**MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF THE REPUBLIC OF MOLDOVA**

**Technical University of Moldova Faculty of Computers, Informatics and Microelectronics Department of Software and Automation Engineering**

**Postoronca Dumitru FAF-233**

# Report

Laboratory work n.5

**on CS**

*Checked by:*
**M. Zaica**, *university assistant*
DISA, FCIM, UTM

Chișinău – 2025

# 1 Purpose of the Laboratory Work

The purpose of this laboratory work is to create and implement an internal Public Key Infrastructure (PKI) using the OpenSSL tool. A PKI is a framework that manages digital certificates and public-key encryption, enabling secure communication and authentication in digital systems.

The main objectives of this laboratory work are:

- Generate a root private key and initialize a Certificate Authority (CA)
- Create a self-signed certificate for the CA
- Implement functionality to issue private keys and certificates for users
- Enable users to generate digital signatures for documents
- Implement certificate verification mechanisms
- Develop a certificate revocation system using Certificate Revocation Lists (CRLs)

The implementation must meet the following technical requirements:

- Use the RSA algorithm for generating all private keys
- User private keys must be at least 2048 bits in length with a validity period of 365 days
- The CA private key must be 4096 bits long with a certificate validity of 10 years (3650 days)
- Support document signing and signature verification
- Provide certificate revocation capabilities

This laboratory work demonstrates the fundamental concepts of asymmetric cryptography, certificate management, and trust hierarchies that form the backbone of modern secure communication systems.

# 2 Theoretical Background

## 2.1 Public Key Infrastructure (PKI)

A Public Key Infrastructure is a comprehensive system for creating, managing, distributing, using, storing, and revoking digital certificates. It serves as the foundation for secure digital communication and authentication in modern computing systems. A PKI consists of:

- **Certificate Authority (CA):** A trusted entity that issues and manages digital certificates
- **Registration Authority (RA):** Verifies the identity of entities requesting certificates
- **Certificate Database:** Stores issued certificates and their metadata
- **Certificate Revocation List (CRL):** A list of revoked certificates
- **Digital Certificates:** Electronic documents that bind public keys to identities

## 2.2 RSA Algorithm

RSA (Rivest-Shamir-Adleman) is an asymmetric cryptographic algorithm that uses a pair of keys: a public key for encryption and verification, and a private key for decryption and signing. The security of RSA is based on the computational difficulty of factoring large prime numbers.

Key properties of RSA:

- **Key Generation:** Two large prime numbers are selected and multiplied to create the modulus
- **Public Key:** Can be shared freely and is used for encryption and signature verification
- **Private Key:** Must be kept secret and is used for decryption and signature generation
- **Key Sizes:** Modern standards recommend minimum 2048-bit keys for general use and 4096-bit keys for long-term security

## 2.3 Digital Certificates and X.509 Standard

A digital certificate is an electronic document that uses a digital signature to bind a public key with an identity. The X.509 standard defines the format of public key certificates and is widely used in protocols such as TLS/SSL, S/MIME, and digital signatures.

An X.509 certificate contains:

- **Version:** The X.509 version number
- **Serial Number:** A unique identifier assigned by the CA
- **Signature Algorithm:** The algorithm used to sign the certificate
- **Issuer:** The Distinguished Name (DN) of the certificate issuer
- **Validity Period:** Start and end dates for certificate validity
- **Subject:** The Distinguished Name of the certificate holder
- **Public Key:** The subject's public key and algorithm identifier
- **Extensions:** Additional attributes (optional)
- **Signature:** The CA's digital signature over the certificate data

## 2.4 Certificate Revocation

Certificate revocation is the process of invalidating a certificate before its natural expiration date. Reasons for revocation include:

- Compromise of the private key
- Change in the subject's information
- Cessation of operation
- CA compromise

Certificate Revocation Lists (CRLs) are signed data structures that contain a list of revoked certificates. When verifying a certificate, clients should check the CRL to ensure the certificate has not been revoked.

### 2.5 Digital Signatures

A digital signature is a mathematical scheme for verifying the authenticity and integrity of digital messages or documents. The signing process uses the private key to create a signature, while verification uses the corresponding public key.

