

Lab 6
Gunnar Yonker

1.

MeetingRoomClientLab6.java – contains the digital certificate exchange and the 4 step key exchange protocol for establishing a session key based on Fig 15.5

MeetingRoomServerLab6.java - contains the digital certificate exchange and the 4 step key exchange protocol for establishing a session key based on Fig 15.5

Certificate_Authority.java – used to generate the digital certifications with the CA

Files used:

clientKeyStore.jks – clients digital certification from Certificate_Authority.java program

ClientPrivateKey.txt – client's private key, 1024 bit

ClientPublicKey.txt – client's public key, 1024 bit

keystoreCA.jks – CA keystore file for verification

LoginCreds.txt – authorized login combinations for reservations

MeetingTimes.txt – All available and reserved meeting times for reservations

receivedclientcert.jks – Server creates and uses this file sent from client to verify digital certificate

receivedservercert.jks – Client creates and uses this file sent from server to verify digital certificate

serverKeyStore.jks – server's digital certification from Certificate_Authority.java program

ServerPrivateKey.txt - server's private key, 1024 bit

ServerPublicKey.txt – server's public key, 1024 bit

Key Exchange Protocol:

Step 1: $E(P_{Ub}, [N_1 || IDA])$

Client(A) sends an encrypted message containing a random nonce and the IDA which is any identifying factor to Server(B). Encrypted using P_{Ub} which is B's public key, B decrypts the message using PR_b (B's private key) to see N_1

Step 2: $E(P_{Ua}, [N_1 || N_2])$

Server(B) will generate a random nonce N_2 . Then send a message containing N_1 and N_2 back to Client(A). It will be encrypted using P_{Ua} , A's public key. A will decrypt it using PR_a , A's private key and check to see if B's N_1 reply matches the N_1 sent to defend against a replay attack.

Step 3: $E(P_{Ub}, N_2)$

Client(A) sends an encrypted message containing N_2 to Server(B), encrypted using P_{Ub} , B's public key. B decrypts the message using PR_b , B's private key and checks to see if N_2 received matches the N_2 sent. This ensures to defend against a replay attack.

Gunnar Yonker

Client(A) encrypts the session key(K_s) using their private key(PR_A). Then A encrypts the encrypted session key using Server(B)'s public key(PUB) and sends it to the server. B then decrypts the message using their private key(PR_B), and then decrypts it again using A's public key(PU_A). A and B now have established a session key that can be used for further encryption and decryption of the reservation program. Key exchange protocol is complete at this point.

Server waiting for client connection to begin:

The screenshot displays the Eclipse IDE with the following components:

