LAB3

Authors

- Hugo BEHERAY
- Zoé CROUZET

Exercise "Preparation of cryptographic security elements for each actor"

- *The CARoot has a key pair CAPubKey and CAPrivKey and a self-signed certificate CACert.crt. We ask you to generate for CARoot these cryptographic security elements*

We generate the key pair by using genrsa command. Then we split it in two, the private key and the public key.

For the Private and Public keys:

```
openssl genrsa -out CaKeyPair
openssl rsa -in CaKEY -pubout -out CAPubKey
mv CaKEY CAPrivKey
ls
CAPrivKey CAPubKey
```

For the self-signed certificate:

We create CACert by signing it with the CA private key.

```
openssl req -x509 -new -key CAPrivKey -out CACert.crt -days 500
ls
CACert.crt  CAPrivKey   CAPubKey   CaKeyPair
```

- *The Server has also a key pair ServerPubKey and ServerPrivKey. We ask you to generate for the Server these cryptographic security elements.*

We do the same step as before to create the public and private key.

```
cd Server/
openssl genrsa -out ServerKeyPair
openssl rsa -in ServerKeyPair  -pubout -out ServerPubKey
mv ServerKeyPair ServerPrivKey
ls
ServerPrivKey   ServerPubKey
```

- *The Client does not have any key.*
- *The Server needs to create a request for a certificate ServerRequest.csr. Afterwards, it will send this request to CARoot. We ask you to create for the Server ServerRequest.csr and send it to CARoot (by using the copy command as we have seen in the last session).*

We create the request for a certificate by entering this command:

```
openssl req -new -newkey rsa:4096 -nodes -keyout ServerKeyPair.key -out
ServerRequest.csr
```

Then we copy it in the CARoot folder by using this command:

```
cp /Users/hugobeheray/Documents/ING4S2/IS\
Security/LAB3/Server/ServerRequest.csr
/Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/CARoot
cd ..
cd CARoot/
ls
CACert.crt  CAPrivKey   CAPubKey    CaKeyPair   ServerRequest.csr
```

- *CARoot will generate ServerCert.crt and send it to the Server. We ask you to generate ServerCert.crt and send it to the Server.*

We generate ServerCert using the Server request, CACert and the CA private key.

```
openssl x509 -req -in ServerRequest.csr -CA CACert.crt -CAkey CAPrivKey -
CAcreateserial -out ServerCert.crt -days 500 -sha256
Signature ok
subject=/C=FR/ST=IDF/L=Paris/O=ECE/OU=IS/CN=secu/emailAddress=student@ece.
fr
Getting CA Private Key
#then we send it to server
cp ServerCert.crt /Users/hugobeheray/Documents/ING4S2/IS\
Security/LAB3/Server
```

- *CARoot will also send CACert.crt to the Server and the Client. Consequently, The Server and Client store CACert.crt as a Trusted Third Party. We ask you to send CACert.crt to the Server and the Client.*

```
#we copy CACert in the Client
cp /Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/CARoot/CACert.crt
/Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/Client

#we copy CACert in the Server
```

```
cp /Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/CARoot/CACert.crt
/Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/Server
```

Exercise "TLS Protocol"

- *The Client generates a file RNc. We ask you to create RNc and write a random number of your choice. Afterwards, you need to send it to the Server (by using the copy command as we have seen in the last session)*

```
touch RNc
vim RNc #we chose 12 as a random number
cp /Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/Client/RNc
/Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/Server
```

- *The Server generates a file RNs. We ask you to create RNs and write a random number of your choice. Afterwards, you need to send it to the Client*

```
touch RNs
vim RNs #we chose 27 as a random number
cp /Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/Server/RNs
/Users/hugobeheray/Documents/ING4S2/IS\ Security/LAB3/Client
```

- *The Server will generate SignS on the hash of RNc and RNs thanks to its private key Server-PrivKey:*
  - *Make a concatenation of RNc and RNs by entering the UNIX command:* `cat RNc RNs > RNcRNs`

  First, we concatenate RNc and RNs in RNcRNs.

  ```
  cat RNc RNs > RNcRNs
  ls
  CACert.crt  RNcRNs   ServerKeyPair.key ServerPubKey

  RNc   RNs   ServerPrivKey  ServerRequest.csr
  ```

