**CS 5780**

**Project Report**

**Simplified SSL (Secure Socket Layer)**

Group ID = 7

Patel Smitkumar (306587208)

Patel Riddhiben (306612701)

**Project Description**

Conceptually a *Secure Socket Layer* can be thought of as a pair of sockets between a server and a client where communication on the actual network socket is secure.

A clever implementation can actually hide the mess of encryption, decryption and key exchange protocol entirely. As far as the server is concerned, it only wants to know if the client is authorized and receive and send data in clear-text even though the actual bytes on the network are encrypted. As far as the client is concerned, it only wants to know that it is connected to the real server and also wants to exchange data in clear-text even though the physical bytes transmitted are encrypted.

**Server's perspective**

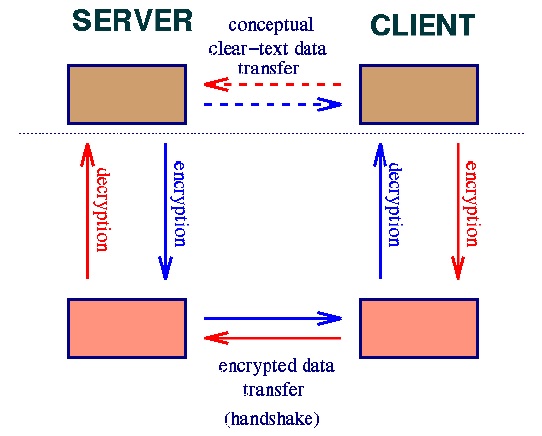
**Authorized Client**

The server stores the user's public key in the user’s profile. The user also stores the server's public key in its server profile. When the user connects to the server, it sends its own identity (user name) encrypted by the server's public key and its company name encrypted by its own private key. The server's users profile also contains information on the user's hash function used in data transfer. This information was communicated off-line.

Only the server can decrypt the identity of the user. The server also has the client's public key, hence it can decrypt its company information and certify the user. Unfortunately, this alone does not prevent a malicious entity to connect to the server by hijacking the encrypted bitstream on the network. While he/she cannot decrypt the transmission, it can be used as a key to open a connection to the server. To circumvent such attacks, data exchange employs a hash function which requires parameters which are agreed between the client and the server but were communicated off-line. If the hashed checksums on the transmitted packets do not agree, the server immediately closes the socket.

**Secure Communication**

Once the client is authorized, communication is performed using the client's unique hash function and a one-time key proposed bythe client.



**Client's Perspective**

**Contacting the Real Server**

The server's public key is used to encode the client's identity. The matching private key is needed to decrypt the client's identity and the proposed one-time key.

**Secure Communication**

In this particular model, connection is initiated by the client and he/she proposes the one-time key.

How to run your program

* **Open Command prompt at workshop directory**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **jar -xvf SSLproject.jar**

**Part A**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.RSA -help**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java -Dprime\_size=500 security.RSA -gen "hello world"**

**Part B**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.Hash**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.Hash 13 2 131 7 hello**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.OneTimeKey**

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.OneTimeKey xyz 123abc**

**Part C**

*Open up two Command Prompts*

*In Command Prompt 1*

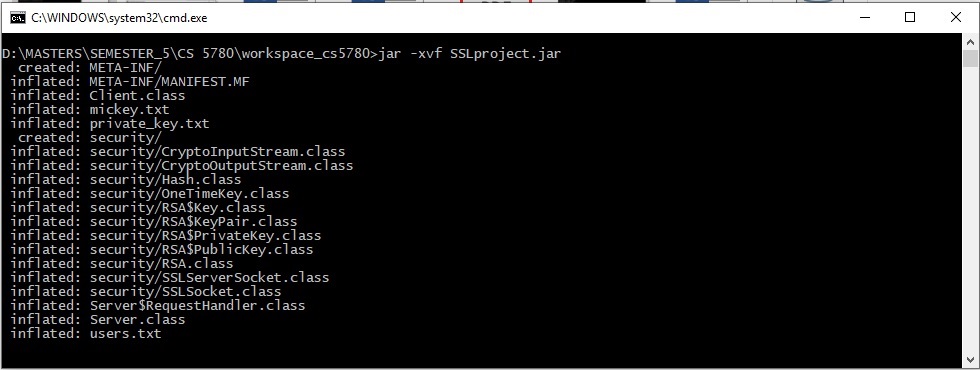
>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java -Dserver.private\_key=private\_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server**

