03-CipherLab - Oblig02

By Kristoffer-Andre Kalliainen (181192)

GitHub repository - 181192

About the project

- CLI support made possible with PicoCLI
- Compiled with Java 10
- Using Maven as build framework and dependency manager
- Maven-Assemby-Plugin to deploy multiple jars from the same source code

To build the project from scratch run `mvn clean install` in the project directory, or provide a path to the `pom.xml`.

Implemented help support either with `-h` or with `--help`:

```
$ java -jar target/BasicClient-1.0.0-jar-with-dependencies.jar --help
Usage: CryptoUltimate [-h] [-m=MESSAGE]
Send message with given argument
  -h, --help              display a help message
  -m, --message=MESSAGE   Message to encrypt
```

Part 01 - Basic

This is just the standard CipherLab.zip.

> *The server*

```
$ java -jar target/BasicServer-1.0.0-jar-with-dependencies.jar
Waiting for requests from client...
Connected to client at the address: /127.0.0.1
Message from DesClient: This is really cool
Waiting for requests from client...
```

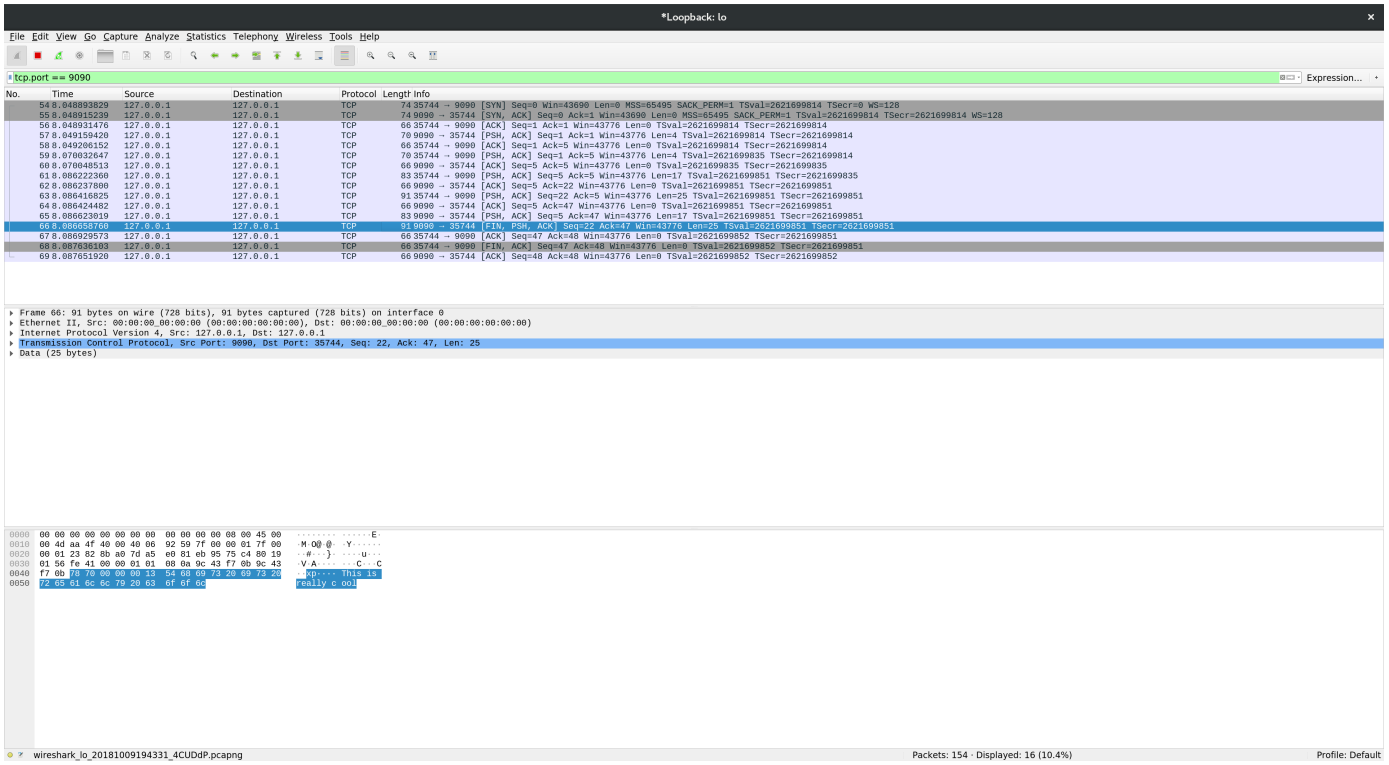
> *The client*

```
$ java -jar target/BasicClient-1.0.0-jar-with-dependencies.jar -m "This is really cool"
Connected to DesServer on localhost/127.0.0.1
Response from server: This is really cool
```

Wireshark

Here's an screenshot from the "basic" server-client program. Since the communication is going over standard tcp we can read the communication in clear text. The highlighted text is the message the client is sending the server

The message that is being sent is `This is really cool`.



