

Unit 1

Introduction and Classical Ciphers

(7 Hours)

1.4. Security: Computer Security, Information Security, Network Security, CIA Triad, Cryptography, Cryptosystem, Cryptanalysis, Security Threats and Attacks, Security Services, Security Mechanisms

1.5. Classical Cryptosystems: Substitution Techniques: Ceasar, Monoalphabetic, Playfair, Hill, Polyalphabetic ciphers, One-time pad Transposition Techniques: Rail Fence Cipher

1.6. Modern Ciphers: Block vs. Stream Ciphers, Symmetric vs. Asymmetric Ciphers

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

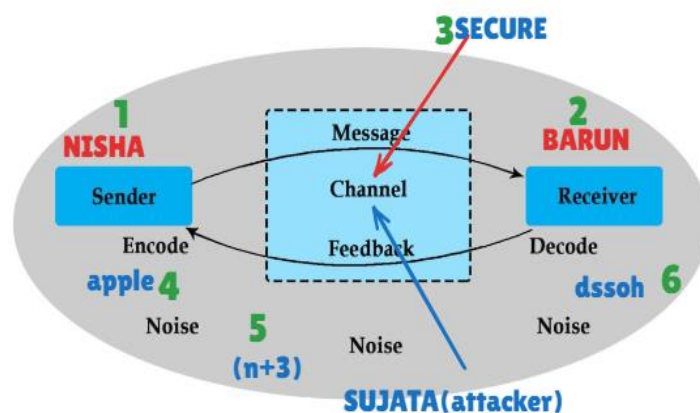
Security

🔒 What is Security?

Security refers to the protection of systems, data, networks, and resources from unauthorized access, misuse, modification, or destruction.

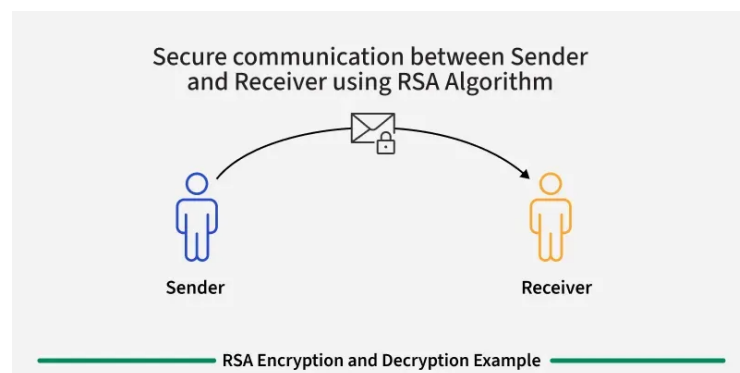
In computing, **security ensures that information and technology resources remain:**

- **Confidential** (only authorized people can access)
- **Accurate & Unchanged** (integrity)
- **Available** when needed (availability)



Security applies to:

- Computer systems
- Information/data
- Networks
- Users and processes



Formal definition:

Security is the practice of defending information and systems from threats and attacks to ensure confidentiality, integrity, and availability.

🔒 Why is Security Needed?

Security is essential for multiple reasons:

1 To Protect Sensitive Information

Prevent unauthorized people from seeing private or confidential data.

Example:

Bank account details, passwords, exam results, personal identity data.

2 To Maintain Data Integrity

Ensure data is not intentionally or accidentally changed.

Example:

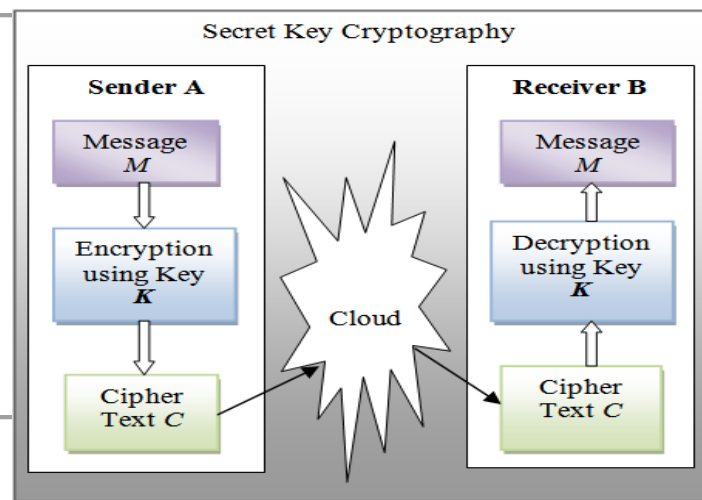
Medical records must remain accurate; financial transactions must not be altered.

3 To Ensure System and Service Availability

Security protects systems from failures, attacks, or downtime.

Example:

Servers must resist DoS attacks so websites remain online.



4 To Prevent Unauthorized Access

Only authorized users should access systems and data.

Example:

Login authentication, role-based access control.

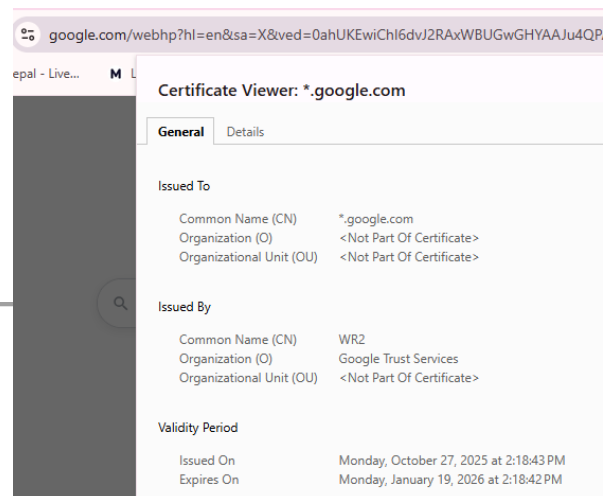
5 To Protect Against Cyber Attacks

Security guards against:

- Viruses
- Worms
- Trojans
- Ransomware
- Hacking attempts
- Man-in-the-middle attacks

6 To Build Trust in Digital Systems

Users must trust the system for secure online banking, e-commerce, cloud storage, etc.



7 To Comply with Legal and Ethical Requirements

Many sectors (banking, healthcare, government) are legally required to secure their systems and data.

8 To Prevent Financial Loss

Security breaches can lead to:

- Theft of money
- Loss of business
- Repair costs
- Legal penalties

Step / Action	Terminal Command / Example	Security Concept Involved	What Happens / Why Needed
https://killercoda.com/killer-shell-cka/scenario/playground			
1. Create a file	<i>cat > file.txt</i> <i>hello this text</i>	File Creation	User creates a plaintext file using basic Linux command.
2. Default file permissions	<i>ls -l file.txt → -</i> <i>rwx-r--r--</i>	Confidentiality (risk)	Others can read the file; data is not private.

Step / Action	Terminal Command / Example	Security Concept Involved	What Happens / Why Needed
https://killercoda.com/killer-shell-cka/scenario/playground			
3. Restrict access	<i>chmod 600 file.txt</i>	Confidentiality (protection)	Only the owner can read/write the file. Prevents unauthorized access.
4. Make file read-only	<i>chmod 400 file.txt</i>	Integrity (protection)	No one can modify the file (including owner unless permissions changed).
5. Detect changes using checksum	<i>sha256sum file.txt</i>	Integrity (verification)	Hash value ensures file hasn't been altered. If hash changes → file tampered.
6. Example of modification	Another user runs: <i>echo "hacked" >> file.txt</i>	Threat: Integrity Attack	File content changes → unauthorized modification.
7. Availability risk (accidental deletion)	<i>rm file.txt</i>	Availability Issue	File can be lost due to mistakes or attacks.
8. Create backup	<i>cp file.txt ~/Backup/file.txt</i>	Availability (protection)	Ensures file is still accessible even if primary file is lost.
9. Check who accessed system	<i>last</i>	Authentication Monitoring	Helps monitor who logged in; detects suspicious activity.
10. Verify file authenticity after download	<i>sha256sum downloaded_file.deb</i>	Integrity & Trust	Ensures downloaded file is not corrupted or infected.
11. Protect system on public WiFi	<i>sudo ufw enable</i>	Network Security / Confidentiality	Firewall blocks unwanted connections.

Step / Action	Terminal Command / Example	Security Concept Involved	What Happens / Why Needed
https://killercoda.com/killer-shell-cka/scenario/playground			
12. Encrypt sensitive file	<i>openssl enc -aes-256-cbc -salt -in file.txt -out file.enc</i>	Confidentiality via Cryptography	Even if someone copies the file, they cannot read it.
13. Decrypt encrypted file	<i>openssl enc -aes-256-cbc -d -in file.enc -out file_new.txt</i>	Restoring Availability	Authorized user reads file safely.

Computer Security vs Information Security vs Network Security

Criteria	Computer Security	Information Security	Network Security
Definition	Protects computer systems (hardware, OS, software) from unauthorized access or damage.	Protects <i>information/data</i> in any form (digital, physical, printed) from unauthorized access, alteration, or destruction.	Protects data <i>during transmission</i> across networks and safeguards network devices.
Focus Area	Individual computers, devices, endpoints.	Data confidentiality, integrity, and availability.	Communication channels, network traffic, routers, switches, Wi-Fi, etc.
Main Goal	Ensure system is secure and functions correctly.	Keep data secure, accurate, private, and accessible.	Protect network infrastructure and secure data in motion.
What is Protected?	Operating system, hard disks, applications, local files.	Sensitive data, records, documents, credentials.	Packets, network connections, network devices, communication pathways.

Criteria	Computer Security	Information Security	Network Security
Threat Examples	Malware, unauthorized logins, OS vulnerabilities.	Data breach, data tampering, insider misuse, data theft.	Sniffing, spoofing, MITM attack, DoS attack, ARP poisoning.
Techniques Used	Antivirus, OS hardening, user authentication, access control, disk encryption.	Encryption, hashing, access control, data classification, backups.	Firewalls, IDS/IPS, VPN, SSL/TLS, IPsec, packet filtering.
Scope	Narrow (device-level protection).	Broad (data-level protection).	Network-level protection across multiple devices.
Example	Protecting a laptop from viruses and unauthorized users.	Encrypting exam marks database or protecting students' personal details.	Securing data while submitting an online form via HTTPS or preventing Wi-Fi attacks.
Responsible Team	System administrators, desktop security team.	Information security team, compliance team.	Network administrators, cybersecurity engineers.
Relation	Component of overall system security.	Superset that includes computer & network security.	Overlaps with both but focuses only on network communication.

CIA Triad

The **CIA Triad** is the foundational model of information security. It consists of **three core principles**:

- **C – Confidentiality**
- **I – Integrity**
- **A – Availability**

These three ensure that information remains protected from unauthorized use, modification, and disruption.



1 Confidentiality

Definition:

Ensures that **only authorized users** can access information.
Prevents disclosure of data to unauthorized persons.

Examples:

- Passwords protect files from unauthorized access
- Encryption (AES, RSA) makes data unreadable to attackers
- Access control (file permissions: chmod 600)
- Secure communication (HTTPS, SSL/TLS)

Threats to Confidentiality:

- Unauthorized access
- Eavesdropping/sniffing
- Data leakage
- Shoulder surfing

2 Integrity

Definition:

Ensures that information remains **accurate, original, complete, and unaltered**.

Examples:

- Hash functions (SHA-256) verify file integrity
- Digital signatures ensure authenticity + no modification
- Checksums used when downloading software
- Version control to detect unauthorized changes

Threats to Integrity:

- Data tampering
- Unauthorized modification
- Malware altering files
- Accidental edits or deletion

3 Availability**Definition:**

Ensures that information and systems are **accessible whenever required** by authorized users.

