Hashing

- irrevesible mathematical function that converts user input into a fixed size output (used for integrity check)
- **Example**
  - SHA256
  - SHA512
  - SH-2
  - SHA256SUM
  - MD5
- **Example on linux**
  - sha256sum flag.txt
  - sha224sum flag.txt

◆ Note That: if the content of the file changed, the hash will change

Encryption

- converting human readable data/message into cipher-text, so only a person holding the right KEY can decrypt the message (used for security purposes, authentication, and authorization)
- **types of Encryption**

1. symmetric encryption

- uses the same key to encrypt and decrypt
- **examples**
  - AES
  - DES
  - 3DES
  - TwoFish
  - BlowFish

2. asymmetric encryption

- uses public key for encryption and private key for decryption
- **Examples**
  - RSA
  - DH
  - DSA
  - ECDH

Encoding

- the purpose of the encoding is to transform data so that it can be properly (and safely) consumed by a different type of a system
- **Example**
  - base64
  - rot13
  - Hex
  - URL

Show the Certificate Authority

- open the browser → edit → advanced → certificates → view certificates

Certificate Authority

A **Certificate Authority** (**CA**) is a trusted organization or entity that issues digital certificates, which are used to verify the identity of a person, device, or organization and to ensure secure communication over a network. The digital certificates issued by a CA are used in a variety of applications, most notably for **SSL/TLS** encryption, which secures websites and protects sensitive data (like credit card information, login credentials, etc.).

Here's how it works in more detail:

1. **Digital Certificates**: A digital certificate contains the public key of the entity it's issued to, along with some identifying information. It's essentially a way to prove that the public key belongs to a particular entity.
2. **Public Key Infrastructure (PKI)**: The CA is a key part of PKI, a framework that uses asymmetric cryptography. In PKI, there are two keys:
    - **Public key**: Distributed widely (e.g., in a certificate).
    - **Private key**: Kept secret by the owner. It's used to decrypt messages encrypted with the corresponding public key.
3. **How It Works**:
    - When you visit a website that uses SSL/TLS, the server presents its **digital certificate** to your browser.
    - The browser checks whether the certificate is valid and if it's signed by a trusted CA (i.e., it checks the digital signature of the certificate).
    - If everything checks out, your browser trusts the server's identity and establishes an encrypted connection.
4. **Trust Model**:
    - CAs are "trusted" because they are verified and vetted by other organizations and governments. The certificates they issue come with the assurance that the entity is who it claims to be.
    - Web browsers and operating systems maintain a list of trusted CAs. If a certificate is signed by one of these trusted CAs, it is automatically trusted.
5. **Root Certificates**: The trust model starts with "root certificates," which are self-signed certificates from trusted root CAs. From there, intermediate CAs can issue certificates that are signed by the root CA, creating a chain of trust.

In essence, CAs act as the gatekeepers of trust on the internet, ensuring that your data and interactions with websites are secure

Intermediate Certificate

An **intermediate certificate** is part of the chain of trust in a public key infrastructure (PKI) system used for securing communications, typically in the context of SSL/TLS certificates for websites. Here's how it fits into the overall picture:

1. **Root Certificate**: At the top of the trust hierarchy, you have a **root certificate**. This is a trusted certificate authority (CA) that signs other certificates. The root certificate is usually pre-installed in browsers and operating systems.
2. **Intermediate Certificate**: These certificates act as intermediaries between the root certificate and the end-entity certificate (like a website's SSL/TLS certificate). They help create a chain of trust. Intermediate certificates are signed by the root certificate and are used to verify the authenticity of the end-entity certificate.
3. **End-Entity Certificate**: This is the certificate used by the website (or service) you are connecting to. It's issued by an intermediate certificate and is what is used to establish an encrypted connection with the client (browser, app, etc.).

Why Intermediate Certificates are Needed:

- **Security**: The root certificate is typically kept offline for security reasons. It doesn't directly sign end-entity certificates. Instead, intermediate certificates sign those certificates on behalf of the root certificate.
- **Flexibility**: Multiple intermediate certificates can be used, and they can be revoked without affecting the root certificate. This gives more flexibility in managing the certificate infrastructure.

When a user visits a website, the browser receives the **end-entity certificate** and tries to build the certificate chain, starting from the website’s certificate and working up to the root certificate. If any intermediate certificates are missing, the chain can be incomplete, and the browser will likely show a warning indicating that the website’s certificate is not trusted.

In practice:

For a website to be fully trusted, the server should send not only its own SSL certificate but also any intermediate certificates needed to form a complete chain of trust, up to the root certificate.

OCSP stands for **Online Certificate Status Protocol**, and it's a protocol used to check the revocation status of digital certificates in real time. OCSP is designed to determine if a particular SSL/TLS certificate (or any digital certificate) is still valid or has been revoked by the certificate authority (CA) that issued it.

Why is OCSP Important?

- **Certificate Revocation**: Sometimes a certificate needs to be revoked before it expires. For example, it could be revoked if the private key is compromised, the organization changes domain names, or the certificate was issued incorrectly. OCSP helps you quickly determine if a certificate is still valid or if it has been revoked, without needing to download a large certificate revocation list (CRL).
- **Real-Time Validation**: It allows for real-time checks of a certificate's status, as opposed to downloading a complete list of revoked certificates, which may not always be up-to-date.

How OCSP Works:

1. **Client Request**: When a client (like a web browser) connects to a server over HTTPS, it can check if the server's certificate is valid by sending an **OCSP request** to the **OCSP responder** (a service operated by the certificate authority or the CA).
2. **OCSP Responder**: The OCSP responder is a server that the CA operates. It receives the request and replies with a certificate status response. This response tells the client whether the certificate is:
    - **Good**: The certificate is valid and has not been revoked.
    - **Revoked**: The certificate has been revoked.
    - **Unknown**: The status of the certificate cannot be determined.
3. **Certificate Validation**: If the client receives a "Good" response from the OCSP server, the certificate is trusted. If it's "Revoked," the client will usually warn the user, and if it's "Unknown," it might treat the certificate as invalid or prompt the user for action.

OCSP Stapling

To improve performance and privacy, a technique called **OCSP Stapling** was introduced. Rather than having the client directly query the OCSP responder, the server can "staple" the OCSP response to its SSL/TLS handshake. This means the server fetches the OCSP response in advance, and then sends it to the client as part of the handshake, which reduces the need for the client to make a separate OCSP request.

- **Benefits of OCSP Stapling**:
    - **Performance**: Reduces the number of external requests (to the CA's OCSP responder) and improves page load times.
    - **Privacy**: The CA doesn't see what websites you visit, as the OCSP request is handled by the server rather than directly by the client.

Why Not Always Use OCSP?

- **Availability**: If the OCSP responder is down or slow, the client might not be able to verify the certificate's status.
- **Privacy**: In the case of direct OCSP checks (without stapling), the CA can track which sites a user visits, which could be a privacy concern.

Overall, OCSP is an important part of maintaining the trustworthiness and security of digital certificates, ensuring that compromised or invalid certificates are quickly flagged.