

B.Sc. (Hons) Computer Applications  
Cyber Security Course number  
CABX)6003

# Unit 2

# Syllabus

## 1. Symmetric Ciphers

- Classical Encryption Techniques
- Block Ciphers and the Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)
- Block Cipher Operation Modes
- Basic Concepts in Number Theory and Finite Fields,

## 2. Asymmetric Ciphers

- Introduction to Public-Key Cryptography
- RSA Algorithm
- Diffie-Hellman Key Exchange Protocol
- ElGamal Cryptosystem
- Elliptic Curve Cryptography (ECC)

# Syllabus

## **3. Quantum Cryptography**

- Introduction to Quantum Cryptography
- Quantum Key Distribution (QKD)
- Quantum Physics and Photon Reception
- Cryptography with Photons
- Implementation of Quantum Cryptographic Systems

## **4. Cryptographic Hash Functions**

- Introduction to Hash Functions
- Common Hash Algorithms
- Applications of Hash Functions

# Syllabus

## **5. Message Authentication Codes (MACs)**

- Concept of Message Authentication
- Types of MACs
- Working of HMAC and CMAC

## **6. Digital Signatures**

- Introduction to Digital Signatures
- Working of Digital Signature Algorithms
- Common Digital Signature Algorithms
- Applications of Digital Signatures

# Symmetric Ciphers

Symmetric encryption is a cryptographic method where the **same key** is used for both the **encryption** of plaintext and the **decryption** of ciphertext. This key must be kept secret between the sender and the receiver to maintain the confidentiality of the data.

- **Key Features:**

- **Same Key for Encryption and Decryption:** The primary characteristic is the use of a single secret key.
- **Fast and Efficient:** Ideal for encrypting large amounts of data due to lower computational overhead compared to asymmetric encryption.
- **Key Distribution Challenge:** Securely sharing the secret key is a significant challenge.

- **Real-Life Examples:**

- **Wi-Fi Security:** WPA2 (Wi-Fi Protected Access 2) encryption in routers.
- **Banking Systems:** Encrypting transaction data.
- **File Encryption:** Tools like BitLocker use symmetric encryption to secure files.

# Symmetric Ciphers

## Advantages:

- 1. Speed:** Faster encryption and decryption processes.
- 2. Efficiency:** Requires less computational power.
- 3. Suitable for Large Data:** Effective for bulk data encryption.

## Disadvantages:

- 1. Key Distribution Problem:** The key must be shared securely, posing a risk if intercepted.
- 2. Scalability Issues:** Managing keys becomes complex in large networks.
- 3. Lack of Non-Repudiation:** Since both parties share the same key, it's difficult to prove who encrypted the message.

# Classical Encryption Techniques

Classical encryption techniques are the foundational methods that paved the way for modern cryptography. They mainly fall into two categories:

**A. Substitution Ciphers:** Substitution ciphers replace elements of the plaintext with corresponding elements of ciphertext.

**1. Caesar Cipher:** A shift cipher where each letter in the plaintext is shifted a fixed number of positions down the alphabet. **Example:** With a shift of 3:

1. Plaintext: HELLO
2. Ciphertext: KHOOR

**2. Vigenère Cipher:** Uses a keyword to determine the shift for each letter, making it a polyalphabetic cipher. **Example:**

1. Plaintext: ATTACKATDAWN
2. Key: LEMON
3. Ciphertext: LXFOPVEFRNHR

Encryption

$$E_i = (P_i + K_i) \bmod 26$$

Decryption

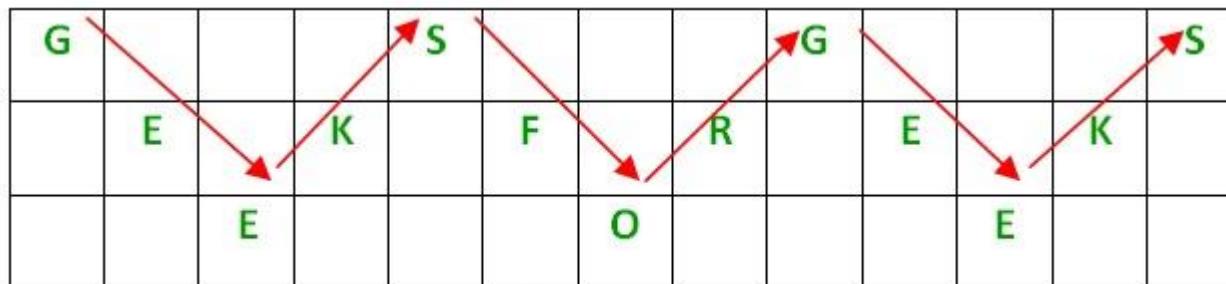
$$D_i = (E_i - K_i) \bmod 26$$

# Classical Encryption Techniques

**B. Transposition Ciphers:** Instead of replacing letters, transposition ciphers rearrange the letters in the plaintext.

**Rail Fence Cipher:** The plaintext is written diagonally over multiple "rails" and then read row by row.

For example, if the message is “GeeksforGeeks” and the number of rails = 3 then cipher is prepared as:



cipher-text = “GsGsekfrek eoe”

# Classical Encryption Techniques

## Columnar Transposition Cipher:

### Encryption

**Given text** = Geeks for Geeks

**Keyword** = HACK

**Length of Keyword** = 4 (no of rows)

**Order of Alphabets in HACK** = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	-	f	o
r	-	G	e
e	k	s	-

Print Characters of column 1,2,3,4

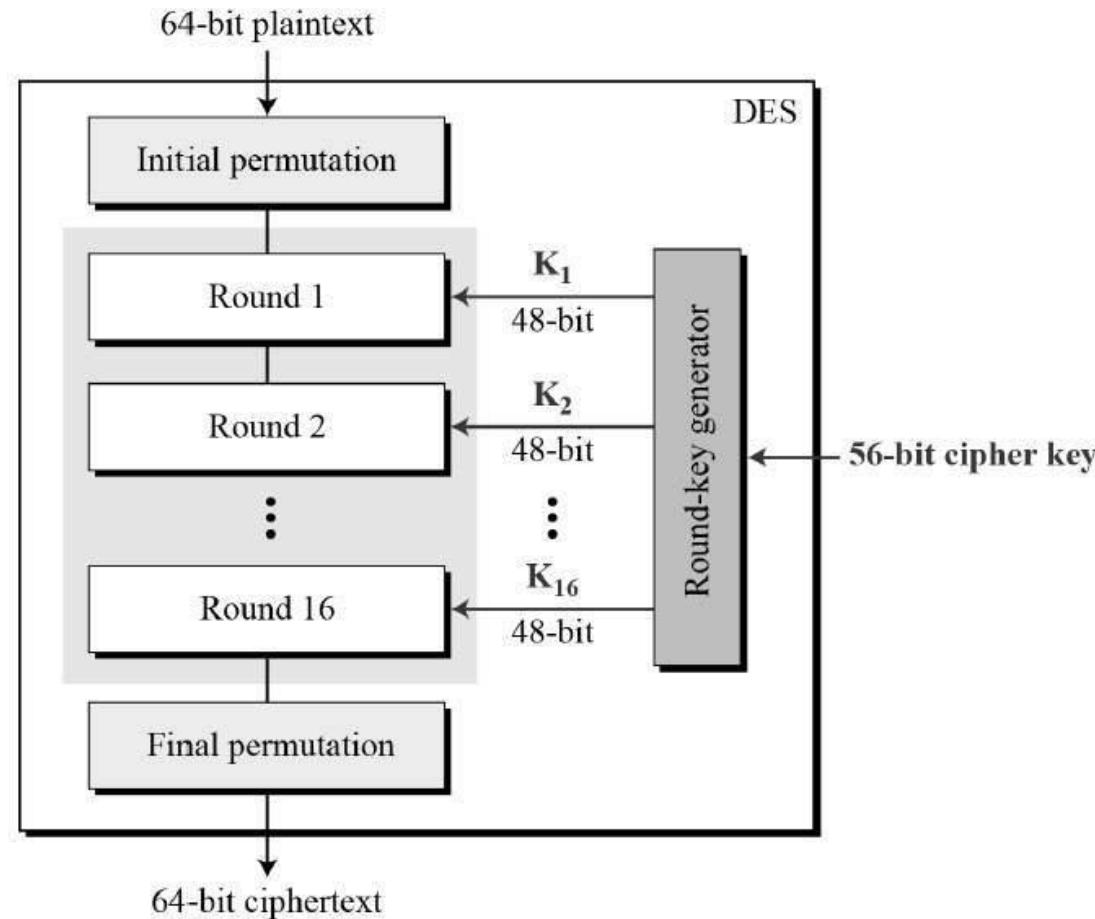
**Encrypted Text** = e kefGsGsreko\_e

# Block Ciphers and the Data Encryption Standard (DES)

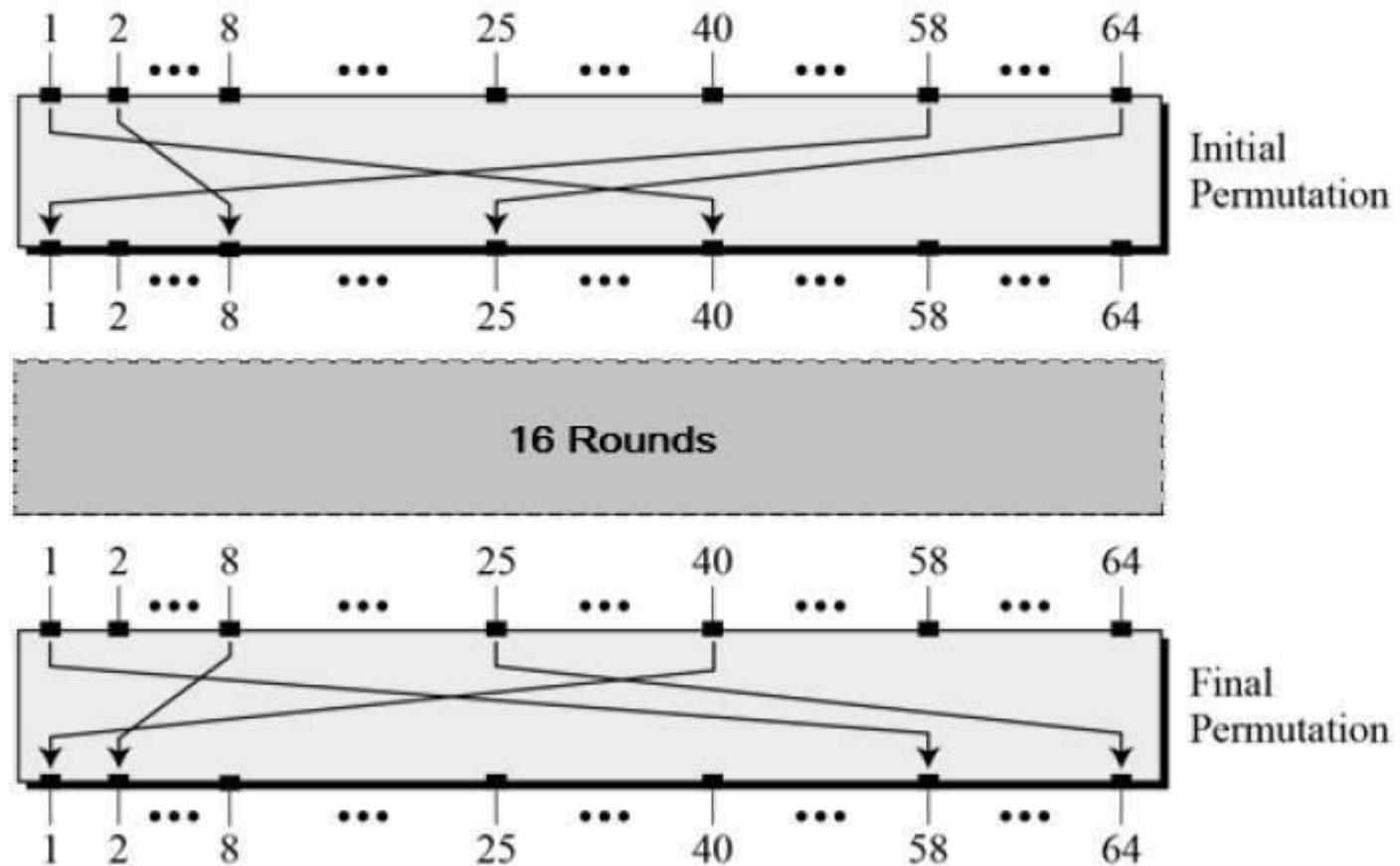
**3. Block Ciphers and DES:** Block ciphers encrypt data in fixed-size blocks (e.g., 64-bit or 128-bit blocks) using a symmetric key.

- **Data Encryption Standard (DES):**
  - Developed in the 1970s by IBM and adopted as a standard by the U.S. government.
  - Encrypts data in 64-bit blocks using a 56-bit key.
- **DES Structure:**
  - **Feistel Network:** The core structure, where data is split into two halves and processed through multiple rounds of permutation and substitution.
  - **16 Rounds of Processing:** Each round applies key mixing, substitution, and permutation.
- **Steps of DES:**
  1. **Initial Permutation (IP):** Rearranges the bits of the plaintext.
  2. **16 Rounds of Encryption:** Each round uses a unique sub-key derived from the main key.
  3. **Final Permutation (IP-1):** The inverse of the initial permutation.
- **Weaknesses of DES:**
  - **Key Length Vulnerability:** A 56-bit key is too short by today's standards, making DES susceptible to brute-force attacks.
  - **Solution: Triple DES (3DES)** applies DES three times with different keys for enhanced security.

# Data Encryption Standard (DES)



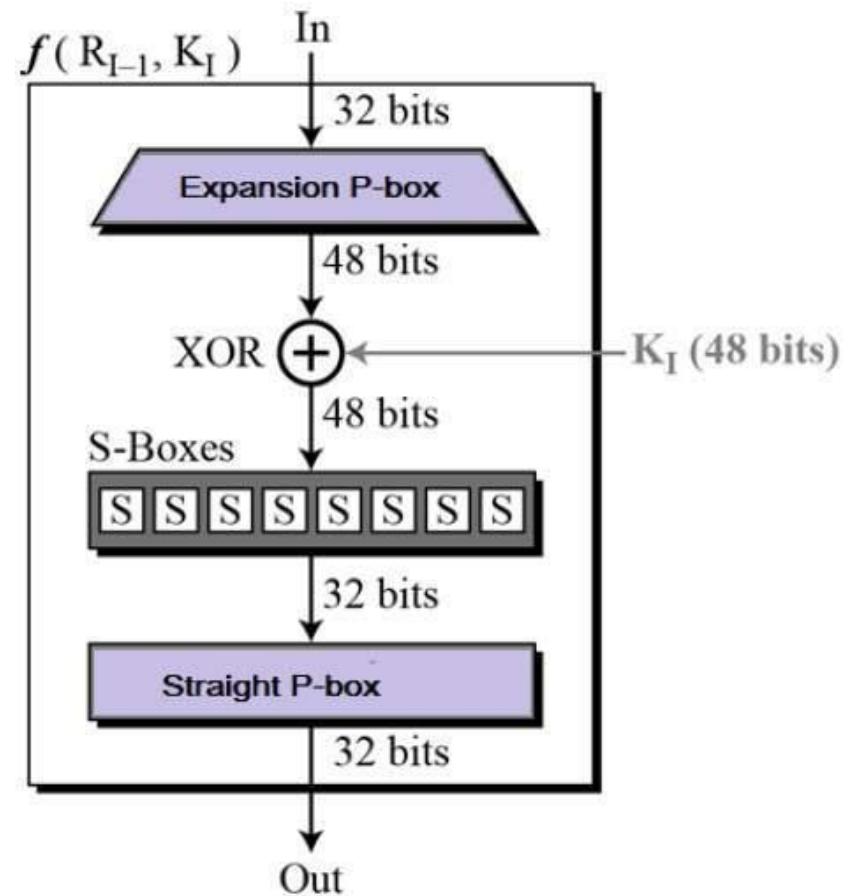
# Data Encryption Standard (DES)



