

# UNIT-1

# Introduction



# Outline

---

- OSI Security Architecture
- Security Attacks
- Security Services
- Security Mechanism
- Symmetric Cipher Model
- Cryptography
- Cryptanalysis and Attacks
- Substitution and Transposition Techniques

# Introduction to Information & N/W Security

---



# OSI Security Architecture

---

- The OSI (Open Systems Interconnection) security architecture focuses on Security Attacks, Mechanisms, and Services.
- **Security Attack:** Any action that compromises the security of information owned by an organization.
- **Security Mechanism:** A process that is designed to detect, prevent, or recover from a security attack.
- **Security Service:** A communication service that enhances the security of the data processing systems and the information transfers of an organization.

# Security Objectives

---

- Security objectives for information and computing services are Confidentiality, Integrity, Availability, Authenticity, Accountability.

## 1) Confidentiality:

- **Data confidentiality:** Assures that private or confidential information is not made available or disclosed to unauthorized individuals.
- **Privacy:** Assures that individuals control what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

# Security Objectives (Cont...)

---

## 2) Integrity:

- **Data integrity:** Assures that information and programs are changed only in a specified and authorized manner.
- **System integrity:** Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

## 3) Availability: Assures that systems work promptly and service is not denied to authorized users.

# Security Objectives (Cont...)

---

## 4) **Authenticity:**

- The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator.
- This means verifying that each input arriving at the system came from a trusted source.

## 5) **Accountability:**

- The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity.
- This supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action.

# Threat and Attack

---

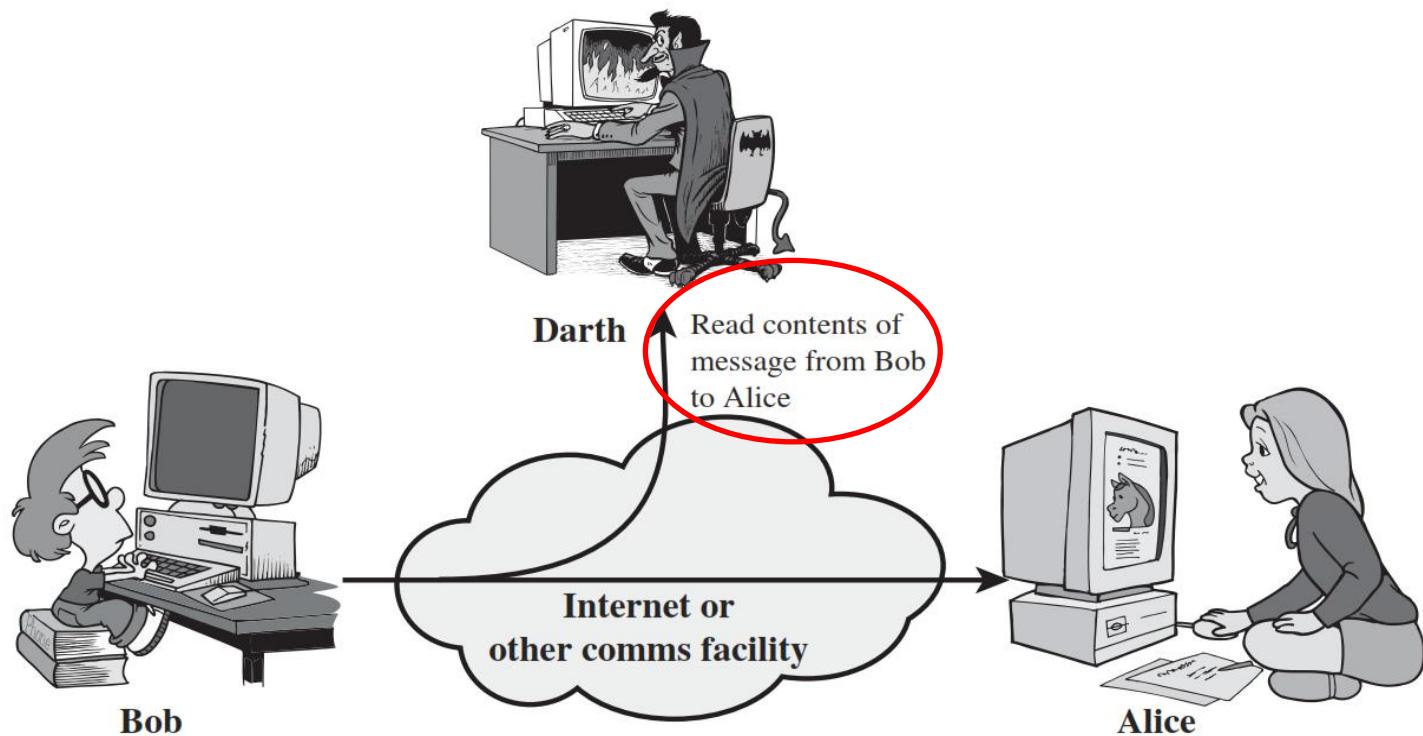
- **Threat:** A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could crack security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.
- **Attack:** An violation on system security that derives from an intelligent threat; that is, an intelligent act that is a calculated attempt to avoid security services and violate the security policy of a system.

# Security Attacks

---

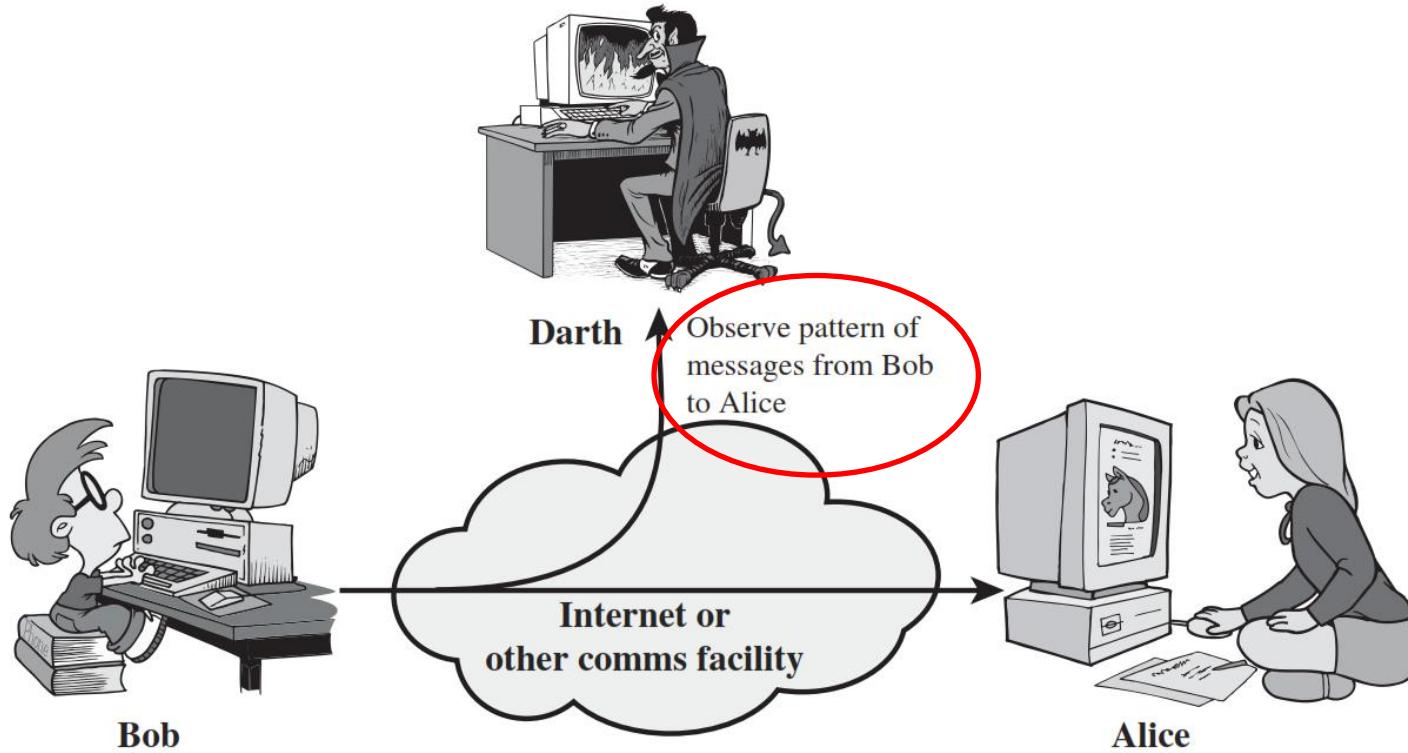
- A **passive attack** attempts to learn or make use of information from the system but does not affect system resources.
  1. Release of message contents
  2. Traffic analysis
- An **active attack** attempts to alter system resources or affect their operation.
  1. Masquerade
  2. Replay
  3. Modification of messages
  4. Denial of service.

# 1) Release of message contents (Passive Attack)



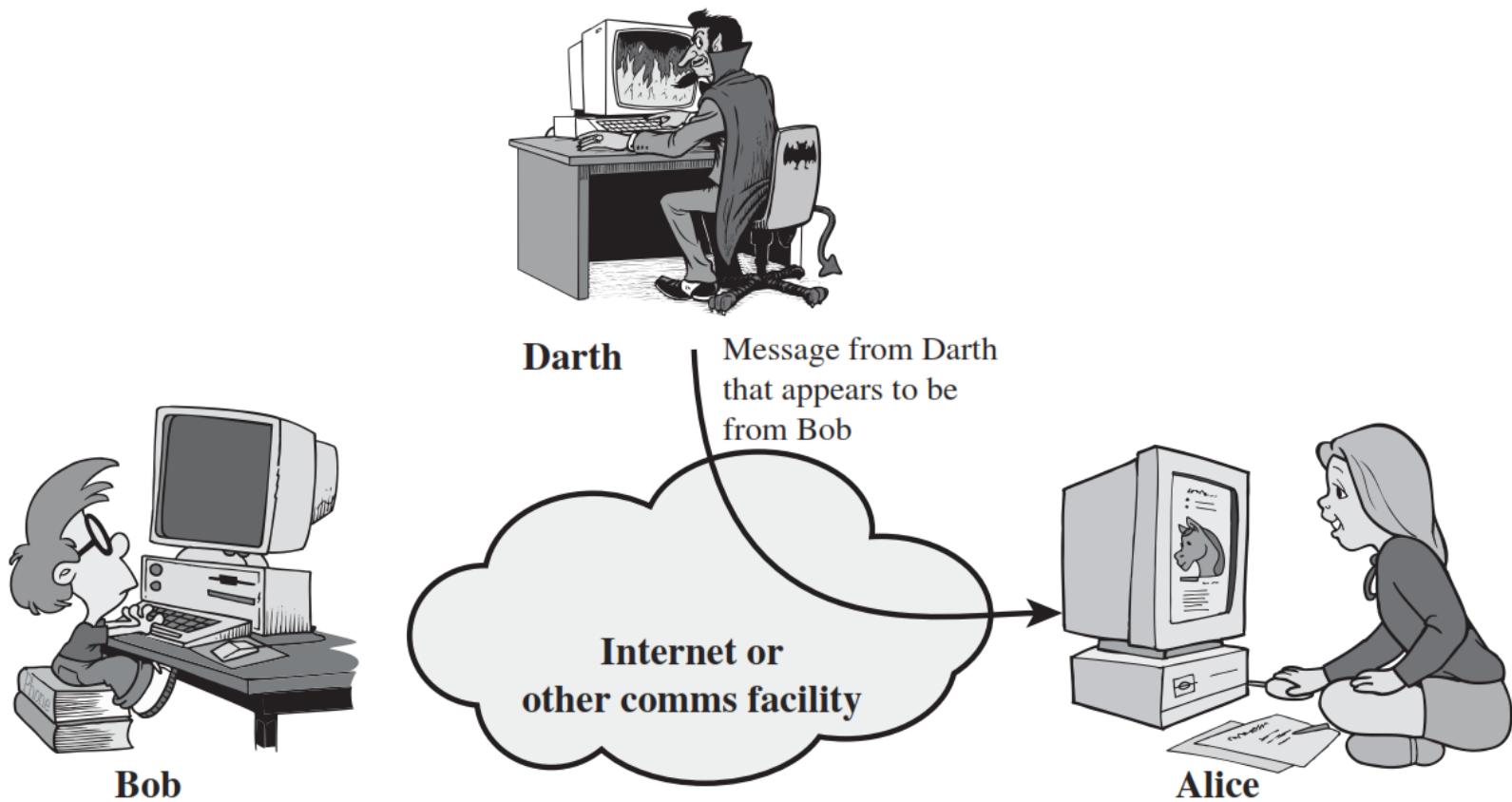
- A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.
- We would like to prevent an opponent from learning the contents of these transmissions.

## 2) Traffic Analysis (Passive Attack)



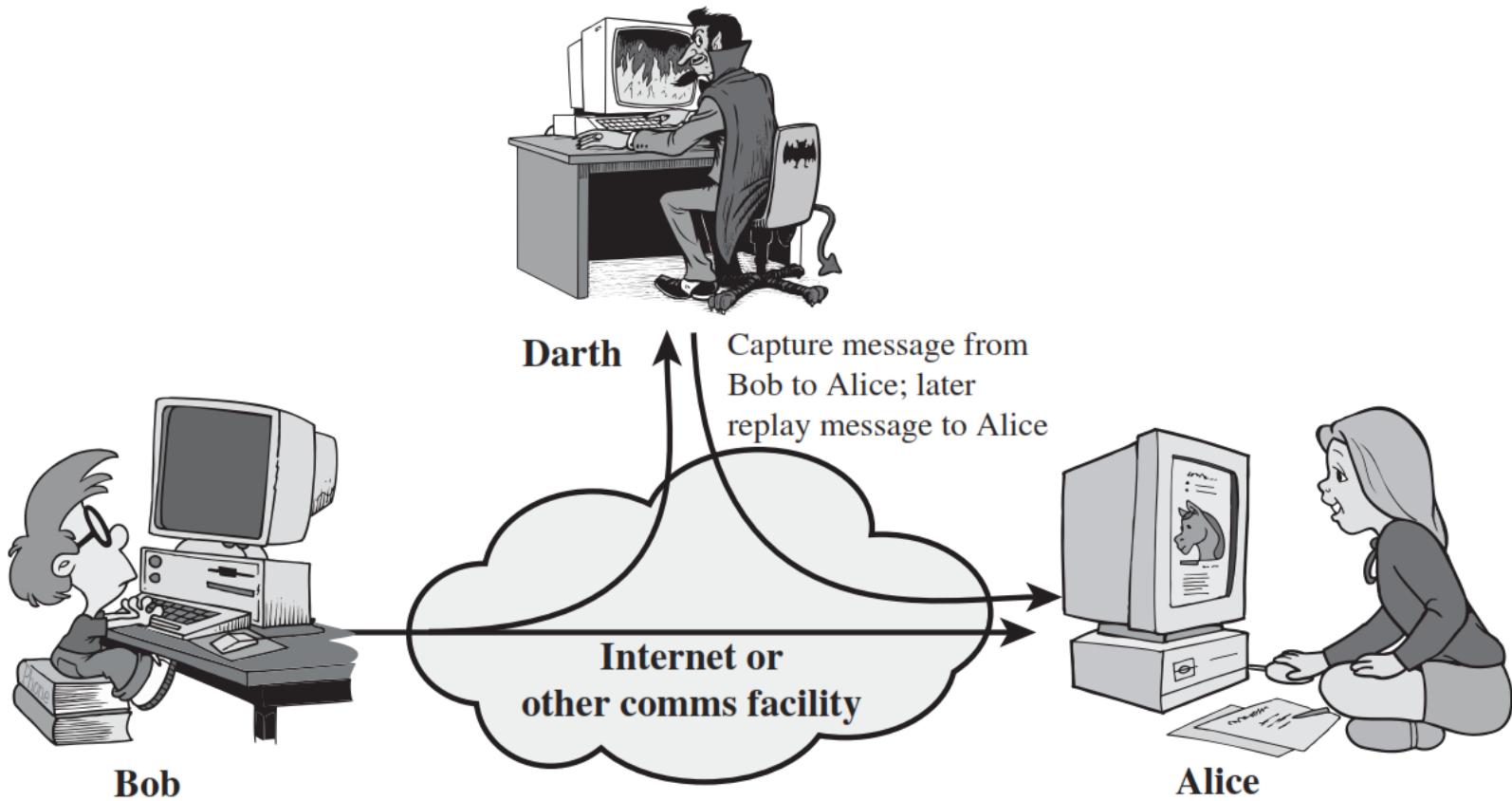
- In such attacks, an adversary, capable of observing network traffic statistics in several different networks, correlates the traffic patterns in these networks.

# 1) Masquerade Attack (Active Attack)



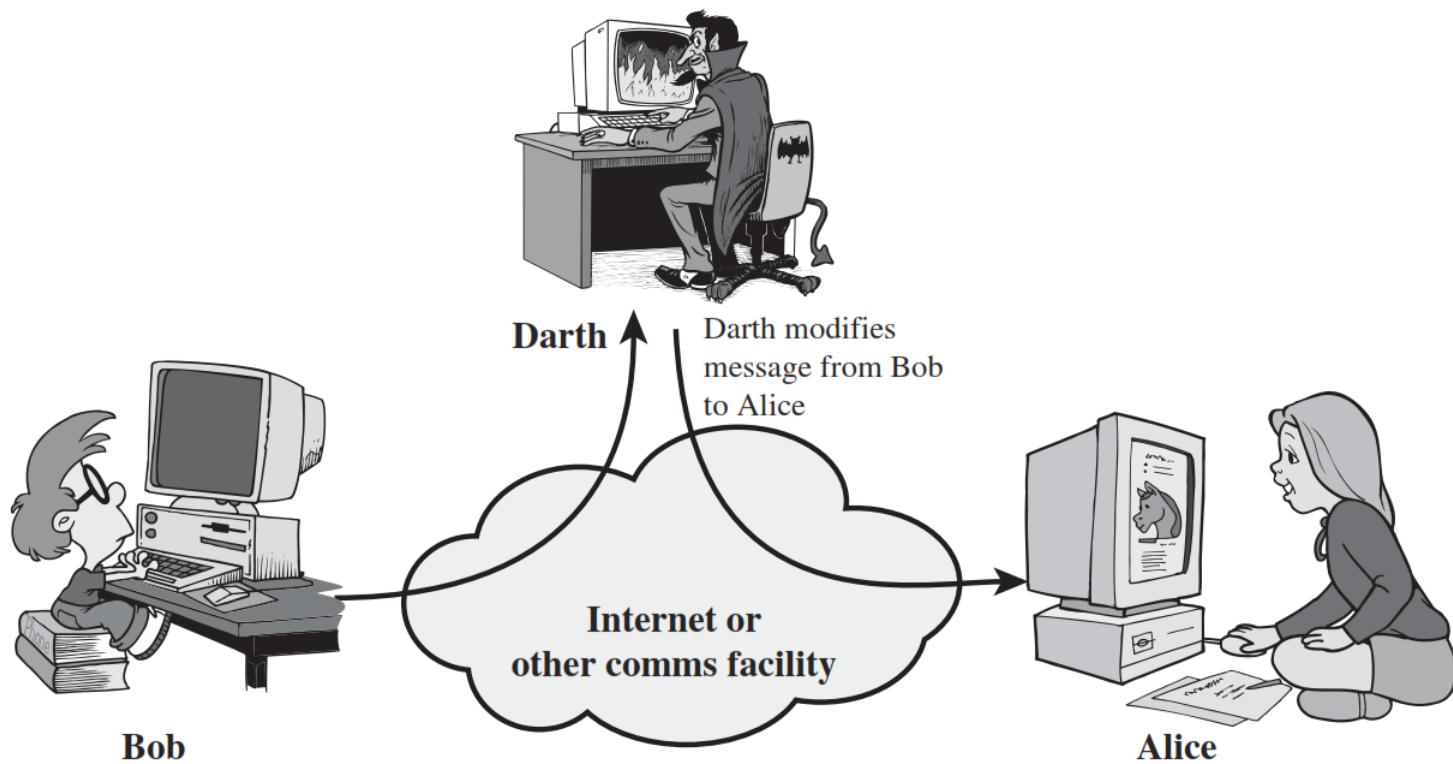
- A **masquerade** takes place when one entity pretends to be a different entity.

## 2) Replay Attack (Active Attack)



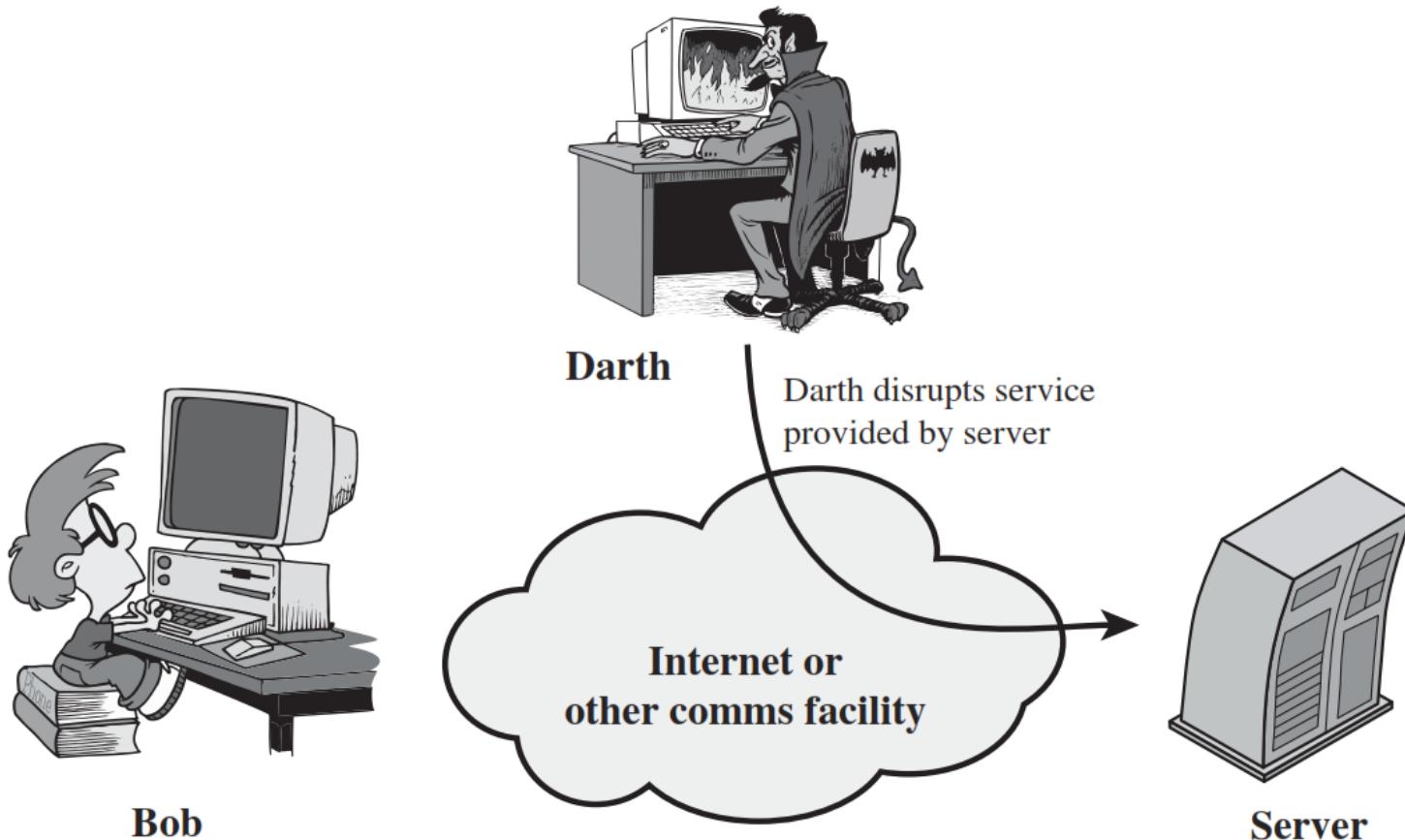
- **Replay attack** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

### 3) Modification of messages Attack (Active Attack)



- **Modification of messages** simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.

# 4) Denial of Service Attack (Active Attack)

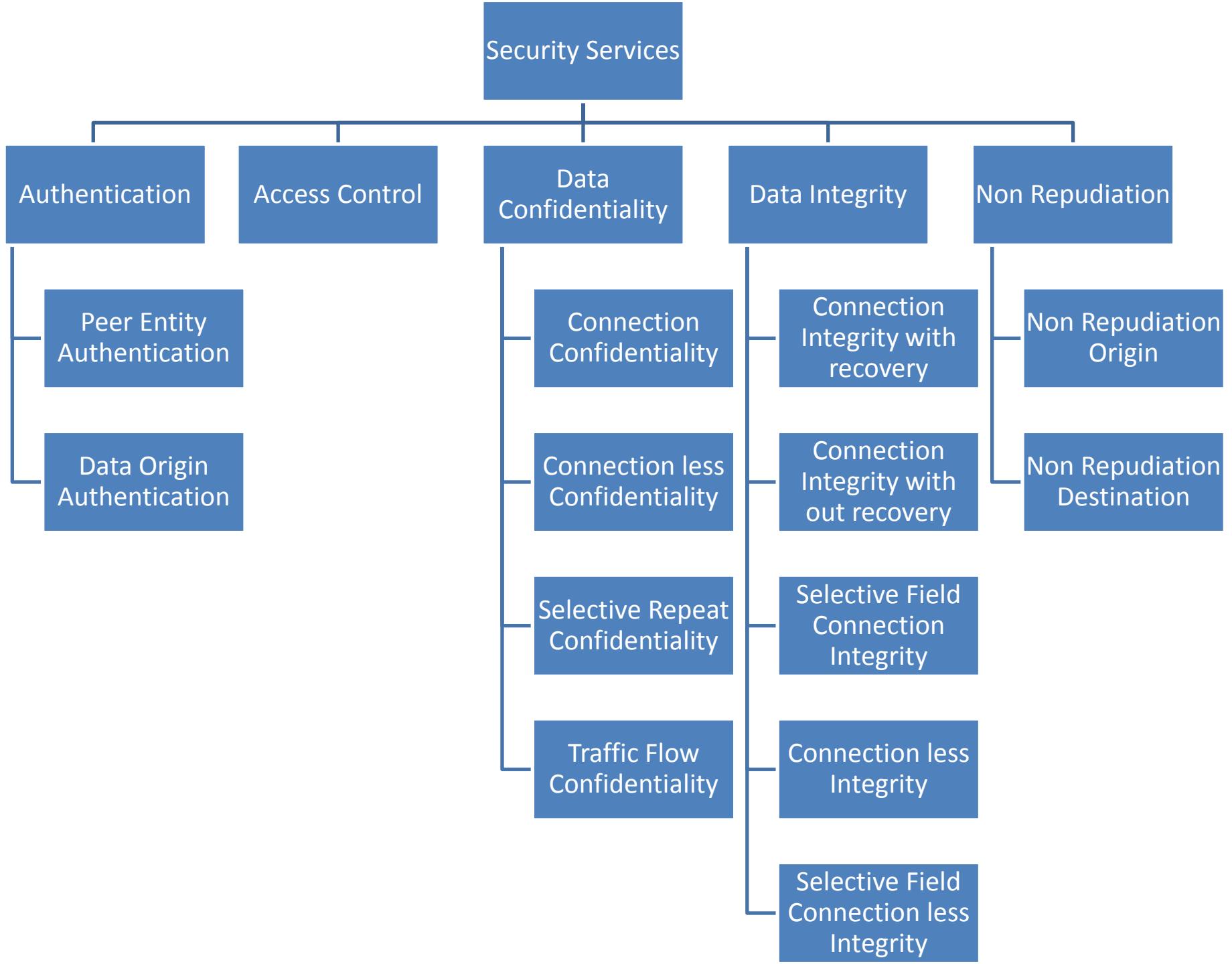


- **The denial of service** attack prevents the normal use or management of communications facilities.

# Security Services (X.800)

---

- X.800 standard defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures security of the systems or of data transfers.



# Authentication

- **Authentication** is the assurance that the communicating entity is the one that it claims to be.

## 1. Peer Entity Authentication:

Used in association with a logical connection to provide confidence in the identity of the entities connected.

## 2. Data-Origin Authentication:

In a connectionless transfer, provides assurance that the source of received data is as claimed.

Who you are ?  
(biometrics)



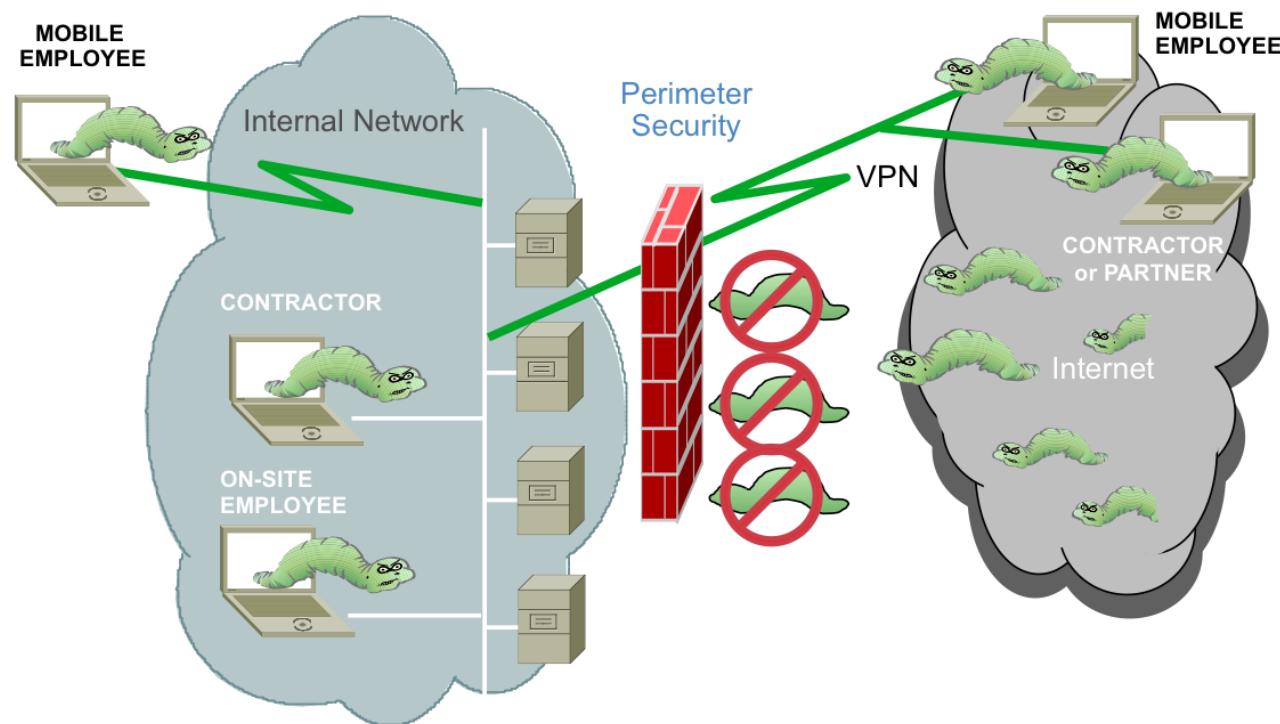
Physical  
authentication  
where you are ?



What you know ?  
**Password**  
**One-time Passwords**  
**Network address**

# Access Control

- **Access control** is the prevention of unauthorized use of a resource
- This service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).



# Data Confidentiality

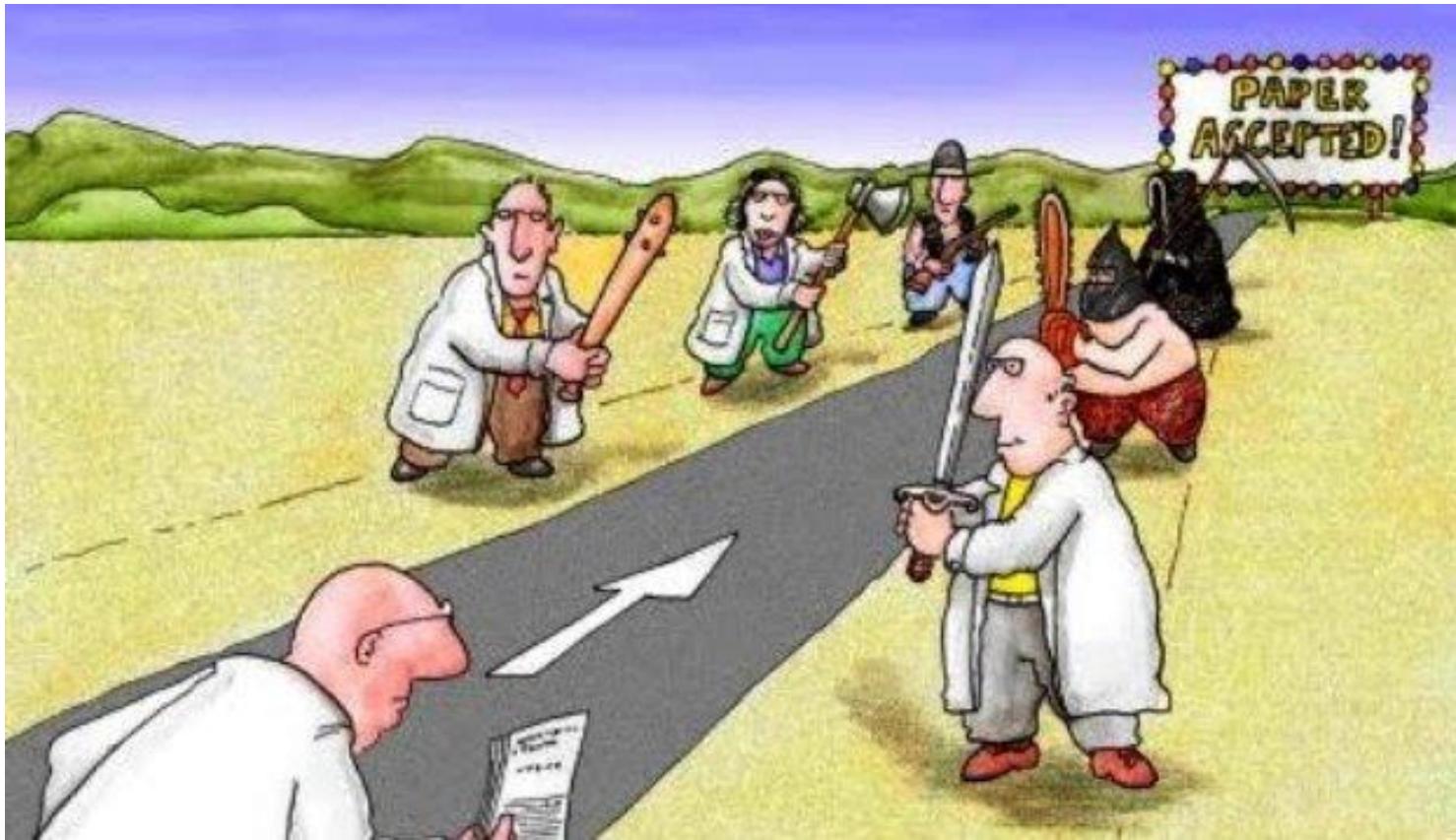
- **Data confidentiality** is the protection of data from unauthorized disclosure.
  1. **Connection Confidentiality:** The protection of all user data on a connection.
  2. **Connectionless Confidentiality:** The protection of all user data in a single data block.
  3. **Selective-Field Confidentiality:** The confidentiality of selected fields within the user data on a connection or in a single data block.
  4. **Traffic-Flow Confidentiality:** The protection of the information that might be derived from observation of traffic flows.



# Data Integrity

---

- Data integrity is the assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).



# Data Integrity (Cont...)

---

- **Connection Integrity with Recovery:** Provides integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data with recovery attempted.
- **Connection Integrity without Recovery:** As above, but provides only detection without recovery.
- **Selective-Field Connection Integrity:** Provides integrity of selected fields within the user data and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

# Data Integrity (Cont...)

---

- **Connectionless Integrity:** Provides integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.
- **Selective-Field Connectionless Integrity:** Provides integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

# Non Repudiation

- **Nonrepudiation** is the assurance that someone cannot deny something.
- Typically, nonrepudiation refers to the ability to ensure that a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.



User A

Transfer Rs. 1,00,000  
To Bank

After few days

I have never  
requested to transfer  
Rs. 1,00,000  
to Bank



Bank

# Non Repudiation (Cont...)

---

- **Nonrepudiation-Origin:** Proof that the message was sent by the specified party.
- **Nonrepudiation-Destination:** Proof that the message was received by the specified party.

# Security Mechanisms (X.800)

---

- **Specific security mechanisms:** Integrated into the appropriate protocol layer in order to provide some of the OSI security services.
- **Pervasive security mechanisms:** Not integrated to any particular OSI security service or protocol layer

# Security Mechanism (Specific Security)

---

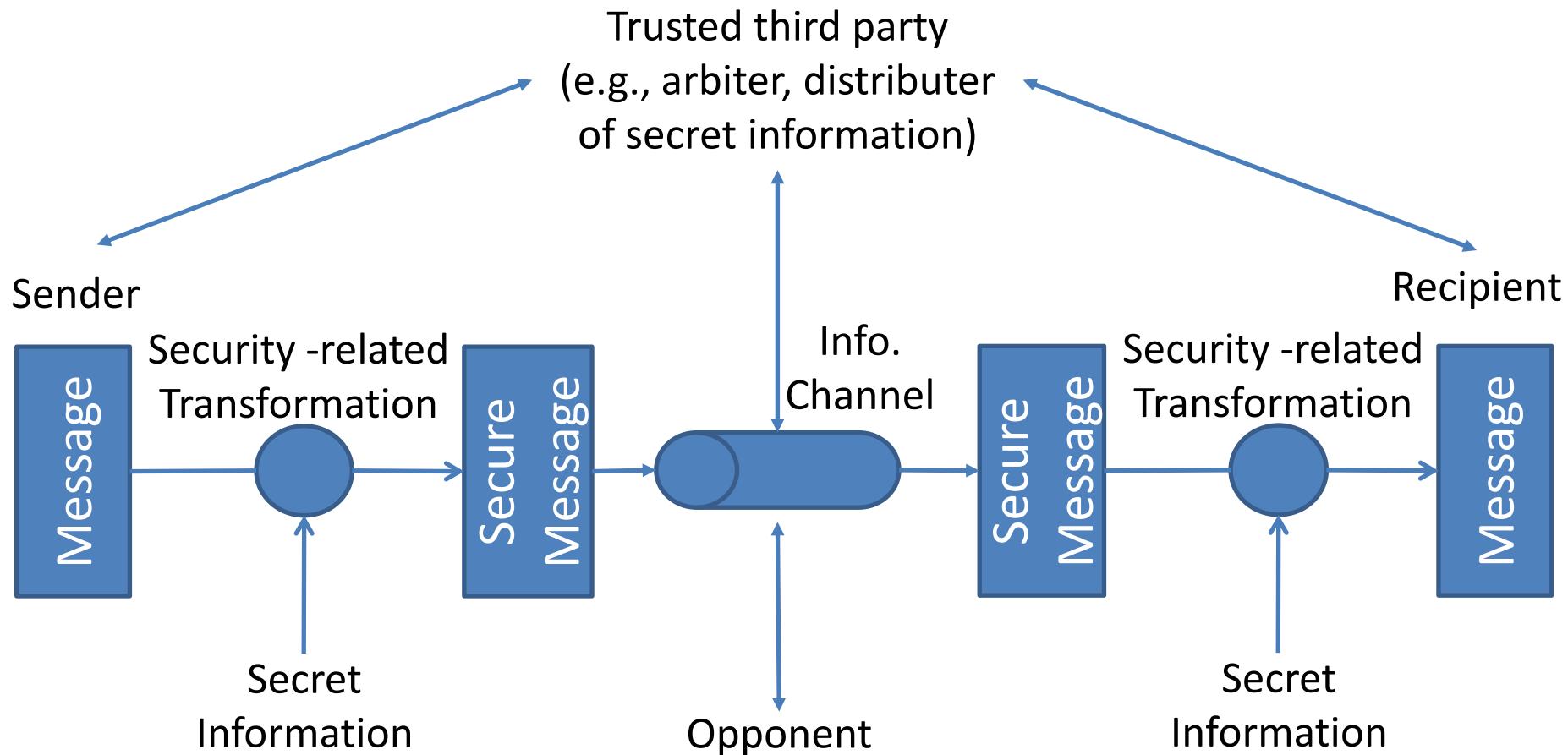
- **Encipherment:** Hiding or covering data using mathematical algorithms.
- **Digital Signature:** The sender can electronically sign the data and the receiver can electronically verify the signature.
- **Access Control:** A variety of mechanisms that enforce access rights to resources.
- **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- **Authentication Exchange:** Two entities exchange some messages to prove their identity to each other.

# Security Mechanism (Specific security)

---

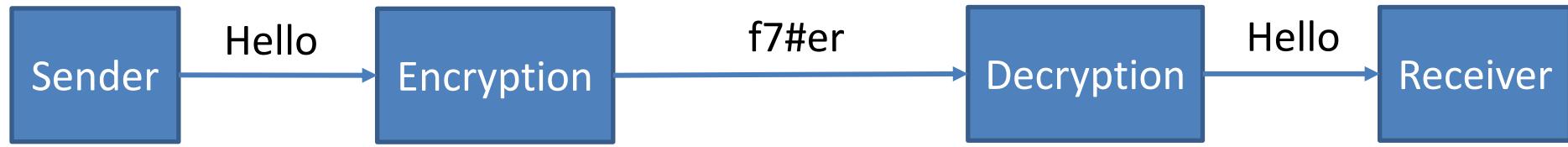
- **Traffic Padding:** The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
- **Routing Control:** Selecting and continuously changing routes between sender and receiver to prevent opponent from eavesdropping.
- **Notarization:** The use of a trusted third party to assure and control the communication.

# Model for Network Security

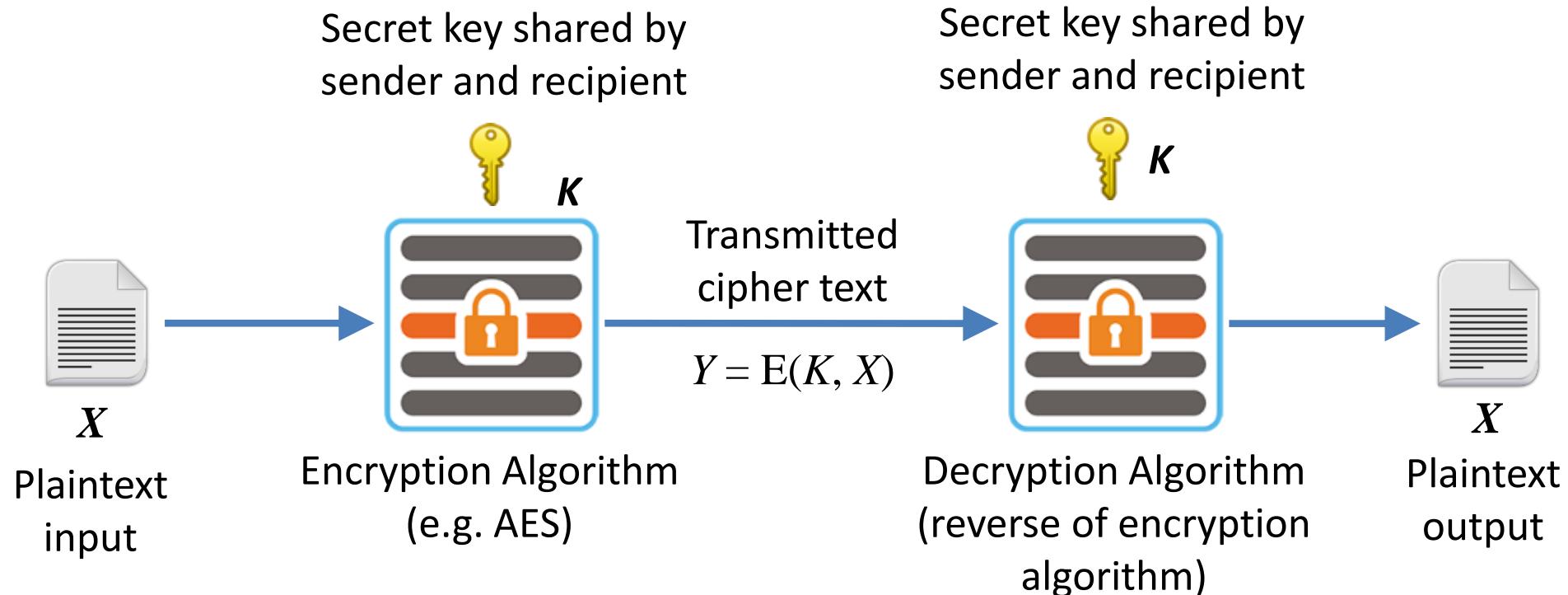


# Encryption and Decryption

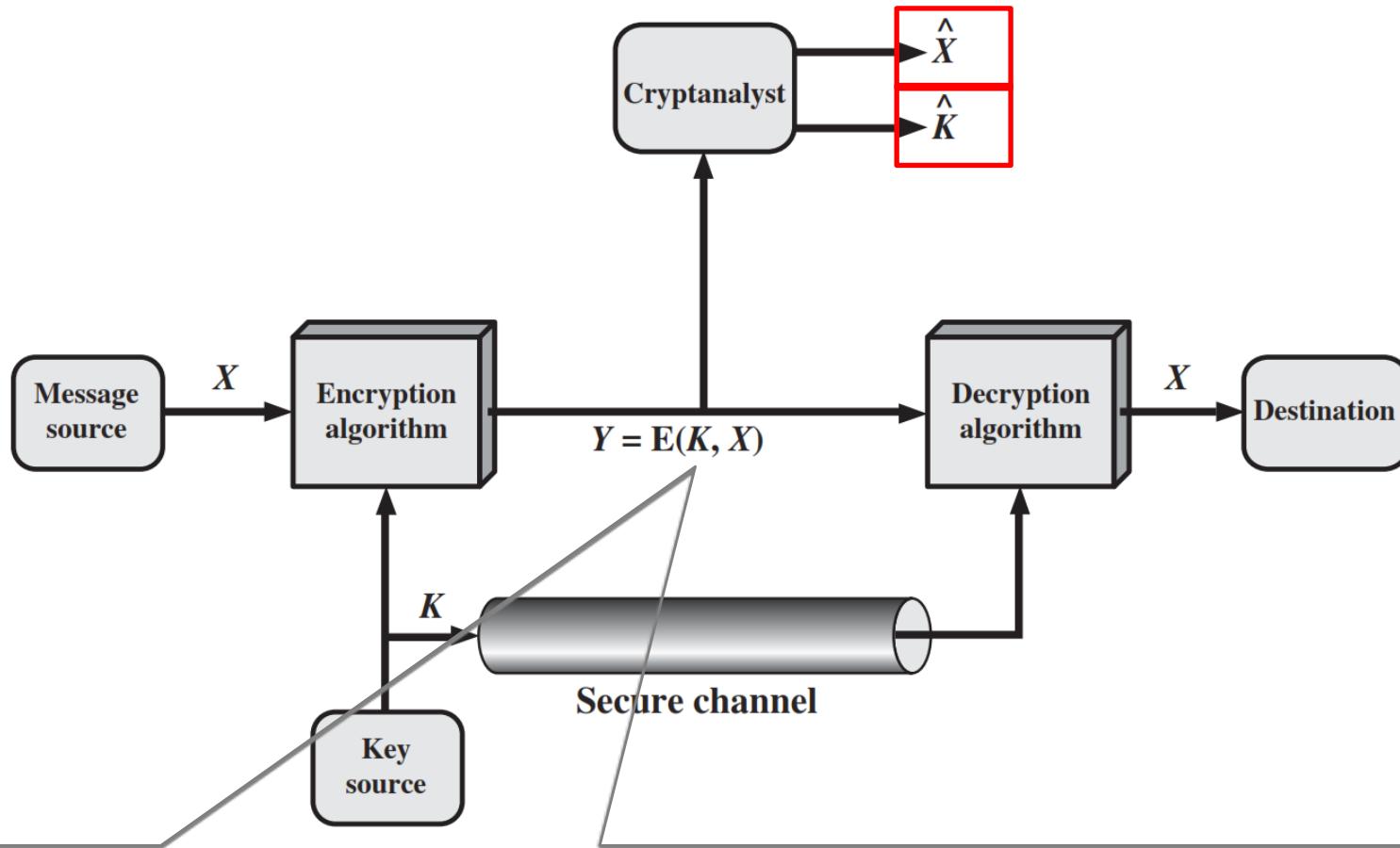
---



# Symmetric Cipher Model (Conventional Encryption)



- **Decryption algorithm implemented using the encryption algorithm with the received key.**
- The decryption algorithm depends on the secret key, plaintext and of the encryption algorithm.
- The ciphertext is decrypted using the same key that was used for the original encryption.
- Plain text is recovered by the client, extracting the plain text from the ciphertext key being used at the time of encryption.



- An opponent, observing  $Y$  but not having access to  $K$  or  $X$ , may attempt to recover  $X$  or  $K$  or both  $X$  and  $K$ .
- If the opponent is interested in only this particular message, then he will focus to recover  $X$  by generating a plaintext estimate  $\hat{X}$ .
- Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover  $K$  by generating an estimate  $\hat{K}$ .

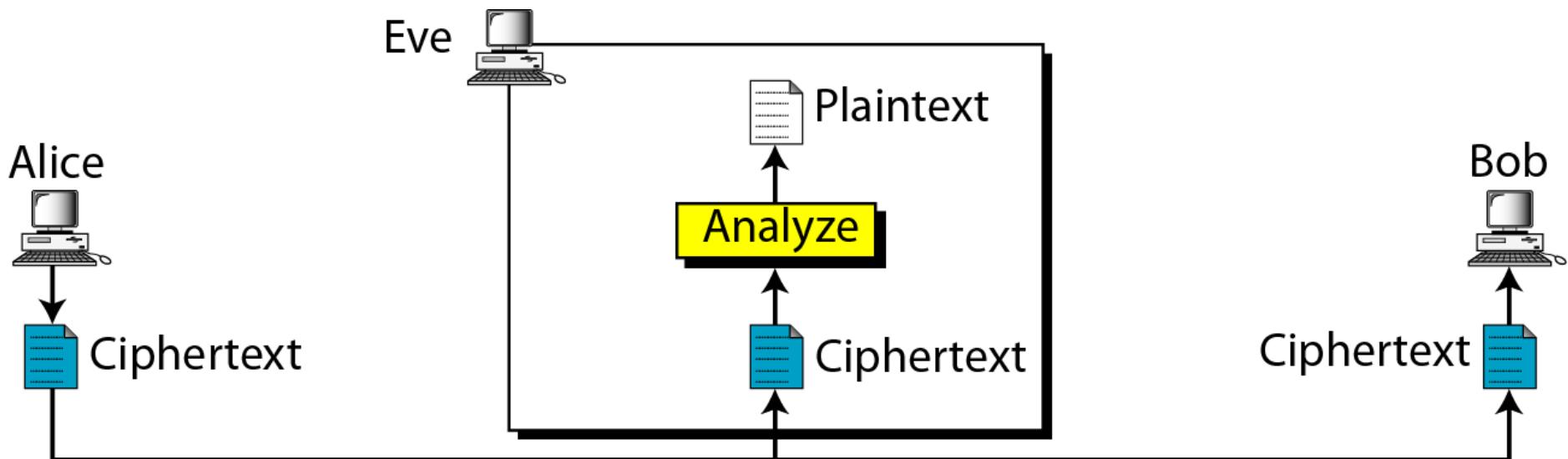
# Cryptanalysis and Brute-Force Attack

---

- **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm and 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 derive a specific plaintext or to derive 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.

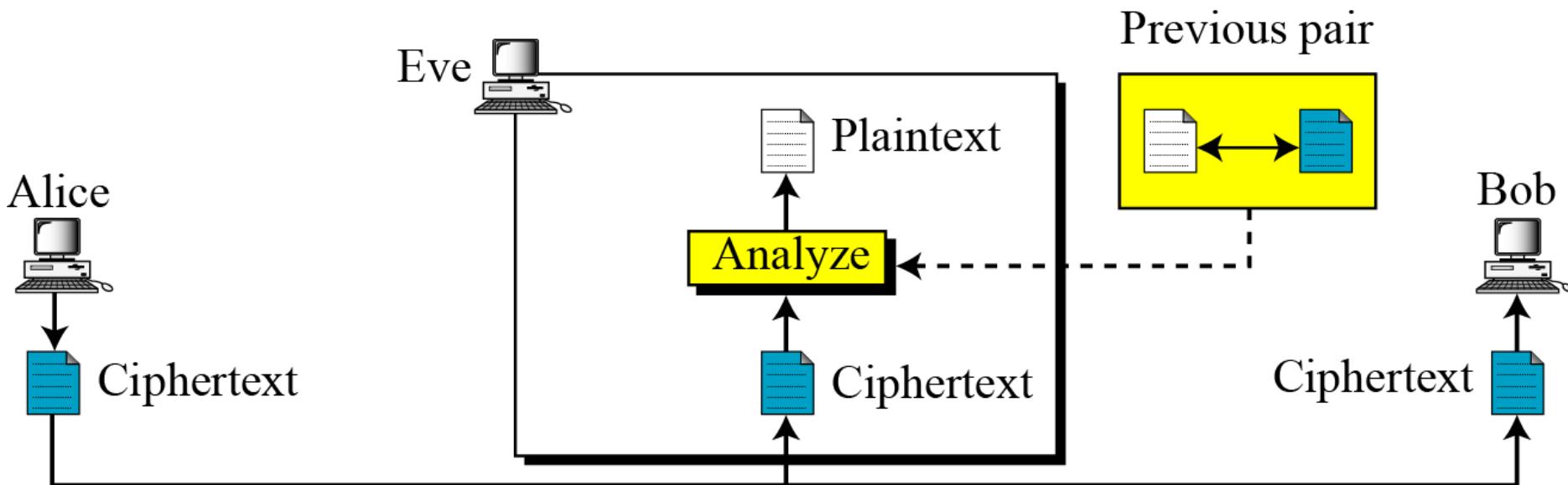
# Attacks on Encrypted Messages

Type of Attack	Known to cryptanalyst
Ciphertext Only	Encryption algorithm, Ciphertext



# Attacks on Encrypted Messages

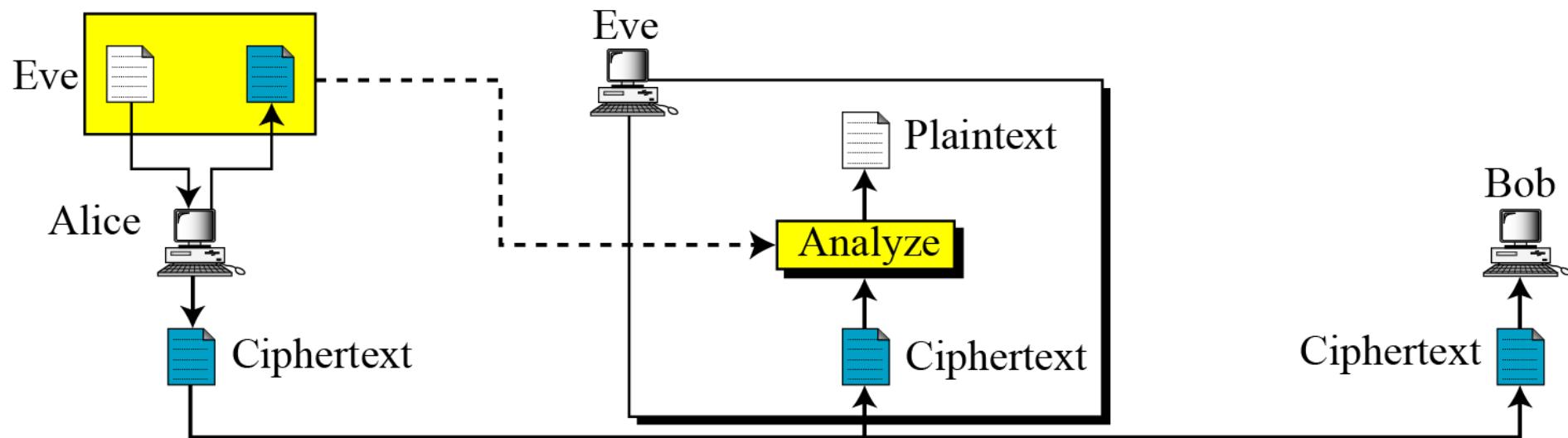
Type of Attack	Known to cryptanalyst
Known Plaintext	Encryption algorithm, Ciphertext, One or more plaintext-cipher text pairs formed with the secret key



# Attacks on Encrypted Messages

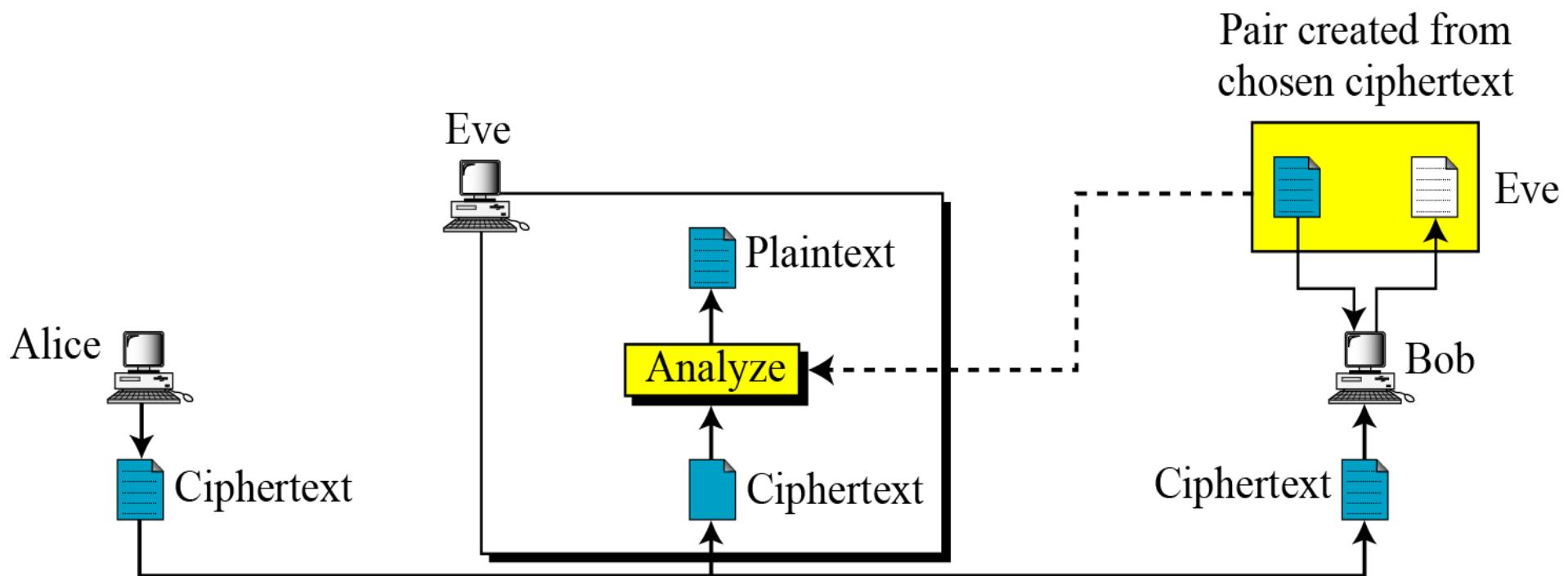
Type of Attack	Known to cryptanalyst
Chosen Plaintext	Encryption algorithm, Ciphertext, Plaintext message chosen by cryptanalyst

Pair created from chosen plaintext



# Attacks on Encrypted Messages

Type of Attack	Known to cryptanalyst
Chosen Ciphertext	Encryption algorithm, Ciphertext, Ciphertext chosen by cryptanalyst, with its corresponding decrypted plaintext generated with the secret key



# Attacks on Encrypted Messages

Type of Attack	Known to cryptanalyst
Chosen text	Encryption algorithm, Ciphertext, Plaintext chosen by cryptanalyst, with its corresponding ciphertext generated with the secret key , Ciphertext chosen by cryptanalyst, with its corresponding decrypted plaintext generated with the secret key

# Substitution Techniques

---

- A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.
  - 1) Caesar Cipher
  - 2) Monoalphabetic Cipher
  - 3) Playfair Cipher
  - 4) Hill Cipher
  - 5) Polyalphabetic Ciphers
  - 6) One-Time Pad

# 1) Caesar Cipher

---

- The **Caesar cipher** involves replacing each letter of the alphabet with the letter standing three places further down the alphabet.
- In encryption each plaintext letter  $P$ , substitute the ciphertext letter  $C$ :

$$C = E(k, P) = (P + k) \bmod 26$$

$$C = E(3, P) = (P + 3) \bmod 26$$

- For decryption algorithm is:

$$P = D(k, C) = (C - k) \bmod 26$$

# Caesar Cipher (Cont...)

- Let us assign a numerical equivalent to each letter

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

$$C = E(3, P) = (P + 3) \bmod 26$$

plain: 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  
cipher: 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

Example:

Plaintext: THE QUICK BROWN FOX

Ciphertext: WKH TXLFN EURZQ IRA

# Brute force attack on Caesar Cipher

---

- The encryption and decryption algorithms are known.
- There are only 25 keys to try.
- The language of the plaintext is known and easily recognizable.

