

BBM465: Information Security Laboratory

Programming Assignment-3

Teachers:

- Ahmet Selman Bozkır
- Ali Baran Taşdemir

Students:

- Mert Tazeoğlu (21946606)
- Mehmet Emin Yıldız (21527602)

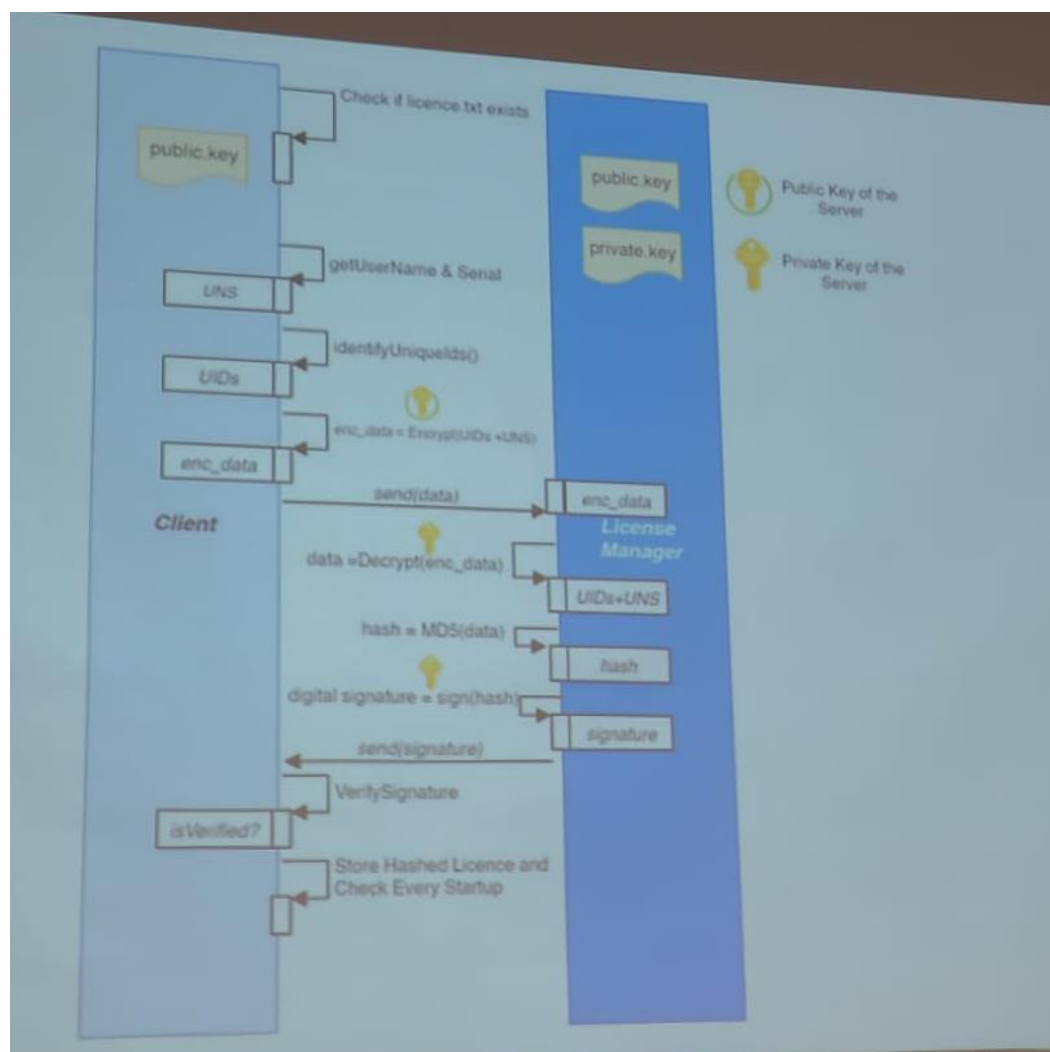
Heading.....	Page
Problem Definition.....	2
Solution Scheme.....	2
Important Implementation Details.....	3
Example Testing (I/O) Scenarios.....	4

Problem Definition

In this assignment, we implemented a licensing framework by utilizing the methods of asymmetric cryptography (RSA encryption algorithm), hashing (MD5 hashing algorithm) and digital signatures.

