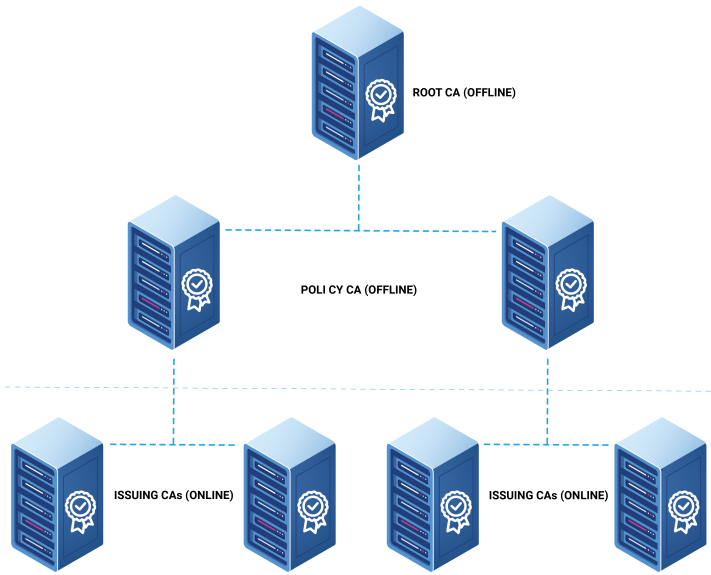
CS553: Crypto In Action Series

Image Source: Dall-E



CS 553

CRYPTOGRAPHY

CIA: Crypto In Action

Public Key Infrastructure

Instructor
Dr. Dhiman Saha

Image Source: https://www.keytos.io/blog/img/what-is-certificate-authority-hierarchy.jpg/

Public key infrastructure

What is PKI?

- System of digital certificates, Certificate Authorities (CAs), and Registration Authorities (RAs) for managing public-key encryption.
- Ensures secure communication and data exchange over the internet.

Key Components of PKI

- Public and Private Keys
- Digital Certificates
- Certificate Authorities (CAs)
  - Root CA: Ultimate trust anchor.
  - Intermediate CAs: Issue and manage certificates under the Root CA.
- Certificate Signing Request (CSR)
- Certificate Revocation List (CRL)

