Standards – DES Algorithm

DR. VASUDHA ARORA

VASUDHA.ARORA@GDGU.ORG, VASUDHARORA6@GMAIL.COM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GD GOENKA UNIVERSITY, GURUGRAM

Data Encryption Standard - DES

DES was developed as a standard for communications and data protection by an IBM research team, in response to a public request for proposals by the NBS - the National Bureau of Standards (which is now known as NIST)

DES - Basics

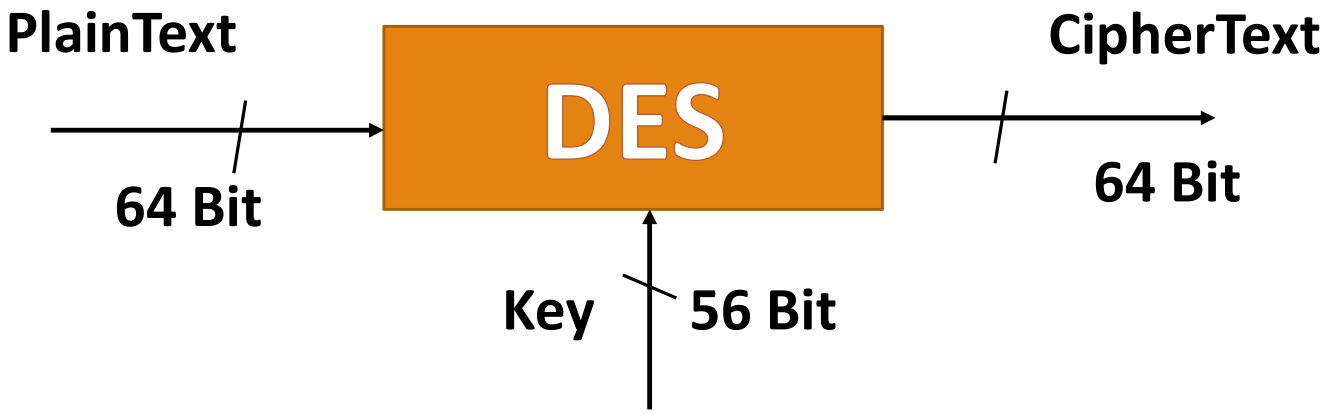
- DES uses the two basic techniques of cryptography - confusion and diffusion.
- At the simplest level, diffusion is achieved through numerous permutations and confusion is achieved through the XOR operation.

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

DES Features

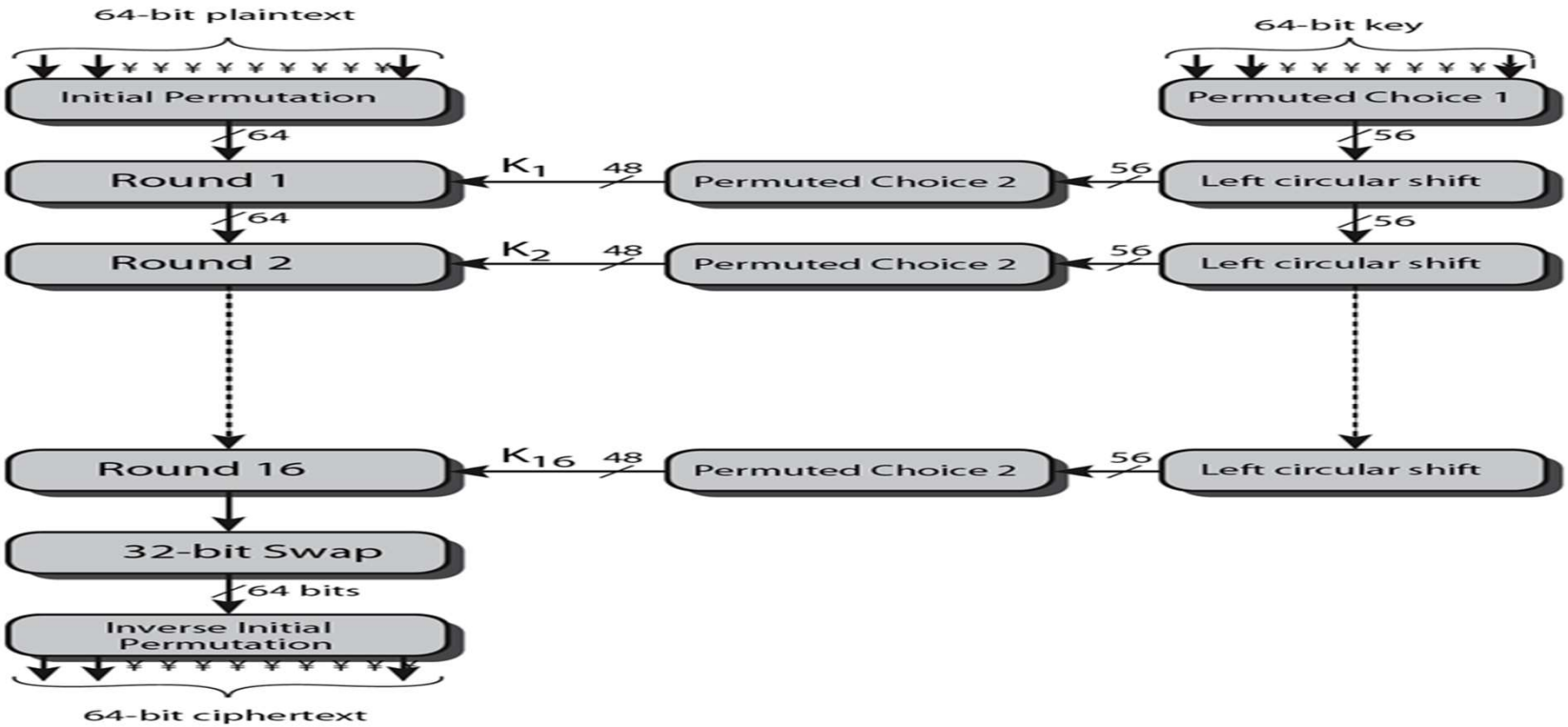
- Features: –
  - Block size = 64 bits
  - Key size = 56 bits (in reality, 64 bits, but 8 are used as parity-check bits for error control)
  - Number of rounds = 16
  - 16 intermediary keys, each 48 bits



DES Features

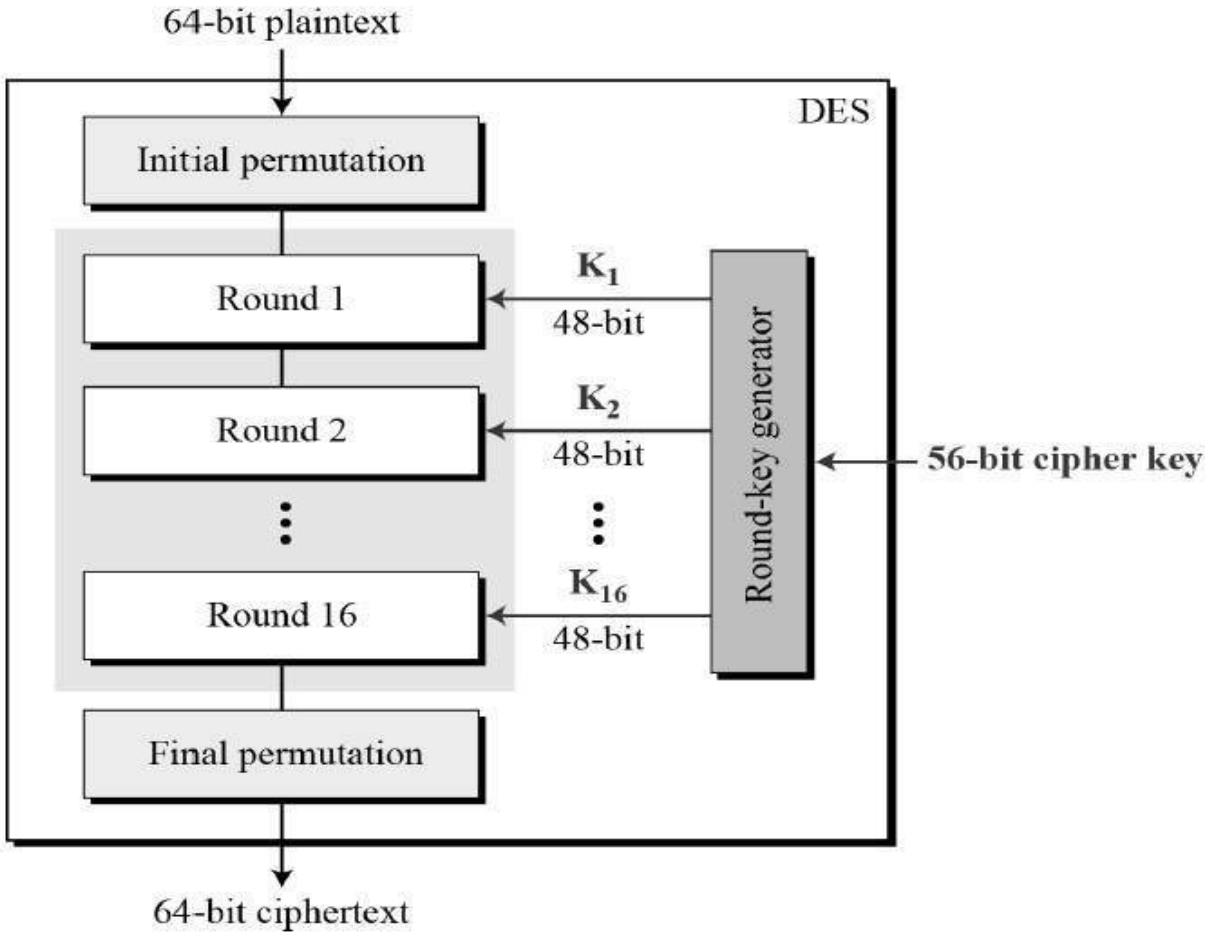
- The algorithm transforms 64 bit input in a series of steps into 64 bit output.
- The stepswith same keys in reverse order are used to reverse the encryption.
- DES has been found vulnerable against very powerful attack, but it has been a landmark in cryptographic algorithms.
- DES is generally used in ECB, CBC and CFB modes.

