Hacettepe University Computer Engineering Department Information Security Lab.

Homework 3

Subject: Asymmetric Cryptography, Hashing and Digital Signatures
Due Date: 07.12.2022 23:59
Language: Java
T.A.: Ali Baran TAŞDEMİR

Introduction

In this project, your aim will be the development of a licensing framework by utilizing the methods of asymmetric cryptography, MD5, and digital signatures. Although the methods that take place in symmetric cryptography are strong, they have some shortcomings such as key publication for both *receiver* and *sender*. This could be a hard procedure if these two counterparts live in different cities or they are geographically far away without having secure communication. Moreover, symmetric cryptography is prone to "man in the middle" attacks which serve a potential vulnerability of changing the content of the message on the way to the receiver.

On the other hand, asymmetric cryptography presents a different perspective in order to solve these drawbacks. First, it offers to use a publicly available *public key* for encryption of the message which will be only decrypted by using the *private key.* Note that, these two keys form a *key pair* and, along with the public key, only the receiver holds the private key for tasks of (1) decryption and (2) digital signature creation. The receiver is able to decrypt the encrypted message and it can create a digital signature of content for further authentication and verification purposes.

Implementation Details

As stated, in this experiment, you are expected to develop a licensing framework by utilizing the methods of asymmetric cryptography, MD5, and digital signatures. The following lines will state the requirements of your assignment.

- Let us assume that you are supposed to create a licensing module for a highly cost software project. According to the requirements, you must design a module that will be located in the software itself (the *Client*) and a server system (the *license manager*). In practice, this scheme is implemented via a network system (e.g. client and web server). Further, *clients* do communicate with a server system through web services. In this assignment, instead, you will do a real-world simulation of this scheme by creating 2 main classes in Java namely "Client" and "LicenseManager".
- According to the requirements, the *client* module must first check the existence of the "license.txt" in the application folder. The license.txt file involves a digital signature signed by the *license manager.* If the license.txt exists, then your client module must verify it by using the public key which is given to you at startup. Note that, the client

holds only the public key (”public.key” as the file name) whereas the license manager holds both the ”private.key” and ”public.key” locally.

- If the ”license.txt” does not exist, then this indicates that the software has never been licensed before. In order to license the application your first duty is to collect the following identities:

  1. the username (string)
  2. the serial number (string having format of ####-####-####)
  3. MAC address of the Ethernet device of the host system (string)
  4. Disk serial number (string)
  5. Motherboard serial number (string)

  For username and serial number, you can use static text. In other words, your experiment will not ask the user to input these data by hand. Here, they are only given to provide a perspective. For the other device-specific data, you can use third-party codes. Nonetheless, you can not use JNI or JNA-based *native* services or packages. More precisely, you should use core Java library-based solutions for this step. Once you collect these data, you need to concatenate them by using $ as a separator character. An example is given below for this plain text (”user-serialid-hw specific info” tuple) content.

  Example of a ”user-serialid-hw specific info” tuple:
  abt\$1234-5678-9012\$F0:2F:74:15:F1:CD\$-455469999\$201075710502043

- After forming the hardware and user-specific unique plain text, you must encrypt this content via the public key (the RSA algorithm will be used as the basis). Following this encryption, you need to obtain an encrypted version of this content.

- Upon having encrypted, the encoded message must then be sent to *license manager* for further processes. Once the server (the license manager) has received the encrypted package it must decrypt the content in order to reveal the user and serial number info along with hardware-specific ids.

- Licence Manager can then do several operations such as storing these fields in a database. However, for simplicity, we skip these stages. Rather the license manager must create a hash of the revealed plain text content. For this purpose, you must use MD5 (i.e. Message Digest - an irreversible message digest mechanism) to produce the hash. An example hash content has been presented below:

  Example of a MD5 hash:
  d734cf14e5d7bd538f957cd3503c13f3

- Following the hash production, you must sign the hash content via the private key. For this procedure, you can utilize the ”Signature” class (provided by the Java Security API) through the ”SHA256WithRSA” scheme. This will create the digital signature of the *hash* to be delivered to the *Client*.

- The response (i.e. digital signature) of the license manager must be verified in the *Client* module with the hash provided by *Client*. To do so, you must do the same procedure (i.e. hashing the plain user-serial-hw specific data) on the client side as well. The verification scheme works only in this way. It is important to keep in mind that, the verification must be carried out by using the public key.

- If the verification of the digital signature is signed by the Licence Manager, then you should prompt the user for the success of the operation and store the digital signature in the ”license.txt”. With this step, the whole licensing stage finishes. You must prompt the user with the message of

  ”Succeed. The license file content is secured and signed by the server.”

