Technische Hochschule Rosenheim
Fakultät für Informatik
Prof. Dr. Reiner Hüttl

IT-Security

Exercise 4

In this exercise we write a Java program to sign a file.

Task 1: Create an asymmetric key pair and a certificate

To create signatures, we need a private key for the signature and
a public key, packed in a certificate, for verification. We use a widely used tool called **openssl**.
Perform the following steps with the tool:

1. Generate a certificate and private key with the command:
   ```
   openssl req -x509 -sha512 -days 365 -newkey rsa:4096 -keyout
   private.key -out certificate.crt
   ```

   Complete the requested information according to the following pattern:
   ```
   Country Name: DE
   State or Province Name: Bayern
   Locality Name: Rosenheim
   Organization Name: TH Rosenheim
   Organizational Unit Name: Informatik
   Common Name: Max Musterstudent
   ```

1. Convert the certificate to a format understandable for Java:
   ```
   openssl x509 -in certificate.crt -outform PEM -out
   certificate.pem
   ```

2. Convert the private key to a format understandable for Java:
   ```
   openssl pkcs8 -in private.key -topk8 -nocrypt -out private.pkcs8
   ```

Note:

- We have now created a self-certified key pair. In practice, a certificate request is generated from the key pair and sent to a trust center. There, a generally accepted and verifiable certificate for the key is generated.
  https://www.ssl.com/how-to/manually-generate-a-certificate-signing-request-csr-using-openssl/

Task 2: Signature of a file

Write a Java class DigitalSignature that signs a file.
For the signature, the methods **sign** and **verify** of the class **DigitalSignature** must be implemented.

Hints:

- Look at the **Signature** class from the **JDK**: https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/security/Signature.html
- Use the algorithm `SHA512withRSA`.

Task 3: Test driver in JUnit for the class Signature

A test driver is already given.

- Take a look at the individual actions in the test driver.
- Run the test
- Change (falsify) the data between the time of signature and verification and check whether the manipulation is noticed. (Start test in debug mode and set breakpoint before verification)

Hints:

- There is a class CryptoUtil that must be included in the project. It contains a method for reading a private signing key (**getPrivateKey**) and a method for reading the Base64 encoded certificate (**getCertificate**).

- Copy the files from task 1 (keystore, certificate) to the project directory (workspace)