- Editors:**
 - `Certificate_Authory.java`
 - `MeetingRoomClientLab6.java`
 - `MeetingRoomServerLab6.java` (Active Editor)
- Code Editor (MeetingRoomServerLab6.java):**

```

20 * File: MeetingRoomServerLab6.java
32 import java.io.*;
50
51 public class MeetingRoomServerLab6 {
52     //Port 755 for localhost connection between client and server
53     private static final int PORT = 755;
54     //Text files being used that hold authorized logins and scheduled meeting times/available meeting times
55     private static final String LOGIN_CREDS_FILE = "LoginCreds.txt";
56     private static final String MEETING_TIMES_FILE = "MeetingTimes.txt";
57
58     public static void main(String[] args) throws NoSuchAlgorithmException {
59
60         while (true) {
61             //Read the PublicKey.txt for the publicKey for encryption
62             String publicKeyStr = "";
63             try (BufferedReader br = new BufferedReader(new FileReader("ClientPublicKey.txt"))) {
64                 publicKeyStr = br.readLine();
65             } catch (IOException e) {
66                 System.out.println("Error reading public key file: " + e.getMessage());
67             }
68             byte[] publicKeyBytes = Base64.getDecoder().decode(publicKeyStr);
69
70             //Read the PrivateKey.txt for the privateKey for encryption
71             String privateKey = "";
72             try (BufferedReader br = new BufferedReader(new FileReader("ServerPrivateKey.txt"))) {
73                 privateKey = br.readLine();
74             } catch (IOException e) {
75                 System.out.println("Error reading private key file: " + e.getMessage());
76             }
77             byte[] privateKeyBytes = Base64.getDecoder().decode(privateKey);
78
79             //Print out that server is up and waiting for response from client
80             try (ServerSocket server = new ServerSocket(PORT)) {
81                 System.out.println("Meeting Room Server up and running, waiting on port " + PORT + "...");
82                 Socket socket = server.accept();
83                 //Socket socket = server.accept();

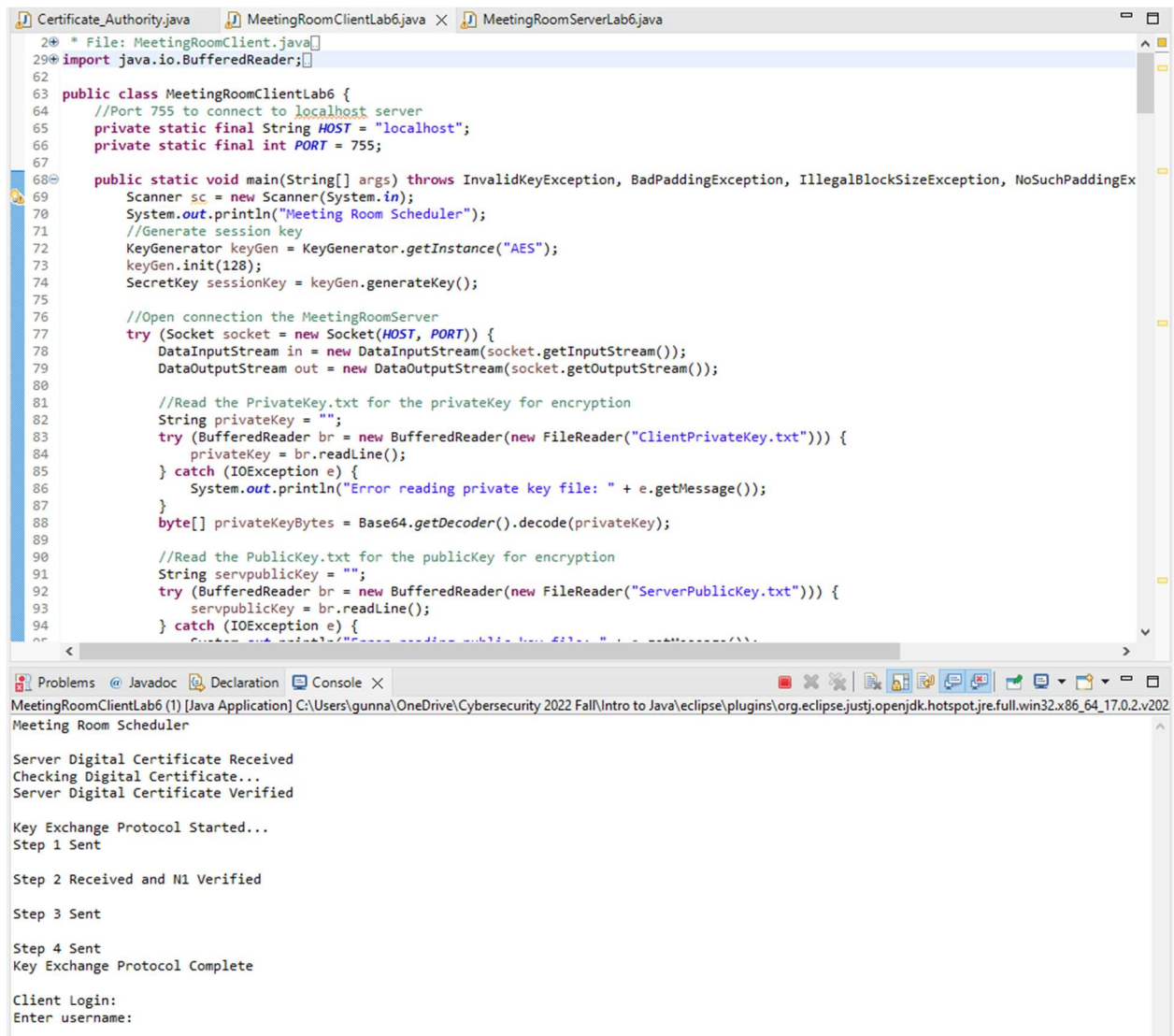
```
- Console:**
 - Problems
 - @ Javadoc
 - Declaration
 - Console

MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\bin\java.exe -Xmx1024m -Xms64m -XX:MaxPermSize=256m -Djava.awt.headless=true -Djava.class.path=C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\rt.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\access-bridge-64.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\access-bridge.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-ffmpeg.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-gstreamer.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-qt.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-sdl.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-vorbis.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v2.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v3.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v4.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v5.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v6.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v7.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v8.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v9.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v10.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v11.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v12.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v13.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v14.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v15.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v16.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v17.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v18.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v19.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v20.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v21.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v22.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v23.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v24.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v25.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v26.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v27.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v28.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v29.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v30.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220816\jre\lib\ext\jfx-media-x264v31.jar;C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full

Lab 6

Gunnar Yonker

Client view when connecting successfully:



The screenshot displays the Eclipse IDE with the `MeetingRoomClientLab6.java` file open. The code defines a `MeetingRoomClientLab6` class that connects to a server at `localhost:755`. It generates a session key, reads a private key from `ClientPrivateKey.txt`, and reads a public key from `ServerPublicKey.txt`. The console output shows the successful execution of the client, including the receipt and verification of a digital certificate and the completion of a four-step key exchange protocol.

```
29 import java.io.BufferedReader;
62
63 public class MeetingRoomClientLab6 {
64     //Port 755 to connect to localhost server
65     private static final String HOST = "localhost";
66     private static final int PORT = 755;
67
68     public static void main(String[] args) throws InvalidKeyException, BadPaddingException, IllegalBlockSizeException, NoSuchPaddingEx
69     {
70         Scanner sc = new Scanner(System.in);
71         System.out.println("Meeting Room Scheduler");
72         //Generate session key
73         KeyGenerator keyGen = KeyGenerator.getInstance("AES");
74         keyGen.init(128);
75         SecretKey sessionKey = keyGen.generateKey();
76
77         //Open connection the MeetingRoomServer
78         try (Socket socket = new Socket(HOST, PORT)) {
79             DataInputStream in = new DataInputStream(socket.getInputStream());
80             DataOutputStream out = new DataOutputStream(socket.getOutputStream());
81
82             //Read the PrivateKey.txt for the privateKey for encryption
83             String privateKey = "";
84             try (BufferedReader br = new BufferedReader(new FileReader("ClientPrivateKey.txt"))) {
85                 privateKey = br.readLine();
86             } catch (IOException e) {
87                 System.out.println("Error reading private key file: " + e.getMessage());
88             }
89             byte[] privateKeyBytes = Base64.getDecoder().decode(privateKey);
90
91             //Read the PublicKey.txt for the publicKey for encryption
92             String servpublicKey = "";
93             try (BufferedReader br = new BufferedReader(new FileReader("ServerPublicKey.txt"))) {
94                 servpublicKey = br.readLine();
95             } catch (IOException e) {
96                 System.out.println("Error reading public key file: " + e.getMessage());
97             }
98         }
99     }
100 }
```

Problems @ Javadoc Declaration Console X

MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v202

Meeting Room Scheduler

Server Digital Certificate Received
Checking Digital Certificate...
Server Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Sent