Examples:

- Backups to recover lost data
- Redundant systems (RAID, load balancers)
- Up-to-date antivirus & patches
- DoS attack protection
- Power backups & stable network

Threats to Availability:

- Hardware failure
- DoS/DDoS attacks
- System crashes
- Natural disasters
- Ransomware locking files

CIA Triad Table Summary

Component	Meaning	Goal	Example
Confidentiality	Only authorized users access data	Privacy	Passwords, encryption
Integrity	Data remains accurate & unchanged	Trust	Hashing, digital signatures
Availability	Data/services available when needed	Access	Backups, DoS protection

CIA Triad lab scenarios.

TABLE 1: Confidentiality Lab Scenario

Step	Command / Action	Output / Observation	Security Concept
1. Create a file	<code>echo "This is a secret message" > secret.txt</code>	File created	Start of confidentiality scenario
2. Check default permissions	<code>ls -l secret.txt</code> → <code>-rw-r--r--</code>	File readable by others	Confidentiality is not ensured
3. Restrict permissions	<code>chmod 600 secret.txt</code>	<code>-rw-----</code>	Only owner can read/write → Confidentiality ensured
4. Optional encryption	<code>openssl enc -aes-256-cbc -salt -in secret.txt -out secret.enc</code>	Encrypted file created	Extra confidentiality layer

Step	Command / Action	Output / Observation	What It Proves
1. Create a file	<code>echo "Student Marks: 86" > marks.txt</code>	File created	Base file for integrity test
2. Generate hash	<code>sha256sum marks.txt > marks.hash</code>	Hash stored in marks.hash	Integrity baseline created
3. Modify file	<code>echo "Student Marks: 56" > marks.txt</code>	File changed	File integrity compromised
4. Verify integrity	<code>sha256sum marks.txt vs cat marks.hash</code>	Hash mismatch	Unauthorized modification detected

 **TABLE 3: Availability Lab Scenario**

Step	Command / Action	Output / Result	Security Concept
1. Create important file	<code>echo "Project work: Cryptography lab" > project.txt</code>	Project file created	Base for availability test
2. Create backup	<code>cp project.txt project_backup.txt</code>	Backup created	Ensures availability
3. Simulate deletion	<code>rm project.txt</code>	File deleted	Data unavailable
4. Restore backup	<code>cp project_backup.txt project.txt</code>	File restored	Availability maintained

 **TABLE 4: Combined CIA Triad Demonstration**

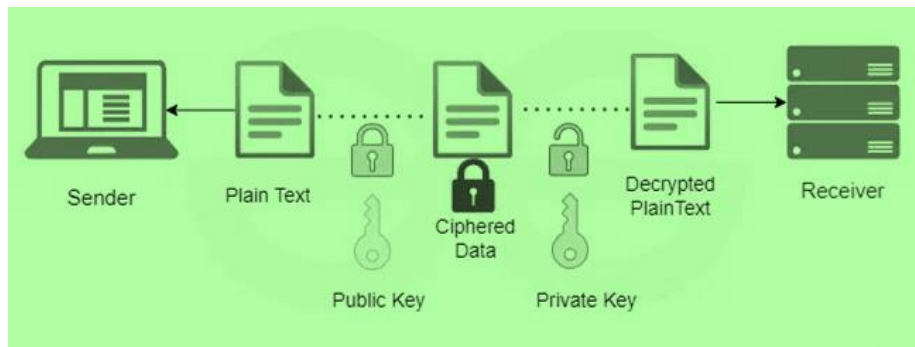
Concept	Command / Action	Observation	What It Shows
Confidentiality	<code>chmod 600 cia_demo.txt</code>	Only owner can read/write	Data privacy ensured
Integrity	<code>sha256sum cia_demo.txt > cia_demo.hash</code>	Hash saved	File integrity baseline created
Integrity Test	Modify file: <code>echo "hacked!" >> cia_demo.txt</code>	File changed	Integrity broken
Integrity Verification	<code>sha256sum cia_demo.txt</code>	Hash mismatch	Modification detected
Availability	Backup: <code>cp cia_demo.txt cia_demo_backup.txt</code>	Backup exists	Data available even if lost
Availability Restore	Restore: <code>cp cia_demo_backup.txt cia_demo.txt</code>	File recovered	Availability preserved

🚩 Summary Table: CIA Triad Concepts

CIA Component	Meaning	Example in Lab	Purpose
Confidentiality	Prevent unauthorized access	<code>chmod 600</code> , encryption	Protects privacy of data
Integrity	Ensure data is accurate & unchanged	Hash check (sha256sum)	Detects tampering or corruption
Availability	Ensure data is accessible when needed	Backup & restore	Prevents data loss and downtime



- The **science of secret writing**.
- Converts **plaintext** → **ciphertext** (encryption) and **ciphertext** → **plaintext** (decryption).
- Uses mathematical algorithms and keys.
- Ensures:
 - ✓ Confidentiality
 - ✓ Integrity
 - ✓ Authentication
 - ✓ Non-repudiation



Example:

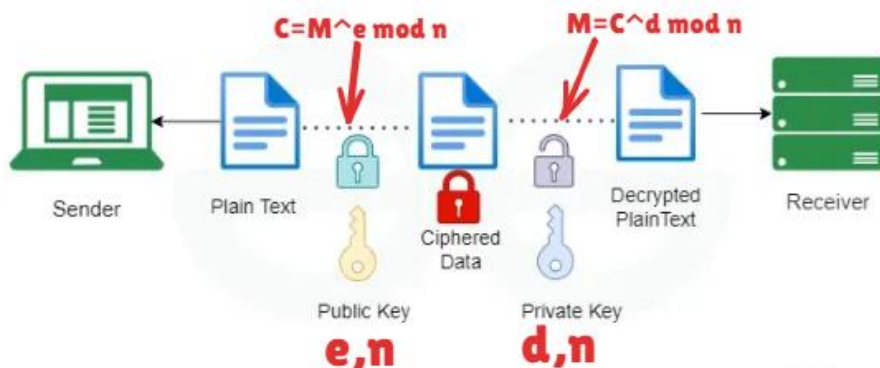
AES encrypts data so only authorized users can read it.

Cryptosystem

- A **complete framework** that supports secure communication.
- Includes:
 1. Plaintext
 2. Ciphertext
 3. Encryption algorithm
 4. Decryption algorithm
 5. Key(s)
 6. Key generation

Example: *RSA Cryptosystem* uses:

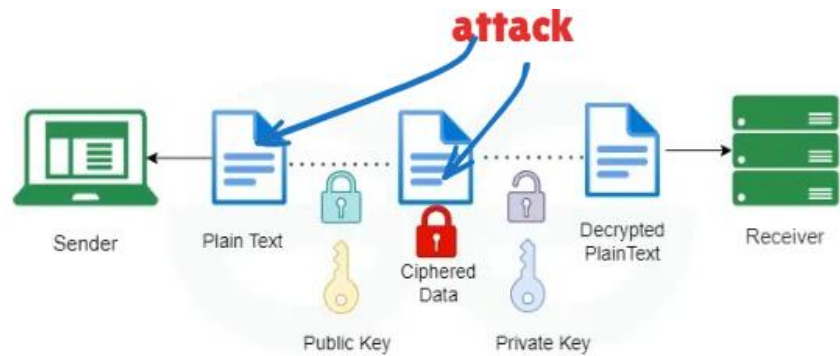
- Public key (e, n)
- Private key (d, n)
- Encryption: $C = M^e \bmod n$
- Decryption: $M = C^d \bmod n$



- The art/science of **breaking ciphers**.
- Goal: recover plaintext or key *without authorization*.
- Used by attackers *and* security professionals (ethical).

Types of Cryptanalysis Attacks:

- Ciphertext-only attack
- Known plaintext attack
- Chosen plaintext attack
- Chosen ciphertext attack
- Brute force attack



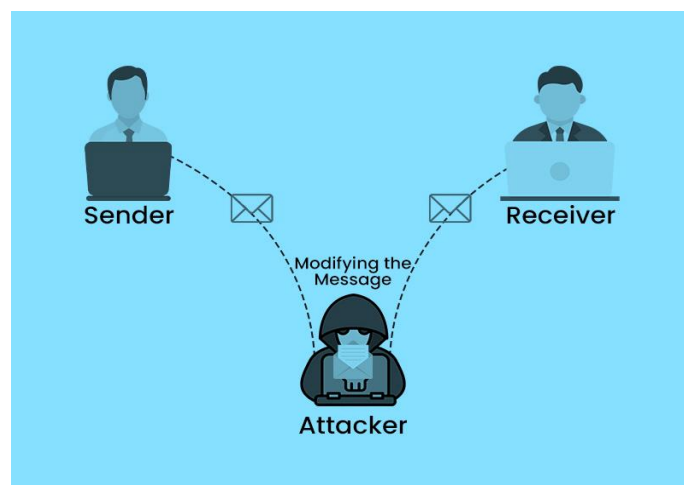
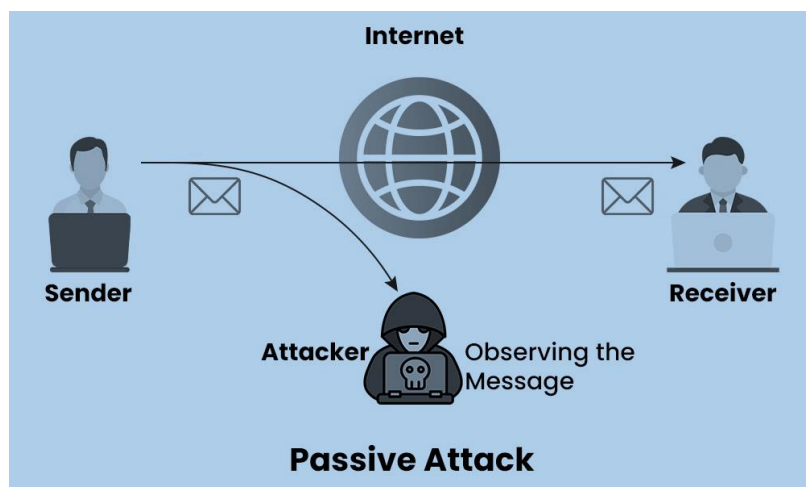
Example:

Trying all 25 keys in Caesar cipher to break encryption.

✅ 3. Summary Table (For Quick Revision)

Feature	Cryptography	Cryptosystem	Cryptanalysis
Focus	Protect data	Provide structure for encryption	Break or attack crypto
Users	Designers / security engineers	System architects	Attackers & analysts
Goal	Secure information	Establish secure communication	Find weaknesses
Output	Ciphertext, signatures	Full system (keys + algorithms)	Plaintext/key recovery
Nature	Constructive	Structural	Destructive / analytical

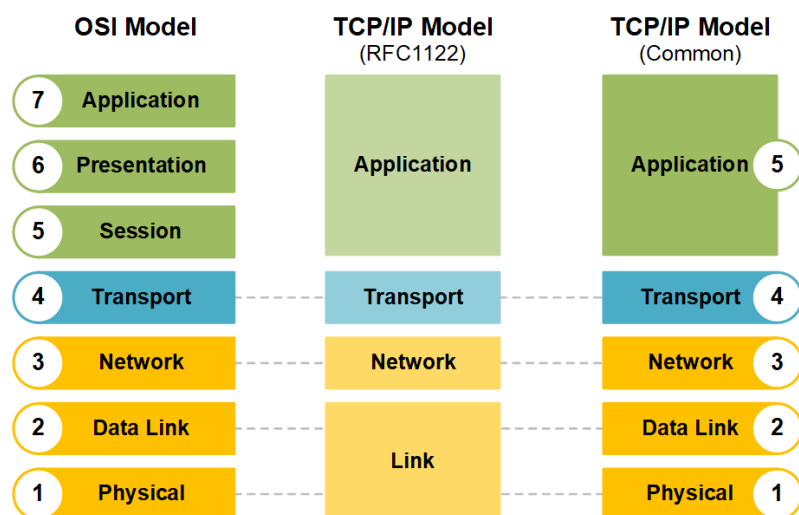
🔪 1. SECURITY THREATS & ATTACKS



Category	Definition	Examples	Impact
Passive Attacks	Attacker only <i>observes</i> data but does not modify it.	Eavesdropping, traffic analysis.	Loss of confidentiality.
Active Attacks	Attacker <i>modifies</i> data or disrupts the system.	Masquerading, DoS, replay attack.	Loss of integrity + availability.
Malware	Malicious software designed to damage or exploit systems.	Virus, worm, trojan, ransomware.	Data loss, corruption, system slowdown.
Social Engineering	Manipulating people to reveal confidential info.	Phishing, fake login pages, phone scams.	Credential leakage, identity theft.
Network Attacks	Attacks on communication channels or networks.	MITM, ARP spoofing, sniffing, session hijacking.	Data theft, impersonation.
Web Attacks	Target web apps/services.	SQL Injection, XSS, CSRF.	Unauthorized access, data breach.
Physical Threats	Damage to hardware.	Theft, fire, power failure.	Total data/system loss.

