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

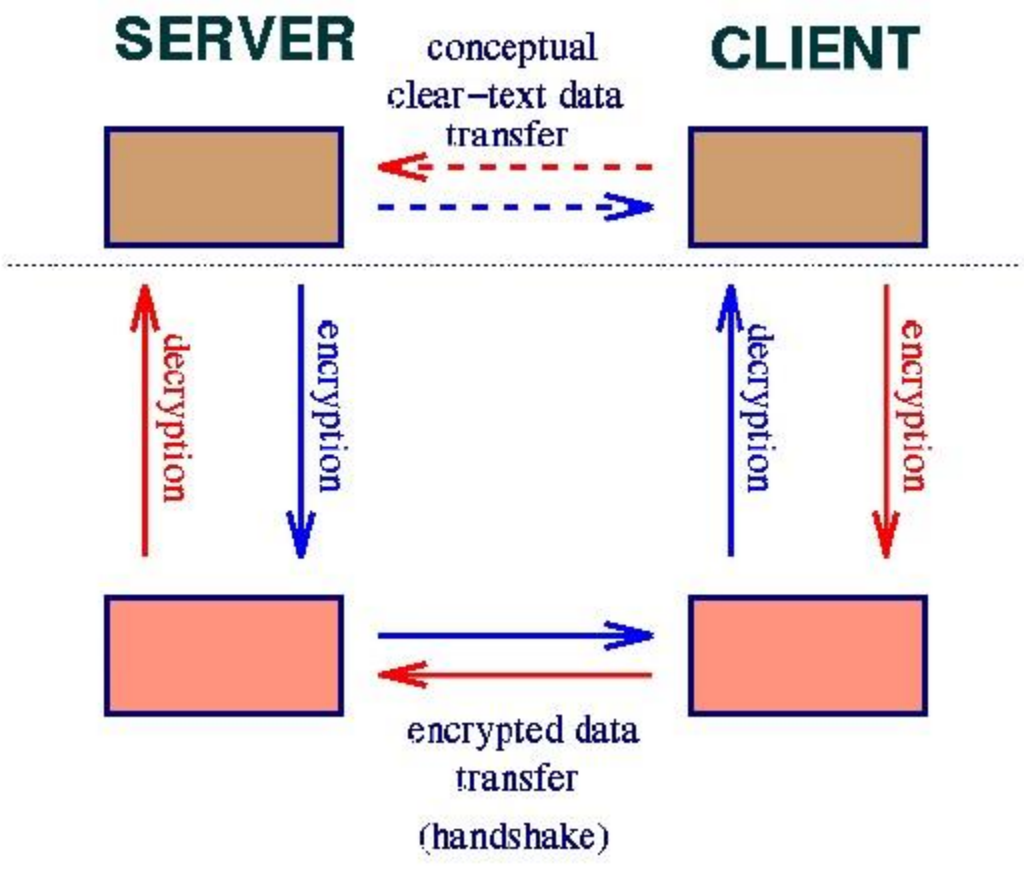
Authorized Client

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

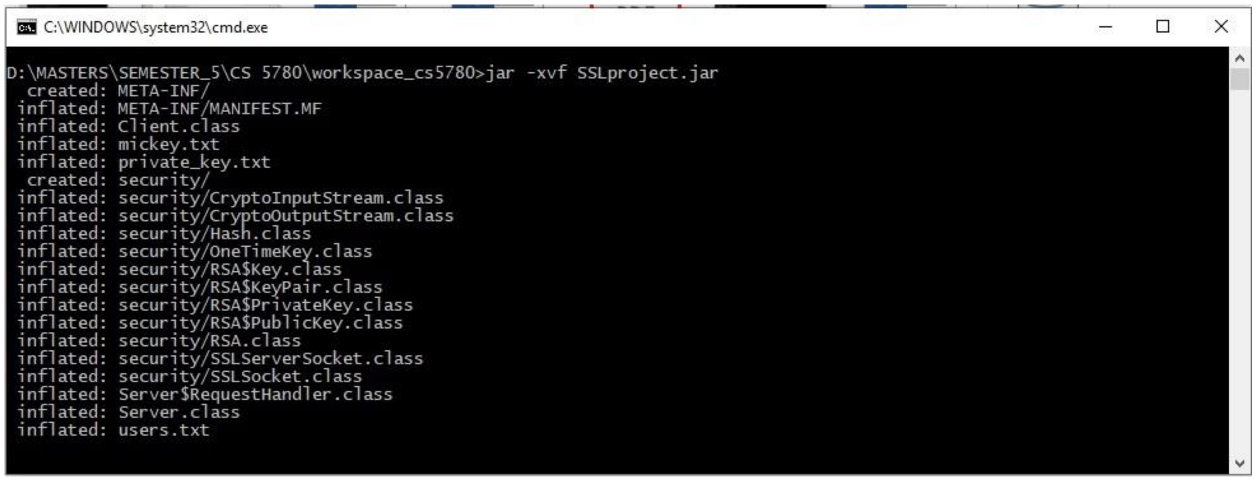
>D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780> **java -Dserver.private_key=private_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server**