Step 2 Received and N1 Verified

Step 3 Sent

Step 4 Sent
Key Exchange Protocol Complete

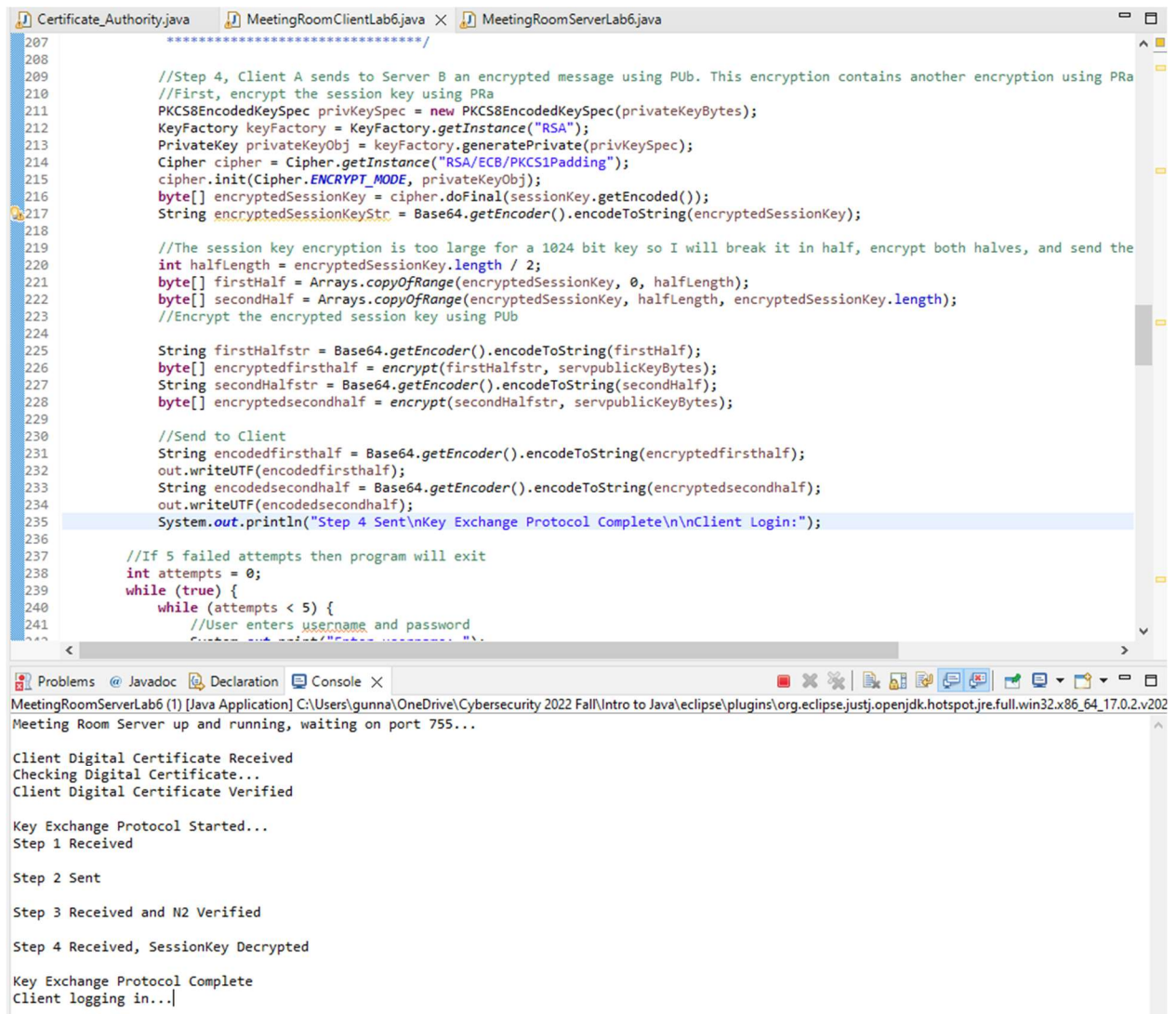
Client Login:
Enter username:

Digital certificate is received from server, verified, and then the key exchange protocol begins and progresses through each step. Once it is complete and the session key is established then the client can login.