DES-Rounds


64-bit plaintext
Initial Permutation
64
Round 1
64
Round 2
Round 16
32-bit Swap
64 bits
Inverse Initial Permutation
64-bit ciphertext
K1
K2
K16
48
Permuted Choice 2
56
Left circular shift
64-bit key
Permuted Choice 1
56

DES Process

- DES is based on two fundamental attributes of Cryptography:
  - confusion
  - Diffusion
- DES consists of 16 rounds: Each round performs substitution. Transposition
- In the first step, 64 bit plain text block is handed over to Initial permutation (IP) function.
- The IP is performed on the plain text.
- IP produces two halves of permuted blocks, L and R
- Each of L and R goes through 16 rounds of encryption.
- At the end of each round they are re-joined and a final permutation is performed on the combined block
- The result of this process produces 64 bit ciphertext.


64-bit plaintext
DES
Initial permutation
Round 1
Round 2
Round 16
Final permutation
K1
48-bit
K2
48-bit
K16
48-bit
Round-key generator
56-bit cipher key
64-bit ciphertext

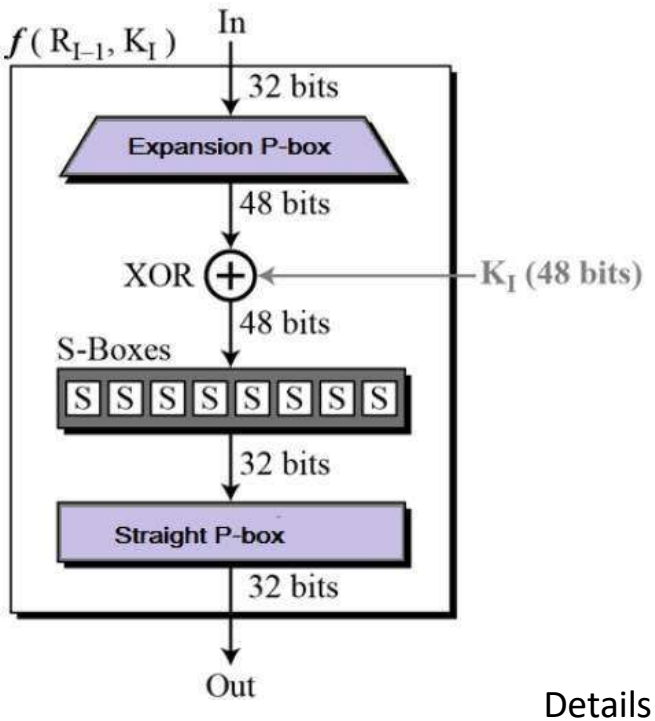
Initial Permutation

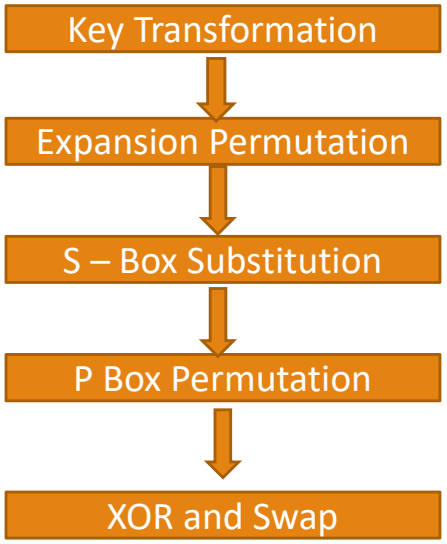
- IP is done only once and is done before the start of first round.
- IP transpositions the bits of plain text block in a predefined order.
- e.g. 1st bit of original plaintext block is replaced by the 58th bit, 2nd bit by 50th bit and so on.
-

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	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure - Initial permutation table

One Round of DES





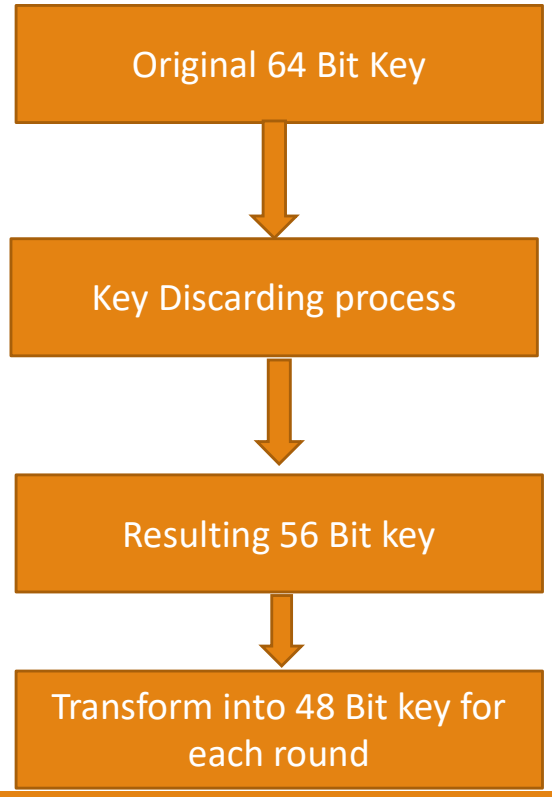
Details of one Round in DES

Key length in DES

- In the DES specification, the key length is 64 bit:
- 8 bytes; in each byte, the 8th bit is a parity-check bit
- Each parity-check bit is the XOR of the previous 7 bits
- Hence before DES process starts, every 8th bit of the key is discarded to produce a key of 56-bit

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

Key Transformation



Key Transformation

- From this 56 Bit key a different 48-Bit Subkey is generated during each round using key transformation
- To generate a subkey of 48 bit for each round, the 56-Bit key is divided into 2 halves, each of 28-bit.
- These halves are circularly shifted left by one or two positions depending on the round number.
  - If the round no. is 1,2,9,16 the shift is done by one position.
  - for all other rounds the circular shift is done by two positions.
- After an appropriate shift, 48 of 56 bits are selected using compression permutation