Part 02 - DES encryption

This is a DES implementation using ECB mode with Bounty Castle Provider and Secret Key using Java KeyStore.

> *The server*

```
$ java -jar target/DesServer-1.0.0-jar-with-dependencies.jar
Waiting for requests from client...
Connected to client at the address: /127.0.0.1
Message from DesClient: DES is also cool
Waiting for requests from client...
```

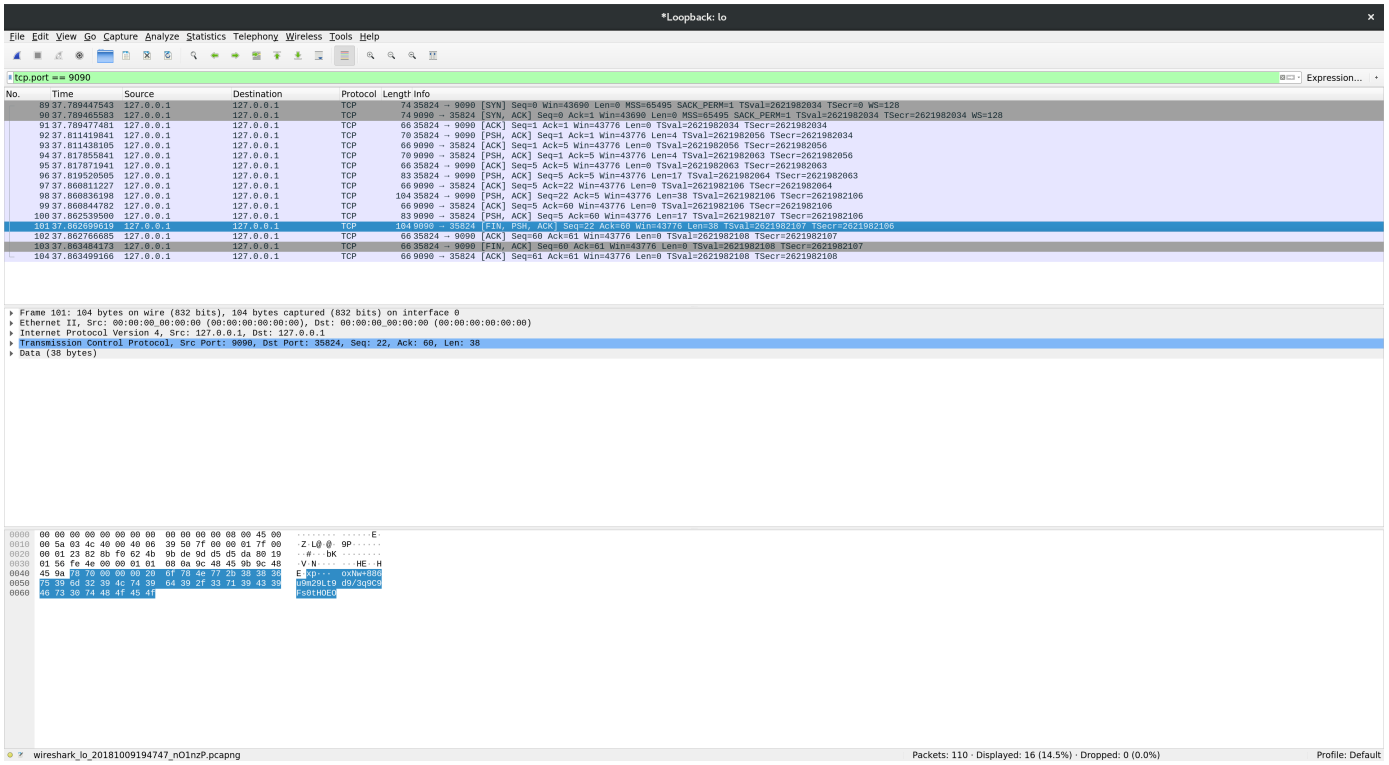
> *The client*

```
$ java -jar target/DesClient-1.0.0-jar-with-dependencies.jar -m "DES is also cool"
Connected to DesServer on localhost/127.0.0.1
Response from server: DES is also cool
```

Wireshark

Here's an screenshot from the "DES" version of the server-client program. The traffic is still unsecured but the message is encrypted. So we can analyse all the steps in the TCP handshake and retrieve information, but the message itself

is encrypted with DES and we can only see a Base64 representation of the message. The highlighted row is the message the client is sending the server. The message is `DES is also cool`



Part 03 - SSL and CA

This is a SSL implementation with Certificate Authority using Java TrustStore and KeyStore.