# Data Encryption Standard (DES)

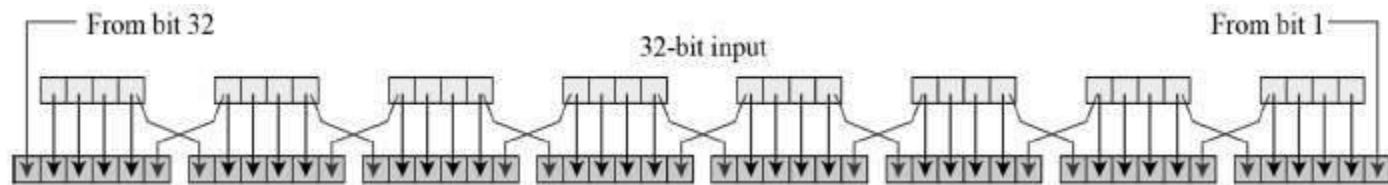
## Round Function:

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



# Data Encryption Standard (DES)

- **Expansion Permutation Box** – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration –

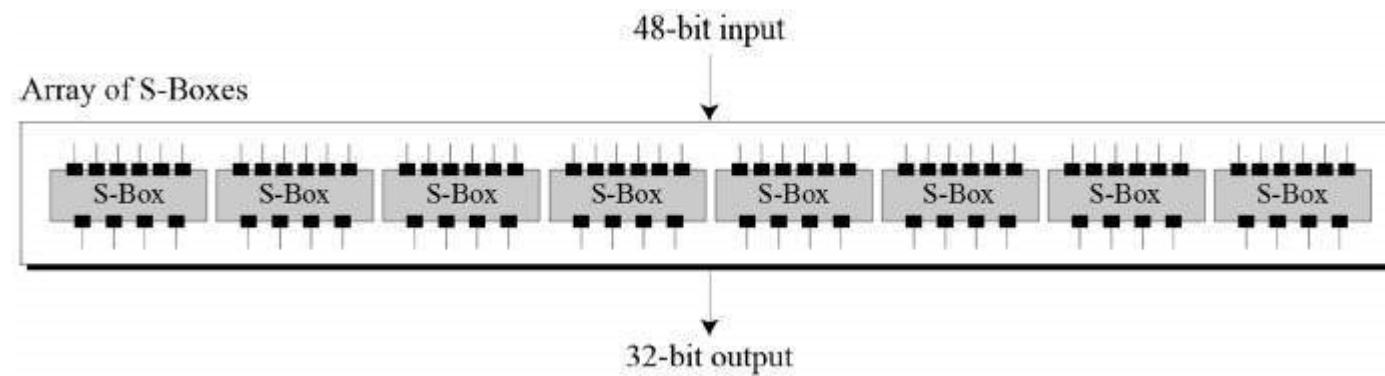


# Data Encryption Standard (DES)

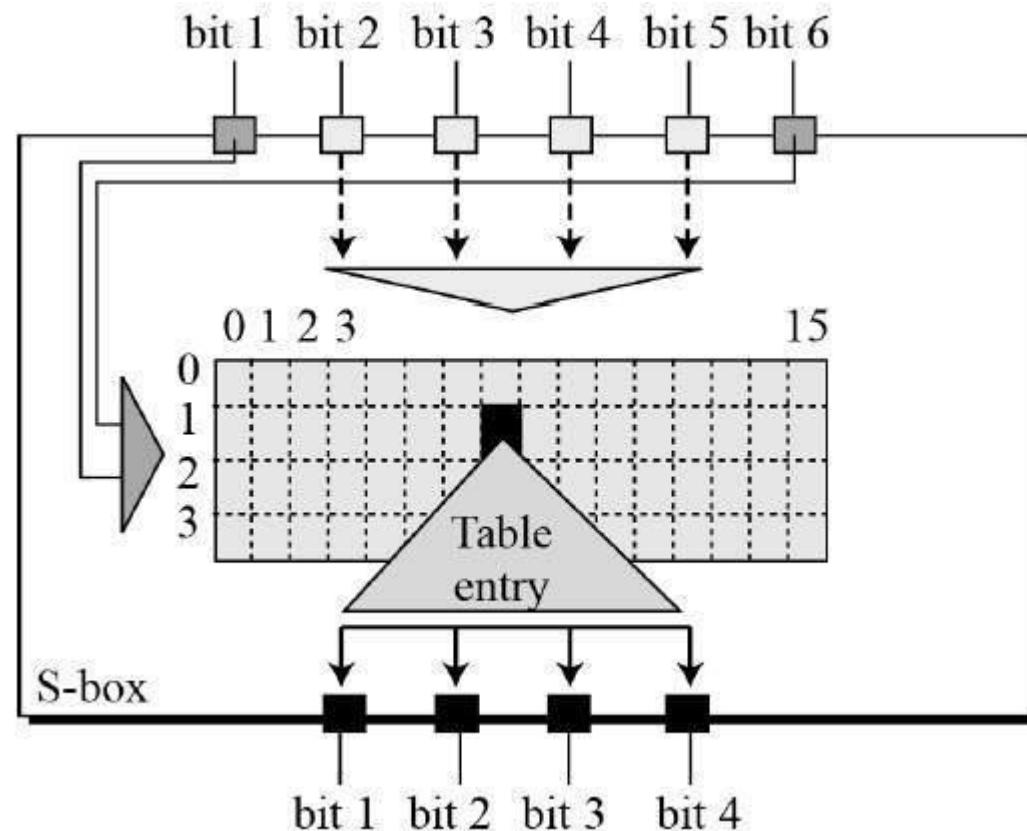
**XOR (Whitener).** – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

# Data Encryption Standard (DES)

**Substitution Boxes.** – The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration –



# Data Encryption Standard (DES)



# Data Encryption Standard (DES)

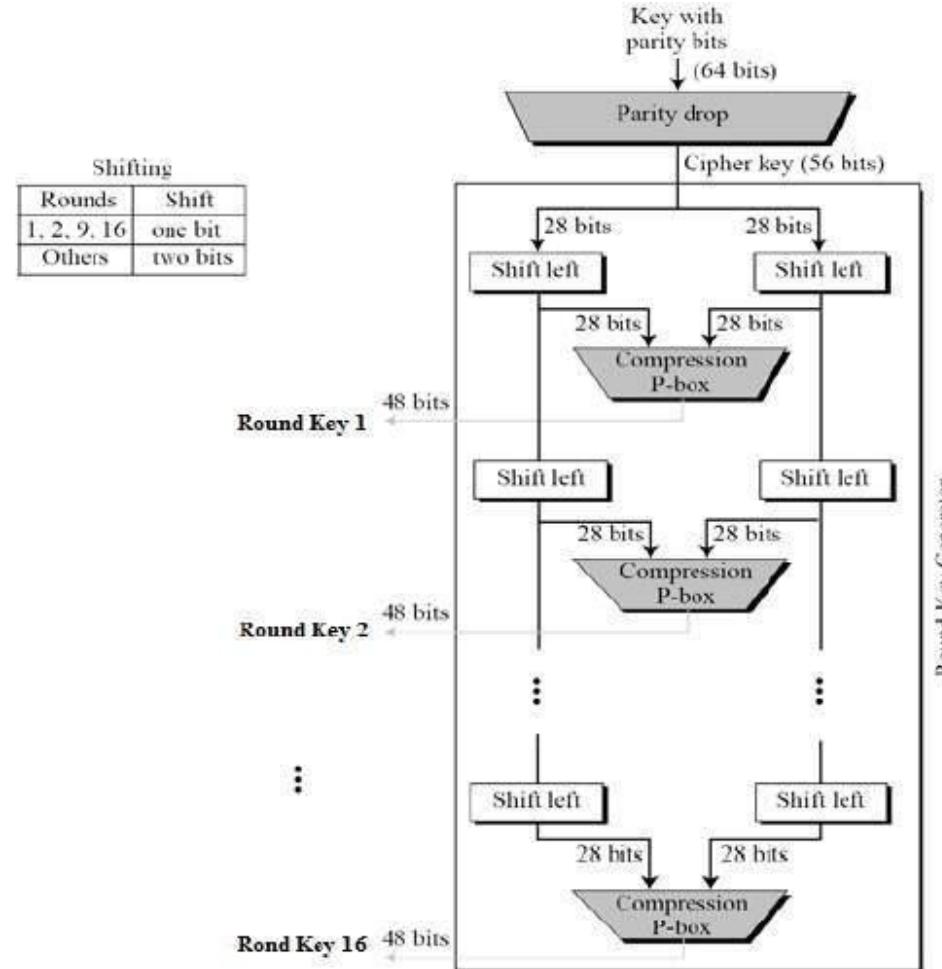
- **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

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

# Data Encryption Standard (DES)

## Key Generation:

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –



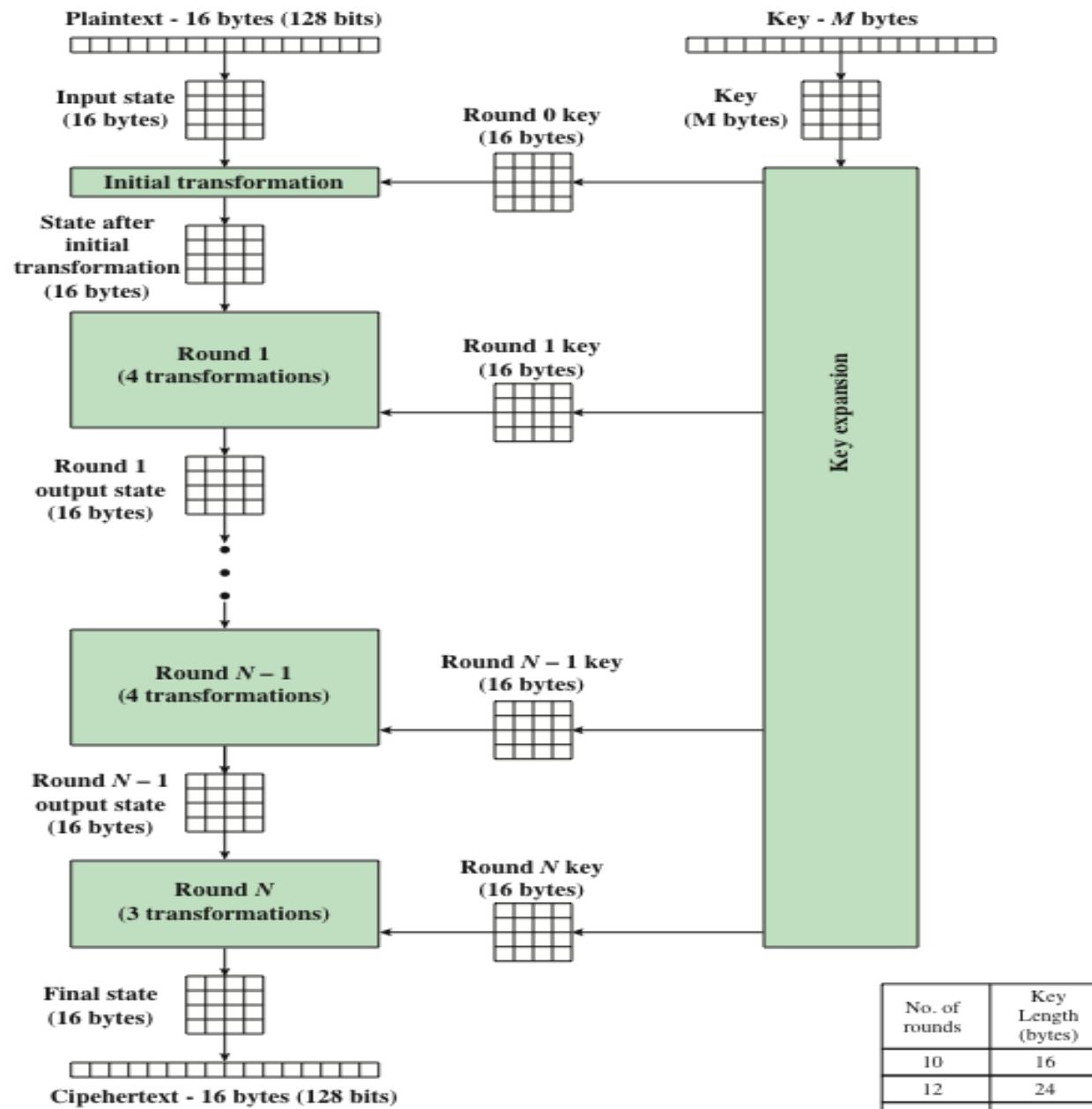
# Advanced Encryption Standard (AES)

## **Advanced Encryption Standard (AES)**

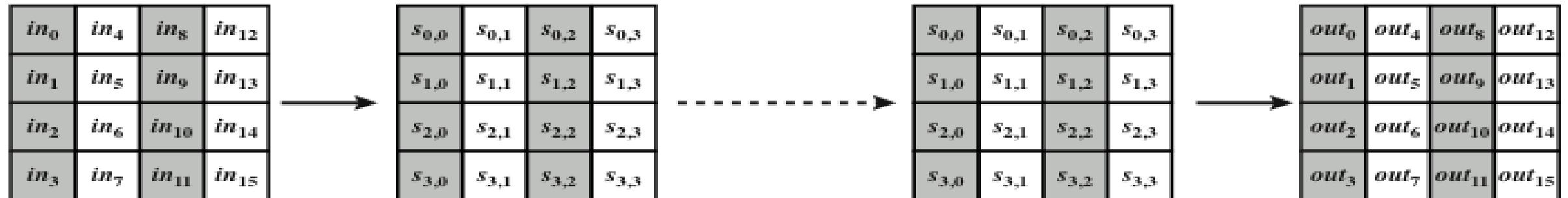
Ref: Cryptography and Network Security by William  
Stallings

# Finite Field Arithmetic

- AES is developed by two Belgium cryptographers, Vincent Rijmen and Joan Daemen.
- In the Advanced Encryption Standard (AES) all operations are performed on 8-bit bytes
- The arithmetic operations of addition, multiplication, and division are performed over the finite field  $\text{GF}(2^8)$
- A field is a set in which we can do addition, subtraction, multiplication, and division without leaving the set
- Division is defined with the following rule:
  - $a/b = a(b^{-1})$
- An example of a finite field (one with a finite number of elements) is the set  $\mathbb{Z}_p$  consisting of all the integers  $\{0, 1, \dots, p - 1\}$ , where  $p$  is a prime number and in which arithmetic is carried out modulo  $p$



**Figure 6.1 AES Encryption Process**



(a) Input, state array, and output

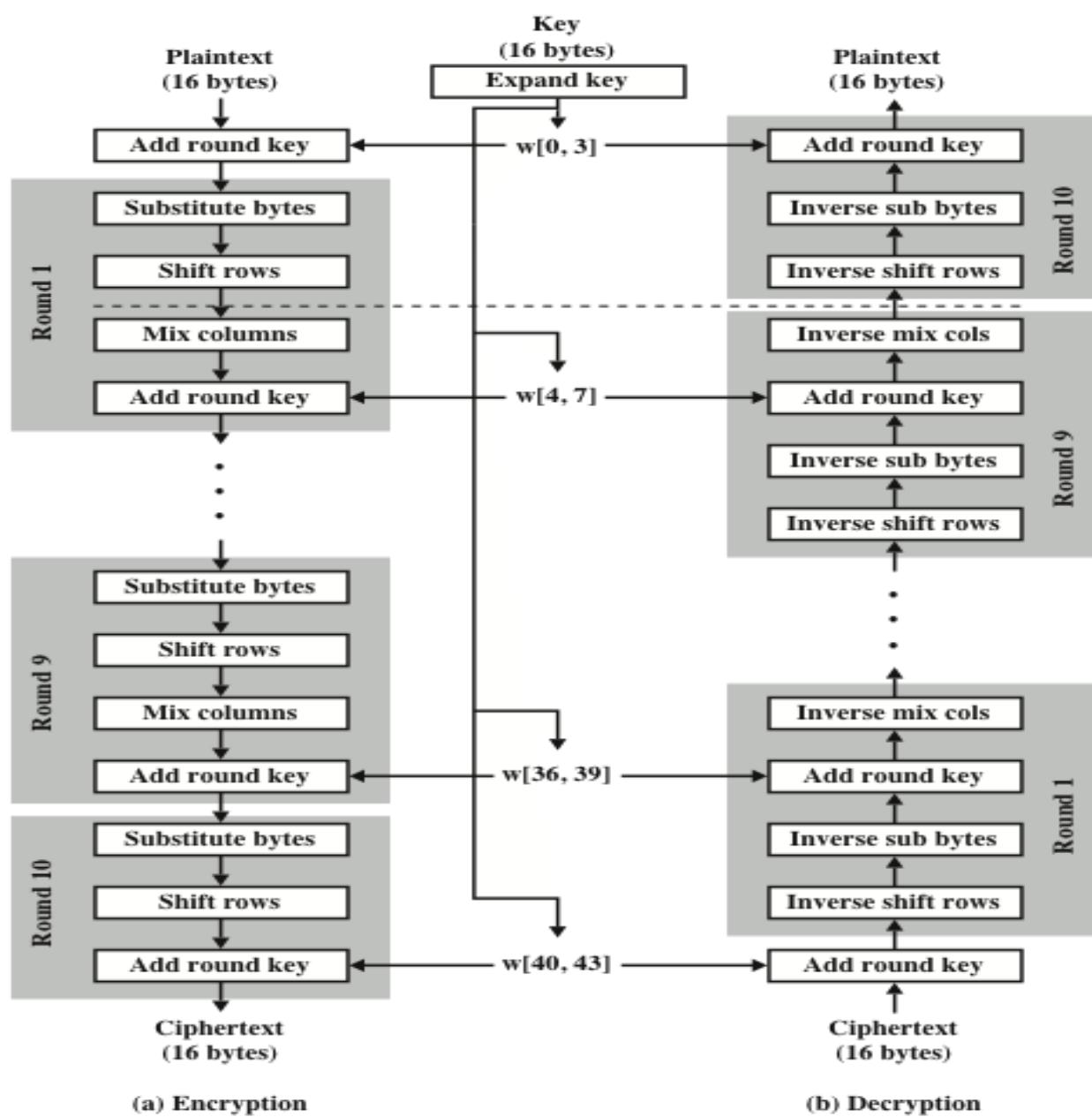


(b) Key and expanded key

**Figure 6.2 AES Data Structures**

# Table 6.1 AES Parameters

<b>Key Size (words/bytes/bits)</b>	4/16/128	6/24/192	8/32/256
<b>Plaintext Block Size (words/bytes/bits)</b>	4/16/128	4/16/128	4/16/128
<b>Number of Rounds</b>	10	12	14
<b>Round Key Size (words/bytes/bits)</b>	4/16/128	4/16/128	4/16/128
<b>Expanded Key Size (words/bytes)</b>	44/176	52/208	60/240