Solution Scheme

In this framework, there are users (in Client.java) who tries to license app on their PC's and server system (in licenseManager.java) where licensing and signature creation processes are done. We can summarize detailed process exactly like that scheme below:



- Asymmetric encryption is important in order to reduce the vulnerability of “man in the middle” attacks.
- Hashing is important in order to increase critical information (password etc.) security, since there isn't any function such as de-hashing.
- Digital signatures are important in order to provide authentication, data integrity and non-repudiation

Important Implementation Details

Let's take a look at how program works step by step with technical details.

Step-1: Initializing Required Components

- After client starts, first of all computers MAC ID, Disk ID (not general HDD's but current logical disk's serial number (for example C drive like example output that you shared))
- Since we are Windows users, we handled that process with using CMD. We executed some commands on CMD with using Java codes and we selected required part with using some code.
- Also in order to get format that your example output, when returning disk's serial number, we also convert it from hexadecimal to decimal format. We used a special implemented function for that.
- Then program initializes private and public keys. Public key is accessible for both of client and server sides but private is only accessible for server side.

Step-2: Checking License.txt File

After initializing required components, program checks license.txt file and returns a boolean value. It's required since future actions (step-5) will differ depending on whether a license is available or not.

Step-3: Creating License Text

- Program creates a license text with concatenating username, serial number, MAC ID, disk ID and motherboard ID. Program generates license text with adding "\$" characters between of them.
- Username (Admin) and serial number (A068-01A1-1201) are hardly coded with using RSA algorithm and encoded versions are used as a static text in the program. Program takes encrypted versions and uses them after decrypting them with using private key (in the server side).
- Then, at the client side generated license text is encrypted with using RSA algorithm and hashed with using MD5 algorithm. Encrypted version is send to the server in order to generate a digital signature.

Step-4: Creating Digital Signature

In the server side, program decrypts license text and with using license text and "SHA256withRSA" scheme. After creating it, program signs that signature with using private key and returns to the client side. Also program prints encrypted & decrypted license texts and MD5 hash value of license texts.

Step-5: Authorization

Right after signature created in the server returns to the client side, program tries to verify the signature with using public key.

Case 5.1: If License Doesn't Exist and Signature Is Verified: In that case, program signs currently returned signature and stores it's byte array values in order to use it in authorization for next times. (in other words, writes byte values in 'license.txt' file)

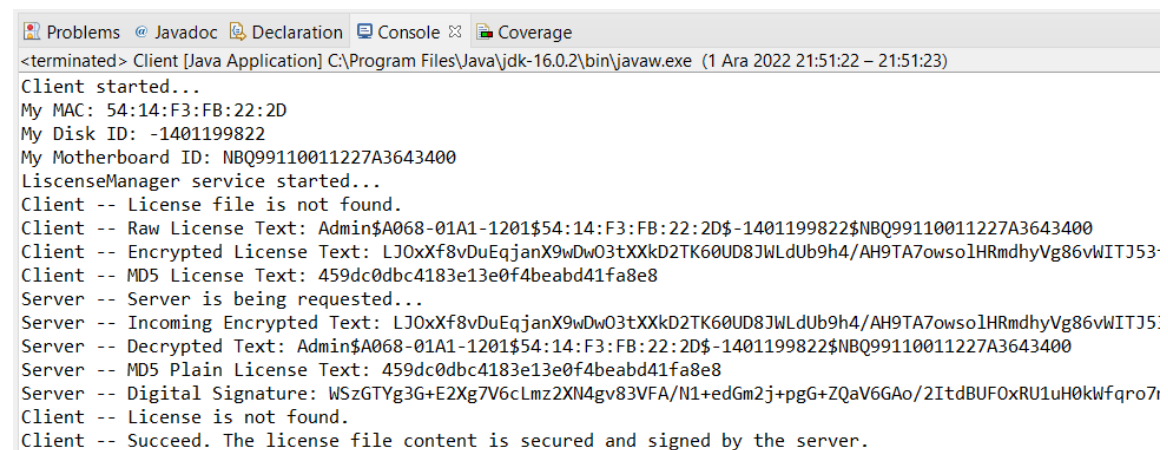
Case 5.2: If Signature Isn't Verified: In that case, program writes a message to console in order to show the error. For example, if you change code and add extra bytes to signature; this case will occur.