2. SECURITY SERVICES

Security Services are *what security SHOULD achieve* (OSI model).



Security Service	Goal	Description	Example
Confidentiality	Privacy	Prevent unauthorized access to data.	Encryption (AES), file permissions.
Integrity	Accuracy	Protect data from unauthorized modification.	Hashing (SHA-256), digital signatures.
Authentication	Identity verification	Check if user/system is genuine.	Password login, certificates.
Authorization / Access Control	Permission control	Decide what actions user can perform.	RBAC, ACL.
Non-Repudiation	Accountability	Sender cannot deny sending data.	Digital signatures.
Availability	Access	Ensure system/services are available when needed.	Backups, DoS protection.
Auditing / Accountability	Tracking	Record user and system activities.	System logs, audit trails.



Security Mechanisms are *how security is implemented* (tools, techniques).

Security Mechanism	Purpose	Real Example / Tool
Encryption	Protect confidentiality	AES, RSA, TLS, SSH
Hash Functions	Ensure integrity	SHA-256, MD5 (deprecated)
Digital Signatures	Integrity + non-repudiation + authentication	RSA Signatures, DSA
Authentication Mechanisms	Verify identity	Passwords, OTP, biometrics, OAuth
Access Control	Allow/deny permissions	RBAC, ACL, file permissions (chmod)
Firewalls	Filter network traffic	UFW, iptables, hardware firewalls
Intrusion Detection/Prevention (IDS/IPS)	Detect attacks	Snort, OSSEC
Security Policies	Rules that define how systems must be used	Password policy, access rules
Cryptographic Protocols	Secure communication	SSL/TLS, IPSec, Kerberos
Physical Security	Protect hardware	Locks, CCTV, security guards
Backup & Recovery	Maintain availability	Rsync, cloud backup



1. Which of the following best describes *Confidentiality* in the CIA Triad?

- A. Ensuring data is accurate
- B. Ensuring data is available
- C. Ensuring only authorized users can access data
- D. Ensuring faster data processing

2. Which type of attack involves only monitoring or listening to network traffic?

- A. DoS attack
- B. Replay attack
- C. Passive attack
- D. Masquerading attack

3. Which of the following is an example of *Information Security*?

- A. Protecting routers from attacks
- B. Encrypting a student's mark-sheet
- C. Restricting CPU access
- D. Disabling unused network ports

4. A complete set of encryption/decryption processes, keys, and algorithms is called:

- A. Cryptography
- B. Cryptanalysis
- C. Cryptosystem
- D. Ciphertext

5. Which one is a symmetric key algorithm?

- A. RSA
- B. AES
- C. Diffie-Hellman
- D. ElGamal

6. Which service ensures that a sender cannot deny sending a message?

- A. Authentication
- B. Integrity

7. Altering data during transmission is an example of which attack?

- A. Replay attack
- B. Traffic analysis
- C. Data modification attack
- D. Shoulder surfing

8. The primary purpose of *cryptanalysis* is to:

- A. Design encryption algorithms
- B. Break or analyze encryption without the key
- C. Store secret keys
- D. Manage digital certificates

9. Which of the following ensures the accuracy and consistency of data?

- A. Integrity
- B. Confidentiality
- C. Availability
- D. Redundancy

10. Which of the following is a *security mechanism*?

- A. Confidentiality
- B. Hashing
- C. Availability
- D. Non-repudiation

11. Which attack floods a system with traffic to make it unavailable?

- A. MITM attack
- B. DDoS attack
- C. Phishing attack
- D. Social engineering

12. A firewall primarily helps in ensuring:

- A. Confidentiality only
- B. Integrity only

13. The process of converting plaintext into ciphertext is known as:

- A. Cryptanalysis
- B. Decryption
- C. Encryption
- D. Key distribution

14. Which one is an example of *security service*?

- A. Digital signature
- B. Encryption algorithm
- C. Authentication
- D. Firewall installation

15. Which threat involves pretending to be another legitimate user or device?

- A. Eavesdropping
- B. Masquerading attack
- C. Brute force attack
- D. Virus infection

 **Answers**

- 1. C
- 2. C
- 3. B
- 4. C
- 5. B
- 6. C
- 7. C
- 8. B
- 9. A
- 10. B
- 11. B

12. C

13. C

14. C

15. B



HISTORY OF CRYPTOGRAPHY

Cryptography is thousands of years old. Humans have always tried to **protect messages**, especially during **wars, diplomacy, politics, and military strategy**.

The evolution of cryptography can be divided into major time periods:

1 Ancient Civilizations (2000 BCE – 500 CE)

a) Egypt (2000 BCE)

- Oldest known cryptography found in **Egyptian hieroglyphs**.
- Used simple **symbol substitutions**.
- Purpose: **to hide religious or royal messages**.



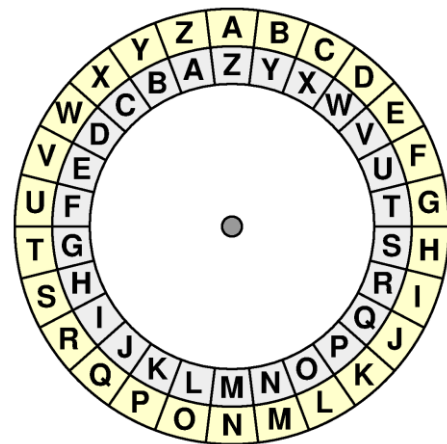
b) Spartan Scytale (600 BCE)

- Used by Spartan warriors.
- A **wooden rod with a strip of leather**.
- Message written across rod → unreadable when unwrapped → readable only on a rod of same size.
- One of the earliest **transposition ciphers**.



c) Ancient Hebrew Atbash Cipher (500 BCE)

- Simple substitution:
 $A \leftrightarrow Z, B \leftrightarrow Y, C \leftrightarrow X$, etc.
- Used in Biblical writings.

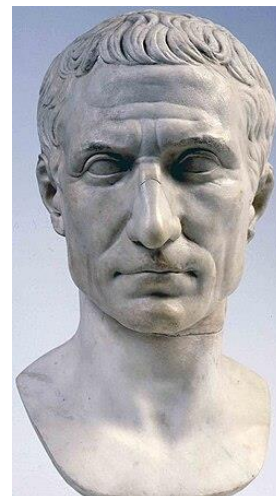


2 Roman Empire (100 BCE – 400 CE)

This is the most famous era in early cryptography, mainly because of **Julius Caesar**.

★ Caesar Cipher (Shift Cipher)

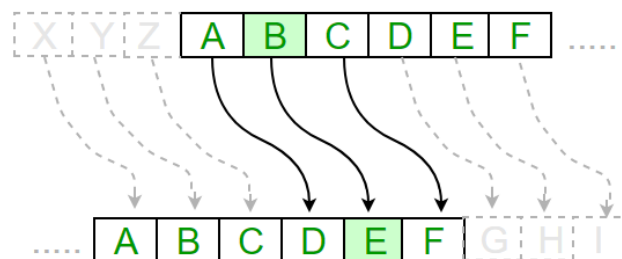
- Introduced by **Julius Caesar**, a Roman general and statesman.
- Used to send **military commands** secretly.
- Each letter shifted by **3 positions**:
 $A \rightarrow D, B \rightarrow E, C \rightarrow F$
- Example:
“ATTACK” → “DWWDFN”



Why it was effective at the time?

- Enemies were mostly illiterate.
- Writing systems were limited.
- Attackers lacked frequency analysis.

But today, it is easily broken by brute force.



3 Medieval Period (500 – 1500 CE)

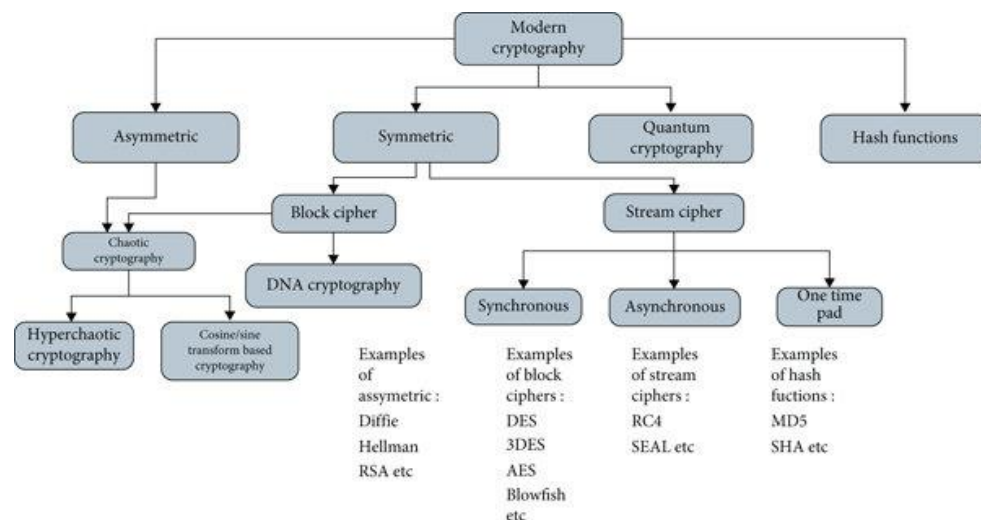


a) Monoalphabetic Substitution (Arab Cryptographers)

- Arabs during the Islamic Golden Age studied **frequency analysis**.
- Scholar **Al-Kindi** (9th century) wrote the **first book on cryptanalysis**.
- This made simple substitution ciphers vulnerable.



4 Renaissance & Early Modern Cryptography (1500 – 1900)



a) Vigenère Cipher (Polyalphabetic Cipher)

- Introduced in 1500s by **Giovan Battista Bellaso**, improved by **Blaise de Vigenère**.
- Used a **keyword** to change alphabets repeatedly.
- Much stronger than Caesar/monoalphabetic ciphers.
- Called the “**unbreakable cipher**” for 300 years.”

b) Playfair Cipher (1854)

- Created by Charles Wheatstone; promoted by Lord Playfair.
- First **digraph substitution cipher**.
- Used in **World War I**.

c) Hill Cipher (1929)

- Developed by Lester S. Hill.
- First cipher based on **linear algebra** and matrix multiplication.