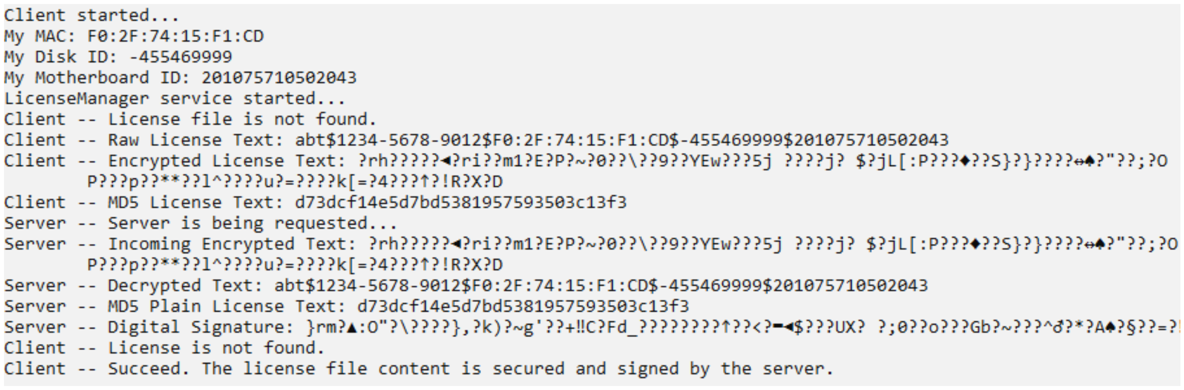
- As you may question, will this whole process be executed every time? The simple answer is NO. Instead, the only thing you must do is to check the existence of the ”license.txt” in the project folder at start-up and verify its content (i.e. build the plain ”user-serialid-hw specific info” tuple at the client side and hash it via MD5) along with the digital signature. If the result of the verification is True then you can inform the user by prompting

  ”Succeed. The license is correct.”

  If the content of the ”license.txt” is destructed by an attacker, the verification fails. In this case, you should warn the user by prompting

  ”The license file has been broken!!”

  and re-execute the licensing process again to obtain a valid digital signature. The output of a working instance of the desired system has been depicted in the figure below:



```
Client started...
My MAC: F0:2F:74:15:F1:CD
My Disk ID: -455469999
My Motherboard ID: 201075710502043
LicenseManager service started...
Client -- License file is not found.
Client -- Raw License Text: abt$1234-5678-9012$F0:2F:74:15:F1:CD$-455469999$201075710502043
Client -- Encrypted License Text: ?rh?????◄?ri??m1?E?P?~?0??\??9??YEw???5j ????j? $?jL[:P???♦??S}?}????↔♠?"??;?O
        P???p??**??l^????u?=????k[=?4???↑?!R?X?D
Client -- MD5 License Text: d73dcf14e5d7bd5381957593503c13f3
Server -- Server is being requested...
Server -- Incoming Encrypted Text: ?rh?????◄?ri??m1?E?P?~?0??\??9??YEw???5j ????j? $?jL[:P???♦??S}?}????↔♠?"??;?O
        P???p??**??l^????u?=????k[=?4???↑?!R?X?D
Server -- Decrypted Text: abt$1234-5678-9012$F0:2F:74:15:F1:CD$-455469999$201075710502043
Server -- MD5 Plain License Text: d73dcf14e5d7bd5381957593503c13f3
Server -- Digital Signature: }rm?▲:O"?\????},?k)?~g'??+‼C?Fd_????????↑??<?▬◄$???UX? ?;0??o???Gb?~???^♂?*?A♠?§??=?
Client -- License is not found.
Client -- Succeed. The license file content is secured and signed by the server.
```

Figure 1: The output of a working instance in case of licensing process

Important Notes

In this chapter, for the sake of clarification, some important notes about your implementation have been described.

- You will be provided with the *public.key* and *private.key* via the Piazza system.
- Your aim will be to design a project including these functionalities.
- Your code environment will be limited to 2 files named "Client.java" and "LicenseManager.java". Here the "Client.java" will be the main "Client" holding the public key owned by the "License Manager". On the other hand, "Licence Manager" can access both of these files.
- For simplicity, you can store the user name and serial id in a file (i.e. "user_serial.txt" for retrieving them quickly. Also, you can store the username and serial id as hard-coded in your Client class.
- Figure 1 exemplifies the system messages during the events. As you can see, the license manager known as *server* side events is listed as "Server –". Similarly, the *Client* side events have been shown as "Client – ". You should pay attention to this format and apply the same scheme for your own implementation.
- You can only use Java's own byte-to-string and string-to-byte helper functions. In other words, you are prohibited to employ third-party JAR files and helper classes.
- You must submit a detailed report to describe what you have done.

Notes

1. In Java, you must use standard crypto API.
2. You should prepare a report involving your approach and details of the implementation you have coded. You must write down the names and ids of your teammates in the report in order to be evaluated correctly.
3. You must submit the homework in groups of two.
4. You should prepare a report that describes your approach to the problem with the details of your implementation. You must write down all group members' names and ids. Reports will be graded too.
5. You can ask your questions about the homework via Piazza. (`www.piazza.com/hacettepe.edu.tr/fall2022/bbm465`)
6. T.A. as himself has the right to partially change this document. However, the modifications will be announced in the Piazza system. In case, it is your obligation to check the Piazza course page periodically.

7. You will submit your work via the submission system.
   (`www.submit.cs.hacettepe.edu.tr`)
   The submission format is given below:
   → <group id.zip>
      → src /Client.java
      → src /LicenseManager.java
      → src /user_serial.txt [Optional]
      → report.pdf

Policy

All work on assignments must be done with your own group unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in pseudo-code) will not be tolerated. In short, turning in someone else's work(from the internet), in whole or in part, as your own will be considered a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.