# Brute force attack on Caesar Cipher

Ciphertext: ZNK WAOIQ HXUCT LUD

Key	Transformed text
1	YMJ VZNHP GWTBS KTC
2	XLI UYMGO FVSAR JSB
3	WKH TXLFN EURZQ IRA
4	VJG SWKEM DTQYP HQZ
5	UIF RVJDL CSPXOGPY
6	THE QUICK BROWN FOX
7	SGD PTHBJ AQNVM ENW
8	RFC OSGAI ZPMUL DMV
9	QEB NRFZH YOLTK CLU
10	PDA MQEYG XNKSJ BKT
11	OCZ LPDXF WMJRI AJS
12	NBY KOCWE VLIQH ZIR
13	MAX JNBVD UKHPG YHQ

Key	Transformed text
14	LZW IMAUC TJGOF XGP
15	KYV HLZTB SIFNE WFO
16	JXU GKYSR RHEMD VEN
17	IWT FJXRZ QGDLC UDM
18	HVS EIWQY PFCKB TCL
19	GUR DHVPX OEBJA SBK
20	FTQ CGUOW NDAIZ RAJ
21	ESP BFTNV MCZHY QZI
22	DRO AESMU LBYGX PYH
23	CQN ZDRLT KAXFW OXG
24	BPM YCQKS JZWEV NWF
25	AOL XBPJR IYVDU MVE

# Substitution Techniques

---

- 1) Caesar Cipher
- 2) Monoalphabetic Cipher**
- 3) Playfair Cipher
- 4) Hill Cipher
- 5) Polyalphabetic Ciphers
- 6) One-Time Pad

## 2) Monoalphabetic Cipher (Simple substitution)

---

- It is an improvement to the Caesar Cipher.
- Instead of shifting the alphabets by some number, this scheme uses some permutation of the letters in alphabet.
- The sender and the receiver decide on a randomly selected permutation of the letters of the alphabet.
- With 26 letters in alphabet, the possible permutations are  $26!$  which is equal to  $4 \times 10^{26}$ .

plain: 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

cipher: y n l k x b s h m i w d p j r o q v f e a u g t z c

# Attack on Monoalphabetic Cipher

- The relative frequencies of the letters in the ciphertext (in percentages) are

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

Ciphertext:

uzqsovuhxmopvgpozpevsg**ZWszopfpesxudbmetsxaizvuephzhmdzshzowsf**  
pappdtsvpqu**ZWymxuzuhsxepyepopdzszufpombZWpfupzhmdjudtmohmq**

- In our ciphertext, the most common digram is ZW, which appears three times. So equate Z with t, W with h and P with e.
- Now notice that the sequence ZWP appears in the ciphertext, and we can translate that sequence as “the.”

# Attack on Monoalphabetic Cipher (Cont...)

---

- If the cryptanalyst knows the nature of the plaintext, then the analyst can exploit the regularities of the language.
- The relative frequency of the letters can be determined and compared to a standard frequency distribution for English.
- If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match.

# Substitution Techniques

---

- 1) Caesar Cipher
- 2) Monoalphabetic Cipher
- 3) Playfair Cipher**
- 4) Hill Cipher
- 5) Polyalphabetic Ciphers
- 6) One-Time Pad

# 3) Playfair Cipher

---

- The Playfair algorithm is based on a  $5 \times 5$  matrix (**key**) of letters.
- The matrix is constructed by filling in the letters of the **keyword** (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. **The letters I and J count as one letter.**

Example:

Keyword= OCCURRENCE

Plaintext= TALL TREES


# Playfair Cipher - Encrypt Plaintext

---

- Playfair, treats digrams (two letters) in the plaintext as single units and translates these units into ciphertext digrams.
- Make Pairs of letters add filler letter “**X**” if same letter appears in a pair.

Plaintext= TALL TREES

Plaintext= TA LX LT RE ES

- If there is an odd number of letters, then add uncommon letter to complete digram, a X/Z may be added to the last letter.

# Playfair Cipher - Encrypt Plaintext

- Map each pair in key matrix

Plaintext= TA LX LT RE ES

Ciphertext= PF IZ TZ EO RT

O	C	U	R	E
N	A	B	D	F
G	H	I/J	K	L
M	P	Q	S	T
V	W	X	Y	Z

- If the letters are different in the same row, they are replaced with the letter in the next column of the previous row, and moving the row to the left side is fine if necessary. If the first letter of the pair should be encrypted first, using the table above, the letter pair **RE** would be mapped as **FD**. Using the table above, the letter pair **TA** would be encoded as **PF**.
- The last step is if the pair has the same letter, replace it with the letter in the next column of the previous row, and moving the row to the left side is fine if necessary.
- The last step is if the pair has the same letter, replace it with the letter in the next column of the previous row, and moving the row to the left side is fine if necessary.
- The last step is if the pair has the same letter, replace it with the letter in the next column of the previous row, and moving the row to the left side is fine if necessary.

# Playfair Cipher Examples

1. Key= “engineering”      Plaintext=“ test this process ”
2. Key= “keyword”          Plaintext=“ come to the window ”
3. Key= “moonmission”     Plaintext=“ greet ”

E N G I R A B C D F H K L M O P Q S T U V W X Y Z	Encrypted Message: pi tu pm gt ue lf gp xg	K E Y W O R D A B C F G H I L M N P Q S T U V X Z	Encrypted Message: lc nk zk vf yo gq ce bw
M O N I S A B C D E F G H K L P Q R T U V W X Y Z	Encrypted Message: hq cz du		

# Substitution Techniques

---

- 1) Caesar Cipher
- 2) Monoalphabetic Cipher
- 3) Playfair Cipher
- 4) Hill Cipher**
- 5) Polyalphabetic Ciphers
- 6) One-Time Pad

# 4) Hill Cipher

- Hill cipher is based on linear algebra
- Each letter is represented by numbers from 0 to 25 and calculations are done modulo 26.
- Encryption and decryption can be given by the following formula:

Encryption:  $C=PK \text{ mod } 26$

Decryption:  $P=CK^{-1} \text{ mod } 26$

$$(c_1 \ c_2 \ c_3) = (p_1 \ p_2 \ p_3) \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \text{ mod } 26$$

# Hill Cipher Encryption

- To encrypt a message using the Hill Cipher we must first turn our keyword and plaintext into a matrix (a  $2 \times 2$  matrix or a  $3 \times 3$  matrix, etc).

**Example: Key = “HILL”, Plaintext = “EXAM”**

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

$$\text{Key Matrix } \begin{pmatrix} H & I \\ L & L \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

$$\text{Plaintext } \begin{pmatrix} E \\ X \end{pmatrix} \begin{pmatrix} A \\ M \end{pmatrix} = \begin{pmatrix} 4 \\ 23 \end{pmatrix} \begin{pmatrix} 0 \\ 12 \end{pmatrix}$$

# Hill Cipher Encryption (Cont...)

$$\text{Key Matrix} = \begin{pmatrix} H & I \\ L & L \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

$$\text{Plaintext} \begin{pmatrix} E \\ X \end{pmatrix} \begin{pmatrix} A \\ M \end{pmatrix} = \begin{pmatrix} 4 \\ 23 \end{pmatrix} \begin{pmatrix} 0 \\ 12 \end{pmatrix}$$

$$C = PK \bmod 26$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 4 \\ 23 \end{pmatrix}$$

$$7 \times 4 + 8 \times 23 = 212$$

$$11 \times 4 + 11 \times 23 = 297$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 4 \\ 23 \end{pmatrix} = \begin{pmatrix} 212 \\ 297 \end{pmatrix}$$

$$\begin{pmatrix} 212 \\ 297 \end{pmatrix} = \begin{pmatrix} 4 \\ 11 \end{pmatrix} \bmod 26 = \begin{pmatrix} E \\ L \end{pmatrix}$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 0 \\ 12 \end{pmatrix}$$

$$7 \times 0 + 8 \times 12 = 96$$

$$11 \times 0 + 11 \times 12 = 132$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 0 \\ 12 \end{pmatrix} = \begin{pmatrix} 96 \\ 132 \end{pmatrix}$$

$$\begin{pmatrix} 96 \\ 132 \end{pmatrix} = \begin{pmatrix} 18 \\ 2 \end{pmatrix} \bmod 26 = \begin{pmatrix} S \\ C \end{pmatrix}$$

Ciphertext = "ELSC"

# Hill Cipher Decryption

---

$$P = CK^{-1} \bmod 26$$

Step:1 Find Inverse of key matrix

Step:2 Multiply the Multiplicative Inverse of the Determinant by the Adjoin Matrix

Step:3 Multiply inverse key matrix with ciphertext matrix to obtain plaintext matrix

# Step: 1 Inverse of key matrix

---

2 X 2 inverse of matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - cb} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

3 X 3 inverse of matrix

$$A^{-1} = \frac{1}{\text{determinant}(A)} \cdot \text{adjoint}(A)$$

# Step: 1 Inverse of key matrix

$$\text{Inverse Key Matrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}^{-1} = \frac{1}{77 - 88} \begin{pmatrix} 11 & -8 \\ -11 & 7 \end{pmatrix}$$

$$= \frac{1}{-11} \begin{pmatrix} 11 & -8 \\ -11 & 7 \end{pmatrix}$$

$$= \frac{1}{15} \begin{pmatrix} 11 & 18 \\ 15 & 7 \end{pmatrix} \text{ mod } 26$$

- $-11 \text{ mod } 26 = 15$
- Because, modulo for negative number is  $= N - (B \% N)$   
 $= 26 - (11 \% 26)$

# Step: 2 Modular (Multiplicative) inverse

---

- The inverse of a number A is  $1/A$  since  $A * 1/A = 1$   
e.g. the inverse of 5 is  $1/5$
- In modular arithmetic we do not have a division operation.
- The modular inverse of  $A \pmod{C}$  is  $A^{-1}$
- $(A * A^{-1}) \equiv 1 \pmod{C}$

Example:

- The modular inverse of  $A \pmod{C}$  is the  $A^{-1}$  value that makes  
 $A * A^{-1} \pmod{C} = 1$

$$A = 3, C = 11$$

Since  $(3 * 4) \pmod{11} = 1$ , 4 is modulo inverse of 3

$$A = 10, C = 17, A^{-1} = 12$$

# Step 2: Modular (Multiplicative) inverse

Determinants' multiplicative inverse Modulo 26

Determinant	1	3	5	7	9	11	15	17	19	21	23	25
Inverse Modulo 26	1	9	21	15	3	19	7	23	11	5	17	25

$$= \frac{1}{15} \begin{pmatrix} 11 & 18 \\ 15 & 7 \end{pmatrix} \text{ mod } 26$$

- Multiplicative inverse of  $\frac{1}{15}$  is 7

## Step 2: Multiply with adjoint of matrix

---

$$= 7 \begin{pmatrix} 11 & 18 \\ 15 & 7 \end{pmatrix} = \begin{pmatrix} 77 & 126 \\ 105 & 49 \end{pmatrix} = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \text{ mod } 26$$

$$= \text{thus, if } K = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \text{ then } K^{-1} = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix}$$

# Hill Cipher Decryption

Inverse Key Matrix =  $\begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix}$       Ciphertext  $\begin{pmatrix} E \\ L \\ C \end{pmatrix} = \begin{pmatrix} 4 \\ 11 \\ 2 \end{pmatrix}$

$$P = CK^{-1} \bmod 26$$

$$\begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} 4 \\ 11 \end{pmatrix}$$

$$25 \times 4 + 22 \times 11 = 342$$

$$1 \times 4 + 23 \times 11 = 257$$

$$\begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} 4 \\ 11 \end{pmatrix} = \begin{pmatrix} 342 \\ 257 \end{pmatrix}$$

$$\begin{pmatrix} 342 \\ 257 \end{pmatrix} = \begin{pmatrix} 4 \\ 23 \end{pmatrix} \bmod 26 = \begin{pmatrix} E \\ X \end{pmatrix}$$

$$\begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} 18 \\ 2 \end{pmatrix}$$

$$25 \times 18 + 22 \times 2 = 494$$

$$1 \times 18 + 23 \times 2 = 64$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 0 \\ 12 \end{pmatrix} = \begin{pmatrix} 494 \\ 64 \end{pmatrix}$$

$$\begin{pmatrix} 494 \\ 64 \end{pmatrix} = \begin{pmatrix} 0 \\ 12 \end{pmatrix} \bmod 26 = \begin{pmatrix} A \\ M \end{pmatrix}$$

Plaintext = "EXAM"

# Substitution Techniques

---

- 1) Caesar Cipher
- 2) Monoalphabetic Cipher
- 3) Playfair Cipher
- 4) Hill Cipher
- 5) **Polyalphabetic Ciphers**
- 6) One-Time Pad

# 5) Polyalphabetic Cipher

---

- Monoalphabetic cipher encoded using only one fixed alphabet
- **Polyalphabetic cipher** is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process.
  1. **Vigenere cipher**
  2. **Vernam cipher**

Plaintext

K  
e  
y

PT = **HELLO**  
KEY = **GM**  
CT = **NQRXU**

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	
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	
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	
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	
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	
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	
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

# Vigenere Cipher

Keyword : **DECEPTIVE**

Key : **DECEPTIVE**

Plaintext : **WEAREDISCOVEREDSAVEYOURSELF**

Ciphertext : **ZICVTWQNNGRZGVVTWAVZHCQYGLMGJ**

$$C = (P_1 + K_1, P_2 + K_2, \dots, P_m + K_m) \bmod 26$$

$$P = (C_1 - K_1, C_2 - K_2, \dots, C_m - K_m) \bmod 26$$

An analyst looking at only the ciphertext would detect the repeated sequences VTW at a displacement of 9 and make the assumption that the keyword is either three or nine letters in length.

Keyword : **DECEPTIVE**

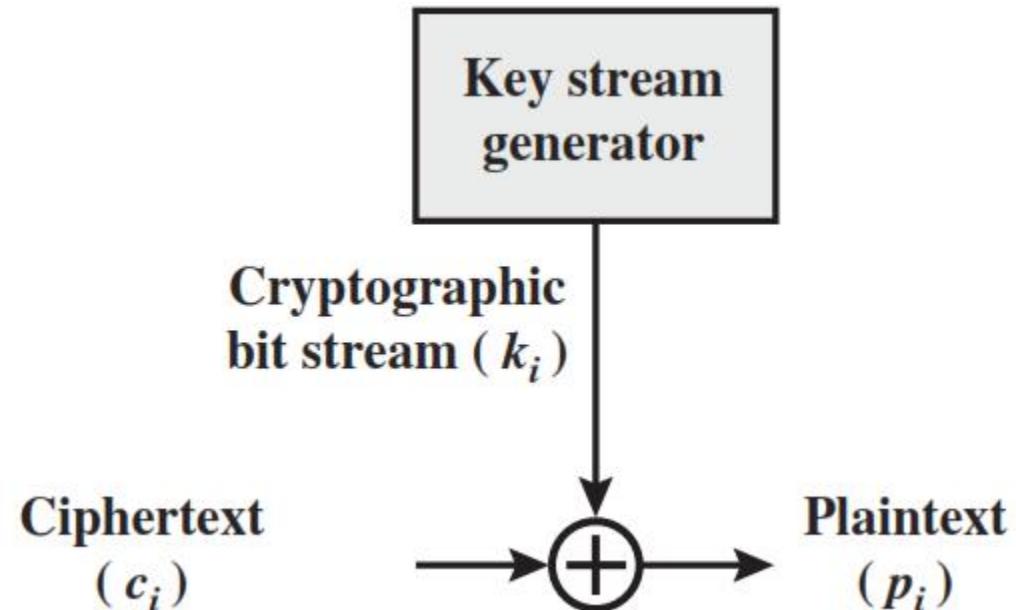
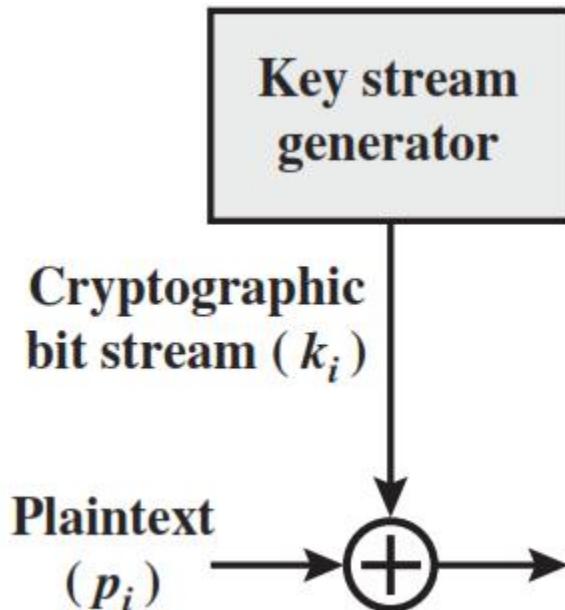
Key : **DECEPTIVE**

Plaintext : **WEAREDISCOVEREDSAVEYOURSELF**

This system  
is referred as  
an **autokey**  
**system**

# Vernam Cipher

- The ciphertext is generated by applying the logical XOR operation to the individual bits of plaintext and the key stream.



# Substitution Techniques

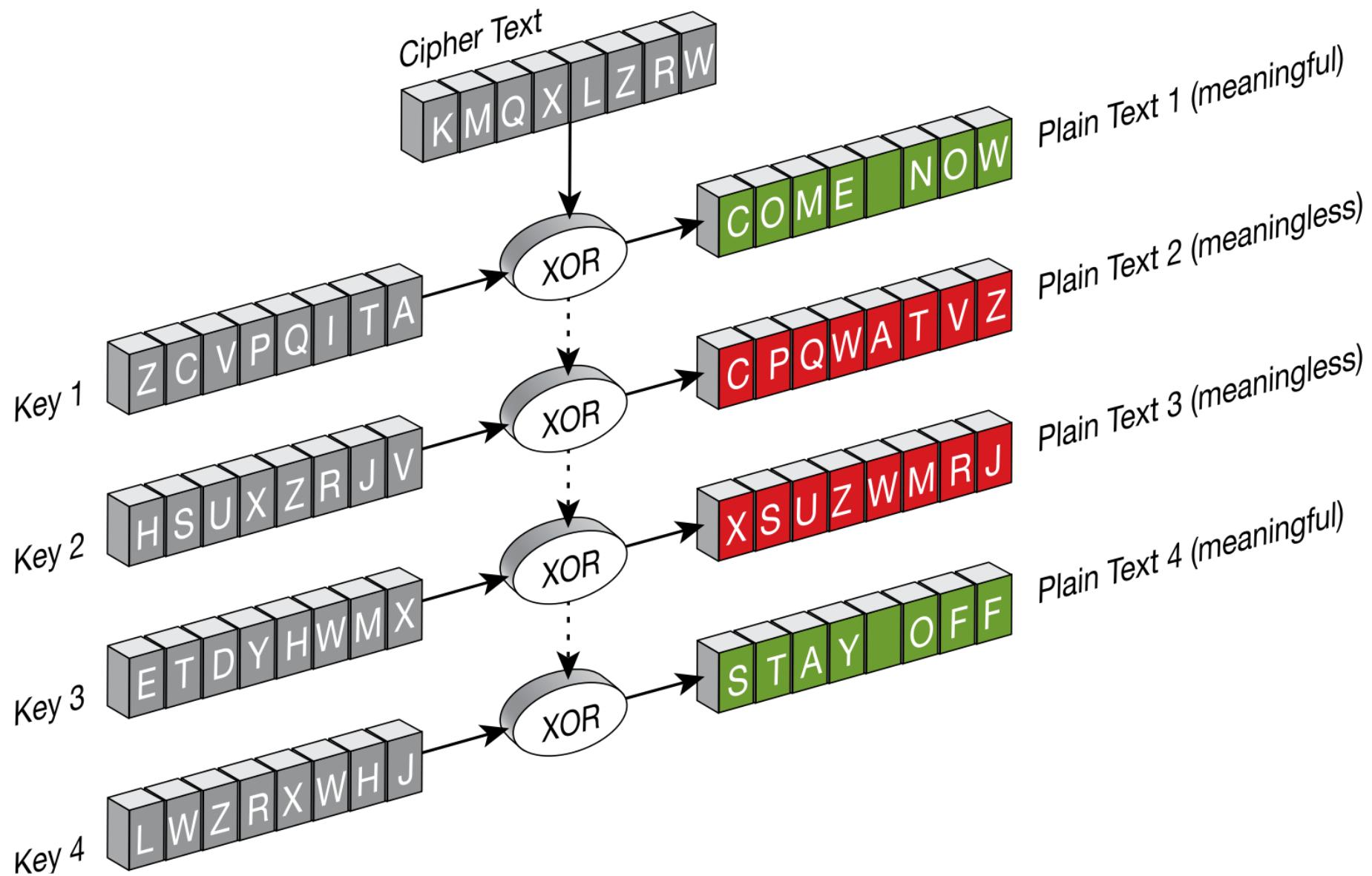
---

- 1) Caesar Cipher
- 2) Monoalphabetic Cipher
- 3) Playfair Cipher
- 4) Hill Cipher
- 5) Polyalphabetic Ciphers
- 6) **One-Time Pad**

# One time pad

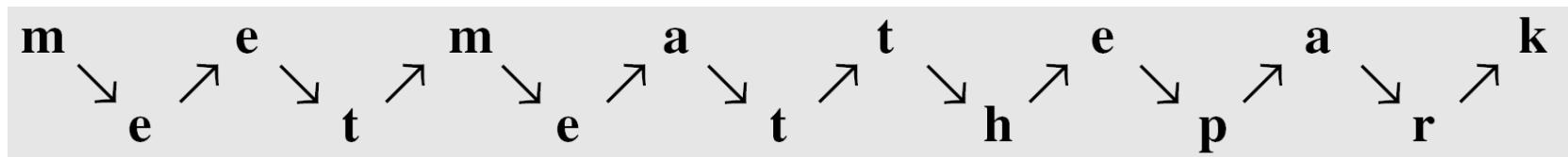
---

- The one-time pad, which is a provably secure cryptosystem, was developed by Gilbert Vernam in 1918.
- The message is represented as a binary string (a sequence of 0's and 1's using a coding mechanism such as ASCII coding).
- **The key is a truly random sequence of 0's and 1's of the same length as the message.**
- message ='IF'
- then its ASCII code =(1001001 1000110)
- key = (1010110 0110001)
- *Encryption:*
  - 1001001 1000110      plaintext
  - 1010110 0110001      key
  - 0011111 1110110      ciphertext



# Transposition Techniques

- A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols.
- The simplest such cipher is the **rail fence technique**, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.
- For example, to send the message “**Meet me at the park**” to Bob, Alice writes



- She then creates the ciphertext “**MEMATEAKETETHPR**”.

# ...Transposition Techniques

- A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns.
- Transposition (Columnar): The order of the columns then becomes the key to the algorithm.

**Key:**            4    3    1    2    5    6    7

**Plaintext:** 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

**Ciphertext:** TTNAAPMTSUOAODWCOIXKNLYPETZ

# Cryptography and Cryptanalysis

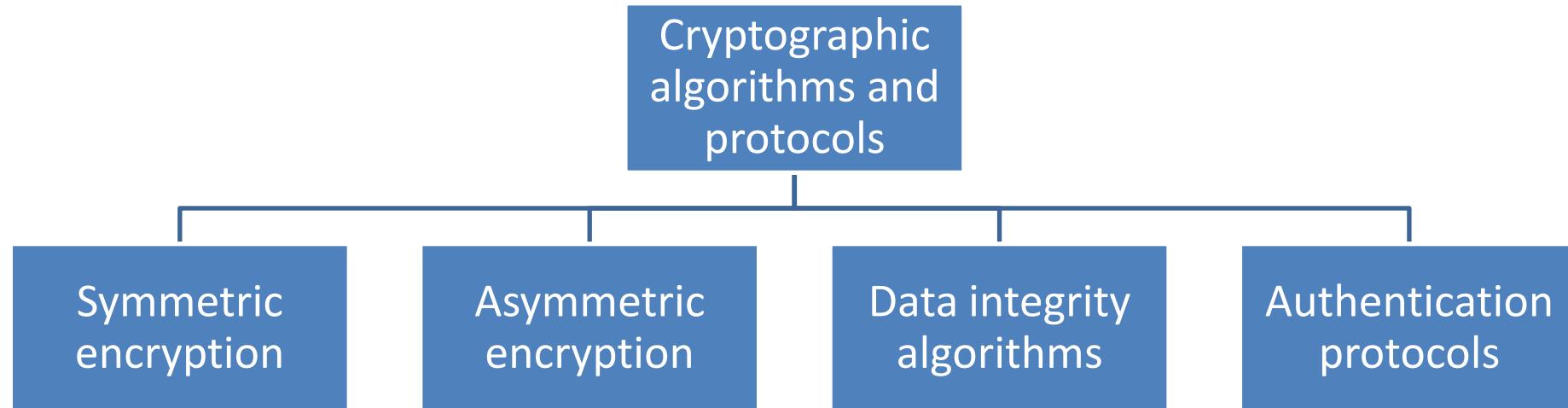
---

- **Cryptography and Cryptanalysis**

- **Cryptography** is the study of the design of techniques for ensuring the secrecy and/or authenticity of information
- **Cryptanalysis** deals with the defeating such techniques to recover information, or forging information that will be accepted as authentic

# Cryptographic Algorithms

- Cryptographic algorithms and protocols can be grouped into four main areas



- **Authentication protocols** is used to prove the identity of users or systems, often by letting them sign a message with their digital signature or password.

# Steganography

---

- an alternative to encryption
- hides existence of message
  - using only a subset of letters/words in a longer message marked in some way
  - using invisible ink
  - hiding in LSB in graphic image or sound file
- has drawbacks
  - high overhead to hide relatively few info bits
- advantage is can obscure encryption use

3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told, by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16+ proposals destroy your basic O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours.

# Demo

---

- <https://stylesuxx.github.io/steganography/>



**THANK  
YOU FOR  
LISTENING  
ANY  
QUESTION ?**

# UNIT-1

# Number Theory



# Prime Numbers

---

- Prime numbers are central to number theory
- Prime numbers only have divisors of 1 and self
  - They cannot be written as a product of other numbers
  - e.g. 2,3,5,7 are prime, 4,6,8,9,10 are not

# Relatively Prime Numbers

---

- Determining the GCD of two positive integers is easy
  - if they are expressed each - as the product of primes
  - e.g. if  $300 = 2^2 \times 3^1 \times 5^2$  and  $18 = 2^1 \times 3^2$  then the  $\text{gcd}(300, 18)$  is given as  $2^1 \times 3^1 \times 5^0 = 6$
- Two numbers  $a$  and  $b$  are relatively prime if they have no common divisors apart from 1
  - e.g. 8 & 15 are relatively prime (even though none of them is prime) since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
  - e.g. check whether 6 and 8 are relative prime or not?
  - e.g. check whether 14 and 9 are relative prime or not?

# Euclidean Algorithm

---

- Efficient way to find the  $\text{GCD}(a,b)$
- Euclidean Algorithm to compute  $\text{GCD}(a,b)$  is:

EUCLID (a, b)

1. A = a; B = b
2. if B = 0 return A (= gcd(a, b) )
3. R = A mod B
4. A = B
5. B = R
6. goto 2

- e.g. check whether 6 and 8 are relative prime or not?
- e.g. check whether 14 and 9 are relative prime or not?

**Applications: Key Generation, Extended Euclidean Algorithm, Diffie-Hellman Key Exchange, Elliptic Curve Cryptography and more**

# Example GCD(1970,1066)

---

$$\begin{array}{lcl} 1970 = 1 \times 1066 + 904 & \text{gcd}(1066, 904) \\ 1066 = 1 \times 904 + 162 & \text{gcd}(904, 162) \\ 904 = 5 \times 162 + 94 & \text{gcd}(162, 94) \\ 162 = 1 \times 94 + 68 & \text{gcd}(94, 68) \\ 94 = 1 \times 68 + 26 & \text{gcd}(68, 26) \\ 68 = 2 \times 26 + 16 & \text{gcd}(26, 16) \\ 26 = 1 \times 16 + 10 & \text{gcd}(16, 10) \\ 16 = 1 \times 10 + 6 & \text{gcd}(10, 6) \\ 10 = 1 \times 6 + 4 & \text{gcd}(6, 4) \\ 6 = 1 \times 4 + 2 & \text{gcd}(4, 2) \\ 4 = 2 \times 2 + 0 & \text{gcd}(2, 0) \end{array}$$

Therefore,  $\text{gcd}(1970, 1066) = 2$       GCD(270,192) ??

# Modular Arithmetic

---

- Two integers "a" and "b" are said to be congruent modulo "n" if their difference "a - b" is divisible by "n."
- This is denoted as " $a \equiv b \pmod{n}$ ." In other words, "a" and "b" leave the same remainder when divided by "n."  
e.g.  $100 \equiv 34 \pmod{11}$  (where 100 and 34 leave same remainder when divided by 11)
- Modular addition rule
  - e.g.  $(a+b) \pmod{n} = (a \pmod{n} + b \pmod{n}) \pmod{n}$
- Process of **modulo reduction:**
  - E.g.  $-12 \pmod{7} \equiv -5 \pmod{7} \equiv 2 \pmod{7}$
  - E.g.  $17 \pmod{5} \equiv 2 \pmod{5}$

# Abstract Algebra

---

- Number theory has increasing importance in cryptography
  - AES, Elliptic Curve, IDEA, Public Key
- Abstract algebra is a branch of mathematics that deals with algebraic structures, such as groups, rings, and fields, in a more abstract and general way than elementary algebra. It is called "abstract" because it focuses on the study of algebraic systems and their properties without necessarily specifying the nature of the elements involved.

# Group

---

- A set of elements or “numbers”
- With some operation whose result is also in the set
- Obeys:
  - Closure: For any two elements "a" and "b" in the group, the result of the operation " $a * b$ " is also in the group.
  - Associative law:  $(a.b).c = a.(b.c)$
  - Has identity  $e$ :  $e.a = a.e = a$
  - Has inverses  $a^{-1}$ :  $a.a^{-1} = e$
- E.g. the set of integers "Z" under addition "+" forms a group.
- If commutative  $a.b = b.a$ 
  - Then forms an **Abelian Group**

# Cyclic Group

---

- A group is cyclic if every element is a power of some fixed element
- It is a group that can be generated by a single element, which is often referred to as the generator of the group.
- In a cyclic group, repeated applications of the group operation to the generator produce all the elements of the group.
- E.g. Cyclic Group of Integers Modulo 5: The set of integers modulo 5 is  $\{0, 1, 2, 3, 4\}$ , and the group operation is addition modulo 5

# ...Cyclic Group

---

Here are the key properties of this cyclic group:

- **Generator:** The integer 1 is a generator for this group. This means that by repeatedly adding 1 (modulo 5), we can generate all the elements of the group:
  - $1 + 1 \equiv 2 \pmod{5}$
  - $2 + 1 \equiv 3 \pmod{5}$
  - $3 + 1 \equiv 4 \pmod{5}$
  - $4 + 1 \equiv 0 \pmod{5}$
  - $0 + 1 \equiv 1 \pmod{5}$
- **Closure:** When we add two integers modulo 5, the result remains within the set  $\{0, 1, 2, 3, 4\}$ , so the group is closed under addition modulo 5.
- **Associativity:** Addition modulo 5 is associative, as the order in which we add elements doesn't affect the result.
- **Identity Element:** The identity element in this group is 0 because adding 0 to any element doesn't change it.
- **Inverse Element:** Each element has an inverse within the group:
  - The inverse of 1 is 4 because  $1 + 4 \equiv 0 \pmod{5}$ .
  - The inverse of 2 is 3 because  $2 + 3 \equiv 0 \pmod{5}$ .
  - The inverse of 3 is 2 because  $3 + 2 \equiv 0 \pmod{5}$ .
  - The inverse of 4 is 1 because  $4 + 1 \equiv 0 \pmod{5}$ .
  - The inverse of 0 is itself ( $0 + 0 \equiv 0 \pmod{5}$ )).

# Ring

---

- A set of “numbers”
- With two binary operations (addition and multiplication) which form
  - An Abelian Group with addition operation
- And multiplication:
  - Has closure
  - Is associative
  - Distributive over addition:  $a(b+c) = ab + ac$
- If multiplication operation is commutative, it forms a **commutative ring**

# Field

---

- A set of numbers with two operations which form:
  - Abelian Group for addition
  - Abelian Group for multiplication (ignoring 0)
  - Ring
- E.g. The field of real numbers (denoted as "R") with ordinary addition and multiplication.
- Hierarchy
  - Group -> Ring -> Field

# Summary: Group, Ring and Field

---

- A group focuses on a single binary operation and is characterized by closure, associativity, identity, and inverses.
- A ring adds the concept of two operations (addition and multiplication) and maintains closure, associativity, and additional properties specific to rings. **It may or may not have a multiplicative inverses for all elements.**
- A field builds upon the ring structure but adds multiplicative inverses and a multiplicative identity, making it the most versatile and fundamental of the three structures.
- Fields are particularly important because they encompass both addition and multiplication, and they play a crucial role in various mathematical and scientific applications, including number theory, linear algebra, and cryptography.

# Galois Fields

---

- Galois Fields (Finite fields) play a key role in cryptography
- It contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.
- The number of elements of a finite field is called its order or, sometimes, its size.
- A Galois field consists of a finite number of elements, denoted as "GF( $p$ )," where " $p$ " is a prime number (forms a Prime Field) or a power of a prime number (forms an Extension Field).
- Extension fields are constructed by defining irreducible polynomials over the prime field
- An irreducible polynomial is a polynomial that cannot be factored into two lower-degree polynomials with coefficients in the base field

# Galois Fields GF(p)

---

- In particular often use the fields
  - GF(p)
  - GF( $2^n$ )
- GF(p) is the set of integers  $Z_p = \{0, 1, \dots, p-1\}$  with arithmetic operations modulo prime p
  - E.g. GF(7)= $\{0, 1, 2, 3, 4, 5, 6\}$
- The arithmetic in GF is “well-behaved”
  - i.e. can do addition, subtraction, multiplication, and division without leaving the field GF(p)

# GF(7) Addition/Multiplication Example

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

(a) Addition modulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

w	-w	$w^{-1}$
0	0	—
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) Additive and multiplicative inverses modulo 7

# Polynomial Arithmetic

---

- can compute using polynomials

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = \sum a_i x^i$$

- nb. not interested in any specific value of  $x$
- which is known as the indeterminate

- several alternatives available

- ordinary polynomial arithmetic
- poly arithmetic with coords mod  $p$
- poly arithmetic with coords mod  $p$  and polynomials mod  $m(x)$

# Ordinary Polynomial Arithmetic

---

- add or subtract corresponding coefficients
- multiply all terms by each other
- eg

$$\text{let } f(x) = x^3 + x^2 + 2 \text{ and } g(x) = x^2 - x + 1$$

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$

$$f(x) - g(x) = x^3 + x + 1$$

$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

# Polynomial Arithmetic with Modulo Coefficients

---

- We are most interested in mod 2
  - i.e. all coefficients are 0 or 1
  - eg. let  $f(x) = x^3 + x^2$  and  $g(x) = x^2 + x + 1$

$$f(x) + g(x) = x^3 + x + 1$$

$$f(x) \times g(x) = x^5 + x^2$$

# Polynomial Division

---

- If  $g(x)$  has no divisors other than itself & 1 say it is **irreducible** (or prime) polynomial
- Arithmetic modulo an irreducible polynomial forms a field (As discussed earlier)
  - $x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \pmod{x^8+x^4+x^3+x+1}$   
(11B)
  - $x^7+x^6+1=C_1$

# Modular Polynomial Arithmetic

---

- can compute in field  $GF(2^n)$ 
  - polynomials with coefficients modulo 2
  - whose degree is less than n
  - hence must reduce modulo an irreducible poly of degree n (for multiplication only)
- form a finite field
- can always find an inverse
  - can extend Euclid's Inverse algorithm to find

# Example GF( $2^3$ )

Table 4.6 Polynomial Arithmetic Modulo ( $x^3 + x + 1$ )

		000	001	010	011	100	101	110	111
		0	1	$x$	$x + 1$	$x^2$	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
+	000	0	1	$x$	$x + 1$	$x^2$	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
	001	1	0	$x + 1$	$x$	$x^2 + 1$	$x^2$	$x^2 + x + 1$	$x^2 + x$
	010	$x$	$x + 1$	0	1	$x^2 + x$	$x^2 + x + 1$	$x^2$	$x^2 + 1$
	011	$x + 1$	$x$	1	0	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	$x^2$
	100	$x^2$	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$	0	1	$x$	$x + 1$
	101	$x^2 + 1$	$x^2$	$x^2 + x + 1$	$x^2 + x$	1	0	$x + 1$	$x$
	110	$x^2 + x$	$x^2 + x + 1$	$x^2$	$x^2 + 1$	$x$	$x + 1$	0	1
	111	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	$x^2$	$x + 1$	$x$	1	0

(a) Addition

		000	001	010	011	100	101	110	111
		0	1	$x$	$x + 1$	$x^2$	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
x	000	0	0	0	0	0	0	0	0
	001	0	1	$x$	$x + 1$	$x^2$	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
	010	$x$	$x$	$x^2$	$x^2 + x$	$x + 1$	1	$x^2 + x + 1$	$x^2 + 1$
	011	0	$x + 1$	$x^2 + x$	$x^2 + 1$	$x^2 + x + 1$	$x^2$	1	$x$
	100	$x^2$	$x^2$	$x + 1$	$x^2 + x + 1$	$x^2 + x$	$x$	$x^2 + 1$	1
	101	$x^2 + 1$	$x^2 + 1$	1	$x^2$	$x$	$x^2 + x + 1$	$x + 1$	$x^2 + x$
	110	$x^2 + x$	$x^2 + x + 1$	1	$x^2 + 1$	$x + 1$	$x$	$x^2$	$x$
	111	$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + 1$	$x$	1	$x^2 + x$	$x^2$	$x + 1$

(b) Multiplication

# Computational Example

---

- In  $\text{GF}(2^3)$  have  $(x^2+1)$  is  $101_2$  &  $(x^2+x+1)$  is  $111_2$
- So addition is
  - $(x^2+1) + (x^2+x+1) = x$
  - $101 \text{ XOR } 111 = 010_2$
- And multiplication is
  - $(x+1) \cdot (x^2+1) = x \cdot (x^2+1) + 1 \cdot (x^2+1)$   
 $= x^3+x+x^2+1 = x^3+x^2+x+1$  (which is 1111)
- Polynomial modulo reduction (get  $q(x)$  &  $r(x)$ ) is
  - $(x^3+x^2+x+1) \bmod (x^3+x+1) = 1 \cdot (x^3+x+1) + (x^2) = x^2$   
(which is 0100)

# Euler Totient Function $\phi(n)$

---

- When doing arithmetic modulo n
- **complete set of residues** is:  $0 \dots n-1$
- **Reduced set of residues** is those numbers (residues) which are relatively prime to n
  - e.g. for  $n=10$ ,
  - Complete set of residues is  $\{0,1,2,3,4,5,6,7,8,9\}$
  - Reduced set of residues is  $Z_n^* = Z_{10}^* = \{1,3,7,9\}$
- **Number of elements** in reduced set of residues is called the **Euler Totient Function  $\phi(n)$**
- $\phi(1) = 1$  since for  $n = 1$  the only integer in the range from 1 to n is 1 itself, and  $\gcd(1, 1) = 1$

# Euler Totient Function $\phi(n)$ (Cont.)

- To compute  $\phi(n)$  need to count number of residues to be excluded
- In general need prime factorization, but
  - For  $p$  ( $p$  prime)  $\phi(p) = p-1$
  - For  $p^*q$  ( $p,q$  prime)  $\phi(p^*q) = (p-1) \times (q-1)$
- Multiplicative group  $Z_n$  is denoted by  $Z_n^*$
- Examples

$$|Z_{10}^*| = \phi(10) = \phi(2)*\phi(5) = (2-1)*(5-1) = 4$$

$$|Z_{37}^*| = \phi(37) = 36$$

$$|Z_{21}^*| = \phi(21) = \phi(3)*\phi(7) = (3-1)*(7-1) = 2*6 = 12$$

$$|Z_{27}^*| = \phi(27) = \phi(3^3) = 3^2 * \phi(3) = 9 * 2 = 18$$

(for a prime  $p$ ,  $\phi(p^n) = p^n - p^{n-1}$ )

# Euler's Theorem

---

- Also known as the **Fermat–Euler Theorem** or **Euler's Totient Theorem**
- States that if  $n$  is a positive integer and  $a$  is a positive integer relatively prime to  $n$  (i.e.  $\gcd(a,n)=1$ ), then  $a^{\phi(n)} \equiv 1 \pmod{n}$
- e.g.

$a=3; n=10;$  (3 is relative prime to 10)

$$\phi(10) = \phi(2) \times \phi(5) = 4;$$

$$\text{hence } 3^4 = 81 \equiv 1 \pmod{10}$$

$$a=2; n=11; \phi(11)=10;$$

$$\text{hence } 2^{10} = 1024 \equiv 1 \pmod{11}$$

# Modular Multiplicative Inverse

---

- The modular multiplicative inverse of A mod C is the B value that makes  $A * B \text{ mod } C = 1$
- **Example: A=3, C=7**

**Calculate  $A * B \text{ mod } C$  for B values 0 through C-1**

$$3 * 0 \equiv 0 \pmod{7}$$

$$3 * 1 \equiv 3 \pmod{7}$$

$$3 * 2 \equiv 6 \pmod{7}$$

$$3 * 3 \equiv 9 \equiv 2 \pmod{7}$$

$$3 * 4 \equiv 12 \equiv 5 \pmod{7}$$

$$3 * 5 \equiv 15 \pmod{7} \equiv \underline{1} \pmod{7} \quad <----- \text{ FOUND INVERSE!}$$

$$3 * 6 \equiv 18 \pmod{7} \equiv 4 \pmod{7}$$

- **Example: A=2 C=6**

**Calculate  $A * B \text{ mod } C$  for B values 0 through C-1**

$$2 * 0 \equiv 0 \pmod{6}$$

$$2 * 1 \equiv 2 \pmod{6}$$

$$2 * 2 \equiv 4 \pmod{6}$$

$$2 * 3 \equiv 6 \equiv 0 \pmod{6}$$

$$2 * 4 \equiv 8 \equiv 2 \pmod{6}$$

$$2 * 5 \equiv 10 \equiv 4 \pmod{6}$$

No value of B makes  $A * B \text{ mod } C = 1$ . Therefore, A has no modular inverse  $(\text{mod } 6)$ .  
This is because 2 is not coprime to 6 (they share the prime factor 2).



# Extended Euclidean Algorithm

---

- It is easy to find multiplicative inverse for small value of n by just constructing multiplication table.
- But this method is not practical for large value of n. so for that we need Extended Euclidean algorithm.
- Euclidean Algorithm can be extended so that in addition to finding  $\text{gcd}(m,b)$ , if gcd is 1, the algorithm returns the multiplicative inverse of b.

# Finding Inverses – Extended Euclidean algorithm

```
EXTENDED_EUCLID(m, b)
```

1.  $(A_1, A_2, A_3) = (1, 0, m);$   
 $(B_1, B_2, B_3) = (0, 1, b)$
2. **if**  $B_3 = 0$   
**return**  $A_3 = \text{gcd}(m, b);$  no inverse
3. **if**  $B_3 = 1$   
**return**  $B_3 = \text{gcd}(m, b); B_2 = b^{-1} \bmod m$
4.  $Q = A_3 \text{ div } B_3$
5.  $(T_1, T_2, T_3) = (A_1 - Q B_1, A_2 - Q B_2, A_3 - Q B_3)$
6.  $(A_1, A_2, A_3) = (B_1, B_2, B_3)$
7.  $(B_1, B_2, B_3) = (T_1, T_2, T_3)$
8. **goto** 2

## Inverse of 49 in GF(37)

i.e. calling Extended\_Euclid(37, 49)

Q	A1	A2	A3	B1	B2	B3
-	1	0	37	0	1	49
0	0	1	49	1	0	37
1	1	0	37	-1	1	12
3	-1	1	12	4	-3	1

- Hence  $49^{-1} \equiv (-3) \pmod{37}$
- But,  $-3 \pmod{37} \equiv 34 \pmod{37}$ . Hence,

**multiplicative inverse of 49 modulo 37 is 34.**

# Inverse of 550 in GF(1759)

i.e. calling Extended\_Euclid(1759, 550)

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	355	1

# Primitive Roots

---

- A primitive root is any number  $g$  in multiplicative group that generates whole group of integers
- e.g.  $\mathbb{Z}_{14}^* = \{1, 3, 5, 9, 11, 13\}$

1 : 1

3 : 3, 9, 13, 11, 5, 1

5 : 5, 11, 13, 9, 3, 1

9 : 9, 11, 1

11 : 11, 9, 1

13 : 13, 1

Thus, 3 and 5 are primitive roots modulo 14

# Discrete Logarithms

---

- Discrete logarithms are fundamental to a number of public-key algorithms, including Diffie-Hellman key exchange and the digital signature algorithm (DSA).
- The inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo  $p$
- That is to find  $x$  such that  $y = g^x \pmod{p}$ 
  - This is written as  $x = \log_g y \pmod{p}$
- If  $g$  is a primitive root then it always exists, otherwise it may not,
  - e.g.  
 $x = \log_3 4 \pmod{13}$  has no answer !  
 $x = \log_2 3 \pmod{13} = 4$  (i.e.  $3=2^4 \pmod{13}$ )
- Exponentiation is relatively easy, but finding discrete logarithms is generally a **hard** problem

# Divisibility & the Division Algorithm

①

'b divides a' if  $a = mb$  [  $b/a$  ]

Eg:- 13 divides 182 ,  $13 \mid 182$   
-5 divides 30 ,  $-5 \mid 30$   
17 divides 0 ,  $17 \mid 0$

## Properties

- ↳ If  $a/1$ , then  $a = \pm 1$
- ↳ If  $a/b$  and  $b/a$ , then  $a = \pm b$
- ↳ Any  $b \neq 0$  divides 0
- ↳ If  $a/b$  and  $b/c$ , then  $a/c$
- ↳ If  $b/g$  and  $b/h$ , then  $b/(mg + nh)$  for arbitrary integers  $m$  &  $n$ .

### Proof

If  $b/g$ , then  $g = b * g_1$ , for some integer  $g_1$   
If  $b/h$ , then  $h = b * h_1$ , " " " $h_1$

$$\begin{aligned} mg + nh &= mbg_1 + nbh_1 \\ &= b * (mg_1 + nh_1) \end{aligned}$$

$\therefore b$  divides  $mg + nh$

Eg:-  $b = 7$ ,  $g = 14$ ,  $h = 63$

$$7 \mid 14 \quad \text{and} \quad 7 \mid 63$$

$$\therefore 7 \mid 14m + 63n \quad \text{where } m = 3, n = 2$$

$$\begin{aligned} 14m + 63n &= 14 * 3 + 63 * 2 \\ &= 2 * 3 [7 + 21] \\ &= 7 [6 + 18] \end{aligned}$$

## Division algorithm

If we divide positive int. 'a' by positive int. 'n'

quotient =  $q$   
remainder =  $r$  ('residue')

$$n \overline{) a \quad q}$$

$$\boxed{a = qn + r}$$

$$0 \leq r \leq n$$

$$q = \lfloor a/n \rfloor$$



$$qn \leq a$$

$$(q+1)n > a$$

The dist<sup>n</sup> from  $qn$  to  $a$  is  $r$ .

Eg:-  $a = 11$ ,  $n = 7$   
 $q = 1$ ,  $r = 4$

$$11 = 7 * 1 + 4$$

$$-11 = 7 * -2 + 3$$

Eg:-  $a = -11$ ,  $n = 7$   
 $q = -2$ ,  $r = 3$

## Euclidean Algorithm

↳ to determine (gcd) greatest common divisor  
or (hcf) highest common factor

GCD: The largest integer that divides both 'a' and 'b'

$$\text{gcd}(a, b) = \max[k, \text{such that } k/a \text{ and } k/b]$$

GCD should be positive

$$\begin{aligned} \text{gcd}(a, b) &= \text{gcd}(a, -b) = \text{gcd}(-a, b) = \text{gcd}(-a, -b) \\ &= \text{gcd}(|a|, |b|) \end{aligned}$$

→ Because all non-zero integers divide 0,  
 $\therefore \gcd(a, 0) = |a|$

→ 'a' and 'b' are relatively prime if  $\gcd(a, b) = 1$

How to find  $\gcd(a, b)$ ?

Divide a by b

$$b) \overline{a}(q_1)$$

$$a = q_1 b + r_1 \quad 0 < r_1 < b$$

$$b = q_2 r_1 + r_2 \quad 0 < r_2 < r_1$$

$$r_1 = q_3 r_2 + r_3 \quad 0 < r_3 < r_2$$

$$\overline{r_1}) \overline{b}(q_2)$$

$$\overline{r_2}) \overline{r_1}(q_3)$$

$$\overline{r_3}$$

$$r_{n-1} = q_{n+1} r_n + 0 \quad 0 < r_n < r_{n-1}$$

$$\gcd(a, b) = r_n$$

Eg:- find  $\gcd$  of 326 and 16

$$16) \overline{326}(20  
320  
\cancel{326}) \overline{16}(2$$

Ans :- 2

$$\begin{array}{r} 12 \\ 4 ) 6(1 \\ 4 \\ \hline 2 ) 4(2 \\ 4 \\ \hline 0 \end{array}$$

## Modular Arithmetic

Two integers 'a' and 'b' are said to be congruent modulo  $n$ ,

$$\text{if } [(a \bmod n) = (b \bmod n)]$$

This is written as

$$a \equiv b \pmod{n}$$

$$\text{Eg: } -73 \equiv 4 \pmod{23}$$

$$21 \equiv -9 \pmod{10}$$

### Properties

$$\hookrightarrow a \equiv b \pmod{n} \text{ if } n \mid (a-b)$$

$$\hookrightarrow a \equiv b \pmod{n} \text{ implies } b \equiv a \pmod{n}$$

$$\hookrightarrow a \equiv b \pmod{n}, b \equiv c \pmod{n} \text{ implies } a \equiv c \pmod{n}$$

### Modular Arithmetic operations properties.

$$\hookrightarrow [(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$$

$$\hookrightarrow [(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$$

$$\hookrightarrow [(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

Eg:- Find  $11^7 \bmod 13$

$$11^2 \bmod 13 = 121 \bmod 13 = 4$$

$$11^4 \bmod 13 = (11^2)^2 \bmod 13 = 4^2 \bmod 13 = 16 \bmod 13 = 3$$

$$11^7 = 11 \times 11^2 \times 11^4$$

$$\begin{aligned} 11^7 \bmod 13 &= (11 * 4 * 3) \bmod 13 \\ &= 132 \bmod 13 \\ &= 2 \end{aligned}$$

Addition modulo 8

$$\mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$$

### Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

## Multiplication modulo 8

(2)

$x \mid$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	<del>8</del> <sup>0</sup>	<del>10</del> <sup>2</sup>	<del>12</del> <sup>4</sup>	<del>14</del> <sup>6</sup>
3	0	3	6	<del>9</del> <sup>1</sup>	<del>12</del> <sup>4</sup>	<del>15</del> <sup>7</sup>	<del>18</del> <sup>2</sup>	<del>21</del> <sup>5</sup>
4	0	4	<del>8</del> <sup>0</sup>	<del>12</del> <sup>4</sup>	<del>16</del> <sup>0</sup>	<del>19</del> <sup>2</sup>	<del>22</del> <sup>4</sup>	<del>25</del> <sup>6</sup>
5	0	<del>5</del> <sup>1</sup>	<del>10</del> <sup>2</sup>	<del>15</del> <sup>7</sup>	<del>20</del> <sup>4</sup>	<del>25</del> <sup>1</sup>	<del>28</del> <sup>6</sup>	<del>31</del> <sup>3</sup>
6	0	<del>6</del> <sup>2</sup>	<del>12</del> <sup>4</sup>	<del>18</del> <sup>0</sup>	<del>24</del> <sup>2</sup>	<del>30</del> <sup>4</sup>	<del>36</del> <sup>6</sup>	<del>42</del> <sup>8</sup>
7	0	<del>7</del> <sup>3</sup>	<del>14</del> <sup>6</sup>	<del>21</del> <sup>5</sup>	<del>28</del> <sup>4</sup>	<del>35</del> <sup>1</sup>	<del>42</del> <sup>7</sup>	<del>49</del> <sup>9</sup>

Additive & Multiplicative inverse modulo 8

$x \mid$	w	-w	$w^{-1}$
0	0	-	-
1	7	-1	-
2	6	-	-
3	5	3	-
4	4	-	-
5	3	5	-
6	2	-	-
7	1	7	-

Set of residues / residue classes (mod n)

$$\mathbb{Z}_n = \{0, 1, 2, \dots, (n-1)\}$$

[residues]

Residue classes (mod n) are  $[0], [1], [2], \dots, [n-1]$   
 where  $[x] = \{a : a \text{ is an integer, } a \equiv x \pmod{n}\}$

Eg:- The residue classes (mod 4) are

$$[0] = \{ \dots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \dots \}$$

$$[1] = \{ \dots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \dots \}$$

$$[2] = \{ \dots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \dots \}$$

$$[3] = \{ \dots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \dots \}$$

The smallest non-negative integer is used to represent the residue class.