**Figure 6.3 AES Encryption and Decryption**

# Detailed Structure

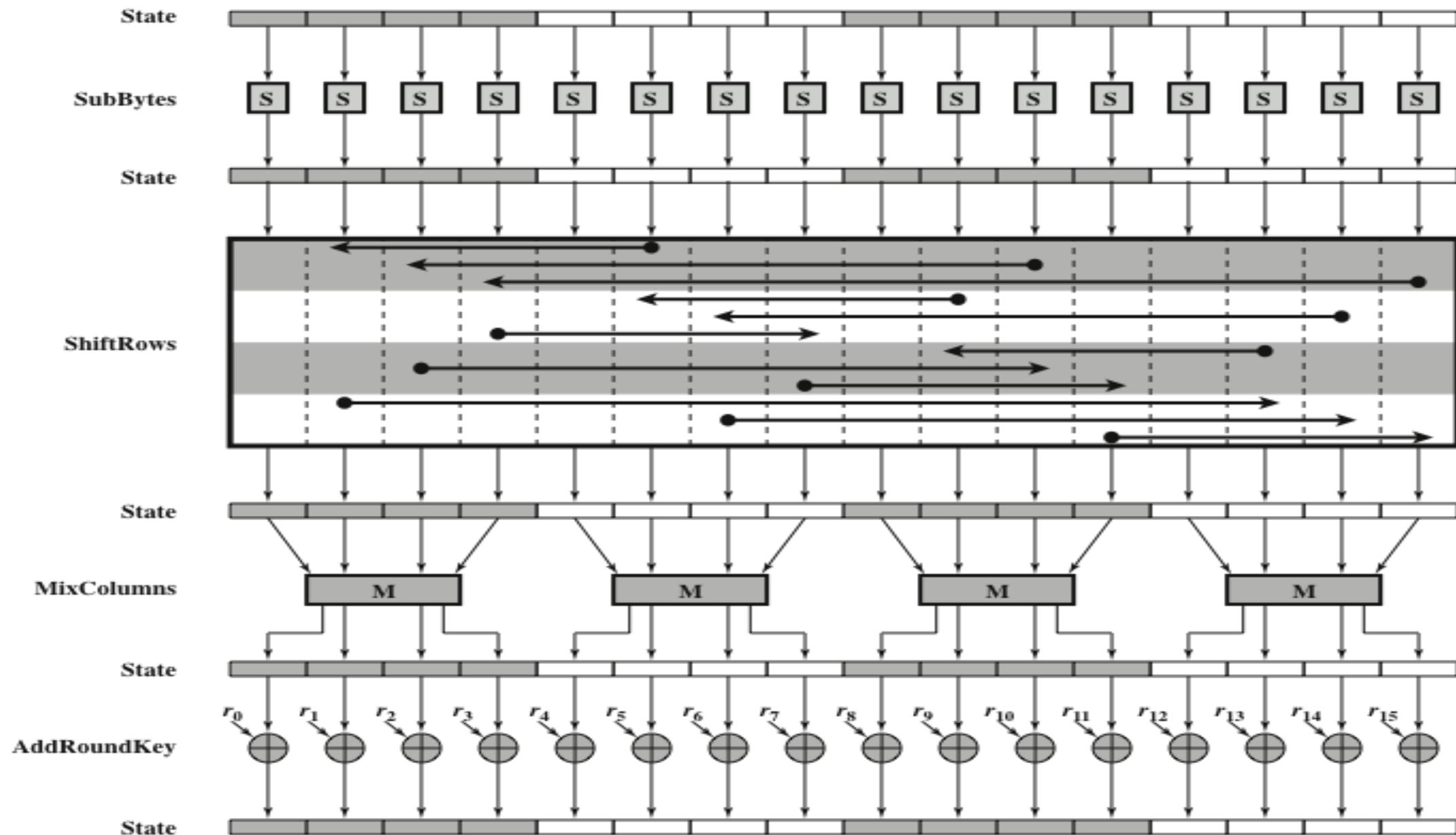
- Processes the entire data block as a single matrix during each round using substitutions and permutation
- The key that is provided as input is expanded into an array of forty-four 32-bit words,  $w[i]$

Four different stages are used.

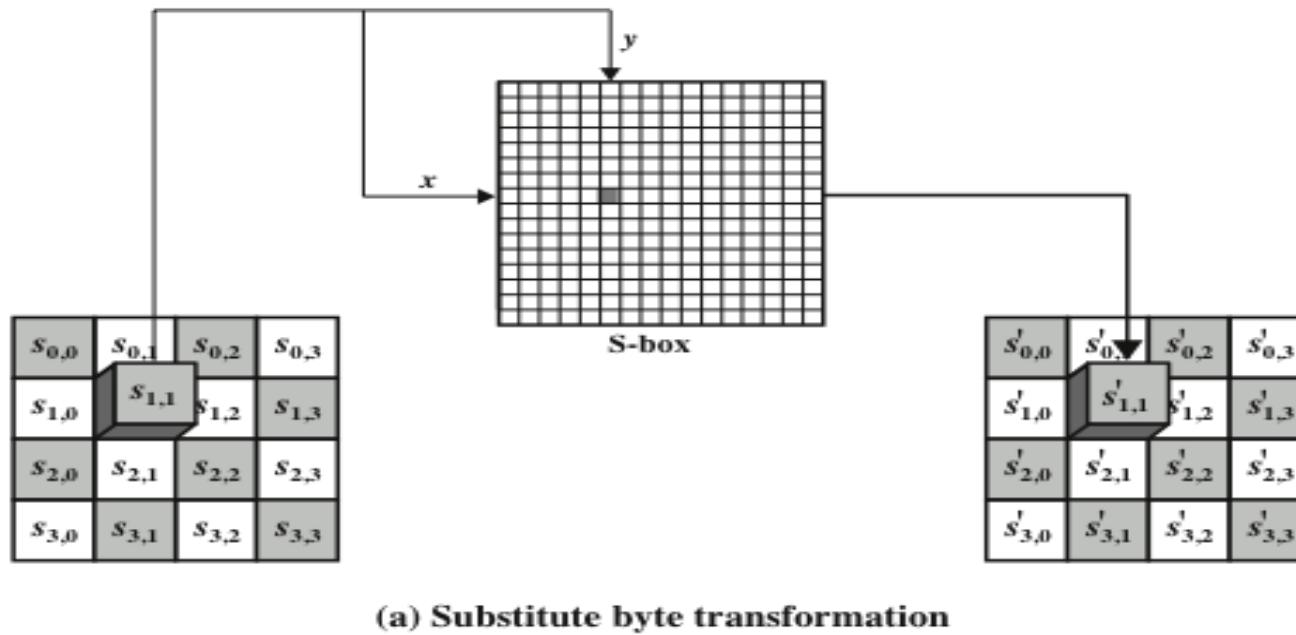
## Four different stages are used:

- Substitute bytes – uses an S-box to perform a byte-by-byte substitution of the block
- ShiftRows – a simple permutation
- MixColumns – a substitution that makes use of arithmetic over  $GF(2^8)$
- AddRoundKey – a simple bitwise XOR of the current block with a portion of the expanded key

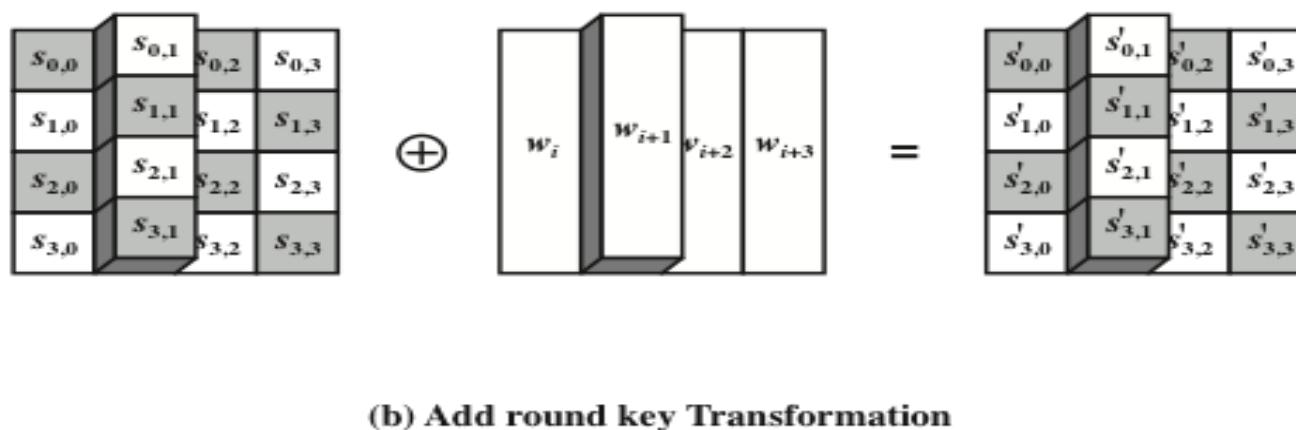
- The cipher begins and ends with an AddRoundKey stage
- Can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on
- The decryption algorithm makes use of the expanded key in reverse order, **however the decryption algorithm is not identical to the encryption algorithm**
- Final round of both encryption and decryption consists of only three stages



**Figure 6.4 AES Encryption Round**



(a) Substitute byte transformation



(b) Add round key Transformation

## Figure 6.5 AES Byte-Level Operations

# Table 6.2

		<b>y</b>															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

(Table can be found on page 163 in textbook)

# Table 6.2

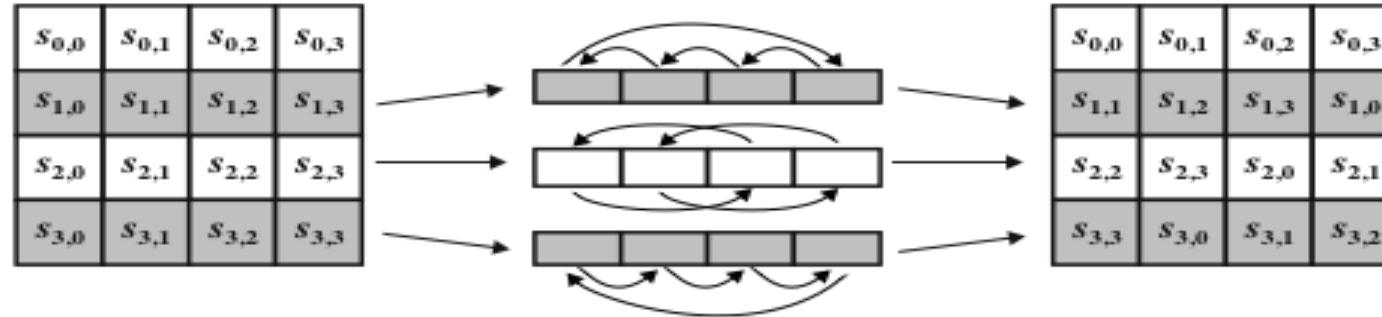
	<i>y</i>																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse S-box

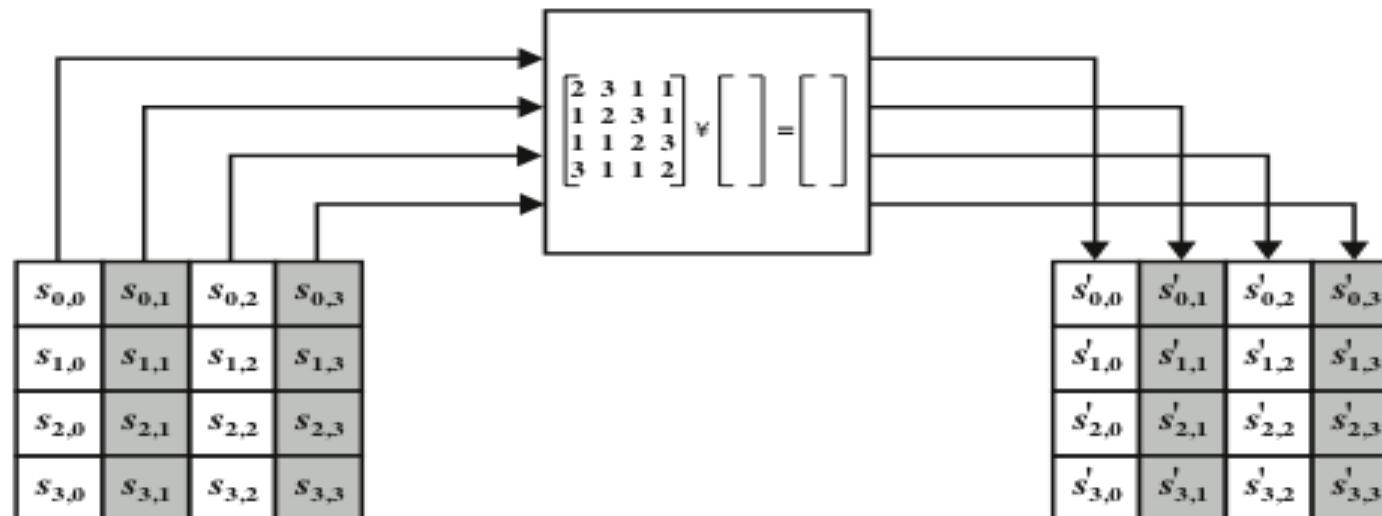
(Table can be found on page 163 in textbook)

# S-Box Rationale

- The S-box is designed to be resistant to known cryptanalytic attacks
- The Rijndael developers sought a design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input
- The nonlinearity is due to the use of the multiplicative inverse



(a) Shift row transformation



(b) Mix column transformation

**Figure 6.7 AES Row and Column Operations**

# Shift Row Rationale

- The State, as well as the cipher input and output, is treated as an array of four 4-byte columns
- On encryption, the first 4 bytes of the plaintext are copied to the first column of State, and so on
- A row shift moves an individual byte from one column to another.
- Transformation ensures that the 4 bytes of one column are spread out to four different columns

# Mix Columns Rationale

- Coefficients of a matrix based on a linear code ensures a good mixing among the bytes of each column
- The mix column transformation combined with the shift row transformation ensures that after a few rounds all output bits depend on all input bits