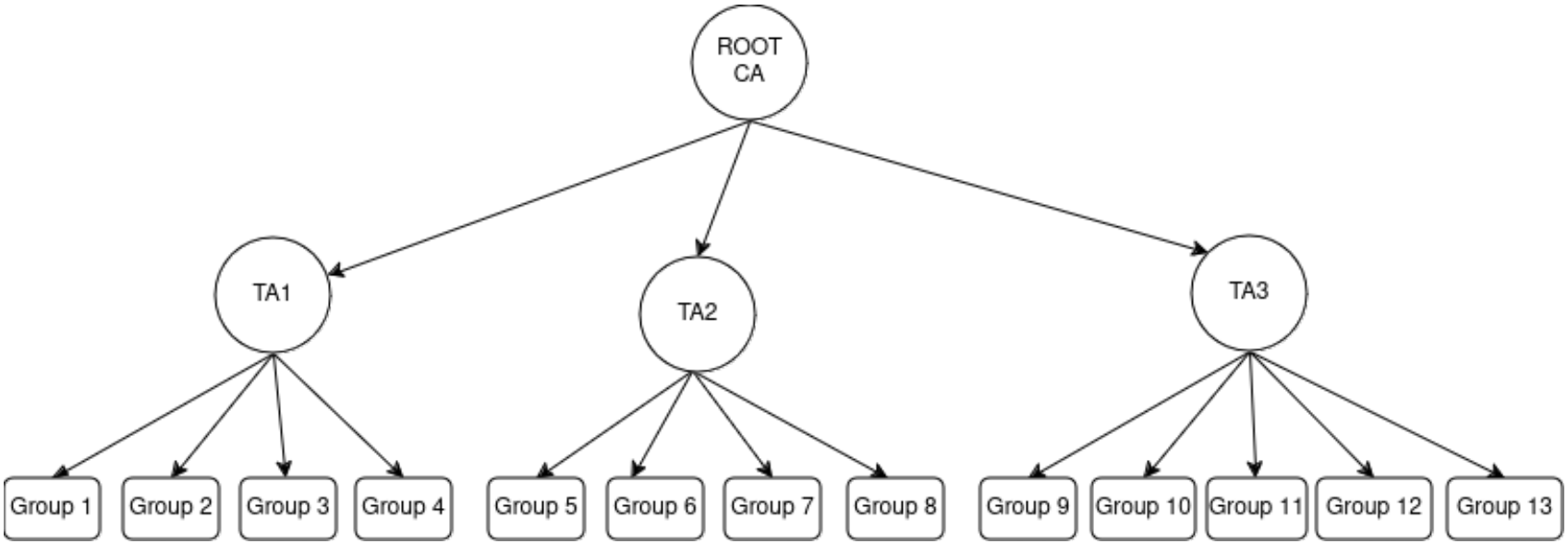
How PKI Works

1. **Key Pair Generation**
   - A user generates a public and private key pair.
2. **Certificate Signing Request (CSR)**
   - The public key, along with identifying information, is submitted to the CA in a CSR.
3. **Certificate Issuance**
   - The CA verifies the identity and issues a digital certificate.
4. **Secure Communication**
   - The certificate is used to establish secure communication channels (e.g., SSL/TLS).
5. **Certificate Validation**
   - The recipient validates the certificate against the CAs public key.

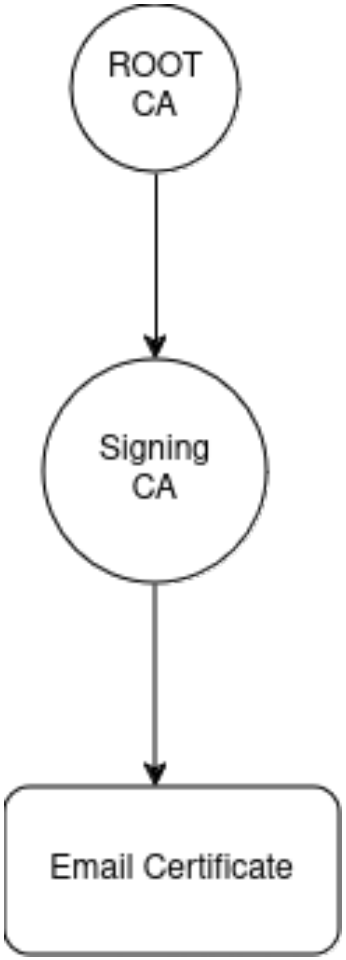
What is OpenSSL?

An open-source toolkit for the SSL/TLS protocols. Provides cryptographic functionality and a robust command-line tool for managing keys, certificates, and more.

Create a custom PKI



Let's see with one example



First create the Simple Root CA and its CA certificate. We then use the Root CA to create the Simple Signing CA. Once the CAs are in place, we issue an email-protection certificate to employee Fred.

Let's see with one example

To get started, fetch the Simple PKI example files and change into the new directory:

```
git clone https://bitbucket.org/stefanholek/pki-example-1
cd pki-example-1
```

General Certificate Options Overview

1. **Root CA Configuration**
   - **Key Pair Generation:** RSA, 2048 bits
   - **Encryption:** Yes (Protect private key)
   - **Hash Algorithm:** SHA-256
   - **Distinguished Name (DN):** Organization, Common Name
   - **Certificate Path:** $dir/ca/$ca.crt
   - **Private Key Path:** $dir/ca/$ca/private/$ca.key
   - **Certificate Extensions:** KeyUsage, BasicConstraints, SubjectKeyIdentifier

General Certificate Options Overview

2. Signing CA Configuration

- **Key Pair Generation:** RSA, 2048 bits
- **Encryption:** Yes (Protect private key)
- **Hash Algorithm:** SHA-256
- **Distinguished Name (DN):** Organization, Common Name
- **Certificate Path:** $dir/ca/$ca.crt
- **Private Key Path:** $dir/ca/$ca/private/$ca.key
- **Certificate Extensions:** KeyUsage, BasicConstraints, SubjectKeyIdentifier
- **Email Address in DN:** Yes (Optional)

3. Email Certificate Request

- **Key Pair Generation:** RSA, 2048 bits
- **Encryption:** Yes (Protect private key)
- **Hash Algorithm:** SHA-256
- **Distinguished Name (DN):** Prompted (Includes domain components, organization, email)
- **Certificate Extensions:** KeyUsage, ExtendedKeyUsage, SubjectAltName

1.1 Create directories

```
mkdir -p ca/root-ca/private ca/root-ca/db crl certs
chmod 700 ca/root-ca/private
```

The ca directory holds CA resources, the crl directory holds CRLs, and the certs directory holds user certificates.

1.2 Create database

```
cp /dev/null ca/root-ca/db/root-ca.db
echo 01 >ca/root-ca/db/root-ca.crt.srl
echo 01 >ca/root-ca/db/root-ca.crl.srl
```

The database files must exist before the openssl ca command can be used.