5 World War Era (1900 – 1945)

a) One-Time Pad (OTP)

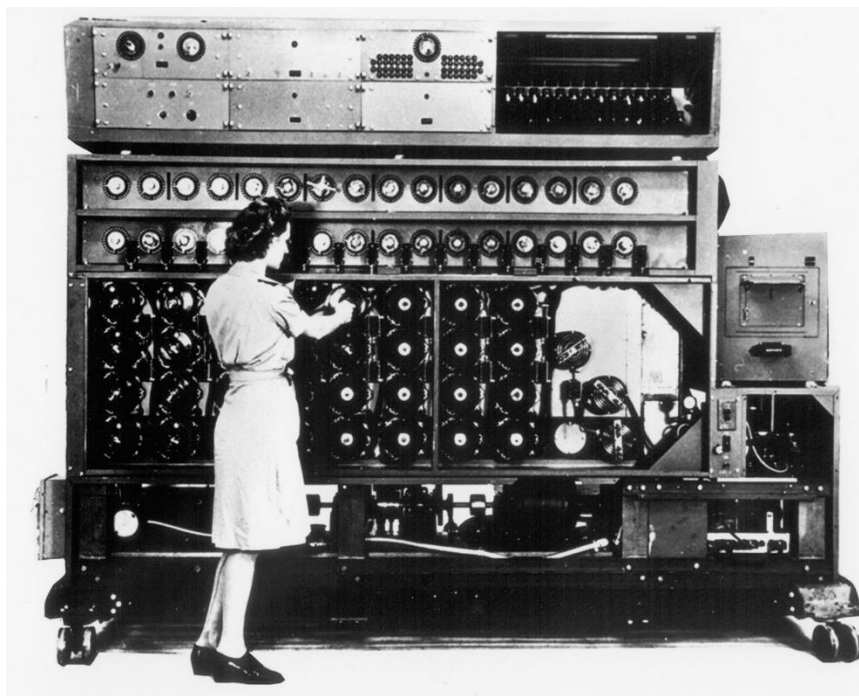
- Invented in 1917.
- If used correctly → **perfect secrecy**.
- Still unbreakable today.

b) Mechanical Ciphers (WWII)

German Enigma Machine

- Most famous machine cipher.
- Used rotating wheels (rotors).
- Broken by **Alan Turing** and team at Bletchley Park.
- Marked the beginning of **modern cryptanalysis**.





Japanese Purple Cipher

- Broken by American cryptanalysts during WWII.

6 Birth of Modern Cryptography (1945 – present)

After WWII, cryptography shifted from mechanical devices to **mathematics and computers**.

a) Claude Shannon (1949)

- Father of modern information theory.
- Defined **confusion, diffusion, entropy**, “perfect secrecy”.
- Inspired modern block ciphers.

b) Symmetric Key Algorithms (1970s–1980s)

- **DES (1977) – Data Encryption Standard**
- Later improved to **Triple DES**
- Eventually replaced by **AES (2001)**

c) Public-Key Cryptography (1976)

- Revolutionized cryptography.

- Invented by **Whitfield Diffie, Martin Hellman**, later formalized by **Rivest–Shamir–Adleman (RSA)**.
- Introduced:
 - Public key
 - Private key
 - Digital signatures
 - Key exchange

d) Internet Age (1990s – present)

Modern cryptography includes:

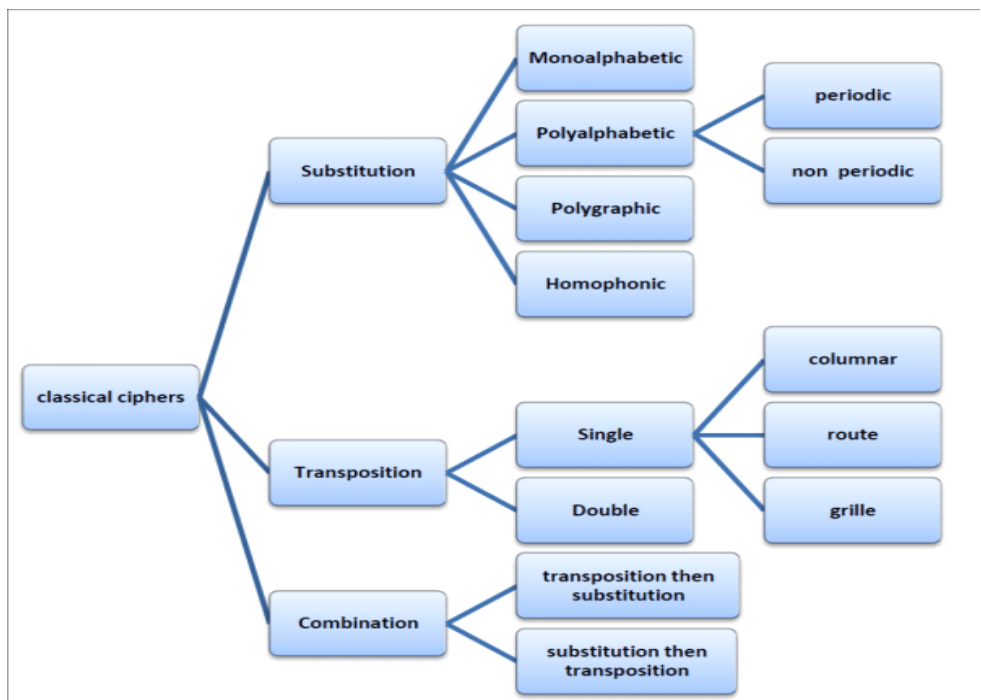
- AES
- RSA
- ECC (Elliptic Curve Cryptography)
- Digital Signatures
- TLS/SSL
- Blockchain cryptography
- Quantum-resistant cryptography

Classical Cryptosystems

Classical cryptosystems are the **oldest methods of encryption**, used before modern computer-based cryptography. They mainly rely on:

- Simple letter substitution
- Rearranging (transposing) characters
- Manual or mechanical techniques

Although simple, they form the **foundation** of modern cryptography.



The diagram divides **Classical Ciphers** into:

1. **Substitution Ciphers**
2. **Transposition Ciphers**
3. **Combination Ciphers**

1 Caesar Cipher (Shift Cipher)

Each letter is shifted by a fixed number (e.g., +3).

$$\text{Formula: } C = (P + k) \bmod 26$$

Let's use $k = 3$ for examples.

✓ Example 1 (Short Word)

Plaintext: **pomegranate**

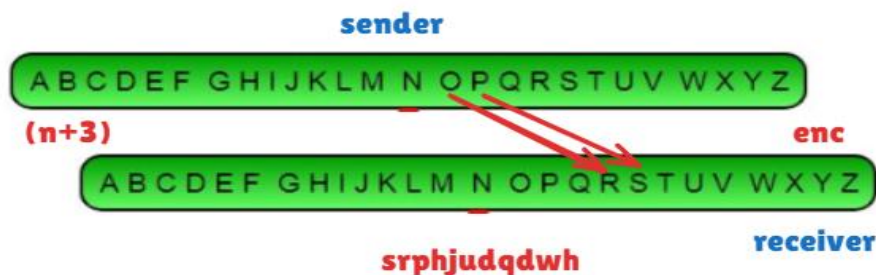
Sender encrypts (shift +3):

p→s, o→r, m→p, e→h, g→j, r→u, a→d, n→q, a→d, t→w, e→h

Ciphertext: **srphjudqdw**

Receiver decrypts (shift -3):

→ **pomegranate**



✓ Example 2 (Long Sentence)

Plaintext:

Lumbini is the birth place of Gautam Buddha

Remove spaces for simplicity:

lumbiniisthebirthplaceofgautambuddha

Shift +3 →

oxpelql lv wkh eluwk sodfh ri jdxwdp exggkd

✓ Example 3

Plaintext: **why security needed**

Ciphertext (+3):

zkb vhfzulwb qhhghg

Decryption (-3):

why security needed

2 Monoalphabetic Substitution Cipher

Each letter is replaced by a unique substitute.

Example mapping (sample):

Numeric Mapping (A-Z → 0-25)

The slide shows the mapping:

This helps us do encryption mathematically.

Encryption & Decryption Formula (Caesar-type Monoalphabetic substitution)**Encrypt** $C = E(a) = (a + k) \bmod 26$ **Decrypt** $a = D(C) = (C - k) \bmod 26$

Where:



- **a** = plaintext letter (0-25)
- **C** = ciphertext letter (0-25)
- **k** = key (shift)

This is a **shift cipher**, which is one type of monoalphabetic cipher.**Monoalphabetic Cipher** – always uses the same letter of the alphabet for the ciphertext letter.

A - 0	H - 7	O - 14	V - 21
B - 1	I - 8	P - 15	W - 22
C - 2	J - 9	Q - 16	X - 23
D - 3	K - 10	R - 17	Y - 24
E - 4	L - 11	S - 18	Z - 25
F - 5	M - 12	T - 19	
G - 6	N - 13	U - 20	

Arithmetic is done as if the alphabet table were circular.

Encrypt: $C = E(a) = (a + k) \bmod 26$ **Decrypt:** $a = D(C) = (C - k) \bmod 26$ **A = 0 H = 7 O = 14 V = 21****B = 1 I = 8 P = 15 W = 22****C = 2 J = 9 Q = 16 X = 23****D = 3 K = 10 R = 17 Y = 24****E = 4 L = 11 S = 18 Z = 25****F = 5 M = 12 T = 19****G = 6 N = 13 U = 20**

<p>Example 1:</p> <p>Encrypt “HELLO” using $k = 3$</p>	<p>Decrypt “KHOOR” using $k = 3$</p> <p>Apply formula: $a = (C - 3) \bmod 26$</p>
<p>H = 7</p> <p>E = 4</p> <p>L = 11</p> <p>L = 11</p> <p>O = 14</p>	<p>K = 10 $\rightarrow 10 - 3 = 7 \rightarrow$ H</p> <p>H = 7 $\rightarrow 7 - 3 = 4 \rightarrow$ E</p> <p>O = 14 $\rightarrow 14 - 3 = 11 \rightarrow$ L</p> <p>O = 14 $\rightarrow 14 - 3 = 11 \rightarrow$ L</p> <p>R = 17 $\rightarrow 17 - 3 = 14 \rightarrow$ O</p>
<p>Step 2: Apply formula:</p> $C = (a + 3) \bmod 26$	<p> Plaintext: HELLO</p>
<p>7 + 3 = 10 \rightarrow K</p> <p>4 + 3 = 7 \rightarrow H</p> <p>11 + 3 = 14 \rightarrow O</p> <p>11 + 3 = 14 \rightarrow O</p> <p>14 + 3 = 17 \rightarrow R</p>	
<p> Ciphertext: KH OOR \rightarrow KHOOR</p>	

Example 2: Monoalphabetic Cipher ($k = 3$) – Word: “pomegranate”

 Caesar-type monoalphabetic: $C = (P + 3) \bmod 26$

Plain Letter	p	o	m	e	g	r	a	n	a	t	e
Plain Value (A=0)	15	14	12	4	6	17	0	13	0	19	4

Plain Letter	p	o	m	e	g	r	a	n	a	t	e
+3 shift	18	17	15	7	9	20	3	16	3	22	7
Cipher Letter	s	r	p	h	j	u	d	q	d	w	h

Sender → Ciphertext: 📌 srphjudqdw h → "srphjudqdw h"

Receiver (Decrypt using $a = C - 3$):

Cipher	s	r	p	h	j	u	d	q	d	w	h
Value	18	17	15	7	9	20	3	16	3	22	7
-3	15	14	12	4	6	17	0	13	0	19	4
Plain	p	o	m	e	g	r	a	n	a	t	e

Receiver → Plaintext: 📌 pomegranate

Example 3:

Plaintext: Lumbini is the birth place of Gautam Buddha ??????????

