Digital Certificates, Signatures, and Validation

1. Key Definitions

• **Digital Certificate**: An electronic document that proves the ownership of a public key. It contains information about the domain/server, the public key, the issuer (CA), and is digitally signed by the CA.

• **Digital Signature**: A cryptographic value generated by applying a private key to the hash of some data. It ensures authenticity and integrity.

• **Hashing**: A one-way function that converts input data into a fixed-length string (hash). Small changes in input produce very different hashes.

• **Certificate Authority (CA)**: A trusted third party that issues digital certificates after validating the identity of the requester.

• **Public/Private Key Pair**: Asymmetric keys: the public key is shared, while the private key is kept secret. The private key can sign, and the public key can verify.

• **CSR (Certificate Signing Request)**: A request generated by a server containing its public key and identity details, asking a CA for a certificate.

2. Steps to Create and Issue a Digital Certificate

• Server generates a key pair (public key + private key).

• Server creates a CSR containing the public key and its identity (domain, org details).

• Server sends CSR to the Certificate Authority (CA).

• CA verifies the identity of the server (via domain validation, organization validation, etc.).

• CA creates a digital certificate (X.509 format).

• The certificate contains: Subject info (server/domain), Public key, Issuer info (CA), Validity period, and Signature Algorithm.

• CA signs the certificate by creating a digital signature:

• - CA hashes the TBSCertificate (the body of the certificate without signature).

• - CA encrypts this hash using its private key.

• - The result is the **Signature Value** embedded in the certificate.

• The signed certificate is returned to the server.

3. How a Client Validates the Certificate

• Client connects to the server (e.g., HTTPS).

• Server sends its digital certificate to the client.

• Client checks the issuer (CA) information in the certificate.

• Client checks its trusted root store (pre-installed list of CA certificates in the OS or browser).

• If the CA that issued the certificate is trusted, the client retrieves the CA’s public key from its root store (not from the received certificate).

• Client verifies the signature:

• - Extracts the TBSCertificate (the data to be verified).

• - Hashes this data using the signature algorithm (e.g., SHA-256).

• - Decrypts the signature value in the certificate using the CA’s public key.

• - Compares the decrypted value with the newly computed hash. If they match, the certificate is valid and not tampered with.

• Client also checks certificate validity period (not expired/revoked).

• If all checks pass, the certificate is trusted, and a secure connection is established.

4. Example

Imagine a website **https://securebank.com** wants HTTPS enabled. 1. SecureBank generates a key pair and creates a CSR with its public key and domain name. 2. CSR is sent to DigiCert (a CA). 3. DigiCert validates SecureBank owns the domain and is a real organization. 4. DigiCert creates a digital certificate for securebank.com, containing SecureBank’s public key, domain info, issuer info (DigiCert), validity dates, and a signature created with DigiCert’s private key. 5. SecureBank installs this certificate on its web server. 6. A customer visits https://securebank.com. The browser receives the certificate. 7. Browser finds DigiCert’s public key in its trusted root store and verifies the signature. 8. Verification succeeds → customer can trust that they are securely connected to SecureBank.