$\mathbb{Z}_n$  is a commutative ring with a multiplicative identity element.

$$\left\{ \begin{array}{l} \text{If } (a+b) \equiv (a+c) \pmod{n} \\ \text{then } b \equiv c \pmod{n} \end{array} \right.$$

$$\text{Eg: } (5+23) \equiv (5+7) \pmod{8}$$

then  $23 \equiv 7 \pmod{8}$

Properties for modular arithmetic for integers in  $\mathbb{Z}_n$

- Commutative laws.  $(w+x) \pmod{n} = (x+w) \pmod{n}$
- Associative laws
- Distributive laws  $(0+w) \pmod{n} = w \pmod{n}$   
 $(1*w) \pmod{n} = w \pmod{n}$
- Identities
- Additive inverse For each  $w \in \mathbb{Z}_n$ , there exists  $z \in \mathbb{Z}$  such that  $w+z \equiv 0 \pmod{n}$

$$\left\{ \begin{array}{l} \text{If } (a \times b) \equiv (a \times c) \pmod{n} \\ \text{then } b \equiv c \pmod{n} \end{array} \right. \text{ if } 'a' \text{ is relatively prime to } n$$

Two integers are relatively prime if their only common positive integer factor is 1

$$ab \equiv ac \pmod{n}$$

$$(a^{-1})ab \equiv (a^{-1})ac \pmod{n}$$

$$b \equiv c \pmod{n}$$

Eg:- 6 and 8 are not relatively prime  $[\gcd(6, 8) \neq 1]$

$$6 \times 3 = 18 \equiv 2 \pmod{8}$$

$$6 \times 7 = 42 \equiv 2 \pmod{8}$$

$$\text{Yet } 3 \not\equiv 7 \pmod{8}$$

With  $a=6$  and  $n=8$ ,

$Z_8$	0	1	2	3	4	5	6	7
Multiply by 6	0	6	12	18	24	30	36	42
Residues	0	6	4	2	0	6	4	2

∴ There is not a unique inverse to the multiply operation

With  $a=5$ . and  $n=8$

$$\gcd(5, 8) = 1$$

$Z_8$	0	1	2	3	4	5	6	7
Multiply by 5	0	5	10	15	20	25	30	35
Residues	0	5	2	7	4	1	6	3

→ Integers 1, 3, 5 and 7 have a multiplicative inverse in  $Z_8$ , but 2, 4 and 6 do not.

### Euclidean Algorithm Revisited

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

$$\text{Eg:- } \gcd(55, 22) = \begin{aligned} &= \gcd(22, 55 \bmod 22) \\ &= \gcd(22, 11) = 11 \end{aligned}$$

~~Result :-~~  $\underline{\gcd(a, b)}$ .  
operator

$\gcd(b, a \bmod b)$ ,  
 $b \in \{a \bmod b\} + r$   
 $b \leq k \cdot r + r'$

### Extended Euclidean Algorithm

(useful in RSA).

→ not only it calculates the gcd 'd', but also 2 additional integers 'x' and 'y' that satisfy the following equation :-

$$ax + by = d = \gcd(a, b)$$

$$\text{Eg:- } \gcd(42, 30) = 6 = 42x + 30y = 6(7x + 5y)$$

	$x$	$y$	$-1$	$0$	$1$	$2$	$3$
3	-3	2	-174	-132	-90	-48	-6
2	-216	-144	-102	-60	-18	24	36
1	186	-114	-72	-30	12	54	96
-1	-156	-84	-42	0	42	84	126
0	-126	-54	-12	30	72	114	156
1	-96	-24	18	60	102	144	186
2	-66	6	48	90	132	174	216
3	-36						

$42x + 30y = 6(7x + 5y)$  is a multiple of 6.

Note  $\gcd(42, 30) = 6$ .

Eg:- Use  $a = 1759$  and  $b = 550$  and solve for  
 $1759x + 550y = \gcd(1759, 550)$

<u>i</u>	<u><math>x_i</math></u>	<u><math>y_i</math></u>	<u><math>x_i</math></u>	<u><math>y_i</math></u>
-1	1759		1	0
0	550		0	1
1	109	3	1	-3
2	5	5	-5	16
3	4	21	106	-339
4	1	1	-111	355
5	[0]	4	(gcd)	(14)

$\gcd(1759, 550) = 1 = 1759(-111) + 550(355)$

$$550 \overline{) 1759} (3$$

$$-1650 \overline{) 169} (5$$

$$5) 109(21$$

$$-105 \overline{) 4} (1$$

$$\frac{4}{1} \overline{) 4} (4$$

$$\begin{cases} x_n = x_{n-2} - q_n x_{n-1} \\ y_n = y_{n-2} - q_n y_{n-1} \end{cases}$$

## Prime Numbers

(3)

An integer  $p > 1$  is a prime no. if and only if its only divisors are  $\pm 1$  and  $\pm p$ .

Any integer 'a' can be factored in a unique way as:-

$$a = p_1^{a_1} \times p_2^{a_2} \times p_3^{a_3} \dots \times p_t^{a_t}$$

where  $p_1 < p_2 < p_3 \dots < p_t$  are prime numbers and each  $a_i$  is a positive integer.

Given,

$$a = \prod_{p \in P} p^{ap} \quad \text{and} \quad b = \prod_{p \in P} p^{bp}$$

If  $a/b$  then  $ap \leq bp \forall p$

If  $k = \gcd(a; b)$ , then  $k_p = \min(ap, bp) \forall p$

Eg:-  $300 = 2^2 \times 3^1 \times 5^2$

$$18 = 2^1 \times 3^2$$

$$\gcd(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$$

## FERMAT'S THEOREM!

(9mp. in public key cryptography)

$p$  is prime.

$a$  is positive integer not divisible by  $p$

$$a^{p-1} \equiv 1 \pmod{p}$$

Eg:-  $a^p = 7$ ,  $p = 19$

$$a^{p-1} = 7^{18}$$

$$7^{18} \pmod{19} = 1$$

$$7^{18} \equiv 1 \pmod{19}$$

$$a^{p-1} \equiv 1 \pmod{p}$$

$$7^2 \pmod{19} = 11$$

$$7^4 \pmod{19} = 121 \pmod{19} = 7$$

$$7^8 \pmod{19} = 49 \pmod{19} = 11$$

$$7^{16} \pmod{19} = 121 \pmod{19} = 7$$

$$7^{18} \pmod{19} = (7 \times 11) \pmod{19} = 1$$

Proof :-

$$p : \{1, 2, \dots, p-1\}$$

multiply each element by  $a$  and modulo  $p$

$$X = \{a \bmod p, 2a \bmod p, \dots, (p-1)a \bmod p\}$$

None of the elements is  $0$  ~~because~~ as  $p$  does not divide  $a$

No two integers in  $X$  are equal.

$$\because ja \equiv ka \pmod{p}$$

where  $1 \leq j < k \leq p-1$

$$\gcd(a, p) = 1$$

$$\text{so } j \not\equiv k \pmod{p}$$

This is impossible because  
j & k are both positive int. less  
than  $p$ :  $j \neq k$

$$[a \times 2a \times 3a \times \dots \times (p-1)a] \not\equiv \pmod{p}$$

$$\Rightarrow [1 \times 2 \times 3 \times \dots \times (p-1)] \pmod{p}$$

$$[a^{p-1} (p-1)!] \pmod{p} \Rightarrow [(p-1)!] \pmod{p}$$

$(p-1)!$  is relatively prime to  $p$

$$a^{p-1} \pmod{p} = 1 \pmod{p}$$

$$\boxed{a^{p-1} \equiv 1 \pmod{p}}$$

Hence Proved

Alternate form of Fermat's theorem

$$\boxed{a^p \equiv a \pmod{p}}$$

$$\text{Eg:- } a=3, p=5$$

$$\begin{array}{c} 3^5 \equiv 3 \pmod{5} \\ 3^2 * 3^2 * 3 \\ \underbrace{4} \quad \underbrace{4} \quad \underbrace{3} \\ 16 \\ \underbrace{1} \\ 3 \end{array}$$

## Euler's Totient function : $\phi(n)$

$\phi(n) \Rightarrow$  defined as the no. of positive integers less than  $n$  & relatively prime to  $n$ .

$\phi(37) = 36$  (All no.s 1 to 36 are relatively prime to 37, because 37 is prime no.)

$$\phi(35) = 24.$$

{1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, ~~14, 15, 16, 17, 18, 19, 20,~~  
23, 24, 26, 27, 29, 31, 32, 33, 34}

For prime numbers ( $p$ ),

$$\phi(p) = p - 1$$

Let 2 prime no.s  $p$  &  $q$ ,  $p \neq q$   
 $n = p \times q$

$$\begin{aligned}\phi(n) &= \phi(p \times q) = \phi(p) * \phi(q) \\ &= (p-1)(q-1)\end{aligned}$$

Ex:-  $\phi(21) = \phi(3) * \phi(7)$   
=  $2 * 6$   
= 12

## Euler's Theorem

For every ' $a$ ' and ' $n$ ' that are relatively prime:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

If  $n$  is prime.

$$\phi(n) = (n-1) \text{ then } a^{\phi(n)} = a^{n-1} \equiv 1 \pmod{n}$$

[ $\because$  Fermat's theorem]

Let set of integers :  $R = \{x_1, x_2, \dots, x_{\phi(n)}\}$

positive Each element  $x_i$  of  $R$  is a unique int less than  $n$  with  $\gcd(x_i, n) = 1$

$$S = \{(ax_1 \pmod{n}), (ax_2 \pmod{n}), \dots, (ax_{\phi(n)} \pmod{n})\}$$

S is a permutation of R because

→ Because  $a$  is relatively prime to  $n$

$\& \quad x_i \quad " \quad " \quad n$

∴  $x_i$  must also be relatively prime to  $n$

Thus all members of S are less than  $n$ ,  
and that are relatively prime to  $n$

→ There are no duplicates in S

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \left[ \prod_{i=1}^{\phi(n)} x_i \right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Euler's Theorem

$$\text{Eg:- } a = 3, \quad n = 10$$

$$\phi(10) = 4 \quad \{1, 3, 7, 9\}$$

$$a^{\phi(n)} = 3^4 = 81$$

$$81 \equiv 1 \pmod{10}$$

Alternate form of Euler's Theorem

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

## Testing for primality

For many cryptographic algorithms, it is necessary to select one or more very large prime nos at random. Thus, the task is :- determining whether a given large no. is prime.

### Properties of Prime Numbers.

(1)  $p$ : prime

$a$  is a +ve integer less than  $p$ ,  
then  $a^2 \bmod p = 1$  iff  $a \bmod p = 1$   
or  $a \bmod p = -1 \bmod p = p-1$

Proof:-

$$(a \bmod p)(a \bmod p) = a^2 \bmod p.$$

for  $a^2 \bmod p = 1$ , either  $a \bmod p = 1$   
or  $a \bmod p = -1$

Conversely, if  $a^2 \bmod p = 1$ , then  $(a \bmod p)^2 = 1$

which is true only for  
 $a \bmod p = \pm 1$ .

(2)  $p$ : prime no.  $> 2$

$k > 0$ ,  $q$ : odd

$$\text{then } p-1 = 2^k q.$$

Let  $a$ : int,  $1 < a < p-1$

Then one of the two conditions is true.

$$(a) a^q \equiv 1 \bmod p \quad a^q, a^{2q}, a^{4q}, \dots, a^{2^{k-1}q}$$

(b) One of the numbers  
is congruent to  $-1 \bmod p$

There is some int  
such that

$$1 \leq j \leq k$$

$$a^{2^{j-1}q} \bmod p = -1 \bmod p = p-1$$

$$\text{or } a^{2^{j-1}q} \equiv -1 \bmod p$$

# Miller-Rabin Test / Rabin-Miller Test

If  $n$  is prime, then either the first element in the list of residues or remainders

$$(a^q, a^{2q}, a^{3q}, \dots, a^{k+1}q, a^{2k}q) \text{ mod } n$$

equals 1

or, some element in the list equals  $(n-1)$

Otherwise,  $n$  is composite (not prime)

$$\text{Eg:- } n = 2047 = 23 * 89$$

$$\text{then } n-1 = 2046 \\ = 2 * 1023$$

$$2^{1023} \text{ mod } 2047 = 1$$

2047 meets the condition, but is not prime

## TEST ( $n$ )

(1) Find int.  $k, q$  with  $k > 0$ ,  $q$ : odd, so that  
 $n-1 = 2^k q$

(2) Select a random int  $a$ ,  $1 < a < n-1$

(3) if  $a^q \text{ mod } n = 1$ , then return ("inconclusive")

(4) For  $j=0$  to  $k-1$ , do

(5) If  $a^{2^j q} \text{ mod } n = n-1$ , then return ("inconclusive")

(6) return ("composite")

Eg:- test for  $n = 29$  (prime)

$$k = 2 \\ q = 7$$

let  $a = 10$  (random)

$$a^q \text{ mod } n = 10^7 \text{ mod } 29 = 17$$

which is neither 1 nor  $n-1$ .

$$\text{Next calculate } 2(10^7)^2 \text{ mod } 29 = 28.$$

Return "inconclusive"

$10^1 \text{ mod } 29$	10
$10^2 \text{ mod } 29$	10
$10^3 \text{ mod } 29$	19
$10^4 \text{ mod } 29$	24
$10^5 \text{ mod } 29$	8
$10^6 \text{ mod } 29$	17
$10^7 \text{ mod } 29$	1

let  $a = 2 \cdot (\text{random})$

$$2^2 \bmod n$$

$$2^7 \bmod 29 = 12$$

$$2^{29} \bmod n$$

$$2^{14} \bmod 29 = 28$$

Rabin Test for all  $a \in [1, 28]$

We get same "inconclusive" result

Test for

Eg:-  $n = 221$  (composite)

$$221 = 13 * 17$$

$$n-1 = 220 = 2 * 110$$

$$= 2 * 2 * 55$$

$$= 2^2 \cdot (55)$$

$$= 2^k \cdot q$$

$$k=2 \\ q=55$$

d) let  $a = 5$

$$a^a \bmod n = 5^{55} \bmod 221 = 112$$

$$\neq 1.$$

$$\neq n-1(220)$$

$$5^{55 \cdot 2} \bmod 221 = 168$$

After checking all values of  $j$ , test returns composite.

Only for  $a = 21, 47, 174, 200$ .  
test returns "inconclusive".

For all other  $a \in [1, 220]$ , test returns "composite"

Miller's test

Repeatedly invoke TEST( $n$ ) using randomly chosen values for  $a$ . If at any point, TEST returns composite, then  $n$  is determined to be non-prime.

If TEST continues to return inconclusive for  $t$  tests, then for sufficiently large values of  $t$ , assume that  $n$  is prime.

## Chinese Remainder Theorem (example)

used to solve a set of different congruent equations with one variable but different moduli which are relatively prime.

$$x \equiv a_1 \pmod{m_1}$$

$$x \equiv a_2 \pmod{m_2}$$

$$\vdots$$

$$x \equiv a_n \pmod{m_n}$$

CRT states that the above eqn has a unique solution if  $m_1, m_2, \dots, m_n$  are relatively prime.

$$x = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + \dots + a_n M_n M_n^{-1}) \pmod{M}$$

Eg: 1 Solve the following equations using CRT

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

$$a_1 = 2$$

$$a_2 = 3$$

$$a_3 = 2$$

$$m_1 = 3$$

$$m_2 = 5$$

$$m_3 = 7$$

Solution :

$$x = (a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1}) \pmod{M}$$

unique soln exists because  $(3, 5, 7)$  are relatively prime.

$$\begin{aligned} M &= m_1 \times m_2 \times m_3 \\ &= 3 \times 5 \times 7 \\ &= 105 \end{aligned}$$

$$M_1 = \frac{M}{m_1} = \frac{105}{3} = 35$$

$$M_2 = \frac{M}{m_2} = \frac{105}{5} = 21$$

$$M_3 = \frac{M}{m_3} = \frac{105}{7} = 15$$

$$M_1 * M_1^{-1} = 1 \pmod{m_1}$$

$$35 * M_1^{-1} = 1 \pmod{3}$$

$$M_1^{-1} = 2$$

$$M_2 * M_2^{-1} = 1 \pmod{m_2}$$

$$21 * M_2^{-1} = 1 \pmod{5}$$

$$M_2^{-1} = 1$$

$$M_3 * M_3^{-1} = 1 \pmod{m_3}$$

$$15 * M_3^{-1} = 1 \pmod{7}$$

$$M_3^{-1} = 1$$

$$\begin{aligned} X &= \left( a_1 M_1 M_1^{-1} + a_2 M_2 M_2^{-1} + a_3 M_3 M_3^{-1} \right) \bmod m \\ &= (2 * 35 * 2 + 3 * 21 * 1 + 2 * 15 * 1) \bmod 105 \\ &= (140 + 63 + 30) \bmod 105 \\ &= 233 \bmod 105 \\ &= 23 \end{aligned}$$

$$23 \equiv 2 \bmod 3$$

$$23 \equiv 3 \bmod 5$$

$$23 \equiv 2 \bmod 7$$

# UNIT-2\_1



**Stream ciphers and block ciphers**

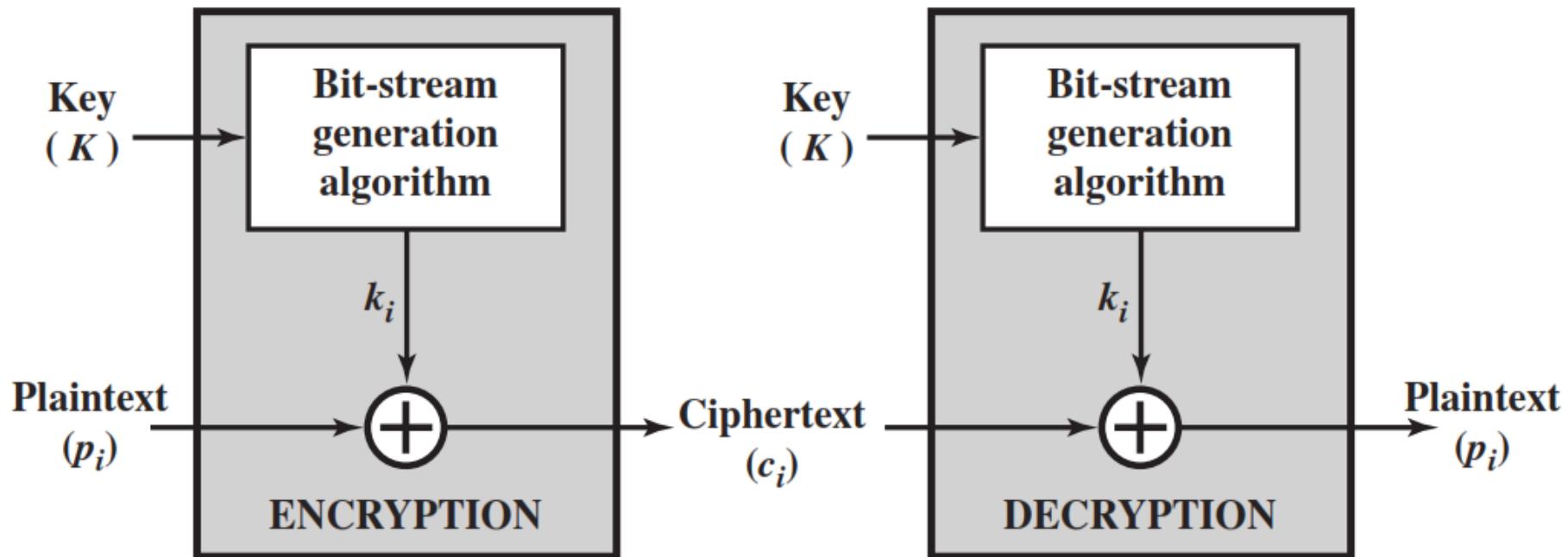
# Unit-2

---

- Stream ciphers and block ciphers
- Block Cipher structure
- Data Encryption standard (DES)
- Design principles of block cipher
- AES with structure
- AES Transformation functions
- Key expansion

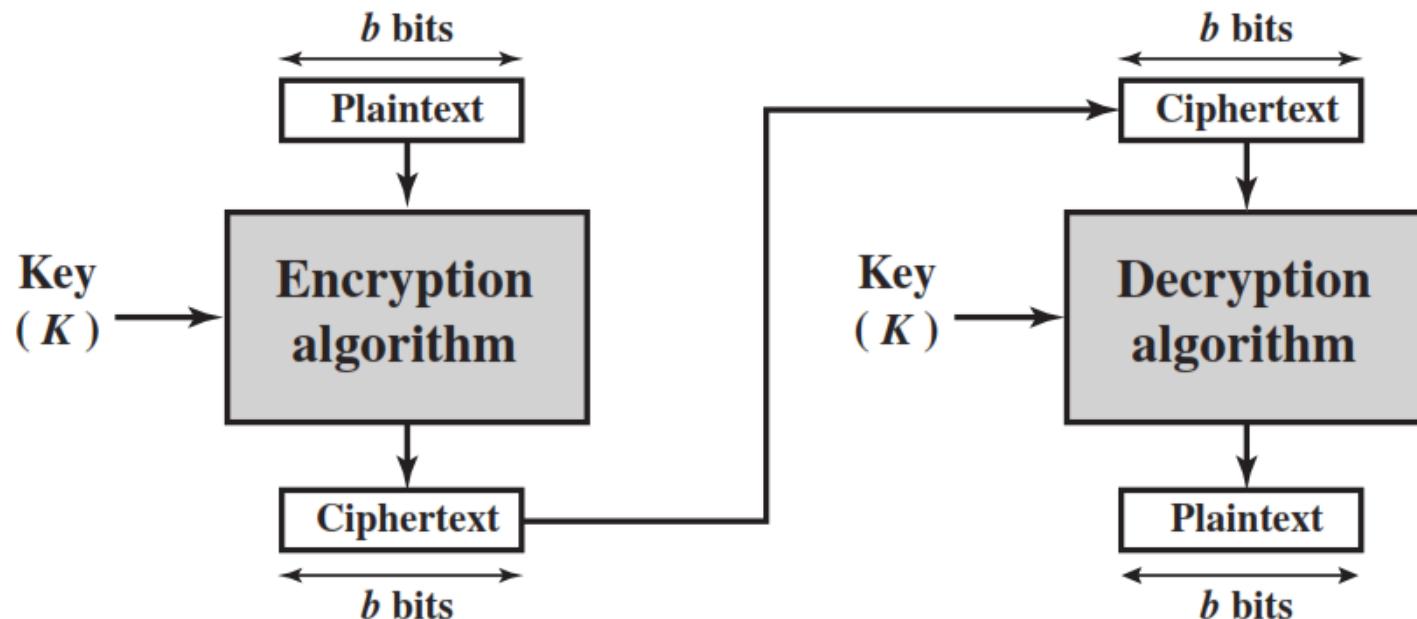
# Stream Cipher

- A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time.
- Examples of classical stream ciphers are Autokeyed Vigenère cipher ,A5/1, RC4 and Vernam cipher.



# Block Cipher

- A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Typically, a block size of **64 or 128** bits is used.
- Examples are Feistel Cipher, DES, Triple DES and AES

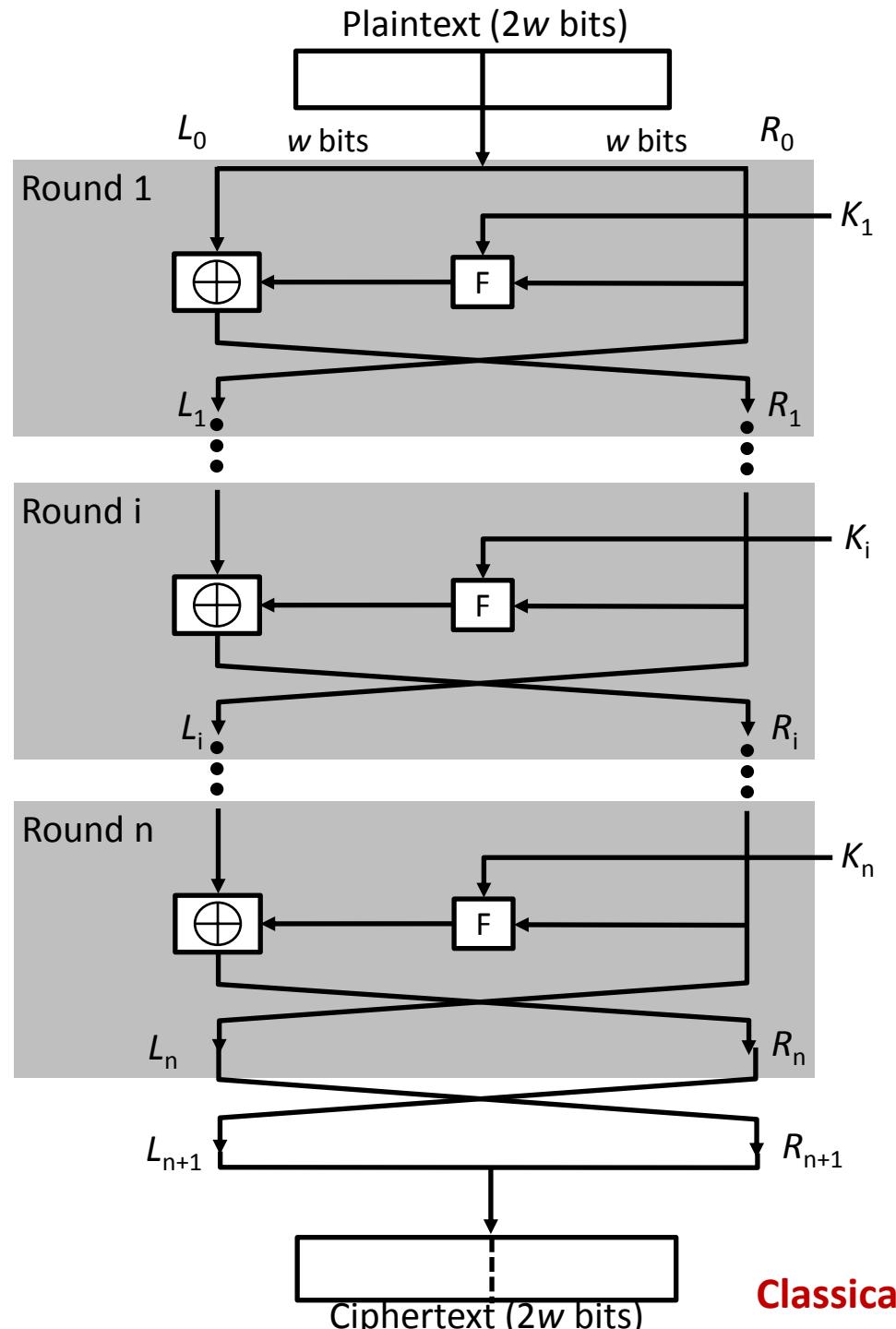


# Diffusion and Confusion

---

- **Diffusion** hides the relationship between the ciphertext and the plaintext.
- This is achieved by having each plaintext digit affect the value of many ciphertext digits.
- **Confusion** hides the relationship between the ciphertext and the key.
- This is achieved by the use of a complex substitution algorithm.

# Feistel Cipher Structure Or Block Cipher Structure



Classical Feistel Network

# Feistel Cipher Structure

---

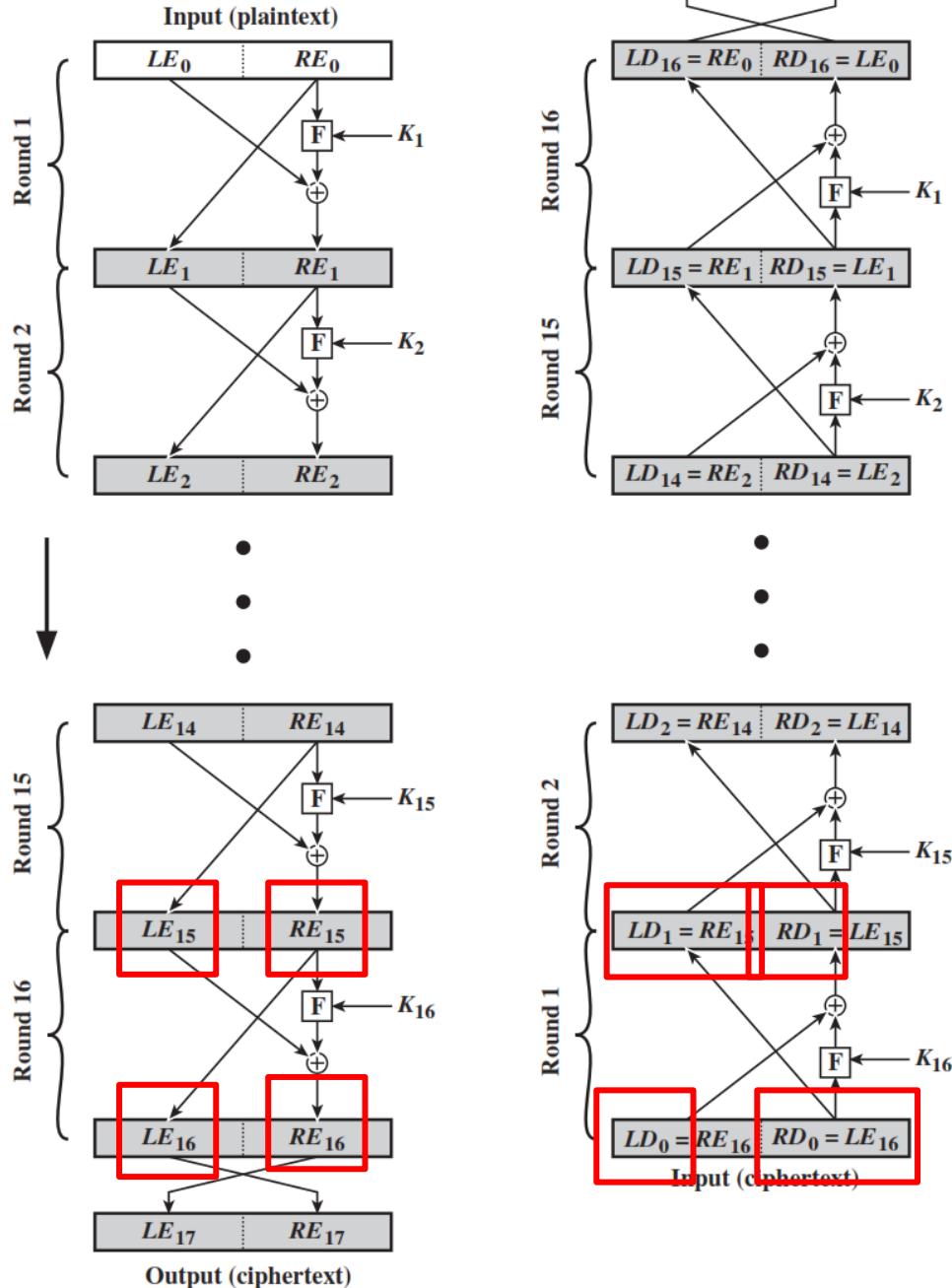
- Input plaintext block of length  $2w$  bits
- key  $K = n$  bits , Sub-keys:  $K_1, K_2, \dots, K_n$  (Derived from  $K$ )
- All rounds have the same structure.
- A **substitution** is performed by taking exclusive-OR on left half( $L_i$ ) of the data and the output of round function  $F$  which has inputs right half( $R_i$ ) and sub key  $k_i$ .
- A **permutation** is performed that consists of interchange of two halves of data.
- This structure is called **Substitution-Permutation Network (SPN)**

# Feistel Network Factors

---

- **Block size:** Common block size of 64-bit. However, the new algorithms uses a 128-bit, 256-bit block size.
- **Key size:** Key sizes of 64 bits or less are now widely considered to be insufficient, These days at least 128 bit, more better, e.g. 192 or 256 bit
- **Number of rounds:** A typical size is 16 rounds.
- **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

# Feistel Encryption & Decryption



- Prove that o/p of first round of Decryption is equal to 32-bit swap of i/p of 16<sup>th</sup> round of Encryption
  - $LD_1 = RE_{15}$  &  $RD_1 = LE_{15}$
  - On Encryption Side:

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$$

- ## ■ On Decryption Side:

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \oplus F(RD_0, K_{16})$$

$$= RE_{16} \oplus F(RE_{15}, K_{16})$$

$$= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$$

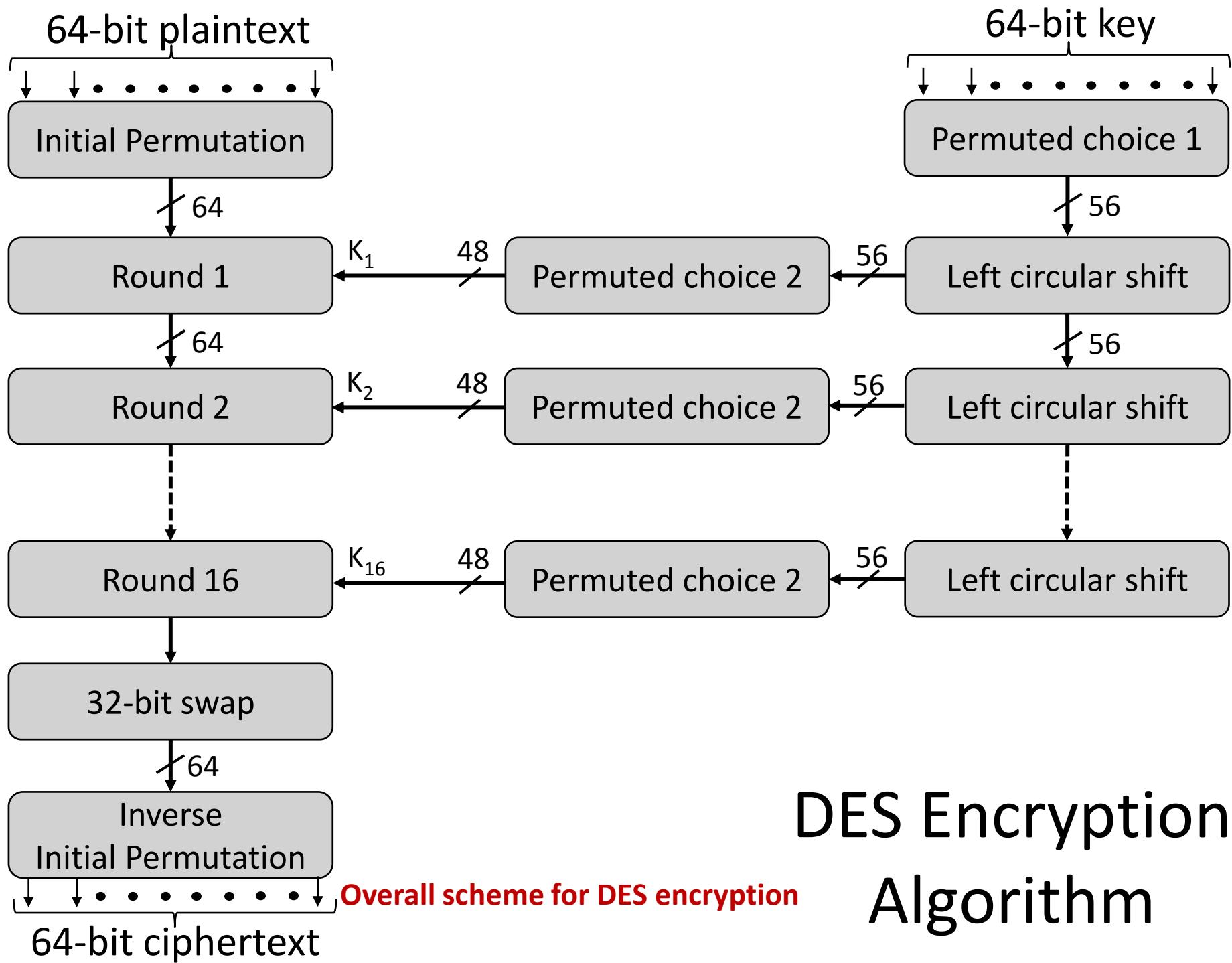
*Thus,*

$$LD_1 = RE_{15} \text{ & } RD_1 = LE_{15} \quad \oplus C]$$

# Data Encryption Standard (DES)

---

- Type: Block Cipher
- Block Size : 64-bit
- Key Size: 64-bit, with only 56-bit effective
- Number of Rounds: 16

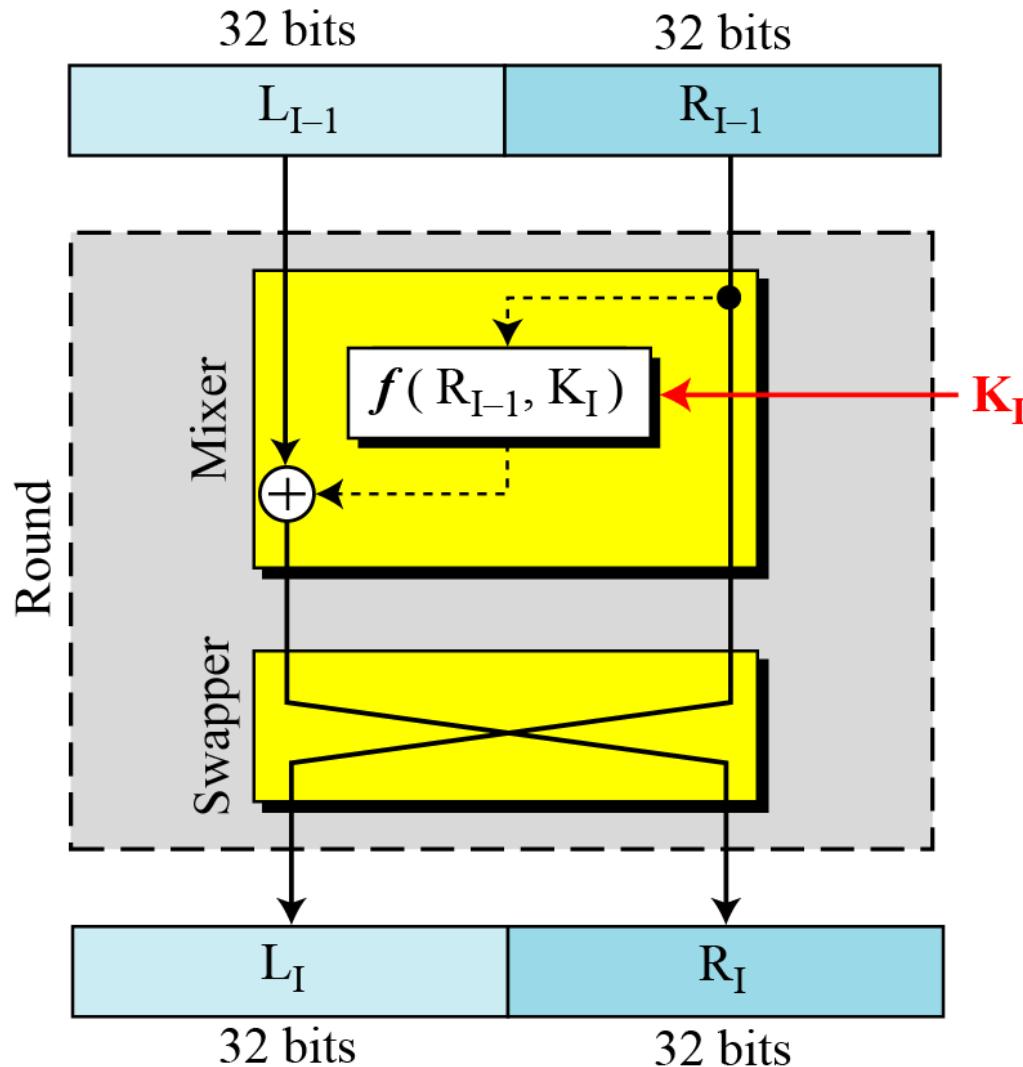


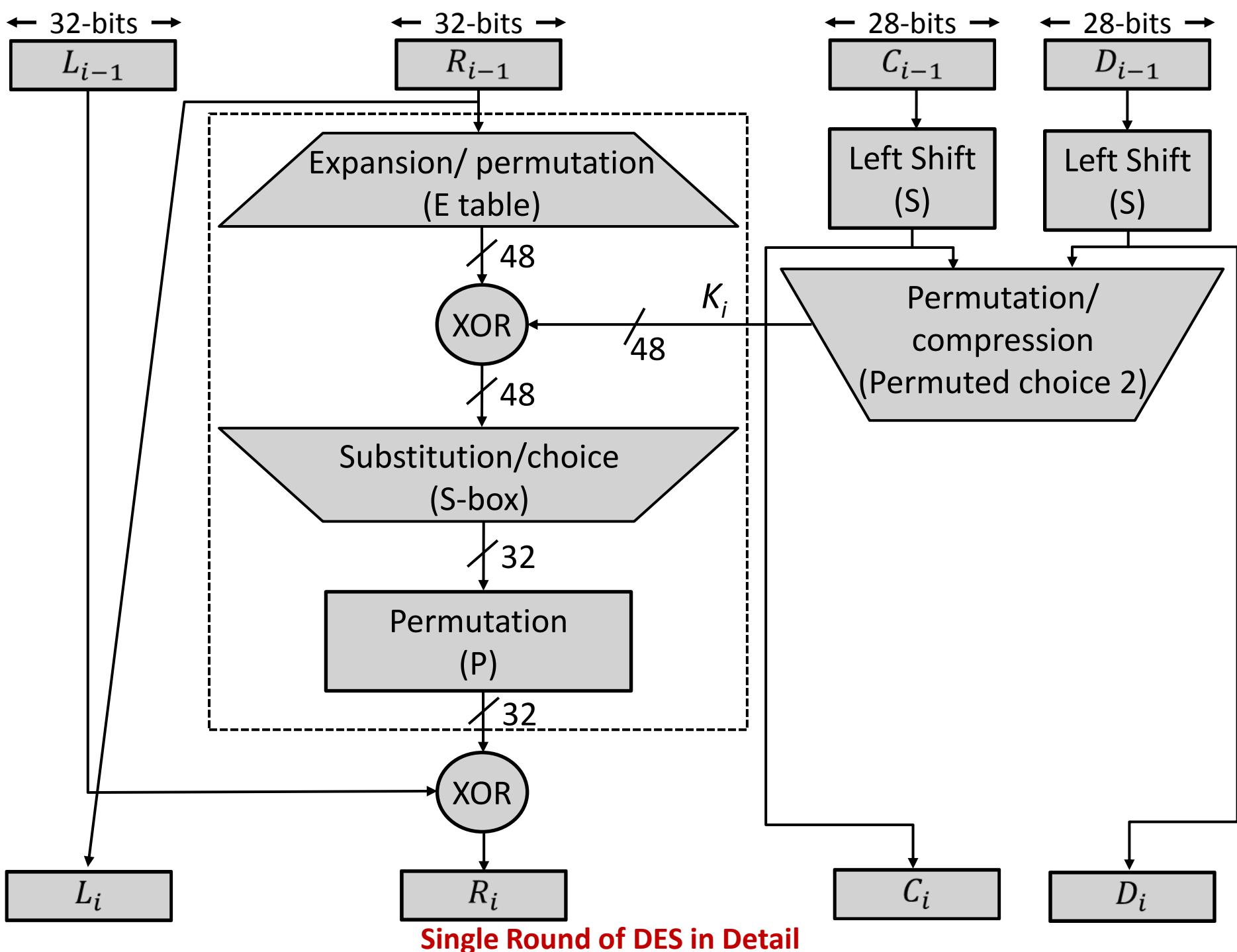
# DES Encryption Algorithm (Cont...)

---

- First, the 64-bit plaintext passes through an **initial permutation** (IP) that rearranges the bits to produce the permuted input.
- This is followed by a phase consisting of sixteen rounds of the same function, which involves both **permutation** and **substitution** functions.
- Finally, the preoutput is passed through a permutation that is the **inverse of the initial permutation** function, to produce the 64-bit ciphertext.
- The 56-bit key is passed through a **permutation function**.
- For each of the sixteen rounds, a subkey ( $K_i$ ) is produced by the combination of a **left circular shift** and a **permutation**.

# DES Single Round



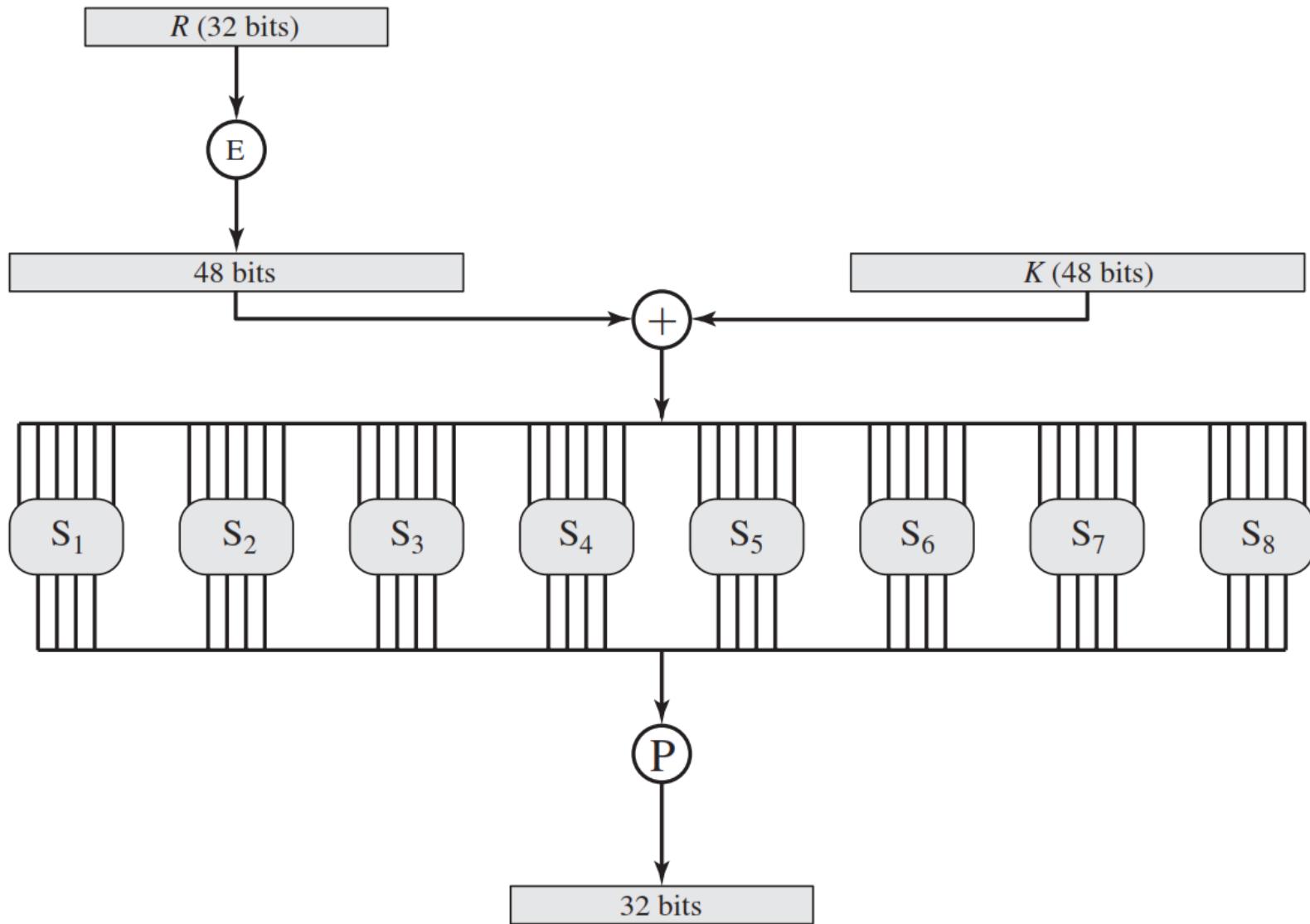


# DES Single Round (Cont...)

---

1. Key Transformation
  - Permutation of selection of sub-key from original key
2. Expansion Permutation (E-table)
  - Right half is expanded from 32-bits to 48-bits
3. S-box Substitution
  - Accepts 48-bits from XOR operation and produce 32-bits using 8 substitution boxes (each S-boxes has a 6-bit i/p and 4-bit o/p).
4. P-Box Permutation
5. XOR and Swap

# Role of S-boxes in the function F



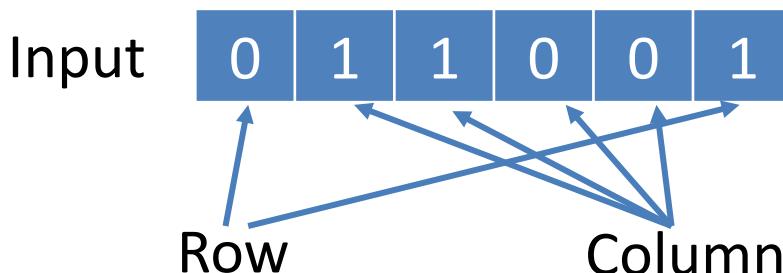
# Role of S-box (Cont...)

- The outer two bits of each group select one row of an S-box.
- Inner four bits selects one column of an S-box.

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

S-box 1

- Example:



# Avalanche Effect

- Desirable property of any encryption algorithm is that a change in one bit of the plaintext or of the key should produce a change in many bits of cipher text.
- DES performs strong **avalanche effect**.

Plaintext: 0000000000000000

Key: 22234512987ABB23

Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001

Key: 22234512987ABB23

Ciphertext: 0A4ED5C15A63FEA3

- Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits.
- This means that changing approximately 1.5 % of the plaintext creates a change of approximately 45 % in the ciphertext.

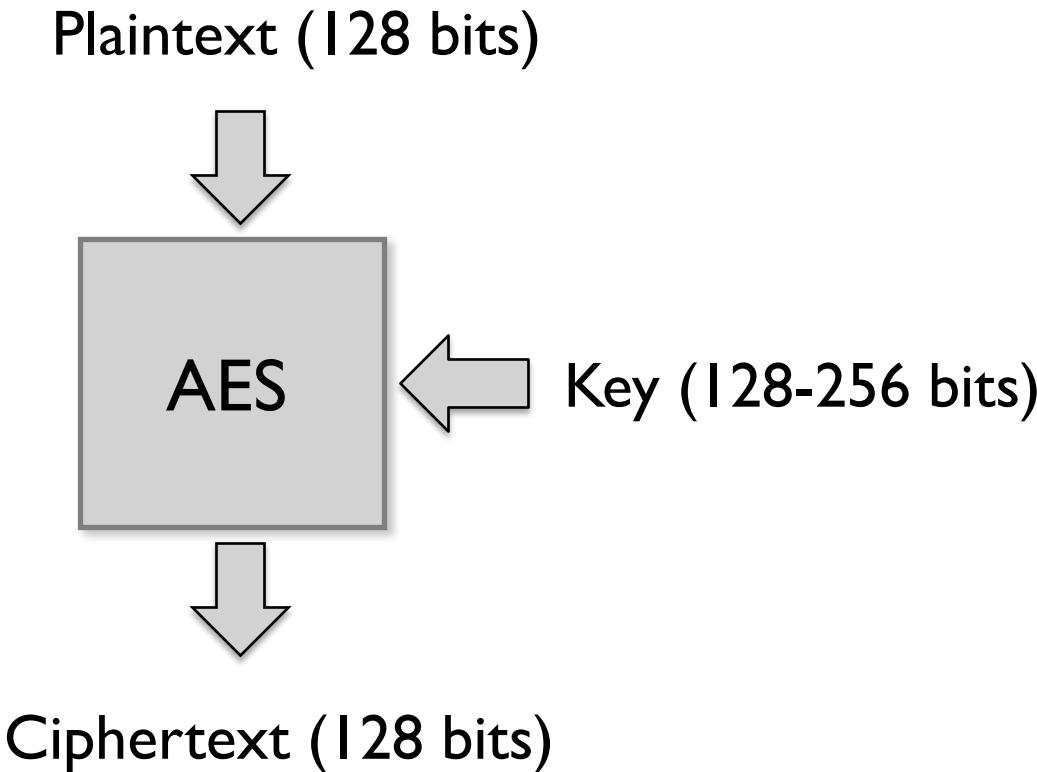
# AES (Advanced Encryption Standard)

- The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits.

<b>Key size (words/ bytes/ bits)</b>	<b>4/16/128</b>	<b>6/24/192</b>	<b>8/32/256</b>
<b>Block size (words/ bytes/ bits)</b>	<b>4/16/128</b>	<b>4/16/128</b>	<b>4/16/128</b>
<b>Round key size (words/ bytes/ bits)</b>	<b>4/16/128</b>	<b>4/16/128</b>	<b>4/16/128</b>
<b>Number of Rounds</b>	<b>10</b>	<b>12</b>	<b>14</b>
<b>Expanded Key Size (words)</b>	<b>44</b>	<b>52</b>	<b>60</b>

- AES designed to have characteristics
  - Resistance against all known attacks
  - Speed and code compactness on a wide range of platforms
  - Design simplicity

# AES (Advanced Encryption Standard)



# AES Overview

- Simple Repeating structure
- Cipher begins and ends with Add Round Key
  - Forms a Vernam Cipher or “One Time Pad”
    - Any other stage applied at the beginning or end is reversible without the key
- Other three stages provide confusion, diffusion and nonlinearity
- n standard rounds, n is 10,12 or 14
- The first n-1 rounds are similar consisting of
  - ByteSub
  - ShiftRow
  - MixColumn
  - AddRoundKey
- The last round only perform the transformations
  - ByteSub
  - ShiftRow
  - AddRoundKey

## Initialization

1. Expand 16-byte key to get the actual **key block** to be used.
2. Initialize 16-byte plaintext block called as **state**.
3. XOR the **state** with the **key block**.

# AES Structure

- The first  $n-1$  rounds consist of four distinct transformation functions.

SubBytes

- The 16 input bytes are substituted using an **S-box**

ShiftRows

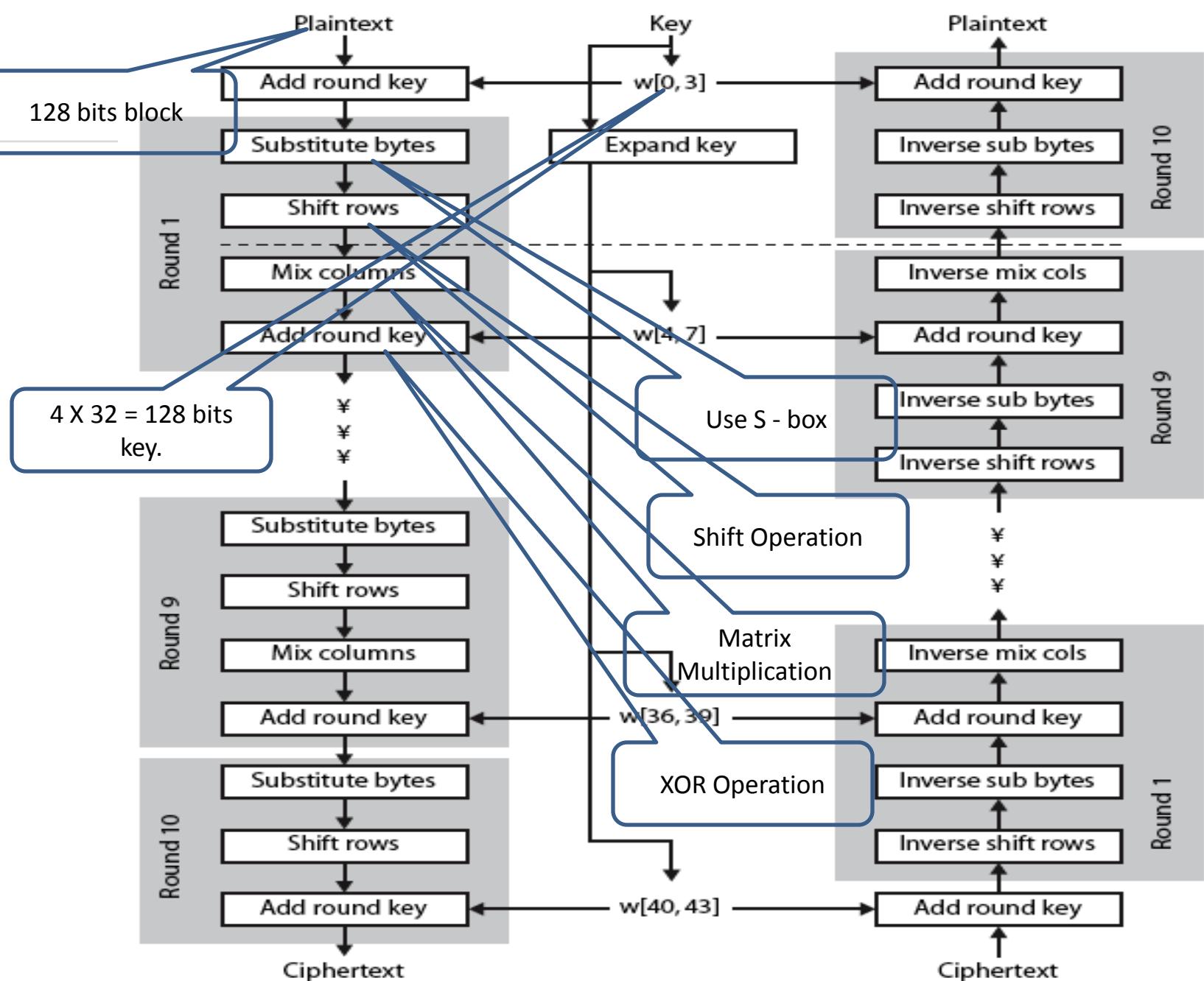
- Each of the four rows of the matrix is shifted to the left

MixColumns

- Each column of four bytes is now transformed using a special mathematical function.

AddRoundKey

- The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key.