Generating Certificates

> *Generate Certificates in TrustStore and KeyStore by running*

```
$ ./generate_certificate.sh

What is your first and last name?
  [Unknown]:  Kristoffer-Andre Kalliainen
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:  HVL
What is the name of your City or Locality?
  [Unknown]:  Bergen
What is the name of your State or Province?
  [Unknown]:  Hordaland
What is the two-letter country code for this unit?
  [Unknown]:  NO
Is CN=Kristoffer-Andre Kalliainen, OU=Unknown, O=HVL, L=Bergen, ST=Hordaland, C=NO correct?
  [no]:  yes
```

```
Certificate stored in file <src/main/resources/server.cer>

Owner: CN=Kristoffer-Andre Kalliainen, OU=Unknown, O=HVL, L=Bergen, ST=Hordaland, C=NO
Issuer: CN=Kristoffer-Andre Kalliainen, OU=Unknown, O=HVL, L=Bergen, ST=Hordaland, C=NO
Serial number: 36642341
Valid from: Mon Oct 08 00:01:42 CEST 2018 until: Sat Jan 05 23:01:42 CET 2019
Certificate fingerprints:
         MD5:  A4:14:45:57:D3:04:7F:20:A0:F3:F5:72:7C:0B:BC:65
         SHA1: 52:94:E7:A1:03:22:45:85:8D:69:A0:D6:82:2A:1F:7F:16:0A:4C:3C
         SHA256: 4C:BC:41:4D:F1:01:60:30:32:CD:CD:0D:A5:7A:F3:5D:F4:3C:37:04:B6:66:7E:F3:EE:AA:
C1:AC:4A:1B:B5:C7
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 72 7A 15 6E 89 42 38 3E   2F DD 20 C9 69 20 6F 9B  rz.n.B8>/. .i o.
0010: 66 6E 40 35                                        fn@5
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
[Storing src/main/resources/cacerts.jceks]
Keystore type: JCEKS
Keystore provider: SunJCE

Your keystore contains 1 entry

Alias name: server-alias
Creation date: Oct 8, 2018
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Kristoffer-Andre Kalliainen, OU=Unknown, O=HVL, L=Bergen, ST=Hordaland, C=NO
Issuer: CN=Kristoffer-Andre Kalliainen, OU=Unknown, O=HVL, L=Bergen, ST=Hordaland, C=NO
Serial number: 36642341
Valid from: Mon Oct 08 00:01:42 CEST 2018 until: Sat Jan 05 23:01:42 CET 2019
Certificate fingerprints:
         MD5:  A4:14:45:57:D3:04:7F:20:A0:F3:F5:72:7C:0B:BC:65
```

```
         SHA1: 52:94:E7:A1:03:22:45:85:8D:69:A0:D6:82:2A:1F:7F:16:0A:4C:3C
         SHA256: 4C:BC:41:4D:F1:01:60:30:32:CD:CD:0D:A5:7A:F3:5D:F4:3C:37:04:B6:66:7E:F3:EE:AA:
C1:AC:4A:1B:B5:C7
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 72 7A 15 6E 89 42 38 3E   2F DD 20 C9 69 20 6F 9B  rz.n.B8>/. .i o.
0010: 66 6E 40 35                                        fn@5
]
]



*******************************************
*******************************************

Keystore type: JCEKS
Keystore provider: SunJCE

Your keystore contains 1 entry

server-alias, Oct 8, 2018, trustedCertEntry,
Certificate fingerprint (SHA1): 52:94:E7:A1:03:22:45:85:8D:69:A0:D6:82:2A:1F:7F:16:0A:4C:3C
```

Generating SecureKey

> *Generate SecureKey by running:*

```
$ ./generate_securekey.sh

Keystore type: JCEKS
Keystore provider: SunJCE

Your keystore contains 1 entry

Alias name: securekey
Creation date: Oct 8, 2018
Entry type: SecretKeyEntry
```

```
****************************************
****************************************
```

