Interview Questions - Data Protection and secure communication

🔐 1. What is the difference between hashing and encryption?

🔑 Encryption

- **Purpose**: Confidentiality — to protect data by transforming it into unreadable form (ciphertext) that can be **reversed** (decrypted) with a key.
- **Types**:
    - **Symmetric**: Same key for encryption & decryption (e.g., AES)
    - **Asymmetric**: Public key encrypts, private key decrypts (e.g., RSA)
- **Usage**:
    - Securing data in transit (e.g., HTTPS)
    - Encrypting files, databases, backups
- **Reversible**: YES, if you have the key.

🧮 Hashing

- **Purpose**: Integrity and identity — generate a **fixed-size, irreversible digest** of input data.
- **Characteristics**:
    - One-way function — **cannot be reversed**
    - Same input always gives the same output
    - Even a small change in input changes output significantly (avalanche effect)
- **Usage**:

    - Password storage (e.g., bcrypt, SHA-256 with salt)
    - File/data integrity checks (e.g., checksums)
- **Reversible**: NO, by design.

⚖️ Summary Table:

Feature	Encryption	Hashing
Reversible	Yes (with key)	No (one-way)
Use case	Confidentiality	Integrity, identity
Output length	Variable or fixed	Always fixed
Key needed?	Yes	No

⚡ 2. Why is asymmetric encryption slower than symmetric?

🔍 Asymmetric Encryption:

- Uses **two mathematically related keys**: public (encrypt) and private (decrypt).
- Algorithms like **RSA**, **ECC**, **DSA** involve **complex mathematical operations** (e.g., exponentiation over large primes).
- These computations are **CPU-intensive** and require **larger key sizes** (e.g., 2048-bit keys).

🔒 Symmetric Encryption:

- Uses **a single shared key** for both encryption and decryption.
- Algorithms like **AES** are **lightweight**, fast, and can be **hardware-accelerated**.
- Operates on blocks or streams of data with efficient bitwise operations.

⏱️ Performance Comparison:

- **Symmetric**: Suitable for encrypting **large volumes of data**.

- **Asymmetric**: Best for **key exchange, authentication**, not bulk data.

🔁 Real-World Approach:

In TLS/HTTPS, the asymmetric algorithm is used **only to exchange a symmetric key**, and then the session continues using **faster symmetric encryption**.

🏛️ 3. How does PKI build trust online?

🌐 PKI (Public Key Infrastructure):

PKI is a **framework of roles, policies, and technologies** that enables secure **authentication**, **encryption**, and **digital signing** over untrusted networks like the internet.

🔐 How it builds trust:

1. **Digital Certificates**:
    - Issued by trusted **Certificate Authorities (CAs)**
    - Bind a public key to an identity (domain, user, organization)
2. **Chain of Trust**:
    - Browser trusts Root CAs → Root CA signs Intermediate CAs → Intermediate signs your website's certificate
    - If the browser trusts the root, it trusts the chain
3. **TLS/SSL in Action**:
    - When you visit a website using HTTPS:
        - It presents a certificate
        - Your browser checks the validity, expiration, and issuer
        - If verified, a secure connection is established
4. **Revocation Mechanisms**:
    - **CRLs (Certificate Revocation Lists)** or **OCSP (Online Certificate Status Protocol)** are used to check if a cert is compromised.

🛡️ Summary:

PKI enables:

- **Authentication**: You're really talking to `example.com`
- **Confidentiality**: Via encryption
- **Integrity**: Data hasn't been tampered with
- **Non-repudiation**: Signed content cannot be denied

🗄️ 4. What is the concept behind securing Data at Rest and Data in Motion?

💾 Data at Rest

- **Definition**: Data that is stored persistently — on disk, SSDs, databases, file systems, backups.
- **Threats**:
  - Physical theft, disk compromise, unauthorized access
- **Protection Methods**:
  - **Full-disk encryption** (e.g., BitLocker, LUKS)
  - **Database-level encryption** (e.g., TDE – Transparent Data Encryption)
  - **File-level encryption**
  - Access control, IAM roles

🌐 Data in Motion (a.k.a. Data in Transit)

- **Definition**: Data actively moving through the network — between client and server, or across services
- **Threats**:

    - Eavesdropping, MITM attacks, session hijacking

- **Protection Methods**:
    - **TLS/SSL encryption** (HTTPS, secure WebSockets)
    - **VPNs**, **IPSec**
    - Mutual TLS (mTLS) for service-to-service communication
    - Token-based authentication to prevent data tampering

🧠 Key Concept:

- Data is vulnerable in **both states**. A good security posture **encrypts data everywhere** and combines it with strong **access controls**, **auditing**, and **monitoring**.