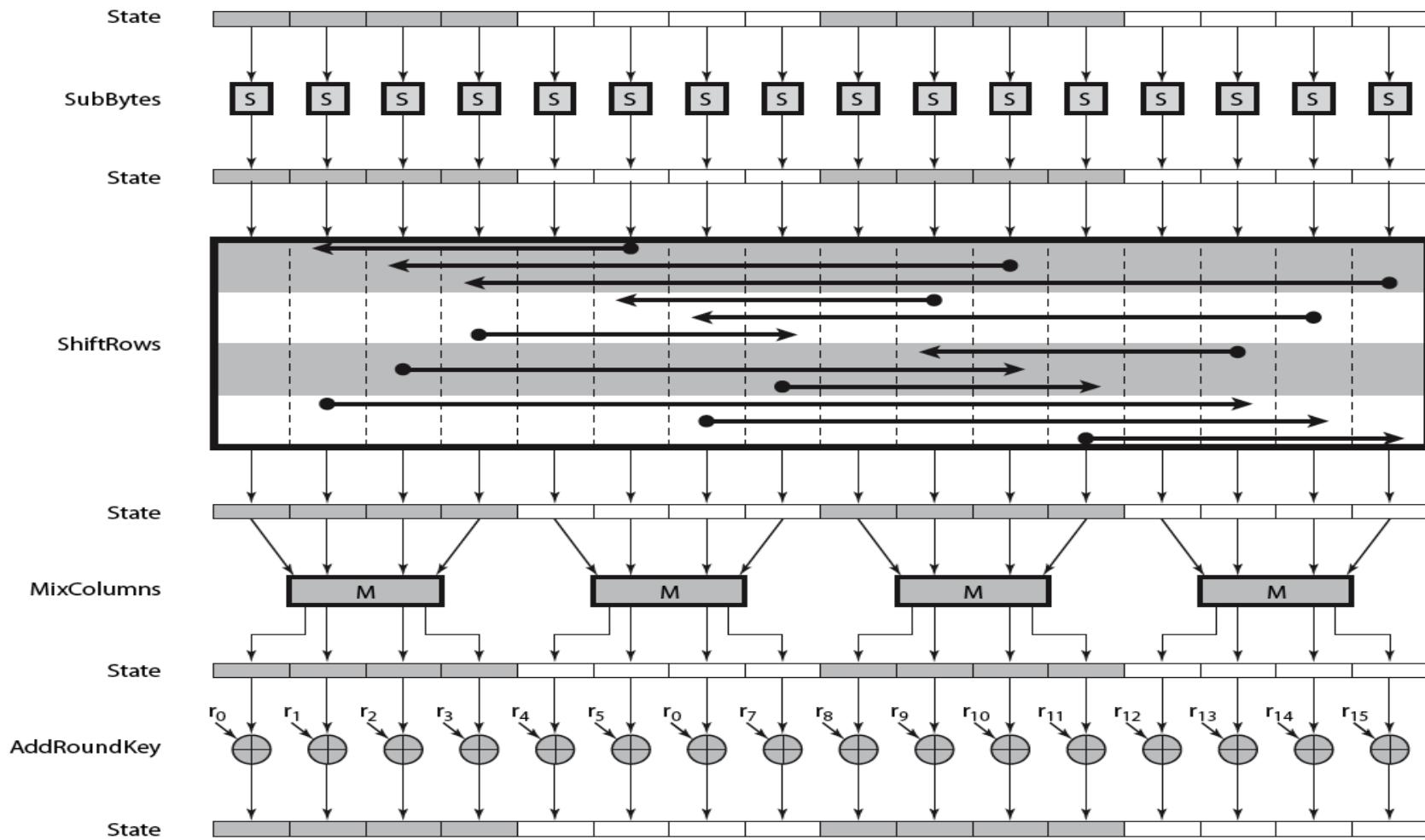


## AES Overall Structure

(a) Encryption

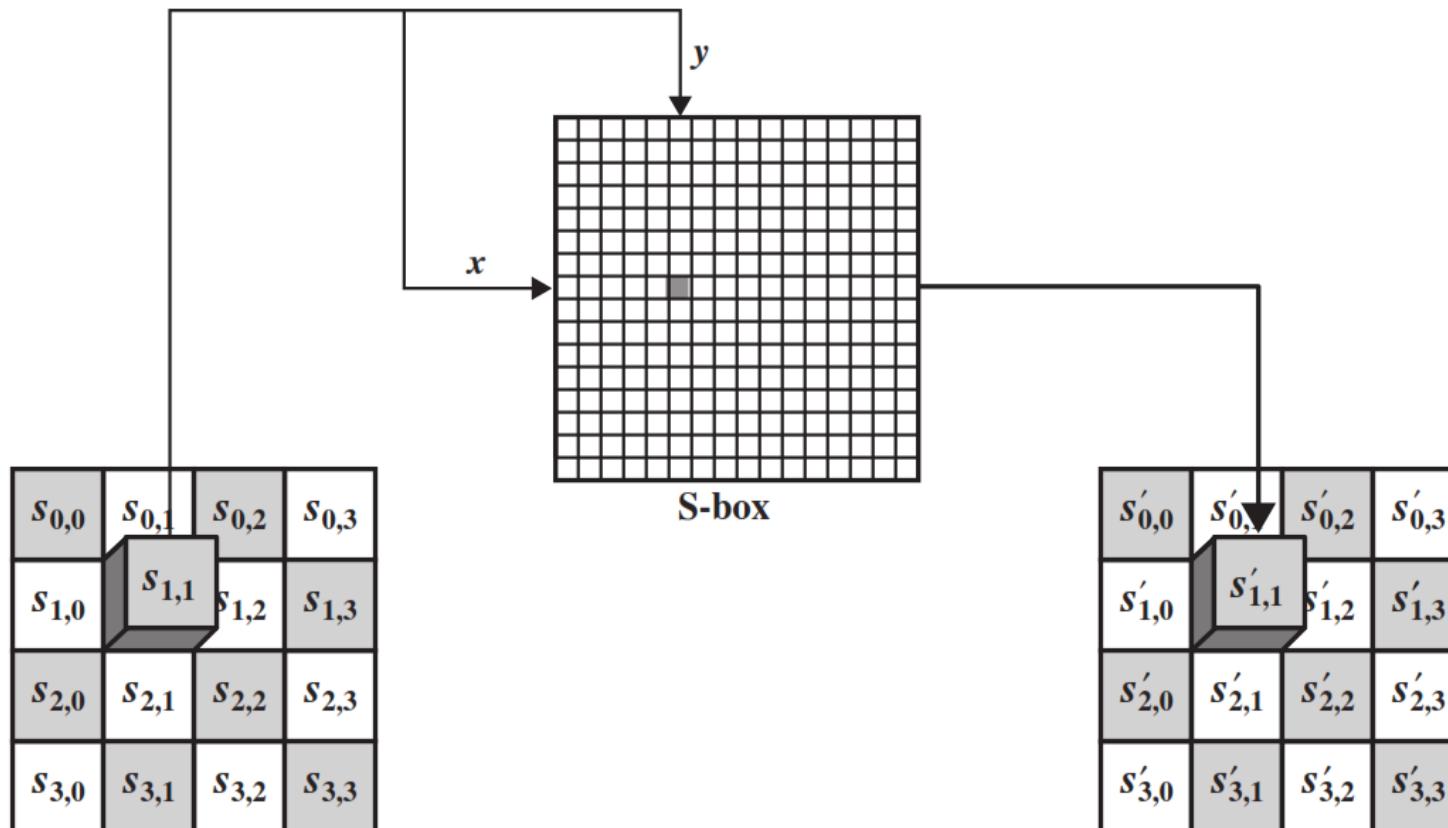
(b) Decryption

# AES Round



# SubByte Transformation

- The forward substitute byte transformation, called **SubBytes**, is a simple table lookup



# ShiftRows

- The first row of **State is not altered**.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6



87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

# MixColumns

- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.
- Mix Columns performs matrix multiplication according to [Galois field arithmetic](#)

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95



47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

# GF(2<sup>8</sup>)

---

- Finite Field/ Galois fields : A field with finite number of elements
- Finite fields play a key role in cryptography
- The finite field with  $p^n$  elements is denoted GF( $p^n$ ) and is also called the **Galois field** of order  $p^n$
- Rijndael (standardised as AES) uses the characteristic 2 finite field with 256 elements, which can also be called the Galois field GF(2<sup>8</sup>)
- Byte b7b6b5b4b3b2b1b0 will have the representation as  
$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$
- Therefore, 01010111 would have the representation as  
$$x^6 + x^4 + x^2 + x + 1$$

# Addition on Bytes

---

- The sum of two elements is the polynomial with coefficients that are given by the sum modulo 2 (i.e.,  $1+1=0$ ) of the coefficients of the two terms.
- Example:  $57+83=?$ 
  - $57 = x^6 + x^4 + x^2 + x + 1$
  - $83 = x^7 + x + 1$
  - $(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$
  - $x^7 + x^6 + x^4 + x^2 = D4$

# Multiplication

---

- Multiplication is performed using a special polynomial called the irreducible polynomial.
- The modulus used for these operations is typically a specific irreducible polynomial of degree 8, which ensures that the resulting values remain within the field.
- Example:  $57 \bullet 83 = ?$ 
  - $57 = x^6 + x^4 + x^2 + x + 1$
  - $83 = x^7 + x + 1$
  - $(x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1$   
AES uses arithmetic in the finite field  $GF(2^8)$  with irreducible (prime) polynomial which is  $x^8 + x^4 + x^3 + x + 1$  (11B)
    - $x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$  modulo  $x^8 + x^4 + x^3 + x + 1$
    - ....
    - $x^7 + x^6 + 1 = C1$

# AddRoundKey

- In the forward add round key transformation, the 128 bits of State are bitwise XORed with the 128 bits of the round key.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

State

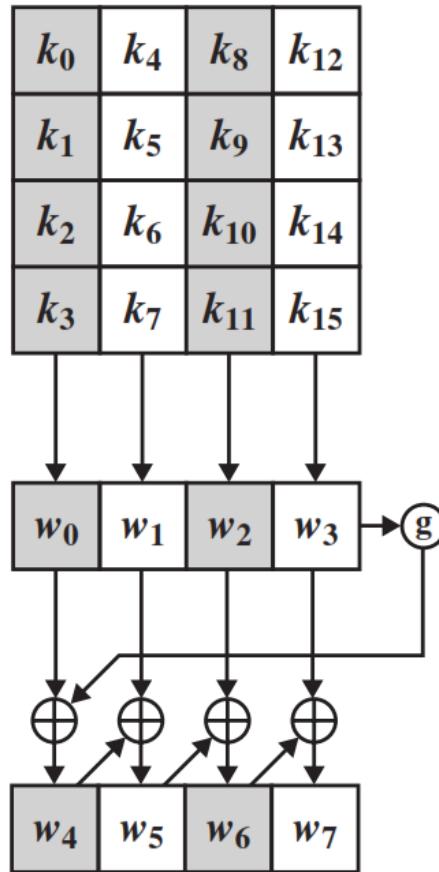
⊕

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

Round Key

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

=



# AES Key Expansion

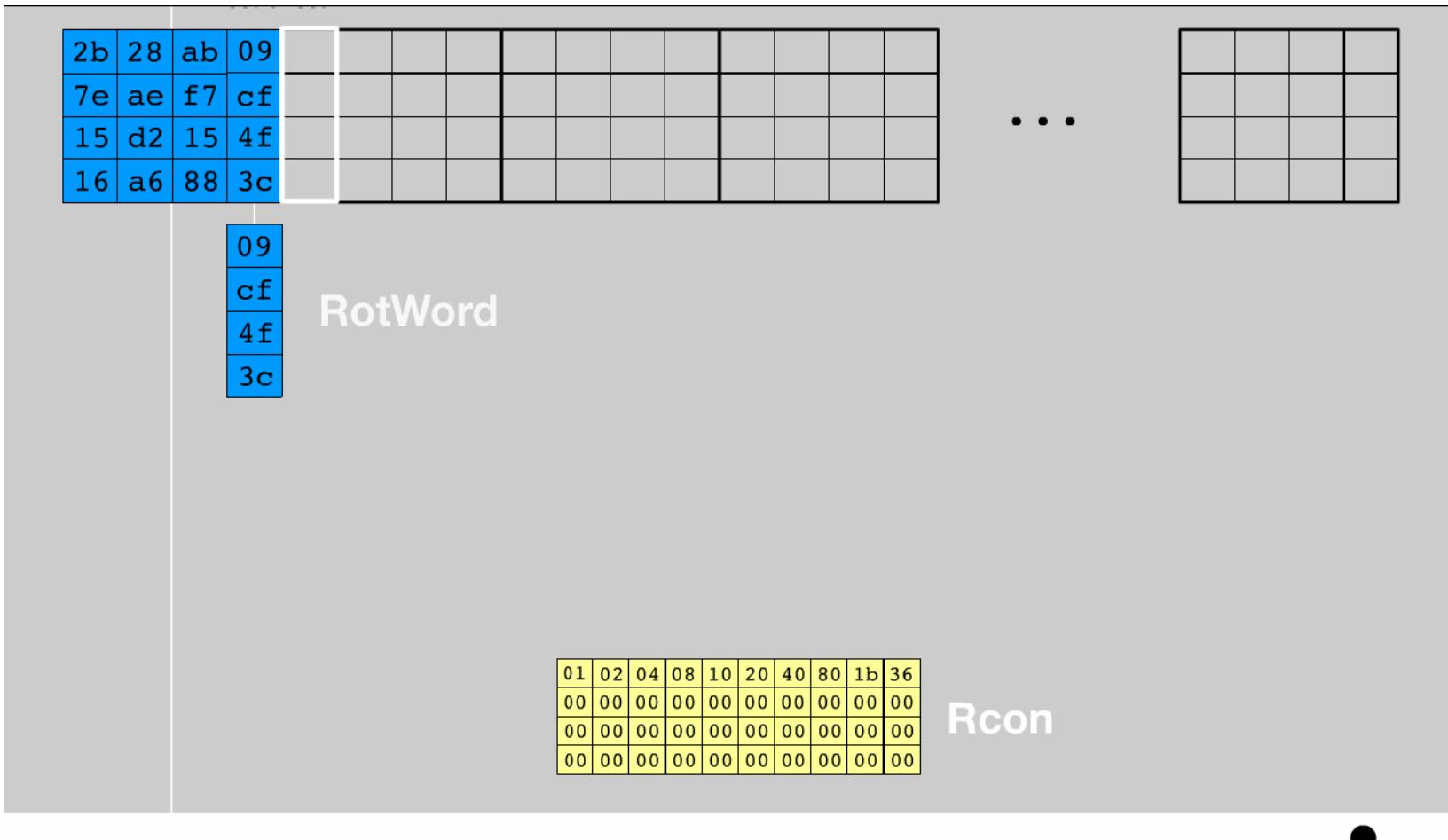
- The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of **44 words** (176 bytes).
- Each added word  $w[i]$  depends on the immediately preceding word,  $w[i - 1]$ .
- In three out of four cases, a simple XOR is used.

# Key Expansion Example

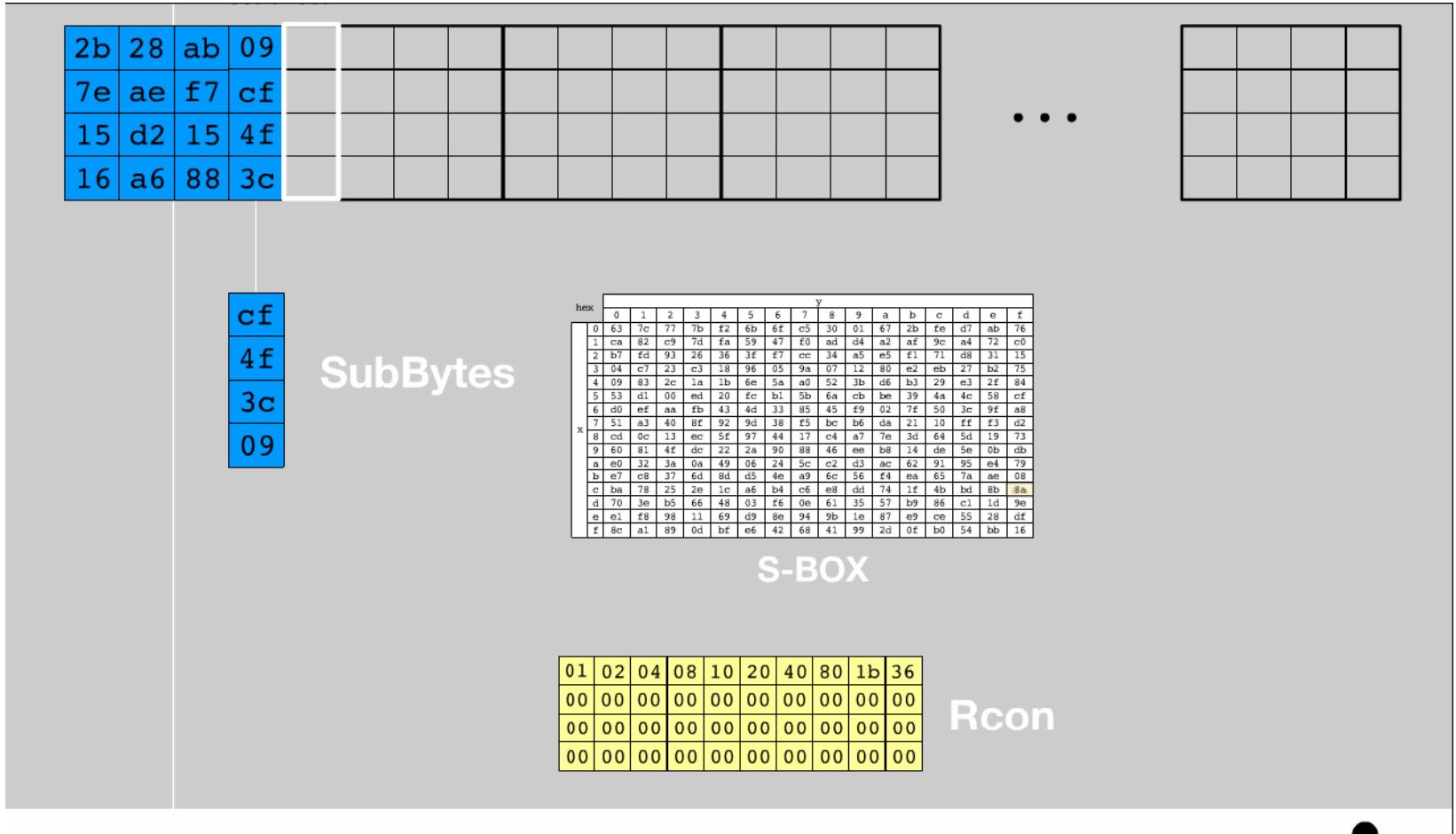
Plaintext:	0123456789abcdeffedcba9876543210
Key:	0f1571c947d9e8590cb7add6af7f6798
Ciphertext:	ff0b844a0853bf7c6934ab4364148fb9

Key Words	Auxiliary Function
$w_0 = 0f\ 15\ 71\ c9$ $w_1 = 47\ d9\ e8\ 59$ $w_2 = 0c\ b7\ ad\ d6$ $w_3 = af\ 7f\ 67\ 98$	$\text{RotWord}(w_3) = 7f\ 67\ 98\ af = x_1$ $\text{SubWord}(x_1) = d2\ 85\ 46\ 79 = y_1$ $\text{Rcon}(1) = 01\ 00\ 00\ 00$ $y_1 \oplus \text{Rcon}(1) = d3\ 85\ 46\ 79 = z_1$
$w_4 = w_0 \oplus z_1 = dc\ 90\ 37\ b0$ $w_5 = w_4 \oplus w_1 = 9b\ 49\ df\ e9$ $w_6 = w_5 \oplus w_2 = 97\ fe\ 72\ 3f$ $w_7 = w_6 \oplus w_3 = 38\ 81\ 15\ a7$	$\text{RotWord}(w_7) = 81\ 15\ a7\ 38 = x_2$ $\text{SubWord}(x_2) = 0c\ 59\ 5c\ 07 = y_2$ $\text{Rcon}(2) = 02\ 00\ 00\ 00$ $y_2 \oplus \text{Rcon}(2) = 0e\ 59\ 5c\ 07 = z_2$
$w_8 = w_4 \oplus z_2 = d2\ c9\ 6b\ b7$ $w_9 = w_8 \oplus w_5 = 49\ 80\ b4\ 5e$ $w_{10} = w_9 \oplus w_6 = de\ 7e\ c6\ 61$ $w_{11} = w_{10} \oplus w_7 = e6\ ff\ d3\ c6$	$\text{RotWord}(w_{11}) = ff\ d3\ c6\ e6 = x_3$ $\text{SubWord}(x_3) = 16\ 66\ b4\ 83 = y_3$ $\text{Rcon}(3) = 04\ 00\ 00\ 00$ $y_3 \oplus \text{Rcon}(3) = 12\ 66\ b4\ 8e = z_3$
$w_{12} = w_8 \oplus z_3 = c0\ af\ df\ 39$ $w_{13} = w_{12} \oplus w_9 = 89\ 2f\ 6b\ 67$ $w_{14} = w_{13} \oplus w_{10} = 57\ 51\ ad\ 06$ $w_{15} = w_{14} \oplus w_{11} = b1\ ae\ 7e\ c0$	$\text{RotWord}(w_{15}) = ae\ 7e\ c0\ b1 = x_4$ $\text{SubWord}(x_4) = e4\ f3\ ba\ c8 = y_4$ $\text{Rcon}(4) = 08\ 00\ 00\ 00$ $y_4 \oplus \text{Rcon}(4) = ec\ f3\ ba\ c8 = 4$

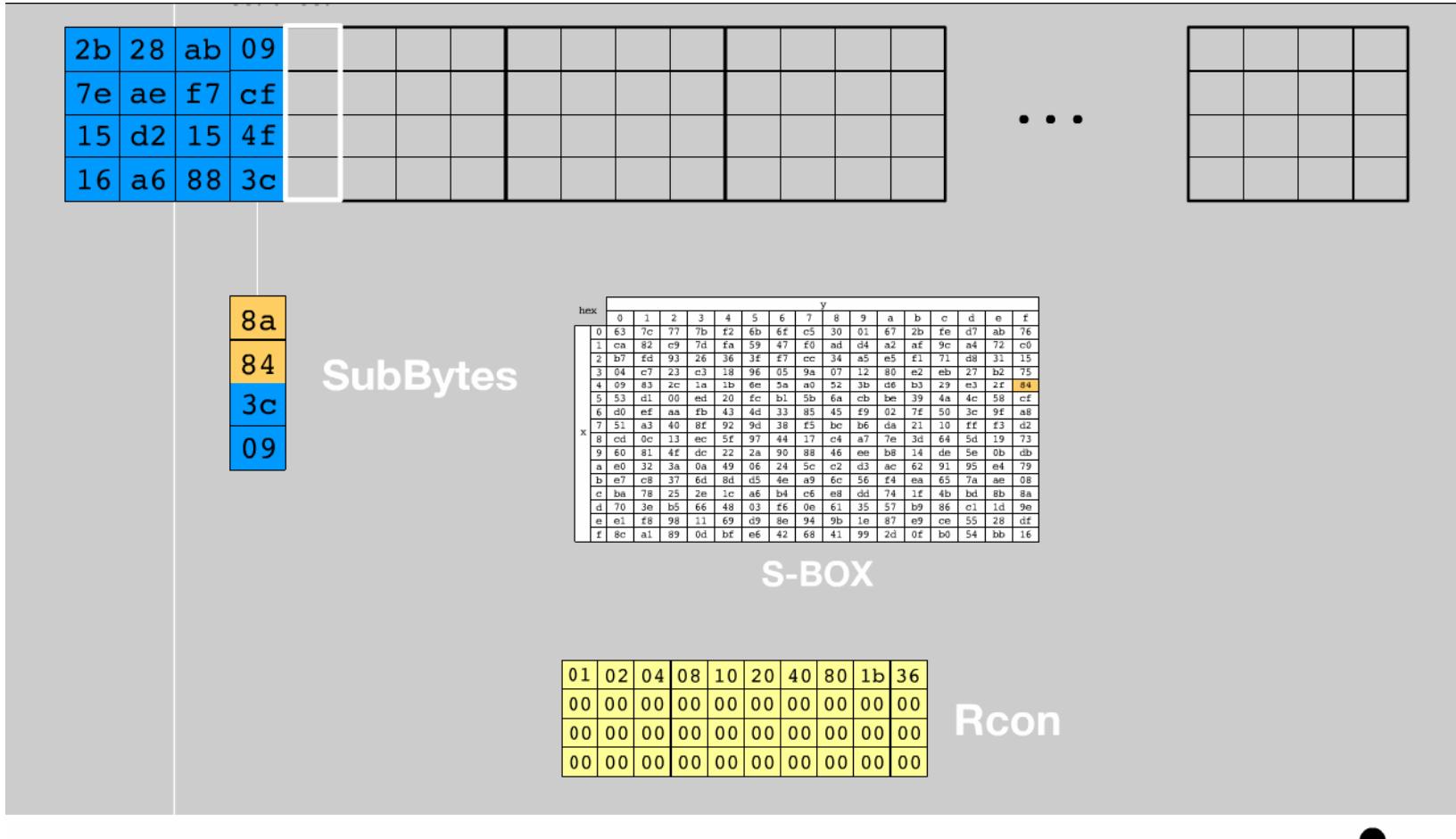
# Key Schedule Generation (For reference)



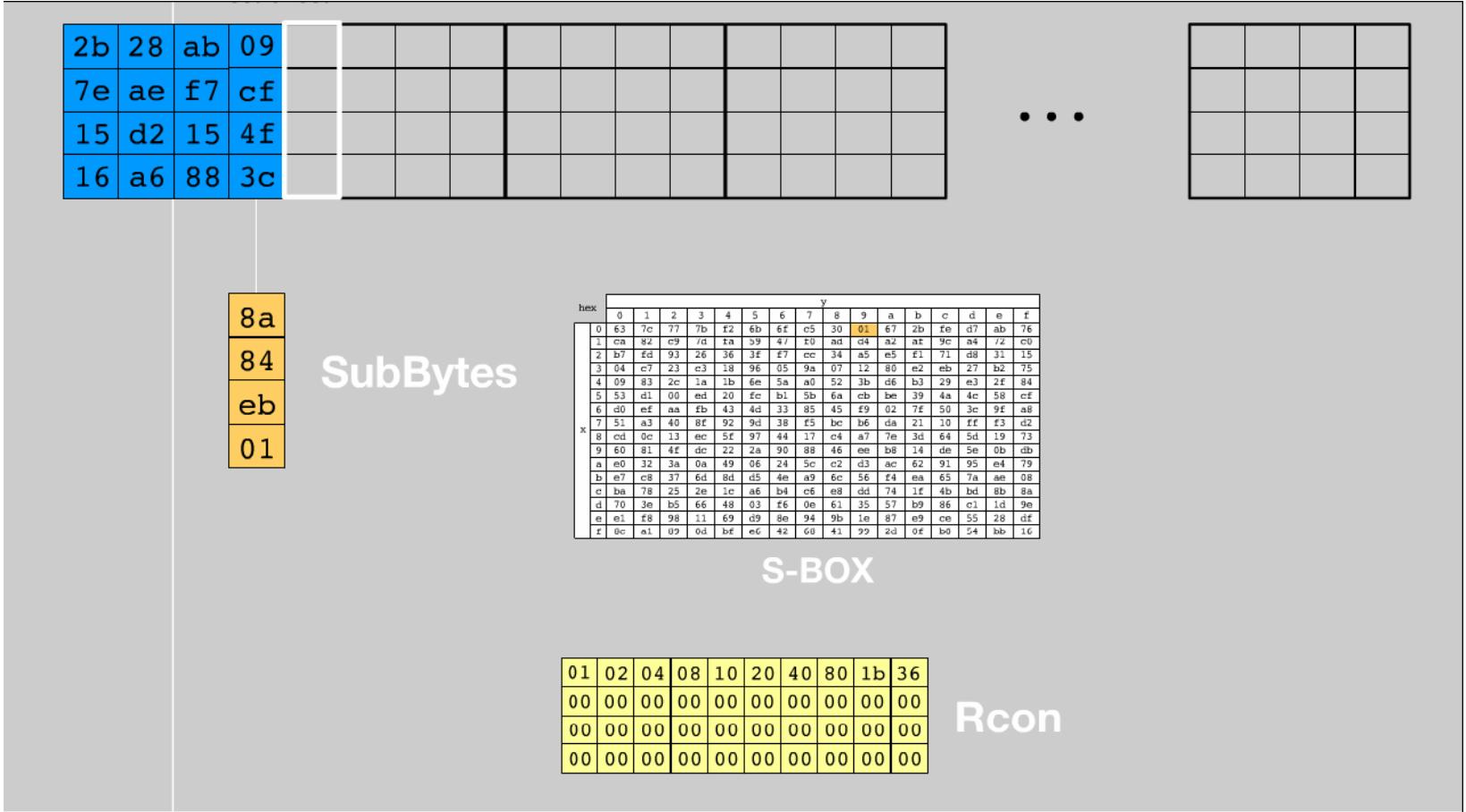
# Key Schedule Generation (Cont.)



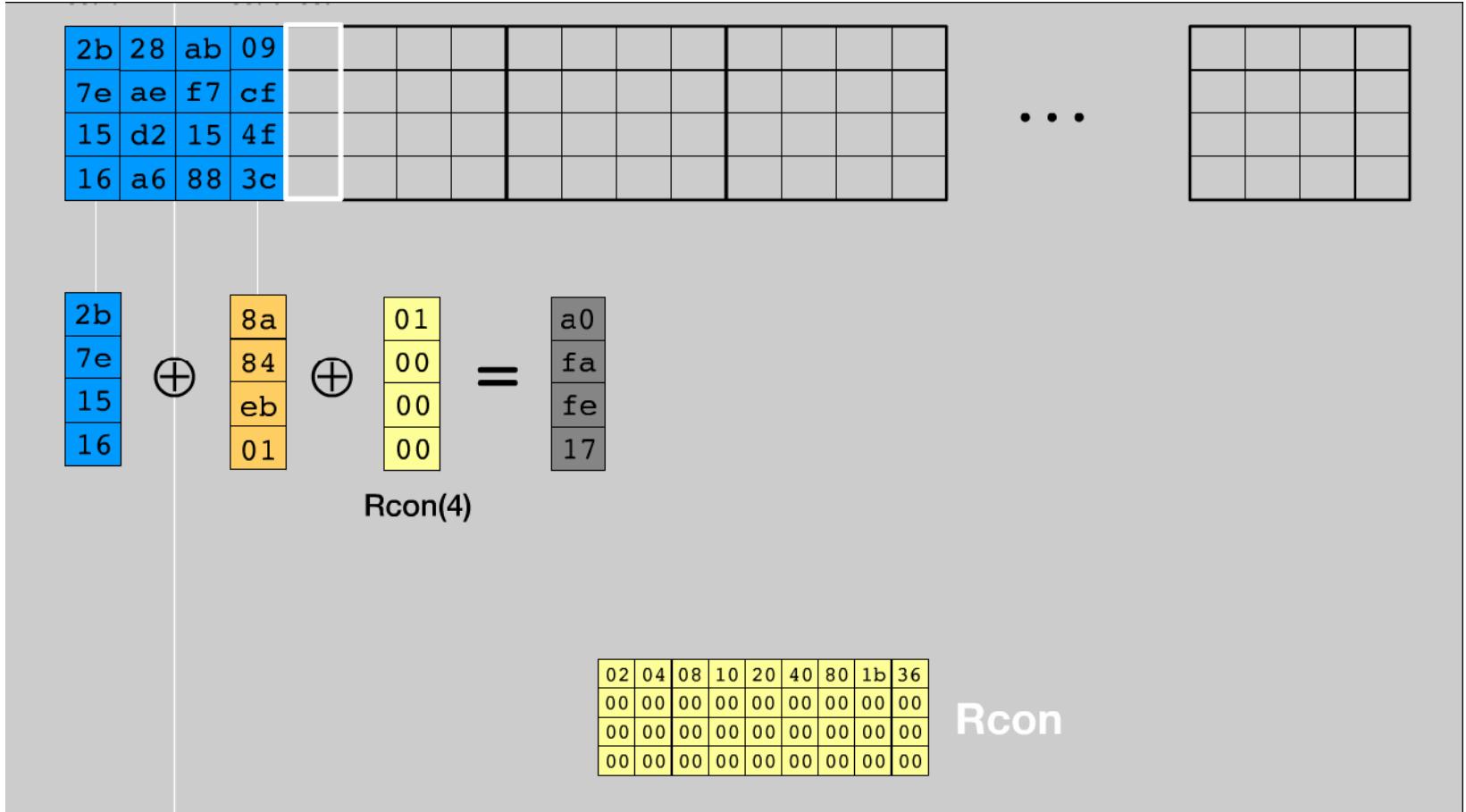
# Key Schedule Generation (Cont.)



# Key Schedule Generation (Cont.)



# Key Schedule Generation (Cont.)



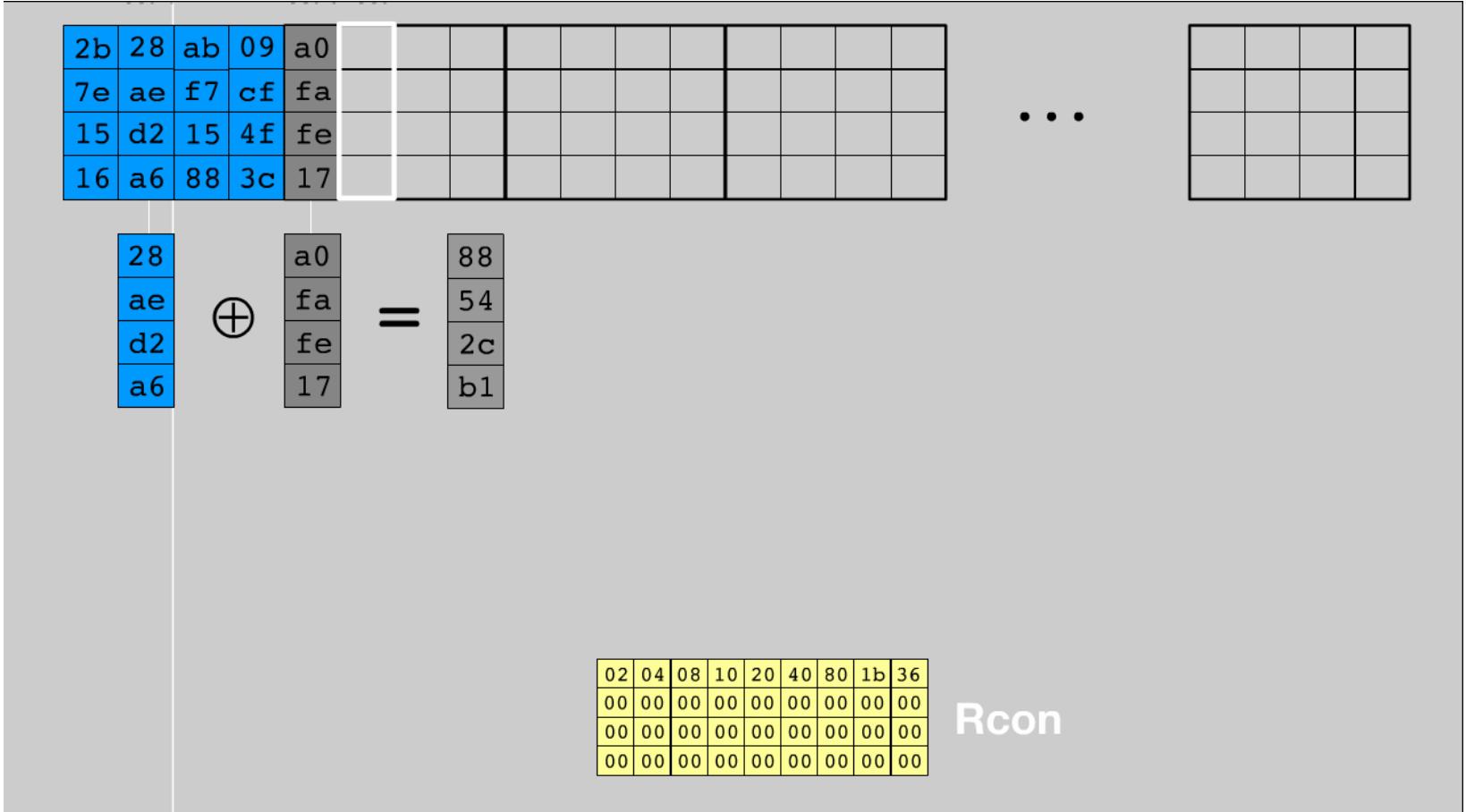
# Key Schedule Generation (Cont.)

• • •

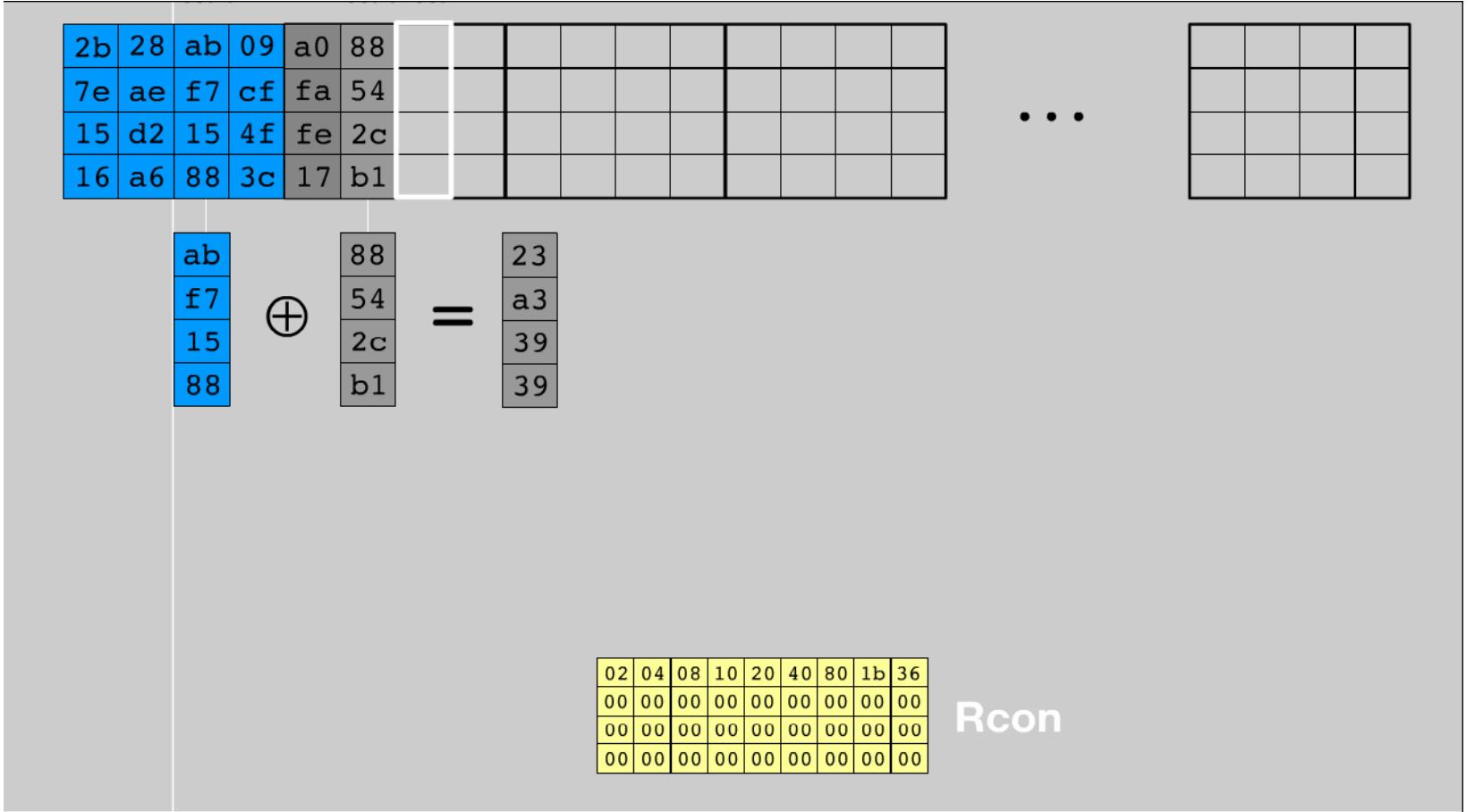
A 4x4 grid of squares, outlined by thick black lines, intended for children to draw a picture within.

Rcon

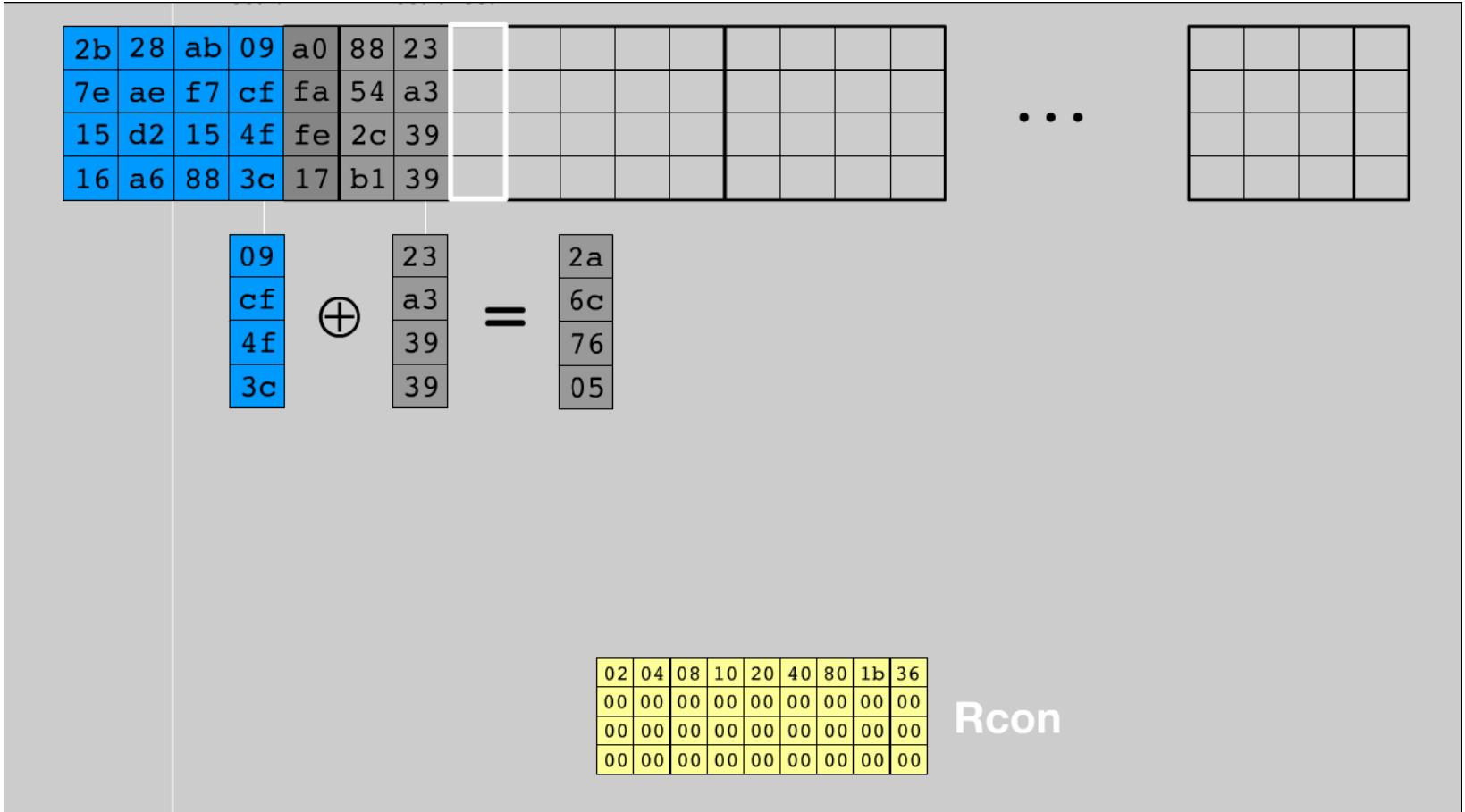
# Key Schedule Generation (Cont.)



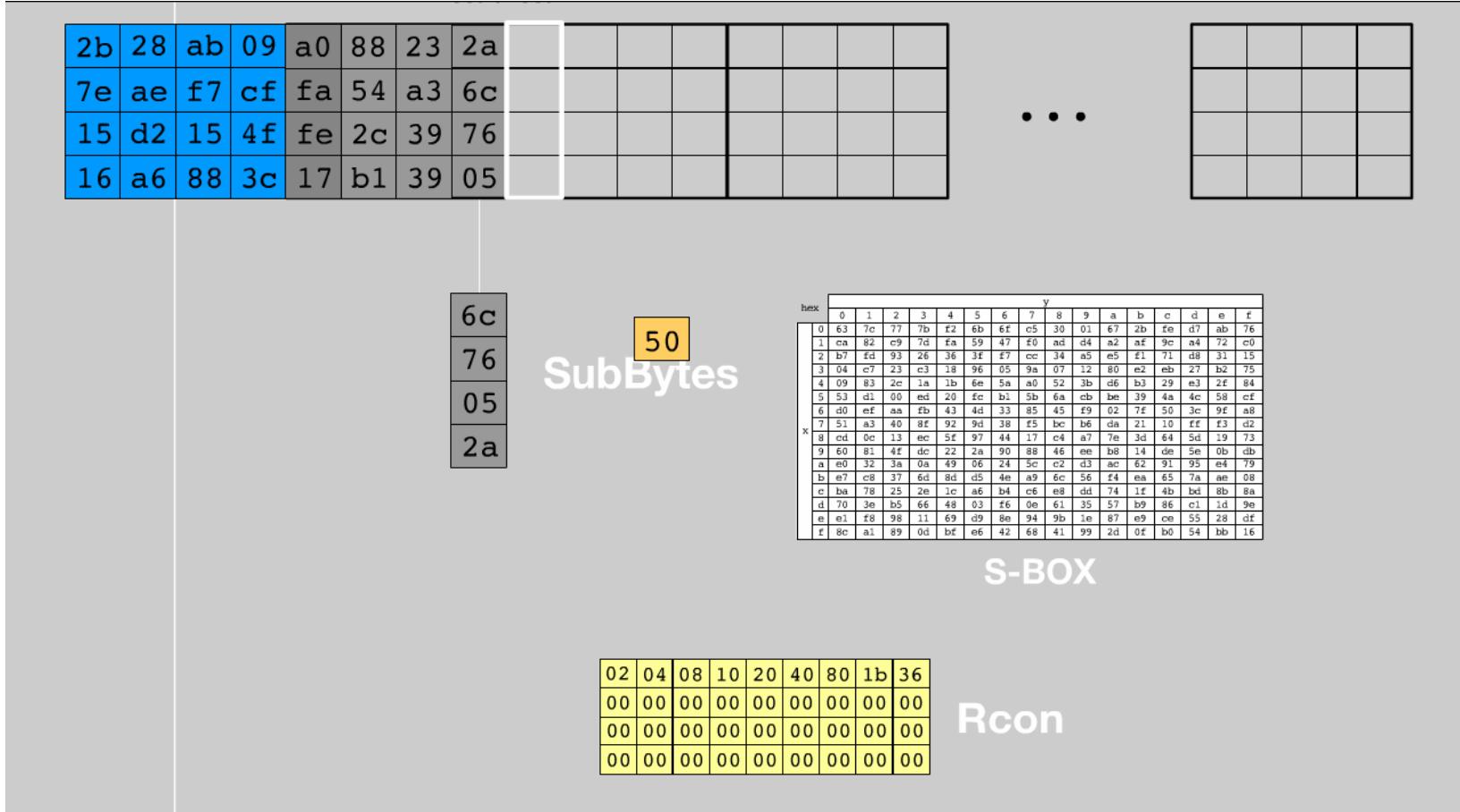
# Key Schedule Generation (Cont.)



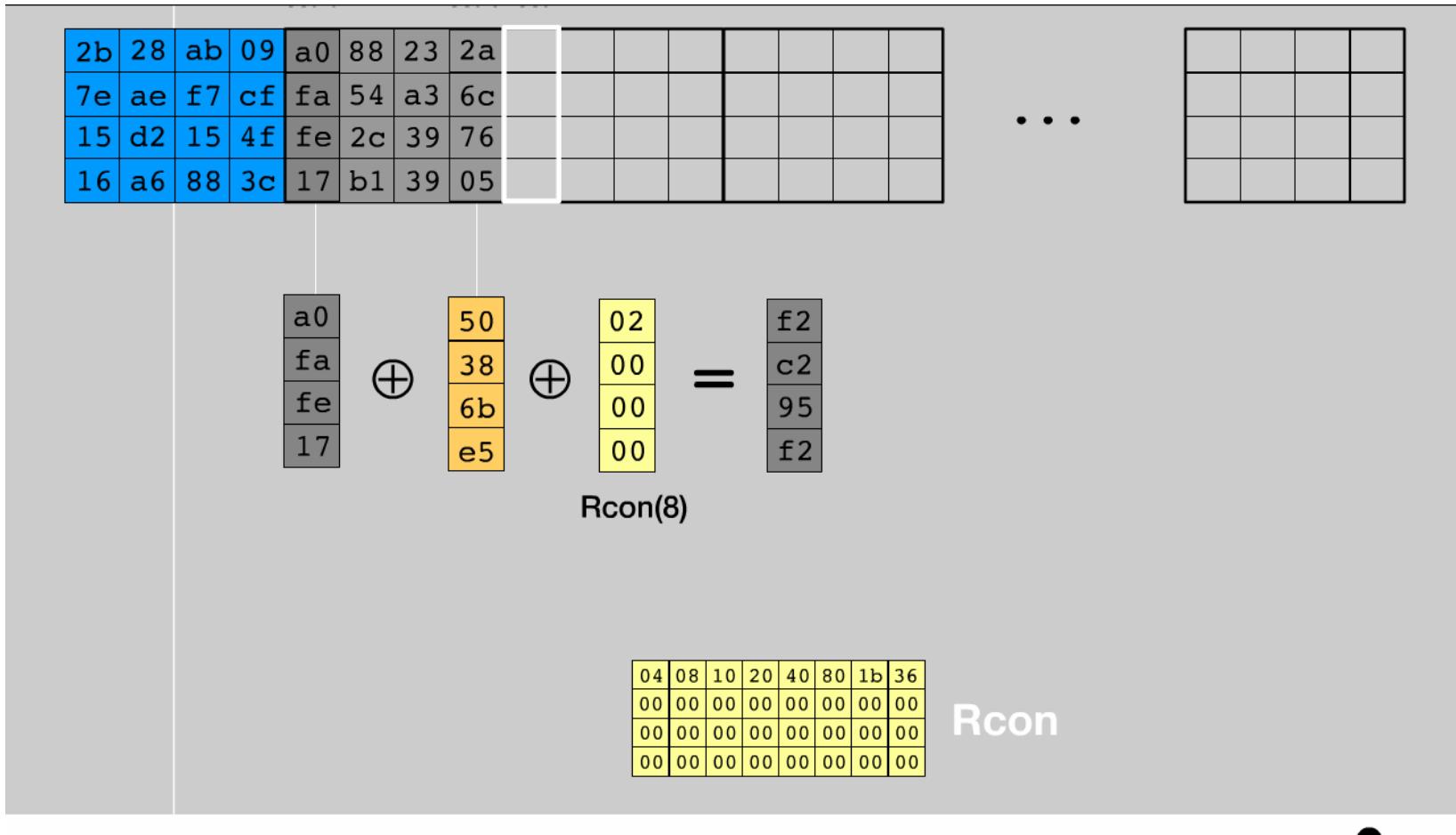
# Key Schedule Generation (Cont.)



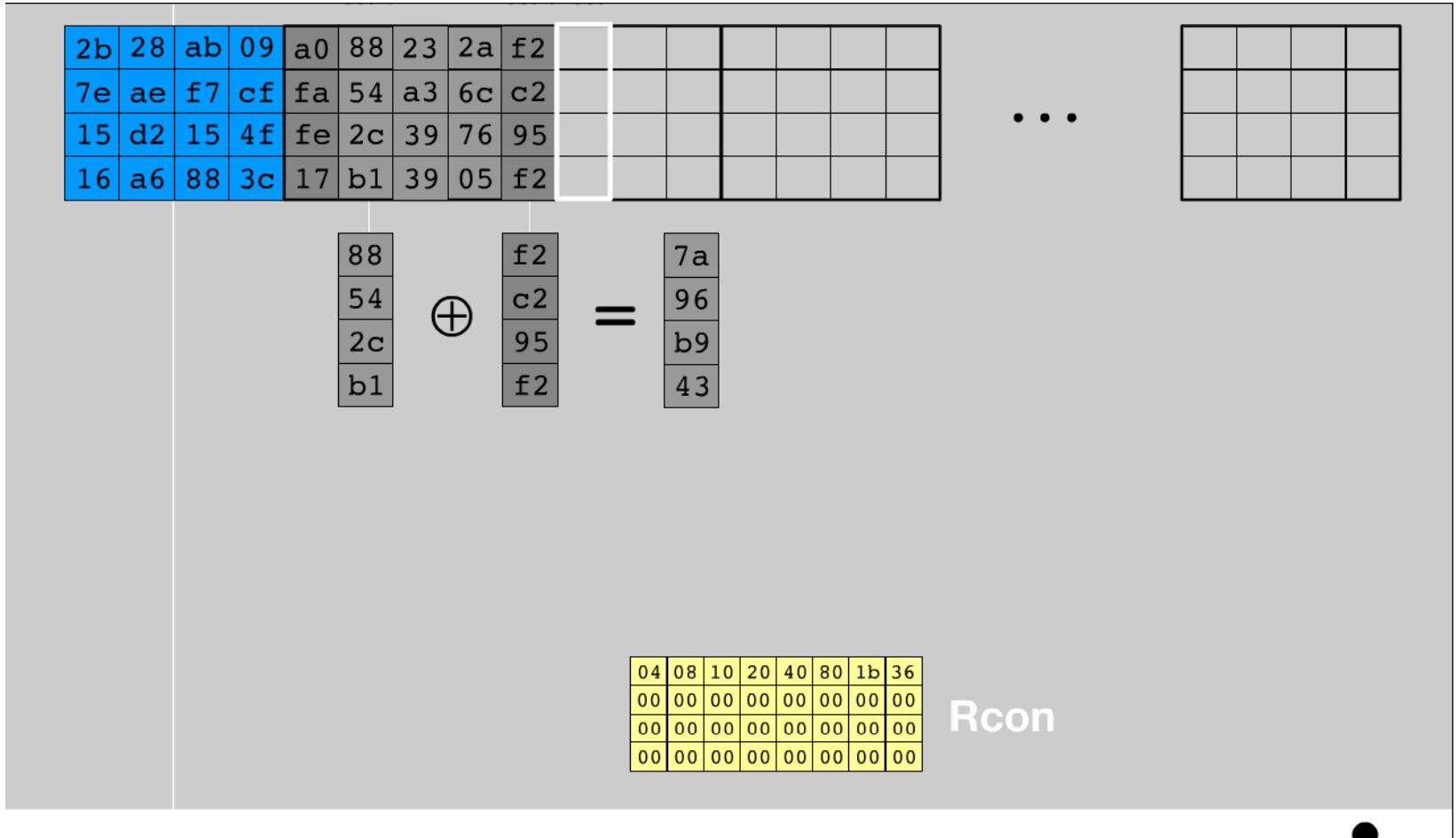
# Key Schedule Generation (Cont.)



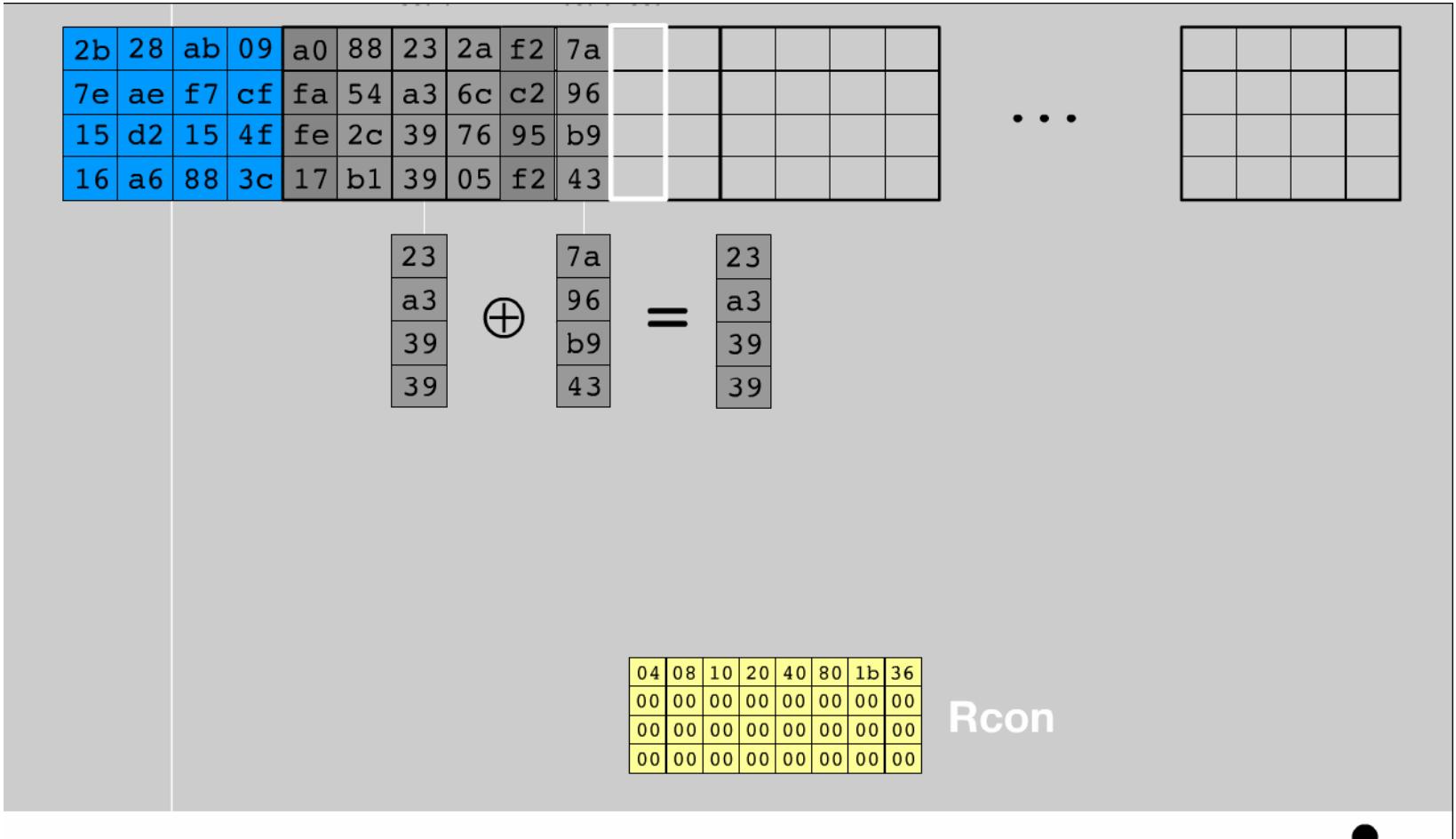
# Key Schedule Generation (Cont.)