1.3 Create CA request

```
openssl req -new
   -config etc/root-ca.conf
   -out ca/root-ca.csr
   -keyout ca/root-ca/private/root-ca.key
```

With the openssl req -new command we create a private key and a certificate signing request (CSR) for the Root CA. You will be asked for a passphrase to protect the private key. The openssl req command takes its configuration from the [req] section of the configuration file.

1.4 Create CA request

```
openssl ca -selfsign
   -config etc/root-ca.conf
   -in ca/root-ca.csr
   -out ca/root-ca.crt
   -extensions root_ca_ext
```

With the openssl ca -selfsign command we issue a root CA certificate based on the CSR. The root certificate is self-signed and serves as the starting point for all trust relationships in the PKI. The openssl ca command takes its configuration from the [ca] section of the configuration file.

2.1 Create directories

```
mkdir -p ca/signing-ca/private ca/root-ca/db crl
certs
chmod 700 ca/signing-ca/private
```

The ca directory holds CA resources, the crl directory holds CRLs, and the certs directory holds user certificates. We will use this layout for all CAs in this tutorial.

2.2 Create database

```
cp /dev/null ca/signing-ca/db/signing-ca.db
echo 01 > ca/signing-ca/db/signing-ca.crt.srl
echo 01 > ca/signing-ca/db/signing-ca.crl.srl
```

The database files must exist before the openssl ca command can be used.

2.3 Create CA request

```
openssl req -new
   -config etc/signing-ca.conf
   -out ca/signing-ca.csr
   -keyout ca/signing-ca/private/signing-ca.key
```

With the openssl req -new command we create a private key and a certificate signing request (CSR) for the Signing CA. You will be asked for a passphrase to protect the private key. The openssl req command takes its configuration from the [req] section of the configuration file.

2.4 Create CA certificate

```
openssl ca
   -config etc/root-ca.conf
   -in ca/signing-ca.csr
   -out ca/signing-ca.crt
   -extensions signing_ca_ext
```

With the openssl ca command we issue a certificate based on the CSR. The command takes its configuration from the [ca] section of the configuration file. **Note** that it is the Root CA that issues the Signing CA certificate.

3.1 Create email request

```
openssl req -new
   -config etc/email.conf
   -out certs/fred.csr
   -keyout certs/fred.key
```

With the openssl req -new command we create the private key and CSR for an email-protection certificate. We use a request configuration file specifically designed for the task. When prompted enter these DN components: DC=org, DC=simple, O=Simple Inc, CN=Fred Flintstone, emailAddress=fred@simple.org. Leave other fields empty.

3.2 Create email certificate

```
openssl ca
   -config etc/signing-ca.conf
   -in certs/fred.csr
   -out certs/fred.crt
   -extensions email_ext
```

We use the Signing CA to issue the email-protection certificate. The certificate type is defined by the extensions we attach. A copy of the certificate is saved in the certificate archive ca/signing-ca under a name derived from the certificates serial number (e.g. ca/signing ca/29BD8AB46D221893E2DFA3F3FBACB2B9A17547DF.pem).

Practical Assignment for Students

Step 1: Generate Your Certificate Signing Request (CSR)

```
openssl req -new
    -config student.conf
    -out groupName.csr
    -keyout groupName.key
```

student.conf file will be shared to you.

- Each student will generate a CSR and submit it to the TA to receive their certificate.

Step 2:Verification Process

- The TA will verify your CSR by generating a random 128-bit number.
- This number will be encrypted with the public key from your CSR.
- The encrypted message will be returned to you.

Step 3: Decrypt the Message

- Decrypt the received text using your private key.
- Send the decrypted text back to the TA.
- If the decrypted text matches the TA's original random number, your CSR will be validated, and you will receive your certificate.

How to perform Decryption Process ?

For example you get encrypted file with the name enc_msg.bin you can use below command to get the text back.

```
openssl pkeyutl -decrypt -inkey yourPrivateKey.key
   -in enc_msg.bin -passin pass:yourPassword
```

change yourPrivateKey.key with your private key file name and yourPassword with your password

How to sign and verify digital document using your certificate

Create a Digital Signature: Use your private key to sign the document.

You can use this to create Digital Signature.

```
openssl dgst -sha256 -sign private.key -out
document.sig document.txt
```

How to sign and verify digital document using your certificate

To ensure the signature is valid and the document hasn't been tampered with, you can verify the signature using your certificate:

```
openssl dgst -sha256 -verify <(openssl x509 -in
your.crt -pubkey -noout) -signature document.sig
document.txt
```

References I

https://pki-tutorial.readthedocs.io/en/latest/simple/index.html