- Uses a **5×5 key square** (I and J share a cell).
- Encrypts **pairs of letters (digraphs)**.
- Rules for each pair:
 1. **Same row** → replace each letter with the one **to the right** (wrap around).
 2. **Same column** → replace each letter with the one **below** (wrap around).
 3. **Rectangle** → each letter is replaced by the letter **in the same row but in the other letter's column**.

We'll use the key "SECURITY".

Key square (I/J together):

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

Pre-processing for Playfair:

1. Keep only letters, make them **UPPERCASE**.
2. Replace **J** → **I**.
3. Split into **pairs**.
 - If a pair has the **same letter** (e.g., "LL"), make it L X and push the extra L to the next pair.
 - If last letter is single, pad an **X**.

Example 1 – "pomegranate"

◆ Sender side (encryption)

Plaintext: pomegranate

→ uppercase letters only: POMEGRANATE

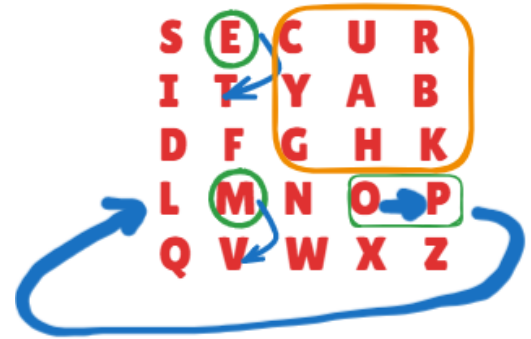
Length is odd (11), so we pad an X at the end.

Pairs after Playfair preprocessing:

PO ME GR AN AT EX

Now encrypt each pair with the square:

- PO → rectangle rule
 - P (row4, col5), O (row4, col4) → same row
 - Move right: P→S (wrap? actually P→S via column 5→1), O→P
 - Result letters for this pair with our key square are LP (from the computed rules).



(We won't detail every pair line by line to keep it short, but the same rules are applied.)

After encrypting all pairs, the sender gets the ciphertext:

👉 Ciphertext: **LP VT KC YO BY UV**

This is what is sent to the receiver.

◆ Receiver side (decryption)

Receiver has: LPVTKCYOBYUV

1. Split into pairs: LP VT KC YO BY UV
2. Apply **inverse rules** (left instead of right in row, up instead of down in column, opposite-corner in rectangle).

After decryption the letter stream becomes:

POMEGRANATEX

The final **X** is only padding (added because of odd length), so the receiver drops it:

👉 Recovered plaintext: POMEGRANATE

Example 2 –“Lumbini is the birth place of Gautam Buddha”	
<p>◆ Sender side (encryption)</p> <p>Plaintext: Lumbini is the birth place of Gautam Buddha</p> <ol style="list-style-type: none"> 1. Remove spaces and make uppercase: <p>LUMBINI ISTHEBIRTHPLACEOFGAUTAMBUDDHA</p>	<p>◆ Receiver side (decryption)</p> <p>Receiver gets: OSPTYLAQDIAFRTBSAFLMYUUMGHHAY BPTSHFKHU</p> <ol style="list-style-type: none"> 1. Split into pairs again:

2. Apply Playfair preprocessing (I/J rule and pair building).
After handling a repeated last I and final A, we get:

**LUMBINIXISTHEBIRTHPLACEOFGAUTA
MBUDDHAX**

3. Form pairs:

LU MB IN IX IS TH EB IR TH PL AC EO FG AU TA
MB UD DH AX

4. Encrypt each pair using the same key square and rules.

After doing this for all pairs, the ciphertext becomes:

 **Ciphertext:**
OSPTYLAQDIAFRTBSAFLMYUUMGHHAYBPTSHF
KHU

(This is the string the sender transmits.)


OS PT YL AQ DI AF RT BS AF I M Y U U M
GH HA YB PT SH FK HU

2. Apply **inverse Playfair rules** on each pair.

The decrypted letter stream is:

LUMBINIXISTHEBIRTHPLACEOFGAUTAM
BUDDHAX

3. Remove padding X where appropriate and reinsert spaces in the natural way:


 **Recovered plaintext:**
Lumbini is the birth place of Gautam Buddha


Hill Cipher (Matrix Cipher)

Processing rules for Hill Cipher

- Convert text to **uppercase**
- Replace **spaces removed**
- Replace **J → I**
- Text should have **even number of letters**, pad with X if needed
- Convert pairs into numeric vectors
- Multiply with key matrix

We will use the standard **2×2 Hill Cipher key matrix**: $K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$

 Encryption: $C = K \times P \bmod 26$

 Decryption: $P = K^{-1} \times C \bmod 26$

where A=0, B=1, ..., Z=25

1 Example 1 : Text: “pomegranate”**◆ SENDER SIDE (ENCRYPTION)****Plaintext** → POMEGRANATE

Length = 11 → add padding → POMEGRANATEX

Step 1: Break into pairs

PO ME GR AN AT EX

Step 2: Convert to numbers (A=0)

P=15, O=14 → (15,14)

M=12, E=4 → (12,4)

G=6, R=17 → (6,17)

A=0, N=13 → (0,13)

A=0, T=19 → (0,19)

E=4, X=23 → (4,23)

Step 3: Multiply each vector by $K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$ Do one example (PO): $\begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 15 \\ 14 \end{bmatrix} = \begin{bmatrix} 87 \\ 100 \end{bmatrix}$

Now mod 26:

 $87 \bmod 26 = 9$ $100 \bmod 26 = 22$

→ (9, 22) → J (9), W (22)

So:

PO → JW

(We repeat same for all pairs.)

Final encrypted output becomes:**👉 Ciphertext: JWQMDUHTFXGD**

(This is what the sender transmits.)

◆ RECEIVER SIDE (DECRYPTION)

Receiver has:

JWQMDUHTFXGD

Step 1: Convert letters → numbers

J=9, W=22... etc.

Step 2: Multiply by inverse matrix K^{-1}

The inverse of K is: $K^{-1} = \begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix} \text{mod } 26$

Now apply: $P = K^{-1} \times C \text{mod } 26$
Each pair will revert to original numeric vectors.

Result becomes:

P O M E G R A N A T E X

Drop padding X → final plaintext:

👉 pomegranate

$$K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$$

$$\det(K) = (3 \times 5) - (3 \times 2) = 9$$

$$K^{-1} \equiv (\det(K))^{-1} \times \text{adj}(K) \pmod{26}$$

$$\text{adj}(K) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \begin{bmatrix} 5 & -3 \\ -2 & 3 \end{bmatrix} \equiv \begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix} \pmod{26}$$

$$K^{-1} \equiv 3 \times \begin{bmatrix} 5 & 23 \\ 24 & 3 \end{bmatrix} \pmod{26} = \begin{bmatrix} 3 \times 5 & 3 \times 23 \\ 3 \times 24 & 3 \times 3 \end{bmatrix} \pmod{26}$$

$$= \begin{bmatrix} 15 & 69 \\ 72 & 9 \end{bmatrix} \pmod{26}$$

$$K^{-1} \equiv \begin{bmatrix} 15 \pmod{26} & 69 \pmod{26} \\ 72 \pmod{26} & 9 \pmod{26} \end{bmatrix} = \begin{bmatrix} 15 & 17 \\ 20 & 9 \end{bmatrix} \pmod{26}$$

2 Example 2 : Text: “Lumbini is the birth place of Gautam Buddha”

◆ SENDER SIDE (ENCRYPTION)

Plaintext → uppercase, spaces removed:

LUMBINIISTHEBIRTHPLACEOFGUATAMBUDDHA

Length = 36 → already even → OK.

LU UM MB BI IN NI II ST HE BI RT HP LA CE OF GA UT AM BU DD HA

Step 2: Convert each pair to numbers

L=11, U=20 → (11,20)
 U=20, M=12 → (20,12)
 M=12, B=1 → (12,1)
 ... continue

Step 3: Multiply by key matrix K

Use: $C = K \times P \bmod 26$

Let's do first pair LU (11,20): $\begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 11 \\ 20 \end{bmatrix} = \begin{bmatrix} 93 \\ 122 \end{bmatrix}$

$93 \bmod 26 = 15 \rightarrow P$
 $122 \bmod 26 = 18 \rightarrow S$

So LU → PS

After computing all pairs, the ciphertext becomes:

👉 **Ciphertext: PSXGNEJFLWOVKUMBUEXEUHCDOWQBYHPUXFSY**

(This is the encrypted message sent.)

◆ RECEIVER SIDE (DECRYPTION)

Receiver gets ciphertext:

PSXGNEJFLWOVKUMBUEXEUHCDOWQBYHPUXFSY

Step 1: Break into pairs: PS XG NE JF LW OV ... etc.

Step 2: Convert letters → numbers: P=15, S=18, etc.

Step 3: Multiply by inverse matrix $P = K^{-1} \times C \bmod 26$

After decrypting all pairs, the numbers revert to original plaintext:

LUMBINIISTHEBIRTHPLACEOFGUATAMBUDDHA

Add spaces naturally:

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

5 Polyalphabetic Cipher (Vigenère)

- Uses a **repeating keyword**.
- Each plaintext letter is shifted by the value of the corresponding keyword letter.
- Formula:

$$\text{Encryption: } C = (P + K) \bmod 26$$

$$\text{Decryption: } P = (C - K) \bmod 26$$

A=0, B=1, C=2 ... Z=25

1 EXAMPLE 1 – “pomegranate”**◆ SENDER SIDE (ENCRYPTION)**

Plaintext: POMEGRANATE

Key: LIME (repeating)

Align key:

P O M E G R A N A T E

L I M E L I M E L I M

Convert letters → numbers:

L=11, I=8, M=12, E=4

Now encrypt each letter:

- $P(15) + L(11) = 26 \rightarrow 0 \rightarrow \mathbf{A}$
- $O(14) + I(8) = 22 \rightarrow \mathbf{W}$
- $M(12) + M(12) = 24 \rightarrow \mathbf{Y}$
- $E(4) + E(4) = 8 \rightarrow \mathbf{I}$
- $G(6) + L(11) = 17 \rightarrow \mathbf{R}$

- $R(17) + I(8) = 25 \rightarrow \mathbf{Z}$
- $A(0) + M(12) = 12 \rightarrow \mathbf{M}$
- $N(13) + E(4) = 17 \rightarrow \mathbf{R}$
- $A(0) + L(11) = 11 \rightarrow \mathbf{L}$
- $T(19) + I(8) = 27 \rightarrow 1 \rightarrow \mathbf{B}$
- $E(4) + M(12) = 16 \rightarrow \mathbf{Q}$

👉 Ciphertext:

AWYIRZMRLBQ

This is what SENDER sends.

◆ RECEIVER SIDE (DECRYPTION)

Receiver has: **AWYIRZMRLBQ**

Align same key:

A W Y I R Z M R L B Q

L I M E L I M E L I M

Now decrypt using: $P = (C - K) \bmod 26$

Example:

$$A(0) - L(11) = -11 \bmod 26 = 15 \rightarrow P$$

$$W(22) - I(8) = 14 \rightarrow O$$

After completing all letters:

👉 Recovered plaintext: **POMEGRANATE**

2 EXAMPLE 2 – “Lumbini is the birth place of Gautam Buddha”

◆ SENDER SIDE (ENCRYPTION)

Plaintext processed: LUMBINI IS THE BIRTH PLACE OF GAUTAM BUDDHA

Remove spaces:

LUMBINI ISTHEBIRTHPLACEOFGAUTAMBUDDHA

Key repeats over entire length:

LIME LIME LIME LIME LIME LIME LIME LIME ...