The digital signature process involves:

1. **Hashing:** The document is hashed using a cryptographic hash function (e.g., SHA-256)
2. **Signing:** The hash is encrypted using the signer's private key
3. **Verification:** The signature is decrypted using the public key and compared with a freshly computed hash

## 3 Implementation Strategy

### 3.1 Technology Selection

For this laboratory work, I chose to implement the PKI system using a Bash shell script combined with OpenSSL command-line tools. This approach was selected for several reasons:

### 3.2 System Architecture

The implemented PKI system follows a hierarchical trust model with the following components:

1. **Certificate Authority (CA):**
   - Root private key: 4096-bit RSA key stored in `ca.key`
   - Self-signed certificate: 10-year validity stored in `ca.crt`
   - Database directory: `demoCA` containing certificate records
2. **Certificate Database:**
   - `index.txt`: Database of issued certificates
   - `serial`: Counter for certificate serial numbers
   - `crlnumber`: Counter for CRL versions

- `newcerts/`: Directory for issued certificates

3. **User Certificate Management:**
   - Private key generation: 2048-bit RSA keys
   - Certificate Signing Request (CSR) creation
   - CA-signed certificates with 365-day validity
4. **Digital Signature System:**
   - Document signing using SHA-256 hash and RSA
   - Public key extraction from certificates
   - Signature verification mechanism
5. **Revocation System:**
   - Certificate revocation commands
   - CRL generation and management
   - Certificate validation against CRL

## 3.3 Configuration Management

The implementation uses a custom OpenSSL configuration file (`my_config.cnf`) that defines:

- CA directory structure
- Database file locations
- Default certificate validity periods (365 days)
- Default signature algorithm (SHA-256)
- CRL validity period (30 days)
- Certificate issuance policy

## 3.4 Implementation Details

The implementation is organized into five distinct phases, each handling a specific aspect of the PKI system.

### 3.4.1 Phase 0: Environment Setup

Before creating any cryptographic materials, the script initializes the CA database structure:

```
mkdir -p demoCA/newcerts
touch demoCA/index.txt
echo 1000 > demoCA/serial
echo 1000 > demoCA/crlnumber
```

Listing 1: Database initialization

- `demoCA/newcerts`: Directory where issued certificates will be stored
- `index.txt`: Certificate database tracking all issued certificates
- `serial`: Starting serial number for certificates (1000 in hexadecimal)
- `crlnumber`: Starting version number for Certificate Revocation Lists

### 3.4.2 Phase 1: Certificate Authority Creation

The CA is the root of trust in the PKI system. Creating the CA involves two steps:

```
openssl genrsa -out ca.key 4096

# Create self-signed certificate valid for 3650 days
openssl req -new -x509 -key ca.key -out ca.crt \
    -days 3650 -subj "/CN=RootCA" # CN - CommonName
```

Listing 2: CA key and certificate generation

`-subj` argument is used to skip the interactive questioning for things like organisation, country, city, etc.

### 3.4.3 Phase 2: User Key and CSR Generation

For each user (demonstrated with user "John"), the system generates a private key and creates a **Certificate Signing Request**:

```
# generate 2048-bit RSA private key for user
openssl genrsa -out john.key 2048

# create Certificate Signing Request
openssl req -new -key john.key -out john.csr \
    -subj "/CN=John"
```

Listing 3: User key and CSR generation

The CSR contains the user's public key and identity information, which the CA will verify and sign.

### 3.4.4 Phase 3: Certificate Issuance

The CA signs the user's CSR to issue a certificate:

```
openssl ca -config my_config.cnf \
    -in john.csr -out john.crt -batch
```

Listing 4: Certificate signing by CA

- `ca`: Certificate Authority operation
- `-config my_config.cnf`: Use custom configuration
- `-batch`: Non-interactive mode

The configuration file specifies that certificates are valid for 365 days (as in requirements).

### 3.4.5 Phase 4: Digital Signature Operations

The system demonstrates document signing and verification:

```
echo "I agree to the terms." > doc.txt

# sign the document using SHA-256 and private key
openssl dgst -sha256 -sign john.key -out doc.sig doc.txt

# extract public key from certificate
openssl x509 -in john.crt -pubkey -noout > john_pub.pem

# verify signature using public key
openssl dgst -sha256 -verify john_pub.pem \
    -signature doc.sig doc.txt
```

Listing 5: Document signing and verification

The signing process:

1. Computes SHA-256 hash of the document

2. Encrypts the hash with the user's private key
3. Stores the signature in a separate file

Verification process:

1. Extracts the public key from the user's certificate
2. Decrypts the signature using the public key
3. Compares with a freshly computed hash of the document

### 3.4.6 Phase 5: Certificate Revocation

The final phase demonstrates certificate revocation and CRL management:

```
openssl ca -config my_config.cnf -revoke john.crt

# generate Certificate Revocation List
openssl ca -config my_config.cnf -gencrl \
    -out revocation_list.crl

# verify certificate against CRL
openssl verify -crl_check -CAfile ca.crt \
    -CRLfile revocation_list.crl john.crt
```

Listing 6: Certificate revocation

The revocation process:

1. Marks the certificate as revoked in the database
2. Generates a new CRL containing the revoked certificate
3. Verification checks show the certificate is no longer valid

# 4 Execution Example

This section demonstrates the complete execution of the PKI system, showing the output at each phase.

