PKI Lab

OBJECTIVE

This lab provides a step-by-step guide to generating RSA keys, encrypting a text file, and decrypting it using OpenSSL. Each command is explained in detail for better understanding.

1 Setting Up the Certificate Authority (CA) Environment

Directory Structure Setup

```bash
mkdir ~/CA
mkdir ~/CA/certs
mkdir ~/CA/crl
mkdir ~/CA/newcerts
touch ~/CA/index.txt
echo "1000" > ~/CA/serial
cp /usr/lib/ssl/openssl.cnf ~/CA
cd ~/CA
```

Explanation:

- `**mkdir ~/CA**`: Creates the main CA directory
- `**mkdir ~/CA/{certs,crl,newcerts}**`: Creates subdirectories for:
- `**certs**`: Stores issued certificates
- `**crl**`: Stores certificate revocation lists
- `**newcerts**`: Stores copies of new certificates
- `**touch ~/CA/index.txt**`: Creates an empty database file where OpenSSL will track issued certificates
- `**echo "1000" > ~/CA/serial**`: Initializes the serial number file (starting at 1000) which gives each certificate a unique ID
- `**cp /usr/lib/ssl/openssl.cnf ~/CA**`: Copies the default OpenSSL configuration file to our CA directory for modification

2 Creating the Root CA Certificate

```bash
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 \
  -keyout ca.key -out ca.crt \
  -subj '/CN=icthub Root CA/C=EG/ST=Alexandria/L=SG/O=ICTHUB'
```

Breakdown of options:

- `**req**`: PKCS#10 certificate request and certificate generating utility
- `**-x509**`: Outputs a self-signed certificate instead of a certificate request
- `**-newkey rsa:4096**`: Creates a new RSA key pair (4096-bit)
- `**-sha256**`: Uses SHA-256 as the message digest
- `**-days 3650**`: Sets validity period to 10 years (3650 days)
- `**-keyout ca.key**`: Specifies where to save the private key
- `**-out ca.crt**`: Specifies where to save the certificate
- `**-subj**`: Sets the subject name (replaces interactive prompts)

Key Components Created:

- `ca.key`: Private key (PEM format)
- `ca.crt`: Self-signed certificate (PEM format)

3 Viewing Certificate and Key Contents

```bash
openssl x509 -in ca.crt -text -noout

openssl rsa -in ca.key -text -noout
```

First Command:

- `x509`: Certificate display and signing utility
- `-in ca.crt`: Input file
- `-text`: Displays certificate in human-readable text
- `-noout`: Prevents output of the encoded version

Second Command:

- `rsa`: RSA key processing tool
- `-in ca.key`: Input file
- `-text`: Displays key components in human-readable text
- `-noout`: Prevents output of the encoded version

4 Generating a Certificate Signing Request (CSR)

```bash
openssl req -newkey rsa:2048 -sha256 \
 -keyout icthub.key -out icthub.csr \
 -subj '/CN=www.icthub.edu/O=ICTHUB/C=EG' \
 -passout pass:1234
```

Options Explained:

- `-newkey rsa:2048`: Creates a new 2048-bit RSA key pair
- `-keyout icthub.key`: Saves private key to file
- `-out icthub.csr`: Saves CSR to file
- `-passout pass:1234`: Encrypts private key with password
- `-subj`: Sets subject information (Common Name, Organization, Country)

Output Files:

- ` icthub.key`: Server's private key
- ` icthub.csr`: Certificate Signing Request

5 Signing the CSR to Create a Certificate

```bash
openssl ca -config ~/CA/openssl.cnf -policy policy_anything \
  -md sha256 -days 3650 -in icthub.csr -out icthub.crt \
  -batch -cert ca.crt -keyfile ca.key
```

Detailed Explanation:

- `**ca**`: Certificate Authority management tool
- `**-config ~/CA/openssl.cnf**`: Specifies our custom config file
- `**-policy policy_anything**`: Uses lenient policy (no subject matching)
- `**-md sha256**`: Uses SHA-256 hash algorithm
- `**-days 3650**`: Sets 10-year validity
- `**-in icthub.csr**`: Input CSR file
- `**-out icthub.crt**`: Output certificate file
- `**-batch**`: Runs in batch mode (non-interactive)
- `**-cert ca.crt**`: Specifies CA certificate
- `**-keyfile ca.key**`: Specifies CA private key

What Happens:

1. Verifies the CSR
2. Creates certificate with serial number from ~/CA/serial
3. Increments the serial number
4. Updates index.txt with new certificate info
5. Saves copy in ~/CA/newcerts/

6 Command Flow Summary

- **CA Setup**: Create infrastructure and root certificate
- **Server Setup**: Generate key pair and CSR
- **Certificate Issuance**: CA signs the CSR

Lab Submission

Students must submit:

- Screenshots of key commands & outputs.

Prepared by:

Marwa M. El-Azab
Cybersecurity Research Assistant (MSc Candidate), Teaching Assistant at AASTMT

Omar M. El-Azab
Cybersecurity Instructor, Technical Advisor, Adjunct Teaching Assistant at AASTMT