Now perform encryption letter-by-letter:

- $L(11) + L(11) = 22 \rightarrow \mathbf{W}$
- $U(20) + I(8) = 28 \rightarrow 2 \rightarrow \mathbf{C}$
- $M(12) + M(12) = 24 \rightarrow \mathbf{Y}$
- $B(1) + E(4) = 5 \rightarrow \mathbf{F}$
- $I(8) + L(11) = 19 \rightarrow \mathbf{T}$
- $N(13) + I(8) = 21 \rightarrow \mathbf{V}$
- $I(8) + M(12) = 20 \rightarrow \mathbf{U}$
- ... continue same pattern ...

After encrypting the whole text, ciphertext becomes:

👉 **Ciphertext:** **WCYFTVUQWYBPPZAMVOEQZCQLXMZWCUNPEHQJ**

(This is the transmitted message.)

◆ RECEIVER SIDE (DECRYPTION)

Receiver gets: **WCYFTVUQWYBPPZAMVOEQZCQLXMZWCUNPEHQJ**

They align the same repeating key:

W C Y F T V U Q W Y B P P Z A M V O ...

L I M E L I M E L I M E L I M E L I ...

Apply: $P = (C - K) \bmod 26$

Example:

$$W(22) - L(11) = 11 \rightarrow \mathbf{L}$$

$$C(2) - I(8) = -6 \rightarrow 20 \rightarrow \mathbf{U}$$

$$Y(24) - M(12) = 12 \rightarrow \mathbf{M}$$

$$F(5) - E(4) = 1 \rightarrow \mathbf{B}$$

LUMBINIISTHEBIRTHPLACEOFGAUTAMBUDDHA

Add appropriate spaces:

👉 **Decrypted sentence:** **Lumbini is the birth place of Gautam Buddha****6 One-Time Pad (OTP)**

OTP gives **perfect secrecy**, so the examples show how each letter is combined with a **true random key of equal length**.

Rules:

- Key must be **random, same length** as the plaintext.
- Encryption uses **mod 26 addition**: $C = (P + K) \bmod 26$
- Decryption uses **mod 26 subtraction**: $P = (C - K) \bmod 26$

Mapping:

A=0, B=1, ... Z=25.

IMPORTANT: Because the key is random, ciphertext looks random too.**1 EXAMPLE 1 – “pomegranate”**Plaintext (uppercase): **POMEGRANATE**Length = 11 → use **11-letter random key**.

◆ **Random Key (generated for example):**
XQMDKJBRSLP

Convert both plaintext and key to numbers:

Key “XQMDKJBRSLP” →

X=23, Q=16, M=12, D=3, K=10, J=9, B=1, R=17,
 S=18, L=11, P=15

Letter	P	O	M	E	G	R	A	N	A	T	E
Value	15	14	12	4	6	17	0	13	0	19	4

- $P(15) + X(23) = 38 \rightarrow 12 \rightarrow \mathbf{M}$
- $O(14) + Q(16) = 30 \rightarrow 4 \rightarrow \mathbf{E}$
- $M(12) + M(12) = 24 \rightarrow \mathbf{Y}$
- $E(4) + D(3) = 7 \rightarrow \mathbf{H}$
- $G(6) + K(10) = 16 \rightarrow \mathbf{Q}$
- $R(17) + J(9) = 26 \rightarrow 0 \rightarrow \mathbf{A}$
- $A(0) + B(1) = 1 \rightarrow \mathbf{B}$
- $N(13) + R(17) = 30 \rightarrow 4 \rightarrow \mathbf{E}$
- $A(0) + S(18) = 18 \rightarrow \mathbf{S}$
- $T(19) + L(11) = 30 \rightarrow 4 \rightarrow \mathbf{E}$
- $E(4) + P(15) = 19 \rightarrow \mathbf{T}$

👉 **Ciphertext (OTP): MEYHQABESET**

This appears random — that's normal.

◆ RECEIVER SIDE (DECRYPTION)

Receiver has:

Ciphertext: **MEYHQABESET**

Key: **XQMDKJBRSLP**

Apply: $P = (C - K) \bmod 26$

Decryption reverses the values:

- $M(12) - X(23) = -11 \rightarrow 15 \rightarrow \mathbf{P}$
- $E(4) - Q(16) = -12 \rightarrow 14 \rightarrow \mathbf{O}$
- $Y(24) - M(12) = 12 \rightarrow \mathbf{M}$
... etc.

Final recovered plaintext: 👉 **pomegranate**

Plaintext (uppercase, no spaces): LUMBINIISTHEBIRTHPLACEOFGUATAMBUDDHA

Length = 36 → generate **36-letter random key**.

◆ **Random Key (generated for example):**

QWEXMPLZKTRBAYUDNSHFOGCVMIQEALWUPD

(36 letters)

◆ SENDER-SIDE ENCRYPTION

Encrypt each letter with the corresponding random key letter.

Example (first 6 letters):

- $L(11) + Q(16) = 27 \rightarrow 1 \rightarrow \mathbf{B}$
- $U(20) + W(22) = 42 \rightarrow 16 \rightarrow \mathbf{Q}$
- $M(12) + E(4) = 16 \rightarrow \mathbf{Q}$
- $B(1) + X(23) = 24 \rightarrow \mathbf{Y}$
- $I(8) + M(12) = 20 \rightarrow \mathbf{U}$
- $N(13) + P(15) = 28 \rightarrow 2 \rightarrow \mathbf{C}$

Continue for entire string...

👉 **Final OTP Ciphertext: BQQYUCFDDFHQXWQHGXZJSSKYQMLNWPJZWTL**

This ciphertext is **totally random-looking**, which is expected.

◆ RECEIVER SIDE DECRYPTION

Receiver uses SAME RANDOM KEY:

QWEXMPLZKTRBAYUDNSHFOGCVMIQEALWUPD

Apply: $P = (C - K) \bmod 26$

Each letter subtracts back to the original.

Final plaintext: 👉 **LUMBINIISTHEBIRTHPLACEOFGUATAMBUDDHA**

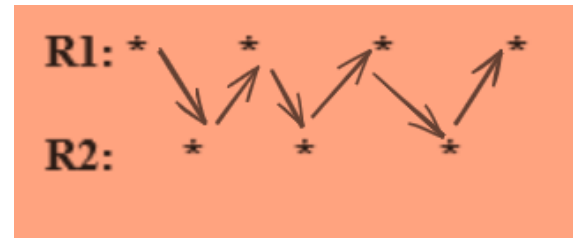
7

Rail Fence writes text in a **zig-zag pattern** across N rails, then reads **row by row**.

Example pattern for **2 rails**:

R1: * * * *

R2: * * *

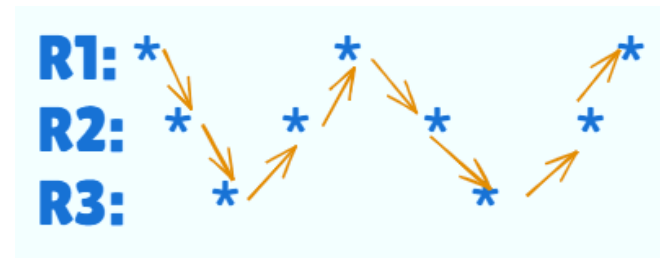


For 3 rails:

R1: * * *

R2: * * *

R3: * *



After writing zig-zag, you **read row-by-row** → ciphertext.

Receiver reverses zig-zag \rightarrow recovers plaintext.

1 EXAMPLE 1 — “pomegranate” with RAILS = 2

Plaintext: **pomegranate**

◆ SENDER SIDE — Encryption (Zig-Zag Writing)

Rails = 2 pattern:

R1: p m g a t

R2: o e r n a e

R1: p m g a a e
R2: o e r n t

Step-by-step zigzag:

- R1: p
- R2: o
- R1: m
- R2: e

- R1: g
- R2: r
- R1: a
- R2: n
- R1: a
- R2: t
- R1: e

R1: p m g a a e
R2: o e r n t

Now read **R1 + R2**:

R1: pmgat

R2: oernate

👉 Ciphertext: pmgatoernate

◆ RECEIVER SIDE — Decryption

Receiver knows rails=2.

1. Count characters → 11
2. Rail1 gets $\text{ceil}(11/2)=6$ chars → "pmgat e"
3. Rail2 gets remaining 5 chars → "oerna te"
4. Reconstruct zig-zag:

R1: p m g a t e

R2: o e r n a

Combine zig-zag:

p o m e g r a n a t e

👉 Recovered plaintext: pomegranate

2 EXAMPLE 2 — “pomegranate” with RAILS = 3

Plaintext: pomegranate

◆ SENDER SIDE — Encryption (Zig-Zag Writing)

Rails = 3 pattern:

R1: p g a

R2: o e r n t

R3: m a e

Let's fill it step-by-step:

Sequence (down → up):

1. R1: p
2. R2: o
3. R3: m
4. R2: e
5. R1: g
6. R2: r
7. R3: a
8. R2: n
9. R1: a
10. R2: t
11. R3: e

So rows become:

R1: p g a

R2: o e r n t

R3: m a e

Read row-by-row: 🖱️ **Ciphertext: pgaoerntmae**

◆ RECEIVER SIDE — Decryption

Receiver knows rails = 3 and ciphertext = pgaoerntmae.

1. Cipher length = 11
2. Rail distribution (wave pattern counts):

For 11 characters, zig-zag index pattern is:

0,1,2,1,0,1,2,1,0,1,2

Count occurrences:

- Rail 1 (index 0): positions 1,5,9 → **3 chars**
- Rail 2 (index 1): positions 2,4,6,8,10 → **5 chars**
- Rail 3 (index 2): positions 3,7,11 → **3 chars**

So split ciphertext:

- R1 = **p g a**
- R2 = **o e r n t**
- R3 = **m a e**

Now rebuild zigzag:

Index pattern again:

0 1 2 1 0 1 2 1 0 1 2

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

R1 R2 R3 R2 R1 R2 R3 R2 R1 R2 R3

Rebuild:

1 → p

2 → o

3 → m

4 → e

5 → g

6 → r

7 → a

8 → n

9 → a

10 → t

11 → e

👉 **Recovered plaintext: pomegranate**

 COMBINED COMPARISON OF ALL CLASSICAL CIPHERS

★ 1. High-Level Comparison Table

Cipher Type	Category	Units Processed	Key Type	Security Level	Notes
Caesar Cipher	Substitution	Single letters	Shift key (0–25)	Very Weak	Only 25 possibilities
Monoalphabetic Cipher	Substitution	Single letters	Single fixed alphabet mapping	Weak	Broken by frequency analysis
Playfair Cipher	Substitution (Digraph)	Pairs of letters	5×5 key matrix	Moderate	Removes single-letter frequency
Hill Cipher	Substitution (Matrix)	Blocks (2+ letters)	Matrix key	Moderate–Strong	Mathematical; vulnerable to known plaintext
Vigenère (Polyalphabetic)	Substitution (Multiple alphabets)	Single letters	Keyword	Medium	Periodic; breakable via Kasiski test
OTP (One-Time Pad)	Substitution (Random)	Single letters	Random key (same length)	Unbreakable	Perfect secrecy
Rail Fence	Transposition	All letters	Number of rails	Weak	Only rearranges letters

★ 2. Comparison Based on Substitution / Transposition

Cipher	Substitution?	Transposition?
Caesar	✓ Yes	✗ No

Cipher	Substitution?	Transposition?
Monoalphabetic	✓ Yes	✗ No
Playfair	✓ Yes (digraph)	✗ No
Hill	✓ Yes (matrix)	✗ No
Vigenère	✓ Yes (multi-shift)	✗ No
OTP	✓ Yes (random)	✗ No
Rail Fence	✗ No	✓ Yes