Lab 6

Gunnar Yonker

Server view:



The screenshot displays the Eclipse IDE with three open Java files: `Certificate_Authority.java`, `MeetingRoomClientLab6.java`, and `MeetingRoomServerLab6.java`. The `MeetingRoomServerLab6.java` file is the active editor, showing code for a server that implements a key exchange protocol. The code includes comments and logic for generating a private key, encrypting a session key, and sending it to the client in two parts. The console output at the bottom shows the server's execution, including receiving a digital certificate, verifying it, and completing the key exchange protocol.

```
207 //Step 4, Client A sends to Server B an encrypted message using Pub. This encryption contains another encryption using PRA
208 //First, encrypt the session key using PRA
209 PKCS8EncodedKeySpec privKeySpec = new PKCS8EncodedKeySpec(privateKeyBytes);
210 KeyFactory keyFactory = KeyFactory.getInstance("RSA");
211 PrivateKey privateKeyObj = keyFactory.generatePrivate(privKeySpec);
212 Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
213 cipher.init(Cipher.ENCRYPT_MODE, privateKeyObj);
214 byte[] encryptedSessionKey = cipher.doFinal(sessionKey.getEncoded());
215 String encryptedSessionKeyStr = Base64.getEncoder().encodeToString(encryptedSessionKey);
216
217 //The session key encryption is too large for a 1024 bit key so I will break it in half, encrypt both halves, and send the
218 int halfLength = encryptedSessionKey.length / 2;
219 byte[] firstHalf = Arrays.copyOfRange(encryptedSessionKey, 0, halfLength);
220 byte[] secondHalf = Arrays.copyOfRange(encryptedSessionKey, halfLength, encryptedSessionKey.length);
221 //Encrypt the encrypted session key using PUB
222
223 String firstHalfstr = Base64.getEncoder().encodeToString(firstHalf);
224 byte[] encryptedfirsthalf = encrypt(firstHalfstr, servPublicKeyBytes);
225 String secondHalfstr = Base64.getEncoder().encodeToString(secondHalf);
226 byte[] encryptedsecondhalf = encrypt(secondHalfstr, servPublicKeyBytes);
227
228 //Send to Client
229 String encodedfirsthalf = Base64.getEncoder().encodeToString(encryptedfirsthalf);
230 out.writeUTF(encodedfirsthalf);
231 String encodedsecondhalf = Base64.getEncoder().encodeToString(encryptedsecondhalf);
232 out.writeUTF(encodedsecondhalf);
233 System.out.println("Step 4 Sent\nKey Exchange Protocol Complete\n\nClient Login:");
234
235 //If 5 failed attempts then program will exit
236 int attempts = 0;
237 while (true) {
238     while (attempts < 5) {
239         //User enters username and password
240         System.out.print("Enter username: ");
241         String username = in.readLine();
242         System.out.print("Enter password: ");
243         String password = in.readLine();
244         //Check if username and password are correct
245         if (username.equals(userName) && password.equals(password)) {
246             System.out.println("Login successful");
247             return true;
248         } else {
249             attempts++;
250             System.out.println("Invalid username or password. Attempts left: " + (5 - attempts));
251         }
252     }
253     System.out.println("Failed login attempts. Program will exit.");
254     return false;
255 }
```

Problems @ Javadoc Declaration Console X

MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220714-1955\bin\java.exe

Meeting Room Server up and running, waiting on port 755...

Client Digital Certificate Received
Checking Digital Certificate...
Client Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Received
Step 2 Sent
Step 3 Received and N2 Verified
Step 4 Received, SessionKey Decrypted

Key Exchange Protocol Complete
Client logging in..|

Digital certificate is received from client, verified, and then the key exchange protocol begins and progresses through each step. Once it is complete and the session key is established then the client can login and server waits to check the login credentials.

Lab 6

Gunnar Yonker

Client Fails 5 times, disconnected due to too many failed attempts:

Client view:

```
Problems @ Javadoc Declaration Console X
<terminated> MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
Key Exchange Protocol Complete

Client Login:
Enter username: t
Enter password: t
Login failed, try again
Enter username: t
Enter password: t
Login failed, try again
Enter username: t
Enter password: t
Login failed, try again
Enter username: t
Enter password: t
Login failed, try again
Enter username: t
Enter password: t
Login failed, try again
Enter username: t
Enter password: t
Login failed, try again
Max login attempts reached, try again later.
```