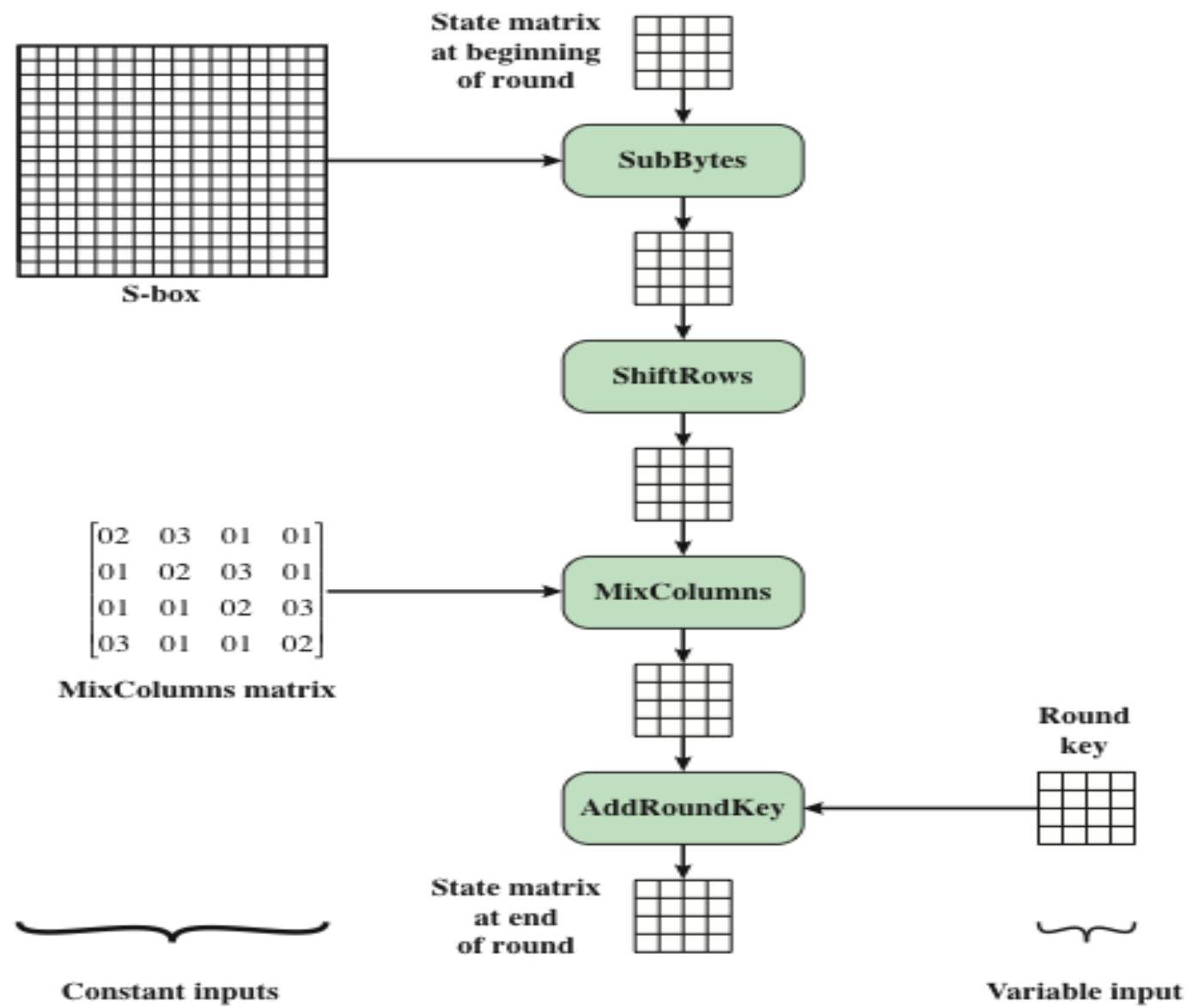
# AddRoundKey Transformation

- The 128 bits of State are bitwise XORed with the 128 bits of the round key
- Operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key
  - Can also be viewed as a byte-level operation

## Rationale:

Is as simple as possible and affects every bit of State

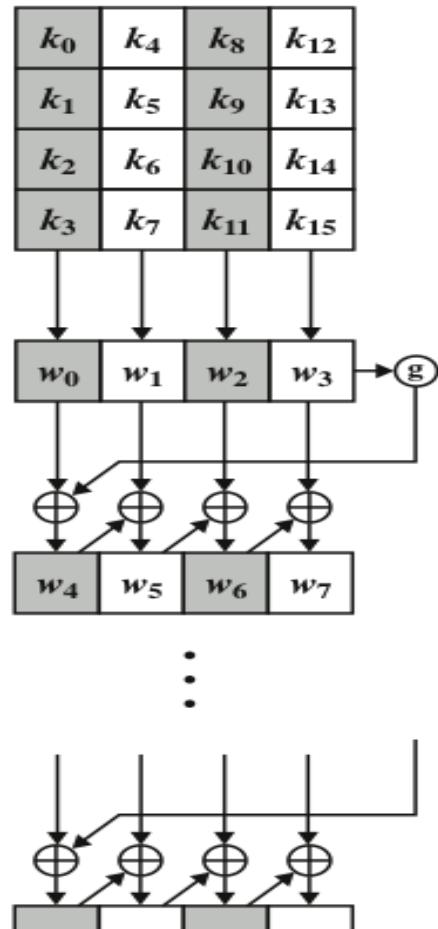
The complexity of the round key expansion plus the complexity of the other stages of AES ensure security



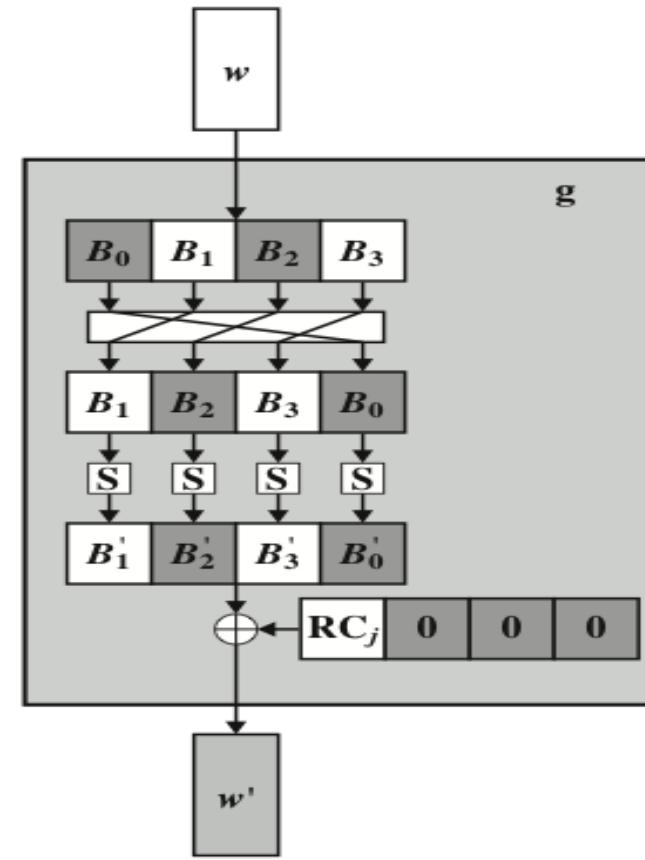
**Figure 6.8 Inputs for Single AES Round**

# AES Key Expansion

- Takes as input a four-word (16 byte) key and produces a linear array of 44 words (176) bytes
  - This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher
- Key is copied into the first four words of the expanded key
  - The remainder of the expanded key is filled in four words at a time
- Each added word  $w[i]$  depends on the immediately preceding word,  $w[i - 1]$ , and the word four positions back,  $w[i - 4]$ 
  - In three out of four cases a simple XOR is used
  - For a word whose position in the  $w$  array is a multiple of 4, a more complex function is used



(a) Overall algorithm



(b) Function g

Figure 6.9 AES Key Expansion

# Basic Concepts in Number Theory and Finite Fields

- **Number Theory in Cryptography:**

1. **Prime Numbers:** Fundamental in key generation for asymmetric algorithms like RSA.
2. **Modular Arithmetic:** Operations performed modulo a number,
3. **Greatest Common Divisor (GCD):** Important for finding relatively prime numbers.

- **Finite Fields (Galois Fields):** A set of finite numbers where you can perform addition, subtraction, multiplication, and division (except by zero). Used in AES for mixing columns and key scheduling.

- **Important Theorems:**

- **Fermat's Little Theorem:** Basis for RSA encryption.
- **Euler's Theorem:** Used in public key cryptography algorithms.

# Basic Concepts in Number Theory and Finite Fields

## Prime Numbers

- ★ Prime Numbers: Has exactly two divisors.
- ★ If 'N' is prime, then the divisors are 1 and N.
- ★ All numbers have prime factors.

Numbers	10	11	100	37	308	14688
Prime Factorization	$2^1 \times 5^1$	$1^1 \times 11^1$	$2^2 \times 5^2$	$1^1 \times 37^1$	$2^2 \times 7^1 \times 11^1$	$2^5 \times 3^3 \times 17^1$
Prime Numbers	2, 5	1, 11	2, 5	1, 37	2, 7, 11	2, 3, 17

# Basic Concepts in Number Theory and Finite Fields

## Prime Numbers – Example

- ★ 2 is a prime number.
- ★ 3 is a prime number.
- ★ 5 is a prime number.
- ★ 7 is a prime number.
- ★ 9 is not a prime number.
- ★ 9 is a composite number.
- ★ 33 is a composite number.



Divisors of 9: 1, 3 and 9

# Basic Concepts in Number Theory and Finite Fields

## Facts about primes

- ★ Only even prime : 2
- ★ Smallest prime number : 2
- ★ Is 1 a prime number? No.
- ★ Except for 2 and 5, all prime numbers end in the digit 1, 3, 7 or 9.

# Basic Concepts in Number Theory and Finite Fields

## Why prime numbers in cryptography?

- ★ Many encryption algorithms are based on prime numbers.
- ★ Very fast to multiply two large prime numbers.
- ★ Extremely computer-intensive to do the reverse.
- ★ Factoring very large prime numbers is very hard i.e. take computers a long time.  
↓

# Basic Concepts in Number Theory and Finite Fields

Are they prime numbers?

- ★ 5393
- ★ 27644437
- ★ 4398042316799
- ★ 1125899839733759
- ★ 18014398241046527
- ★ 1298074214633706835075030044377087

**Note:** Cryptographic algorithms use large prime numbers.

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Congruence

★ In cryptography, congruence ( $\equiv$ ) instead of equality (=).

Examples:

$$15 \equiv 3 \pmod{12}$$

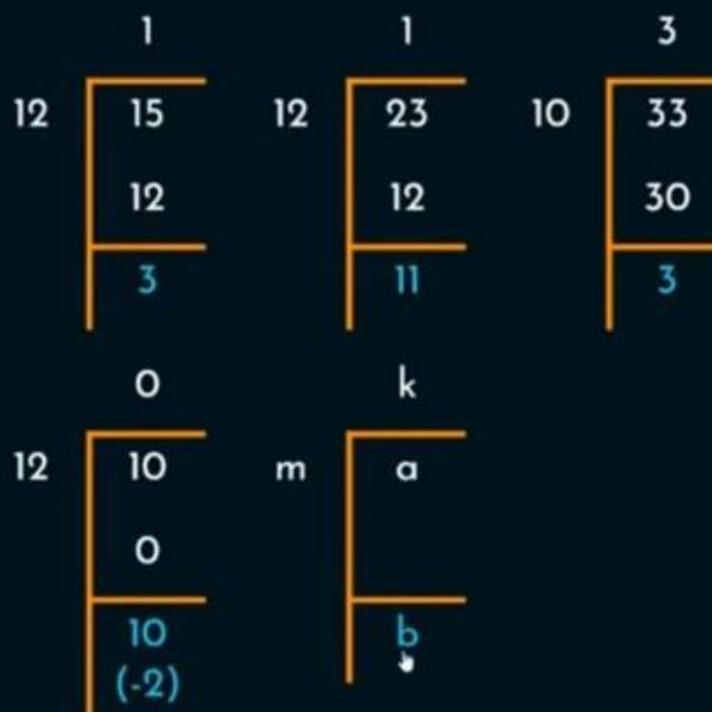
$$23 \equiv 11 \pmod{12}$$

$$33 \equiv 3 \pmod{10}$$

$$10 \equiv -2 \pmod{12}$$

$$\therefore a \equiv b \pmod{m}$$

$$\text{i.e. } a = km + b$$



# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Valid or Invalid

- ★  $38 \equiv 2 \pmod{12}$  ✓
- ★  $38 \equiv 14 \pmod{12}$  ✓
- ★  $5 \equiv 0 \pmod{5}$  ✓
- ★  $10 \equiv 2 \pmod{6}$  ✗
- ★  $13 \equiv 3 \pmod{13}$  ✗
- ★  $2 \equiv -3 \pmod{5}$  ✓

\*

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Properties of Modular Arithmetic

1.  $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2.  $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3.  $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Example:

$$\begin{aligned} [(15 \bmod 8) \times (11 \bmod 8)] \bmod 8 &= (15 \times 11) \bmod 8 \\ &= 165 \bmod 8 \\ &= 5 \end{aligned}$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Properties of Modular Arithmetic

Property	Expression
Commutative Laws	$(a + b) \text{ mod } n = (b + a) \text{ mod } n$ $(a \times b) \text{ mod } n = (b \times a) \text{ mod } n$
Associative Laws	$[(a + b) + c] \text{ mod } n = [a + (b + c)] \text{ mod } n$ $[(a \times b) \times c] \text{ mod } n = [a \times (b \times c)] \text{ mod } n$
Distributive Laws	$[a \times (b + c)] \text{ mod } n = [(a \times b) + (a \times c)] \text{ mod } n$
Identities	$(0 + a) \text{ mod } n = a \text{ mod } n$ $(1 \times a) \text{ mod } n = a \text{ mod } n$
Additive Inverse	For each $a \in \mathbb{Z}_n$ , there exists a ' $-a$ ' such that $a + (-a) \equiv 0 \text{ mod } n$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Modular Exponentiation

- ❖ It is a type of exponentiation performed over a modulus.
- ❖  $a^b \text{ mod } m$  or  $a^b \pmod{m}$ .

Examples:

$$2^{33} \text{ mod } 30$$



$$3^{100} \text{ mod } 29$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Example 1

Solve  $23^3 \bmod 30$ .

$$\begin{aligned}23^3 \bmod 30 &= -7^3 \bmod 30 \quad || \text{ } 23 \bmod 30 \text{ can be } 23 \text{ or } -7. \\&= -7^3 \bmod 30 \\&= -7^2 \times -7 \bmod 30 \\&= 49 \times -7 \bmod 30 \\&= -133 \bmod 30 \\&= -13 \bmod 30 \\&= 17 \bmod 30\end{aligned}$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Example 1

Solve  $88^7 \bmod 187$ .

