!

**Hacettepe University**

**Computer Science and Engineering**

**BBM465 - Information Security Lab.**

**Experiment - 2**

Ahmet Faruk BAYRAK - 21426716

Erim Gür - 21426988

18.11.2019

Dr. A. Selman Bozkır

# 1. Software Using Documentation

## 1.1. Software Usage

The program runs on command prompt and the output will be printed out on the console. Also the program will create a license.txt.

Program takes inputs as a command line argument. There are 2 arguments: first one is the name of private key file and the second one is the name of the public key file. These key files has to be given before running. After giving the arguments, you have to just click the run button.

Example of giving an argument is: private.key public.key

And one important thing about testing the software: do not change license.txt from IDE. Because when you open license.txt on IDE, IDE will change the encoding format of txt. After this change, program will not work because bytes of text will be changed since IDE has changed the encoding format. So If you want to test the software by changing license.txt, look for license.txt path on IDE and open it directly on the folder.

## 1.2. Error Messages

Program has only 1 error message: If the existed license.txt is not verified then program is going to print “The license file has been broken!!”
After that the program is going to create a new verified license file.

# 2. Software Design

## 2.1. Description of the Program

### 2.1.1. Problem

In this experiment, the problem that has been expected us to design a simple licensing framework by utilizing the methods of asymmetric cryptography, MD5 and digital signatures. If there is an existed license already, the program has to check whether the license is verified. If the license is not verified then the program is going to create a verified license file. But If the existed license file is verified then program is going to say that “Succeed. The license is correct.”

### 2.1.2. Solution

For the solution, first of all I took the arguments and assign them. First argument is private key file. Since the client should not know the private key, I did not use that argument. I took the second argument which is public key file. And then I created and read public key with the help of public_key_reader() function.

After that the program checks whether license.txt file exist. If license.txt exist then program checks whether license.txt file is verified. To check the verification, program sends information of client to verify_digital_signature() function. This function does the verification with public key. This function returns boolean. If the license is not verified then program creates a new license by the help of prepare_for_creating_license() function. In this function, program takes the information of the client and encrypt it with RSA and public key of License Manager (server). And client sends this encrypted text to server by the help of create_license() function in the LicenseManager class.

First of all, server takes its private key and decrypt the text which is send by client. After decryption, server hash the plaintext with MD5. Finally, server signs this hashed text with “SHA256withRSA” using its private key and creates “license.txt”

After that program comes back to the client side to verify the license. To do this program uses verify_digital_signature() function as we mentioned above.

While this process is running, there are some print statements.

- 1) Running side..: When the client needs to create a new license, program prints “Client started…” and If License Manager (server) side is about to start program prints “LicenseManager service started.”

- 2) Information of the Client: In the client side, client prints its hardware informations such as Mac Address, Disk ID and Motherboard ID.

- 3) Client And Server Side Process Informations: In this part, client and server side prints some informations about the their process. For client, it prints: “Raw Licence Text”, “Encrypted License Text” and “MD5fied Plain License Text”. For server side, it prints: “Incoming Encrypted Text”, “Decrypted Text”, “MD5fied Plain License Text” and “Digital Signature”.

- 4) Success Messages: If license is verified, program prints success message. If license.txt is already exist and verified, program prints "Succeed. The license is correct.”

If license.txt is not already exist but just created and verified, program prints “Succeed. The License file content is secured and signed by the server.”

- 5) Error Messages: If the license verification fails, program prints error message such as "The license file has been broken!!"

Here is the functions that I used in the program:

- **byte []** encrypt(PublicKey key, byte [] plaintext)

First parameter is public key and the second parameter is the plaintext which is desired to encrypt.

This function encrypt the plaintext with public key using RSA algorithm and returns to plaintext.

- **byte []** decrypt(Private key, byte [] ciphertext)

First parameter is private key and the second parameter is the ciphertext which is desired to decrypt.

This function decrypt the ciphertext with private key using RSA algorithm and returns the decrypted ciphertext.

- **PrivateKey** private_key_reader(String filename)

Only parameter is the file name of the private key. This will be given by the argument.

In this function, program reads private key from file and creates it.

- **PublicKey** public_key_reader(String filename)

Only parameter is the file name of the public key. This will be given by the argument.

In this function, program reads public key from file and creates it.

- **boolean** verify_digital_signature(byte [] plaintext, int old_or_new)

First parameter is text of hardware and user specific unique plain text and the second parameter is using for determining whether license.txt was already exist or just created. It will change the print statements. This parameter will be 0 for already existed license text and 1 for just created license text.

This function verify the digital signature with the public key using “SHA256withRSA” and MD5 hashing algorithms.

If the license text is verified it prints success messages but If the license text is not verified it prints error messages.

Finally this functions returns true If the license is verified and returns false If the license is not verified. This informations will be used in later.

- **void** prepare_for_creating_license(String username, String serial_number, String mac_address, String disk_serial_number, String motherboard_ID, String private_key_file)

The first five parameter is the information of the hardware of the client. Last one is the name of private key.

This function encrypt the plaintext and send it to the server.

- **void** create_license(byte [] plaintext, String private_key_file)

First parameter is the encrypted plaintext which is came from client and the second parameter is the name of private key.

This function firstly. Decrypt the upcoming plaintext. After that hash it with MD5 and signs it with “SHA256withRSA” algorithm. Finally it creates “license.txt” and writes this digital signature to in it.

- **String** bytesToHex(byte [] bytes)

This function converts upcoming bytes to hexadecimal. I used this function for printing desired hash output.

### 2.1.3. Algorithm

1) Program takes arguments.

2) Program reads public.key file and creates public key by the help of public_key_reader() function.

3) Program takes the information about client’s hardware.

4) Program checks If license.txt is already exist.

5) If there is a license already, program calls verify_digital_signature() function to check whether license is verified.

6) If the existed license is verified, program prints success message.

7) If the existed license is not verified or there is no license.txt, program prints error message and calls prepare_for_creating_license() function to prepare client side for creating a new verified license.

8) In the prepare_for_creating_license() function, program encrypts the the hardware and user specific unique plain text using RSA algorithm with public key of server and sends encrypted plain text to the server (License Manager) calling create_license() function.

9) On the server side, firstly server reads private.key file and creates private key by the help of private_key_reader() function.

10) After that client decrypts the incoming encrypted plain text using private key of server

11) After decryption, server hash this text using MD5.

12) And finally, server creates digital signature using SHA256withRSA and private key. After creating digital signature, server puts this signature into license.txt and sends it back to the client.

13) On this point, client checks this incoming digital signature and verify it with public key.

14) If this license is verified, program prints success message

15) If this license is not verified, program prints error message and creates a new one.