Server view:

```
Problems @ Javadoc Declaration Console X
MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v202
Meeting Room Server up and running, waiting on port 755...

Client Digital Certificate Received
Checking Digital Certificate...
Client Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Received

Step 2 Sent
|
Step 3 Received and N2 Verified

Step 4 Received, SessionKey Decrypted

Key Exchange Protocol Complete
Client logging in...
Client Disconnected, restarting...
Meeting Room Server up and running, waiting on port 755
```

Lab 6

Gunnar Yonker

Client successfully logs in and reserves a timeslot:

Client view:

```
Problems @ Javadoc Declaration Console X
MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20

Step 4 Sent
Key Exchange Protocol Complete

Client Login:
Enter username: gunnar
Enter password: student
Login successful

Available meeting times:
11am
12pm
1pm
2pm
3pm
4pm

Enter desired meeting time: 1pm
Meeting time slot reserved at 1pm for gunnar
Do you want to make another reservation (y/n)? |
```

Server view:

```
Problems @ Javadoc Declaration Console X
MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20

Client Digital Certificate Received
Checking Digital Certificate...
Client Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Received

Step 2 Sent

Step 3 Received and N2 Verified

Step 4 Received, SessionKey Decrypted

Key Exchange Protocol Complete
Client logging in...

Client logged in, showing available times...
Client timeslot reserved
```

If client chooses to not reserve again program exits and server restarts:

Client view:

```
Problems @ Javadoc Declaration Console X
MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20

<terminated> MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20

Step 4 Sent
Key Exchange Protocol Complete

Client Login:
Enter username: gunnar
Enter password: student
Login successful

Available meeting times:
11am
12pm
1pm
2pm
3pm
4pm

Enter desired meeting time: 1pm
Meeting time slot reserved at 1pm for gunnar
Do you want to make another reservation (y/n)? n
```


Lab 6

Gunnar Yonker

Server view:

```
MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v202
Checking Digital Certificate...
Client Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Received

Step 2 Sent
|
Step 3 Received and N2 Verified

Step 4 Received, SessionKey Decrypted

Key Exchange Protocol Complete
Client logging in...

Client logged in, showing available times...
Client timeslot reserved
Client Disconnected, restarting...
Meeting Room Server up and running, waiting on port 755...
```

If client says y, then they are prompted to login and make another reservation:

Client view:

```
MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v202
11am
12pm
2pm
3pm
4pm

Enter desired meeting time: 4pm
Meeting time slot reserved at 4pm for gunnar
Do you want to make another reservation (y/n)? y
Enter username: gunnar
Enter password: student
Login successful

Available meeting times:
11am
12pm
2pm
3pm

Enter desired meeting time:
```

Server view:

```
MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v202
Checking Digital Certificate...
Client Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Received

Step 2 Sent

Step 3 Received and N2 Verified

Step 4 Received, SessionKey Decrypted

Key Exchange Protocol Complete
Client logging in...

Client logged in, showing available times...
Client timeslot reserved

Client logged in, showing available times...
```

Lab 6

Gunnar Yonker

If all time slots are reserved, client is notified and program exits:

Client view:



```
Problems @ Javadoc Declaration Console X
<terminated> MeetingRoomClientLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86
Checking Digital Certificate...
Server Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Sent

Step 2 Received and N1 Verified


Step 3 Sent

Step 4 Sent
Key Exchange Protocol Complete

Client Login:
Enter username: gunnar
Enter password: student
Login successful

No available times
```

Server view:



```
Problems @ Javadoc Declaration Console X
MeetingRoomServerLab6 (1) [Java Application] C:\Users\gunna\OneDrive\Cybersecurity 2022 Fall\Intro to Java\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_17.0.2.v20
Client Digital Certificate Received
Checking Digital Certificate...
Client Digital Certificate Verified

Key Exchange Protocol Started...
Step 1 Received

Step 2 Sent

Step 3 Received and N2 Verified

Step 4 Received, SessionKey Decrypted

Key Exchange Protocol Complete
Client logging in...

Client logged in, showing available times...
Client Disconnected, restarting...
Meeting Room Server up and running, waiting on port 755...
```