II.2317 Cybersecurity (ISEP)

Nour El Madhoun

LAB 2

nour.el-madhoun@isep.fr

- **Important remarks:**
  - Firstly, we will give you the exact syntax for the commands to give you a general idea of how OpenSSL commands work. Afterwards, you have to do some research on the internet to find the right syntax and answer the questions.
  - The **three LABs** must be worked on by a team of **two students**.
  - Do not copy the commands as they are written in the pdfs because this can generate errors. It will be better if you write them.
  - You have to provide a **single report** for your work for the **three LABs**.
  - The report must contain **screenshots of all the parts with** *.
  - Do not forget to indicate **your names** in the report.
  - The report must be submitted on moodle before the **deadline that will be given during the session** by your supervisor.
  - If you have any questions, please contact your supervisor (and copy the e-mail to Nour El Madhoun).

1 Exercise "Symmetric cryptography" → (6.5 points)

In this exercise you will perform the scenario shown in Figure 1.

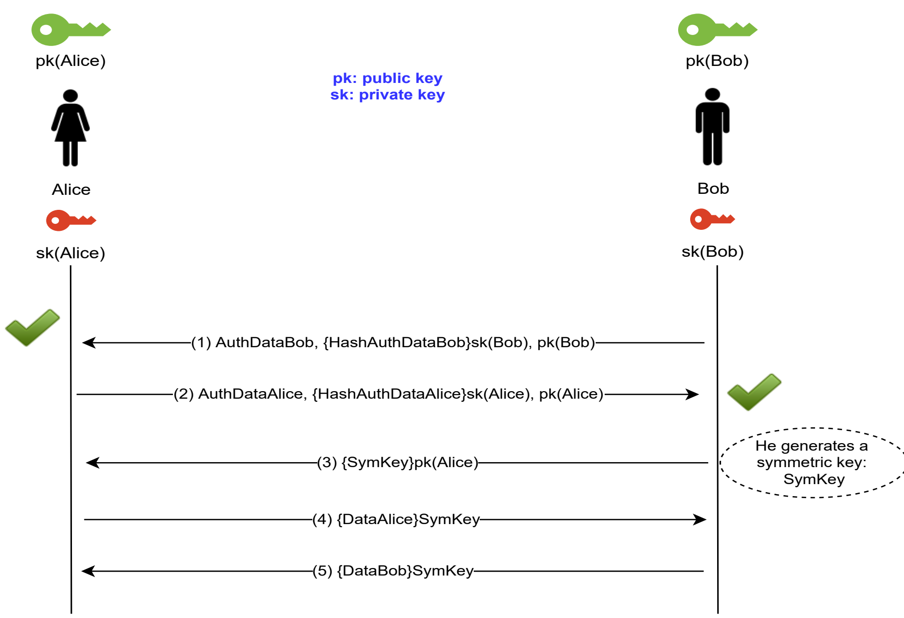

Figure 1: Exchanges between Alice And Bob

1. Please follow these steps to begin your practical work:
   (a) Open a ***Terminal***.
   (b) Create a new folder named ***LAB2*** and access this folder.
   (c) Create a new folder named ***Alice***.

(d) Create a new folder named ***Bob***.

(e) Repeat the same steps for ***Alice*** and ***Bob*** as you did in the previous LAB. At the end, you must have two files for ***Alice***: ***AlicePublicKey*** and ***AlicePrivateKey***, and two files for ***Bob***: ***BobPublicKey*** and ***BobPrivateKey***.

2. The objective of the message ***(1)*** is to: authenticate ***Bob*** to ***Alice***, ensure the non-repudiation for ***Bob***, guarantee the integrity of ***AuthDataBob***:

   (a) Access ***Bob***'s folder.

   (b) * Create a file named ***AuthDataBob*** and write a text of your choice → **(0.25 pt)**.

   (c) As we have seen in the previous LAB, in order to generate a signature, we have used two OpenSSL commands: the first one allows to generate the hash of the data and the second one allows to sign the hash thanks to the private key. In this LAB, we will use only one OpenSSL command that allows to generate the hash and sign it.