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 88^1 \times 88^1 \bmod 187 = 88 \times 88 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 88^2 \times 88^2 \bmod 187 = 77 \times 77 = 5929 \bmod 187 = 132$$

$$\begin{aligned}88^7 \bmod 187 &= 88^4 \times 88^2 \times 88^1 \bmod 187 = (132 \times 77 \times 88) \bmod 187 \\&= 894,432 \bmod 187\end{aligned}$$

$$88^7 \bmod 187 = 11$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Example 2

What is "the last two digits" of  $29^5$ ?

$$29^1 \bmod 100 = 29 \text{ or } -71$$

$$29^2 \bmod 100 = 29^1 \times 29^1 \bmod 100 = 29 \times 29 = 841 \bmod 100 = 41 \text{ or } -59$$

$$29^4 \bmod 100 = 29^2 \times 29^2 \bmod 100 = 41 \times 41 = 1681 \bmod 100 = 81 \text{ or } -19$$

$$29^5 \bmod 100 = 29^4 \times 29^1 \bmod 100$$

$$= -19 \times 29 \bmod 100$$

$$= -551 \bmod 100$$

$$= -51 \bmod 100$$

$$= 49$$

$$29^5 \bmod 100 = 49$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Example 3

Solve  $3^{100} \bmod 29$ .

$$3^1 \bmod 29 = 3 \bmod 29 = 3 \text{ or } -26.$$

$$3^2 \bmod 29 = 3^1 \times 3^1 \bmod 29 = 3 \times 3 \bmod 29 = 9 \bmod 29 = 9 \text{ or } -20.$$

$$3^4 \bmod 29 = 3^2 \times 3^2 \bmod 29 = 9 \times 9 \bmod 29 = 81 \bmod 29 = 23 \text{ or } -6.$$

$$3^8 \bmod 29 = 3^4 \times 3^4 \bmod 29 = -6 \times -6 \bmod 29 = 36 \bmod 29 = 7 \text{ or } -22.$$

$$3^{16} \bmod 29 = 3^8 \times 3^8 \bmod 29 = 7 \times 7 \bmod 29 = 49 \bmod 29 = 20 \text{ or } -9.$$

$$3^{32} \bmod 29 = 3^{16} \times 3^{16} \bmod 29 = -9 \times -9 \bmod 29 = 81 \bmod 29 = 23 \text{ or } -6.$$

$$3^{64} \bmod 29 = 3^{32} \times 3^{32} \bmod 29 = -6 \times -6 \bmod 29 = 36 \bmod 29 = 7 \text{ or } -22.$$

$$3^{100} \bmod 29 = 3^{64} \times 3^{32} \times 3^4 \bmod 29.$$

$$= 7 \times -6 \times -6 \bmod 29$$

$$= 252 \bmod 29$$

$$3^{100} \bmod 29 = 20$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Understanding GCD – Example 1

	12	33
Divisors	1, 2, 3, 4, 6, 12	1, 3, 11, 33
Common Divisors		1, 3
Greatest Common Divisor (GCD)		3

$$\therefore \text{GCD}(12, 33) = 3$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Understanding GCD – Example 2

	25	150
Divisors	1, 5, 25	1, 2, 3, 5, 6, 10, 15, 25, 30, 50, 75, 150
Common Divisors		1, 5, 25
Greatest Common Divisor (GCD)		25

$$\therefore \text{GCD}(25, 150) = 25$$

\*

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmetic

## Understanding GCD - Example 3

	13	31
Divisors	1, 13	1, 31
Common Divisors		1
Greatest Common Divisor (GCD)		1

$$\therefore \text{GCD}(13, 31) = 1$$

# Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

## Euclid's Algorithm for finding GCD

Find the GCD(12, 33).

Q	A	B	R
2	33	12	9
1	12	9	3
3	9	3	0
X	3	0	X



# Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

## Euclid's Algorithm for finding GCD

Find the  $\text{GCD}(750, 900)$ .

Q	A	B	R
1	900	750	150
5	750	150	0
X	150	0	X



# Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

## Euclid's Algorithm for finding GCD

Find the  $\text{GCD}(252, 105)$ .

Q	A	B	R
2	252	105	42
2	105	42	21
2	42	21	0
X	21	0	X



# Basic Concepts in Number Theory and Finite Fields

## Relatively Prime Numbers

Question 1: Are 4 and 13 relatively prime?

Solution:

	4	13
Divisors	1, 2, 4	1, 13
Common Divisors		1
Greatest Common Divisor (GCD)		1

$$\text{GCD}(4, 13) = 1$$

Yes, 4 and 13 are relatively prime numbers.

# Basic Concepts in Number Theory and Finite Fields

## Relatively Prime Numbers

a	b	GCD(a, b)	Relatively Prime?	Remarks
11	17	1	Yes	'a' and 'b' are prime
11	21	1	Yes	'a' is prime and 'b' is composite
12	77	1	Yes	'a' and 'b' are composite

# Basic Concepts in Number Theory and Finite Fields

**Euler's Totient Function**

Example 1: Find  $\Phi(5)$ .

Solution:

Here  $n=5$ .

Numbers less than 5 are 1, 2, 3 and 4.

GCD	Relatively Prime?
$\text{GCD}(1, 5) = 1$	✓
$\text{GCD}(2, 5) = 1$	✓
$\text{GCD}(3, 5) = 1$	✓
$\text{GCD}(4, 5) = 1$	✓

$\therefore \Phi(5) = 4$ .

# Basic Concepts in Number Theory and Finite Fields

## Euler's Totient Function

Example 2: Find  $\Phi(11)$ .

Solution:

Here  $n=11$ .

Numbers less than 11 are 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

GCD	Relatively Prime?
$\text{GCD}(1, 11) = 1$	✓
$\text{GCD}(2, 11) = 1$	✓
$\text{GCD}(3, 11) = 1$	✓
$\text{GCD}(4, 11) = 1$	✓
$\text{GCD}(5, 11) = 1$	✓

$$\therefore \Phi(11) = 10.$$

GCD	Relatively Prime?
$\text{GCD}(6, 11) = 1$	✓
$\text{GCD}(7, 11) = 1$	✓
$\text{GCD}(8, 11) = 1$	✓
$\text{GCD}(9, 11) = 1$	✓
$\text{GCD}(10, 11) = 1$	✓



# Basic Concepts in Number Theory and Finite Fields

## Euler's Totient Function

Example 3: Find  $\Phi(8)$ .

Solution:

Here  $n=8$ .

Numbers less than 8 are 1, 2, 3, 4, 5, 6, and 7.

GCD	Relatively Prime?
$\text{GCD}(1, 8) = 1$	✓
$\text{GCD}(2, 8) = 2$	✗
$\text{GCD}(3, 8) = 1$	✓
$\text{GCD}(4, 8) = 4$	✗

GCD	Relatively Prime?
$\text{GCD}(5, 8) = 1$	✓
$\text{GCD}(6, 8) = 2$	✗
$\text{GCD}(7, 8) = 1$	✓

$$\therefore \Phi(8) = 4$$

# Basic Concepts in Number Theory and Finite Fields

## Euler's Totient Function

$\Phi(n)$	Criteria of 'n'	Formula
	'n' is prime.	$\Phi(n) = (n-1)$
	$n = p \times q$ . 'p' and 'q' are primes.	$\Phi(n) = (p-1) \times (q-1)$
	$n = a \times b$ . Either 'a' or 'b' is composite. Both 'a' and 'b' are composite.	$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots$ <p style="text-align: center;">↓ where <math>p_1, p_2, \dots</math> are distinct primes.</p>

# Basic Concepts in Number Theory and Finite Fields

## Euler's Totient Function

Example 3: Find  $\Phi(35)$ .

Solution:

Here  $n=35$ .

'n' is a product of two prime numbers 5 and 7.

Let us assign  $p=5$  and  $q=7$ .

$$\Phi(n) = (p-1) \times (q-1)$$

$$\Phi(35) = (5-1) \times (7-1)$$

$$\Phi(35) = 4 \times 6$$

$$\Phi(35) = 24$$

So, there are 24 numbers that are lesser than 35 and relatively prime to 35.

# Basic Concepts in Number Theory and Finite Fields

## Euler's Totient Function

Example 4: Find  $\Phi(1000)$ .

Solution:

Here  $n = 1000 = 2^3 \times 5^3$ .

Distinct prime factors are 2 and 5.

$$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \cdot$$

$$\Phi(1000) = 1000 \times \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{5}\right)$$

$$\Phi(1000) = 1000 \times \left(\frac{1}{2}\right) \left(\frac{4}{5}\right)$$

$$\Phi(1000) = 400$$

# Basic Concepts in Number Theory and Finite Fields

## Euler's Totient Function

Example 5: Find  $\Phi(7000)$ .

Solution:

Here  $n = 7000 = 2^3 \times 5^3 \times 7^1$

Distinct prime factors are 2, 5 and 7.

$$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \left(1 - \frac{1}{p_3}\right) \dots$$

$$\Phi(7000) = 7000 \times \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{5}\right) \left(1 - \frac{1}{7}\right)$$

$$\Phi(7000) = 7000 \times \left(\frac{1}{2}\right) \left(\frac{4}{5}\right) \left(\frac{6}{7}\right)$$

$$\Phi(7000) = 2400$$

# Basic Concepts in Number Theory and Finite Fields

## Fermat's Little Theorem

If 'p' is a prime number and 'a' is a positive integer not divisible by 'p' then  $a^{p-1} \equiv 1 \pmod{p}$

# Basic Concepts in Number Theory and Finite Fields

## Fermat's Little Theorem

Example 1: Does Fermat's theorem hold true for  $p=5$  and  $a=2$ ?

Solution:

Given:  $p=5$  and  $a=2$ .

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

$$2^{5-1} \equiv 1 \pmod{5}$$

$$2^4 \equiv 1 \pmod{5}$$

$$16 \equiv 1 \pmod{5}$$

Therefore, Fermat's theorem holds true for  $p=5$  and  $a=2$ .

# Basic Concepts in Number Theory and Finite Fields

## Fermat's Little Theorem

Example 2: Prove Fermat's theorem holds true for  $p=13$  and  $a=11$ .

Solution:

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

$$11^{13-1} \equiv 1 \pmod{13}$$

$$11^{12} \equiv 1 \pmod{13}$$

$$-2^{12} \equiv 1 \pmod{13}$$

$$-2^{4 \times 3} \equiv 1 \pmod{13}$$

$$3^3 \equiv 1 \pmod{13}$$

$$27 \equiv 1 \pmod{13}$$

Therefore, Fermat's theorem holds true for  $p=13$  and  $a=11$ .

# Basic Concepts in Number Theory and Finite Fields

## Fermat's Little Theorem

Example 3: Prove Fermat's theorem does not hold for  $p=6$  and  $a=2$ .

Solution:

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

$$2^{6-1} \equiv 1 \pmod{6}$$

$$2^5 \equiv 1 \pmod{6}$$

$$32 \equiv 1 \pmod{6}$$

$$32 \not\equiv 1 \pmod{6}$$

Therefore, Fermat's theorem does not hold true for  $p=6$  and  $a=2$ .

# Basic Concepts in Number Theory and Finite Fields

## Euler's Theorem

For every positive integer 'a' & 'n', which are said to be relatively prime, then  $a^{\phi(n)} \equiv 1 \pmod{n}$ .

# Basic Concepts in Number Theory and Finite Fields

## Euler's Theorem

Example 1: Prove Euler's theorem hold true for  $a=3$  and  $n=10$ .

Solution:

Given:  $a=3$  and  $n=10$ .

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

$$3^{\Phi(10)} \equiv 1 \pmod{10}$$

$$\Phi(10) = 4$$

$$3^4 \equiv 1 \pmod{10}$$

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

Therefore, Euler's theorem holds true for  $a=3$  and  $n=10$ .

# Basic Concepts in Number Theory and Finite Fields

## Euler's Theorem

Example 2: Does Euler's theorem hold true for  $a=2$  and  $n=10$ ?

Solution:

Given:  $a=2$  and  $n=10$ .

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

$$2^{\Phi(10)} \equiv 1 \pmod{10}$$

$$\Phi(10) = 4$$

$$2^4 \equiv 1 \pmod{10}$$

$$16 \equiv 1 \pmod{10}$$

Therefore, Euler's theorem does not hold for  $a=2$  and  $n=10$ .

# Basic Concepts in Number Theory and Finite Fields

## Euler's Theorem

Example 3: Does Euler's theorem hold true for  $a=10$  and  $n=11$ ?

Solution:

Given:  $a=10$  and  $n=11$ .

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

$$10^{\Phi(11)} \equiv 1 \pmod{11}$$

$$\Phi(11) = 10$$

$$10^{10} \equiv 1 \pmod{11}$$

$$-1^{10} \equiv 1 \pmod{11}$$

$$1 \equiv 1 \pmod{11}$$

Therefore, Euler's theorem holds for  $a=10$  and  $n=11$ .

# Basic Concepts in Number Theory and Finite Fields

## Multiplicative Inverse

Under mod n

$$A \times A^{-1} \equiv 1 \pmod{n}$$

$$3 \times ? \equiv 1 \pmod{5}$$

$$3 \times 2 \equiv 1 \pmod{5}$$

$$2 \times ? \equiv 1 \pmod{11}$$

$$2 \times 6 \equiv 1 \pmod{11}$$

$$4 \times ? \equiv 1 \pmod{5}$$

$$4 \times 4 \equiv 1 \pmod{5}$$

$$5 \times ? \equiv 1 \pmod{10}$$

# Basic Concepts in Number Theory and Finite Fields

## Group

A group  $G$  denoted by  $\{G, \bullet\}$ , is a set under some operations  $(\bullet)$  if it satisfies the **CAIN properties**.

- ❖ **C** - Closure
- ❖ **A** - Associative
- ❖ **I** - Identity
- ❖ **N** - iNverse.

# Basic Concepts in Number Theory and Finite Fields

## Group and Abelian Group

	Property	Explanation
Abelian Group	Closure	$a, b \in G$ , then $(a \bullet b) \in G$ .
	Associative	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in G$ .
	Identity element	$(a \bullet e) = (e \bullet a) = a$ for all $a, e \in G$ .
	Inverse element	$(a \bullet a') = (a' \bullet a) = e$ for all $a, a' \in G$ .
	Commutative	$(a \bullet b) = (b \bullet a)$ for all $a, b \in G$ .

# Basic Concepts in Number Theory and Finite Fields

## Example

Question: Is  $(\mathbb{Z}, +)$  a group?

Solution:

$$\mathbb{Z} = \{ \dots, -3, -2, -1, 0, 1, 2, 3, \dots \}$$

CAIN Property	Explanation	Satisfied?
Closure	If $a, b \in \mathbb{Z}$ , then $(a + b) \in \mathbb{Z}$ . If $a = 5, b = -2 \in \mathbb{Z}$ then $(a + b) = 3 \in \mathbb{Z}$	✓
Associative	$a + (b + c) = (a + b) + c$ for all $a, b, c \in \mathbb{Z}$ . $5 + (3 + 7) = (5 + 3) + 7 \in \mathbb{Z}$	✓
Identity element	$(a + 0) = (0 + a) = a$ for all $a \in \mathbb{Z}$ . $(5 + 0) = (0 + 5) = 5$ for all $a \in \mathbb{Z}$ .	✓
Inverse element	$(a + a') = (a' + a) = 0$ for all $a, a' \in \mathbb{Z}$ . $(5 + -5) = (-5 + 5) = 0$ for all $5, -5 \in \mathbb{Z}$	✓
Commutative	$(a + b) = (b + a)$ for all $a, b \in \mathbb{Z}$ . $(5 + 9) = (9 + 5)$ for all $9, 5 \in \mathbb{Z}$ .	✓

# Basic Concepts in Number Theory and Finite Fields

## Cyclic Group

A group  $G$  denoted by  $\{G, \bullet\}$ , is said to be a cyclic group, if it contains at-least one generator element.

# Basic Concepts in Number Theory and Finite Fields

## Cyclic Group

Question 1: Prove that  $(G, *)$  is a cyclic group, where  $G = \{1, \omega, \omega^2\}$ .

Solution:

Composition Table

*	1	$\omega$	$\omega^2$	$1^1 = 1$	$\omega^1 = \omega$	$(\omega^2)^1 = \omega^2$
1	1	$\omega$	$\omega^2$	$1^2 = 1 * 1 = 1$	$\omega^2 = \omega * \omega = \omega^2$	$(\omega^2)^2 = \omega^4 = \omega^3 * \omega = \omega$
$\omega$	$\omega$	$\omega^2$	1	$1^3 = 1 * 1 * 1 = 1$	$\omega^3 = \omega^2 * \omega = 1$	$(\omega^2)^3 = \omega^6 = \omega^3 * \omega^3 = 1$
$\omega^2$	$\omega^2$	1	$\omega$	$1^4 = 1 * 1 * 1 * 1 = 1$	$\omega^4 = \omega^3 * \omega = \omega$	$(\omega^2)^4 = \omega^8 = \omega^3 * \omega^3 * \omega^2 = \omega^2$

**Not a Generator**

**Generator**

**Generator**

The generators of  $(G, *)$  are  $\omega$  and  $\omega^2$ .

$\therefore (G, *)$  is a cyclic group.

# Basic Concepts in Number Theory and Finite Fields

## Cyclic Group

Question 2: When does group  $G$  with operation ' $\cdot$ ', is said to be a cyclic group?

Solution:

Let us take an element  $x$

$$G = \{ \dots, x^{-4}, x^{-3}, x^{-2}, x^{-1}, 1, x, x^2, x^3, x^4, \dots \}$$

= Group generated by  $x$

If  $G = \langle x \rangle$  for some  $x$ , then we call  $G$  a cyclic group.



# Basic Concepts in Number Theory and Finite Fields

## Cyclic Group

Question 3: When does group  $G$  with operation '+', is said to be a cyclic group?

Solution:

Let us take an element  $y$

$$G = \{ \dots, -4y, -3y, -2y, -y, 0, y, 2y, 3y, 4y, \dots \}$$

= Group generated by  $y$

If  $G = \langle y \rangle$  for some  $y$ , then we call  $G$  a cyclic group.



