Asymmetric Encryption (Public-Key Cryptography)

Definition:

Asymmetric encryption is a type of encryption that uses **two separate keys** —

- **Public Key** for **encryption**, and
- **Private Key** for **decryption**.

These two keys are **mathematically related**, but it's **computationally impossible** to derive one from the other.

How It Works:

1. The sender encrypts the message using the **receiver's public key**.
2. Only the receiver can decrypt it using their **private key**.
3. Even if others know the public key, they cannot decrypt the message.

This ensures **confidentiality** and **secure communication**, especially over insecure networks like the internet.

Example:

Let's say **Alice** wants to send a secret message to **Bob**.

1. Bob generates two keys:
    - Public Key → shared with everyone
    - Private Key → kept secret
2. Alice encrypts her message using **Bob's public key**.
3. Bob receives the encrypted message and decrypts it using **his private key**.

Thus, only Bob can read Alice's message — even if someone intercepts it.

Popular Algorithms:

- **RSA (Rivest–Shamir–Adleman):** Most common asymmetric encryption algorithm used in web security (SSL/TLS).
- **ECC (Elliptic Curve Cryptography):** A modern, faster, and more secure alternative to RSA.
- **DSA (Digital Signature Algorithm):** Used for digital signing and verification.

Real-Life Uses:

- **SSL/TLS Certificates:** Secure website communication (HTTPS).
- **Email Encryption:** Using tools like PGP (Pretty Good Privacy).
- **Digital Signatures:** Verify authenticity of software or documents.
- **Cryptocurrency:** Used in wallet addresses and transactions (e.g., Bitcoin public/private keys).

Advantages:

- Secure key sharing (no need to exchange private keys).
- Provides authentication and non-repudiation.
- Essential for establishing secure channels (e.g., HTTPS).

Disadvantages:

- Slower than symmetric encryption.
- Computationally more expensive.

Simple Example (RSA Concept):

If Bob's public key = **(n, e)** and private key = **(n, d)**:

- Encryption: **C = (M^e) mod n**
- Decryption: **M = (C^d) mod n**

Only the private key **d** can decrypt the ciphertext **C**, even though the public key **e** is known.

—----------

💡 Example for Better Understanding

Symmetric Encryption (Same Key):

- **Key:** 1234
- **Encrypt:** "HELLO" → Encrypted with 1234
- **Decrypt:** Receiver uses the same 1234 to decrypt.
- **Faster and Cost effective**

If the key (1234) is leaked, anyone can decrypt the message.

Asymmetric Encryption (Two Keys):

- **Public Key:** Used to encrypt “HELLO”

- **Private Key:** Only owner can decrypt it.
- **Slower with Costly but more secure**

Even if someone gets the public key, they **cannot decrypt** the message — only the private key can.

—----------

In real-world systems like **HTTPS**, both methods are combined:

1. **Asymmetric encryption** is first used to securely exchange a **temporary symmetric key**.

2. Then, **symmetric encryption** handles the actual data transmission (for speed).

This gives the **security of asymmetric** + **speed of symmetric** encryption.