*In Command Prompt 2*

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java Client DESKTOP-2IFDOME3445 mickey < users.txt**

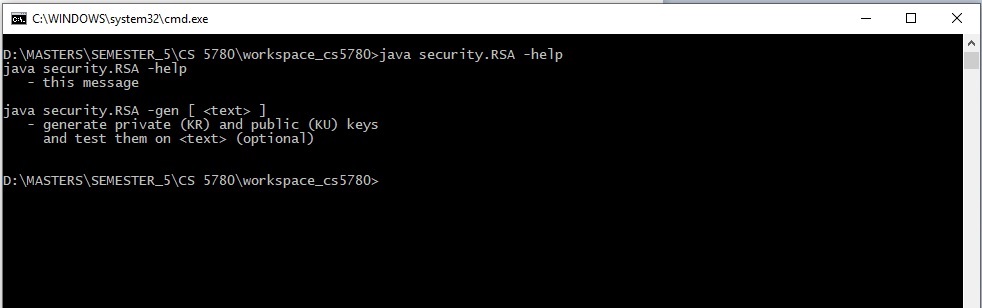
Output

1) D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **jar -xvf SSLproject.jar**

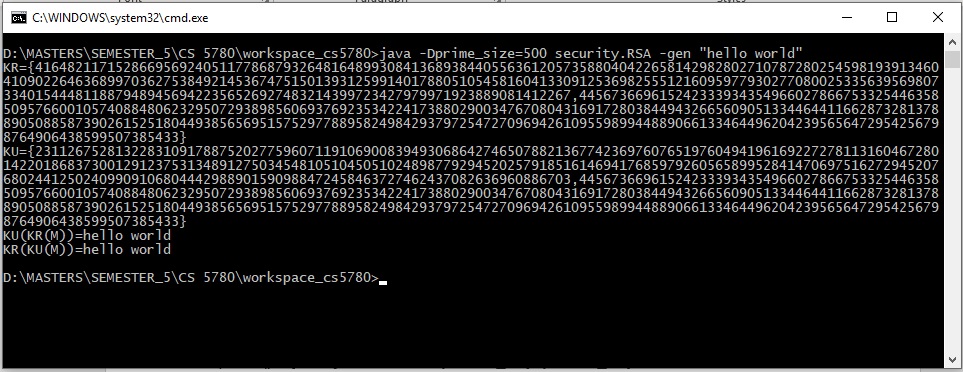


**Part A**

2)D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.RSA -help**

****

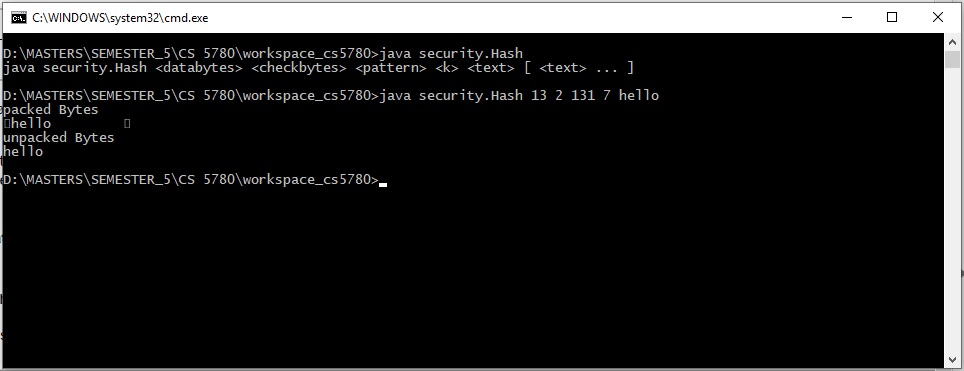
3)D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java -Dprime\_size=500 security.RSA -gen "hello world"**