## 4.1 Script Execution Output

```
$ ./labScript.sh
--- Setting up Environment ---
--- Generating CA Key and Certificate ---
--- Generating User Key and CSR ---
--- Signing User Certificate ---
Using configuration from my_config.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'John'
Certificate is to be certified until Nov 22 13:54:52 2026 GMT (365 days)

Write out database with 1 new entries
Database updated
--- Testing Signature ---
Verified OK
--- Revoking Certificate ---
Using configuration from my_config.cnf
Revoking Certificate 1000.
Database updated
Using configuration from my_config.cnf
--- Checking Revocation Status ---
SUCCESS: The certificate is correctly identified as REVOKED.
```

## 4.2 Generated Files Analysis

### 4.2.1 Certificate Authority Files

After execution, the following CA files are created:

- **ca.key**: 4096-bit RSA private key (protected file, 600 permissions)
- **ca.crt**: Self-signed X.509 certificate valid for 10 years

We can inspect the CA certificate:

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

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            19:35:0f:bc:f4:d6:22:f5:d2:39:c0:6f:2f:31:6e:f9:b7:87:a0:2d
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=RootCA
        Validity
            Not Before: Nov 22 13:54:52 2025 GMT
            Not After : Nov 20 13:54:52 2035 GMT
        Subject: CN=RootCA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:a8:c2:3d:...[4096 bits]
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                48:50:3C:75:C1:78:CE:E5:63:DC:04:B2:EF:2F:EA:38:BB:F3:C6:0F
            X509v3 Authority Key Identifier:
                48:50:3C:75:C1:78:CE:E5:63:DC:04:B2:EF:2F:EA:38:BB:F3:C6:0F
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        42:ae:44:af:37:15:6f:.....
```

Key observations:

- Version 3 X.509 certificate
- Self-signed (Issuer = Subject = RootCA)
- 10-year validity period (3650 days)
- SHA-256WithRSAEncryption signature algorithm

- 4096-bit RSA public key

### 4.2.2 User Certificate Files

The user "John" has the following files:

- **john.key**: 2048-bit RSA private key
- **john.csr**: Certificate Signing Request
- **john.crt**: CA-signed certificate valid for 365 days
- **john_pub.pem**: Extracted public key

Inspecting the user certificate:

```
$ openssl x509 -in john.crt -text -noout

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=RootCA
        Validity
            Not Before: Nov 22 13:54:52 2025 GMT
            Not After : Nov 22 13:54:52 2026 GMT
        Subject: CN=John
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d6:12:5d:...[2048 bits]
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                99:05:52:70:54:92:35:A0:23:6D:62:64:B2:65:DC:BB:F7:C5:BC:35
            X509v3 Authority Key Identifier:
                48:50:3C:75:C1:78:CE:E5:63:DC:04:B2:EF:2F:EA:38:BB:F3:C6:0F
    Signature Algorithm: sha256WithRSAEncryption
```

```
    Signature Value:
        8c:75:62:e3:...
```

Key observations:

- Issued by RootCA (establishing trust chain)
- Serial number 1000 (first issued certificate)
- 1-year validity period (365 days)
- 2048-bit RSA public key
- SHA-256WithRSAEncryption signature algorithm

### 4.2.3 Digital Signature Files

The document signing process creates:

- **doc.txt**: Original document ("I agree to the terms.")
- **doc.sig**: Digital signature (256 bytes for 2048-bit RSA)

The signature verification confirms:

```
$ openssl dgst -sha256 -verify john_pub.pem \
    -signature doc.sig doc.txt
Verified OK
```

### 4.2.4 Certificate Revocation List

After revocation, the system generates:

- **revocation_list.crl**: Certificate Revocation List

Inspecting the CRL:

```
$ openssl crl -in revocation_list.crl -text -noout

Certificate Revocation List (CRL):
    Version 1 (0x0)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=RootCA
```