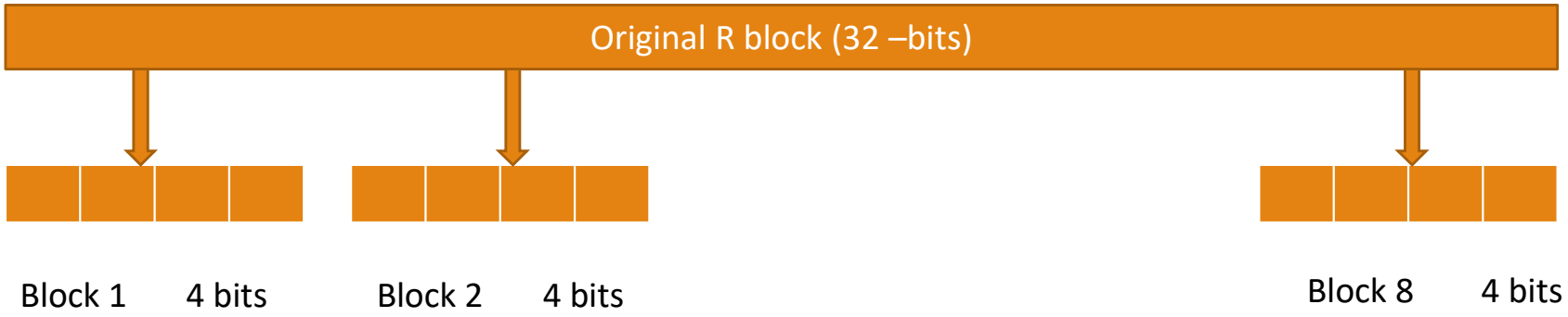
Compression Permutation

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

Figure - compression permutation

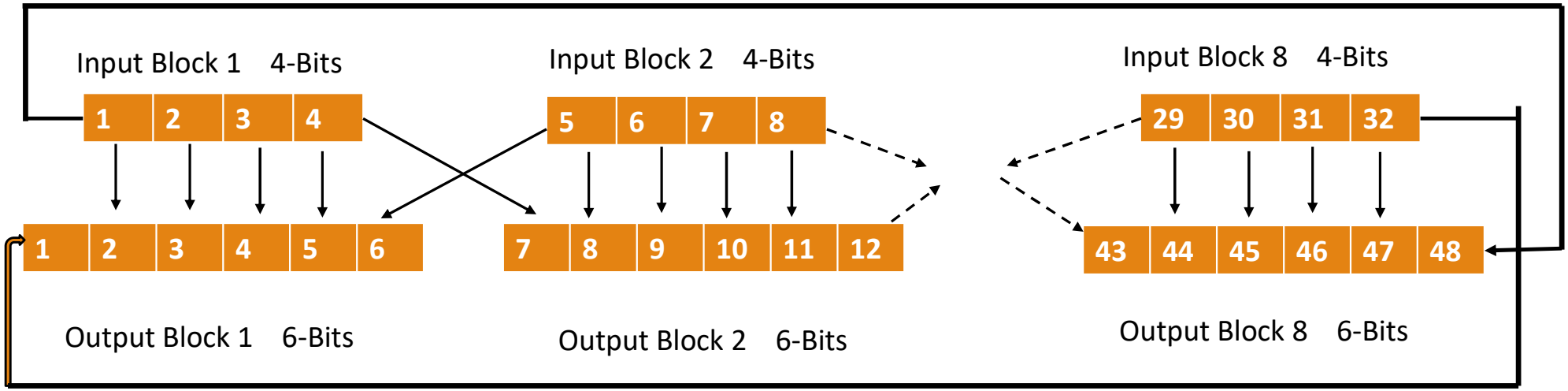
Expansion Permutation

- After initial permutation , the 64-bit permuted block is divided into two equal halves L and R , 32-bit each.
- During ***Expansion Permutation*** , R block is expanded from 32 bits to 48 bits, and hence the name *Expansion Permutation*.
- ***Process for Expansion from 32 bits to 48 bits:***
  - The 32-bit R block is divided into 8 blocks, with each consisting of 4 bits



Expansion Permutation

- Each 4bit block is then expanded to a corresponding 6-bit block as follows:



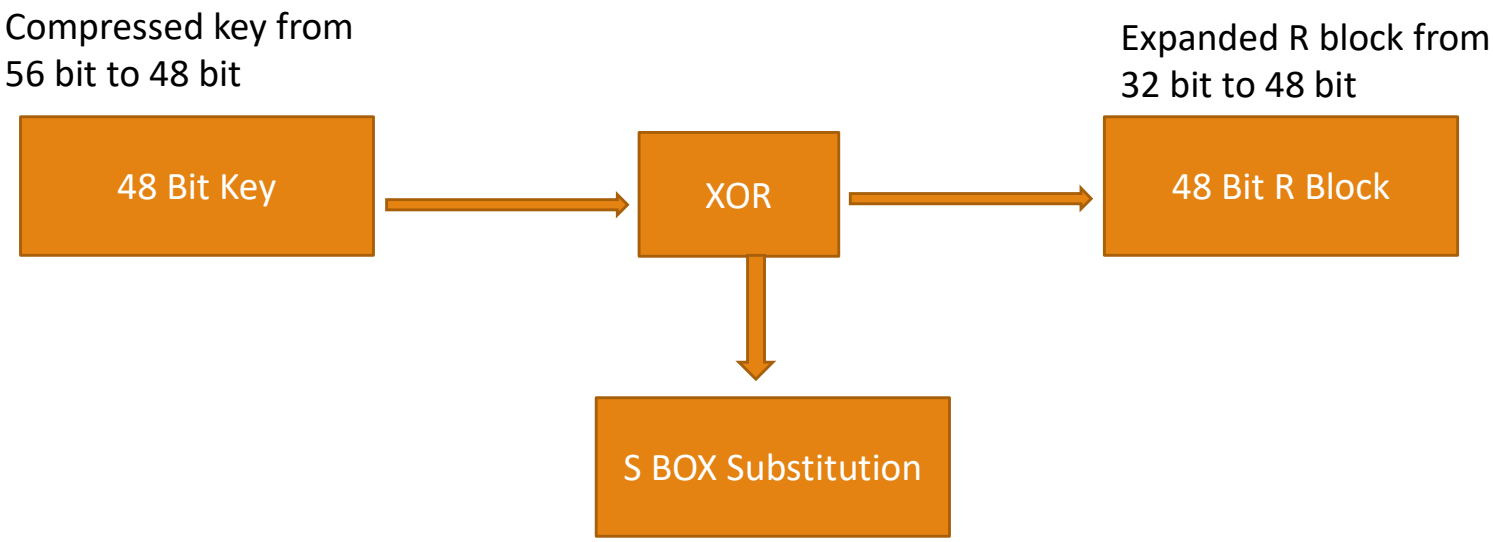
Expansion Permutation

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

E

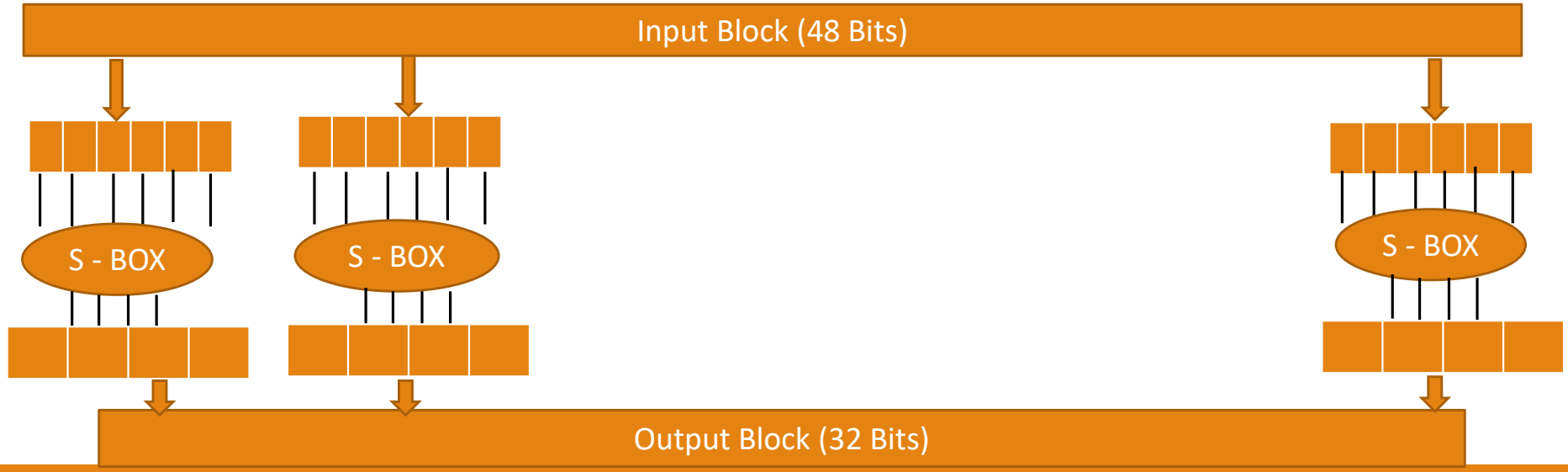
- The Key Transformation Process compresses the 56-bit key into 48 bits.
- The expansion permutation expands the R block from 32 bits to 48 bits.
- Now the the 48 bit key is XORed with 48 bit R block and resulting 48 bit output is given as an input to the next step