> *The server*

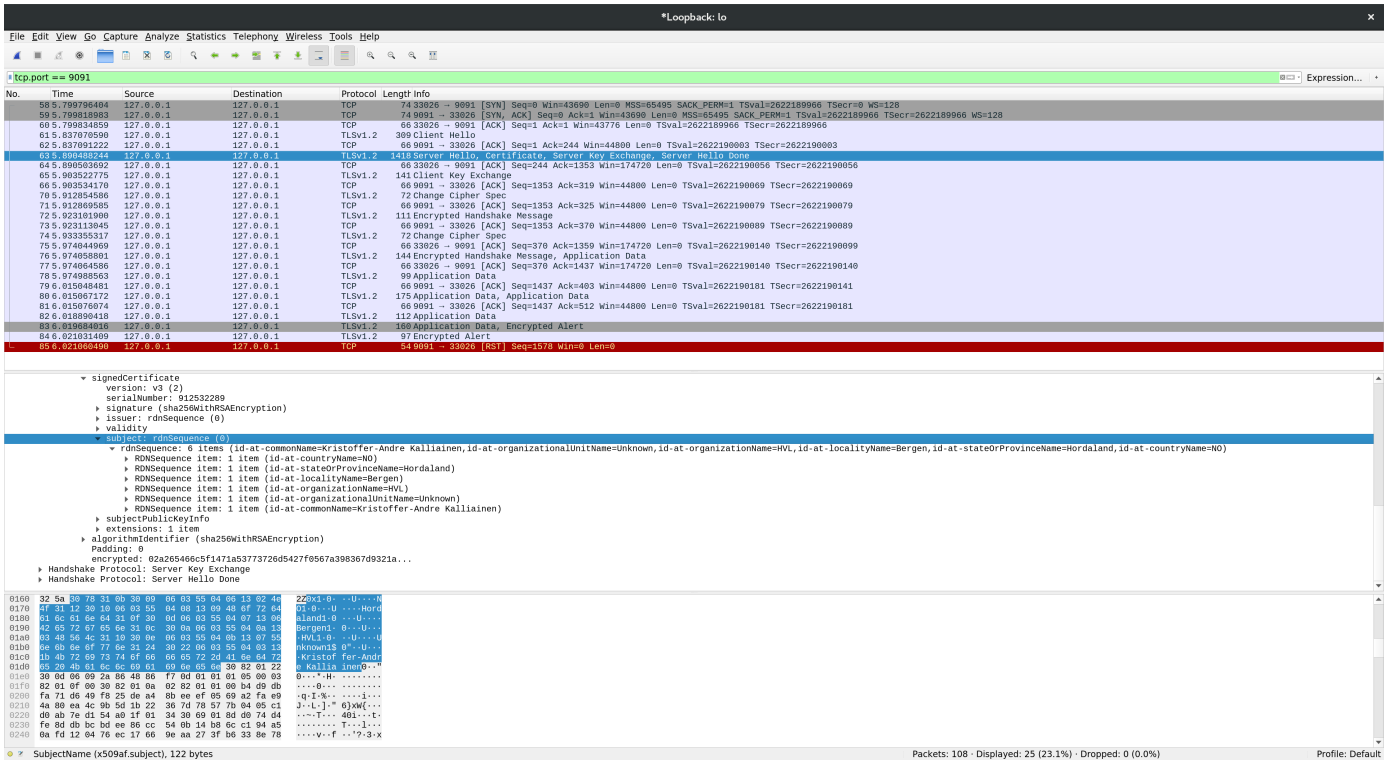
```
$ java -jar target/SslServer-1.0.0-jar-with-dependencies.jar
Waiting for requests from client...
Connected to client at the address: /127.0.0.1
Message from DesClient: ThisIsACoolTest
Waiting for requests from client...
```

> *The client*

```
$ java -jar target/SslClient-1.0.0-jar-with-dependencies.jar -m ThisIsACoolTest
Connected to DesServer on localhost/127.0.0.1
Response from server: ThisIsACoolTest
```

