Labo Distributed Systems 5: Cryptografische primitieven, sleutel- en certificaatbeheer en SSL

Inleiding

In dit labo maakt de lezer kennis met de cryptografische mogelijkheden aangeboden door Java.
In eerste instantie wordt bekeken hoe de verschillende cryptografische primitieven kunnen aangewend worden in Java. Vervolgens gaan we dieper in op het opslaan en beheren van sleutels en certificaten.

DEEL 1: Cryptografische primitieven

1.1 Literatuurstudie

Voor dit labo neem je het volgende document in meer detail door. Concentreer vooral op de codevoorbeelden. Deze kunnen helpen bij het vervolg van het labo

- De Java Cryptography Architecture (JCA) Reference Guide binnen general security: http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html

Voor meer info over security in Java kan je de *Programmers guide, General Security API specification* en *Security Tools* raadplegen op:

- Guides: http://docs.oracle.com/javase/7/docs/technotes/guides/security/index.html
- Tools: http://docs.oracle.com/javase/7/docs/technotes/tools/index.html#security

1.2 Opdracht

Voorbereiding

- Download de serialiseerbare klasse *Person* via Toledo.
- Maak 3 objecten aan van deze klasse: *person1*, *person2* en *person1Fake*. *person1* en *person1Fake* met dezelfde inhoud, met uitzondering van 1 karakter en *person2* met andere inhoud.
- Gebruik de serialisatie methoden beschikbaar in de *Person* klasse om de drie objecten om te zetten naar een byte array.

Hashing / Message Digest

- Maak van *person1* en *person1Fake* een SHA-256 hash.
- Print de hashes van de byte array als een String object op het scherm en vergelijk ze. Wat merk je op?

Symmetrische Encryptie

- Wat zijn de mogelijke sleutellengtes voor het AES en het tripleDES algoritme?
- Creëer een symmetrische sleutel voor AES
- Encrypteer een korte tekst met behulp van deze sleutel
- Decrypteer de geëncrypteerde tekst en vergelijk.

Asymmetrische Encryptie

- Wat zijn de mogelijke sleutellengtes voor het RSA en het DSA algoritme? Vergelijk met de symmetrische sleutels.
- Creëer een asymmetrische sleutel. Steek de publieke en private sleutel in een aparte variabele.
- Print de publieke sleutel in een gestandaardiseerde vorm op het scherm.
- Encrypteer en decrypteer dezelfde tekst als in de vorige opgave met het asymmetrische sleutelpaar. Vergelijk de uitvoeringstijden van beide encryptiemethodes.

Digitale Handtekeningen

- Neem de byte-array van *person2* en plaats er een digitale handtekening op.
- Verifieer de digitale handtekening

DEEL 2: Certificaat- en sleutelbeheer

2.1 Literatuurstudie

Ga naar de website van portecle: http://portecle.sourceforge.net/. Portecle is een tool die je toelaat op een eenvoudige manier sleutels en certificaten te beheren. Bestudeer de mogelijkheden van portecle en lees de helppagina's. Download en installeer het pakket.

2.2 Opdracht

De mogelijkheden van portecle

- Maak twee nieuwe JKS keystores (store1.jks en store2.jks). Beveilig store1.jks met een paswoord. Veronderstel dat Werner eigenaar is van store1.jks. Freya is eigenaar van store2.jks.
- Genereer een RSA sleutelpaar in store1.jks. Beveilig dit met een paswoord. Geef werner als alias op. Die alias zal gebruikt worden om sleutelmateriaal vanuit de Java

applicatie op te halen. Maak vervolgens een RSA sleutelpaar in store2.jks (geen paswoordbeveiliging). Geef als alias freya op. Merk op dat portecle automatisch een selfsigned certificate aanmaakt.

- Bekijk de certificaten met een certificate viewer.
- Exporteer de certificaten die aangemaakt zijn uit store1.jks en store2.jks. Importeer de certificaten vervolgens in de andere keystore. Daardoor zit het certificaat van Werner in de keystore van Freya en vice versa.
- Bekijk de informatie die in elk van de keystores aanwezig is. Je kan onder andere een report opvragen van alle gegevens in een keystore. Je kan ook bepaalde informatie opvragen in gestandaardiseerde formaten.