In Command Prompt 2

>D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780> **java Client DESKTOP-2IFDOME 3445 mickey < users.txt**

Output

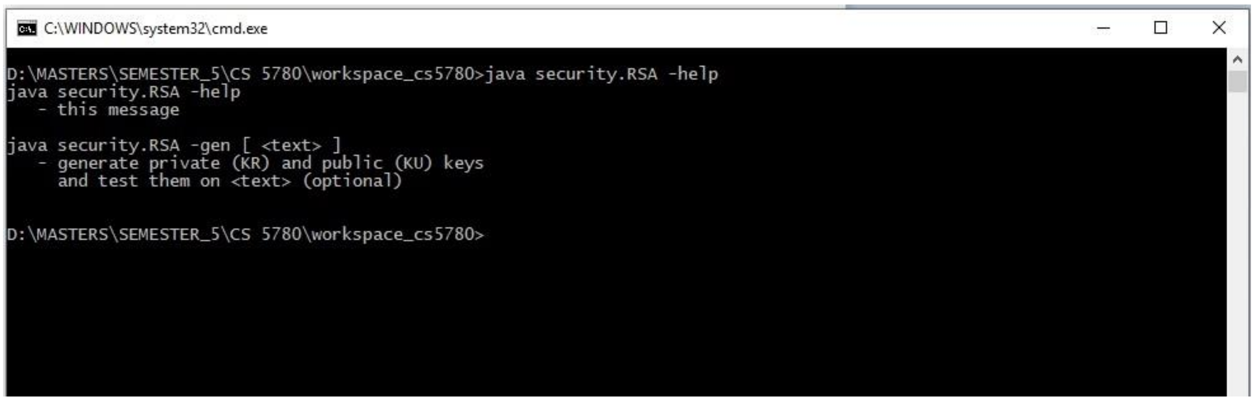
1) D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780> **jar -xvf SSLproject.jar**



```
D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>jar -xvf SSLproject.jar
  created: META-INF/
 inflated: META-INF/MANIFEST.MF
 inflated: Client.class
 inflated: mickey.txt
 inflated: private_key.txt
  created: security/
 inflated: security/CryptoInputStream.class
 inflated: security/CryptoOutputStream.class
 inflated: security/Hash.class
 inflated: security/OneTimeKey.class
 inflated: security/RSA$Key.class
 inflated: security/RSA$KeyPair.class
 inflated: security/RSA$PrivateKey.class
 inflated: security/RSA$PublicKey.class
 inflated: security/RSA.class
 inflated: security/SSLServerSocket.class
 inflated: security/SSLSocket.class
 inflated: Server$RequestHandler.class
 inflated: Server.class
 inflated: users.txt
```

<u>Part A</u>

2)D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780> **java security.RSA -help**