★ 3. Comparison Based on Frequency Behavior

Cipher	Preserves Letter Frequency?	Hard to Break?
Caesar	✗ No	Very Easy
Monoalphabetic	✗ No	Easy
Playfair	✗ No (digraph freq)	Moderate
Hill	✗ No	Moderate
Vigenère	✗ No (multiple alphabets)	Harder
OTP	✗ No	Impossible to break

Cipher	Preserves Letter Frequency?	Hard to Break?
Rail Fence	✓ Yes	Very Easy

★ 4. Key Characteristics Comparison

Cipher	Key Length	Key Space	Strength
Caesar	1 number	25	Very small
Monoalphabetic	26-letter mapping	26!	Large but breakable
Playfair	Keyword	Large	Good for manual cipher
Hill	Matrix	Determinant $\neq 0 \text{ mod } 26$	Good for block encryption
Vigenère	Keyword	Depends on keyword length	Medium
OTP	Key = message length	Infinite random	Perfect
Rail Fence	Integer (rails)	Small	Very weak

★ 5. Comparison Based on Block Size

Cipher	Operates On
Caesar	1 letter
Monoalphabetic	1 letter

Cipher	Operates On
Playfair	2 letters (digraph)
Hill	2+ letters (block of n)
Vigenère	1 letter (but alphabets change)
OTP	1 letter
Rail Fence	Entire text (position changes)

★ 6. Comparison Based on Modern Applicability

Cipher	Used Today?	Purpose
Caesar	No	Educational / puzzles
Monoalphabetic	No	Basic learning
Playfair	Rare	Military history
Hill	Rare	Teaching matrix encryption
Vigenère	Rare	Teaching polyalphabetic
OTP	Yes (but difficult to implement)	Military, diplomatic communication
Rail Fence	No	Teaching transposition



1. Which classical cipher operates ONLY on rearranging letter positions?

- A. Caesar
- B. Vigenère
- C. Rail Fence
- D. Hill Cipher

2. Which cipher is mathematically unbreakable when used correctly?

- A. Hill Cipher
- B. One-Time Pad
- C. Playfair Cipher
- D. Monoalphabetic Cipher

3. Which cipher encrypts text in blocks rather than single letters?

- A. Caesar
- B. Playfair
- C. Hill Cipher
- D. Rail Fence

4. Vigenère Cipher is classified as which type of cipher?

- A. Polyalphabetic substitution
- B. Transposition cipher
- C. Homophonic cipher
- D. Polygraphic cipher

5. Which cipher preserves the frequency of letters in the plaintext?

- A. Monoalphabetic
- B. Playfair
- C. Hill Cipher
- D. Rail Fence

6. Which cipher substitutes pairs (digraphs) rather than single letters?

- A. Playfair Cipher
- B. OTP
- C. Vigenère
- D. Caesar

7. Which cipher uses a keyword to repeatedly shift letters?

- A. Caesar
- B. Vigenère
- C. Hill
- D. Rail Fence

8. Which cipher has the smallest key space?

- A. Monoalphabetic
- B. Caesar
- C. Vigenère
- D. Playfair

9. Which classical cipher is based on linear algebra and matrix multiplication?

- A. Rail Fence
- B. OTP
- C. Hill Cipher
- D. Caesar

10. Which cipher is MOST vulnerable to frequency analysis?

- A. Caesar
- B. Monoalphabetic
- C. Playfair
- D. Hill

11. Which cipher removes single-letter frequency by encrypting digraphs?

- A. Vigenère
- B. Playfair
- C. Hill
- D. OTP

12. What is the main weakness of One-Time Pad?

- A. Can be brute-forced
- B. Key must be random and as long as message
- C. Same letter always maps the same
- D. Works only on pairs

13. Which cipher uses only a number (shift) as the key?

- A. Rail Fence
- B. Caesar
- C. Hill
- D. OTP

14. Which cipher is considered a polygraphic substitution system?

- A. Caesar
- B. Playfair
- C. Monoalphabetic
- D. Vigenère

15. Which cipher changes alphabets multiple times during encryption?

- A. Monoalphabetic
- B. Caesar
- C. Vigenère
- D. Rail Fence

 **ANSWER KEY**

- 1. C
- 2. B
- 3. C
- 4. A
- 5. D
- 6. A
- 7. B
- 8. B
- 9. C
- 10. B

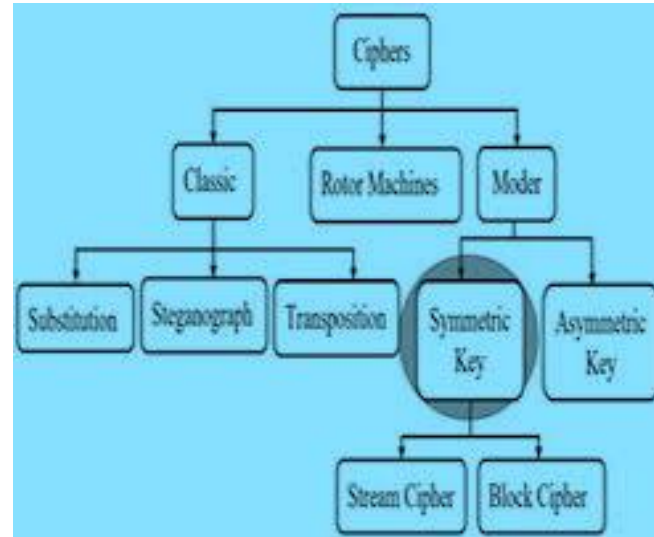
- 11. B
- 12. B
- 13. B
- 14. B
- 15. C

Modern Ciphers: Block vs. Stream Ciphers, Symmetric vs. Asymmetric Ciphers

Modern cryptography uses advanced mathematical algorithms to protect data.

Modern ciphers fall into two major categories:

- 1 Block vs. Stream Ciphers
- 2 Symmetric vs. Asymmetric Ciphers



1 BLOCK CIPHERS

✓ **Definition :** A **block cipher** encrypts **fixed-size blocks** (e.g., 64-bit, 128-bit) at a time. Plaintext is

```

0100011001101111011100100010000001110100011010000110010100100000010000
010110110001101100011010010110000101101110011000110110010100101110
  
```

Examples:

- **DES** – 64-bit blocks
- **AES** – 128-bit blocks
- **Blowfish**
- **IDEA**

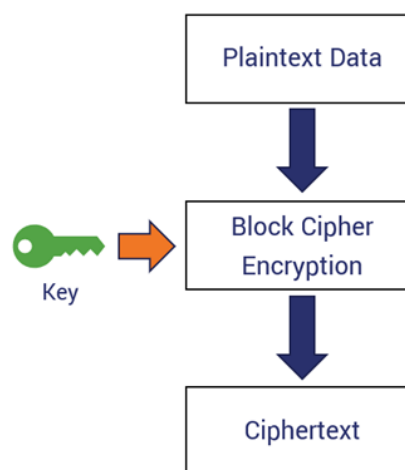
```

010001100110111101110010001000001110100011010000
110010100100000100000101101100011011000110100101
10000101101110011000110110010100101110

```

✓ Characteristics

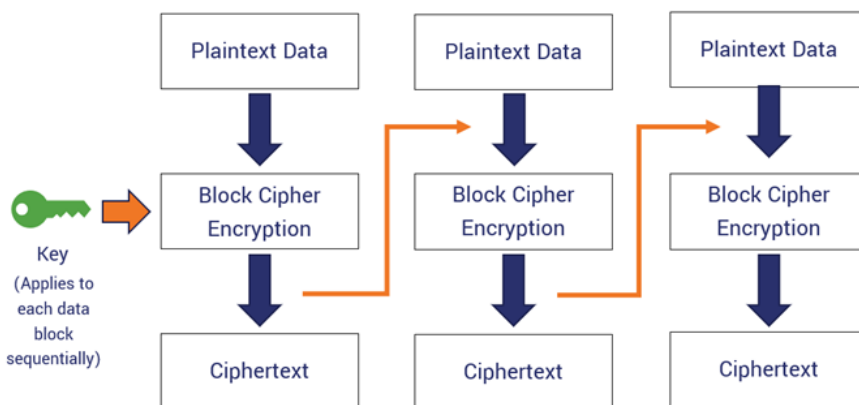
- Operates on blocks (not individual bits)
- Uses substitution-permutation networks
- Requires padding for incomplete blocks
- Works in different **modes**: ECB, CBC, CFB, OFB, CTR

How a Basic Block Cipher Works**✓ Advantages**

- High security
- Can detect error propagation
- Suitable for file and disk encryption

✓ Disadvantages

- Slower for real-time encryption
- Requires padding

How a Basic Block Cipher Chaining Operation Works**2 STREAM CIPHERS**

✓ Definition: A stream cipher encrypts plaintext **one bit or one byte at a time**, using a keystream generated from the key.

```

010001100110111101110010001000001110100011010000
110010100100000100100001101111011100100110010001
10010100100001

```

- RC4
 - A5/1 (GSM)
 - ChaCha20
 - Salsa20
-

✓ Characteristics

- Produces a continuous pseudorandom key stream
 - XOR is typically used for encryption
 - No padding required
 - Excellent for real-time applications (voice, video)
-

✓ Advantages

- Very fast
- Low memory usage
- Ideal for streaming, voice, video

✓ Disadvantages

- Keystream must never repeat
 - Key reuse makes cipher breakable
 - More sensitive to implementation errors
-

★ BLOCK vs. STREAM CIPHER COMPARISON

Feature	Block Cipher	Stream Cipher
Processing	Block-wise (fixed size)	Bit/byte-wise

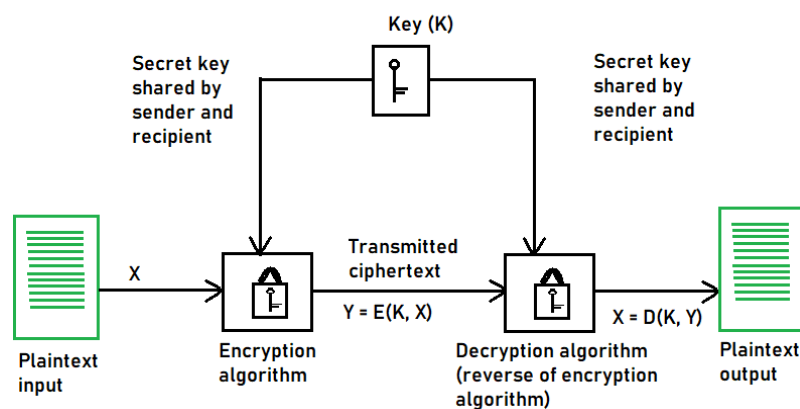
Feature	Block Cipher	Stream Cipher
Padding	Required	Not required
Speed	Slower	Faster
Best Use	File, database, disk encryption	Voice, video, streaming
Error Propagation	May affect a whole block	Usually minimal
Examples	AES, DES	RC4, ChaCha20

3 SYMMETRIC KEY CRYPTOGRAPHY

✓ **Definition:** Both sender and receiver share the same secret key for encryption and decryption.