S-Box Substitution



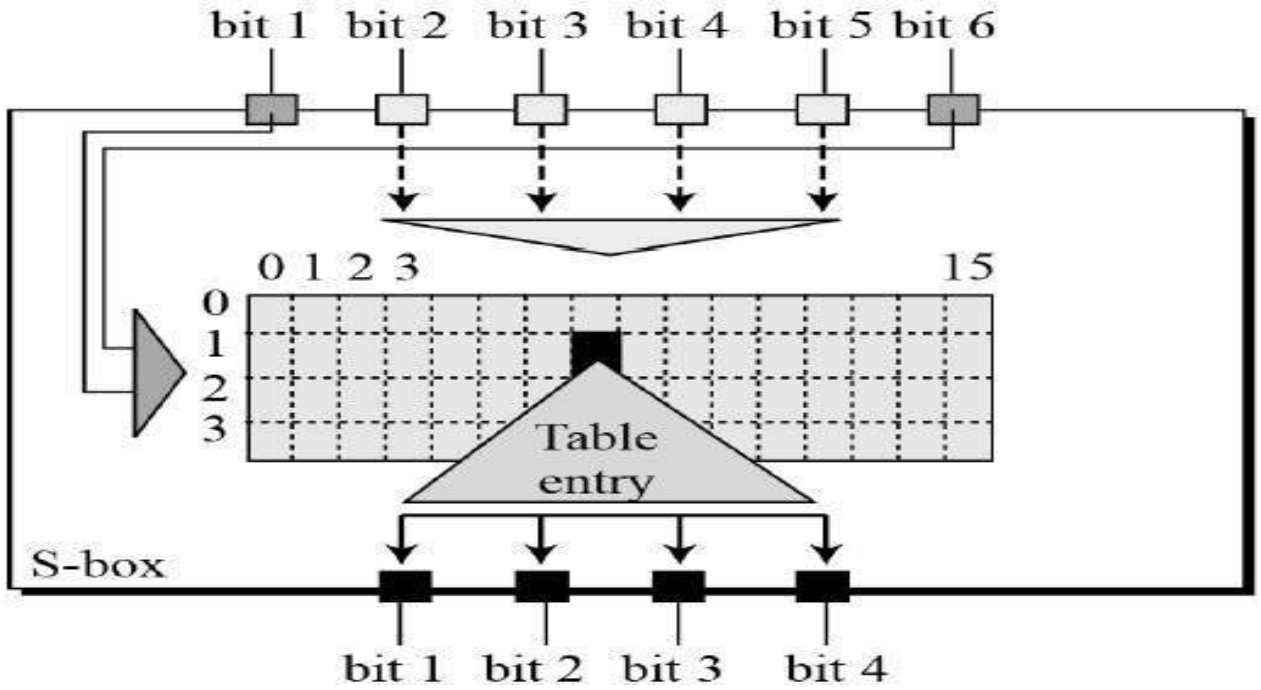
S-Box Substitution

- S-Box Substitution is performed by 8 substitution boxes (also called S-boxes).
- Each of the S-boxes have 6 bit input and a 4 bit output
- The 48 bit block is divided into 8 sub blocks of 6 bits each.



S_5																
	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

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

S-Box Substitution

Assume for Input Block 6

1	0	1	1	0	1

Example: 101101

Row No. 11 , decimal 3

Column No. 0110, decimal 6

corresponding cell has value 15 equivalent to binary 1111

P-Box Permutation

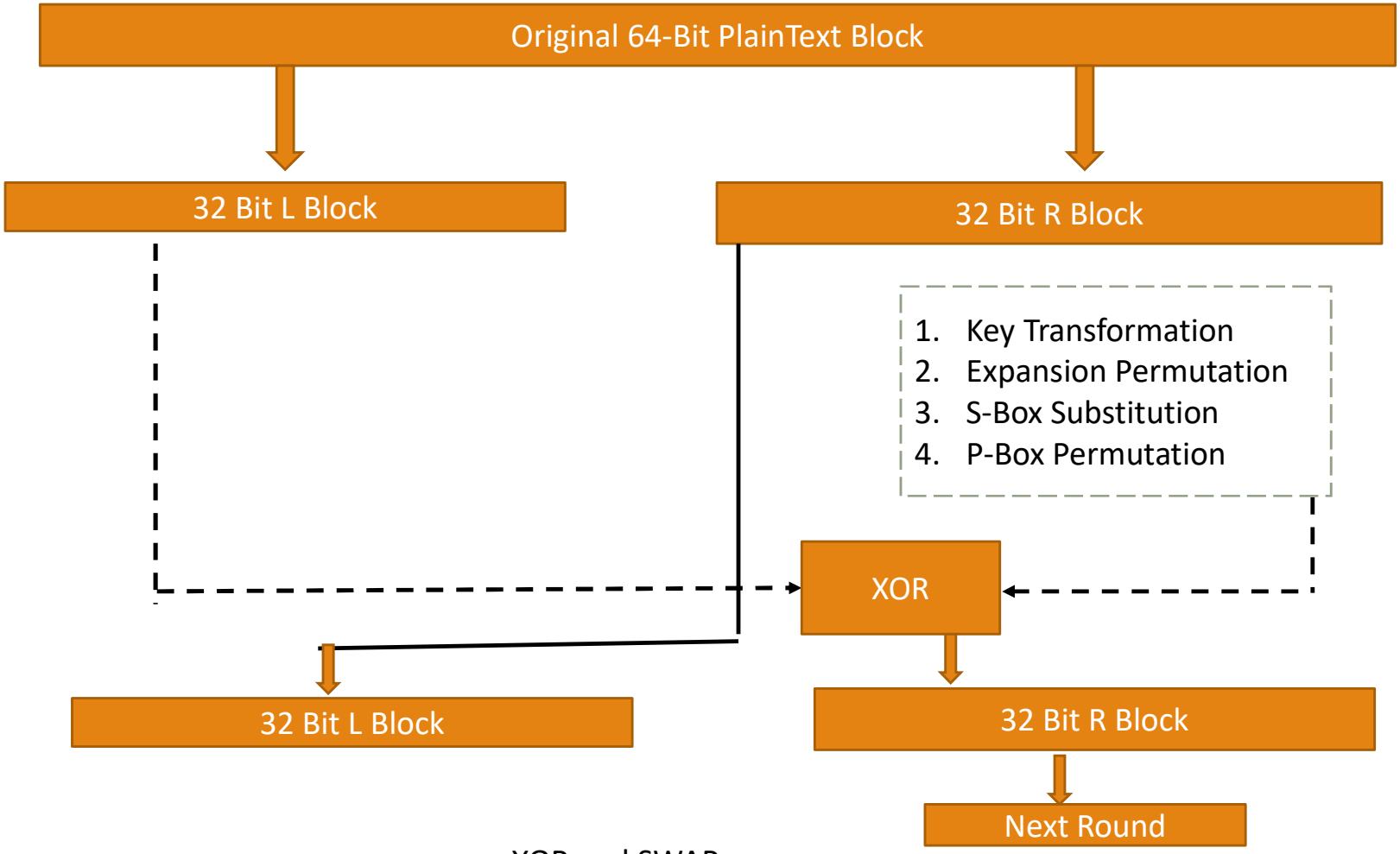
- Output of previous step are combined together to form a 32-bit block.
- This 32-bit block is used as input for next step, i.e., P –box Permutation.
- These 32 bits are permuted using a P-Box.
- This is a simple permutation or transposition of bits in a specified order without any expansion or compression as follows:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

P Box Permutation

XOR and SWAP

- All the steps were performed on 32 bit R block.
- In this step L block (32 bit) after Initial Permutation is XORed with the 32-bit Output of P-Box Permutation
- The output is the new R block and the original R block after IP becomes the new L-Block for the next Round.


Original 64-Bit PlainText Block
32 Bit L Block
32 Bit R Block
1. Key Transformation
2. Expansion Permutation
3. S-Box Substitution
4. P-Box Permutation
XOR
32 Bit L Block
32 Bit R Block
Next Round

XOR and SWAP

Final Permutation

- At the end of 16 rounds, the **Final Permutation** is performed (Only Once).

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

Final Permutation

DES Decryption

- Decryption uses the same algorithm as encryption, except that the subkeys K1, K2,…K16 are applied in reversed order
- The values of the various tables and the operations as well as their sequence are so carefully chosen that the algorithm is reversible.

The Strengths of DES

- Strength of DES lies in its key, which must be kept secret.
- DES uses a 56-Bit key, so there are 2^{56} possible keys which is approximately 7.2×10^{16} , Therefore, Brute Force Approach appears impractical at that time.

Distribution of Public Keys and Digital Certificates

DR. VASUDHA ARORA

VASUDHA.ARORA@GDGU.ORG, VASUDHARORA6@GMAIL.COM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GD GOENKA UNIVERSITY, GURUGRAM

The Critical Questions???