  - *Apply the hash function SHA256 on RNcRNs to find its hash HashRNcRNs.*

  Then, we hash it using sha256.

  ```
  openssl dgst -sha256 -out HashRNcRNs RNcRNs
  ```

- *We ask you to proceed to generate SignS thanks to ServerPrivKey.*

We generated SignS thanks to the server private key and the Hash of RNc and RNs

```
  openssl rsautl -sign -in HashRNcRNs -inkey ServerPrivKey -out SignS
```

- *You can send now SignS and ServerCert.crt to the Client.*

```
  cp SignS /Users/hugobeheray/Documents/ING4S2/IS\
Security/LAB3/Client
  cp ServerCert.crt /Users/hugobeheray/Documents/ING4S2/IS\
Security/LAB3/Client
```

- *The Client needs to verify ServerCert.crt. We ask you to verify it as you did in the last session.*

```
  openssl verify -CAfile CACert.crt ServerCert.crt
```

- *The Client needs to verify SignS. We ask you to verify it as follows:*
  - *Extract ServerPubKey from ServerCert.crt*

First, we extract the server public key from the certificate using x509 and we store in ServerPubKey.

```
openssl x509 -pubkey -noout -in ServerCert.crt  > ServerPubKey
```

  - *Repeat the same steps to verify a signature as you did in part 5 of Exercise 2 in LAB1*

    ```
    openssl rsautl -verify -in SignS -pubin -inkey ServerPubKey -out
    HashAuthData
    openssl dgst -sha256 -out Hash AuthData
    diff Hash HashAuthData
    ```

- *The Client will generate a pre-master symmetric key PMsc and sends it encrypted to the Server:*
  - *We ask you to generate PMsc.*

We generate the symetric key as we did in previous labs with rand command.

```
  openssl rand -hex -out PMsc 64
```

  - *Encrypt PMsc thanks to ServerPubKey by naming the encrypted symmetric key PMscEncrypted*

We use rsault command to encrypt PMsc, we store it in PMscEncrypted.

```
openssl rsautl -encrypt -inkey ServerPubKey -pubin -in PMsc -out
PMscEncrypted
```

  - *Send PMscEncrypted to the Server*

We send the file by copying it into the server folder.

```
  cp PMscEncrypted /Users/hugobeheray/Documents/ING4S2/IS\
Security/LAB3/Server
```

  - *Decrypt PMscEncrypted thanks to ServerPrivateKey by naming the decrypted key PMsc.*

```
  openssl rsautl -decrypt -inkey ServerPrivKey -in PMscEncrypted -out
PMsc
```

- *Both the Client and server will generate the Master Symmetric key MS which is calculated from the hash of PMsc, RNc and RNs:*
  - *We ask you to calculate MS.*

We generate MS by concatening the hash of PMsc and RNcRNs

First, we hash PMsc:

```
  openssl dgst -sha256 -out HashPMsc PMsc
```

Then we cat both hash into MS.

```
  cat HashRNcRNs HashPMsc > MS
```

We do this Client and Server side.

  - *We ask you to create the file ClientData and encrypt it thanks to MS.*

We wrote "Data from client" in the file.

```
  touch ClientData
  vim ClientData
  openssl enc -in ClientData -out ClientDataEnc -e -des-cbc -k MS
```

  - *Send the encrypted result to the Server and decrypt it.*

```
cp ClientDataEnc ../Server
openssl enc -in ClientDataEnc -out ClientData  -d -des-cbc -k MS
cat ClientData
Data from client
```

We can see that we get the previous message that we wrote Client side

- *The Server will answer to the Client:*
  - *Create the file ServerData and encrypt it thanks to MS.*

We wrote "Data from server" in the file.

```
touch ServerData
vim ServerData
openssl enc -in ServerData -out ServerDataEnc -e -des-cbc -k MS
```

  - *Send the encrypted result to the Client and decrypt it.*

```
openssl enc -in ServerDataEnc -out ServerData  -d -des-cbc -k MS
cat ServerData
Data from Server
```

We can see that we get the previous message that we wrote Server side