# Key Schedule Generation (Cont.)



# Key Schedule Generation (Cont.)



# Key Schedule Generation (Cont.)

<table border="1"><tr><td>2b</td><td>28</td><td>ab</td><td>09</td><td>a0</td><td>88</td><td>23</td><td>2a</td><td>f2</td><td>7a</td><td>23</td><td>73</td><td>3d</td><td>47</td><td>1e</td><td>6d</td></tr><tr><td>7e</td><td>ae</td><td>f7</td><td>cf</td><td>fa</td><td>54</td><td>a3</td><td>6c</td><td>c2</td><td>96</td><td>a3</td><td>59</td><td>80</td><td>16</td><td>23</td><td>7a</td></tr><tr><td>15</td><td>d2</td><td>15</td><td>4f</td><td>fe</td><td>2c</td><td>39</td><td>76</td><td>95</td><td>b9</td><td>39</td><td>f6</td><td>47</td><td>fe</td><td>7e</td><td>88</td></tr><tr><td>16</td><td>a6</td><td>88</td><td>3c</td><td>17</td><td>b1</td><td>39</td><td>05</td><td>f2</td><td>43</td><td>39</td><td>7f</td><td>7d</td><td>3e</td><td>44</td><td>3b</td></tr></table>	2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d	7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88	16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b	<p>...</p>	<table border="1"><tr><td>d0</td><td>c9</td><td>e1</td><td>b6</td></tr><tr><td>14</td><td>ee</td><td>3f</td><td>63</td></tr><tr><td>f9</td><td>25</td><td>0c</td><td>0c</td></tr><tr><td>a8</td><td>89</td><td>c8</td><td>a6</td></tr></table>	d0	c9	e1	b6	14	ee	3f	63	f9	25	0c	0c	a8	89	c8	a6
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d																																																																			
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a																																																																			
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88																																																																			
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b																																																																			
d0	c9	e1	b6																																																																															
14	ee	3f	63																																																																															
f9	25	0c	0c																																																																															
a8	89	c8	a6																																																																															
<b>Cipher Key</b>	<b>Round key 1</b>	<b>Round key 2</b>	<b>Round key 3</b>		<b>Round key 10</b>																																																																													

# Unit 2

## Modes of Operations, Multiple encryptions and triple DES



# Outline

---

- Block cipher modes of operations
  - Electronic Code Book Mode
  - Cipher Block Chaining Mode
  - Cipher Feedback Mode
  - Output Feedback Mode
  - Counter Mode
- Multiple encryptions
- Triple DES

# Block Cipher Modes of Operations

---

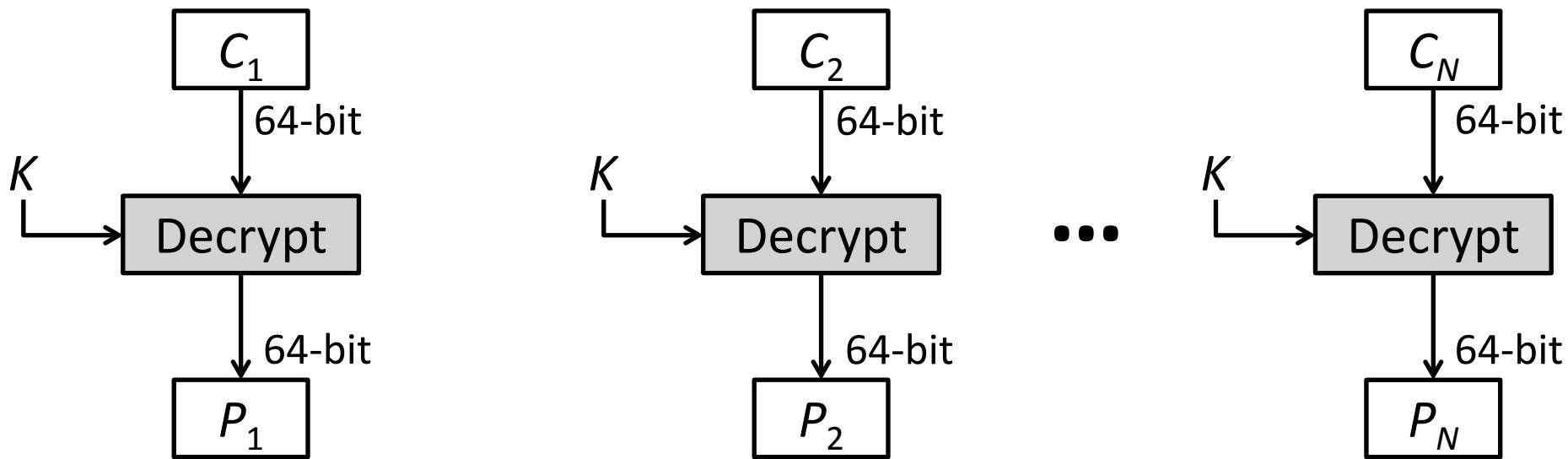
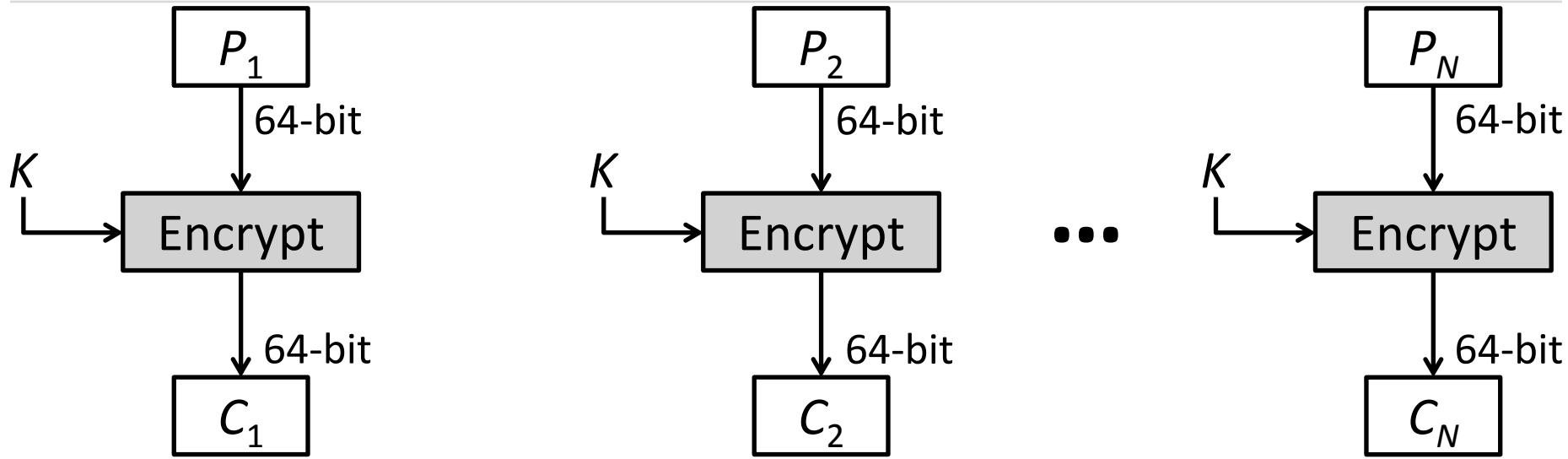
- To apply a block cipher in a **variety of applications**, five "modes of operation" have been defined.
- The **five modes** are intended to cover a wide variety of applications of encryption for which a block cipher could be used.
- These modes are intended for use with any symmetric block cipher, including triple DES and AES.
  1. Electronic Code Book (ECB)
  2. Cipher Block Chaining (CBC)
  3. Cipher Feedback (CFB)
  4. Output Feedback (OFB)
  5. Counter (CTR)

# 1. Electronic Code Book (ECB)

---

- In **ECB** Mode Plaintext is handled one block at a time and each block of plaintext is encrypted using the same key.
- The term **codebook** is used because, for a given key, there is a unique ciphertext for every block of plaintext.

# 1. ECB Encryption & Decryption



# Electronic Code Book - Cont...

---

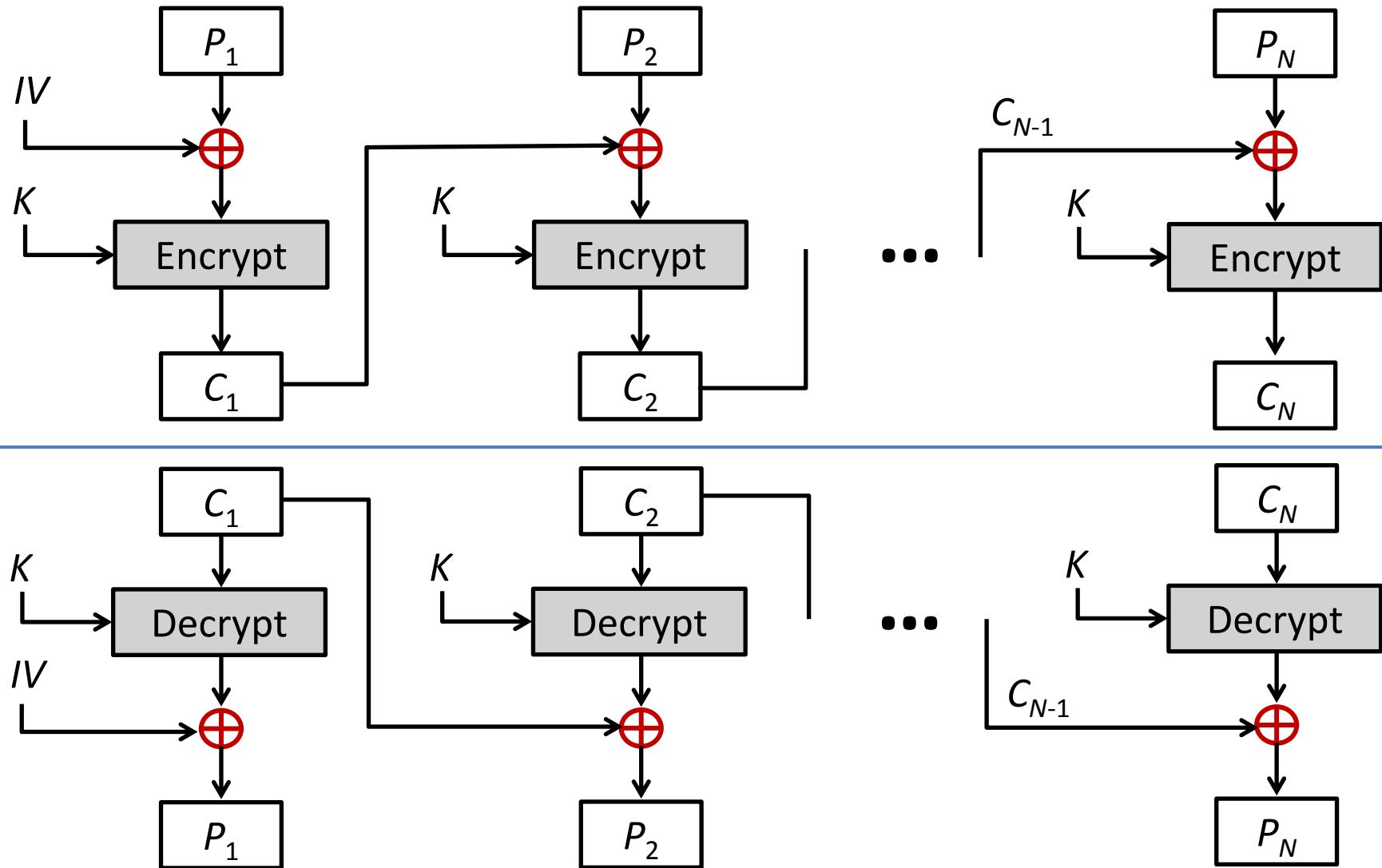
- **Strength:** it's simple.
- **Weakness:**
  - Repetitive information contained in the plaintext may show in the ciphertext also.
  - If the message has repetitive elements with a period of repetition a multiple of b bits, then these elements can be identified by the analyst.
- **Typical application:**
  - Secure transmission of short pieces of information (e.g. a temporary encryption key)

# 2. Cipher Block Chaining (CBC)

---

- **CBC** is a technique in which the same plaintext block, if repeated, produces different ciphertext blocks.
- In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.
- To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext.
- On decryption, the **IV** is XORed with the output of the decryption algorithm to recover the first block of plaintext.

# 2. CBC - Encryption & Decryption



CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \dots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$
-----	--	--

## 2. Cipher Block Chaining (CBC) – Cont...

---

- **Strength:** because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than  $b$  bits
- **Typical application:**
  - General-purpose block oriented transmission
  - Authentication

Cons:

No Parallelism

Pros:

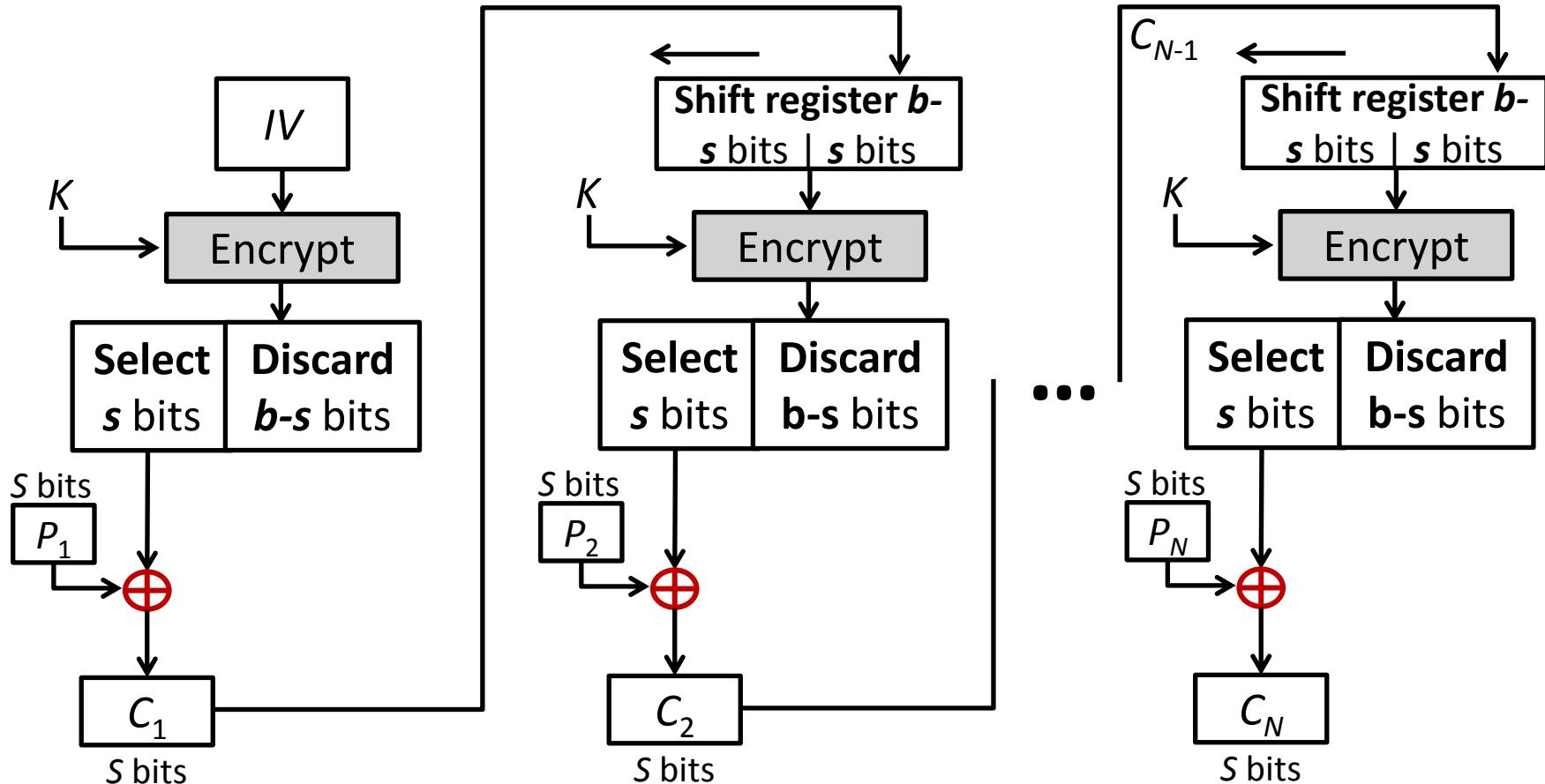
Confidentiality + Availability

# 3. Cipher Feedback Mode (CFB)

---

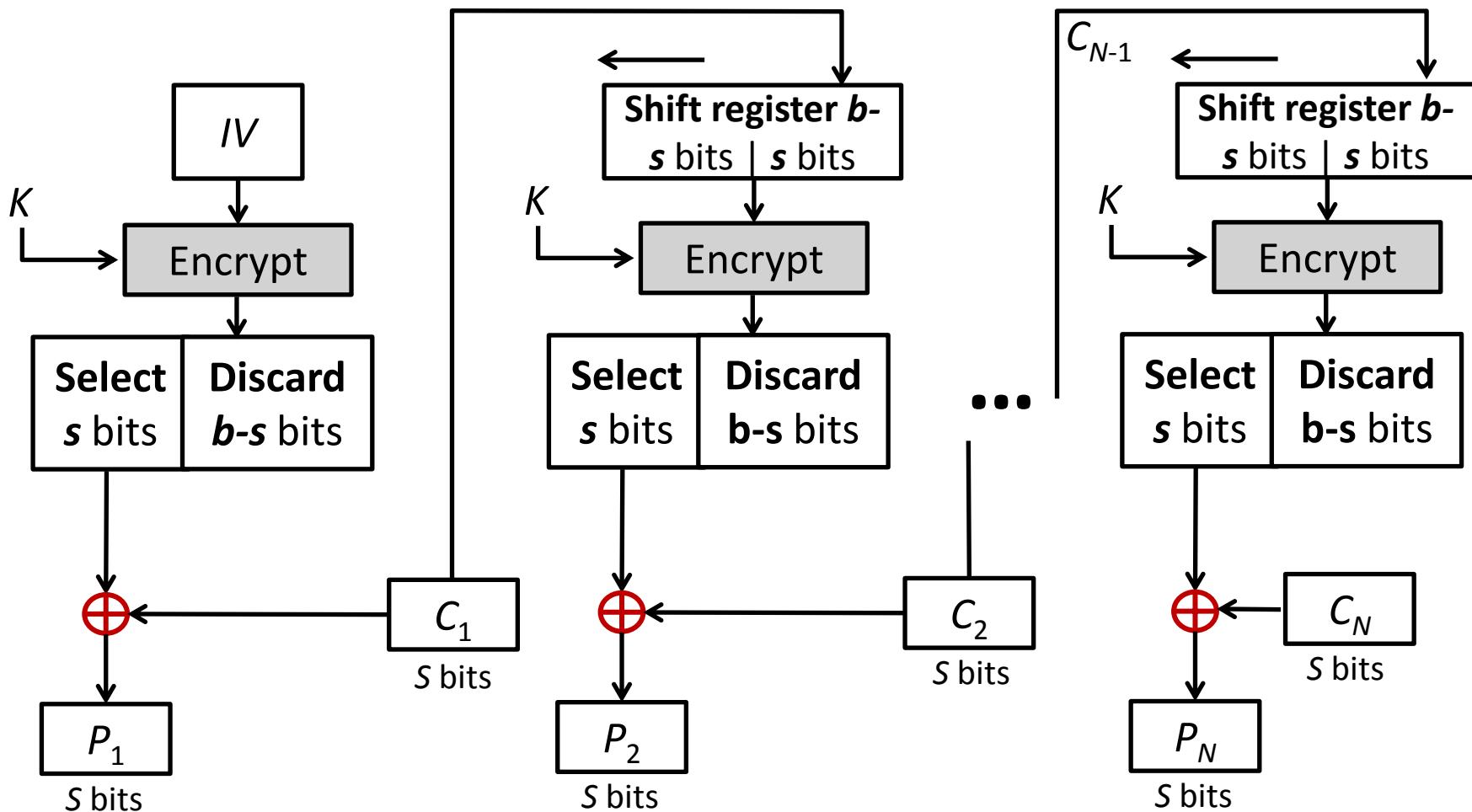
- For AES, DES, or any block cipher, encryption is performed on a block of  $b$  bits. In DES,  $b = 64$  and in AES,  $b = 128$ .
- However, it is possible to convert a block cipher into a stream cipher, using cipher feedback (CFB) mode, output feedback (OFB) mode, and counter (CTR) mode.
- A stream cipher eliminates the need to pad a message to be an integral number of blocks.

# 3. CFB Encryption



CFB	$I_1 = IV$
	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$
	$O_j = E(K, I_j) \quad j = 1, \dots, N$
	$C_j = P_j \oplus \text{MSB}_s(O_j) \quad j = 1, \dots, N$

## 3. CFB Decryption



$$I_1 = IV$$

$$I_j = \text{LSB}_{b-s}(I_{j-1}) \| C_{j-1} \quad j = 2, \dots, N$$

$$O_j = \mathbf{E}(K, I_j) \quad j = 1, \dots, N$$

$$P_j = C_j \oplus \text{MSB}_s(O_j) \quad j = 1, \dots, N$$

# CFB Mode

---

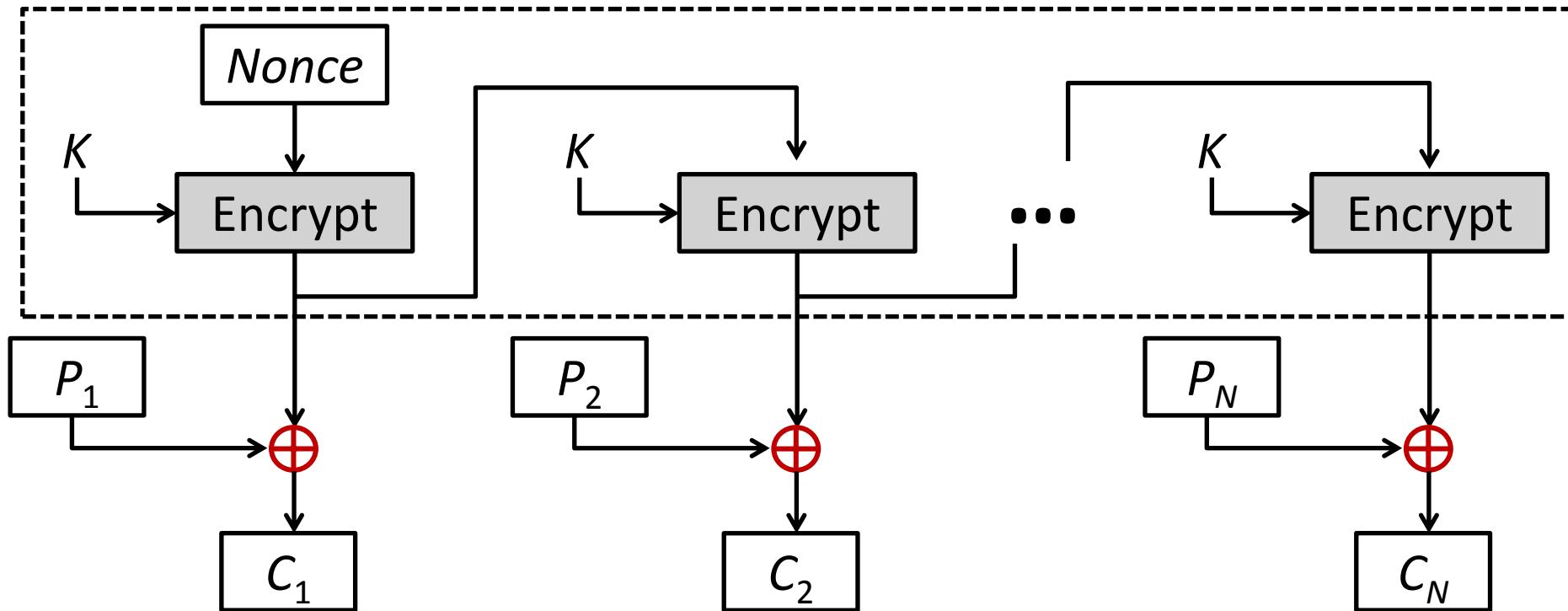
- The input to the encryption function is a **b-bit shift register** that is initially set to some initialization vector (IV).
- The leftmost (most significant) **s bits** of the output of the encryption function are XORed with the first segment of plaintext **P<sub>1</sub>** to produce the first unit of ciphertext **C<sub>1</sub>**, which is then transmitted.
- In addition, the contents of the shift register are shifted left by s bits, and C<sub>1</sub> is placed in the rightmost (**least significant**) s bits of the shift register.
- For **decryption**, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit.

# 4. Output Feedback Mode (OFB)

---

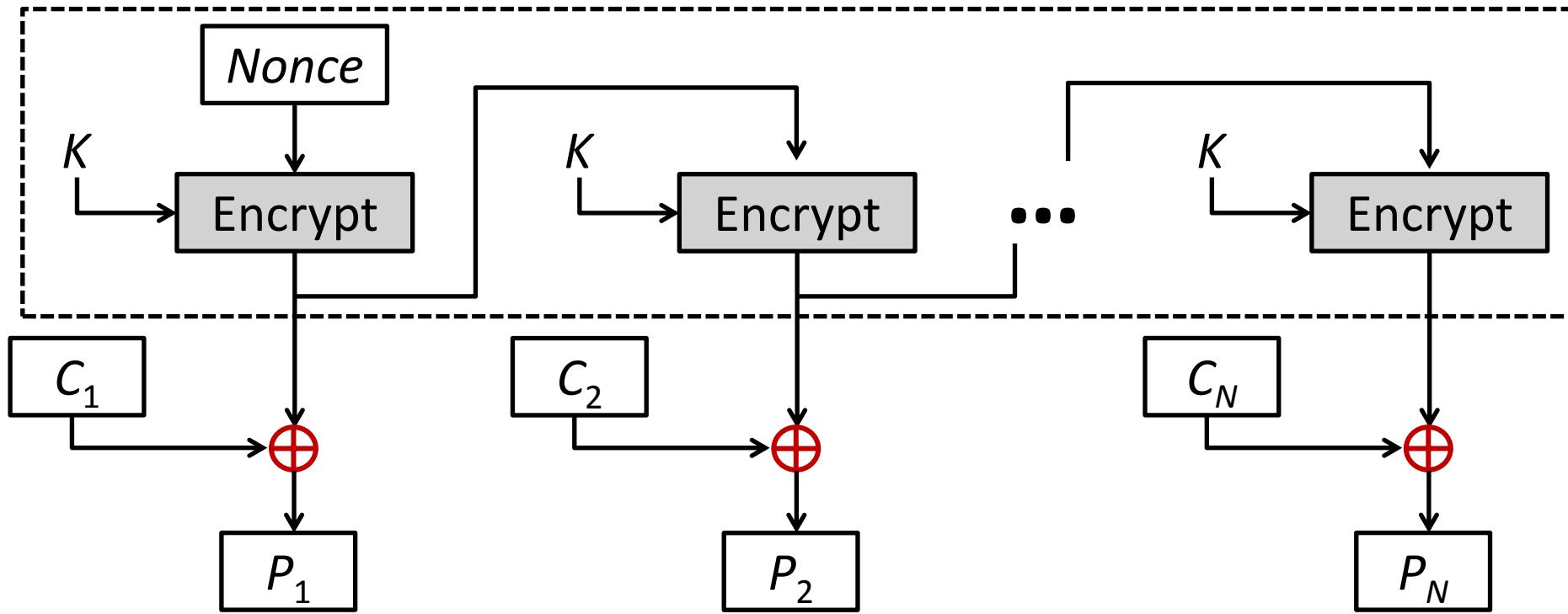
- The output feedback (**OFB**) mode is similar in structure to that of **CFB**.
- For **OFB**, the output of the encryption function is fed back to become the input for encrypting the next block of plaintext.
- In **CFB**, the output of the XOR unit is fed back to become input for encrypting the next block.
- The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, whereas **CFB** operates on an s-bit subset.
- **Nonce:** A time-varying value that has at most a negligible chance of repeating, for example, a **random value** that is freshly generated for each use, a timestamp, a sequence number, or some combination of these.

# 4. OFB Encryption



OFB	$I_1 = \text{Nonce}$ $I_j = O_{j-1} \quad j = 2, \dots, N$ $O_j = E(K, I_j) \quad j = 1, \dots, N$ $C_j = P_j \oplus O_j \quad j = 1, \dots, N - 1$ $C_N^* = P_N^* \oplus \text{MSB}_u(O_N)$
-----	--

# 4. OFB Decryption



$$I_1 = \text{Nonce}$$

$$I_j = O_{j-1} \quad j = 2, \dots, N$$

$$O_j = E(K, I_j) \quad j = 1, \dots, N$$

$$P_j = C_j \oplus O_j \quad j = 1, \dots, N - 1$$

$$P_N^* = C_N^* \oplus \text{MSB}_u(O_N)$$

# OFB Mode

---

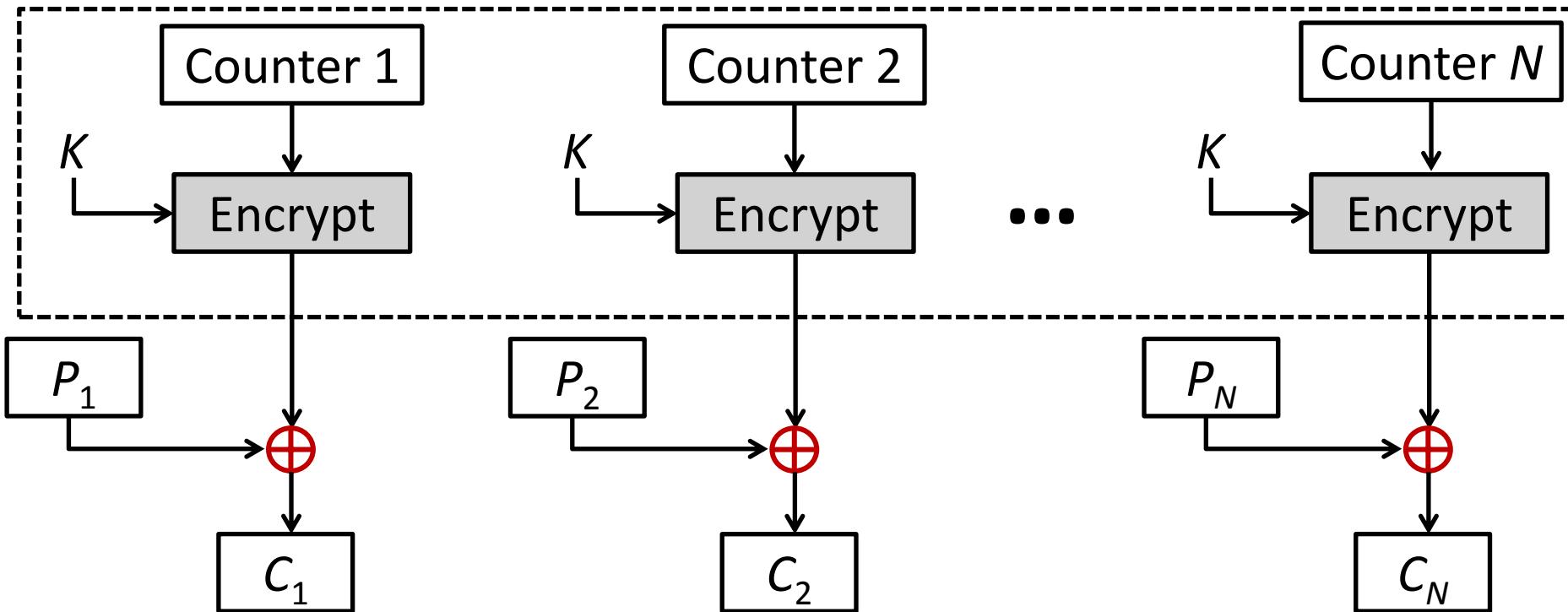
- Each bit in the ciphertext is independent of the previous bit or bits. i.e. Feedback is independent of transmission
- This **avoids error propagation which**
- It allows many error correcting codes to function normally even when applied before encryption
- It helps recover from ciphertext bit errors, but cannot self-synchronize
- Pre-compute of forward cipher is possible
- Flipping a bit in the ciphertext produces a flipped bit in the plaintext at the same location

# 5. Counter Mode (CTR)

---

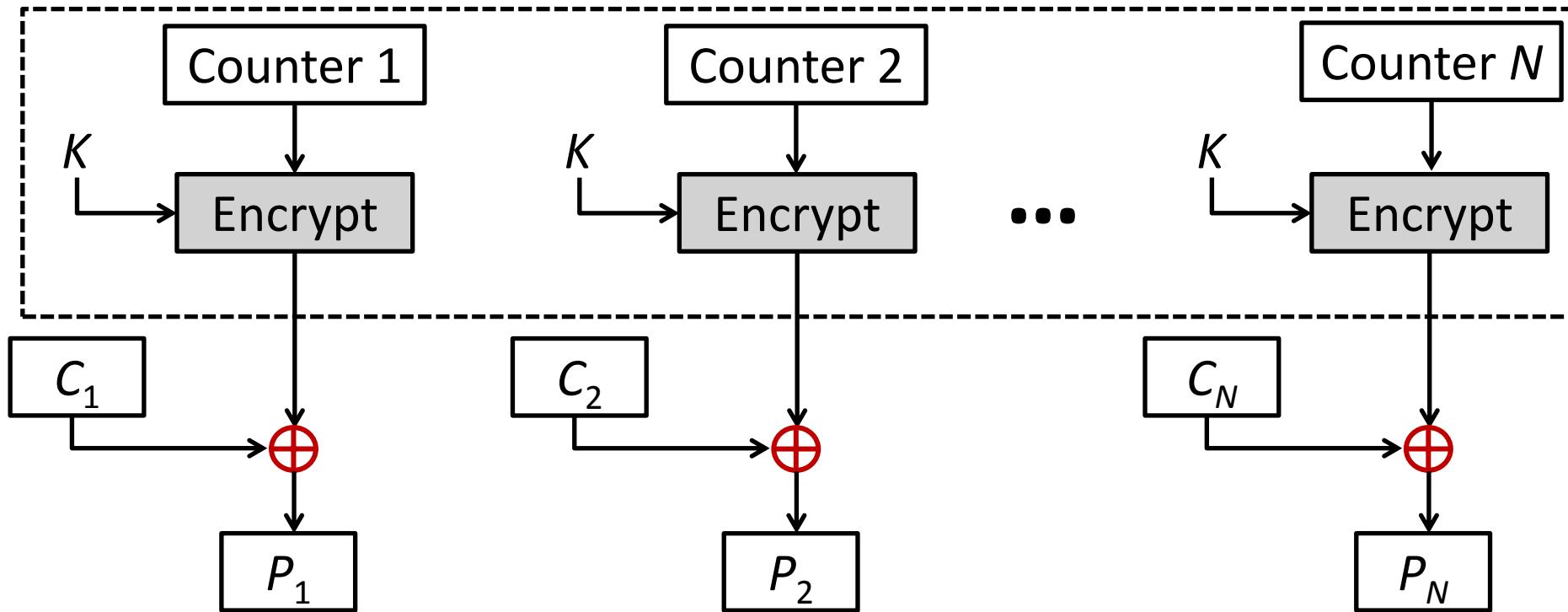
- Counter (**CTR**) mode has increased recently with applications to ATM (asynchronous transfer mode) network security and IP sec (IP security).
- A **counter** equal to the plaintext block size is used.
- The counter value must be different for each plaintext block that is encrypted.
- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block

# 5. CTR Encryption



CTR	$C_j = P_j \oplus E(K, T_j) \quad j = 1, \dots, N - 1$ $C_N^* = P_N^* \oplus \text{MSB}_u[E(K, T_N)]$
-----	---

# 4. CTR Decryption



$$P_j = C_j \oplus E(K, T_j) \quad j = 1, \dots, N - 1$$

$$P_N^* = C_N^* \oplus \text{MSB}_u[E(K, T_N)]$$

# Advantages of the CTR Mode

---

- Strengths:
  - Needs only the encryption algorithm
  - Random access to encrypted data blocks
  - blocks can be processed (encrypted or decrypted) in parallel
  - Simple; fast encryption/decryption
  
- Counter
  - Must be unknown and unpredictable
  - pseudo-randomness in the key stream is a goal

# Summary of All Modes

Operation Mode	Description	Type of Result
ECB	Each n-bit block is encrypted independently with same key	Block Cipher
CBC	Same as ECB, but each block is XORed with previous cipher text	Block Cipher
CFB	Each s-bit block is XORed with s-bit key which is part of previous cipher text	Stream Cipher
OFB	Same as CFB, except that the input to the encryption algorithm is the output of preceding encryption algorithm	Stream Cipher
CTR	Same as OFB, but a counter is used instead of nonce	Stream Cipher

# Typical Applications of All Modes

Mode	Typical Application
Electronic Codebook (ECB)	<ul style="list-style-type: none"><li>Secure transmission of single values (e.g., an encryption key)</li></ul>
Cipher Block Chaining (CBC)	<ul style="list-style-type: none"><li>General-purpose block-oriented transmission</li><li>Authentication</li></ul>
Cipher Feedback (CFB)	<ul style="list-style-type: none"><li>General-purpose stream-oriented transmission</li><li>Authentication</li></ul>
Output Feedback (OFB)	<ul style="list-style-type: none"><li>Stream-oriented transmission over noisy channel (e.g., satellite communication)</li></ul>
Counter (CTR)	<ul style="list-style-type: none"><li>General-purpose transmission</li><li>Useful for high-speed requirements</li></ul>

# Use of Stream Ciphers

---

- Although the five modes of operations enable the use of block ciphers for encipherment of messages or files in large units (ECB, CBC, and CTR) and small units (CFB and OFB), sometimes pure streams are needed for enciphering small units of data such as characters or bits.
- Stream ciphers are more efficient for real-time processing.
- Several stream ciphers have been used in different protocols during the last few decades.
- Examples: RC4, A5/1

# RC4

---

- RC4 means Rivest Cipher 4 invented by Ron Rivest in 1987 for RSA Security. It is a Stream Ciphers.
- Stream Ciphers operate on a stream of data byte by byte.
- RC4 stream cipher is one of the most widely used stream ciphers because of its simplicity and speed of operation.
- It is a variable key-size stream cipher with byte-oriented operations. It uses either 64 bit or 128-bit key sizes.
- It is generally used in applications such as Secure Socket Layer (SSL), Transport Layer Security (TLS), and also used in IEEE 802.11 wireless LAN std.

# A5/1

---

- A5/1 is a stream cipher used to provide over-the-air communication privacy in the GSM cellular telephone standard.
- It is one of several implementations of the A5 security protocol.

# Synchronous & Asynchronous Stream Ciphers

---

- With **synchronous stream** ciphers, the bits of the key stream do not depend on the ciphertext bits.
- With **asynchronous stream** ciphers the key stream can be inferred (provided one knows the secret key) from previous bits of the cipher stream.
- CTR would be an example of a synchronous streaming mode and CFB would be an example of an asynchronous mode.

# Synchronous & Asynchronous stream ciphers (Cont.)

---

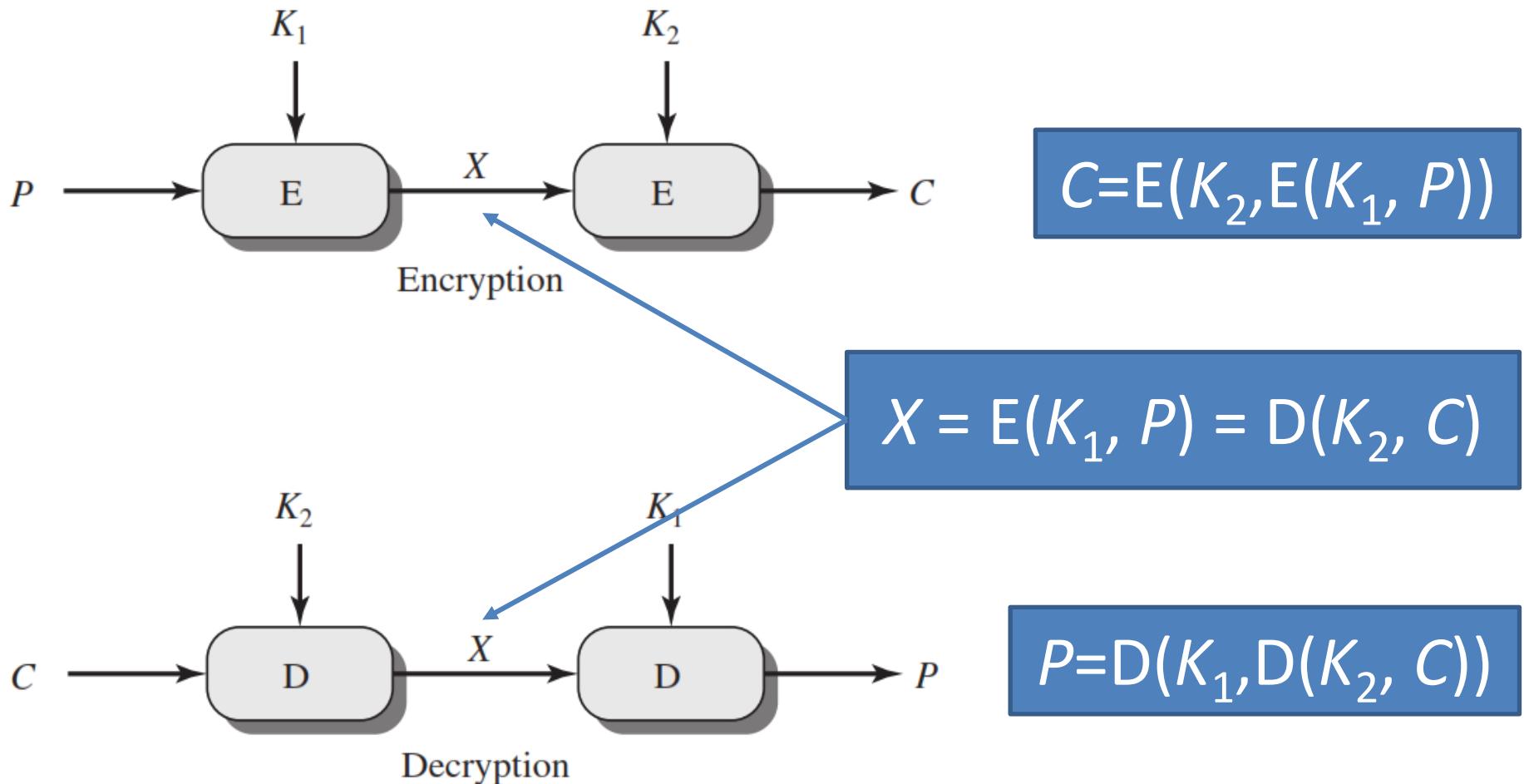
- Synchronous ciphers have these advantages and disadvantages:
  - Key stream can be pre-computed before plaintext or ciphertext is provided, often with parallelism. typically faster and more efficient than block ciphers when encrypting large volumes of data in real-time
  - As the keystream is deterministic, they have the disadvantage of ciphertext malleability (a known change in ciphertext produces a known change in plaintext) and so will need to be combined with some form of message authentication.
- Asynchronous ciphers have these advantages and disadvantages:
  - The key stream can only be computed once the plaintext/ciphertext is provided. However, changing the ciphertext changes subsequent key stream and so there is considerably less malleability.
  - They also allow easy synchronization at any point in transmission.
  - They are more computationally intensive and slower than synchronous stream ciphers because of the additional complexity in generating the keystream

# Multiple Encryption

---

- Given the **potential vulnerability of DES to a brute-force attack**, there has been considerable interest in finding an alternative.
- One approach is to design a completely new algorithm, of which **AES** is a prime example.
- Another alternative, which would preserve the existing investment in software and equipment, is to use **multiple encryption with DES and multiple keys**.

# Double DES



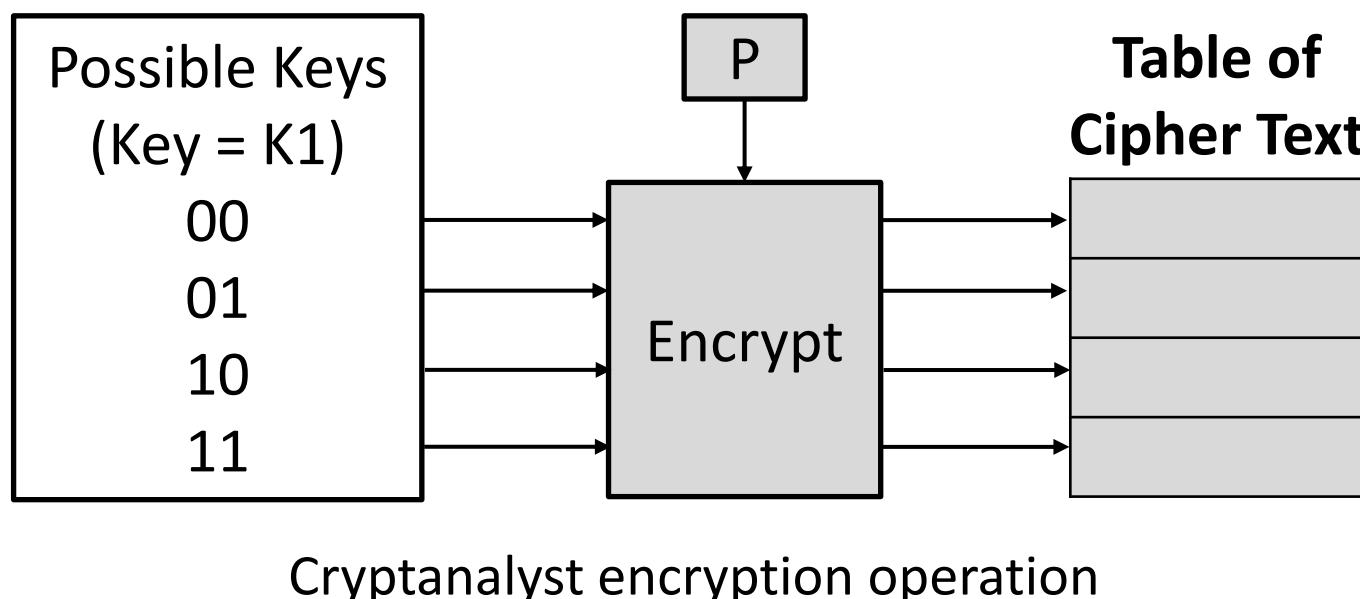
# Meet in the Middle Attack

---

- **Meet-in-the-middle (MITM) attacks** are often executed to decode multiple data encryption standard (DES) techniques.
- This attack involves encryption from one end, decryption from the other and matching the results in the middle.
- An attacker can use a MITM attack to bruteforce **Double DES**
  - an attacker encrypts the plaintext with all possible keys for the first encryption ( $2^{56}$  operations) and then decrypts the ciphertext with each possible key for the second encryption ( $2^{56}$  operations)

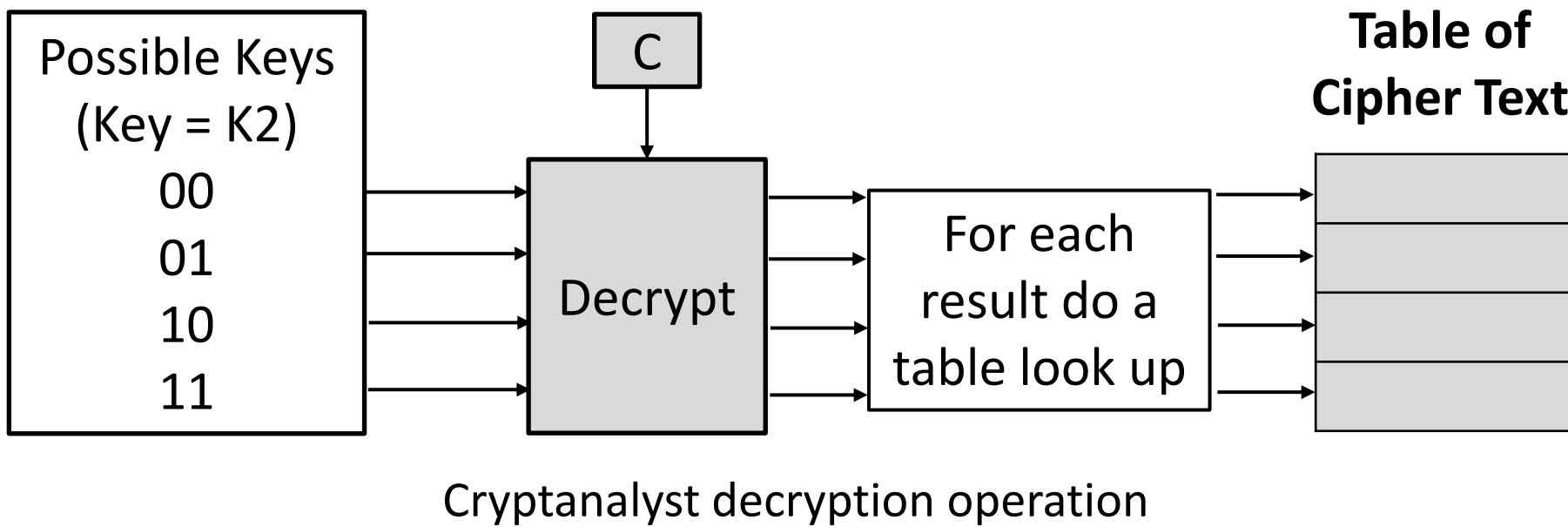
# Meet in the Middle Attack Step-1

- Suppose cryptanalyst knows **P** and corresponding **C**.
- Now, the aim is to obtain the values of **K<sub>1</sub>** and **K<sub>2</sub>**.
- For all possible values ( $2^{56}$ ) of key K1, the cryptanalyst would encrypt the P by performing E(K1,P).
- The cryptanalyst would store output in a table.



# Meet in the Middle Attack Step-2

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



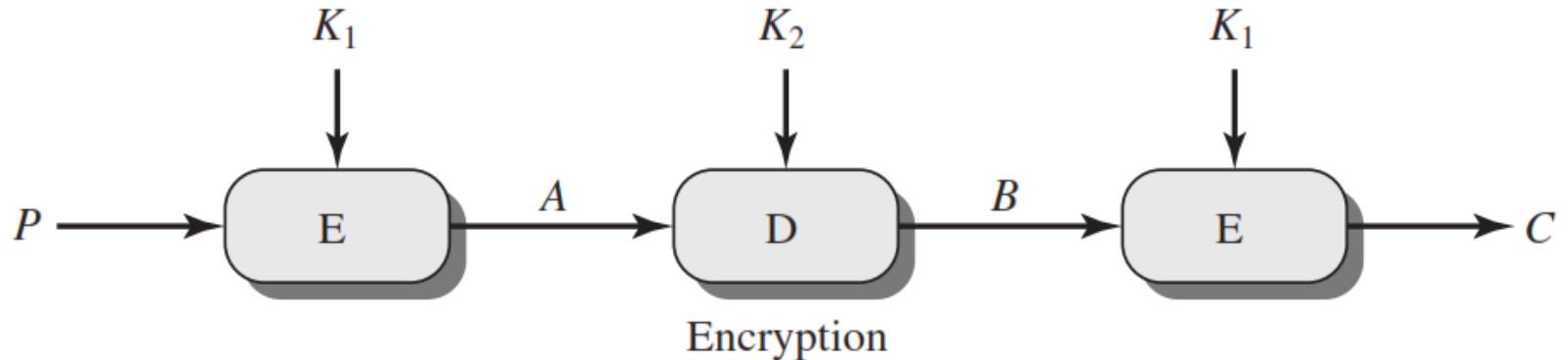
Thus, **K1 and K2 can be obtained**

# Triple DES

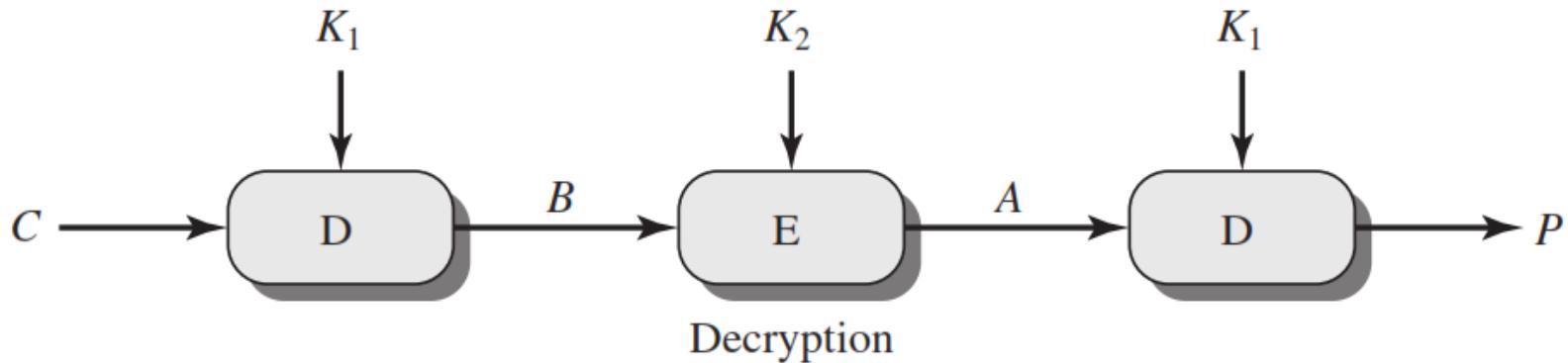
---

- An obvious counter to the meet in the middle attack is to use 3 stages of encryption with 3 different keys
- However, if 3 different keys are used, it has limitations of requiring a key length of  $56 \times 3 = 168$  bits which may be somewhat unwisely.
- As an alternative, Tuchman proposed a triple encryption method that uses **only 2 keys**.