How can the recipient know with certainty the sender's public key? (to validate a digital signature)

How can the sender know with certainty the recipient's public key? (to send an encrypted message)

Distribution of Public Keys

Before two parties exchange data using Public Key cryptography, each wants to be sure that the other party is authenticated

Before B accepts a message with A's Digital Signature, B wants to be sure that the public key belongs to A and not to someone masquerading as A on an open network

To be sure one of the following methods can be considered for distribution and management of public private key pairs:

- public announcement
- publicly available directory
- public-key authority
- public-key certificates

Public Announcement

- users distribute public keys to recipients or broadcast to community at large
- eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
- anyone can create a key claiming to be someone else and broadcast it
- until forgery is discovered can masquerade as claimed user

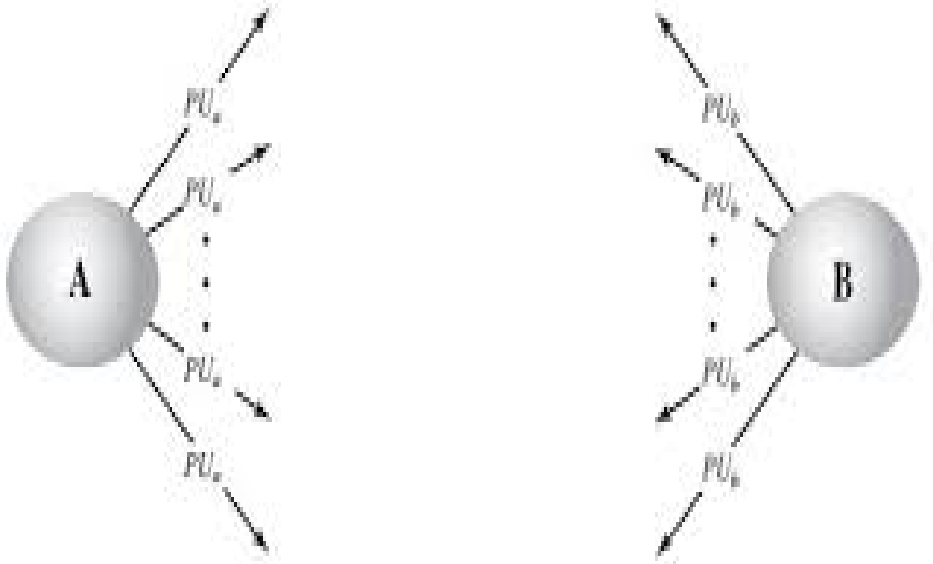
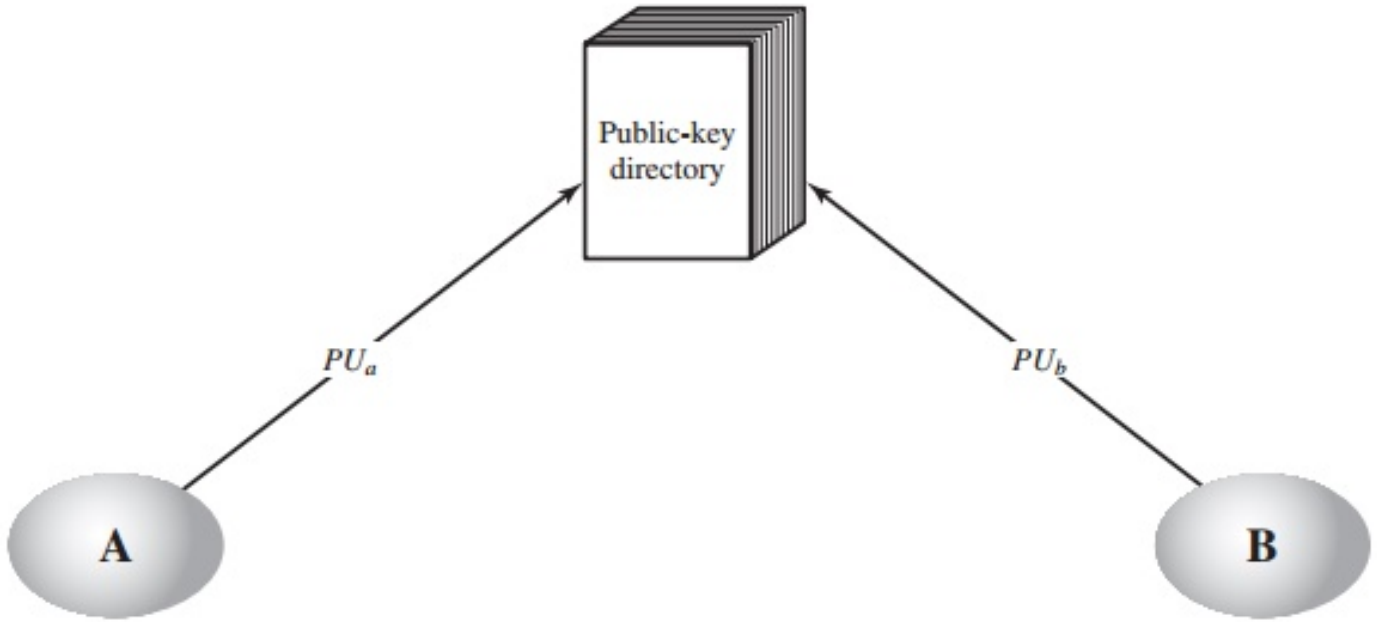


Figure 14.9 Uncontrolled Public Key Distribution

publicly available directory



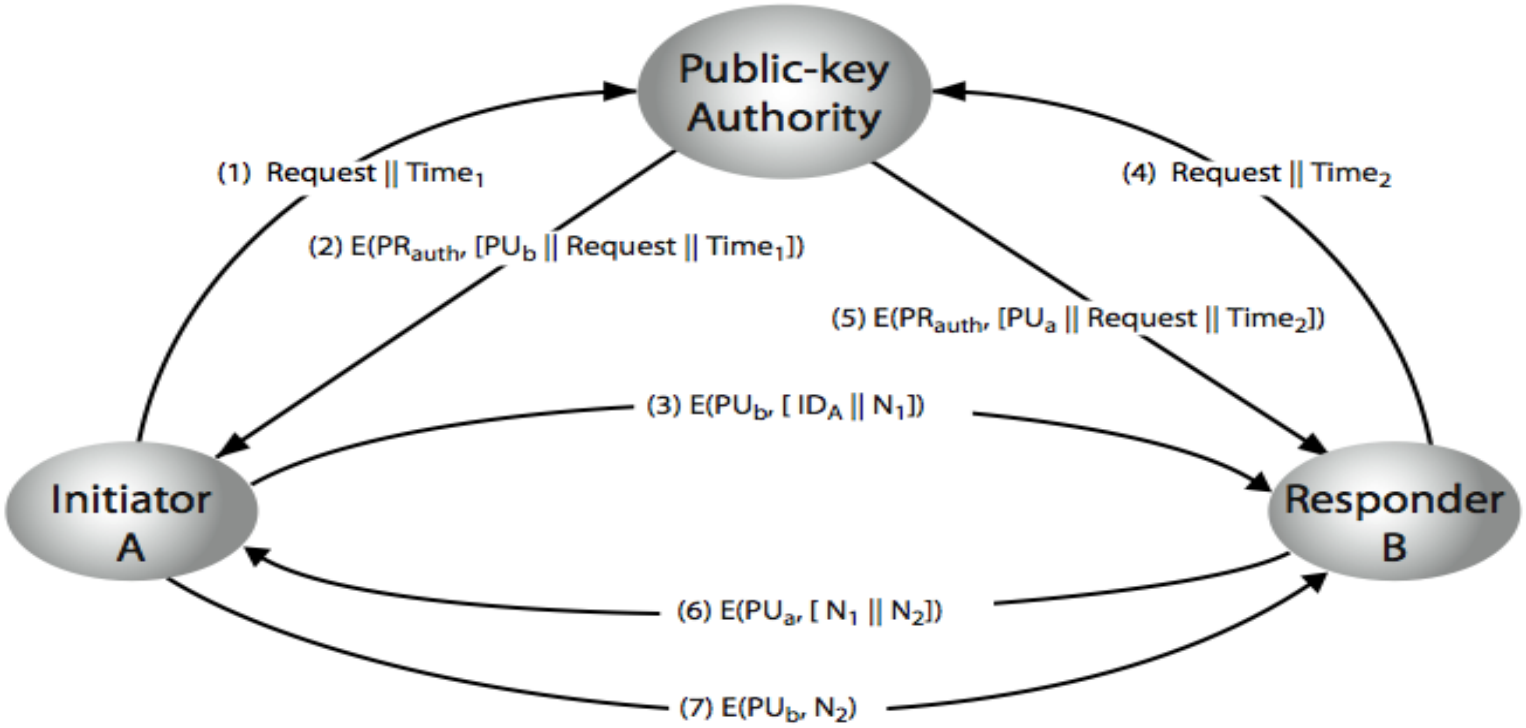
publicly available directory

- The authority maintains a directory with a {name, public key} entry for each participant.
- Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
- A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
- Periodically, the authority publishes the entire directory or updates to the directory. For example, a hard-copy version much like a telephone book could be published, or updates could be listed in a widely circulated newspaper.
- Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory

publicly available directory

- can obtain greater security by registering keys with a public directory
- still vulnerable to tampering or forgery

public-key authority



public-key authority

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, PRauth. Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
   - B's public key, PUb, which A can use to encrypt messages destined for B
   - The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
   - The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely.