# Basic Concepts in Number Theory and Finite Fields

## Rings

A ring  $R$  denoted by  $\{R, +, *\}$ , is a set of elements with two binary operations, called addition and multiplication, such that for all  $a, b, c \in R$  the following axioms are obeyed:

- ❖ Group (A1-A4), Abelian Group(A5).
- ❖ Closure under multiplication (M1): If  $a, b \in R$  then  $ab \in R$
- ❖ Associativity of multiplication (M2):  $a (bc) = (ab) c$  for all  $a, b, c \in R$
- ❖ Distributive laws (M3) :

$$a (b + c) = ab + ac \text{ for all } a, b, c \in R$$

$$(a + b) c = ac + bc \text{ for all } a, b, c \in R$$

Note:

Subtraction  $[a - b = a + (-b)]$

# Basic Concepts in Number Theory and Finite Fields

## Commutative Rings

A ring is said to be commutative, if it satisfies the following additional condition:

**Commutativity of multiplication (M4):**  $ab = ba$  for all  $a, b \in R$

# Basic Concepts in Number Theory and Finite Fields

## Integral Domain

An integral domain is a commutative ring that obeys the following axioms:

**Multiplicative identity (M5):** There is an element  $1 \in R$  such that  $a1 = 1a = a$  for all  $a \in R$ .

**No zero divisors (M6):** If  $a, b \in R$  and  $ab = 0$ , then either  $a = 0$  or  $b = 0$ .

# Basic Concepts in Number Theory and Finite Fields

## Fields

A field  $F$ , sometimes denoted by  $\{F, +, *\}$ , is a set of elements with two binary operations, called addition and multiplication, such that for all  $a, b, c \in F$  the following axioms are obeyed:

(A1-M6):  $F$  is an integral domain; that is,  $F$  satisfies axioms A1 - A5 and M1 - M6.

(M7) Multiplicative inverse: For each  $a$  in  $F$ , except 0, there is an element  $a^{-1}$  in  $F$  such that

$$aa^{-1} = (a^{-1})a = 1$$

Note:  $a/b = a(b^{-1})$ .

Familiar examples of Fields:

- ❖ Rational numbers
- ❖ Real numbers
- ❖ Complex numbers



# Basic Concepts in Number Theory and Finite Fields

## Groups, Rings and Fields



# Basic Concepts in Number Theory and Finite Fields

## Finite Fields

- ❖ A finite field or Galois field (so-named in honor of Évariste Galois) is a field that 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 most common examples of finite fields are given by the integers  $(\text{mod } p)$  when  $p$  is a prime number.

### Application areas:

- ❖ Mathematics and computer science - Number theory, Algebraic geometry, Galois theory, Finite geometry, Cryptography and Coding theory.

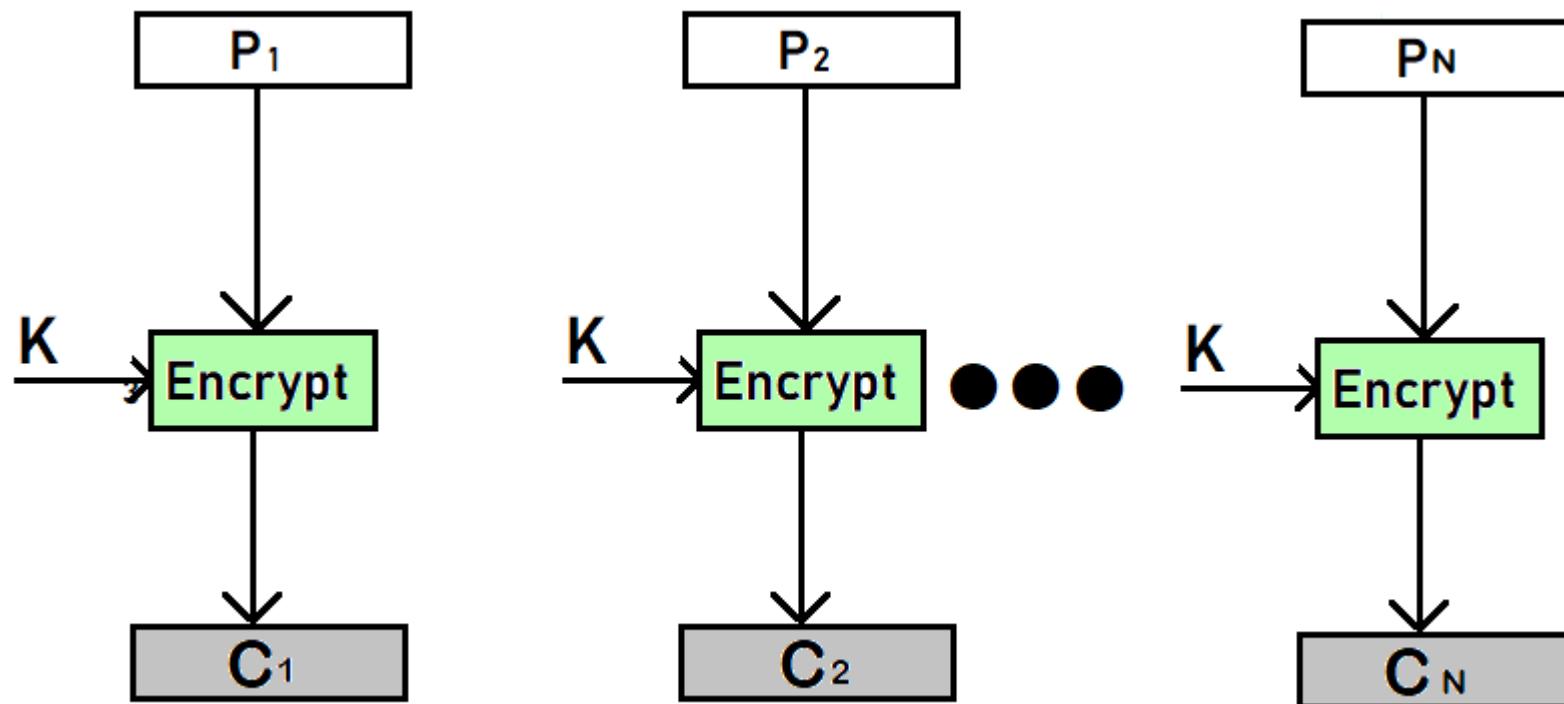
# Block Cipher Operation Modes

- **Block Cipher Operation Modes:** Operation modes determine how blocks of plaintext are encrypted.
- **Common Modes:**
  1. **Electronic Codebook (ECB):**
  2. **Cipher Block Chaining (CBC):**
  3. **Cipher Feedback (CFB):**
  4. **Output Feedback (OFB):** Similar to CFB but pre-generates keystream blocks.
  5. **Counter (CTR) Mode:** Uses a counter that increments with each block, allowing parallel encryption and decryption.

# Block Cipher Operation Modes

## Electronic Codebook (ECB):

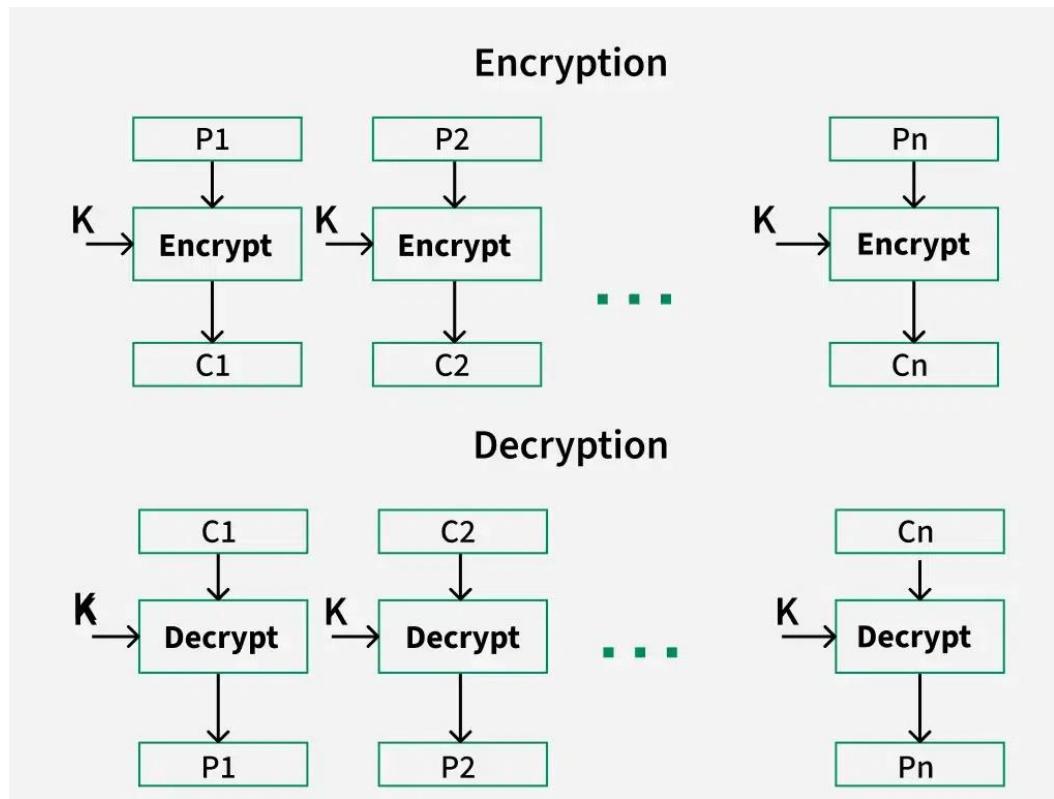
- 1. Description:** Each block is encrypted independently.
- 2. Weakness:** Identical plaintext blocks produce identical ciphertexts.



# Block Cipher Operation Modes

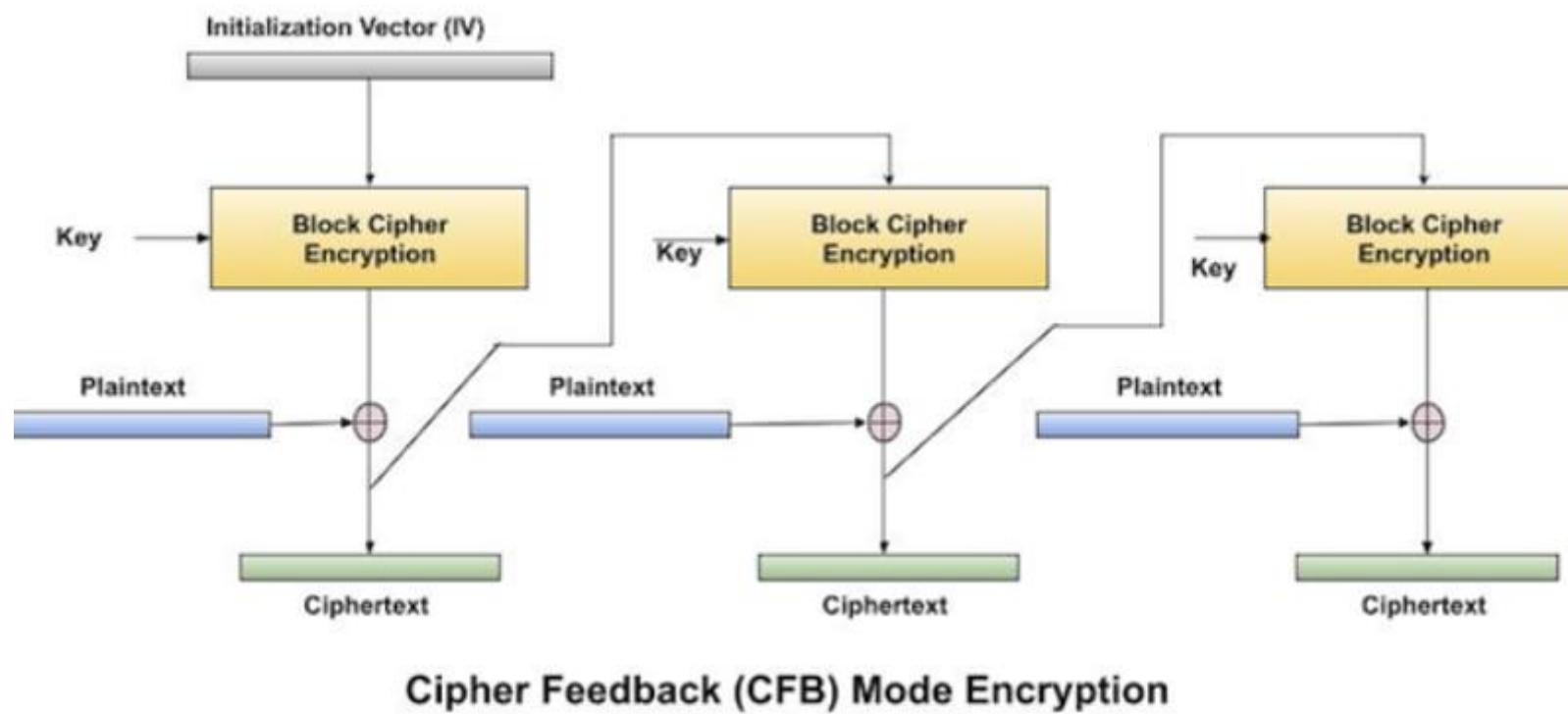
## Cipher Block Chaining (CBC):

- Description:** Each plaintext block is XORed with the previous ciphertext block before encryption.
- Advantage:** Introduces randomness, preventing patterns.



# Block Cipher Operation Modes

**Cipher Feedback (CFB):** Converts a block cipher into a stream cipher, suitable for streaming data.



# Asymmetric Ciphers

**Public-key cryptography, also known as asymmetric cryptography, is a cryptographic system that uses a pair of keys:**

- **Public Key:** Shared openly and used for encryption.
- **Private Key:** Kept secret and used for decryption.
- **Key Features:**
  - **Two-Key Mechanism:** One key encrypts, the other decrypts.
  - **Non-Repudiation:** Provides proof of the origin of data through digital signatures.
  - **Secure Key Distribution:** Eliminates the need to share secret keys over insecure channels.
- **Applications:**
  - **Secure Communication:** SSL/TLS for secure web browsing.
  - **Digital Signatures:** Ensuring authenticity and integrity of digital documents.
  - **Cryptocurrencies:** Used in Bitcoin and blockchain technologies.
- **Advantages:**
  1. **Enhanced Security:** No need to share private keys.
  2. **Scalability:** Easy to manage in large networks.
  3. **Authentication:** Ensures message authenticity and integrity.
- **Disadvantages:**
  1. **Slower than Symmetric Encryption:** Due to complex mathematical operations.
  2. **Key Length:** Requires longer keys for equivalent security compared to symmetric algorithms.

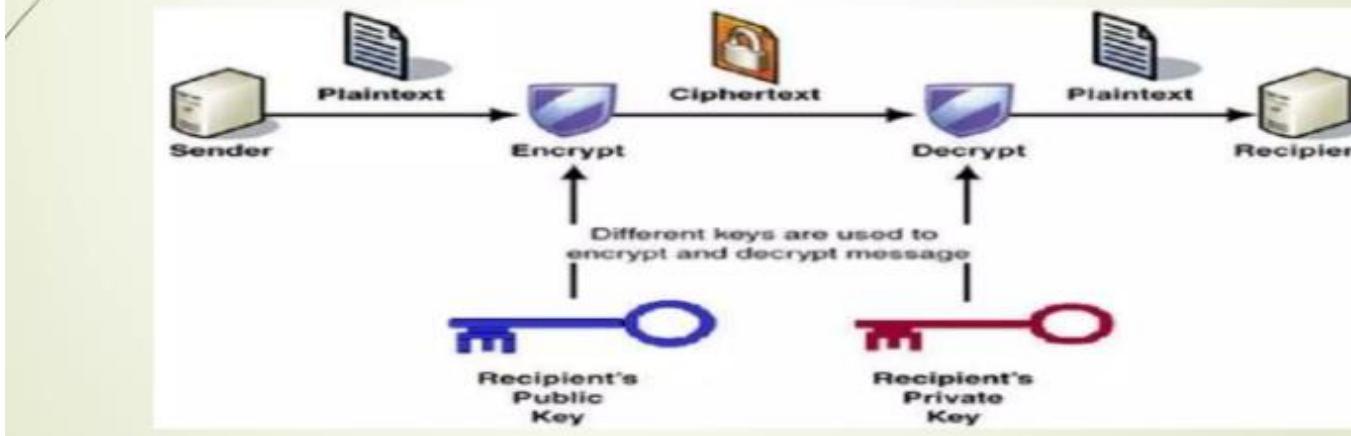
# RSA Algorithm

- The **RSA algorithm** (named after Rivest, Shamir, and Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission.
- **Key Concepts:**
  - Based on the mathematical difficulty of **factoring large prime numbers**.
  - Involves **modular arithmetic** and **Euler's theorem**.
- **Key Generation:**
  1. Choose two large prime numbers, p and q.
  2. Compute  $n=p \times q$  (modulus).
  3. Calculate  $\phi(n)=(p-1) \times (q-1)$  (Euler's totient function).
  4. Select public exponent e such that  $1 < e < \phi(n)$  and  $\text{gcd}(e, \phi(n))=1$ .
  5. Determine private key d such that  $d \times e \equiv 1 \pmod{\phi(n)}$ .
- **Encryption:** Ciphertext  $C = M^e \pmod{n}$ , where M is the plaintext.
- **Decryption:** Plaintext  $M = C^d \pmod{n}$ .
- **Applications:**
  - Secure email encryption
  - Digital signatures for document verification.
- **Security:**
  - Based on the difficulty of factoring large numbers.
  - Vulnerable to quantum attacks (future concern).

