Ministry of Education and Research of the Republic of Moldova
Technical University of Moldova
Department of Software and Automation Engineering

# REPORT

Laboratory work No. 5

**Discipline**: Cryptography and Security

Elaborated: Gavriliuc Tudor, FAF - 221

Checked: asist. univ., Nirca Dumitru

Chișinău 2024

Theme: Public Key Infrastructure(PKI) and Digital Signature Algorithm (DSA)

Overview

The PKI (Public Key Infrastructure) project involves the creation of a Root Certificate Authority (CA), generation of user-specific certificates, signing files, and verifying signatures. The program implements key cryptographic functionalities and demonstrates intermediate steps for each operation, ensuring transparency and education on cryptographic processes.

Objective

To simulate the key processes in a PKI system, including:

1. Root CA Creation: Generating a self-signed certificate and private key.
2. User Certificate Generation: Creating user-specific private keys and signing their certificates using the Root CA.
3. File Signing: Using a user's private key to generate a digital signature for a given file.
4. Signature Verification: Using the user's certificate to verify the authenticity of a signed file.

---

Theoretical Notes

1. Key and Certificate Generation

- Root Certificate Authority (CA):
  - Purpose: To act as the trusted entity that signs user certificates.
  - Process:
    1. Generate a private key using RSA (4096 bits).
    2. Create a self-signed certificate using a defined subject and issuer.
    3. Store the private key and certificate in predefined directories.
  - Output:
    - rootCA.key: Private key of the Root CA.
    - rootCA.pem: Public certificate of the Root CA.
- User Certificates:
  - Purpose: To assign unique credentials to users and enable cryptographic operations.
  - Process:
    1. Generate a user-specific RSA private key (2048 bits).
    2. Create a certificate request.
    3. Sign the certificate request using the Root CA's private key.

4. Store the user's private key and signed certificate.
    - Output:
        - <username>.key: User's private key.
        - <username>.crt: User's signed certificate.

2. Digital Signature

- File Signing:
    - Purpose: To ensure the integrity and authenticity of a file.
    - Mechanism:
        1. Read the file to be signed.
        2. Use the user’s private key to create a signature using the RSA-PSS scheme with SHA-256.
        3. Store the signature in a separate file.
    - Output:
        - <file_path>.sig: Signature file corresponding to the signed file.
- Signature Verification:
    - Purpose: To validate the file’s integrity and ensure it was signed by the expected entity.
    - Mechanism:
        1. Use the user’s public key from their certificate to verify the signature.
        2. Compare the computed hash with the signed hash to determine authenticity.
    - Output: Verification result (success or failure).

---

Key Functionalities and Their Purpose

1. generate_ca()

- Purpose: Generates a Root CA private key and self-signed certificate.
- Inputs:
    - None (uses predefined subject attributes for the Root CA).
- Outputs:
    - rootCA.key: CA private key.
    - rootCA.pem: CA self-signed certificate.
- Mechanism:
    - RSA key generation, certificate creation with Basic Constraints, and signing using the CA's private key.

2. generate_user_certificate(username)

- Purpose: Generates a private key and signed certificate for a specific user.
- Inputs:
  - username: The name of the user for whom the certificate is being generated.
- Outputs:
  - <username>.key: User's private key.
  - <username>.crt: User's signed certificate.
- Mechanism:
  - RSA key generation, certificate creation, signing with the Root CA's private key, and Basic Constraints indicating it’s not a CA certificate.

3. sign_file(username, file_path)

- Purpose: Creates a digital signature for a file using the user's private key.
- Inputs:
  - username: Name of the user whose private key will be used for signing.
  - file_path: Path of the file to be signed.
- Outputs:
  - <file_path>.sig: Signature file.
- Mechanism:
  - File reading, RSA-PSS signing with SHA-256, and writing the signature to a file.

4. verify_signature(username, file_path, signature_path)

- Purpose: Verifies the signature of a signed file using the user’s public key.
- Inputs:
  - username: Name of the user whose certificate will be used for verification.
  - file_path: Path of the file to be verified.
  - signature_path: Path of the signature file.
- Outputs:
  - Verification result (success or failure).
- Mechanism:
  - File and signature reading, public key extraction from the user’s certificate, RSA-PSS verification with SHA-256.

## Results:

```
Choose an option:
1. Generate CA (Root Certificate Authority)
2. Generate User Certificate
3. Sign a File
4. Verify File Signature
5. Exit
Enter your choice: 3
Enter the username for signing: tudor
Enter the file path to sign: C:\Users\tudor\PycharmProjects\CS\Lab_5\document_to_sign.txt
Signing file: C:\Users\tudor\PycharmProjects\CS\Lab_5\document_to_sign.txt...
File signed: C:\Users\tudor\PycharmProjects\CS\Lab_5\document_to_sign.txt.sig

Choose an option:
1. Generate CA (Root Certificate Authority)
2. Generate User Certificate
3. Sign a File
4. Verify File Signature
5. Exit
Enter your choice: 4
Enter the username for verification: tudor
Enter the file path to verify: C:\Users\tudor\PycharmProjects\CS\Lab_5\document_to_sign.txt
Enter the signature file path: C:\Users\tudor\PycharmProjects\CS\Lab_5\document_to_sign.txt.sig
Verifying signature for: C:\Users\tudor\PycharmProjects\CS\Lab_5\document_to_sign.txt...
Signature verified successfully.

Choose an option:
1. Generate CA (Root Certificate Authority)
2. Generate User Certificate
3. Sign a File
4. Verify File Signature
5. Exit
Enter your choice: 5
Exiting the program.

Process finished with exit code 0
```

## Conclusion

The PKI project demonstrates the practical implementation of cryptographic principles essential for securing digital communications. By simulating the creation of a Root Certificate Authority, generating user-specific certificates, signing files, and verifying digital signatures, the project showcases the core functionalities of Public Key Infrastructure.

The step-by-step approach highlights critical processes such as key generation, key scheduling, and digital signature management, ensuring clarity and educational value. Furthermore, displaying all intermediate values during these operations enhances transparency and fosters a deeper understanding of the cryptographic mechanisms involved.