# Triple DES



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



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

# Asymmetric Ciphers

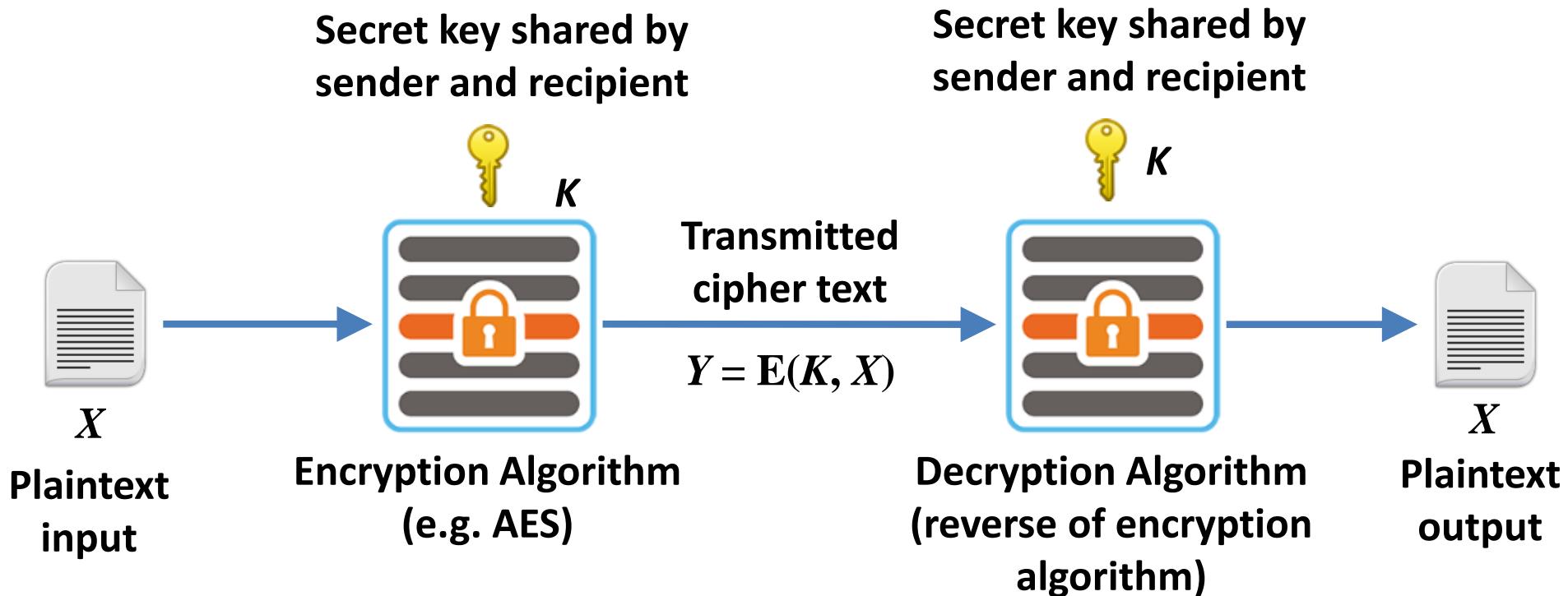


# Outline

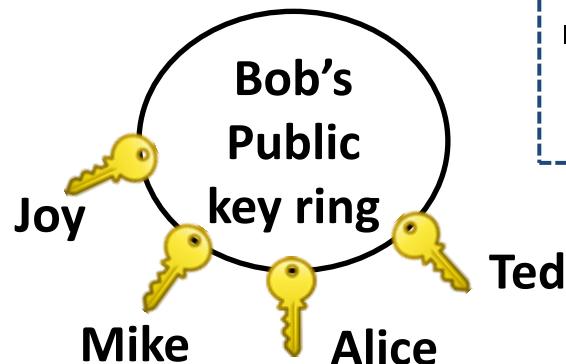
---

- Public Key Cryptosystems with Applications
- Requirements and Cryptanalysis
- RSA algorithm
- RSA computational aspects and security
- Diffie-Hillman Key Exchange algorithm
- Man-in-Middle attack

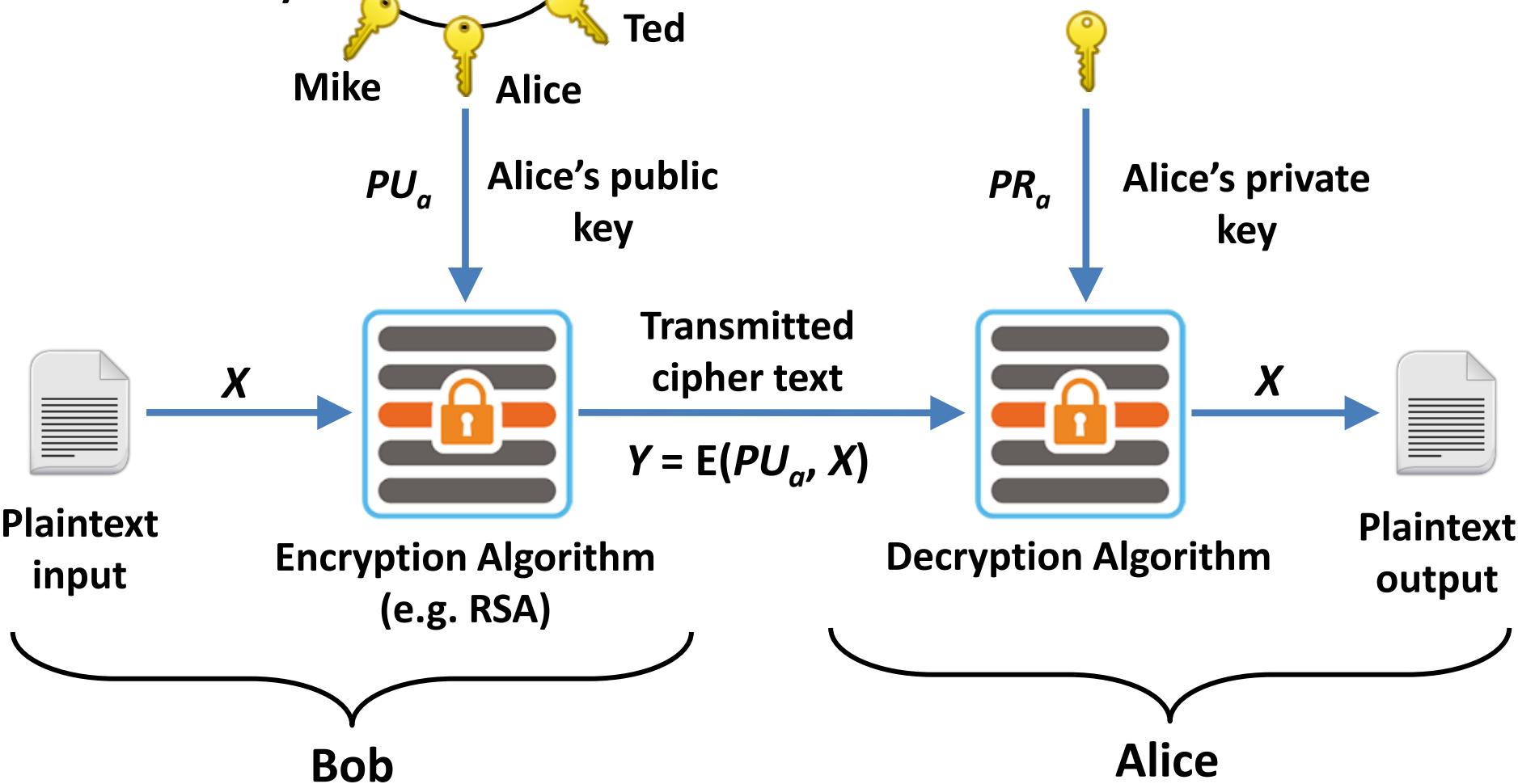
# Symmetric Key Encryption



# Asymmetric Key Encryption with Public Key

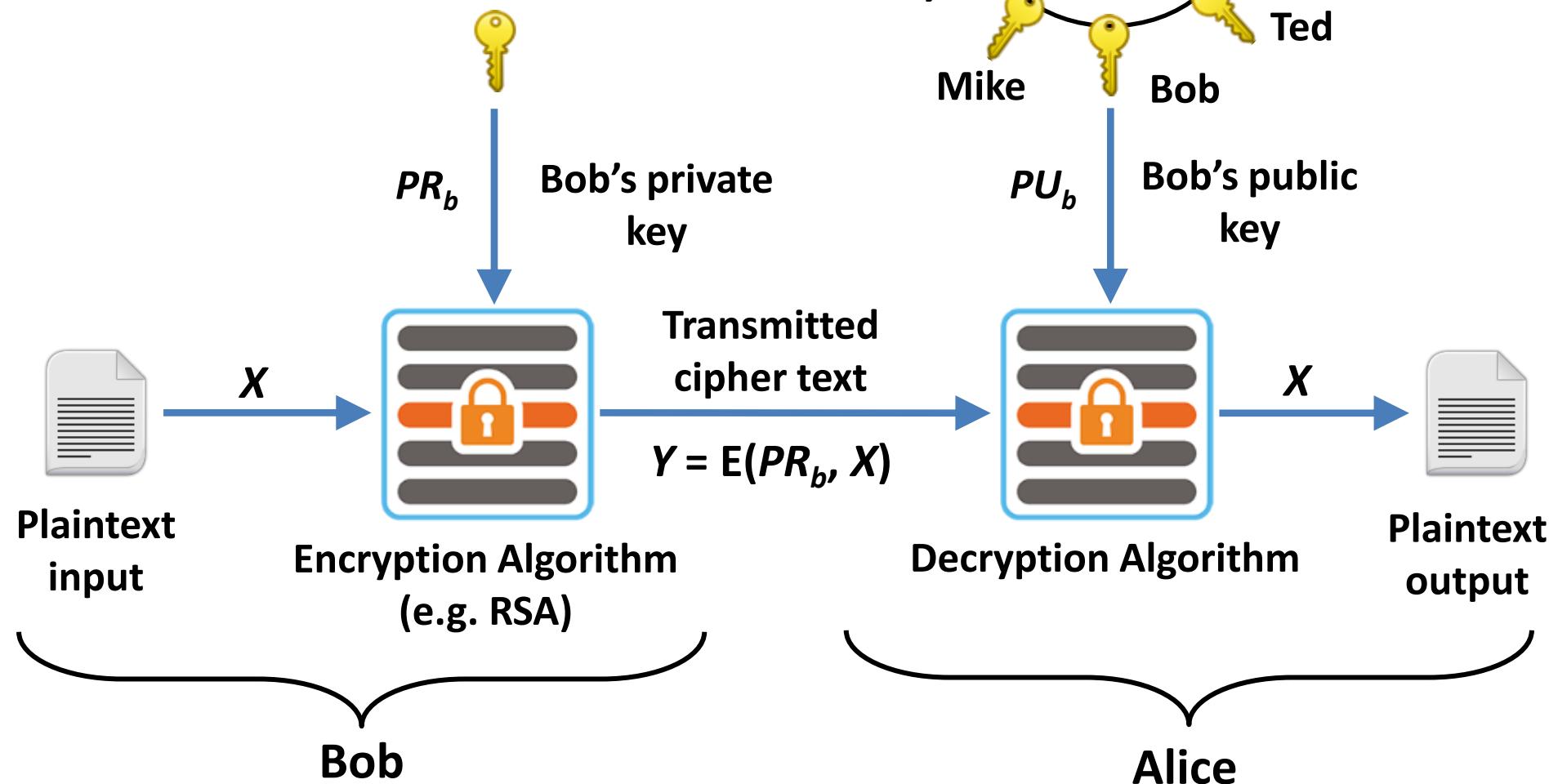


- The entire encrypted message serves as a **confidential message**.

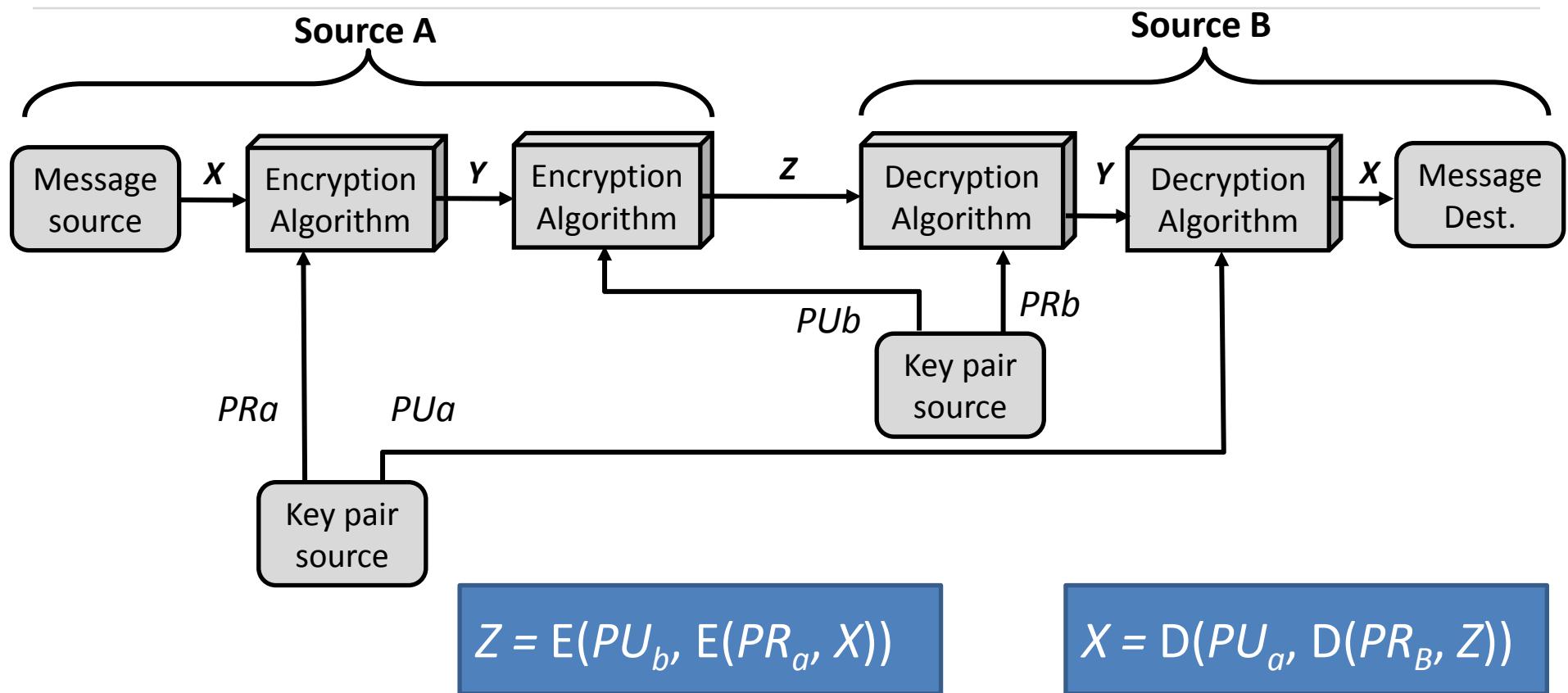


# Asymmetric key Encryption with Private Key

- The entire encrypted message serves as a **digital signature**.



# Authentication and Confidentiality



# Applications for Public-Key Cryptosystems

---

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender “signs” a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties. E.g. Diffie–Hellman key exchange scheme

# RSA Algorithm

---

- RSA is a block cipher in which the Plaintext and Ciphertext are represented as integers between 0 and n-1 for some n.
- Large messages can be broken up into a number of blocks.
- Each block would then be represented by an integer.

**Step-1:** Generate Public key and Private key

**Step-2:** Encrypt message using Public key

**Step-3:** Decrypt message using Private key

# Step-1: Generate Public key and Private key

- Select two large prime numbers:  $p$  and  $q$
- Calculate modulus :  $n = p * q$
- Calculate Euler's totient function :  $\phi(n) = (p-1) * (q-1)$
- Select  $e$  such that  $e$  is relatively prime to  $\phi(n)$  and  $1 < e < \phi(n)$

Two numbers are relatively prime if they have no common factors other than 1.

- Determine  $d$  such that  $d * e \equiv 1 \pmod{\phi(n)}$
- Publickey :  $PU = \{ e, n \}$
- Privatekey :  $PR = \{ d, n \}$

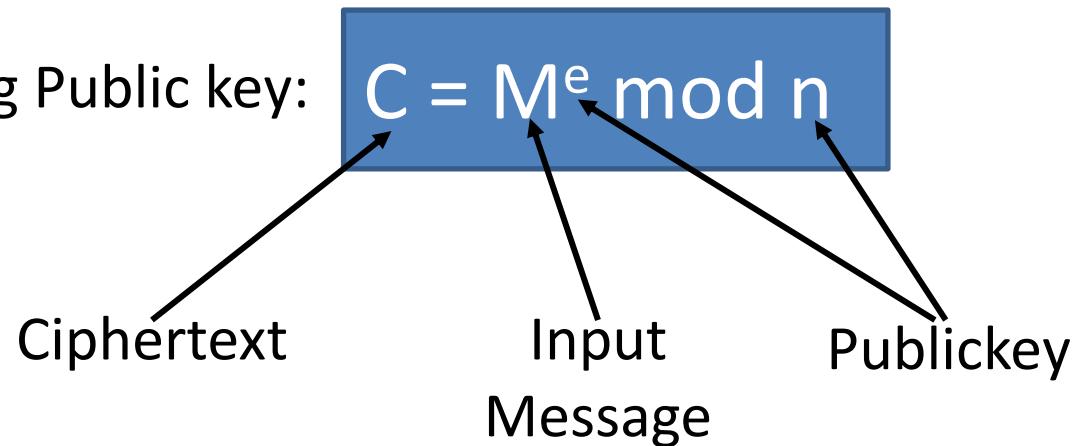
# Step-1: Generate Public key and Private key

- Select two large prime numbers:  $p = 3$  and  $q = 11$
- Calculate modulus :  $n = p * q, n = 33$
- Calculate Euler's totient function :  $\phi(n) = (p-1) * (q-1)$   
$$\phi(n) = (3 - 1) * (11 - 1) = 20$$
- Select  $e$  such that  $e$  is relatively prime to  $\phi(n)$  and  $1 < e < \phi(n)$
- We have several choices for  $e : 7, 11, 13, 17, 19$  Let's take  $e = 7$
- Determine  $d$  such that  $d * e \equiv 1 \pmod{\phi(n)}$
- $? * 7 \equiv 1 \pmod{20}, 3 * 7 \equiv 1 \pmod{20}$
- Public key :  $PU = \{ e, n \}, PU = \{ 7, 33 \}$
- Private key :  $PR = \{ d, n \}, PR = \{ 3, 33 \}$

- This is equivalent to finding  $d$  which satisfies  $de = 1 + j.\phi(n)$  where  $j$  is any integer.
  - We can rewrite this as  
$$d = (1 + j. \phi(n)) / e$$
- \*Find Modular Multiplicative Inverse using Extended Euclidean algorithm

# Step-2 : Encrypt Message

- Encryption Using Public key:



$$PU = \{ e, n \}, PU = \{ 7, 33 \}$$

For message  $M = 14$

$$C = 14^7 \text{ mod } 33$$

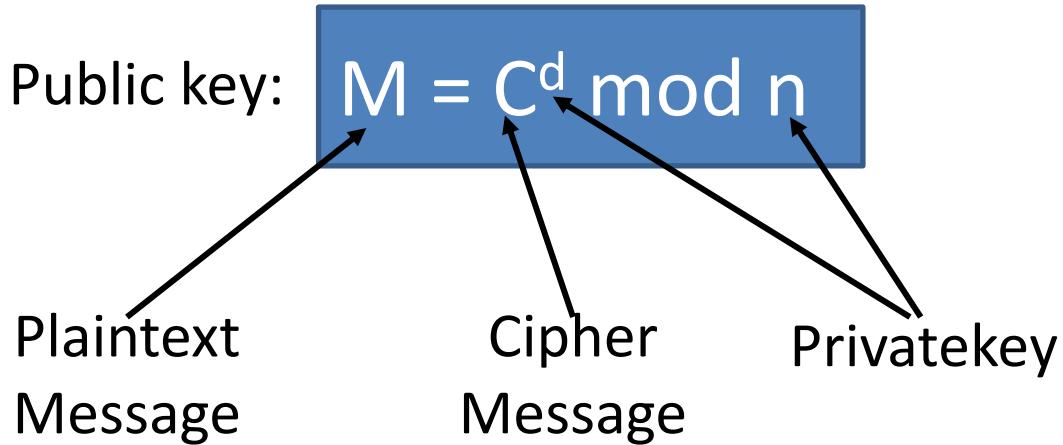
$$C = [(14^1 \text{ mod } 33) \times (14^2 \text{ mod } 33) \times (14^4 \text{ mod } 33)] \text{ mod } 33$$

$$C = (14 \times 31 \times 4) \text{ mod } 33 = 1736 \text{ mod } 33$$

$$C = 20$$

# Step-3 : Decrypt Message

- Encryption Using Public key:



$$PR = \{ d, n \}, PR = \{ 3, 33 \}$$

For Ciphertext  $C = 20$

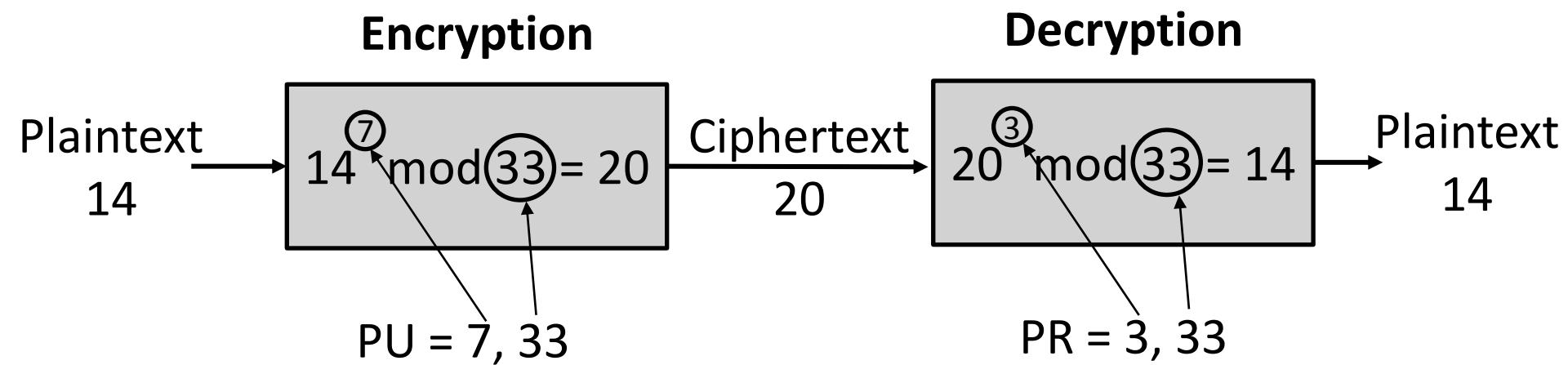
$$M = 20^3 \text{ mod } 33$$

$$M = [(20^1 \text{ mod } 33) \times (20^2 \text{ mod } 33)] \text{ mod } 33$$

$$M = (20 \times 4) \text{ mod } 33 = 80 \text{ mod } 33$$

$$M = 14$$

# Example RSA Algorithm



# RSA Example

- Find  $n$ ,  $\phi(n)$ ,  $e$ ,  $d$  for  $p=7$  and  $q= 19$  then demonstrate encryption and decryption for  $M = 6$

$$n = p * q = 7 * 19 = 133$$

$$\phi(n) = ( p - 1 ) * ( q - 1 ) = 108$$

Finding  $e$  relatively prime to 108  
 $e = 2 \Rightarrow \text{GCD}( 2, 108 ) = 2$  (no)  
 $e = 3 \Rightarrow \text{GCD}( 3, 108 ) = 3$  (no)  
 $e = 5 \Rightarrow \text{GCD}( 5, 108 ) = 1$  (Yes)

- Finding  $d$  such that  $(d * e) \bmod \phi(n) = 1$
  - We can rewrite this as  $d = (1 + j \cdot \phi(n)) / e$
- $j = 0 \Rightarrow d = 1 / 5 = 0.2 \leftarrow$  integer ? (no)
- $j = 1 \Rightarrow d = 109 / 5 = 21.8 \leftarrow$  integer ? (no)
- $j = 2 \Rightarrow d = 217 / 5 = 43.4 \leftarrow$  integer ? (no)
- $j = 3 \Rightarrow d = 325 / 5 = 65$  integer ? (yes)
- \*OR Find Modular Multiplicative Inverse using Extended Euclidean algorithm

Public key :

**PU = { e, n } = {5, 133}**

Private key :

**PR = { d, n } = {65, 133}**

# RSA Example – cont...

- Encryption:

$$C = M^e \bmod n$$

$$PU = \{ e, n \}, PU = \{ 5, 133 \}$$

For message  $M = 6$

$$C = 6^5 \bmod 133$$

$$C = 7776 \bmod 33$$

$$C = 62$$

- Decryption:

$$M = C^d \bmod n$$

$$PR = \{ d, n \}, PR = \{ 65, 133 \}$$

For  $C = 62$

$$M = 62^{65} \bmod 133$$

$$M = 2666 \bmod 33$$

$$M = 6$$

# RSA Example

---

- P and Q are two prime numbers. P=7, and Q=17. Take public key E=5. If plain text value is 10, then what will be cipher text value according to RSA algorithm?
- $n = 119$
- $\phi(n) = 96$
- $e = 5$
- $d = 77$
- PU = { 5, 119 }
- PR = {77, 119}
- $C = 10^5 \bmod 119 \Rightarrow C = 40$

# RSA Security

---

- possible approaches to attacking RSA are:
  - brute force key search - infeasible given size of numbers
  - mathematical attacks - based on difficulty of computing  $\phi(n)$ , by factoring modulus n
  - timing attacks - on running of decryption
  - chosen ciphertext attacks - given properties of RSA

# Mathematical Attack

---

- mathematical approach takes 3 forms:
  - factor  $n=p \cdot q$ , hence compute  $\phi(n)$  and then  $d$
  - determine  $\phi(n)$  directly and compute  $d$
  - find  $d$  directly
- currently assume 1024-2048 bit RSA is secure

# Timing Attacks

---

- Developed by Paul Kocher in mid-1990's
- Exploit timing variations in operations
  - E.g. multiplying by small vs large number
- Infer operand size based on time taken
- Infer time taken in exponentiation
- Countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

# Chosen Ciphertext Attacks

---

- RSA is vulnerable to a Chosen Ciphertext Attack (CCA)
- Attackers can choose ciphertexts & get decrypted plaintext back
- Countermeasure with random pad of plaintext

# Primitive root

---

- Let  $p$  be a prime number
- Then  $a$  is a primitive root for  $p$ , if the powers of  $a$  modulo  $p$  generates all integers from  $1$  to  $p - 1$  in some permutation.

$$a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$$

- Example:  $p = 7$  then primitive root is  $3$  because powers of  $3$  mod  $7$  generates all the integers from  $1$  to  $6$

$$3^1 = 3 \equiv 3 \pmod{7}$$

$$3^2 = 9 \equiv 2 \pmod{7}$$

$$3^3 = 27 \equiv 6 \pmod{7}$$

$$3^4 = 81 \equiv 4 \pmod{7}$$

$$3^5 = 243 \equiv 5 \pmod{7}$$

$$3^6 = 729 \equiv 1 \pmod{7}$$

# Discrete Logarithm

---

- For any integer  $b$  and a primitive root  $a$  of prime number  $p$ , we can find a unique exponent  $i$  such that

$$b = a^i \pmod{p} \text{ where } 0 \leq i \leq (p - 1)$$

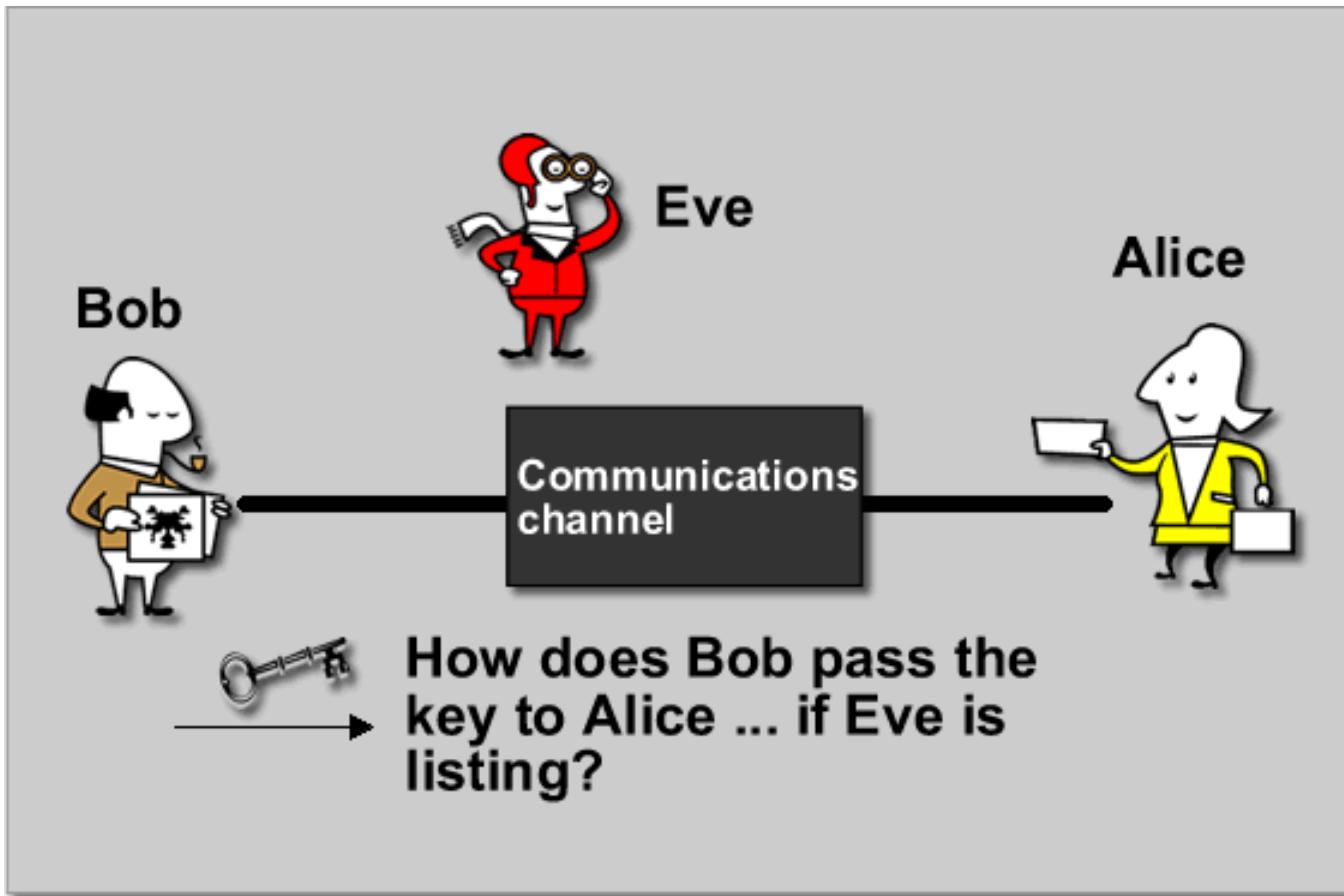
- The exponent  $i$  is referred as the discrete logarithm of  $b$  for the base  $a$ , mod  $p$ . It expressed as below.

$$\text{dlog}_{a,p}(b)$$

# Key Establishment Problem

- Securing communication requires that the data is encrypted before being transmitted
- Associated with encryption and decryption are keys that must be shared by the participants.
- The problem of securing the data then becomes the problem of securing the key establishment
- Task: If the participants do not physically meet, then how do the participants establish a shared key?

# Key Establishment Problem (cont.)

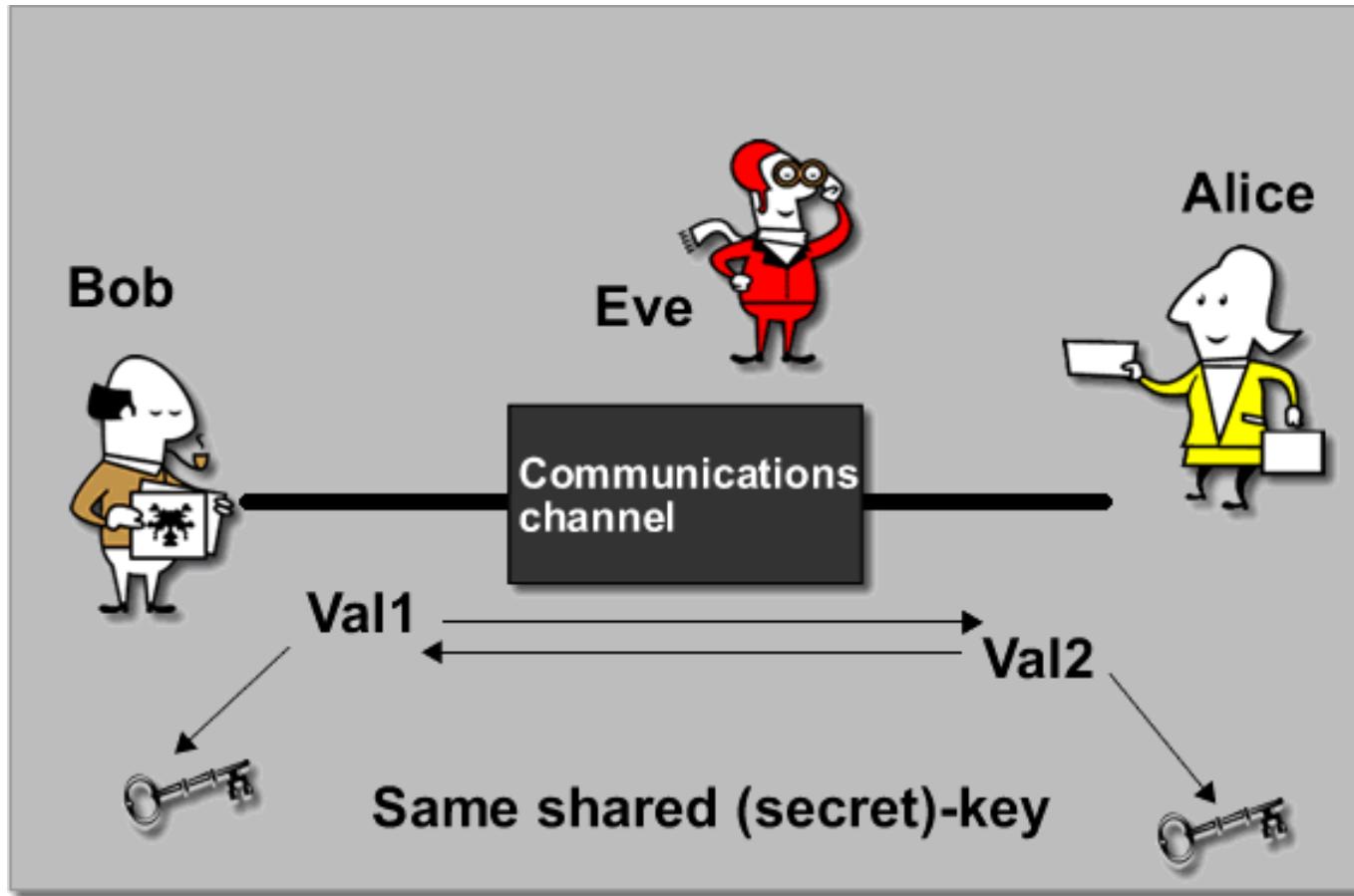


# Diffie-Hellman Key Agreement

- Discovered by Whitfield Diffie and Martin Hellman
- Diffie-Hellman key agreement protocol
  - Exponential key agreement
  - Allows two users to exchange a secret key
  - Requires no prior secrets
  - Real-time over an **untrusted** network



# Diffie-Hellman Key Agreement (cont.)



# Diffie-Hellman Algorithm

- Requires two large numbers, one prime  $p$ , and generator  $g$  ( $2 \leq g \leq p-2$ ), a primitive root of  $p$ , ( $p$  and  $g$  are both publicly available numbers).
- Users pick random private values  $x$  ( $x < p$ ) and  $y$  ( $y < p$ )
- Compute public values (keys)
  - $R_1 = g^x \text{ mod } p$
  - $R_2 = g^y \text{ mod } p$
- Keys  $R_1$  and  $R_2$  are exchanged
- Compute shared, private key
  - $k_{\text{alice}} = (R_2)^x \text{ mod } p$
  - $k_{\text{bob}} = (R_1)^y \text{ mod } p$
- Algebraically it can be shown that  $k_{\text{alice}} = k_{\text{bob}}$ 
  - Users now have a symmetric secret key to encrypt

# Proof

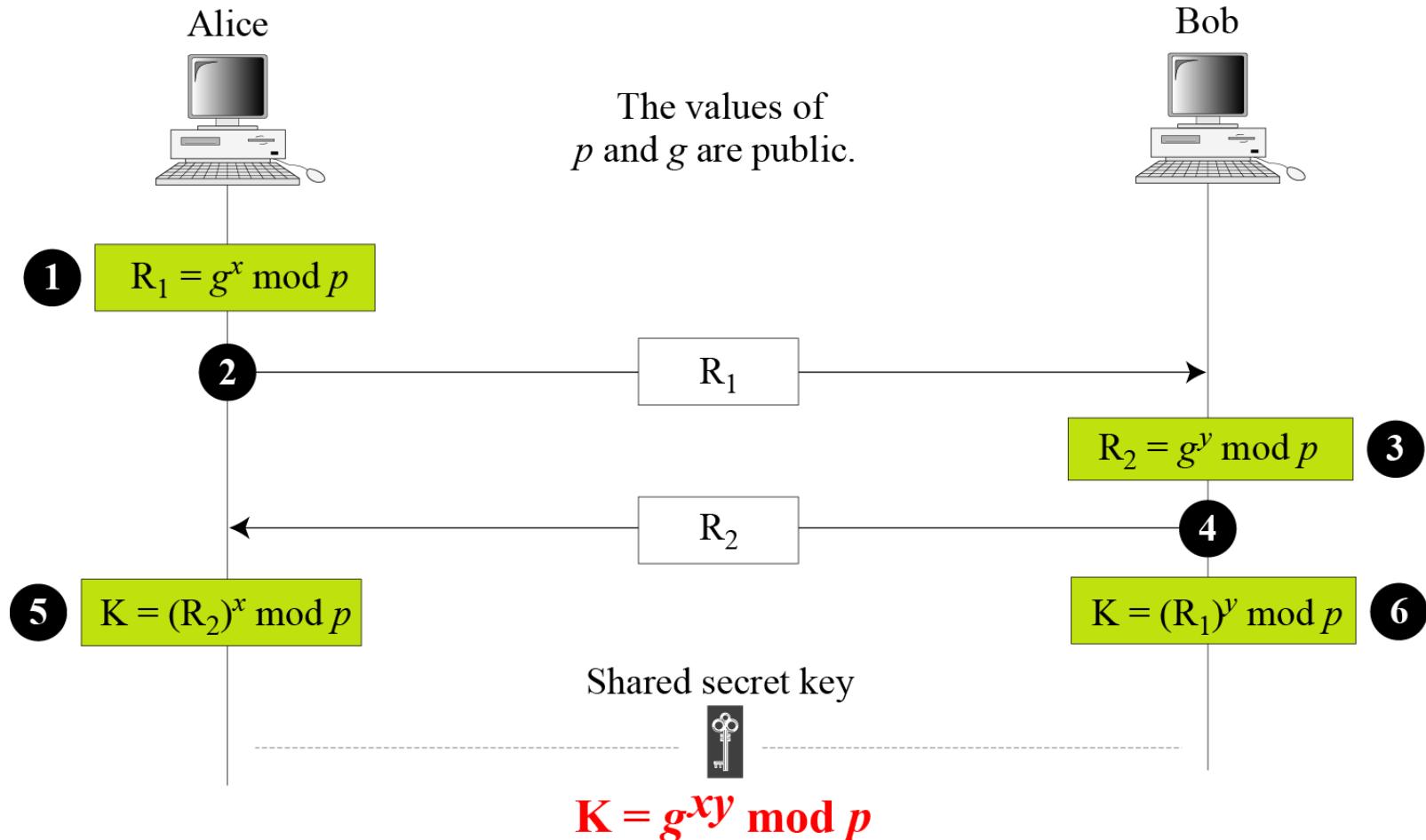
- We know

$$R1 = g^x \bmod p$$

$$R2 = g^y \bmod p$$

- $k_{\text{alice}} = (R2)^x \bmod p$   
 $= (g^y \bmod p)^x \bmod p$   
 $= (g^y)^x \bmod p$   
 $= (g)^{yx} \bmod p$   
 $= (g^x)^y \bmod p$   
 $= (g^x \bmod p)^y \bmod p$   
 $= (R1)^y \bmod p$   
 $= k_{\text{bob}}$

# Key Exchange



# Example

- Alice and Bob get public numbers
  - $P = 19, G = 3$  [Primitive roots of modulus 19 are 2,3,10,13,14,15]
- Alice and Bob pick private values  $x=15$  &  $y=10$  respectively
- Alice and Bob compute public values
  - $R_1 = 3^{15} \text{ mod } 19 = 12$
  - $R_2 = 3^{10} \text{ mod } 19 = 16$
  - Alice and Bob exchange public numbers
- Alice and Bob compute symmetric keys
  - $k_{\text{alice}} = (R_2)^x \text{ mod } p = (16)^{15} \text{ mod } 19 = 7$
  - $k_{\text{bob}} = (R_1)^y \text{ mod } p = (12)^{10} \text{ mod } 19 = 7$
- Alice and Bob now can talk securely!

**Example: P=23 and G=5 ?**

# Security of Diffie-Hellamn

- This protocol susceptible to two attacks:
  - The Man-in-the-middle attack
  - The Discrete logarithmic attack

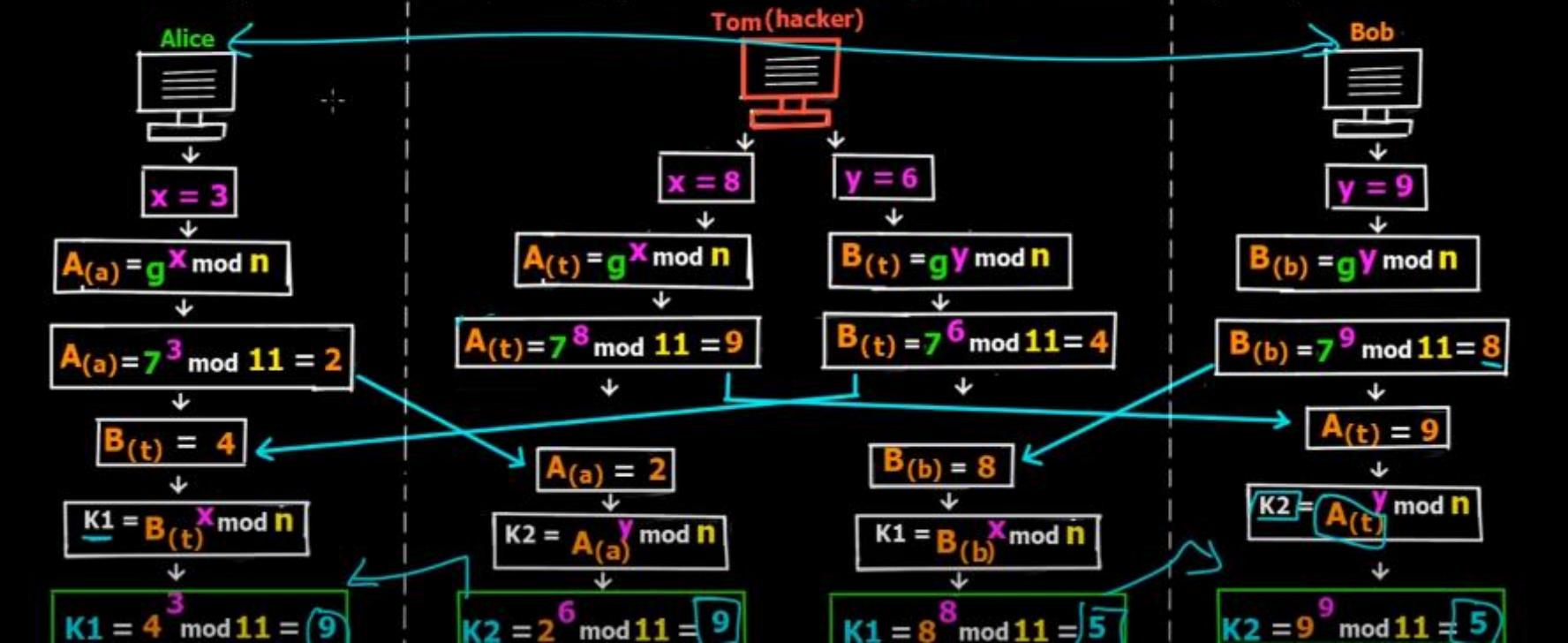
# Man in the middle attack

---

- Suppose Alice and Bob wish to exchange keys, and Darth is the adversary.
1. Darth prepares for the attack by generating two random private keys  $X_{D1}$  and  $X_{D2}$  and then computes corresponding public keys  $Y_{D1}$  and  $Y_{D2}$ .
  2. Alice transmits  $Y_A$  to Bob.
  3. Darth intercepts  $Y_A$  and transmits  $Y_{D1}$  to Bob. Darth also calculates  $K_2 = (Y_A)^{XD2} \bmod q$ .
  4. Bob receives  $Y_{D1}$  and calculates  $K_1 = (Y_{D1})^{XB} \bmod q$ .
  5. Bob transmits  $Y_B$  to Alice.
  6. Darth intercepts  $Y_B$  and transmits  $Y_{D2}$  to Alice. Darth calculates  $K_1 = (Y_B)^{XD1} \bmod q$ .
  7. Alice receives  $Y_{D2}$  and calculates  $K_2 = (Y_{D2})^{XA} \bmod q$ .

## Man-in-the-Middle Attack (Bucket-Bridge-Attack)

1 Alice & Bob agree upon 2 large prime numbers  $n = 11$   $g = 7$ . These numbers are publicly known



# ElGamal Encryption Algorithm

---

- ElGamal encryption uses asymmetric key encryption for communicating between two parties and encrypting the message. It is based on the Diffie–Hellman key exchange.
- This cryptosystem is based on the difficulty of finding **discrete logarithm** in a cyclic group that is even if we know  $g^a$  and  $g^k$ , it is extremely difficult to compute  $g^{ak}$ .
- ElGamal is generally used to encrypt only the symmetric key (Not the plaintext). This is because asymmetric cryptosystems like ElGamal are usually slower than symmetric ones

# ElGamal Encryption

## ❖ Keys & parameters

- Domain parameter = { $p, g$ }
- Choose  $x \in [1, p-1]$  and compute  $y = g^x \bmod p$
- Public key ( $p, g, y$ )
- Private key  $x$

## ❖ Encryption: $m \rightarrow (C_1, C_2)$

- Pick a random integer  $k \in [1, p-1]$
- Compute  $C_1 = g^k \bmod p$
- Compute  $C_2 = m \times y^k \bmod p$

## ❖ Decryption

- $m = C_2 \times C_1^{-x} \bmod p$
- $C_2 \times C_1^{-x} = (m \times y^k) \times (g^k)^{-x} = m \times (g^x)^k \times (g^k)^{-x} = m \bmod p$

# ElGamal Example

---

- Keys & parameters

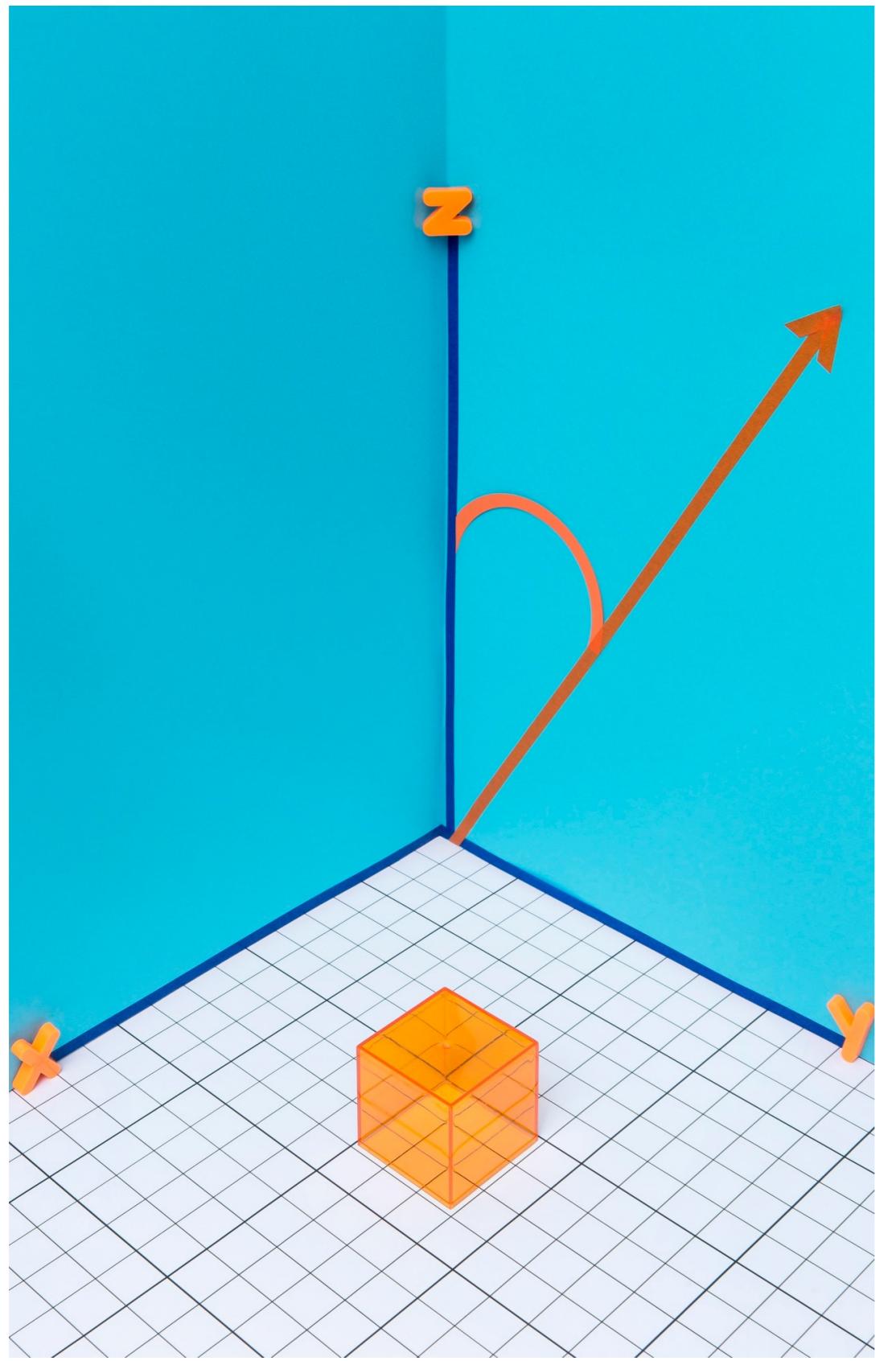
- Let prime number  $p=23$  and generator  $g=7$
- Choose  $x=9$  and  $y= g^x \bmod p = 7^9 \bmod 23=15$
- Public key:  $\{23, 7, 15\}$
- Private key=9

- Encryption for  $m=20$

- Choose random  $k=3$
- $C1= 7^3 \bmod 23=21$
- $C2=20 \times 15^3 \bmod 23=20 \times 17 \bmod 23=18$
- Send  $(C1,C2)=(21,18)$  as a ciphertext

- Decryption

- $M=18 \times 21^{-9} \bmod 23 = 20$

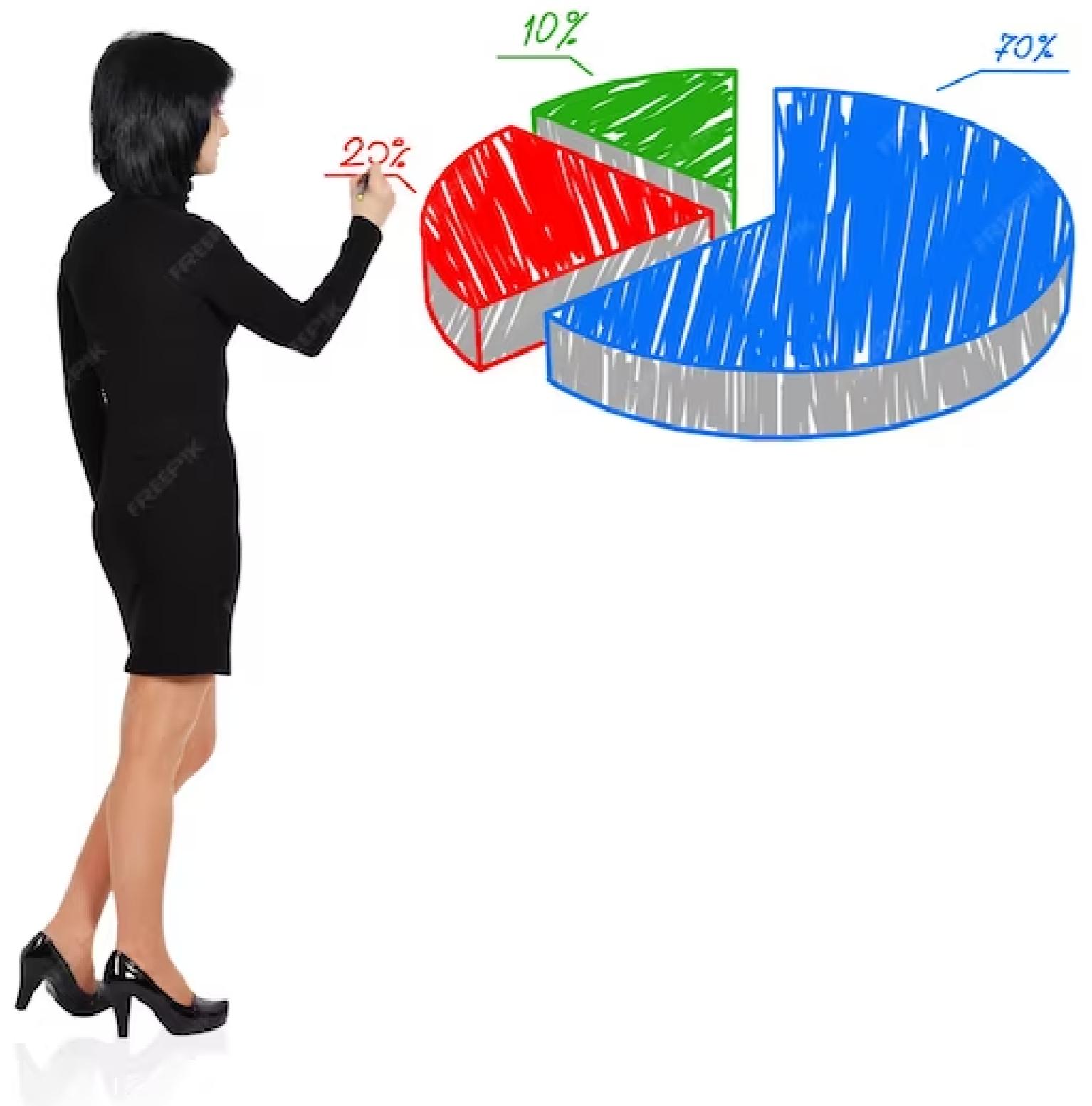


# Unraveling the ECC Encryption and Decryption Process: A Comprehensive Overview



# Introduction

Welcome to the presentation on **Unraveling the ECC Encryption and Decryption Process**. This comprehensive overview will delve into the intricacies of the **Elliptic Curve Cryptography** algorithm, highlighting its strengths and applications in modern cryptography.



## What is ECC?

**Elliptic Curve Cryptography (ECC)** is a public-key cryptographic algorithm based on the mathematics of elliptic curves over finite fields. It offers a high level of security with smaller key sizes compared to other encryption methods. ECC is widely used in various applications, including secure communication, digital signatures, and secure key exchange.



## Key Generation

The ECC encryption process begins with key generation. A private key is randomly generated, and a corresponding public key is derived using mathematical operations on the elliptic curve.

The security of ECC relies on the difficulty of solving the **elliptic curve discrete logarithm problem**.



# Encryption

To encrypt a message using ECC, the sender converts the plaintext into a point on the elliptic curve. The sender then generates a random number, performs mathematical operations on the curve, and combines the result with the plaintext point to produce the ciphertext. Only the intended recipient with the private key can decrypt the ciphertext.

# Decryption

Decryption in ECC involves the recipient using their private key to perform mathematical operations on the ciphertext point, resulting in the recovery of the original plaintext point.

The private key operation exploits the properties of the elliptic curve to reverse the encryption process and retrieve the original message.



# Strengths of ECC

ECC offers several advantages over other encryption methods. It provides a high level of security with smaller key sizes, making it more efficient in terms of computational resources and storage. ECC is resistant to attacks such as brute force and integer factorization, making it suitable for resource-constrained devices and applications.





## Applications of ECC

ECC finds applications in various domains. It is widely used in secure communication protocols like **TLS/SSL** to ensure encrypted data transmission. ECC is also utilized in **digital signatures** to verify the authenticity and integrity of digital documents. Additionally, ECC plays a crucial role in **secure key exchange** algorithms such as **Diffie-Hellman**.



## Challenges and Considerations

While ECC offers numerous benefits, it also poses challenges and considerations. Implementation errors, side-channel attacks, and the selection of appropriate elliptic curves are critical factors to address. Additionally, the choice of key size and the need for standardized ECC parameters require careful consideration to ensure optimal security.



## Future Developments

Ongoing research and development in ECC aim to further enhance its security and efficiency. Advancements include the exploration of post-quantum ECC, which focuses on developing ECC variants resistant to attacks by quantum computers. Additionally, efforts are being made to optimize ECC implementations for different platforms and improve interoperability.

# Conclusion

In conclusion, **Elliptic Curve Cryptography** is a powerful encryption and decryption algorithm that provides robust security with smaller key sizes. Its widespread adoption in various applications underscores its effectiveness in protecting sensitive data. As technology advances, ECC continues to evolve, ensuring secure communication and data integrity in an increasingly interconnected world.

# Thanks!

Do you have any questions? [addyouremail@freepik.com](mailto:addyouremail@freepik.com)  
+91 620 421 838  
[yourcompany.com](http://yourcompany.com)



# Cryptographic Hash Functions



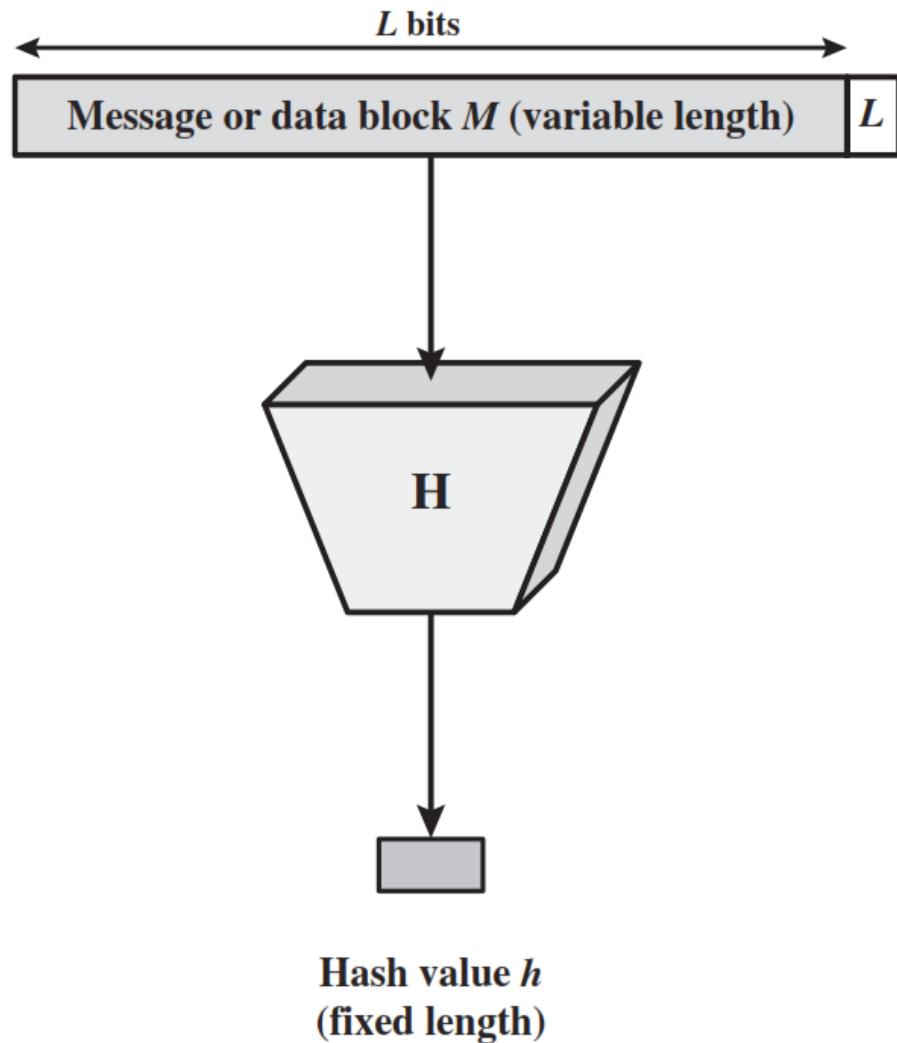
# Outline

---

- Cryptographic Hash Functions
- Applications
- Simple hash functions
- Requirements and security
- Hash functions based on Cipher Block Chaining
- Secure Hash Algorithm (SHA)

# Hash Function

- A hash function  $H$  accepts a variable-length block of data  $M$  as input and produces a fixed-size hash value  $h = H(M)$ .
- A “good” hash function has the property that the results of applying a change to any bit or bits in  $M$  results with high probability, in a change to the hash code.