4. , 5 B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

public-key authority

At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2). Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.

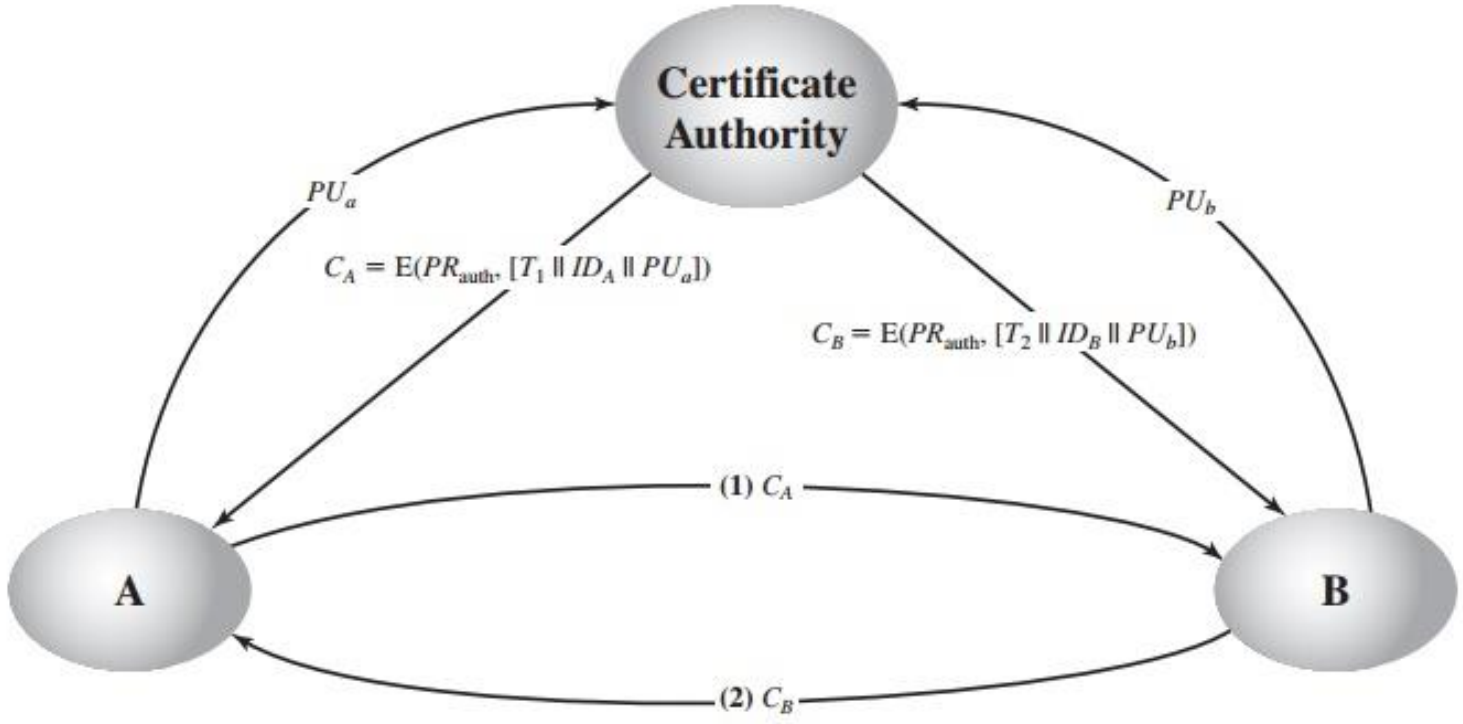
7. A returns N2, which is encrypted using B's public key, to assure B that its correspondent is A.

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use—a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

public-key authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed
- The public-key
  authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

public-key certificates/Digital Ceritificates



public-key certificates/Digital Ceritificates

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.

2.Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.

3. Only the certificate authority can create and update certificates.

4.Any participant can verify the currency of the certificate.

Stream Ciphers & Block Ciphers

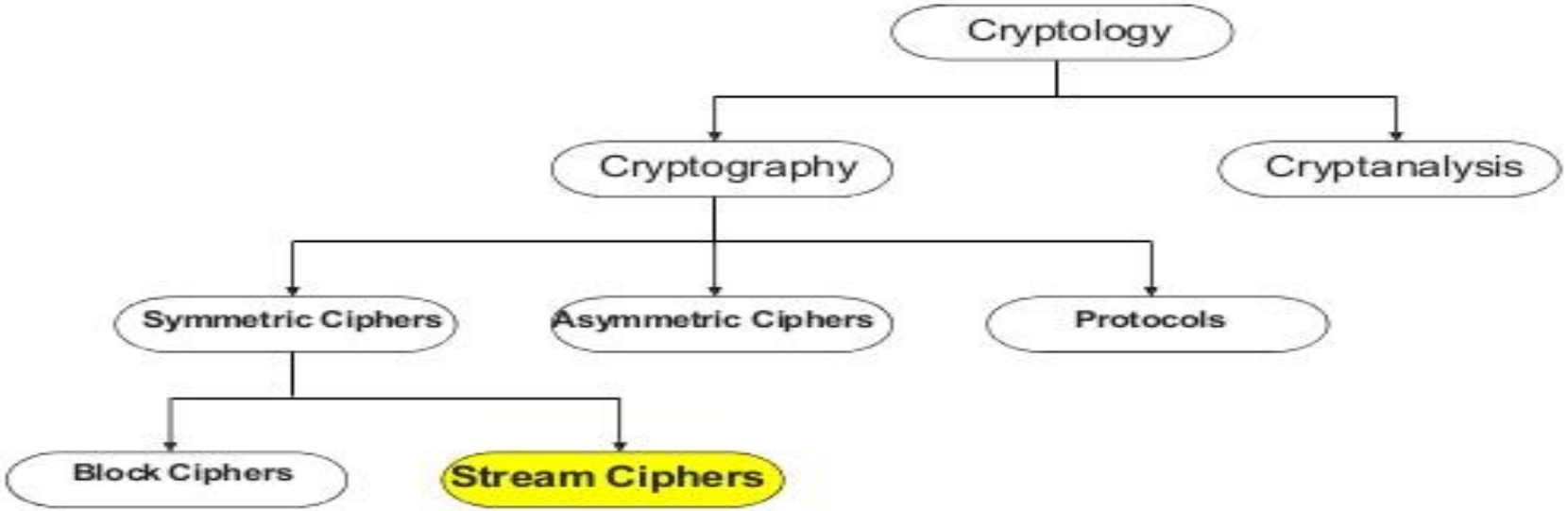
DR. VASUDHA ARORA

VASUDHA.ARORA@GDGU.ORG, VASUDHARORA6@GMAIL.COM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GD GOENKA UNIVERSITY, GURUGRAM

- **Stream Ciphers in the Field of Cryptology**



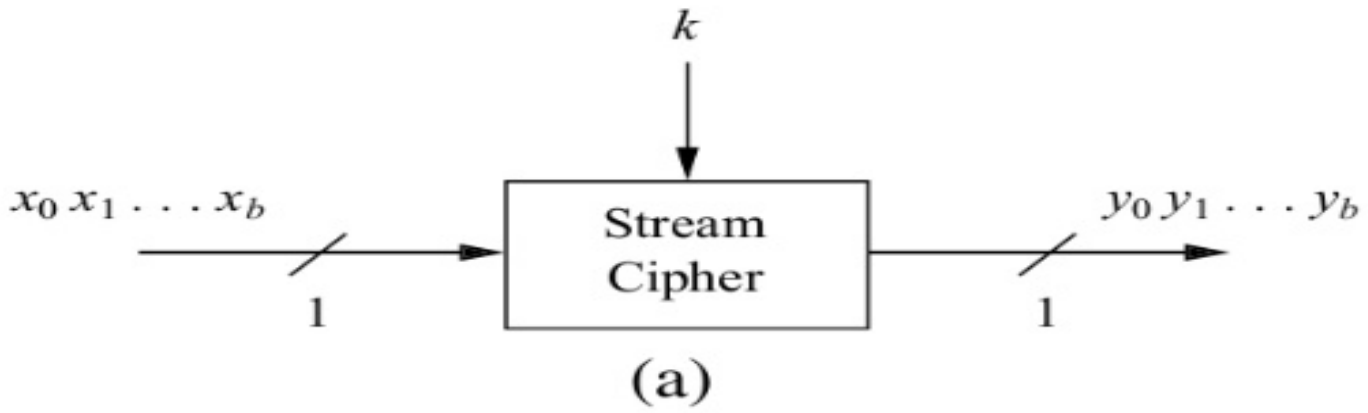
Stream Ciphers were invented in 1917 by Gilbert Vernam

Stream Ciphers

- Stream ciphers are a special class of ciphers in which the encryption and decryption algorithm is applied to the individual bits or bytes of the plain-text.
- Stream ciphers are especially well suited for encrypting and decrypting the type of data that is used in network communication systems-data in transit.
- Some examples of a stream cipher algorithm are the RC4 cipher and the A5 algorithm that is used in cellular-based Global System for Mobile (GSM) communications.
- For a stream cipher implementation to remain secure, its pseudorandom generator should be unpredictable and the key should never be reused.
- Stream ciphers are designed to approximate an idealized cipher, known as the **One-Time Pad (OTP).**
- The algorithm works by combining the plain-text bits or bytes with a pseudo-random bit stream, one bit or byte at a time.