   - Consequently, please enter the following command to generate the signature of ***Bob*** on ***AuthDataBob***:
     openssl dgst -sha256 -sign BobPrivateKey -out BobSignature AuthDataBob
   - ***BobSignature*** is exactly equivalent to ***{HashAuthDataBob}sk(Bob)*** (see Figure 1). In order to simulate the sending of the message ***(1)*** from ***Bob*** to ***Alice***, you only need to copy ***AuthDataBob***, ***BobSignature***, ***BobPublicKey*** to ***Alice***'s folder.

   (d) Go back to the parent folder ***LAB2*** and access ***Alice***'s folder.

   (e) Check that ***AuthDataBob***, ***BobSignature***, ***BobPublicKey*** have been copied correctly to ***Alice***'s folder.

   (f) * We ask you now to proceed to verify ***BobSignature*** thanks to ***BobPublicKey***. To do this, you need to use only one OpenSSL command which is ***openssl dgst*** and the options: ***-sha256***, ***-verify***, ***-signature***. What do you notice? → **(1.5 pt)**.

   (g) For more information about the options: ***-sha256***, ***-verify***, ***-signature***, please use the command: ***openssl dgst -help***

3. The objective of the message ***(2)*** is to: authenticate ***Alice*** to ***Bob***, ensure the non-repudiation for ***Alice***, guarantee the integrity of ***AuthDataAlice***. We will repeat the same steps for this message as we did for message ***(1)***:

   (a) Create a file named ***AuthDataAlice*** and write a text of your choice.

   (b) * Generate ***AliceSignature*** → **(0.75 pt)**.

   (c) Copy ***AuthDataAlice***, ***AliceSignature***, ***AlicePublicKey*** to ***Bob***'s folder.

   (d) Go back to the parent folder ***LAB2*** and access ***Bob***'s folder.

   (e) Check that ***AuthDataAlice***, ***AliceSignature***, ***AlicePublicKey*** have been copied correctly to ***Bob***'s folder.

   (f) * Verify ***AliceSignature*** thanks to ***AlicePublicKey***. What do you notice? → **(0.75 pt)**.

4. The objective of the message ***(3)*** is to share a symmetric key in a secure manner:

   (a) Make sure you are in ***Bob***'s folder.

   (b) Generate a symmetric key ***SymKey*** by entering the OpenSSL command: ***openssl rand -hex -out SymKey 64***

   (c) Check that ***SymKey*** has been created correctly.

(d) * We ask you now to encrypt ***SymKey*** thanks to ***AlicePublicKey*** by naming the encrypted symmetric key ***SymKeyEncrypted*** → **(0.75 pt)**.

(e) Check that ***SymKeyEncrypted*** has been created correctly.

(f) In order to simulate the sending of the message ***(3)*** from ***Bob*** to ***Alice***, you only need to copy ***SymKeyEncrypted*** to ***Alice***'s folder.

(g) Go back to the parent folder ***LAB2*** and access ***Alice***'s folder.

(h) Check that ***SymKeyEncrypted*** has been copied correctly to ***Alice***'s folder.

(i) * Decrypt ***SymKeyEncrypted*** thanks to ***AlicePrivateKey*** by naming the decrypted key ***SymKey*** → **(0.5 pt)**.

5. The objective of messages ***(4)*** and ***(5)*** is to exchange confidential data in a secure and rapid manner thanks to the symmetric cryptography:

(a) Make sure you are in ***Alice***'s folder.

(b) Create a file named ***DataAlice*** and write a text of your choice.

(c) We will now encrypt ***DataAlice*** using the symmetric cryptography by naming the encrypted document ***DataAliceEncrypted***. To do this, please enter the OpenSSL command: ***openssl enc -e -aes-128-cbc -salt -pbkdf2 -kfile SymKey -in DataAlice -out DataAliceEncrypted***

(d) For more information about the options above, please use the command: ***openssl enc -help***

(e) In order to simulate the sending of the message ***(4)*** from ***Alice*** to ***Bob***, you only need to copy ***DataAliceEncrypted*** to ***Bob***'s folder.

(f) Go back to the parent folder ***LAB2*** and access ***Bob***'s folder.

(g) Check that ***DataAliceEncrypted*** has been copied correctly to ***Bob***'s folder.