# Applications of Cryptographic Hash Functions

---

- 1. Message authentication**
- 2. Digital Signature**
- 3. One-way password file**

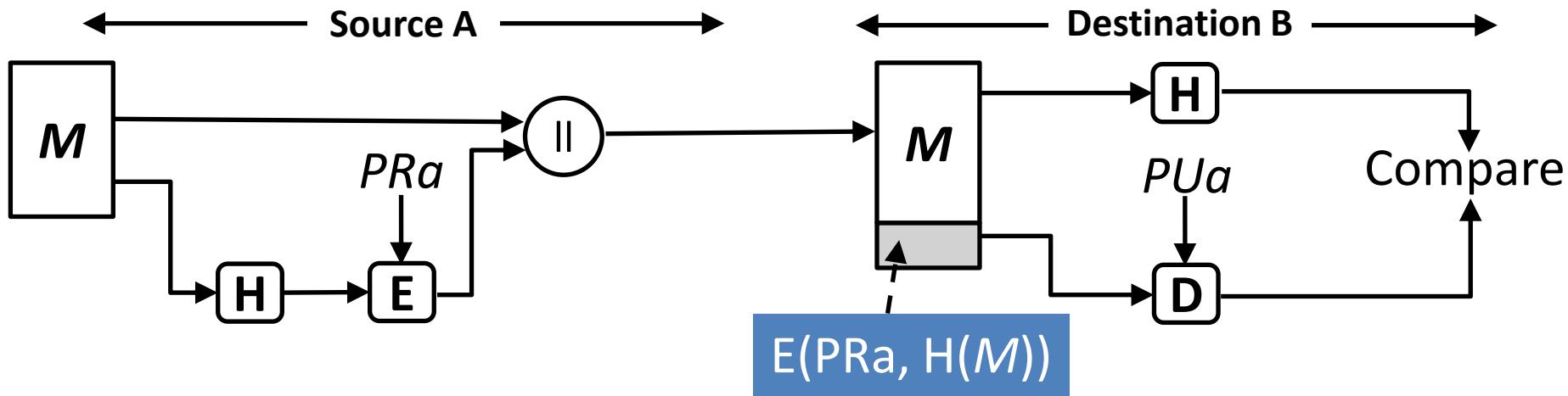
Some topics of Unit 3/Unit 4 will be covered by the Expert

# Digital Signature

---

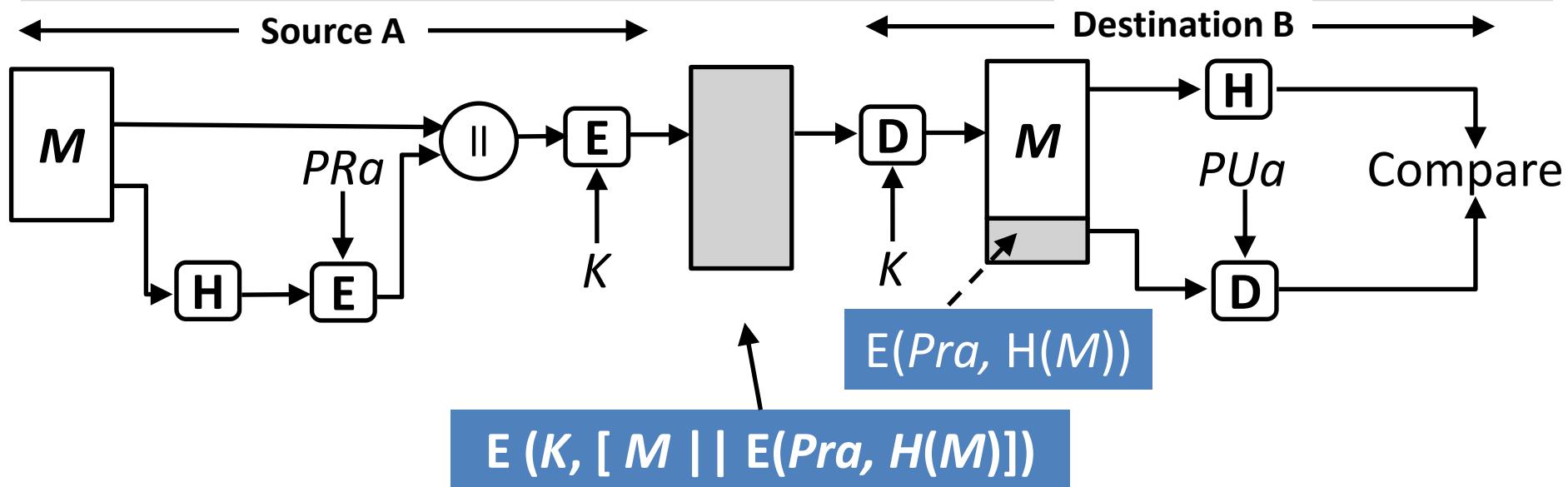
- A **digital signature** is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document.
- The operation of the digital signature is similar to that of the Message Authentication Code (**MAC**).
- In the case of the **digital signature**, the hash value of a message is encrypted with a user's **private key**.
- Anyone who knows the user's **public key** can verify the integrity of the message that is associated with the **digital signature**.

# Digital Signature method - 1



- The hash code is encrypted, using public-key encryption with the sender's private key. This provides **authentication**.
- It also provides a digital signature, because only the sender could have produced the encrypted hash code.

# Digital Signature method - 2



- If **confidentiality** as well as a **digital signature** is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.

# Security Requirements

---

1. Disclosure
2. Traffic analysis
3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification
7. Source repudiation
8. Destination repudiation

# Requirements for hash functions

---

1. It can be applied to any *sized message M*.
2. It should produce *fixed-length output h*.
3. It should be *easy to compute  $h=H(M)$*  for any *message M*.
4. Given the hash value *h*, it is *infeasible* to find *y* such that ( $H(y) = h$ )
  - *One-way property*
5. For given block *x*, it is computationally infeasible to find *y*,  
*y ≠ x* with  $H(y) = H(x)$ 
  - *Weak collision resistance (i.e., it's hard to find another input 'y' that produces the same hash output)*
6. It is computationally infeasible to find messages *m<sub>1</sub>* and *m<sub>2</sub>*  
where their hash values are equal i.e.  $H(m_1) = H(m_2)$ 
  - *Strong collision resistance*

# Simple Hash Function

---

- The input (message, file, etc.) is viewed as a sequence of  $n$ -bit blocks.
- The input is processed one block at a time in an iterative fashion to produce an  $n$ -bit hash.
- One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as

$$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

- where

$C_i$  =  $i$ th bit of the hash code,  $1 \dots i \dots n$

$m$  = number of  $n$ -bit blocks in the input

$b_{ij}$  =  $i$ th bit in  $j$ th block

$\oplus$  = XOR operation

# SHA

---

- In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA) as virtually every other widely used hash function had been found to have substantial cryptanalytic weaknesses
- SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.
- When weaknesses were discovered in SHA, different versions had been developed subsequently.

# SHA - Secure Hash Algorithm

	SHA - 1	SHA - 224	SHA - 256	SHA - 384	SHA - 512
Message Digest Size (bits)	160	224	256	384	512
Message Size (bits)	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size (bits)	512	512	512	1024	1024
Word Size (bits)	32	32	32	64	64
No. of Steps	80	64	64	80	80

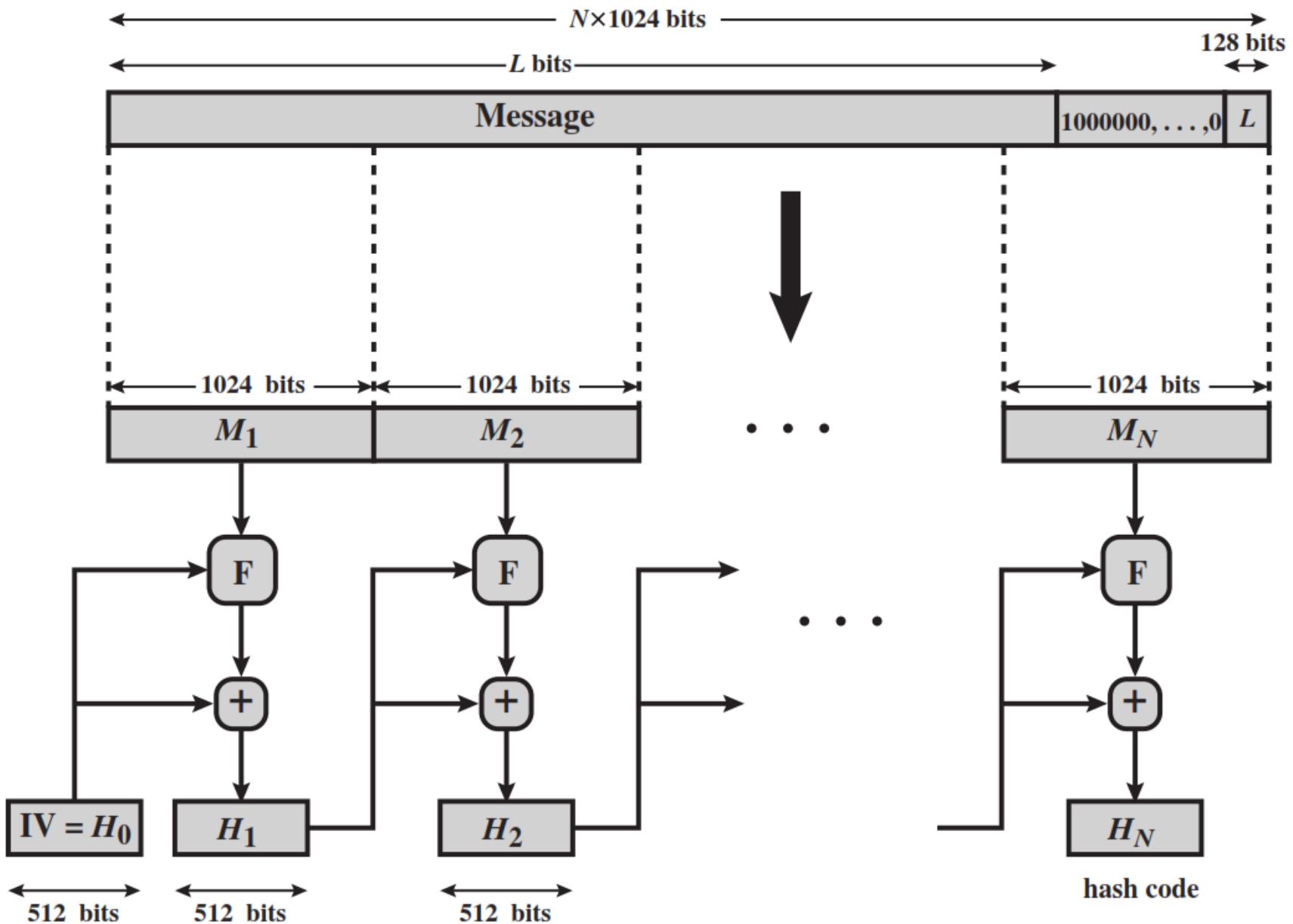
- Message digest=> Hash Value
- There is technically a limit for Message Size as per the padding scheme requirement

# SHA - 512

---

- The algorithm takes as input a message with a maximum length of less than **2<sup>128</sup>** bits
- It produces output of a **512-bit** message digest.
- The input is processed in **1024-bit** blocks.

# Message Digest Generation using SHA -512



# Step -1 Append Padding Bits

---

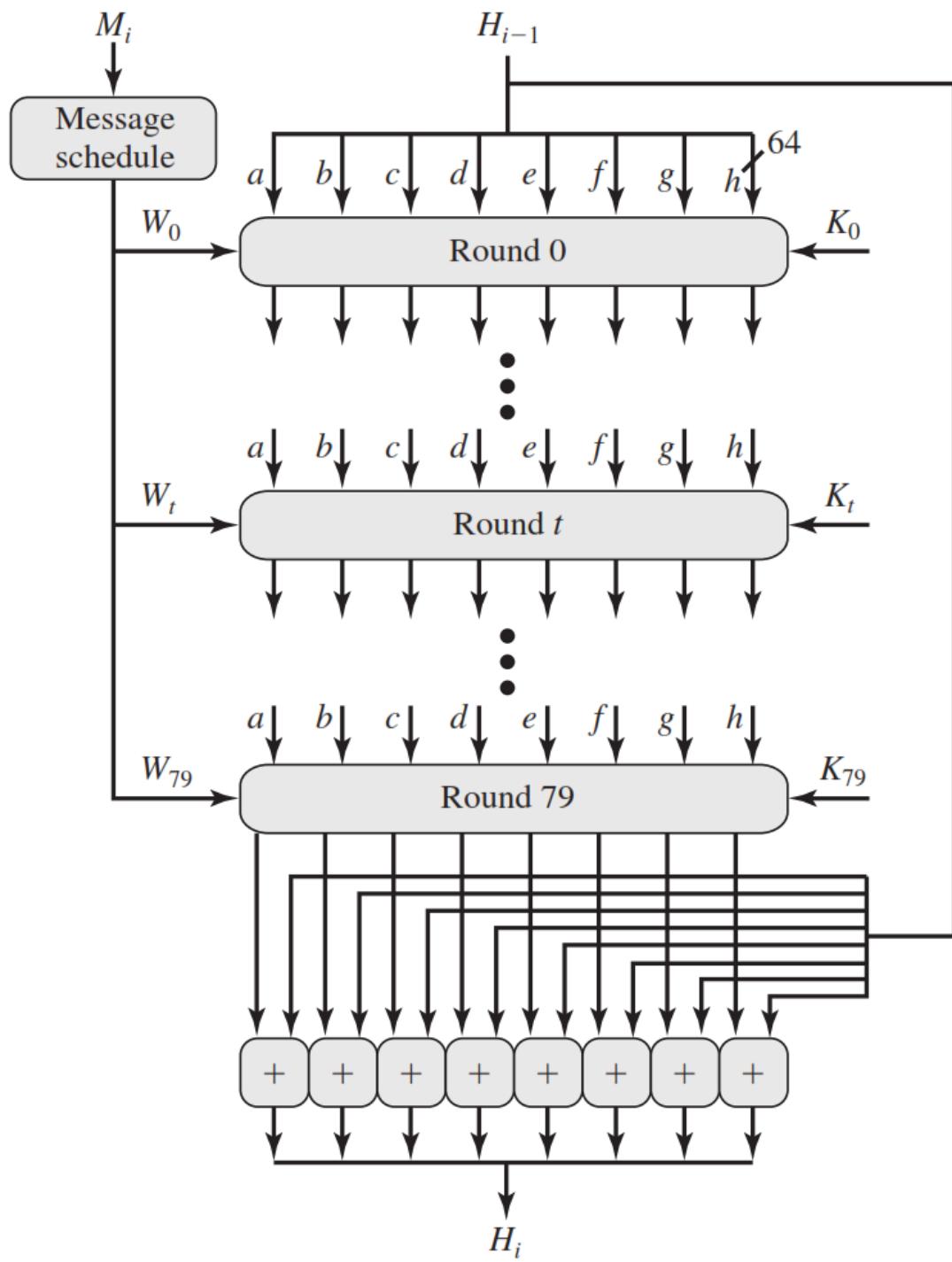
- The message is padded so that its length is congruent to **896 modulo 1024** [ $\text{length} \equiv 896 \pmod{1024}$ ] . (To keep space for Appending Message Length-Check Step 2)
- Padding is always added, even if the message is already of the desired length.
- Thus, the number of padding bits is in the range of **1 to 1024**.
- The padding consists of a single 1 bit followed by the necessary number of 0 bits.

# Step -2 Append Length

---

- A block of **128** bits is appended to the message.
- This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).
- The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length.
- In Figure, the expanded message is represented as the sequence of 1024-bit blocks  $M_1, M_2, \dots, M_N$ , so that the total length of the expanded message is  $N * 1024$  bits.

# SHA-512 Processing of a Single 1024-Bit Block



# Step -3 Initialize hash buffer

---

- The outcome of the first two steps produces a message that is an integer multiple of 1024 bits in length.
- the expanded message is represented as the sequence of 1024-bit blocks  $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_N$ , so that the total length of expanded message is  $N \times 1024$  bits.
- A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as **eight 64-bit registers (a, b, c, d, e, f, g, h)**.

## Step -4 Process message in 1024-bit (128-word) blocks

---

- The heart of the algorithm is a module that consists of **80 rounds**; this module is labelled F

# SHA-512 Processing of a Single 1024-Bit Block

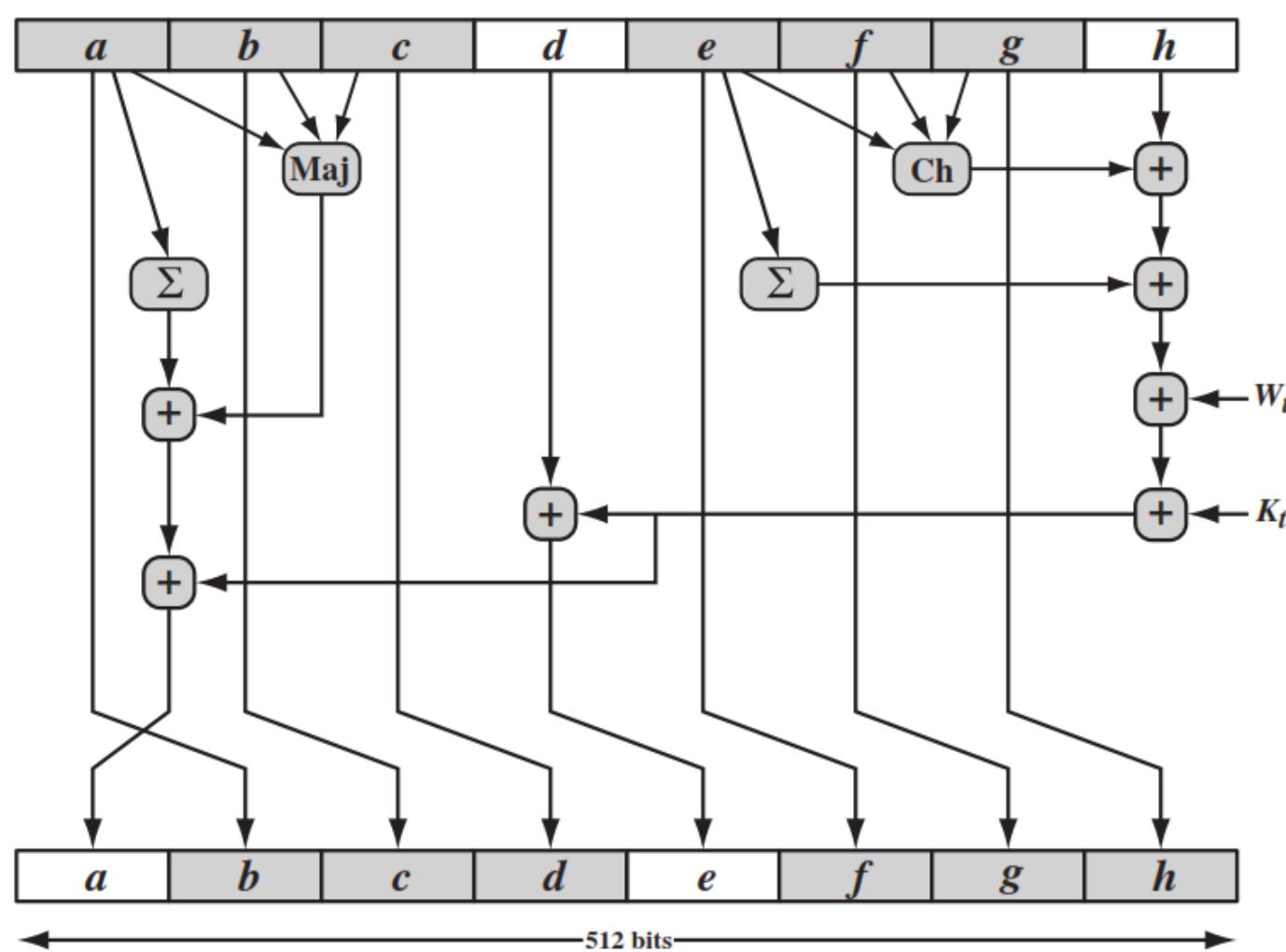
---

- Each round takes as input the 512-bit buffer value,  $abcdefg$ , and updates the contents of the buffer.
- At input to the first round, the buffer has the intermediate hash value,  $H_{i-1}$ .
- Each round  $t$  makes use of a **64-bit value  $W_t$** , derived from the current 1024-bit block being processed.
- The output of the eightieth round is added to the input to the first round ( $H_{i-1}$ ) to produce  $H_i$ .

# Step – 5 Output

---

- After all  **$N$  1024-bit** blocks have been processed, the output from the  $N$ th stage is the **512-bit** message digest



	$h = g$
	$g = f$
	$f = e$
	$e = d + T_1$
	$d = c$
	$c = b$
	$b = a$
	$a = T_1 + T_2$

$$T_1 = h + \text{Ch}(e, f, g) + \left( \sum_{1}^{512} e \right) + W_t + K_t$$

$$T_2 = \left( \sum_{0}^{512} a \right) + \text{Maj}(a, b, c)$$

# SHA-512 Round Function

# SHA-512 Round Function Elements

---

- $\text{Maj}(a,b,c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$  **returns a result based on majority value among these inputs**
- $\Sigma(a) = \text{ROTR}(a,28) \text{ XOR } \text{ROTR}(a,34) \text{ XOR } \text{ROTR}(a,39)$  (**ROTR(a,28) means rotate right by 28 positions**)
- $\Sigma(e) = \text{ROTR}(e,14) \text{ XOR } \text{ROTR}(e,18) \text{ XOR } \text{ROTR}(e,41)$
- $+ = \text{addition modulo } 2^{64}$
- $K_t = \text{a 64-bit additive constant}$
- $W_t = \text{a 64-bit word derived from the current input block.}$

# Message Authentication Codes



# Outline

---

- Message Authentication Codes
- MAC requirements and security

# Message Authentication

---

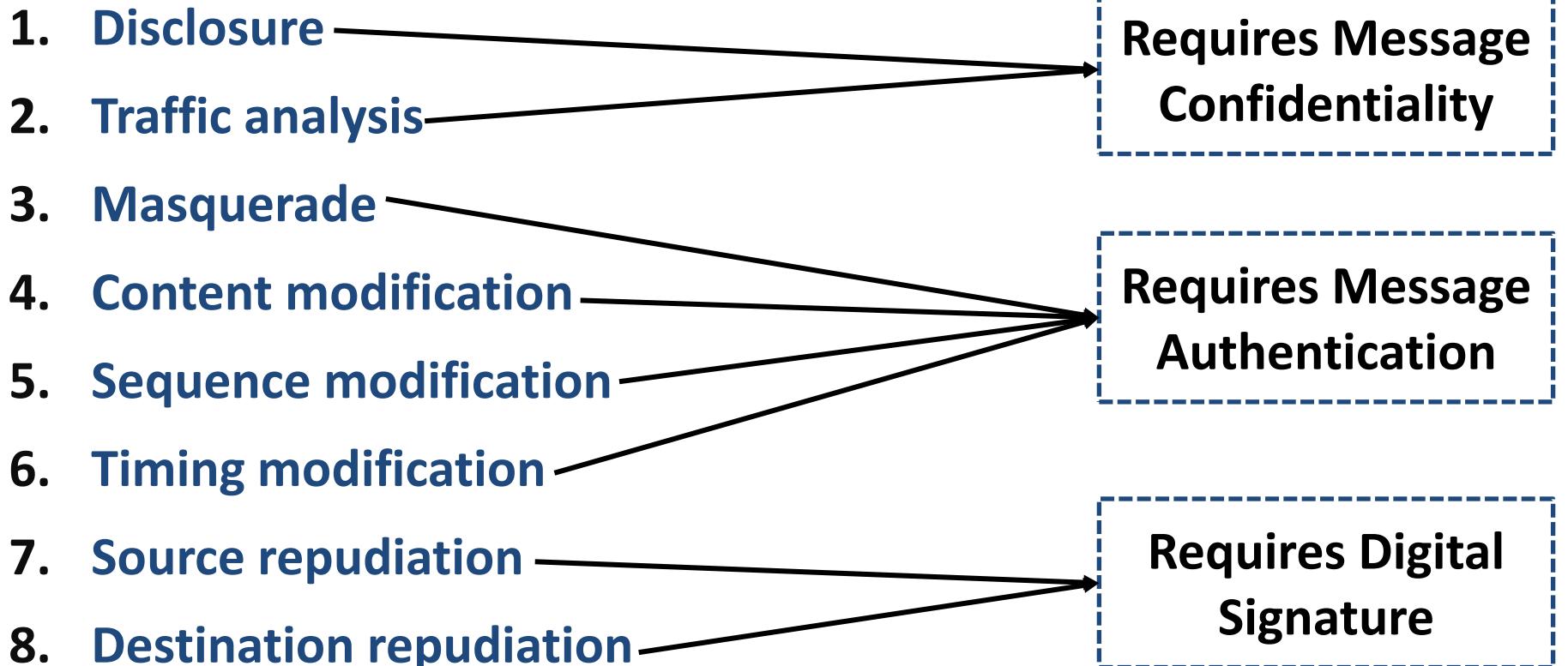
- **Message authentication** is a procedure to verify that received messages come from the genuine source and have not been altered.
- Message authentication may also verify sequencing and timeliness.
- Message authentication is a mechanism or service used to verify the **integrity of a message**.
- Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).

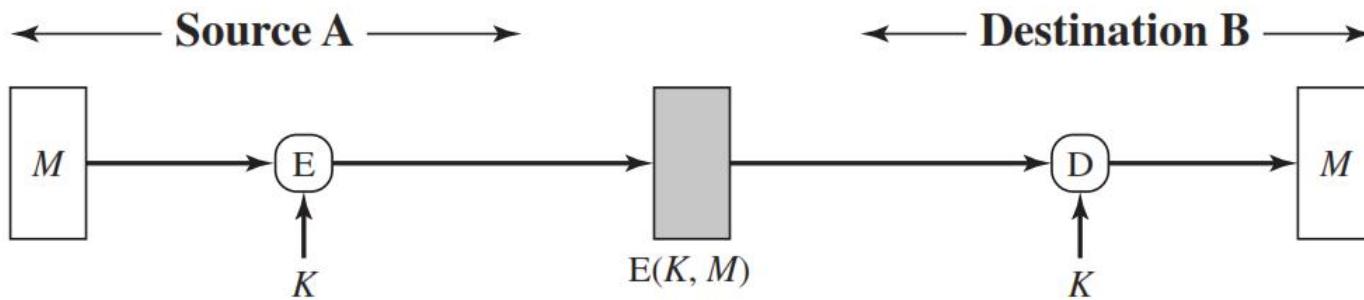
# Message Authentication Requirements

---

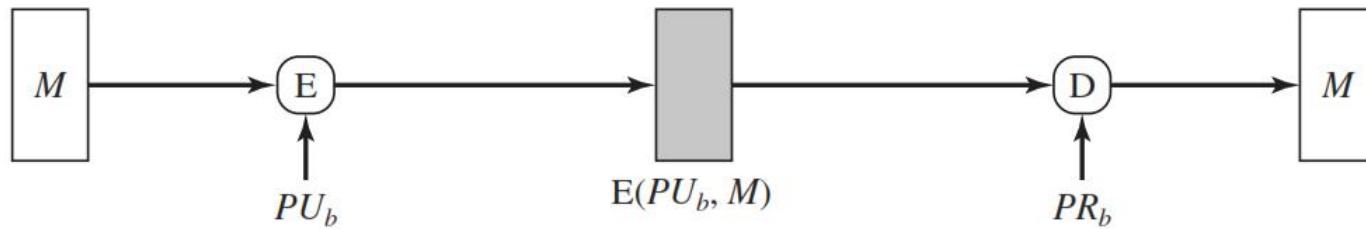
1. **Disclosure:** Disclosure of message contents
2. **Traffic analysis:** Discovery of the pattern of traffic between parties
3. **Masquerade:** Insertion of messages into the network from a fraudulent source
4. **Content modification:** Changes to the contents of a message
5. **Sequence modification:** Any modification to a sequence of messages between parties
6. **Timing modification:** Delay or replay of messages
7. **Source repudiation:** Denial of transmission of message by source
8. **Destination repudiation:** Denial of receipt of message by destination

# Message Authentication Requirements

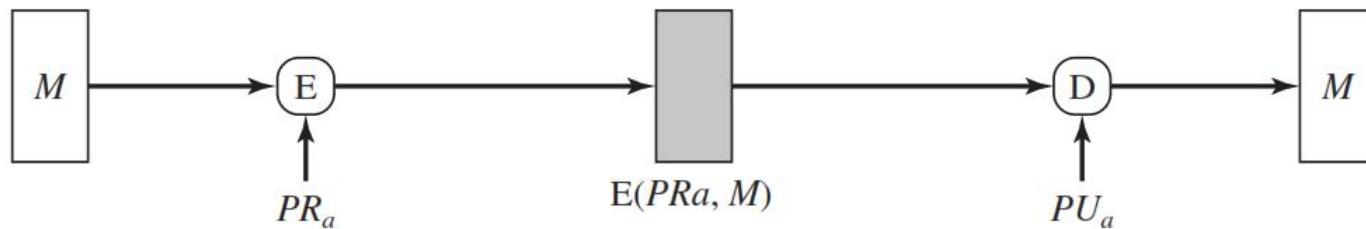




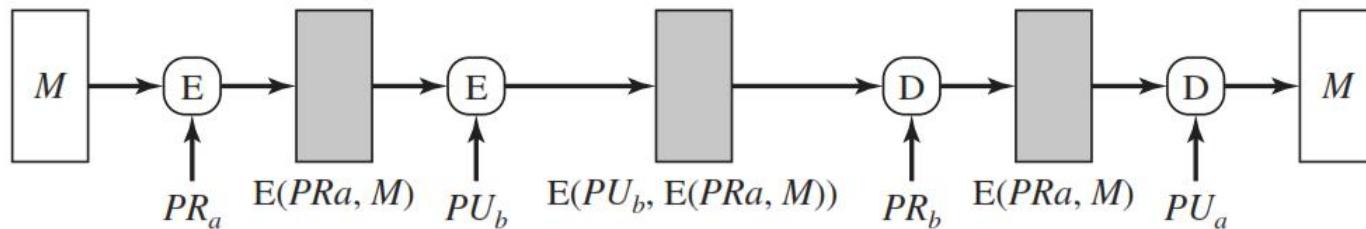
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

# Digital Signature

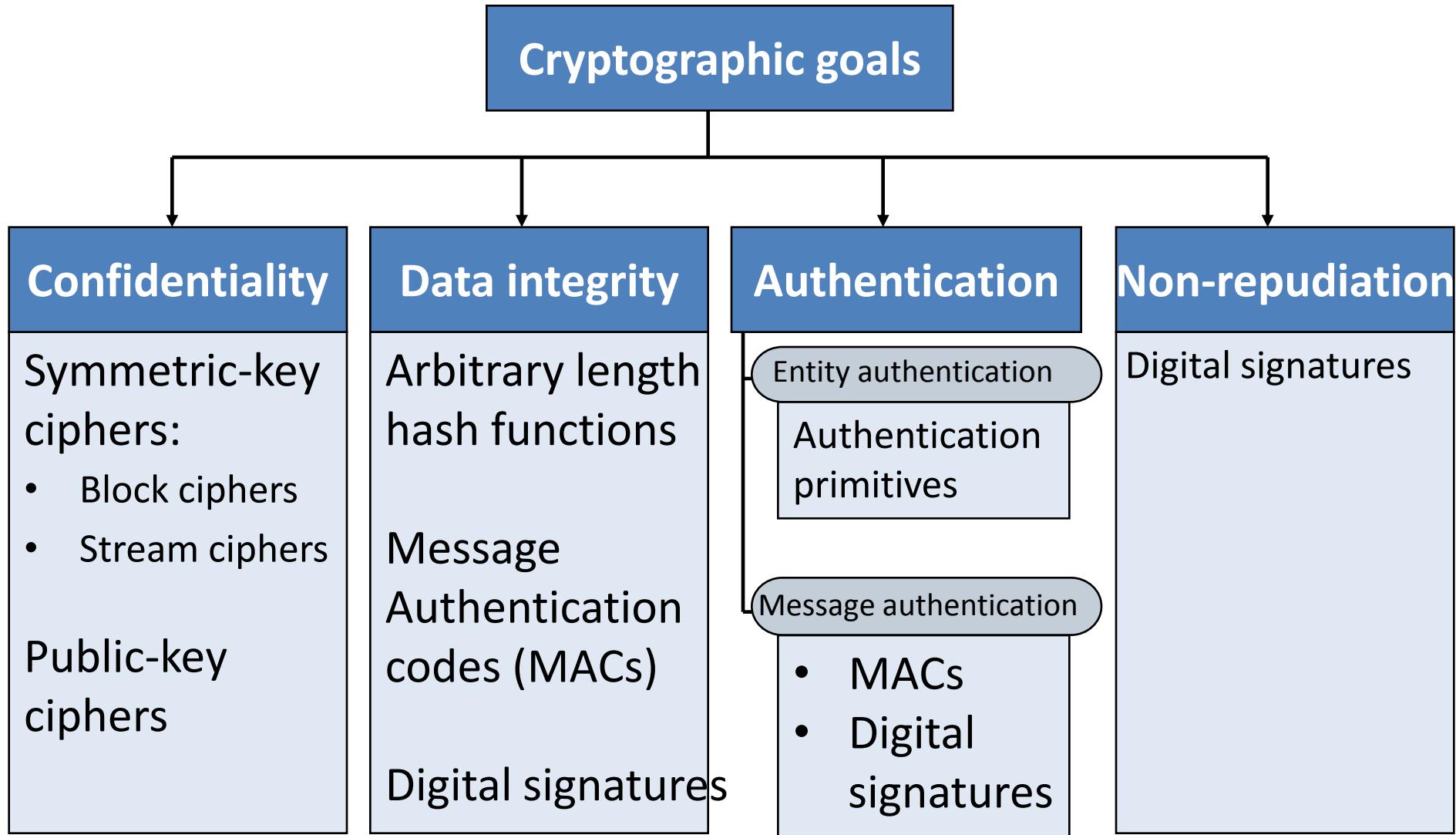


# Outline

---

- Digital Signature
- Digital Signature properties
- Requirements and security

# Cryptographic Goals

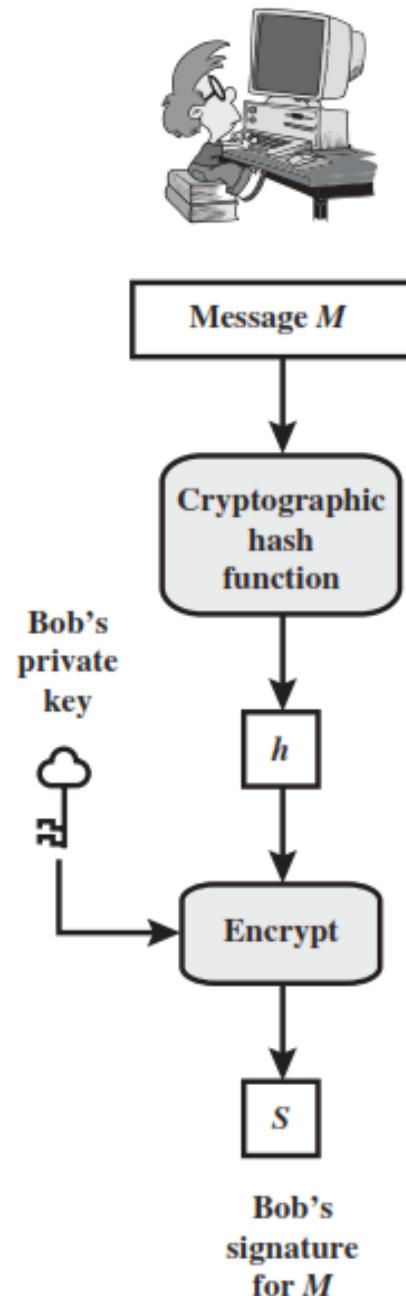


# Digital Signature

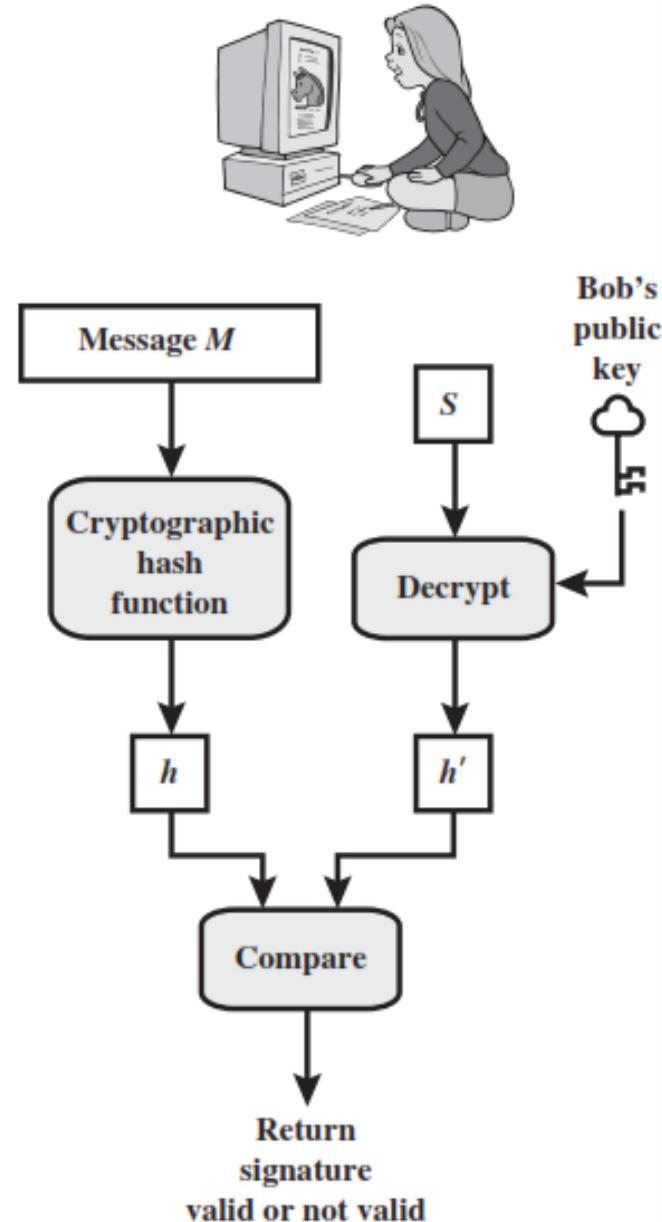
---

- A **digital signature** is an authentication mechanism that enables the creator of a message to attach a code that acts as a **signature**.
- Typically the **signature** is formed by taking the hash of the message and encrypting the message with the creator's private key.
- The **signature** guarantees the source and integrity of the message.
- The **digital signature standard (DSS)** is an NIST standard that uses the secure hash algorithm (SHA).

**Bob**



**Alice**



# Hash code, MAC and Digital Signature

---

## Hash Code

- A **hash** of the message, if appended to the message itself, only protects against accidental changes to the message. An attacker can modify the message and can simply calculate a new hash and use it instead of the original one. So this **only gives integrity**.

## MAC

- A message authentication code (MAC) (sometimes also known as keyed hash) protects against message forgery by anyone who doesn't know the secret key. Thus, we have both **integrity** and **authentication**, but **not non-repudiation**.

# Hash code, MAC and Digital Signature

---

## Digital Signature

- A **digital signature** is created with a private key, and verified with the corresponding public key of an asymmetric key-pair.
- Only the holder of the private key can create this signature, and normally anyone knowing the public key can verify it.

# Attacks and Forgeries on Digital Signature

---

- **Key-only attack:** C only knows A's public key.
- **Known message attack:** C is given access to a set of messages and their signatures.
- **Generic chosen message attack:** C chooses a list of messages before attempting to break A's signature scheme, independent of A's public key. C then obtains from A, valid signatures for the chosen messages. The attack is generic, because it does not depend on A's public key; the same attack is used against everyone.
- **Directed chosen message attack:** Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.
- **Adaptive chosen message attack:** C is allowed to use A as an “oracle.” This means A may request signatures of messages that depend on previously obtained message-signature pairs.

# ...Attacks and Forgeries on Digital Signature

---

- **Total break:** C determines A's private key.
- **Universal forgery:** C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
- **Selective forgery:** C forges a signature for a particular message chosen by C.
- **Existential forgery:** C succeeds in forging the signature of one message, not necessarily of his choice

# Digital Signature Requirements

---

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information unique to the sender to prevent both forgery and denial.
3. It must be relatively easy to produce the digital signature.
4. It must be relatively easy to recognize and verify the digital signature.
5. It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be practical to retain a copy of the digital signature in storage.

# An Insight to Cryptosystems & Authentication Protocols

---

By

**Dr. Shashidhar R**

Security & Blockchain Researcher,

**Samsung R&D Institute India**

# Security Services

- **Authentication** - assurance that communicating entity is the one claimed have both peer-entity & data origin authentication
- **Access Control** - prevention of the unauthorized use of a resource
- **Data Confidentiality** –protection of data from unauthorized disclosure
- **Data Integrity** - assurance that data received is as sent by an authorized entity
- **Non-Repudiation** - protection against denial by one of the parties in a communication
- **Availability** – resource accessible/usable

# Hash Functions

A Hash Function is a mathematical function that maps data of arbitrary size to a fixed-size output.

In Blockchain, Hash Functions are used to provide a unique and tamper-proof digital fingerprint of data.

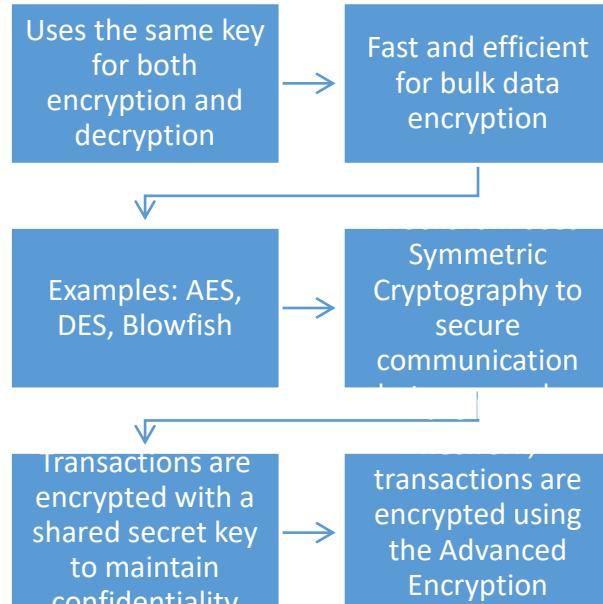
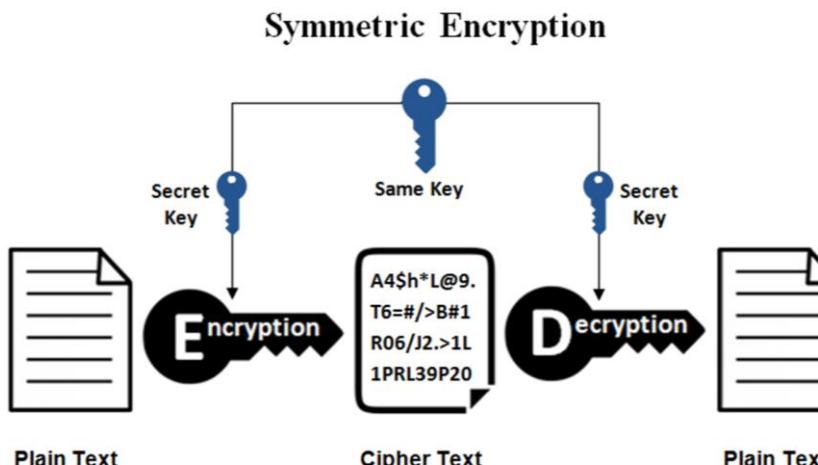
The SHA-256 (Secure Hash Algorithm 256-bit) is commonly used Hash Function in Blockchain.

In Blockchain, Hash Functions are used to create the Hash of each block's data, which includes transactions, timestamp, and a reference to the previous block's Hash.

The Hash of the current block is included in the next block, creating a chain of blocks that are linked together cryptographically.

Input	cryptographic hash function	Digest
Fox		<b>DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17</b>
The red fox jumps over the blue dog		<b>0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC</b>
The red fox jumps over the blue dog		<b>8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819</b>
The red fox jumps ovr the blue dog		<b>FCD3 7FDB 5AF2 C6FF 915F B401 C0A9 7D9A 46AF FB45</b>
The red fox jumps oer the blue dog		<b>8ACA D682 D588 4C75 4BF4 1799 7D88 ECF8 92B9 6A6C</b>

# Symmetric Cryptography



with a 256-bit key

# Public Key Cryptography

Public key cryptography is a type of asymmetric cryptography used in Blockchain.

It uses a pair of keys – a public key and a private key – for secure communication.

In Blockchain, public key cryptography is used for digital signatures.

The public key is used for encryption and is available to anyone, while the private key is kept secret and used for decryption.

Asymmetric Key Cryptography



Encryption



Encryption Key  
Public key



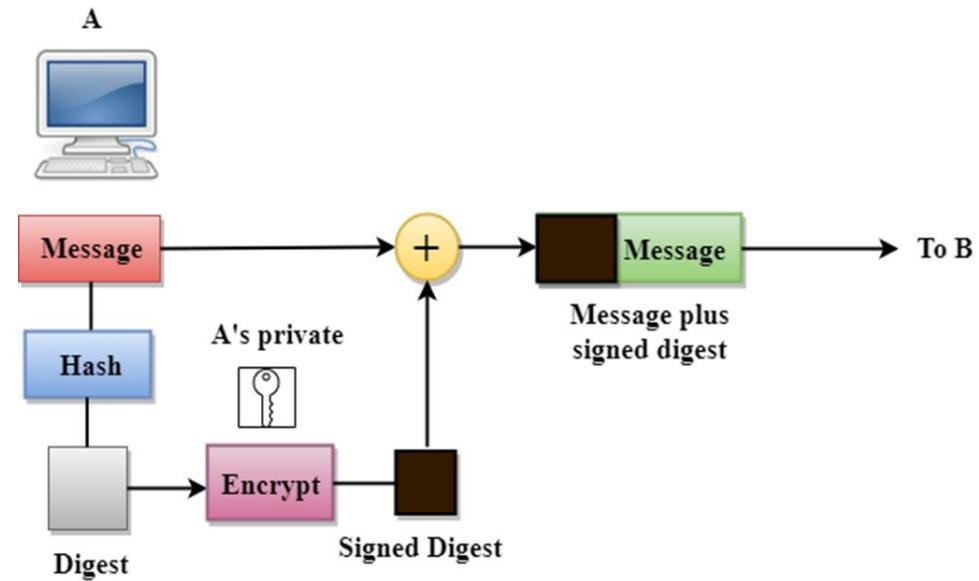
Decryption Key  
Private Key



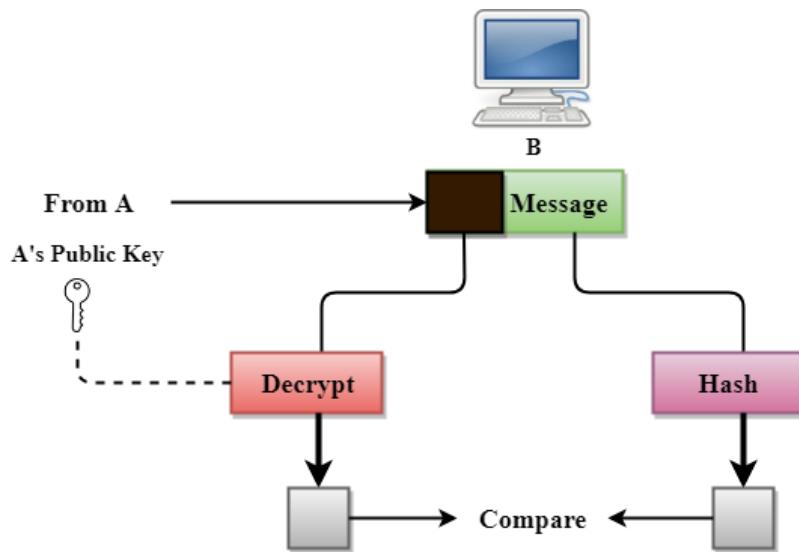
Decryption

# Digital Signatures Creation

- Importance of digital signatures in ensuring security and authenticity in blockchain transactions
- A digital signature is created using a private key and the message to be signed



# Digital Signature Verification



- A digital signature is verified using the corresponding public key and the original message
- The advantages of using digital signatures in blockchain, such as improved security, authenticity, and non-repudiation

# Message Authentication & Digital Signature



# Outline

---

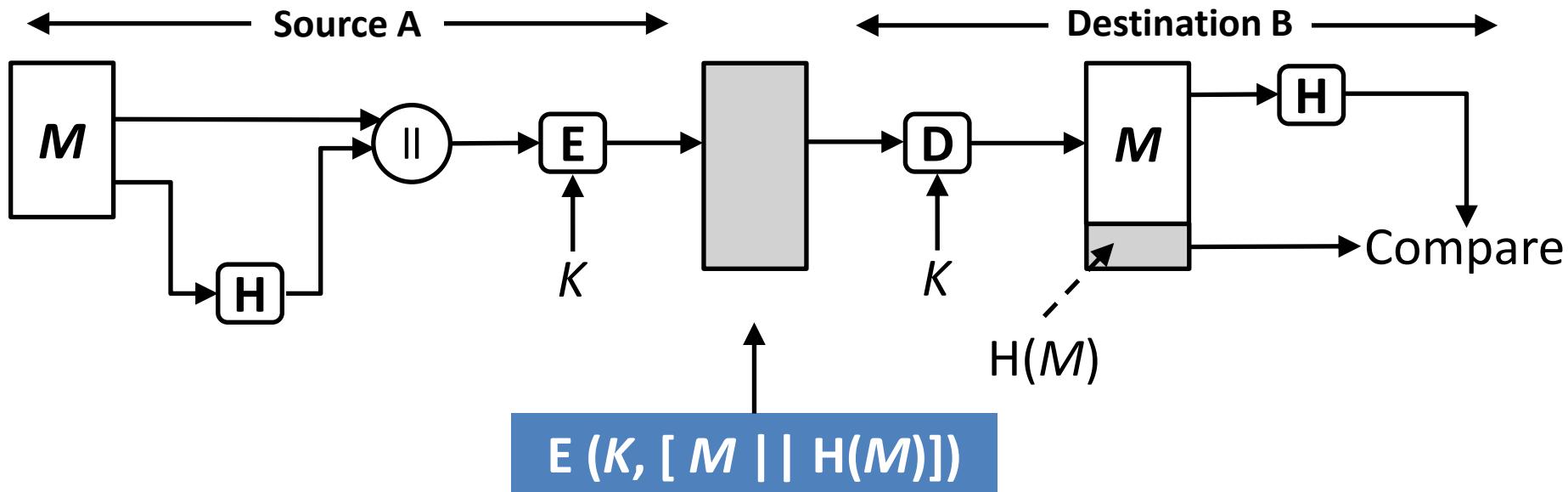
- Message authentication
- MAC, HMAC, DAA, CMAC
- Digital Signature
- Digital Signature Requirements
- RSA approach
- DSA Signing and Verifying
- Elgamal Fdigital Slnature

# 1. Message Authentication

---

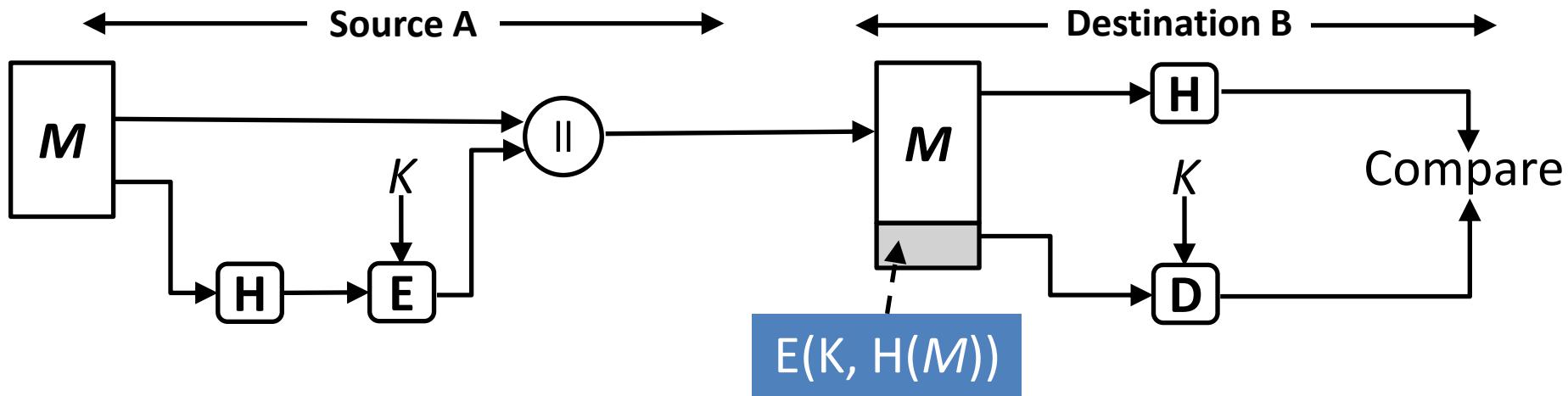
- **Message authentication** is a mechanism or service used to verify the integrity of a message.
- Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).
- When a hash function is used to provide message authentication, the **hash function value** is often referred to as a **message digest**.

# Message authentication method - 1



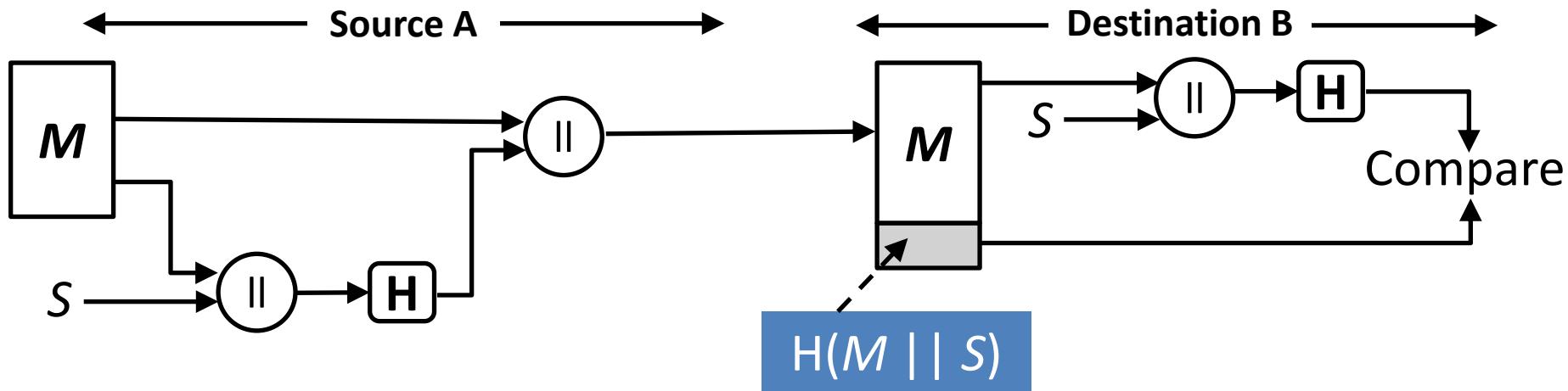
- Only A and B share the secret key, the message must have come from A and has not been altered.
- The hash code provides the structure required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

# Message authentication method - 2



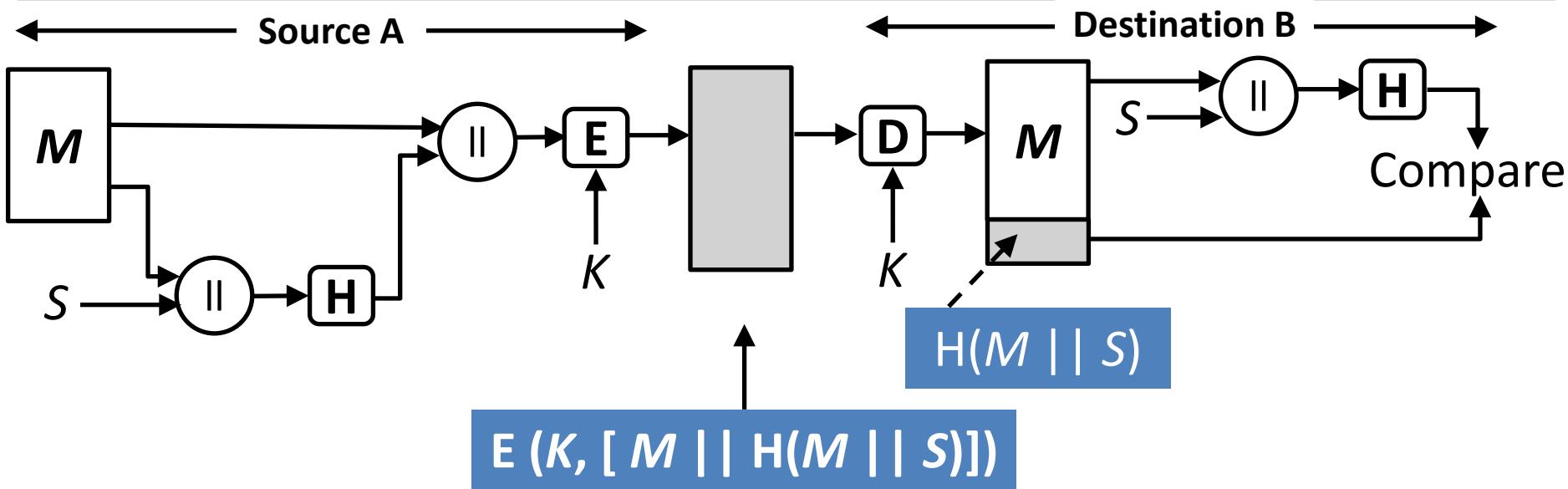
- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality.