Stream Ciphers

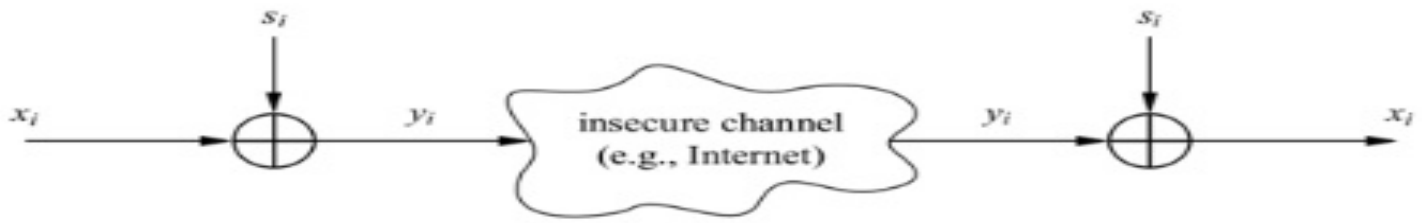
- **Stream Cipher**



Stream Ciphers

- **Encryption and Decryption with Stream Ciphers**

Plaintext x_i, ciphertext y_i and key stream s_i consist of individual bits

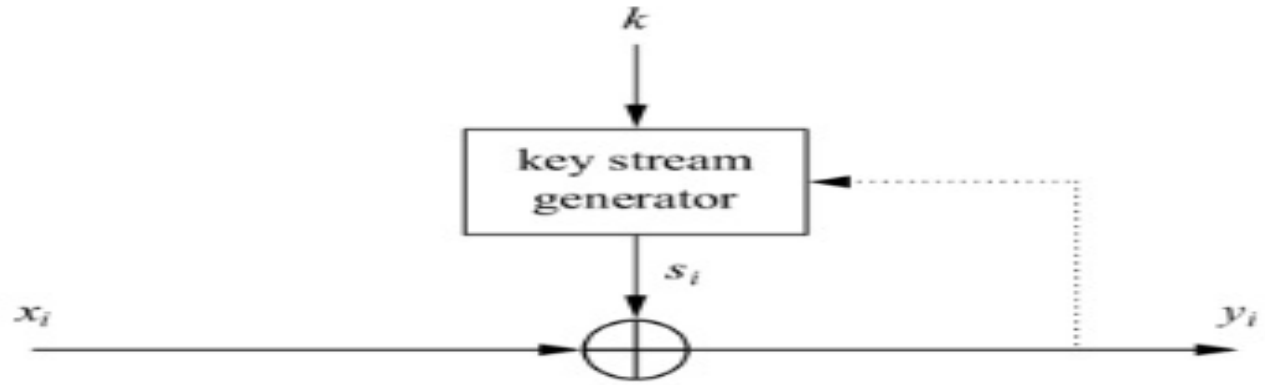


- Encryption and decryption are simple additions modulo 2 (aka XOR)
- Encryption and decryption are the same functions

- **Encryption:** $y_i = e_{s_i}(x_i) = x_i + s_i \bmod 2$ $\qquad x_i, y_i, s_i \in \{0,1\}$
- **Decryption:** $x_i = e_{s_i}(y_i) = y_i + s_i \bmod 2$

Stream Ciphers

- **Synchronous vs. Asynchronous Stream Cipher**



  - Security of stream cipher depends entirely on the key stream s_i:
    - Should be **random** , i.e., $Pr(s_i = 0) = Pr(s_i = 1) = 0.5$
    - Must be **reproducible** by sender and receiver
  - **Synchronous Stream Cipher**
    - Key stream depend only on the key (and possibly an initialization vector IV)
  - **Asynchronous Stream Ciphers**
    - Key stream depends also on the ciphertext (dotted feedback enabled)

Stream Ciphers

- **Why is Modulo 2 Addition a Good Encryption Function?**
  - Modulo 2 addition is equivalent to XOR operation
  - For perfectly random key stream s_i, each ciphertext output bit has a 50% chance to be 0 or 1

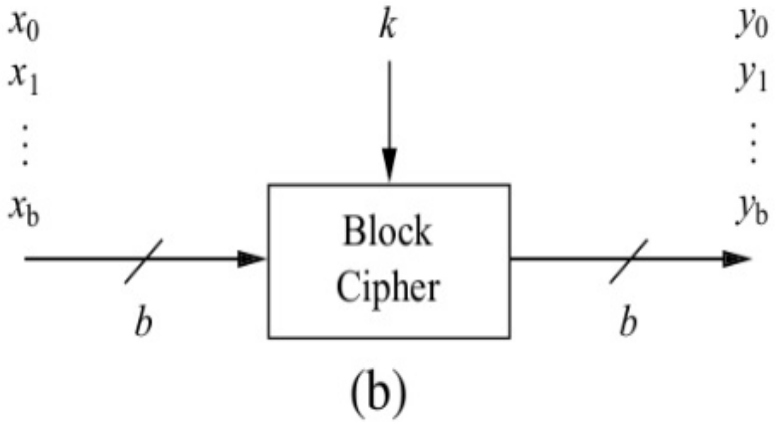
    Good statistic property for ciphertext
  - Inverting XOR is simple, since it is the same XOR operation

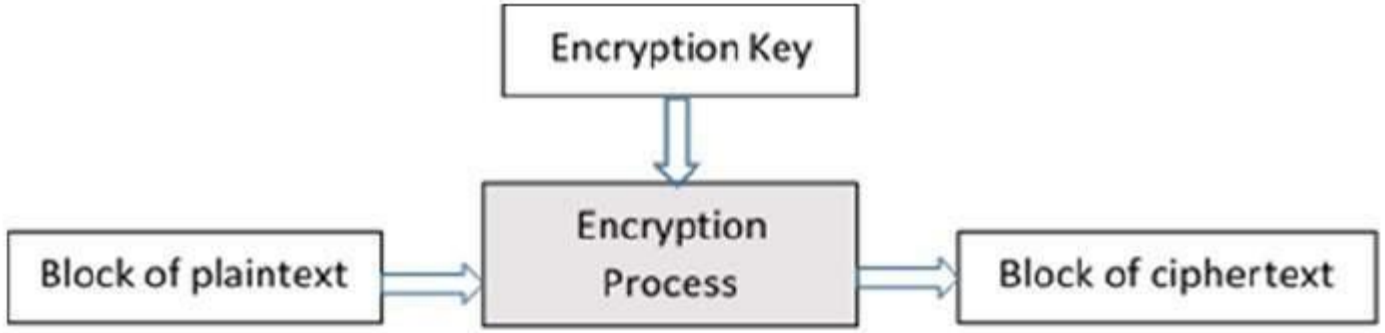
x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