(h) We will now decrypt ***DataAliceEncrypted*** thanks to ***SymKey*** by naming the decrypted document ***DataAlice***. To do this, please enter the OpenSSL command: ***openssl enc -d -aes-128-cbc -salt -pbkdf2 -kfile SymKey -in DataAliceEncrypted -out DataAlice***

(i) In order to verify the content of ***DataAlice***, you can use the following UNIX command: ***cat DataAlice***

(j) * You can now do the same steps for the message ***(5)*** → **(2 pt)**.

2 Exercise "Certificates X509" → (3.5 points)

1. Go back to the parent folder ***LAB2***.

2. Retrieve the certificate of the server ***www.lcl.fr*** and store the result in the file ***CertificateLCL*** by entering the OpenSSL command: ***openssl s_client -connect www.lcl.fr:443 > CertificateLCL***

3. Display the name of the ***CA*** that signed ***CertificateLCL*** by entering the OpenSSL command: ***openssl x509 -noout -in CertificateLCL -issuer***

4. Please use the command ***openssl x509 -help*** for more information.

5. * Display the validity date of ***CertificateLCL*** → **(0.5 pt)**.

6. * Display the signature of ***CertificateLCL*** → **(0.5 pt)**.

7. * Display the serial number of ***CertificateLCL*** → **(0.5 pt)**.
8. * Display the public key of ***CertificateLCL*** → **(0.5 pt)**.
9. The objective of this step is to generate a self-signed certificate for a ***CA*** and a certificate for a server thanks to the private key of a ***CA***:

```
nour@nour: ~
root@nour:/home/nour/LAB2# openssl req -new -key ServerPrivateKey
-out ServerRequest.csr
You are about to be asked to enter information that will be incorp
orated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
 or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile de France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:EPITA
Organizational Unit Name (eg, section) []:First year
Common Name (e.g. server FQDN or YOUR name) []:myserver.fr
Email Address []:youremail

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Figure 2: Certificate Request for the Server: ***ServerRequest.csr***

(a) Make sure you are in ***LAB2***'s folder.

(b) Generate a key pair (2048 bits) for a server protected by a password of your choice (for example: password) by entering the OpenSSL command: ***openssl genrsa -out ServerKeyPair -passout pass:password 2048***

```
nour@nour: ~
root@nour:/home/nour/LAB2# openssl req -x509 -new -key CAKeyPair -
out CACertificate.crt -days 500
You are about to be asked to enter information that will be incorp
orated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
 or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Ile de France
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyCA
Organizational Unit Name (eg, section) []:MyCA
Common Name (e.g. server FQDN or YOUR name) []:MyCA
Email Address []:CAemail
root@nour:/home/nour/LAB2#
```

Figure 3: Self-Signed Certificate ***CACertificate.crt*** for the *CA*

(c) * Extract the public key of the server ***ServerPublicKey*** as we have seen in the previous LAB → **(0.5 pt)**.

(d) Rename ***ServerKeyPair*** by entering the UNIX command:
mv ServerKeyPair ServerPrivateKey

(e) Generate the certificate request for the server ***ServerRequest.csr*** as shown in Figure 2. Put the same information as shown in Figure 2 for ***Country Name, State or Province Name, Locality Name, Organization Name, Organizational Unit Name, Common Name, Email Address*** and leave blank by pressing enter for ***A challenge password, An optional company name***.

(f) * Generate a key (4096 bits) pair ***CAKeyPair*** for a ***CA*** protected by a password of your choice (for example: password) → **(0.5 pt)**.

(g) Generate a self-signed certificate ***CACertificate.crt*** for the *CA* as shown in Figure 3. Put the same information as shown in Figure 3 for ***Country Name, State or Province Name, Locality Name, Organization Name, Organizational Unit Name, Common Name, Email Address***.

(h) Now, we will generate the certificate of the server ***ServerCertificate.crt*** by using the certificate and keys of the ***CA***. To do this, please enter the OpenSSL command:
openssl x509 -req -in ServerRequest.csr -CA CACertificate.crt -CAkey CAKeyPair -CAcreateserial -out ServerCertificate.crt -days 500 -sha256

(i) * You can verify the certificate of the server by entering the OpenSSL command: ***openssl verify -CAfile CACertificate.crt ServerCertificate.crt*** . What do you notice? → **(0.5 pt)**