# Message authentication method - 3



- It is possible to use a hash function but no encryption for message authentication.
- A and B share a common secret value  $S$ .
- A computes the hash value over the concatenation of  $M$  and  $S$  and appends the resulting hash value to  $M$ .
- Because B possesses  $S$ , it can recompute the hash value to verify.
- An opponent cannot modify an intercepted message.

# Message authentication method - 4



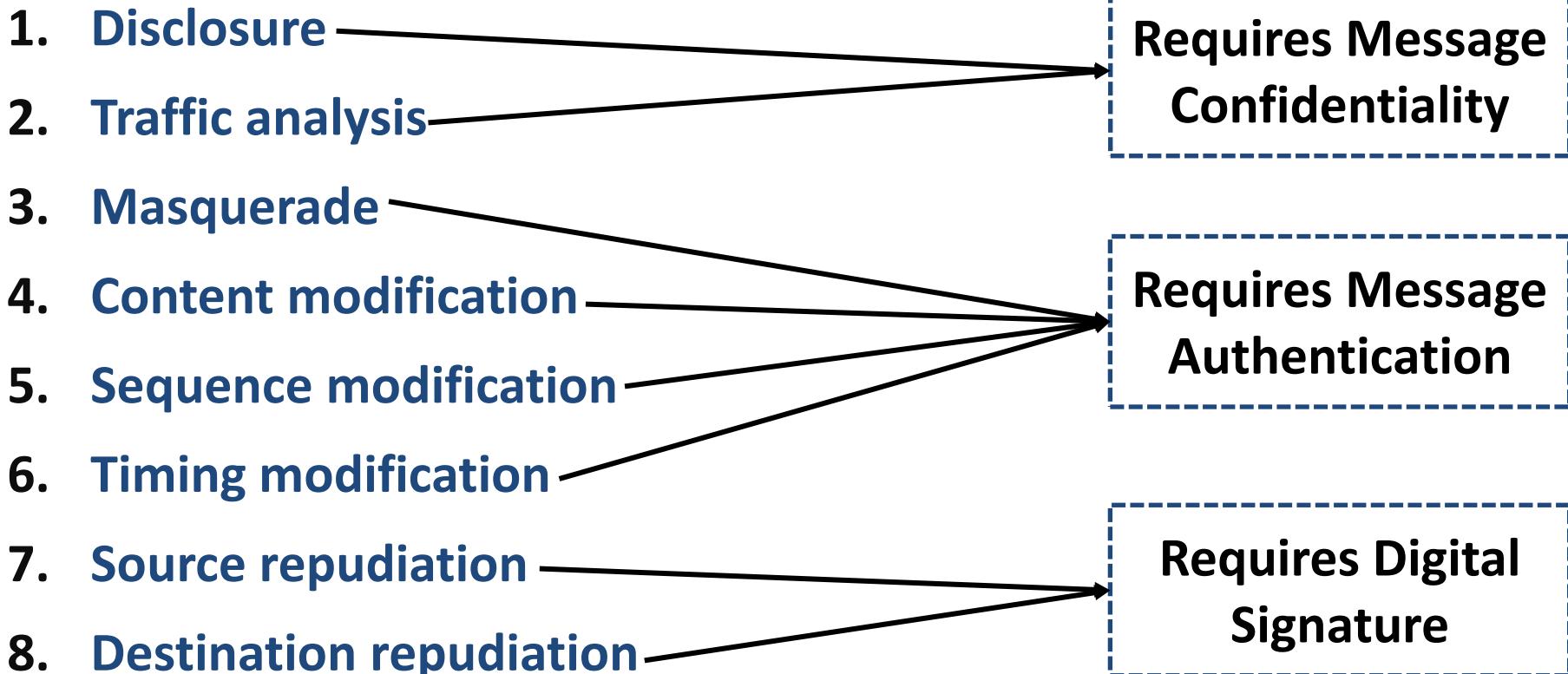
- Confidentiality can be added to the approach of method (3) by encrypting the entire message plus the hash code.

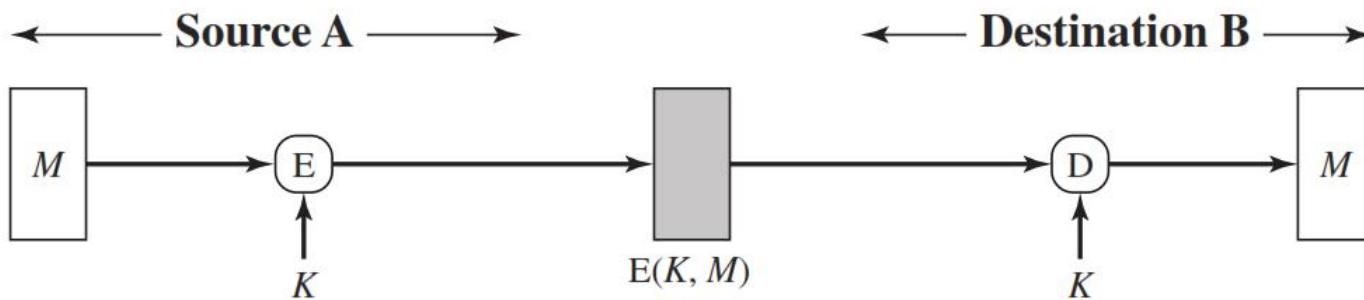
# MAC (Message Authentication Code)

---

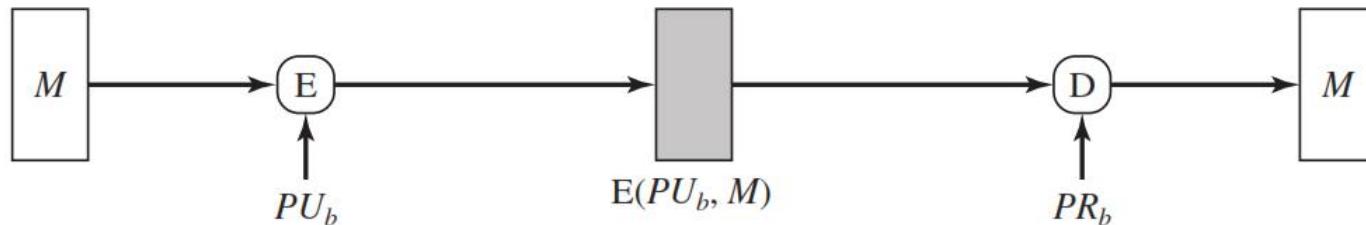
- More commonly, message authentication is achieved using a **MAC** also known as **keyed hash function**.
- MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.
- A **MAC** function takes as input a secret key and a data block and produces a hash value, referred to as the **MAC**.
- The combination of hashing and encryption results in an overall function that is, in fact, a MAC (Method -2 in previous slide).

# Message Authentication Requirements

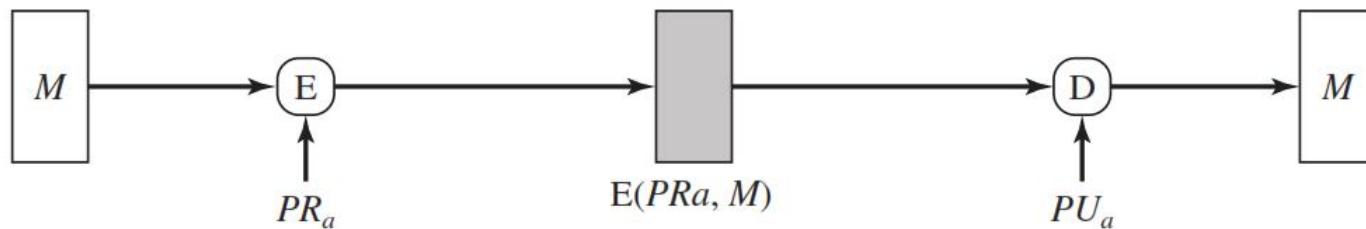




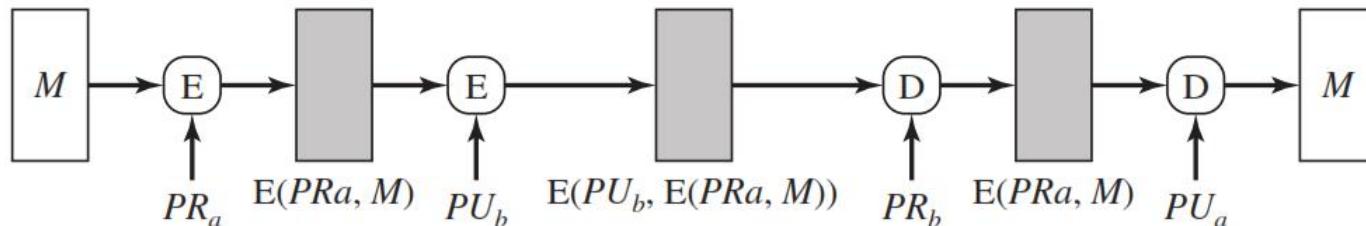
(a) Symmetric encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

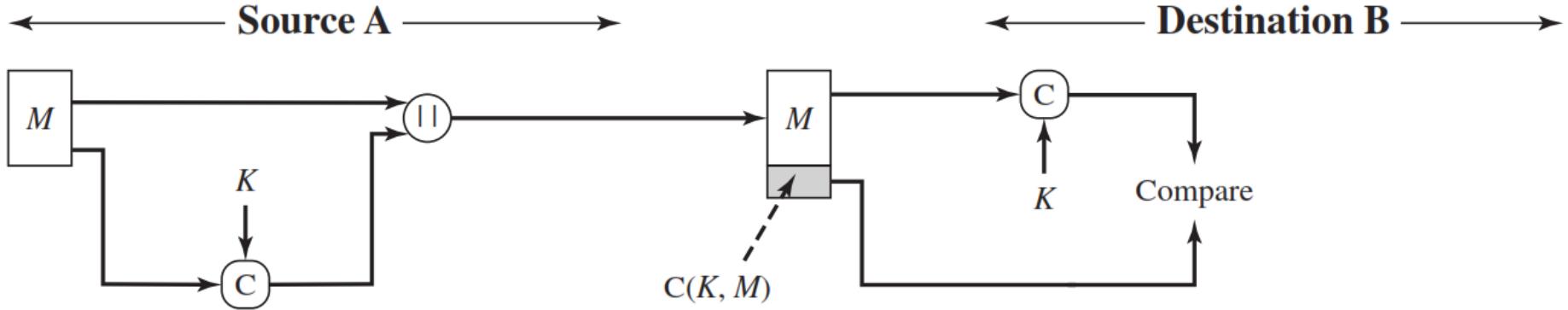
# Message Authentication Code

---

- An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or **MAC**
- MAC is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K.
- When A has a message to send to B, it calculates the MAC as a function of the message and the key

$$\text{MAC} = C(K, M)$$

# Message Authentication Code



(a) Message authentication

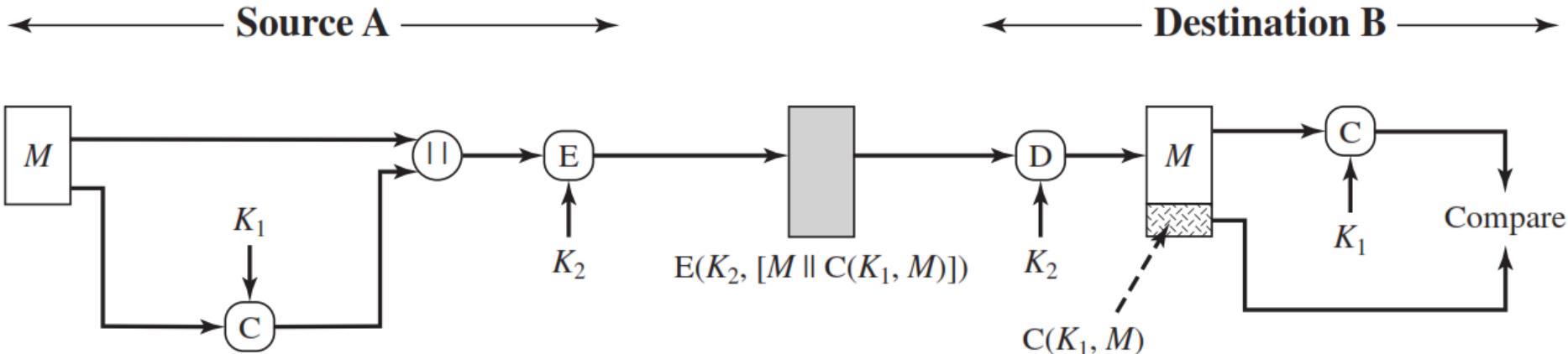
- The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.
- Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

# Message Authentication code - Cont...

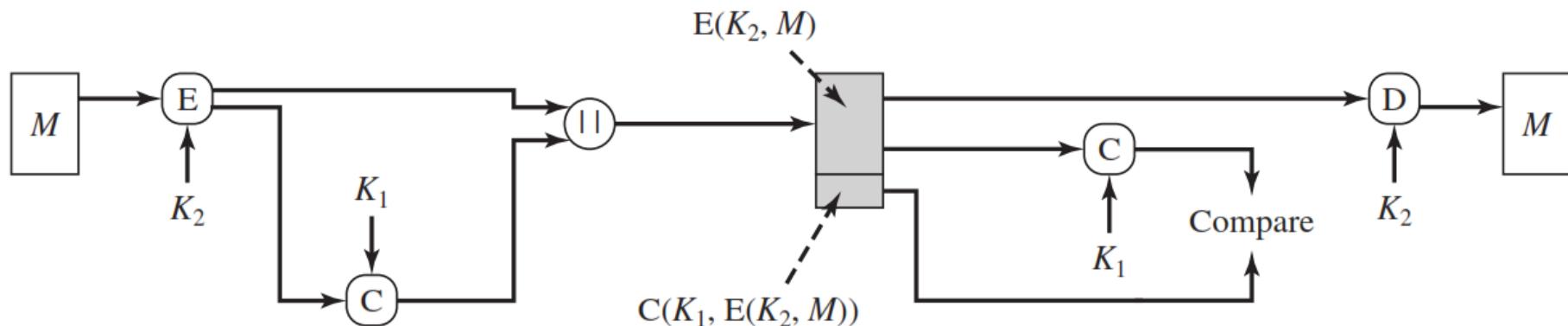
---

- The receiver is assured that the message is from the alleged sender.
- Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
- A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must be for decryption.
- In general, the MAC function is a many-to-one function. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys.
- If an  $n$ -bit MAC is used, then there are  $2^n$  possible MACs

# Message Authentication code - Cont...



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

# MAC Based on Hash Functions - HMAC

---

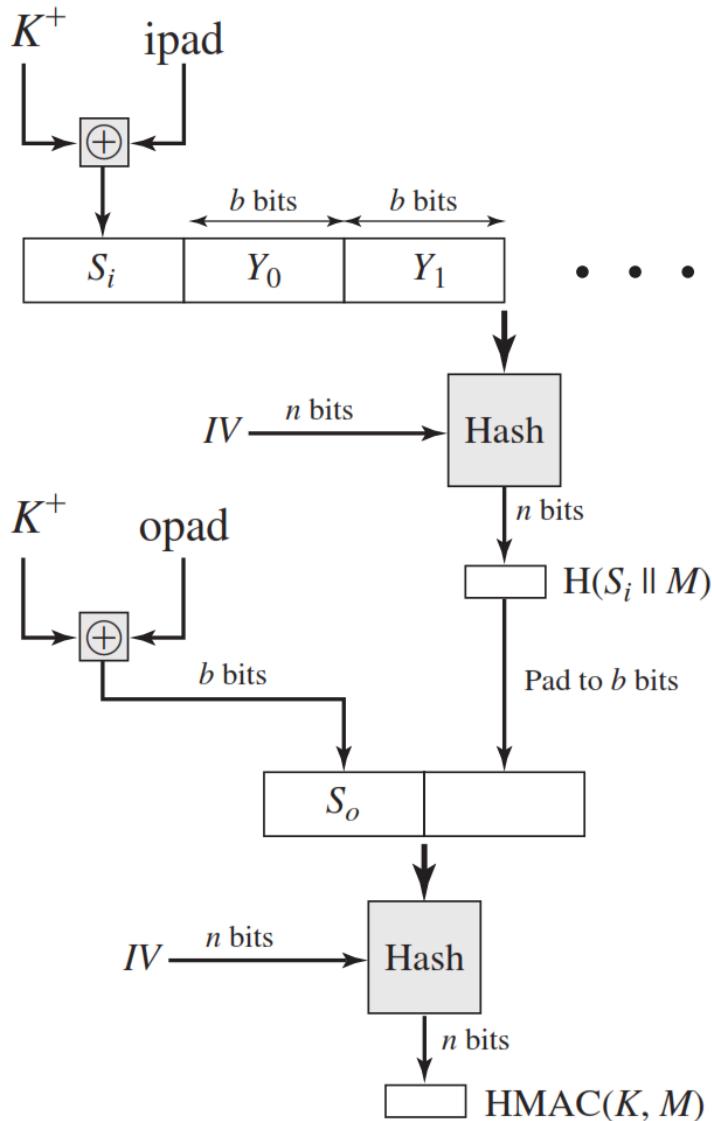
- Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES.
- Library code for cryptographic hash functions is widely available.

# Design objectives for HMAC

---

- To use, without modifications, available hash functions.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

# HMAC Structure



1. Append zeros to the left end of  $K$  to create a  $b$ -bit string  $K^+$
2. XOR  $K^+$  with  $\text{ipad}$  to produce the  $b$ -bit block  $S_i$ .
3. Append  $M$  to  $S_i$ .
4. Apply  $H$  to the stream generated in step 3.
5. XOR  $K^+$  with  $\text{opad}$  to produce the  $b$ -bit block  $S_o$ .
6. Append the hash result from step 4 to  $S_o$ .
7. Apply  $H$  to the stream generated in step 6 and output the result.

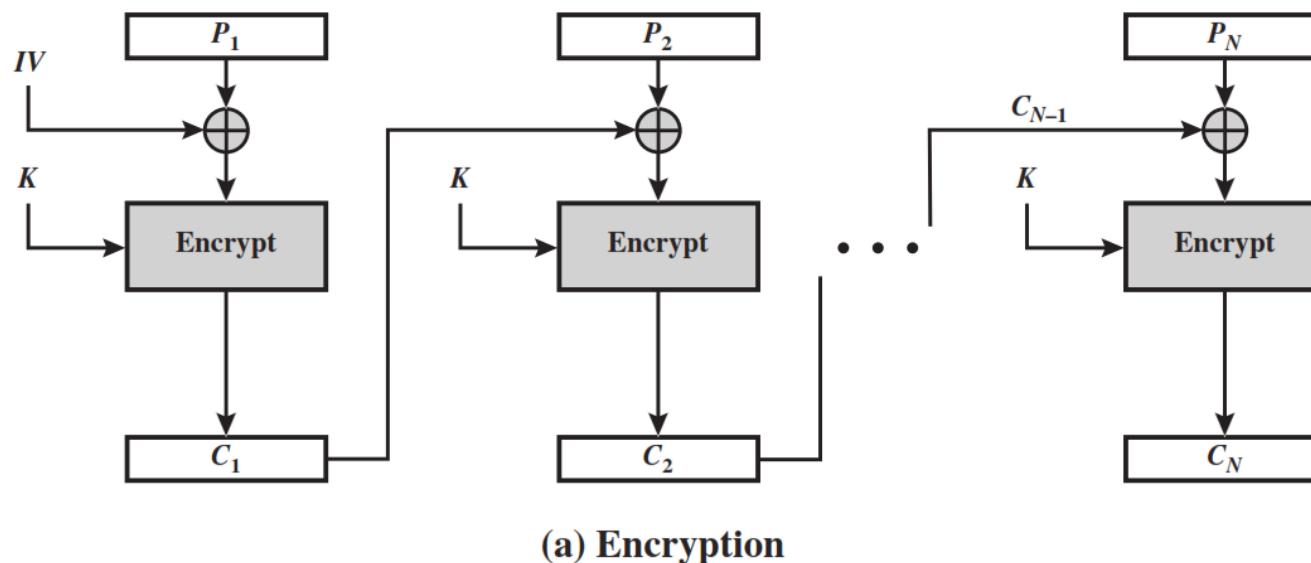
# HMAC Structure

---

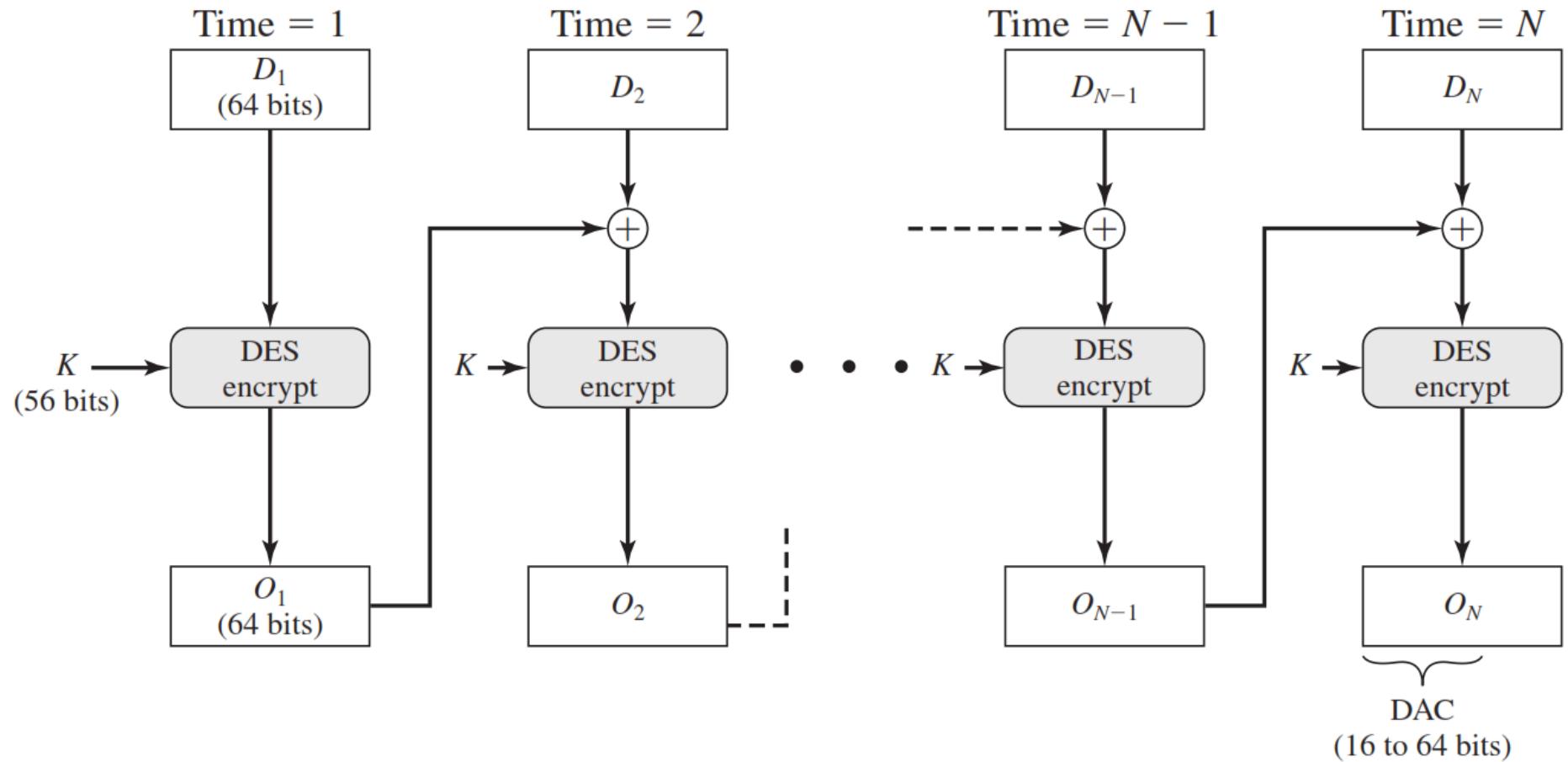
- $H$  = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)
- $IV$  = initial value input to hash function
- $M$  = message input to HMAC
- $Y_i$  =  $i^{\text{th}}$  block of  $M$
- $L$  = number of blocks in  $M$
- $n$  = length of hash code produced by embedded hash function
- $K^+$  =  $K$  padded with zeros on the left so that the result is  $b$  bits in length
- $\text{ipad} = 00110110$  (36 in hexadecimal) repeated  $b/8$  times
- $\text{opad} = 01011100$  (5C in hexadecimal) repeated  $b/8$  times

# MAC based on Block Ciphers

- The **Data Authentication Algorithm** (DAA), based on DES, has been one of the most widely used MACs for a number of years.
- The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES (Figure 6.4) with an initialization vector of zero.



# Data Authentication Algorithm (DAA)



# Data Authentication Algorithm (DAA)

---

- The data (e.g., message, record, file, or program) to be authenticated are grouped into contiguous 64-bit blocks:
- $D_1, D_2, \dots, D_n$ . If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm  $E$  and a secret key  $K$ , **a data authentication code (DAC)** is calculated as follows

$$O_1 = E(K, D)$$

$$O_2 = E(K, [D_2 \oplus O_1])$$

$$O_3 = E(K, [D_3 \oplus O_2])$$

.

.

.

$$O_N = E(K, [D_N \oplus O_{N-1}])$$

# Cipher-Based Message Authentication Code (CMAC)

---

- **Cipher-based Message Authentication Code (CMAC)** mode of operation for use with AES and triple DES.
- First, let us define the operation of CMAC when the message is an integer multiple  $n$  of the cipher block length  $b$ . For AES,  $b = 128$ , and for triple DES,  $b = 64$ . The message is divided into  $n$  blocks ( $M_1, M_2, \dots, M_n$ )

# Cipher-Based Message Authentication Code (CMAC)

---

- The algorithm makes use of a k-bit encryption key K and a b-bit constant, K1.
- For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits.
- CMAC is calculated as follows

$$C_1 = E(K, M_1)$$

$$C_2 = E(K, [M_2 \oplus C_1])$$

$$C_3 = E(K, [M_3 \oplus C_2])$$

•

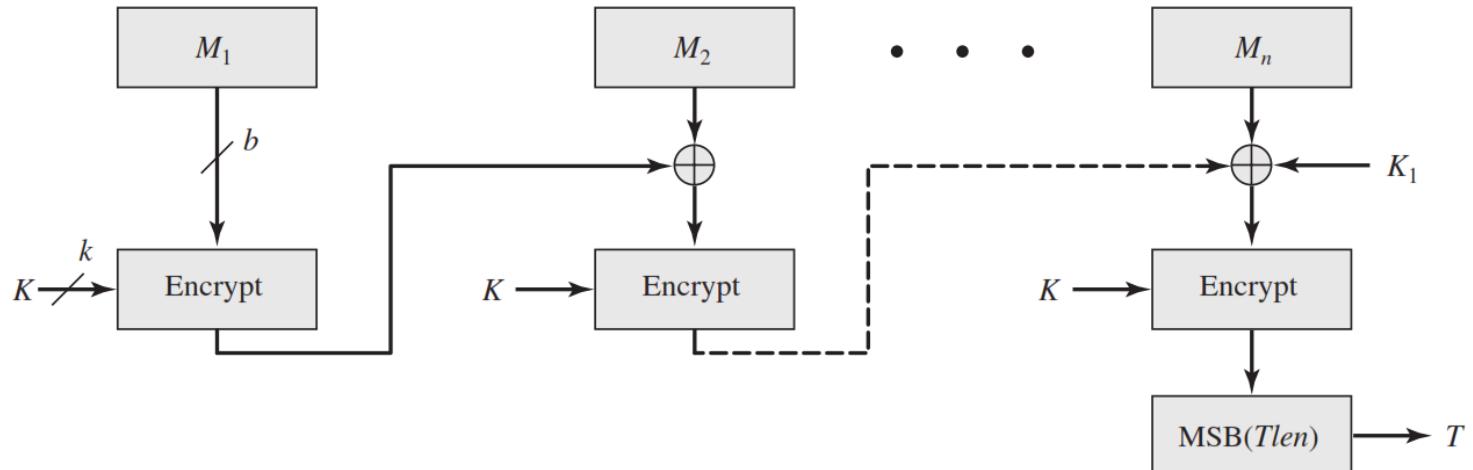
•

•

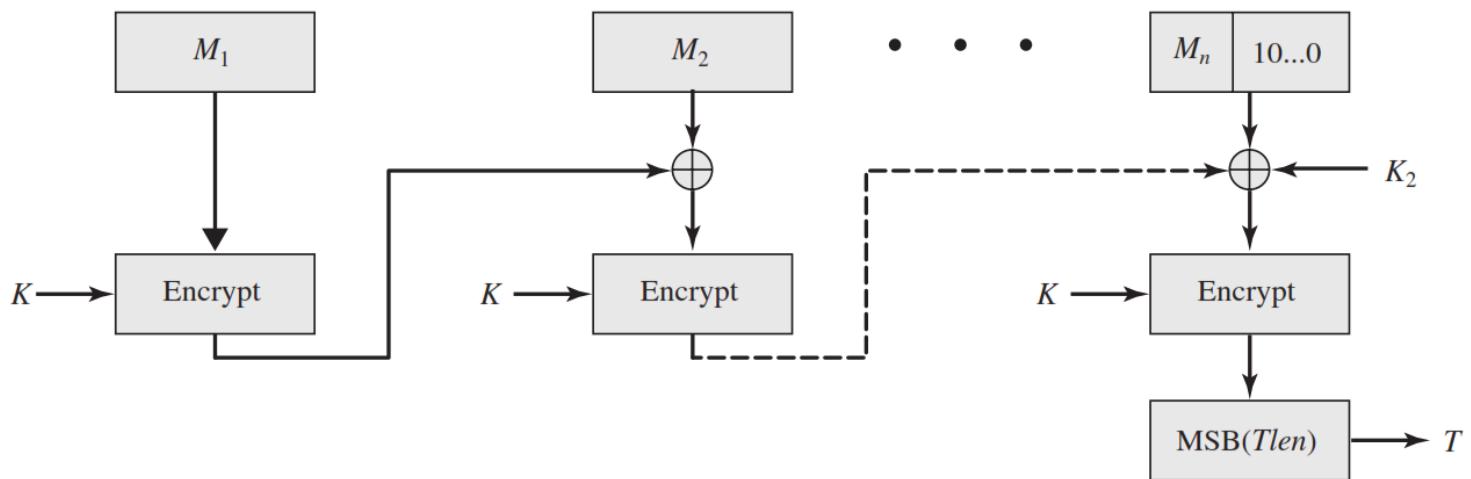
$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$

$$T = \text{MSB}_{T\text{len}}(C_n)$$

# Cipher-Based Message Authentication Code (CMAC)



(a) Message length is integer multiple of block size



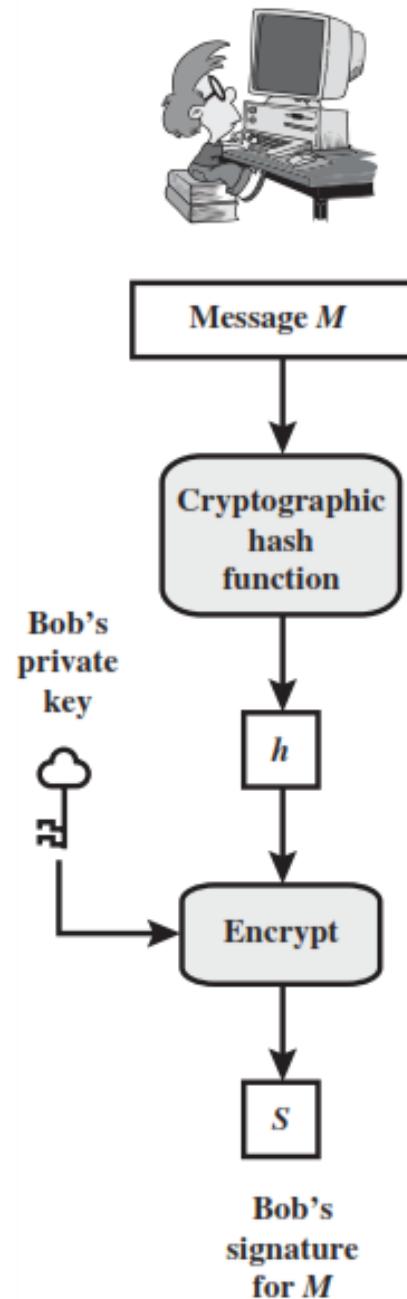
(b) Message length is not integer multiple of block size

# Digital Signature

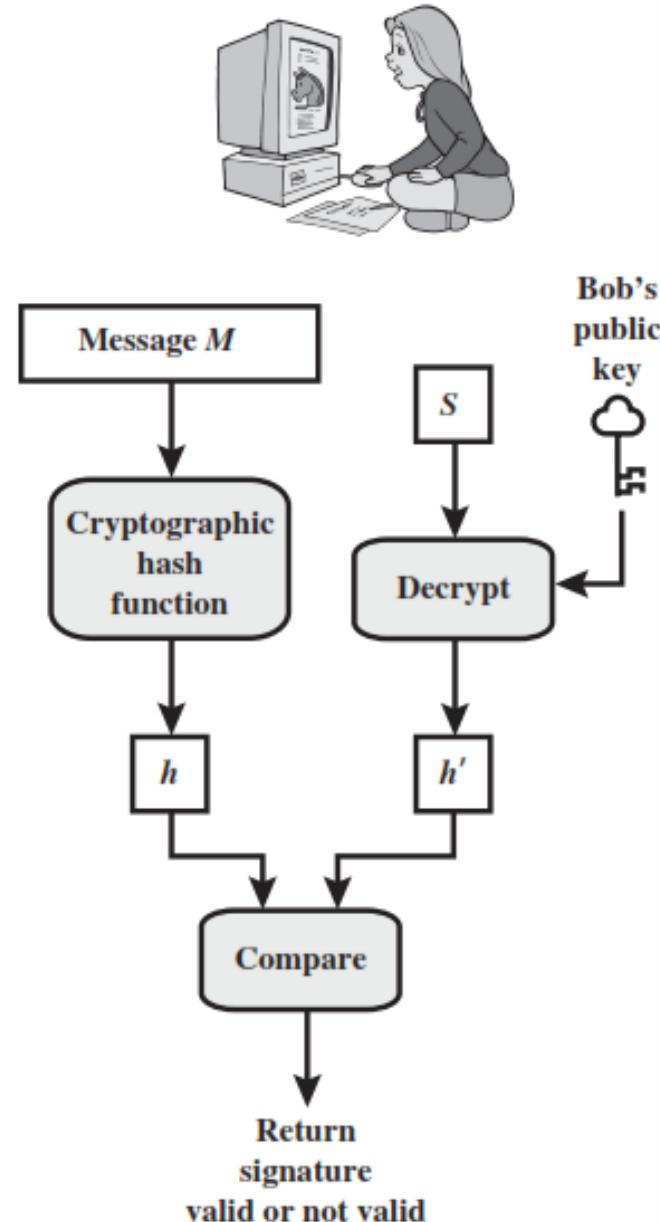
---

- A **digital signature** is an authentication mechanism that enables the creator of a message to attach a code that acts as a **signature**.
- Typically the **signature** is formed by taking the hash of the message and encrypting the message with the creator's private key.
- The **signature** guarantees the source and integrity of the message.
- The **digital signature standard (DSS)** is an NIST standard that uses the secure hash algorithm (SHA).

**Bob**



**Alice**



# Digital Signature Requirements

---

1. The signature must be a **bit pattern** that depends on the message being signed.
2. The signature must use some information **unique** to the sender to prevent both forgery and denial.
3. It must be relatively **easy to produce** the digital signature.
4. It must be relatively **easy to recognize** and **verify** the digital signature.
5. It must be computationally **infeasible to forge** a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be **practical to retain a copy** of the digital signature in storage.

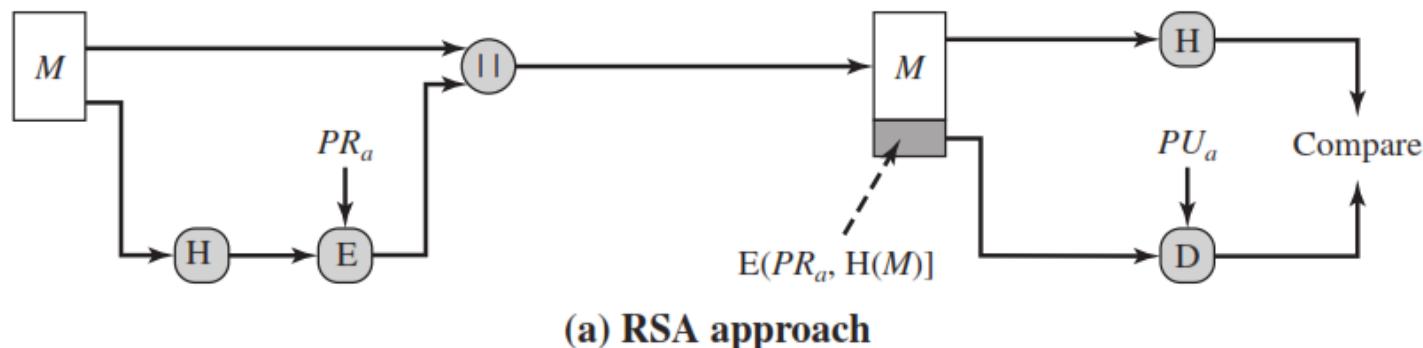
# Digital Signature Standard / DSA

---

- The **DSS** uses an algorithm that is designed to provide only the digital signature function.
- Unlike RSA, it cannot be used for encryption or key exchange.

# RSA Approach

- In the **RSA** approach, the message to be signed is input to a hash function that produces a **secure hash code** of fixed length.
- This hash code is then encrypted using the sender's private key to form the **signature**.
- Both the message and the signature are then transmitted.
- The recipient takes the message and produces a **hash code**.



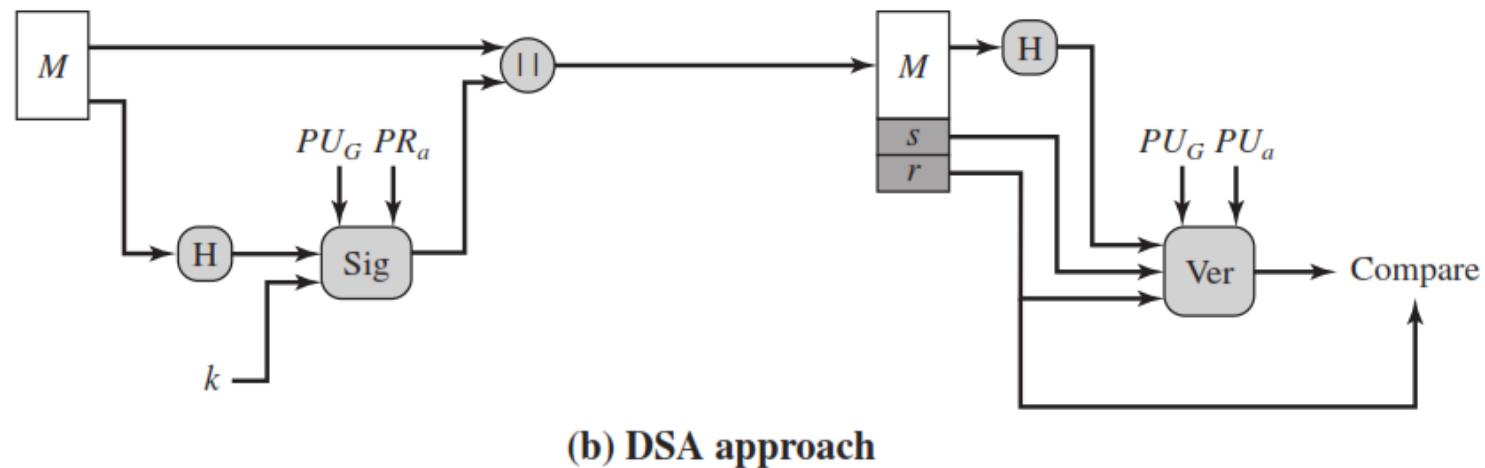
# RSA Approach

---

- The recipient also decrypts the signature using the sender's public key.
- If the calculated hash code matches the decrypted signature, the signature is accepted as valid.
- Because only the sender knows the private key, only the sender could have produced a valid signature.

# DSA Approach

- The **hash code** is provided as input to a **signature function** along with a random number  **$k$**  generated for this particular signature.
- The signature function also depends on the sender's private key ( **$PR_a$** ) and a set of parameters known to a group of communicating principals.
- We can consider this set to constitute a global public key ( **$PU$** )
- The result is a signature consisting of two components, labelled  **$s$**  and  **$r$** .



# DSA Approach

---

- At the receiving end, the hash code of the incoming message is generated.
- This plus the signature is input to a **verification function**.
- The verification function also depends on the global public key as well as the sender's public key ( **$PU_a$** ), which is paired with the sender's private key.
- The output of the verification function is a value that is equal to the signature component  **$r$**  if the signature is valid.
- *The signature* function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

# Digital Signature Algorithm

## Global Public-Key Components

- p prime number where  $2^{L-1} < p < 2^L$   
for  $512 \leq L \leq 1024$  and  $L$  a multiple of 64;  
i.e., bit length of between 512 and 1024 bits  
in increments of 64 bits
- q prime divisor of  $(p - 1)$ , where  $2^{N-1} < q < 2^N$   
i.e., bit length of  $N$  bits
- g  $= h(p - 1)/q \bmod p$ ,  
where  $h$  is any integer with  $1 < h < (p - 1)$   
such that  $h^{(p-1)/q} \bmod p > 1$

# Digital Signature Algorithm

## User's Private Key

$x$  random or pseudorandom integer with  $0 < x < q$

## User's Public Key

$y = g^x \text{ mod } p$

## User's Per-Message Secret Number

$k$  random or pseudorandom integer with  $0 < k < q$

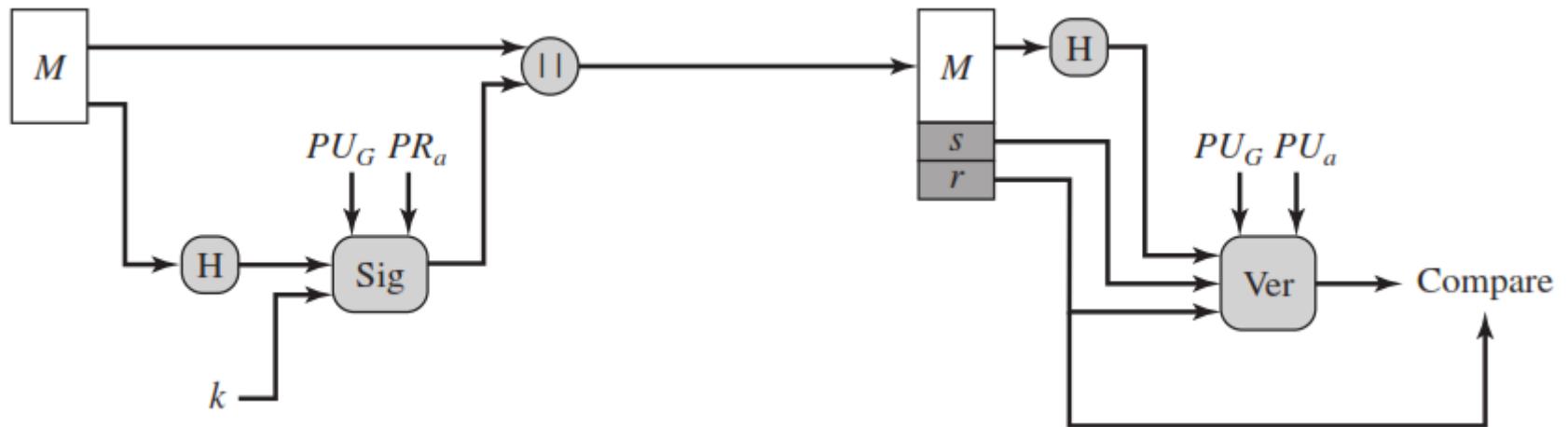
# Digital Signature Algorithm

## Signing

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1} (H(M) + xr)] \bmod q$$

Signature =  $(r, s)$



# Digital Signature Algorithm

## Verifying

$$w = (s')^{-1} \bmod q$$

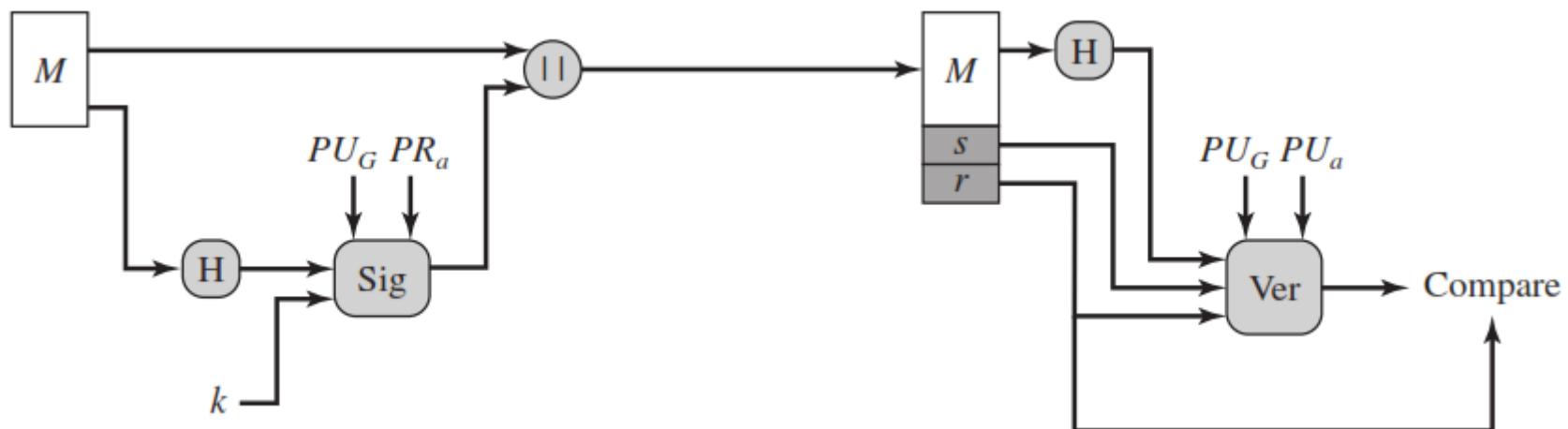
$$u_1 = [H(M')w] \bmod q$$

$$u_2 = (r')w \bmod q$$

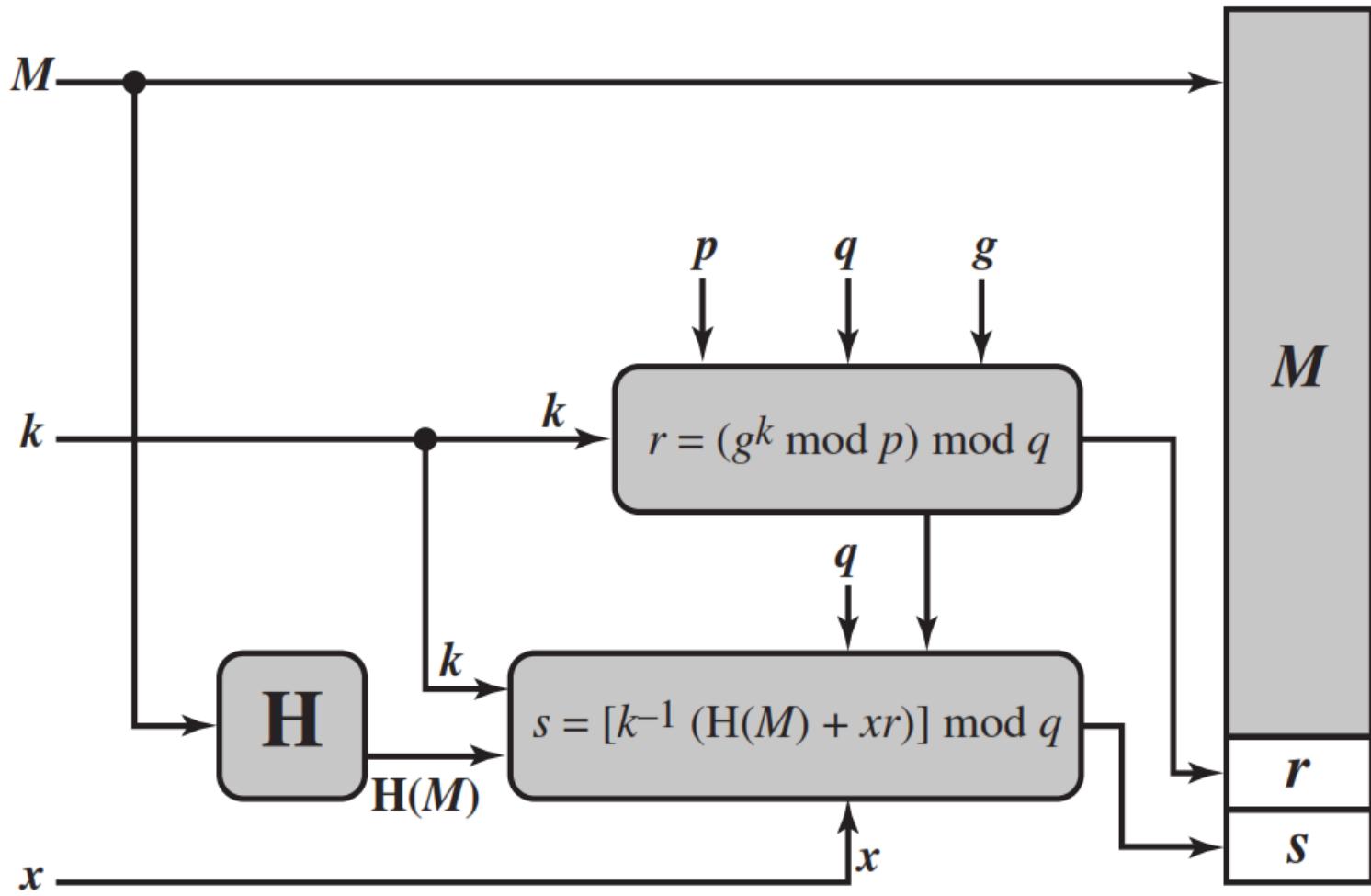
$$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$$

$$\text{TEST: } v = r'$$

$M$  = message to be signed  
 $H(M)$  = hash of  $M$  using SHA-1  
 $M', r', s'$  = received versions of  $M, r, s$

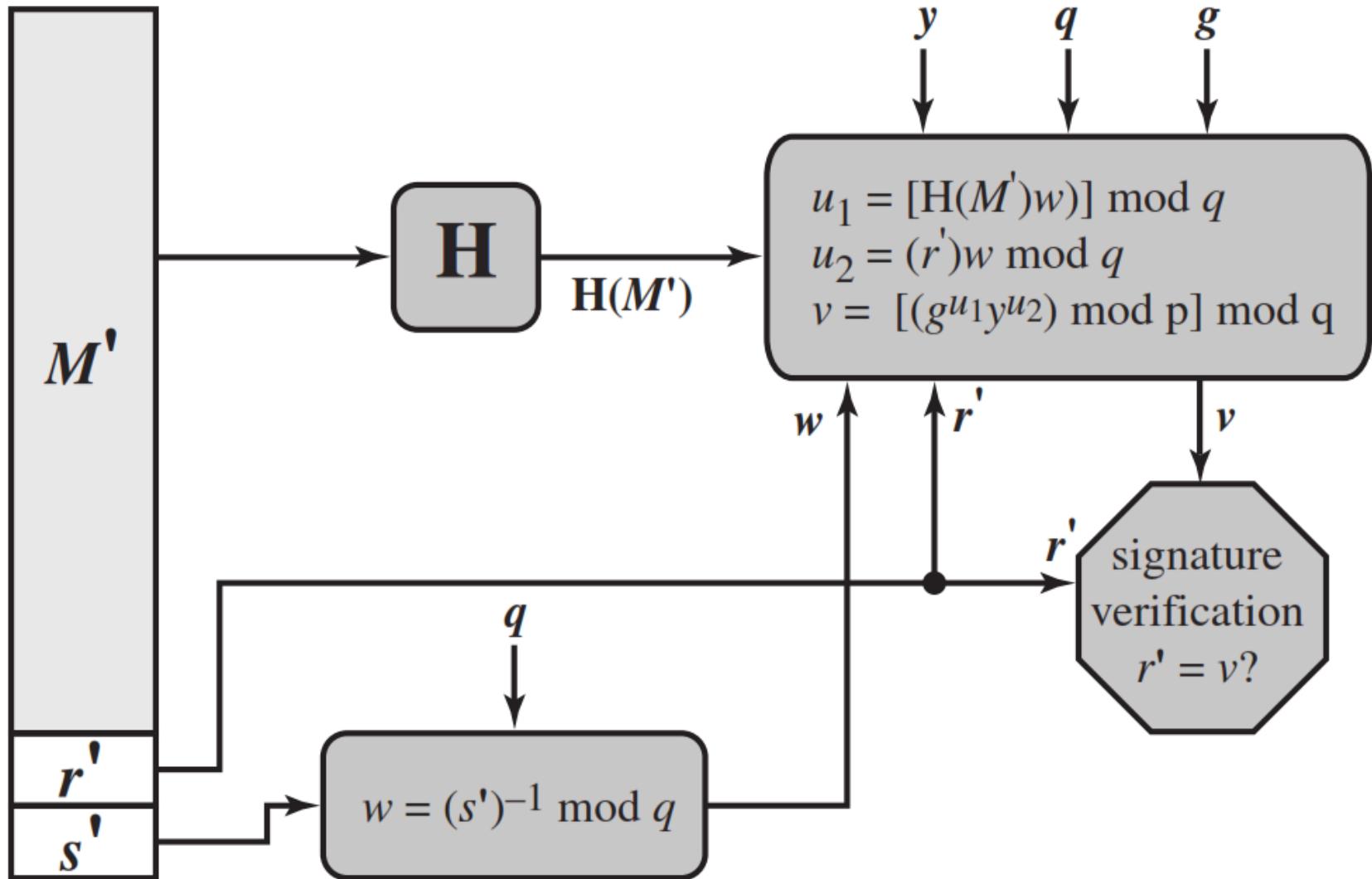


# DSA Signing



(a) Signing

# DSA Verifying



# ElGamal Digital Signatures

---

- Uses private key for encryption (signing)
- Uses public key for decryption (verification)
- Each user (eg. A) generates their key
  - chooses a secret key (number):  $1 < x_A < q-1$
  - compute their **public key**:  $y_A = a^{x_A} \text{ mod } q$

# ElGamal Digital Signature

---

- Alice signs a message  $M$  to Bob by computing
  - the hash  $m = H(M)$ ,  $0 \leq m \leq (q-1)$
  - chose random integer  $K$  with  $1 \leq K \leq (q-1)$  and  $\gcd(K, q-1) = 1$
  - compute temporary key:  $S_1 = a^k \pmod{q}$
  - compute  $K^{-1}$  the inverse of  $K \pmod{(q-1)}$
  - compute the value:  $S_2 = K^{-1} (m - x_A S_1) \pmod{(q-1)}$
  - signature is:  $(S_1, S_2)$
- Any user B can verify the signature by computing
  - $V_1 = a^m \pmod{q}$
  - $V_2 = y_A^{S_1} S_1^{S_2} \pmod{q}$
  - Signature is valid if  $V_1 = V_2$

# ElGamal Signature Example

---

- Use field GF(19)  $q=19$  and  $a=10$
- Alice computes her key:
  - A chooses  $x_A=16$  & computes  $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash  $m=14$  as  $(3, 4)$ :
  - choosing random  $K=5$  which has  $\gcd(18, 5)=1$
  - computing  $S_1 = 10^5 \bmod 19 = 3$
  - finding  $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing  $S_2 = 11(14-16 \cdot 3) \bmod 18 = 4$
- Any user B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3 \cdot 3^4 = 5184 = 16 \bmod 19$
  - since  $16 = 16$  signature is valid