# RSA Algorithm Example

## RSA Algorithm :

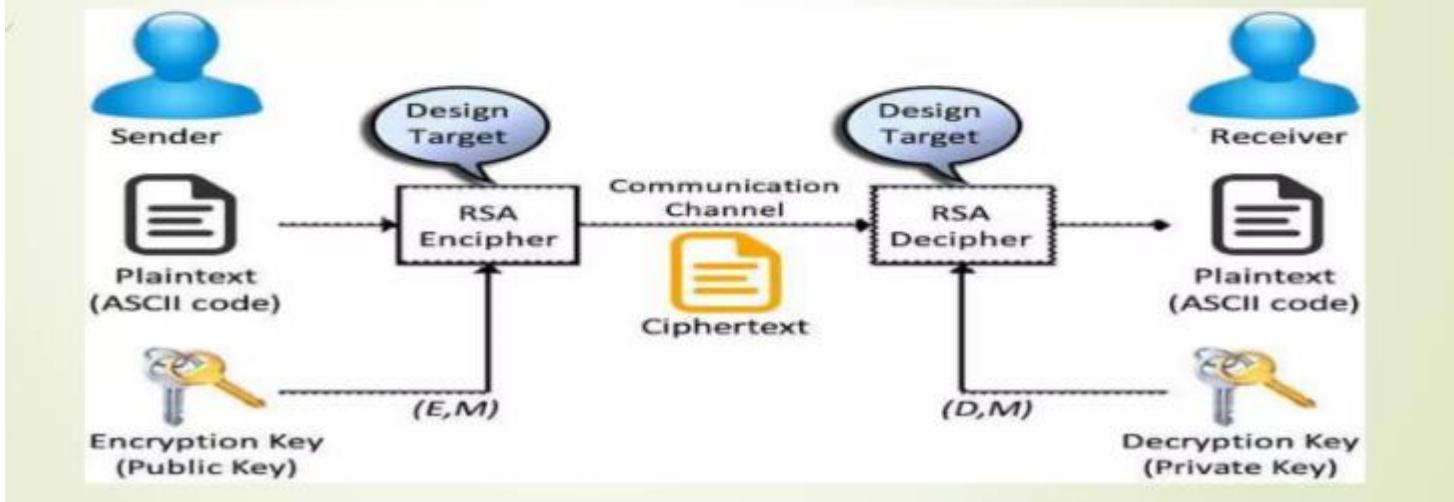
- ▶ RSA is invented by Rivest, Shamir and Adleman of MIT.
- ▶ It is most widely used for secure data transmission.
- ▶ RSA Algorithm is known as Public key Cryptography.



# RSA Algorithm Example

## RSA Algorithm :

- RSA Algorithm consists of following steps
  - Key Generation
  - Encryption
  - Decryption



# RSA Algorithm Example

## ► Key Generation :

1. Select two prime numbers  $p$  &  $q$
  2. Calculate  $n = p * q$
  3. Calculate  $m = \Phi(n) = (p - 1) * (q - 1)$
  4. Choose a small number  $e$ , co prime to  $m$ , with  
 $\text{GCD}(\Phi(n), e) = 1$ ,  $1 < e < \Phi(n)$
  5. Calculate  $d \text{ mod } \Phi(n) = 1$
6.  $d = ((\emptyset(n) * i) + 1) / e$

# RSA Algorithm Example

## ► Encryption :

Plaintext,  $M < n$

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

## ► Decryption :

Ciphertext,  $C$

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



# RSA Algorithm Example

## **RSA Algorithm Example :**

1. Let  $p = 3, q = 11$
2.  $n = p * q = 3 * 11 = 33$
3.  $\Phi(n) = (p-1)*(q-1) = 20$
4. Let  $e = 7$  such that  $1 < e < 20$ ,  $\text{GCD}(7, 20) = 1$
5.  $ed = 1 \bmod \Phi(n)$

$$7 * d = 1 \bmod 20$$

$$\text{So } 7 * d \bmod 20 = 1$$

►  $7 * 3 \bmod 20 = 1$

► Here  $d = 3$

# RSA Algorithm Example

## Encryption :

Plaintext,  $M < e$

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

Let  $M = 31$

$$\text{Then , } C = 3^{17} \bmod 33$$

$$\text{We get } C = 4$$

## Decryption :

Ciphertext,  $C$

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

$$M = 4^3 \bmod 33$$

$$M = 31$$

# Diffie-Hellman Key Exchange Protocol

- Developed by **Whitfield Diffie and Martin Hellman** in 1976, this protocol allows two parties to securely share a secret key over an insecure channel.
- Based on the **discrete logarithm problem**, which is computationally hard to solve.
- **Working Process:**
  1. Both parties agree on a large prime number  $p$  and a base  $g$  (public values).
  2. Each party selects a private key ( $a$  and  $b$ ).
  3. They compute public keys:
    - $i.$   $A = g^a \text{ mod } p$
    - $ii.$   $B = g^b \text{ mod } p$
  4. Exchange public keys.
  5. Compute shared secret key:
    1. Party 1:  $K = B^a \text{ mod } p$
    2. Party 2:  $K = A^b \text{ mod } p$
- Both parties now have the same **shared secret key**, without it being transmitted directly.
- **Applications:**
  - Secure key exchange in SSL/TLS protocols.
  - VPN and secure network communications.
- **Security:**
  - Vulnerable to **Man-in-the-Middle (MITM) attacks** if not authenticated.
  - Strong against eavesdropping due to the complexity of the discrete logarithm problem.

# ElGamal Cryptosystem

- The **ElGamal Cryptosystem** was developed by Tahar ElGamal and is based on the **Diffie-Hellman key exchange**.
- Relies on the hardness of the discrete logarithm problem.
- Provides both **encryption** and **digital signature** capabilities.
- **Key Generation:**
  1. Choose a large prime  $p$  and a primitive root  $g$ .
  2. Select a private key  $x$ .
  3. Compute public key  $y = g^x \text{ mod } p$ .
- **Encryption:**
  1. Choose a random number  $k$ .
  2. Compute  $c_1 = g^k \text{ mod } p$
  3. Compute  $c_2 = (M \times y^k) \text{ mod } p$ .
  4. Ciphertext =  $(c_1, c_2)$ .
- **Decryption:** Recover plaintext  $M = (c_2 \times c_1^{-x}) \text{ mod } p$ .
- **Applications:**
  - Encryption in PGP (Pretty Good Privacy).
  - Digital signature schemes.
- **Security:**
  - Provides **semantic security** if random  $k$  is chosen properly.
  - Larger ciphertext size compared to RSA.

# Elliptic Curve Cryptography (ECC)

- **Elliptic Curve Cryptography (ECC)** is a modern public-key cryptographic system based on the algebraic structure of **elliptic curves over finite fields**. Offers the same level of security as RSA with **much smaller key sizes**, making it efficient for devices with limited resources. Based on the difficulty of the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**.
- **Elliptic Curve Equation:**  $y^2 = x^3 + ax + b$  Where a and b are constants that define the curve.
- **Key Generation:**
  1. Choose an elliptic curve and a base point G.
  2. Select a private key d.
  3. Compute public key Q=d×G.
- **Encryption (Elliptic Curve Integrated Encryption Scheme - ECIES):**
  1. Generate an ephemeral key pair.
  2. Use public key encryption to derive a shared secret.
  3. Apply symmetric encryption to encrypt the message.
- **Applications:**
  - **Mobile Devices:** Due to lower computational requirements.
  - **Cryptocurrencies:** Bitcoin and other blockchain technologies.
  - **Secure Communication:** SSL/TLS encryption.
- **Advantages:**
- **Higher Security with Smaller Keys:** A 256-bit ECC key provides comparable security to a 3072-bit RSA key.
- **Efficient Performance:** Faster encryption/decryption processes with lower power consumption.
- **Security:** Resistant to traditional attacks but may be vulnerable to quantum computing attacks in the future.

# Quantum Cryptography: A New Era of Secure Communication

- **Introduction to Quantum Cryptography:** Quantum Cryptography leverages the principles of **quantum mechanics** to enable highly secure communication. Unlike classical cryptography, which relies on complex mathematical problems, quantum cryptography ensures security through the laws of physics.
- **Why Quantum Cryptography?**
  - Traditional encryption methods like RSA and ECC may become **vulnerable to quantum computers**.
  - Quantum cryptography offers **unbreakable security** due to its reliance on fundamental quantum properties.

# Quantum Cryptography: A New Era of Secure Communication

**Fundamentals of Quantum Mechanics Relevant to Cryptography:** Quantum cryptography is based on two fundamental principles of quantum mechanics:

1. Superposition
2. Entanglement

# Quantum Cryptography: A New Era of Secure Communication

## Principles of Superposition and Entanglement

### 1. Superposition:

- A quantum bit (**qubit**) can exist in **multiple states** (0 and 1) simultaneously until measured.
- Example: An electron's spin can be **both up and down** before observation.

### 2. Entanglement:

- Two qubits become **entangled** when their states are interdependent.
- Measuring one qubit immediately determines the state of the other, even at a distance.
- Used in **Entanglement-Based Quantum Key Distribution (QKD)**.
- **Implication in Cryptography:** Eavesdropping attempts disturb quantum states, alerting the legitimate users.

# Quantum Cryptography: A New Era of Secure Communication

- **Quantum Key Distribution (QKD):** Quantum Key Distribution (QKD) enables two parties to generate and share a secret encryption key securely.
- **Why QKD is Secure?**
  - **No-Cloning Theorem:** Quantum states cannot be copied without detection.
  - **Observation Causes Disturbance:** Any eavesdropper attempting to measure qubits alters their state, making interception impossible.

# Quantum Cryptography: A New Era of Secure Communication

- **BB84 Protocol: Working Mechanism, Key Exchange, and Security Proof:** BB84, proposed by Bennett and Brassard in 1984, is the most widely used QKD protocol.
- **Key Exchange Process:**
  1. **Bit Representation Using Polarized Photons:** Alice (sender) sends randomly polarized photons (e.g., horizontal, vertical, diagonal).
  2. **Measurement by Bob (Receiver):** Bob measures the photons randomly using different bases.
  3. **Key Reconciliation:** Alice and Bob compare a subset of their results to detect eavesdropping.
  4. **Final Key Agreement:** A secret key is established if no interference is detected.
- **Security Proof:** If an attacker intercepts the photons, their quantum states change, revealing the intrusion.

# Quantum Cryptography: A New Era of Secure Communication

- **E91 Protocol and Entanglement-Based QKD:** Proposed by **Artur Ekert in 1991**, the **E91 protocol** relies on **quantum entanglement** rather than classical measurements.
- **Key Process:**
  1. Alice and Bob receive **pairs of entangled photons** from a trusted source.
  2. They measure their photons using different orientations.
  3. Comparing results, they detect any eavesdropping attempts based on **Bell's Theorem**.
- **Advantage Over BB84:** More secure since entanglement correlations cannot be duplicated.

# Implementation Challenges and Real-World Applications

- **Challenges:**
  - **Photon Loss:** Signal degradation over long distances.
  - **Quantum Noise:** Measurement errors due to environmental factors.
  - **Hardware Limitations:** Requires specialized quantum hardware.
- **Real-World Applications:**
  - **Secure Banking Transactions:** Implemented in financial sectors to protect sensitive data.
  - **Government Communications:** Used for secure military and diplomatic channels.
  - **Quantum Networks:** China's **Micius satellite** demonstrated the first global QKD communication.

# Quantum Physics and Photon Reception

- **Quantum States of Photons:**
  - **Polarization Basis:** Used to encode quantum information.
  - **Quantum Bit (Qubit):** Represented by the polarization state of a single photon.
- **Measurement Techniques and Photon Polarization**
  1. **Linear Polarization:** Photons aligned horizontally or vertically.
  2. **Circular Polarization:** Rotating polarization states used for secure transmission.
  3. **Quantum Eraser Experiment:** Demonstrates that quantum information changes based on measurement choices.

# Cryptography with Photons

- **Secure Communication Using Quantum Channels**
  - **Quantum Repeaters:** Enhance long-distance QKD without classical relays.
  - **Quantum Teleportation:** Securely transfers quantum information over distances.
- **Photon-Based Encryption Techniques**
  - **Quantum One-Time Pad:** Uses QKD-generated keys to encrypt messages unconditionally.
  - **Quantum Secure Direct Communication (QSDC):** Eliminates the need for post-key exchange encryption.

# Implementation of Quantum Cryptographic Systems

- **Hardware Requirements (Quantum Sources, Detectors)**
  - **Quantum Random Number Generators (QRNGs):** Generate truly random keys.
  - **Single-Photon Detectors:** Essential for accurately measuring quantum states.
- **Practical Limitations and Future Developments**
  - 1. Scalability Issues:** Quantum systems are expensive and require specialized infrastructure.
  - 2. Quantum Computing Threat:** Future quantum computers may break existing encryption, necessitating **post-quantum cryptography**.
  - 3. Integration with Classical Networks:** Research continues on **hybrid quantum-classical encryption**.

# Conclusion: The Future of Quantum Cryptography

- **Unbreakable Security:** Quantum cryptography provides a **physically secure** alternative to mathematical encryption.
- **Global Adoption:** Countries and organizations are investing in **quantum-safe technologies**.
- **Next Steps:** Research is progressing in **quantum internet**, where secure global communication is based entirely on quantum principles.

# Introduction to Hash Functions

- A **hash function** is a cryptographic algorithm that takes an input (message) and produces a **fixed-length hash value (digest)**. This output uniquely represents the original data in a condensed form.
- **Mathematical Representation:**  $H(M)=h$  where:
  - $M$  = Input message
  - $h$  = Hash output (digest)
  - $H$  = Hash function
- **Key Features of Cryptographic Hash Functions:**
  1. **Deterministic:** The same input always produces the same output.
  2. **Fixed Output Size:** Regardless of input size, the hash output has a fixed length (e.g., 256-bit for SHA-256).
  3. **Fast Computation:** Hashing should be computationally efficient.
  4. **Irreversible (One-Way Function):** It should be computationally infeasible to derive the original input from the hash.
  5. **Unique (Collision Resistance):** Different inputs should produce different hash outputs.

# Properties of Cryptographic Hash Functions

A secure cryptographic hash function must satisfy the following properties:

## 1. Pre-Image Resistance (One-Way Property)

- Given a hash value  $h$ , it should be computationally infeasible to find an input  $M$  such that:  $H(M)=h$
- **Purpose:** Prevents attackers from retrieving original data (used in password storage).
- **Security Concern:** If a hash function is not pre-image resistant, brute-force attacks can reveal the input.

## • 2. Collision Resistance

- It should be infeasible to find two distinct inputs  $M_1$  and  $M_2$  that produce the same hash output:  $H(M_1)=H(M_2)$
- **Purpose:** Ensures data integrity and prevents tampering.
- **Security Concern:** If a collision exists, an attacker can replace data with malicious content having the same hash.

# Properties of Cryptographic Hash Functions

- **3. Avalanche Effect**
- A small change in input should produce a **completely different** hash output.