```
    Last Update: Nov 22 13:54:52 2025 GMT
    Next Update: Dec 22 13:54:52 2025 GMT
Revoked Certificates:
    Serial Number: 1000
        Revocation Date: Nov 22 13:54:52 2025 GMT
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        86:a1:b8:6d:5f.......
```

The CRL shows:

- Certificate 1000 (John's certificate) is revoked
- CRL version number matches the crlnumber file
- Next update scheduled for 30 days later
- Signed by RootCA

## 4.3 Verification Results

The script successfully demonstrates all required functionality:

1. **CA Creation:** 4096-bit key and 10-year self-signed certificate
2. **Certificate Issuance:** 2048-bit user key with 365-day validity
3. **Digital Signatures:** Document signing and verification work correctly
4. **Revocation:** Certificate is properly revoked and appears in CRL
5. **Validation:** Revoked certificate fails verification as expected

# 5 Conclusion

This laboratory work provided a comprehensive understanding of Public Key Infrastructure and certificate management systems. By implementing a complete PKI system using OpenSSL, I gained practical experience with several fundamental concepts of modern cryptography and secure communication.

# A Full Source Code

## A.1 Main PKI Script: labScript.sh

```
#!/bin/bash

# --- Phase 0: setup Database ---
echo "--- Setting up Environment ---"
rm -rf demoCA newcerts *.pem *.key *.crt *.csr *.txt *.sig *.crl
mkdir -p demoCA/newcerts
touch demoCA/index.txt
echo 1000 > demoCA/serial
echo 1000 > demoCA/crlnumber

# --- Phase 1: Create CA (4096 bits, 3650 days) ---
echo "--- Generating CA Key and Certificate ---"
openssl genrsa -out ca.key 4096
openssl req -new -x509 -key ca.key -out ca.crt \
    -days 3650 -subj "/CN=RootCA"

# --- Phase 2: Create User John (2048 bits) ---
echo "--- Generating User Key and CSR ---"
openssl genrsa -out john.key 2048
openssl req -new -key john.key -out john.csr -subj "/CN=John"

# --- Phase 3: Issue Certificate (365 days) ---
echo "--- Signing User Certificate ---"
openssl ca -config my_config.cnf -in john.csr \
    -out john.crt -batch

# --- Phase 4: Sign and Verify ---
echo "--- Testing Signature ---"
echo "I agree to the terms." > doc.txt
# Sign
openssl dgst -sha256 -sign john.key -out doc.sig doc.txt
# Extract Public Key
openssl x509 -in john.crt -pubkey -noout > john_pub.pem
# Verify
openssl dgst -sha256 -verify john_pub.pem \
    -signature doc.sig doc.txt

# --- Phase 5: Revocation ---
echo "--- Revoking Certificate ---"
openssl ca -config my_config.cnf -revoke john.crt
openssl ca -config my_config.cnf -gencrl \
```

```
    -out revocation_list.crl

echo "--- Checking Revocation Status ---"
openssl verify -crl_check -CAfile ca.crt \
    -CRLfile revocation_list.crl john.crt > /dev/null 2>&1

if [ $? -ne 0 ]; then
    echo "SUCCESS: Certificate is correctly revoked."
else
    echo "FAILURE: Certificate still valid."
fi
```

Listing 7: Complete Bash script implementing the PKI system.

## A.2 OpenSSL Configuration: my_config.cnf

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir = ./demoCA
database = $dir/index.txt
new_certs_dir = $dir/newcerts
certificate = ./ca.crt
serial = $dir/serial
clrnumber = $dir/crlnumber
private_key = ./ca.key
default_days = 365
default_crl_days = 30
default_md = sha256
policy = policy_anything

[ policy_anything ]
commonName = supplied
```

Listing 8: OpenSSL configuration file for CA operations.

### A.2.1 Policy Section

The `policy_anything` section defines which fields must be present in certificate requests:

- **commonName = supplied**: The Common Name (CN) must be provided by the requester

This minimal policy allows flexible certificate issuance suitable for testing and demonstration purposes. Production systems would typically enforce stricter policies including organizational unit, organization, and country fields.

## A.3 Usage Instructions

To execute the PKI system:

```
# Make the script executable
chmod +x labScript.sh

# Run the script
./labScript.sh
```

To issue additional user certificates, use the Phase 2 and Phase 3 commands with different user names:

```
# Generate key and CSR for a new user
openssl genrsa -out alice.key 2048
openssl req -new -key alice.key -out alice.csr \
    -subj "/CN=Alice"

# Sign the certificate
openssl ca -config my_config.cnf -in alice.csr \
    -out alice.crt -batch
```