**Part B**

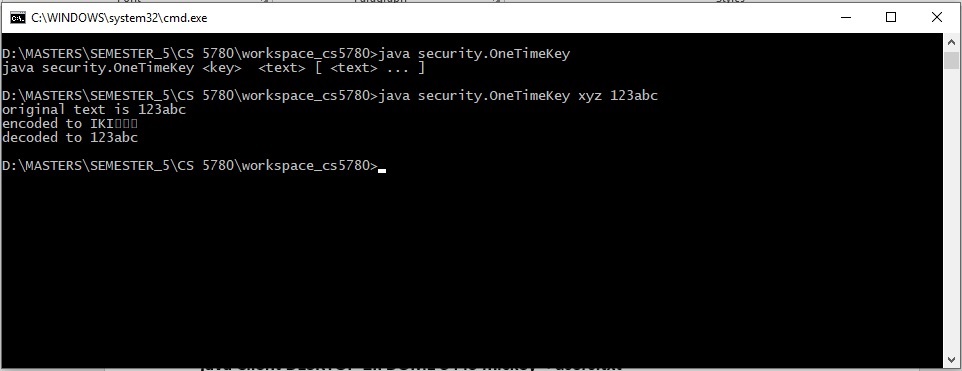
4)D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.Hash**

5)D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.Hash 13 2 131 7 hello**

****

6)D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.OneTimeKey**

7)D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java security.OneTimeKey xyz 123abc**

****

**Part C**

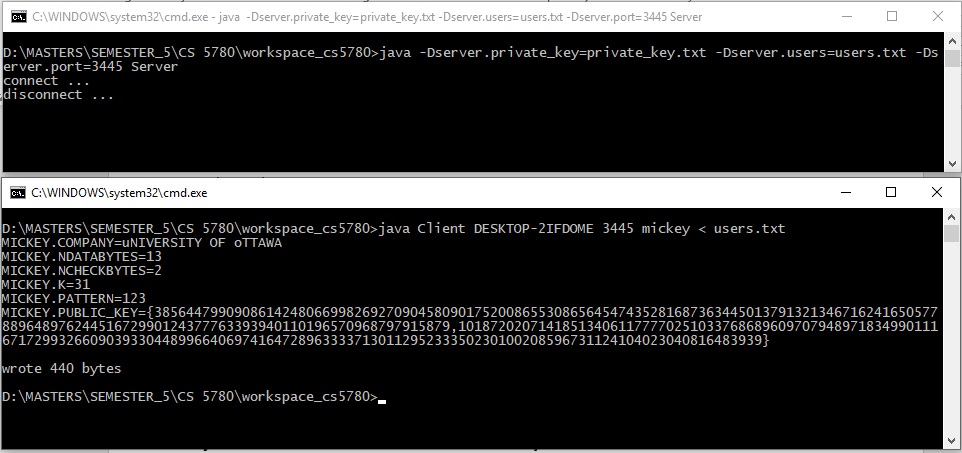
*Open up two Command Prompts*

*In Command Prompt 1*

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java -Dserver.private\_key=private\_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server**

*In Command Prompt 2*

>D:\MASTERS\SEMESTER\_5\CS 5780\workspace\_cs5780> **java Client DESKTOP-2IFDOME3445 mickey < users.txt**



**CONTRIBUTION OF TEAM**

Smitkumar Patel

* Coded for
* RSA
* Hash
* OneTimekey
* SSL server Socket
* Socket
* Documentation

Riddhiben Patel

* Coded for
* Server
* Client
* CryptoInputStream
* CryptoOutputStream
* Documentation