1. Objective:

- Use keytool and jarsigner
- Manage certificate with java.security.keyStore

2. Information

Information on the keystore to be created

- Name: myKeyStore
- Password: desstelecom
- Used Algorithm: DSA

Information on the certificate

- Alias name: yourname
- Alias password: yourname

3. Exercises

Create a keystore and a certificate

- Use keytool
- Generate a certificate in command line
- Check that the certificate is in the keystore
- Is the certificate is signed, and by whom?

Copy all the commands and results in a text file.

Keystore management in Java

Use of Java.security.keyStore

- Create an instance of a your keystore, generated by keytool in exercice 1.
- Get the certificate associed to your alias
- Get the public key
- Get the private key

Use of signedObject

- Sign an object with your certificate
- Check the signature of your object

Jar's signature with jarsigner

- Create a jar file
- Sign it with jarsigner
- Check the signature with jarsigner

Copy all the commands and results in a text file

1. EAP message flow

The EAP protocol is organized around two main phase: (i) identity exchange, and (ii) challenge-response exchange. Those messages are exchanged between an authenticator (system in charge of authentication), and a supplicant (the system to be authenticated). The message flow is depicted in Figure 1, where the authentication is based on a MD5 challenge response.

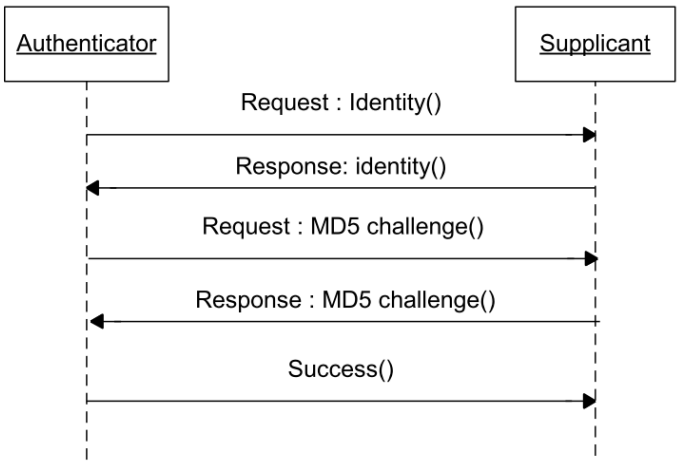

Figure 1. EAP Message flow

Identity exchange

In this step, the authenticator and supplicant exchange their identity. In figure 1., the authenticator send its identity with Request:Identity(). The supplicant replies with a Response:Identity().

Challenge-response exchange

Once supplicant and authenticator have exchanged their identity, the authenticator sends a challenge to the supplicant. In Figure 1., the authenticator sends a random String (Request :MD5-Challenge). The supplication hashes the challenge with MD5 and sends it back to the authenticator (Response:MD5-Challenge). The authenticator then computes the MD5 hash, and checks if it equals the MD5-hash received from the supplicant.

In case of successful authentication, the authenticator sends a EAP-Success, otherwise a EAP-Failure.

2. EAP Packet structure

The structure of EAP packet is depicted in Figure 2.

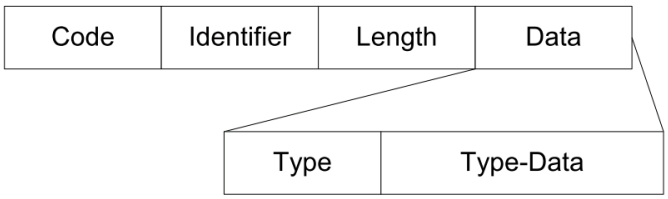

Figure 2. EAP Packet Structure

Code field

Code field describes the type of packet

- EAP-request
- EAP-response
- EAP-failure
- EAP-success

Identifier field

Identifier field provides a packet identifier, which is used for the binding between a request and its response. The two packets will have the same **identifier field.**

Length field

Length field is the size of the **data field.**

Data field

Data field is composed of two fields: **type** and **type-data**.

Type characterizes the type of data. The EAP's RFC defines a set of data types:

- **Identity**
- **Notification**
- **NAK**
- **EAP-MD5, EAP-OTP, EAP-GTC ,EAP-PAP ,EAP-TLS, LEAP, EAP-TTLS, PEAP**

Type-data contains the data of type **Type**.

EAP Packets with MD5 challenge response

Message	Code	Type	Type-Data
Request:Identity()	EAP-Request	Identity	Name
Response:Identity	EAP-Response	Identity	Name
Request:MD5Challenge()	EAP-Request	EAP-MD5	Challenge
Response:MD5Challenge()	EAP-Response	EAP-MD5	MD5 hash
Success	EAP-Success		
Failure	EAP-Failure		

3. Information

Frame.java - EAP Packet
Data.java - EAP Data field
ManInTheMiddle - is the ManInTheMiddle attacker
The authentication server has to be started first. It listens on port 1080.
The supplicant has to be started after.

4. Exercises

Implement the EAP-MD5 protocol

Modify Supplicant.authenticate() method

Modify AuthenticationServer.handleFrame(Frame) method

Implement a man in the middle attack over EAP-MD5 protocol

Modify Supplicant.main() method in order to create a socket with the ManInTheMiddle.

Start the authenticationserver, the ManInTheMiddle et then the Supplicant.

Implement the EAP-TLS protocol

The EAP-TLS protocol is based on the supplicant's certificate.

At authentication step, the supplication provides its certificate.

The supplication has to sign a challenge sent by the authenticator.