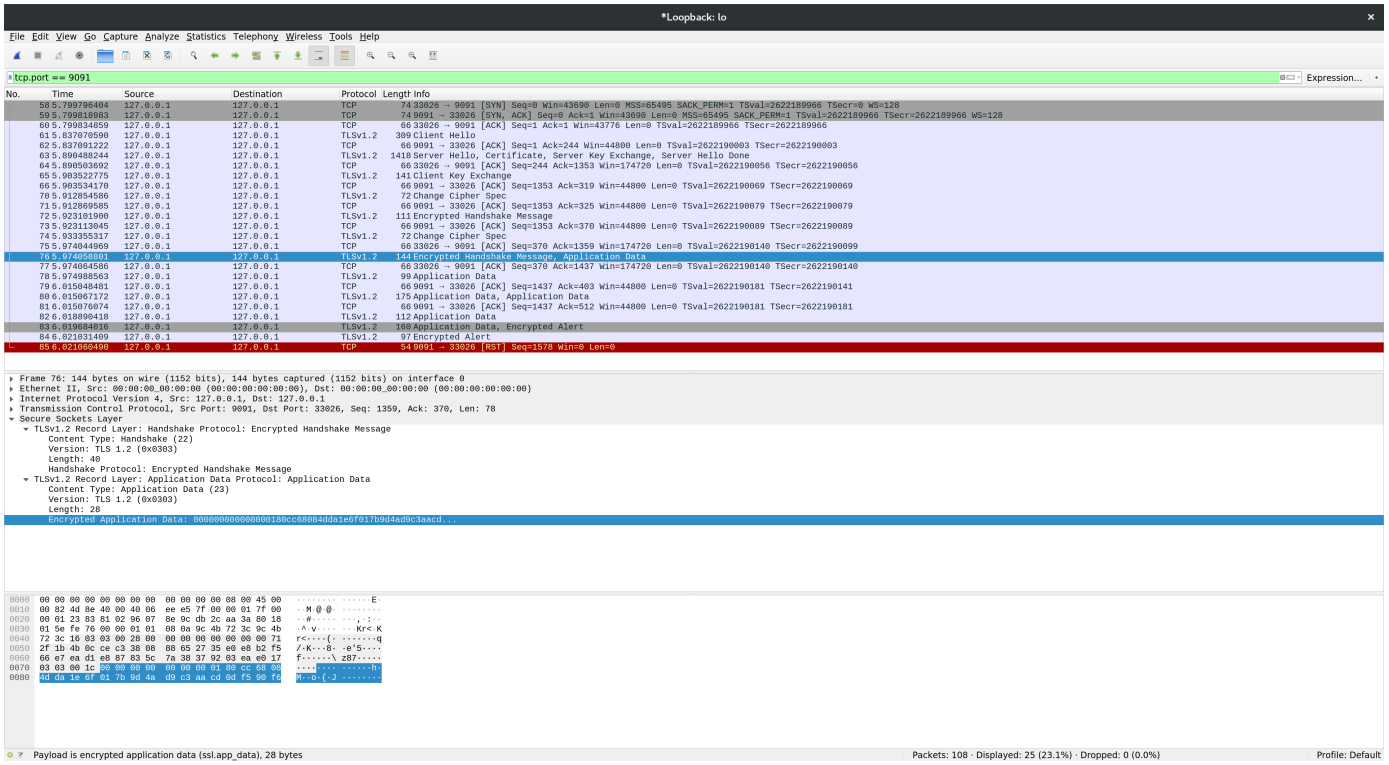
Wireshark

The following picture shows the certificate in the wireshark output.



Here's an screenshot from the "SSL" version of the server-client program. The traffic is now encrypted with TLS 1.2, so we can't see the information being sent over TCP, it's all encrypted. The highlighted row on the image is showing the message being sent from the client to the server. As we see the traffic is encrypted and we can't retrieve any usefull

information about the message being sent.



Resources

Here's some resources used for inspiration, and some resources for solving the problems that I had solving this assignment. Especially there was some struggling with the TrustStore and KeyStore for the CA-certificate. And when I needed to store a SecureKey in the KeyStore I needed to use a JCEKS that support storing SecureKeys and other keypair, instead of the "standard" JKS format.

- Java2s - Encrypting a string with DES
- Stackoverflow - Base64 Encoding in Java
- pyJKS - Difference between JKS and JCEKS keystores
- neil Madden - Java keystores
- Should I use ECB or CBC encryption mode for my block cipher?
- JenKov - Java KeyStore
- Java Code Examples for java.security.KeyStore.SecretKeyEntry
- Generate Secure Keys with Keytool
- keytool - Key and Certificate Management Tool
- DES with ECB example
- Java Code Examples for javax.net.ssl.SSLServerSocketFactory
- Java Security Tutorial – Step by Step guide to create SSL connection and certificates
- How do I load a file from resource folder?
- Maven - Bounty Castle Provider