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

IT-Security

Übung 4

In dieser Übung schreiben wir ein Java Programm zur Signatur einer Datei.

Aufgabe 1: Erzeugung eines asymmetrischen Schlüsselpaars und eines Zertifikats

Zur Erstellung von Signaturen benötigen wir einen privaten Schlüssel für die Signatur und eine öffentlichen Schlüssel, verpackt in ein Zertifikat, zur Verifikation.
Wir verwenden dazu ein Tool im JavaSDK mit dem Namen **keytool**. Führen sie folgende Schritte mit dem Tool durch:

- Erzeugen sie ein Schlüsselpaar mit dem Befehl:
  keytool -genkey -dname "***" -alias *** -keypass *** -keystore mykeystore -storepass *** -keyalg RSA -sigalg SHA512withRSA -keysize 4096

  Ersetzen sie die mit *** gekennzeichneten Stellen mit sinnvollen Werten für den Alias des Schlüssels, Passwörtern für den Schlüssel und den KeyStore.

  Für dname geben sie **einen X.500 Distinguished Name** nach dem Muster "cn=Hans Wurst, ou=Informatik, o=FH Rosenheim, c=DE" ein.

  Der keystore wird in der Datei mykeystore gespeichert,
  z.B. C:\WS\ITSecurity\Uebung2\Aufgabe1\mykeystore

- Speichern sie das Zertifikat und den öffentlichen Schlüssel in einer Datei:
  keytool -export -alias *** -keypass *** -keystore mykeystore -storepass *** -file test.cer
  Das Zertifikat wird in der Datei test.cer gespeichert.
  z.B. C:\WS\ITSecurity\Uebung2\Aufgabe1\test.cer

- Öffnen sie die Datei test.cer mit einem Doppelklick in Windows und speichern sie sie Base64 kodiert unter dem Namen Base64.cer wieder ab.
  Auf MAC exportieren sie das Zertifikat in das Format PEM

- Kopieren sie die Dateien mykeystore und Base64.cer in ihr Java Projektverzeichnis

Anmerkung:

- Wir haben jetzt ein selbst zertifiziertes Schlüsselpaar erzeugt. In der Praxis wird aus dem Schlüsselpaar ein Zertifikats-Request erzeugt und an ein Trust Center geschickt. Dort wird ein allgemein anerkanntes und verifizierbares Zertifikat für den Schlüssel erzeugt.

Aufgabe 2: Signatur einer Datei

Schreiben sie eine Java Klasse **DigitalSignatur** die eine Datei signiert.
Zur Signatur sind dabei die Methoden

```
// Sign the content of a file
public static byte[] sign(String filename, PrivateKey signatureKey)
// store signatur as Byte-Array in a file
public static void saveSignature(String sigFile, byte[] signedData)
```

zu implementieren.

Aufgabe 3: Verifikation der Signatur

Ergänzen sie in der Klasse aus Aufgabe 2 die Methode

```
// verify a signature
public static boolean verify(String filename, String sigFilename,
Certificate cert)
```

Aufgabe 4: Testtreiber in JUnit für die Klasse Signatur

Ein Testtreiber ist bereits gegeben.

- Ändern sie in dem Testtreiber die Werte für die Passwörter und den Alias.
- Schauen sie sich die einzelnen Aktionen in dem Testtreiber an.
- Führen sie den Test durch
- Ändern (fälschen) sie die Daten zwischen dem Zeitpunkt der Signatur und der Verifikation und überprüfen sie ob die Manipulation bemerkt wird.

Hinweise:

- Es ist eine Klasse CryptoUtil gegeben die in das Projekt eingebunden werden muss. Sie enthält eine Methode zum Einlesen eines privaten Signaturschlüssel aus einem Keystore (**getPrivateKey()**) und eine Methode zum Einlesen des Base64 kodierten Zertifikats **(getCertificate()**).

- Folgende Klassen und Interfaces sind notwendig:
  java.security. Signature,
  java.security.Key, java.security.PrivateKey,
  java.security.cert.CertificateFactory, java.security.cert.Certificate,
  java.security.KeyStore
  (s. http://docs.oracle.com/javase/6/docs/api/java/security/package-summary.html)

- Kopieren sie die Dateien aus Aufgabe 1 (Keystore, Zertifikat) in das Projekt-Verzeichnis (Workspace)