Chapter 2 of Understanding Cryptography by Christof Paar and Jan Pelzl

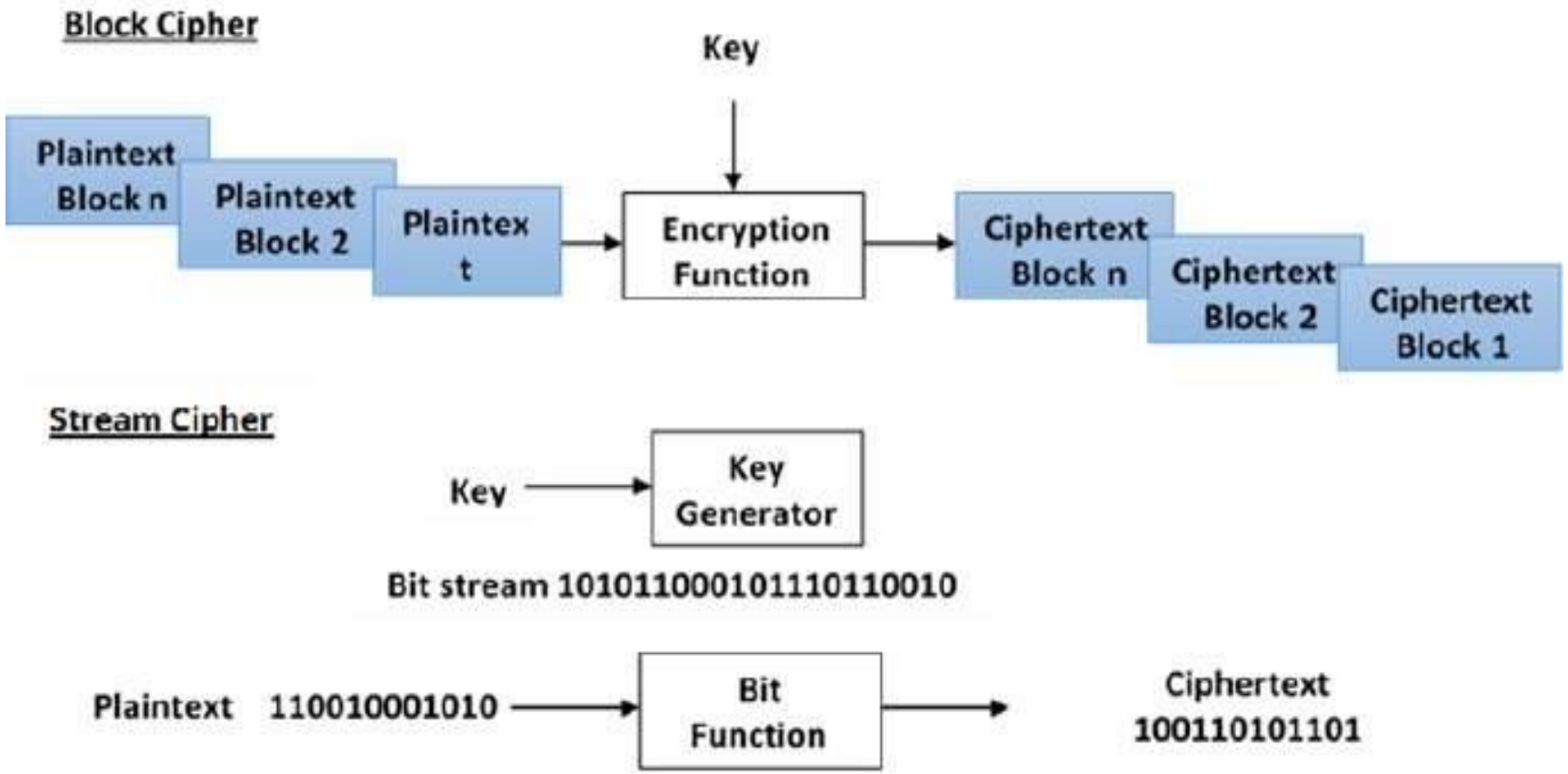
Block Ciphers

- **Block Cipher**





Stream Cipher Vs Block Cipher



Block Ciphers

- A block cipher takes a block of plaintext bits and generates a block of ciphertext bits, generally of same size.
- The size of block is fixed in the given scheme.
- The choice of block size does not directly affect to the strength of encryption scheme. The strength of cipher depends up on the key length.

Block Size

Though any size of block is acceptable, following aspects are borne in mind while selecting a size of a block.

Avoid very small block size − Say a block size is m bits. Then the possible plaintext bits combinations are then 2^m. If the attacker discovers the plain text blocks corresponding to some previously sent ciphertext blocks, then the attacker can launch a type of 'dictionary attack' by building up a dictionary of plaintext/ciphertext pairs sent using that encryption key. A larger block size makes attack harder as the dictionary needs to be larger.

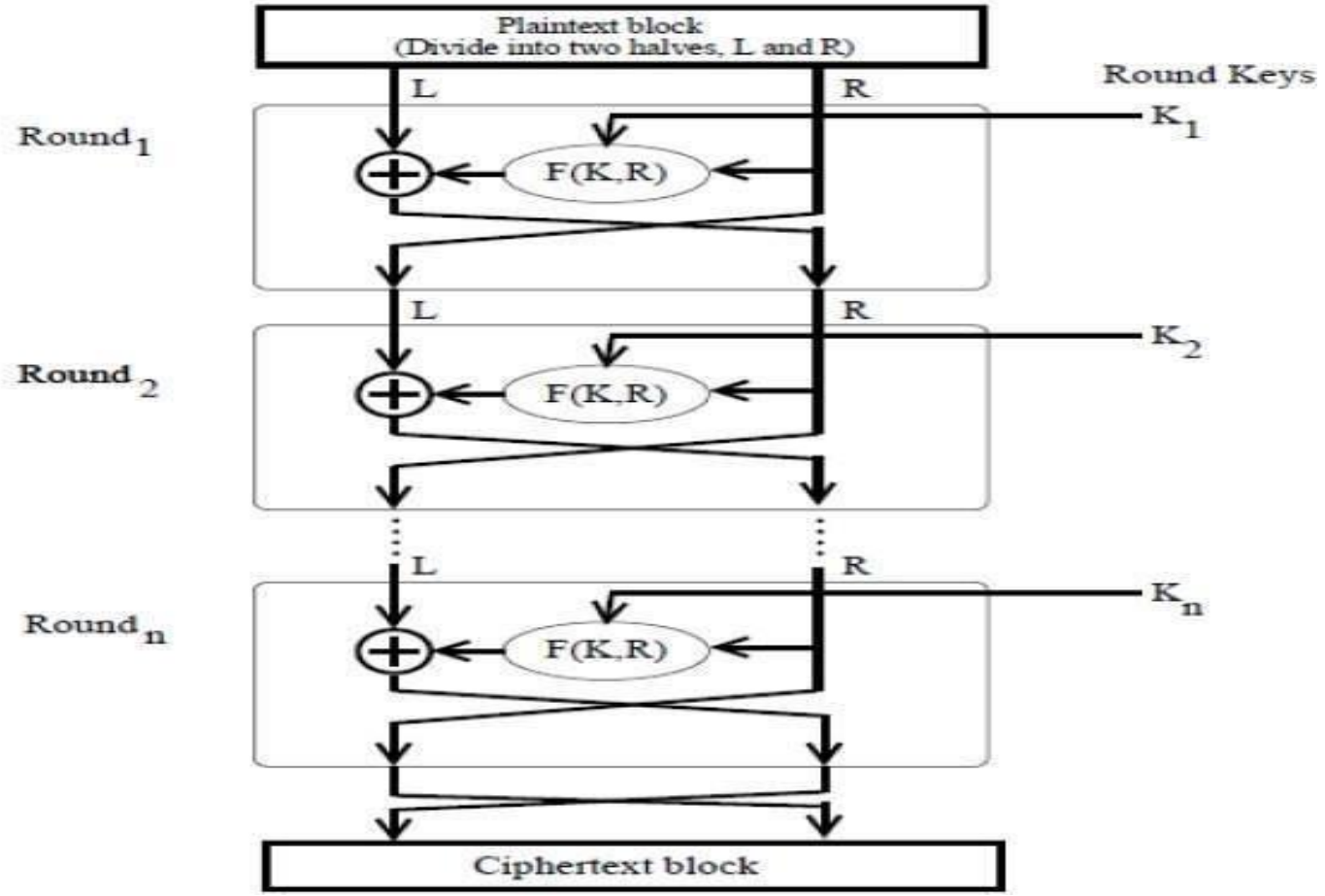
Do not have very large block size − With very large block size, the cipher becomes inefficient to operate. Such plaintexts will need to be padded before being encrypted.

Multiples of 8 bit − A preferred block size is a multiple of 8 as it is easy for implementation as most computer processor handle data in multiple of 8 bits.

Padding in Block Cipher

- Block ciphers process blocks of fixed sizes (say 64 bits).
- The length of plaintexts is mostly not a multiple of the block size. For example, a 150-bit plaintext provides two blocks of 64 bits each with third block of balance 22 bits. The last block of bits needs to be padded up with redundant information so that the length of the final block equal to block size of the scheme. In our example, the remaining 22 bits need to have additional 42 redundant bits added to provide a complete block. The process of adding bits to the last block is referred to as **padding**.
- Too much padding makes the system inefficient. Also, padding may render the system insecure at times, if the padding is done with same bits always.

Feistel Structure for Block Cipher



Feistel Structure – Encryption Process

- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.
- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function ‘f’ that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a subkey) is derived from the encryption key. This means that each round uses a different key, although all these subkeys are related to the original key.
- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a ‘round’. The number of rounds are specified by the algorithm design.
- Once the last round is completed then the two sub blocks, ‘R’ and ‘L’ are concatenated in this order to form the ciphertext block.

Feistel Structure – Decryption Process

- The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as encryption process.
- The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.
- The final swapping of ‘L’ and ‘R’ in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.