Aanspreken van keystore in Java

- Eerst moet een keystore ingeladen worden in de java applicatie. Bij het inladen worden twee parameters meegegeven: een inputstream en een paswoord. Een codevoorbeeld vind je hier:

  ```
  keyStore = KeyStore.getInstance("JKS");
  String fileName = directorynaam + bestandsnaam + ".jks";
  FileInputStream fis = new FileInputStream(fileName);
  keyStore.load(fis,password);
  fis.close();
  ```

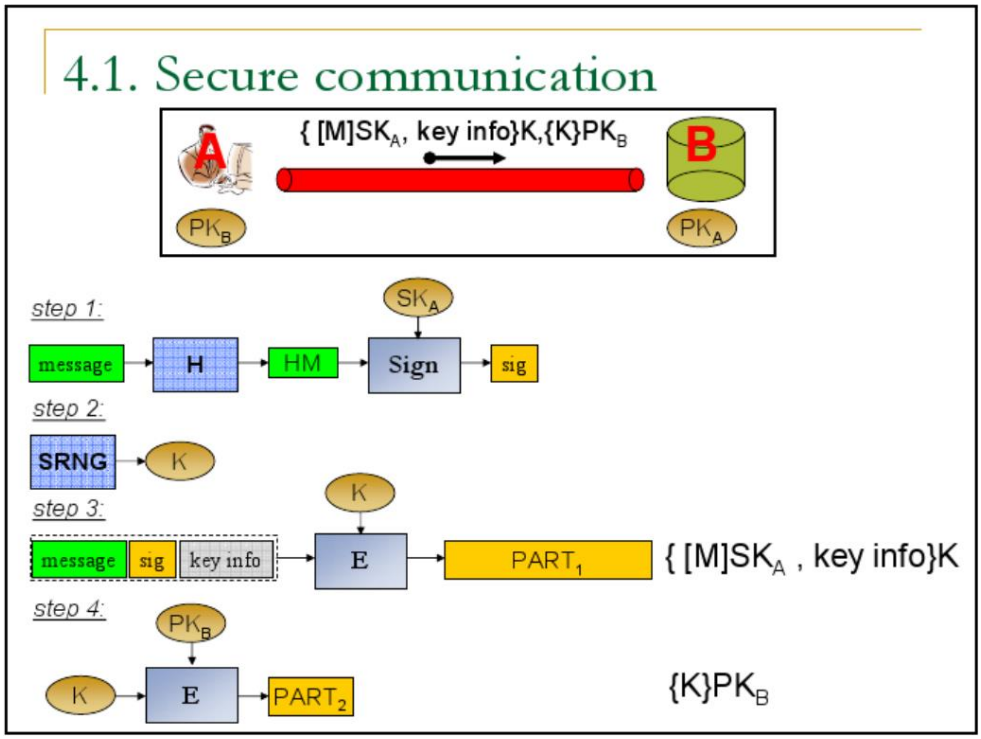
- Vervolgens kan je sleutels en certificaten ophalen uit de keystore. Voor sommige sleutels en/of certificaten is een paswoord vereist. De sleutels worden aangesproken via een alias. Gelijkaardig kan men certificaten ophalen. Codevoorbeelden vind je hieronder:

  ```
  PrivateKey key = (PrivateKey) keyStore.getKey(alias,password);
  Certificate cert = (Certificate) keyStore.getCertificate(alias,password);
  ```

- Haal nu in de keystore van Werner het certificaat op van Freya (dit is store1.jks). Encrypteer een boodschap met de publieke sleutel in het certificaat van Freya.
- Haal vervolgens de private sleutel uit de keystore van Freya (dit is store2.jks). Decrypteer het bericht met de private sleutel van Freya.

DEEL 3: Secure Communication

Implementeer nu het secure communication protocol uit sectie 4.1 in de cursus. Hiervoor dien je de voorgaande cryptografische operaties te combineren. Implementeer de cryptografische operaties voor zowel de zender als de ontvanger. Controleer bij de ontvanger de integriteit van de doorgestuurde boodschap.



Stuur de code op het einde van het labo door naar laurens.lemaire@cs.kuleuven.be