Case 5.3: If License Exists: In that case, program reads 'license.txt' file and converts it into a byte array format (that byte array is our existing signature). Than program tries to verify signature and if it verifies it and prints information message about it. Otherwise, it don't verifies it but prints information message about it.

Example Testing (I/O) Scenarios

Case-1: New License Creation With Successful Process

"license.txt" doesn't exist and no attacks were made. Output is:



```
Problems @ Javadoc Declaration Console Coverage
<terminated> Client [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (1 Ara 2022 21:51:22 – 21:51:23)
Client started...
My MAC: 54:14:F3:FB:22:2D
My Disk ID: -1401199822
My Motherboard ID: NBQ99110011227A3643400
LicenseManager service started...
Client -- License file is not found.
Client -- Raw License Text: Admin$A068-01A1-1201$54:14:F3:FB:22:2D$-1401199822$NBQ99110011227A3643400
Client -- Encrypted License Text: LJ0xXf8vDuEqjanX9wDw03tXXkD2TK60UD8JWLdUb9h4/AH9TA7owso1HRmdhyVg86vWITJ5:
Client -- MD5 License Text: 459dc0dbc4183e13e0f4beabd41fa8e8
Server -- Server is being requested...
Server -- Incoming Encrypted Text: LJ0xXf8vDuEqjanX9wDw03tXXkD2TK60UD8JWLdUb9h4/AH9TA7owso1HRmdhyVg86vWITJ5:
Server -- Decrypted Text: Admin$A068-01A1-1201$54:14:F3:FB:22:2D$-1401199822$NBQ99110011227A3643400
Server -- MD5 Plain License Text: 459dc0dbc4183e13e0f4beabd41fa8e8
Server -- Digital Signature: WSzGTYg3G+E2Xg7V6cLmz2XN4gv83VFA/N1+edGm2j+pgG+ZQaV6GAo/2ItdBuFOxRU1uH0kWFqro7i
Client -- License is not found.
Client -- Succeed. The license file content is secured and signed by the server.
```

Case-2: New License Creation With Unsuccessful Process

"license.txt" doesn't exist and an attack was made between client and server file transmission. (For example you edited existing code and added a string after encoded text in the server.) Output is:

```
Problems @ Javadoc Declaration Console Coverage
<terminated> Client [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (1 Ara 2022 21:56:07 – 21:56:08)
Client started...
My MAC: 54:14:F3:FB:22:2D
My Disk ID: -1401199822
My Motherboard ID: NBQ99110011227A3643400
LicenseManager service started...
Client -- License file is found.
Server -- Attack! High probability to change of encrypted text content!
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "java.security.Signature.sign()" because "signature" is null
    at Client.main(Client.java:165)
```

Case-3: Existing License Verification With Successful Process

“license.txt” exists and no attacks were made. Output is:

```
Problems @ Javadoc Declaration Console Coverage
<terminated> Client [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (1 Ara 2022 21:58:30 – 21:58:31)
Client started...
My MAC: 54:14:F3:FB:22:2D
My Disk ID: -1401199822
My Motherboard ID: NBQ99110011227A3643400
LicenseManager service started...
Client -- License file is found.
Client -- Succeed. The license is correct.
```

Case-4: Existing License Verification With Unsuccessful Process

“license.txt” exists but an attack was made. For example contents of the licenst.txt was edited (license.txt includes only integers but this it can be either adding integers or non integers). Output is:

```
Problems @ Javadoc Declaration Console Coverage
<terminated> Client [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (1 Ara 2022 22:00:02 – 22:00:03)
Client started...
My MAC: 54:14:F3:FB:22:2D
My Disk ID: -1401199822
My Motherboard ID: NBQ99110011227A3643400
LicenseManager service started...
Client -- License file is found.
Client -- The license file has been broken!!
```