- **Example:**

$H("Hello") = 2CF24DBA5FB0A30E26E83B2AC5BCD773$

$H("hello") = 5D41402ABC4B2A76B9719D911017C592$

- The change in only **one letter** ( $H \rightarrow h$ ) produces a vastly different hash.

# Role of Hash Functions in Security

Cryptographic hash functions are used to ensure **data integrity** and **security** in various applications:

## A. Data Integrity

- Hashing ensures that data remains unaltered by comparing hash values before and after transmission.
- Used in **checksums, file verification, and digital signatures.**

## B. Digital Signatures

- Hashing is used in signing documents and verifying authenticity.
- A sender signs a document by hashing it and encrypting the hash with their private key.
- The recipient verifies it by decrypting the hash and comparing it to the computed hash of the received document.

# Common Cryptographic Hash Algorithms

<b>Algorithm</b>	<b>Output Size</b>	<b>Security Status</b>
<b>MD5</b>	128-bit	Broken (Collision Attacks)
<b>SHA-1</b>	160-bit	Broken (Practical Collisions Found)
<b>SHA-2</b>	224, 256, 384, 512-bit	Secure (Recommended)
<b>SHA-3</b>	224, 256, 384, 512-bit	More Secure than SHA-2

# Common Cryptographic Hash Algorithms

- **A. MD5 (Message Digest Algorithm 5)**
  - **Developed by:** Ronald Rivest (1991)
  - **Key Features:**
    - Produces a 128-bit hash.
    - Fast but vulnerable to collision attacks.
  - **Security Issue:** Collisions can be generated in seconds, making it insecure for cryptographic use.
  - **Usage Today:** Used in file integrity verification but **not recommended for security applications**.
- **B. SHA-1 (Secure Hash Algorithm 1)**
  - **Developed by:** NSA (1995)
  - **Key Features:**
    - Produces a 160-bit hash.
    - Used in SSL, TLS, and digital certificates.
  - **Security Issue:** In 2017, Google demonstrated a practical **collision attack** (SHA-1 shattered attack).
  - **Usage Today:** Deprecated and replaced by SHA-2.

# Common Cryptographic Hash Algorithms

- **C. SHA-2 (Secure Hash Algorithm 2)**
  - Developed by: NSA (2001)
  - **Key Features:**
    - Provides **multiple hash sizes**: SHA-224, SHA-256, SHA-384, SHA-512.
    - Resistant to collision attacks.
  - **Usage Today:** Widely used in **TLS, SSL, cryptocurrencies (Bitcoin uses SHA-256), and digital signatures**.
- **D. SHA-3 (Secure Hash Algorithm 3)**
  - Developed by: NIST (2015)
  - **Key Features:**
    - Uses **Keccak algorithm**, making it more resistant to attacks.
    - Has variants SHA3-224, SHA3-256, SHA3-384, SHA3-512.
  - **Usage Today:** Recommended for post-quantum security applications.

# Applications of Hash Functions

- **A. Password Storage**
- Hashing is used to store passwords securely.
- Example:
  - A password "mysecurepassword" is stored as:  $H(\text{mysecurepassword}) = 5F4DCC3B5AA765D61D8327DEB882CF99$
  - To protect against **rainbow table attacks**, a **random salt** is added before hashing:  $H(\text{"mysecurepassword"} + \text{salt}) = 8D969EEF6ECAD3C29A3A629280E686CF$

# Applications of Hash Functions

- **B. Data Verification**
- Hashing is used to verify **file integrity**.
- Example:
  - A file's hash is computed before transmission.
  - The receiver recomputes the hash after downloading.
  - If both hashes match, the file was **not tampered** with.
- **C. Blockchain and Cryptocurrencies**
- **Proof of Work (PoW)**: Bitcoin uses **SHA-256** to create new blocks in the blockchain.
- **Immutable Records**: Every block in a blockchain contains the hash of the previous block, ensuring data integrity.

# Applications of Hash Functions

- **Example: Bitcoin Block Hashing**
  - **Block 1 Hash: 0000XABCD1234...**
  - **Block 2 Hash: 0000EFG56789...**
- Any change in Block 1 will alter its hash, making **tampering impossible**.

# Why Are Hash Functions Crucial?

- Provide **data integrity** by detecting unauthorized modifications.
- Used in **password security, digital signatures, and cryptocurrencies**.
- **SHA-3 is the future** due to its enhanced security over SHA-2.

# Concept of Message Authentication

- **What is Message Authentication?**
  - Message Authentication ensures the **integrity** and **authenticity** of a message. It verifies that:
  - The message has **not been altered** during transmission.
  - The message is from the **intended sender** (authentic).

# Concept of Message Authentication

- A **Message Authentication Code (MAC)** is a cryptographic checksum generated from a message and a secret key. It is used to:
- **Authenticate** the sender of the message.
- **Verify** the integrity of the message.
- **Mathematical Representation:**
- $\text{MAC} = F(K, M)$
- Where:
  - M = Message
  - K = Secret Key
  - F = MAC generation algorithm

# Difference Between MACs and Hash Functions

Criteria	MAC (Message Authentication Code)	Hash Function
<b>Key Requirement</b>	Requires a <b>secret key</b>	<b>No key</b> required
<b>Purpose</b>	Ensures both <b>integrity</b> and <b>authenticity</b>	Ensures <b>integrity</b> only
<b>Usage</b>	Used in secure communications (e.g., APIs, banking)	Used for data verification (e.g., file checksums)
<b>Security Level</b>	Stronger security due to key-based verification	Weaker if not combined with a key

## Key Point:

- **MAC = Hash Function + Secret Key** (in HMAC)
- MACs provide **authentication**, while hashes provide **integrity**.

# Importance of MACs in Data Integrity and Authentication

- **Data Integrity:** Detects any unauthorized modifications to the message during transmission.
- **Authentication:** Ensures the message comes from a trusted sender because only the sender and receiver know the secret key.
- **Non-Repudiation (in certain cases):** Provides proof that the message was sent by a specific entity.
- **Real-World Applications:**
  - Secure email protocols (e.g., **SSL/TLS**).
  - Financial transactions (e.g., **banking APIs**).
  - Secure communication in **IoT devices**.

# Types of MACs

## A. HMAC (Hash-Based Message Authentication Code)

- HMAC is a MAC that uses a **cryptographic hash function** (like SHA-256) and a **secret key** to produce the authentication code.
- **Formula:**
- $\text{HMAC}(K, M) = H((K' \oplus \text{opad}) \parallel H((K' \oplus \text{ipad}) \parallel M))$
- Where:
  - $K'$  = Secret key (padded/truncated)
  - $H$  = Hash function (e.g., SHA-256)
  - $\text{opad}$  = Outer padding (0x5c)
  - $\text{ipad}$  = Inner padding (0x36)
  - $\parallel$  = Concatenation operator
  - $\oplus$  = XOR operation

# Types of MACs

- **How HMAC Works:**

1. The secret key is **padded** to the block size of the hash function.
2. It is combined with the **inner padding** and the message, then hashed.
3. The result is combined with the **outer padding** and hashed again to produce the final HMAC.

- **Security Strength:**

- Depends on the **strength of the hash function** (e.g., SHA-256).
- Resistant to **length extension attacks** that affect plain hashes.

# Types of MACs

## B. CMAC (Cipher-Based Message Authentication Code)

- CMAC is a MAC that uses a **symmetric block cipher** (like AES) instead of a hash function.
- **Formula:**
- $\text{CMAC}(K, M) = \text{AES}(K, M) \oplus \text{Subkey}$
- Where:
  - K = Symmetric secret key
  - M = Message
  - **Subkey** = Derived from the encryption process

# Types of MACs

- **How CMAC Works:**

1. The message is divided into blocks.
2. Each block is processed using a **block cipher** (e.g., AES).
3. The final block is XORed with a **subkey** to generate the CMAC.

- **Security Strength:**

- **Strong security** due to reliance on AES (or similar block ciphers).
- More efficient than HMAC when hardware-based encryption is available.

# Working of HMAC and CMAC

## Key Generation and Verification Processes

- **HMAC Generation:**
  1. **Key Agreement:** Both sender and receiver share a secret key K.
  2. **Message Input:** The sender applies HMAC using K and the message M.
  3. **HMAC Generation:** The output (digest) is appended to the message and sent to the receiver.
- **HMAC Verification:**
  1. The receiver uses the same key K to generate an HMAC for the received message.
  2. If the computed HMAC matches the received HMAC, the message is **authentic** and **unaltered**.

# Working of HMAC and CMAC

- **CMAC Generation:**

1. **Key Sharing:** A symmetric key KKK is securely shared.
2. **Encryption Process:** The message is encrypted using AES, and a subkey is applied.
3. **CMAC Output:** The final encrypted block acts as the MAC.

- **CMAC Verification:**

1. The receiver re-encrypts the received message using the same key and subkey.
2. If the computed CMAC matches the received CMAC, the message is **authenticated**.

# Working of HMAC and CMAC

## B. Security Considerations

- **HMAC Security:**
  - Strong if the **underlying hash function** (e.g., SHA-256) is secure.
  - Resistant to collision attacks due to the key-based double hashing process.
  - Vulnerable if the **key is weak** or poorly managed.
- **CMAC Security:**
  - Inherits the security of AES (or the underlying cipher).
  - Resistant to forgery if the key is kept secret.
  - Efficient in **hardware-based systems** (like embedded devices).

# Practical Applications of HMAC and CMAC

Application Area	HMAC	CMAC
Network Security	SSL/TLS authentication	IPsec encryption
Data Integrity	Verifying software updates	Secure firmware validation
Banking & Finance	API authentication (OAuth)	Secure PIN verification
Cryptocurrencies	Blockchain transaction signing	Payment processing in EMV cards
IoT Devices	Device-to-cloud authentication	Lightweight encryption in smart devices

# Why Are MACs Important?

- **MACs = Integrity + Authentication:** They ensure messages are **authentic** and **unchanged**.
- **HMAC** is versatile for software-based systems (secure APIs, emails).
- **CMAC** excels in hardware applications (smart cards, IoT devices).
- Both play a vital role in modern cryptography, securing **financial transactions, communications, and digital identities**.

# What is a Digital Signature?

- A **Digital Signature** is a cryptographic technique used to:
  - Verify the authenticity of digital messages or documents.
  - Ensure data integrity, confirming that the content has not been altered.
  - Provide non-repudiation, meaning the sender cannot deny their involvement.
- It works using **asymmetric cryptography**, where:
  - A **private key** is used to **sign** the message.
  - A **public key** is used to **verify** the signature.

# What is a Digital Signature?

- Purpose and Importance in Cybersecurity
  - 1. Authentication:** Verifies the sender's identity.
  - 2. Data Integrity:** Ensures the message hasn't been tampered with.
  - 3. Non-Repudiation:** Prevents the sender from denying the origin of the message.
- Real-World Use Cases:
  - Secure emails (e.g., **PGP** encryption).
  - Legal documents and contracts.
  - Blockchain transactions.

# Digital Signatures vs. Handwritten Signatures

Criteria	Digital Signature	Handwritten Signature
Technology	Based on cryptographic algorithms	Ink-based signature
Verification	Verified using public-key cryptography	Verified visually
Tamper Detection	Alteration invalidates the signature	Alterations can go unnoticed
Security Level	High (depends on encryption strength)	Low (can be forged easily)
Non-Repudiation	Strong, cryptographic proof	Weak, can be denied or disputed
Usage	Emails, software, digital contracts	Paper documents, physical agreements

# Working of Digital Signature Algorithms

## A. Signing and Verification Processes

### Step 1: Signing Process (Sender Side)

**1. Hashing the Message:** The original message  $M$  is passed through a **hash function** (e.g., SHA-256) to create a unique digest  $H(M)$ .

**2. Encrypting the Hash (Signing):** The hash is encrypted using the sender's **private key** to generate the **digital signature**:  
 $\text{Signature} = \text{EncryptPrivate Key}(H(M))$

**3. Sending:** The sender sends both the **original message** and the **digital signature** to the receiver.

# Working of Digital Signature Algorithms

## **Step 2: Verification Process (Receiver Side)**

**1. Hashing the Received Message:** The receiver hashes the received message to generate  $H(M')$ .

**2. Decrypting the Signature:** The receiver decrypts the digital signature using the sender's **public key** to retrieve the original hash:  
 $H(M) = \text{DecryptPublic Key}(\text{Signature})$

**3. Comparison:** The receiver compares  $H(M)$  and  $H(M')$ :

- If they match, the message is **authentic** and **unaltered**.
- If they don't match, the message has been **tampered with** or the signature is **invalid**.

# Key Generation, Hashing, and Encryption Integration

**1. Key Generation:** A pair of **asymmetric keys** is generated:

1. **Private Key:** Used for signing.

2. **Public Key:** Shared with others to verify signatures.

**2. Hashing:** A **cryptographic hash function** ensures data integrity.

**3. Encryption:** The hash is encrypted with the sender's private key to create the signature.

# Common Digital Signature Algorithms

## A. RSA Digital Signature Algorithm

- **Based on:** The difficulty of **factoring large prime numbers**.
- **Process:**
  - Hash the message using SHA-256 (or similar).
  - Encrypt the hash with the sender's **private RSA key**.
  - Verify by decrypting the signature with the sender's **public RSA key**.
- **Applications:** SSL/TLS certificates, software signing, secure emails.

# Common Digital Signature Algorithms

## B. DSA (Digital Signature Algorithm)

- **Based on:** Discrete logarithm problem.
- **Process:**
  - Generate a signature using a combination of private key and random numbers.
  - Verification involves mathematical operations with the public key and hash.
- **Faster** than RSA in signature generation but **slower** in verification.
- **Applications:** Government communications, secure protocols.

# Common Digital Signature Algorithms

## C. ECDSA (Elliptic Curve Digital Signature Algorithm)

- **Based on: Elliptic Curve Cryptography (ECC).**
- **Advantages:**
  - Smaller key sizes with stronger security.
  - Faster signature generation and verification.
- **Applications:** Cryptocurrencies (e.g., Bitcoin), mobile security, IoT devices.

# Applications of Digital Signatures

- **A. Secure Emails**
  - Ensures that emails are **authentic** and **unmodified**.
  - Tools: **PGP (Pretty Good Privacy)** and **S/MIME**.
- **B. Software Distribution: Code Signing Certificates** verify that software comes from a trusted source and hasn't been tampered with.
- **C. Blockchain Technology**
  - Used in **cryptocurrencies** (like Bitcoin) to sign transactions.
  - Ensures data integrity and prevents double-spending.

# Legal Frameworks and Compliance Requirements

## 1. eIDAS Regulation (EU)

- **eIDAS (Electronic Identification, Authentication, and Trust Services)** is an EU regulation that standardizes electronic signatures.
- **Types of Electronic Signatures:**
  - Simple Electronic Signature (SES)
  - Advanced Electronic Signature (AES)
  - Qualified Electronic Signature (QES): Equivalent to handwritten signatures in legal contexts.

## 2. Other Global Standards

- **ESIGN Act (USA):** Legal recognition of electronic signatures.
- **UNCITRAL Model Law (International):** Framework for global electronic transactions.

# The Importance of Digital Signatures

- **Secure Communication:** Verifies sender identity and message integrity.
- **Global Acceptance:** Legal recognition in many countries (eIDAS, ESIGN).
- **Future Trends:** Integration with **blockchain**, **IoT**, and **post-quantum cryptography**.

# Conclusion

## Key Takeaways

- **Symmetric Ciphers:** Fast encryption with a single key (e.g., DES, AES).
- **Asymmetric Ciphers:** Public-private key encryption for secure communication (e.g., RSA, ECC).
- **Quantum Cryptography:** Unbreakable security based on quantum mechanics (e.g., BB84 Protocol).
- **Hash Functions:** Ensuring data integrity (e.g., SHA-2, SHA-3).
- **Message Authentication Codes (MACs):** Verifying message authenticity (e.g., HMAC, CMAC).
- **Digital Signatures:** Providing non-repudiation and authentication (e.g., RSA, ECDSA).

# Conclusion

## Core Principles

- **Confidentiality:** Protecting data through encryption.
- **Integrity:** Ensuring data is unaltered using hashes and MACs.
- **Authentication:** Verifying sender identity with digital signatures.
- **Non-Repudiation:** Preventing denial of actions via cryptographic proofs.

# Conclusion

## Future Trends

- **Post-Quantum Cryptography:** Preparing for quantum threats.
- **Blockchain Security:** Leveraging cryptography for decentralized applications.
- **Zero Trust Security:** Continuous verification in dynamic environments.