Examples:

- AES
- DES / 3DES
- Blowfish
- Twofish
- RC4, ChaCha20 (stream)



✓ **Characteristics**

- Fast
- Suitable for large data
- Requires a secure key sharing mechanism

✓ **Advantages**

- Much faster than asymmetric
- Works well for bulk encryption

✓ Disadvantages

- Key distribution problem
- Not suitable for authentication alone

4 ASYMMETRIC KEY CRYPTOGRAPHY

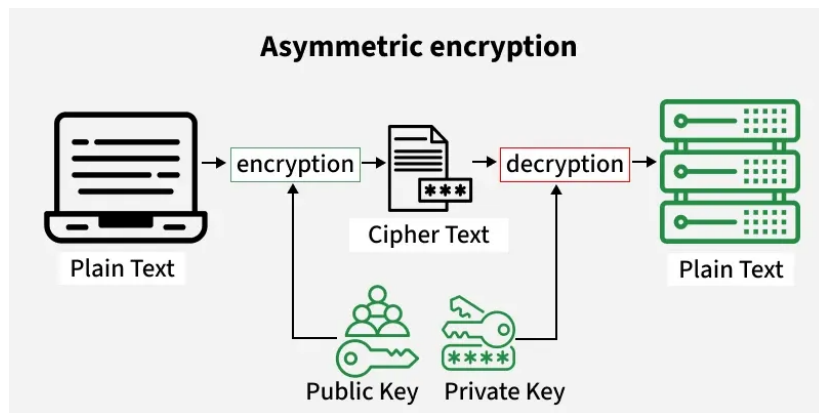
✓ Definition: Uses two different keys:

- **Public Key** (shared with everyone)
- **Private Key** (kept secret)

Encryption and decryption use **different keys**.

Examples:

- **RSA**
- **Diffie–Hellman**
- **ECC (Elliptic Curve Cryptography)**
- **DSA (Digital Signature Algorithm)**



✓ Characteristics

- Slow compared to symmetric
- Used for key exchange, digital signatures, authentication
- No need for secure key distribution

✓ Advantages

- High security
- Solves the key distribution problem
- Enables digital signatures

- Much slower than symmetric
- Not suitable for large data encryption

★ SYMMETRIC vs. ASYMMETRIC COMPARISON

Feature	Symmetric	Asymmetric
Keys Used	One shared key	Public key + private key
Speed	Fast	Slow
Encryption Amount	Large data	Small data (mainly keys)
Key Distribution	Difficult	Easy (public key openly shared)
Security	Depends on key secrecy	Depends on mathematical hardness
Examples	AES, DES, RC4	RSA, ECC, DH

★ NOTE

Modern ciphers can be classified into **block and stream ciphers**. **Block ciphers such as AES and DES** operate on fixed-size blocks of plaintext and use substitution-permutation networks for high security. Stream ciphers like RC4 or ChaCha20 encrypt data one bit or byte at a time using a keystream, making them faster and suitable for real-time applications.

Modern systems also differentiate between symmetric and asymmetric key cryptography. **Symmetric encryption uses the same key** for both encryption and decryption and is very fast but faces **key distribution challenges**. Asymmetric encryption uses **a public-private key pair (e.g., RSA, ECC)**, **solves key distribution**, and is used for **authentication and digital signatures but is slower**.

Both categories work together in modern protocols such as **TLS**.



Scenario Overview

- Juna is on an Ubuntu machine running **Apache2**.
- Juna wants to share a **secret file** via the website, but not in plaintext.
- So Juna **encrypts** the file using **OpenSSL + password**, puts only the **encrypted** file in /var/www/html.
- Muna downloads that **encrypted file** via browser or curl, and then **decrypts** it using the same password.
- We also add a **hash (SHA-256)** to show **Integrity**, and Apache itself ensures **Availability**.

TABLE: Secure File Sharing Scenario (Juna → Muna) Using Apache2 + OpenSSL

1 SYSTEM SETUP

Component	Description
OS	Ubuntu machine
Web Server	Apache2 serving /var/www/html
Sender	Juna (creates + encrypts file)
Receiver	Muna (downloads + decrypts file)
Crypto Tool	OpenSSL (enc, sha256sum)

Component	Description
Encryption Type	AES-256-CBC with password (or key file)

2 SENDER SIDE (Juna) – FILE CREATION

Step	Action	Command / Description
1	Create file	nano secret-report.txt
2	Add contents	“This is a secret report for Muna...”
3	Save file outside /var/www/html	Keeps plaintext private

3 SENDER SIDE (Juna) – ENCRYPTION

Step	Purpose	Command
Generate encrypted file	Confidentiality (AES-256)	openssl enc -aes-256-cbc -salt -pbkdf2 -in secret-report.txt -out secret-report.txt.enc
Enter password	Protects data	Prompts for encryption password
Publish encrypted file	Make file downloadable	sudo cp secret-report.txt.enc /var/www/html/

Step	Description	Command
Generate hash	Ensures file not altered	sha256sum secret-report.txt.enc > secret-report.txt.enc.sha256
Publish hash	For verification	sudo cp secret-report.txt.enc.sha256 /var/www/html/

5 RECEIVER SIDE (Muna) – DOWNLOAD

Step	Description	Command
Download encrypted file	From Apache server	curl http://<server-ip>/secret-report.txt.enc -o secret-report.txt.enc
Download hash	For verification	curl http://<server-ip>/secret-report.txt.enc.sha256 -o secret-report.txt.enc.sha256

6 RECEIVER SIDE (Muna) – VERIFY INTEGRITY

Purpose	Command	Expected Output
Check SHA-256 integrity	sha256sum -c secret-report.txt.enc.sha256	OK (file not modified)

Step	Description	Command
Decrypt using same password	Restores original content	<code>openssl enc -d -aes-256-cbc -salt -pbkdf2 -in secret-report.txt.enc -out secret-report-decrypted.txt</code>
Output	Plaintext retrieved	“This is a secret report for Muna...”

★ 8 OPTIONAL – USING A KEY FILE INSTEAD OF PASSWORD

Step	Action	Command
Create key file	Random 32 bytes	<code>openssl rand 32 > secret.key</code>
Encrypt with key file	More secure	<code>openssl enc -aes-256-cbc -salt -pbkdf2 -in secret-report.txt -out secret-report.txt.enc -pass file:./secret.key</code>
Decrypt with same key	Muna uses same file	<code>openssl enc -d -aes-256-cbc -salt -pbkdf2 -in secret-report.txt.enc -out secret-report-decrypted.txt -pass file:./secret.key</code>

🛡️ 9 CIA TRIAD MAPPED TO THIS SCENARIO

CIA Component	How it is Achieved	Tools Used
Confidentiality	File is encrypted with AES-256 password/key before hosting	<code>openssl enc -aes-256-cbc</code>
Integrity	Juna provides SHA-256 hash; Muna verifies before decrypting	<code>sha256sum</code>
Availability	Apache2 hosts encrypted file 24/7 for download	<code>/var/www/html & apache2</code>

 10 SUMMARY TABLE — WHO DOES WHAT

Actor	Role	Actions Performed
Juna (Sender)	Protects & publishes	Creates file → Encrypts → Generates hash → Uploads to Apache
Apache Server	Makes encrypted data available	Serves encrypted file and hash
Muna (Receiver)	Verifies & decrypts	Downloads → Verifies SHA-256 → Decrypts with password/key

HW

LAB ASSIGNMENT: Secure File Sharing Using Apache2 and OpenSSL

Lab No.: 03

Course: Cryptography / Network Security

OS: Ubuntu Linux

Tools: Apache2, OpenSSL, SHA-256

Roles:

- **Sender:** Juna
 - **Receiver:** Muna
-

1. LAB OBJECTIVES

After completing this lab, the student will be able to:

1. Demonstrate **Confidentiality** through AES-256 encryption using OpenSSL.
 2. Demonstrate **Integrity** using SHA-256 hashing.
 3. Demonstrate **Availability** using Apache2 to host encrypted files.
 4. Perform **file encryption, publishing, downloading, verification, and decryption**.
 5. Understand symmetric key encryption using **passwords** and **key files**.
-

2. LAB TOPOLOGY / SCENARIO

Juna is working on an Ubuntu server with Apache2 installed.
She must securely share a sensitive file with Muna through the web server.

However, the **server should only host encrypted files**, not plaintext.

Muna will download the encrypted file, verify integrity, and decrypt it on her own machine.

This process demonstrates the **CIA Triad**.

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

3. LAB PRE-REQUISITES

Students must:

- Know basic Linux commands
- Understand AES encryption
- Understand hashing
- Have Apache2 installed:

```
sudo apt update
```

```
sudo apt install apache2 openssl -y
```

Verify Apache:

```
systemctl status apache2
```

4. LAB TASKS

TASK 1 — Juna Creates a Secret File

Create a file containing sensitive information:

```
nano ~/secret-report.txt
```

Sample content:

Project: Cryptography Lab

Author: Juna

Receiver: Muna

Content: Highly confidential!

Save the file.

TASK 2 — Juna Encrypts the File Using OpenSSL (AES-256)

Use password-based encryption:

```
openssl enc -aes-256-cbc -salt -pbkdf2 \
```

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

-in ~/secret-report.txt \

-out ~/secret-report.txt.enc

Enter and confirm encryption password.

TASK 3 — Juna Generates SHA-256 Integrity Hash

```
sha256sum ~/secret-report.txt.enc > ~/secret-report.txt.enc.sha256
```

This ensures integrity.

TASK 4 — Juna Publishes Only Encrypted File & Hash

```
sudo cp ~/secret-report.txt.enc /var/www/html/
```

```
sudo cp ~/secret-report.txt.enc.sha256 /var/www/html/
```

Set correct permissions:

```
sudo chmod 644 /var/www/html/secret-report.txt.enc*
```

Files now available at:

<http://<server-ip>/secret-report.txt.enc>

<http://<server-ip>/secret-report.txt.enc.sha256>

TASK 5 — Muna Downloads the Encrypted File

On Muna's system:

```
curl http://<server-ip>/secret-report.txt.enc -o secret.enc
```

```
curl http://<server-ip>/secret-report.txt.enc.sha256 -o secret.enc.sha256
```

TASK 6 — Muna Verifies Integrity

```
sha256sum -c secret.enc.sha256
```

Expected output:

secret.enc: OK

If NOT OK → file modified → do not decrypt.

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

TASK 7 — Muna Decrypts the File

```
openssl enc -d -aes-256-cbc -salt -pbkdf2 \
-in secret.enc \
-out decrypted.txt
```

Enter the same password Juna used.

TASK 8 — Demonstrate CIA Triad in This Lab

Students must explain:

CIA Property	Achieved By	Explanation
Confidentiality	AES-256 encryption	Only Muna with password can decrypt
Integrity	SHA-256 hash	Ensures file not modified during transfer
Availability	Apache2 Web Server	File accessible anytime through HTTP

 **5. LAB QUESTIONS (Write Answers in Report)**

- Q1. Why is it important to encrypt the file before placing it into /var/www/html?**
- Q2. What is the purpose of using the -pbkdf2 option in OpenSSL encryption?**
- Q3. How does SHA-256 ensure integrity? Explain with an example from the lab.**
- Q4. What happens if someone intercepts the encrypted file but does not have the password?**
- Q5. Explain how the CIA Triad is demonstrated in this lab.**
- Q6. Rewrite the encryption and decryption commands and explain each parameter (e.g., -salt, -aes-256-cbc).**
- Q7. What security risk occurs if Juna uses the same password for multiple files?**
- Q8. Repeat the encryption using a key file instead of a password. Compare advantages.**
- Q9. Can Apache2 alone provide confidentiality? Why or why not?**

Sanjeev Thapa, Er. DevOps, SRE, CKA, RHCSA, RHCE, RHCSA-Openstack, MTCNA, MTCTCE, UBSRS, HEv6, Research Evangelist

6. LAB SUBMISSION REQUIREMENTS

Students must submit:

1. **Screenshots** of encryption, hashing, uploading, downloading, verifying, and decrypting.
 2. **A short report** explaining CIA triad in context of this lab.
 3. **Commands used + Outputs.**
 4. **Answers to all lab questions.**
-

7. LAB OUTCOME

By completing this lab, students understand:

- Practical use of OpenSSL
- How symmetric encryption works
- Integrity verification
- Secure file publishing
- CIA Triad implementation
- Real-world cryptography workflow