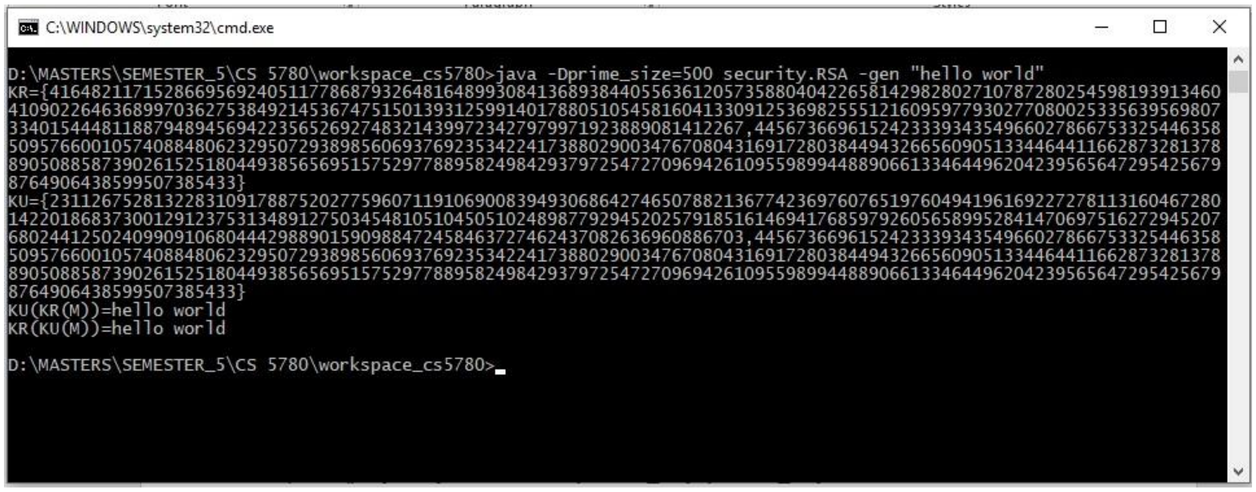


```
D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java security.RSA -help
java security.RSA -help
    - this message

java security.RSA -gen [ <text> ]
    - generate private (KR) and public (KU) keys
      and test them on <text> (optional)


D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>
```

3)D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780> **java -Dprime_size=500 security.RSA -gen "hello world"**



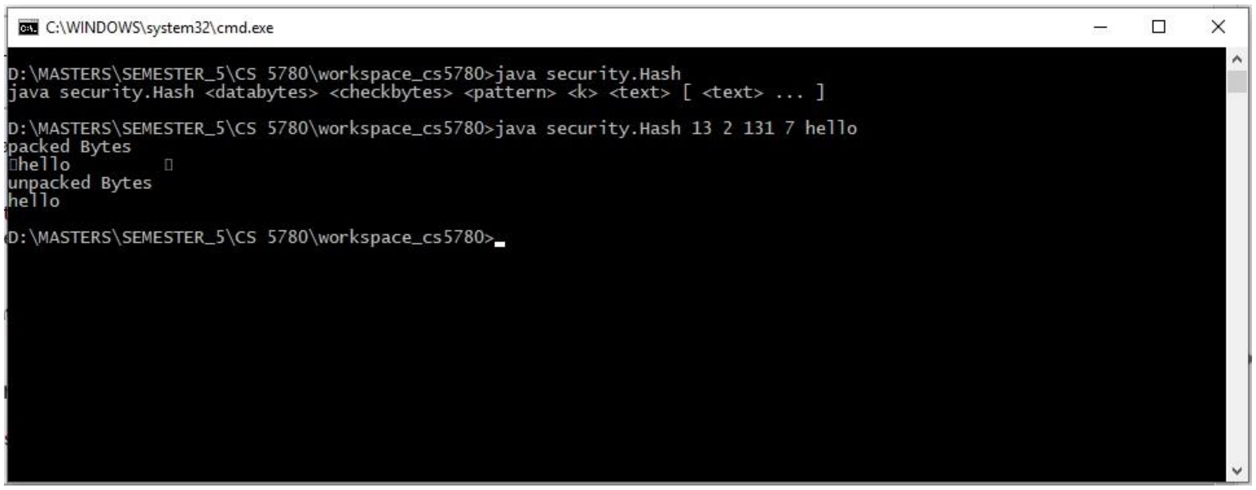
```
D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java -Dprime_size=500 security.RSA -gen "hello world"
KR={41648211715286695692405117786879326481648993084136893844055636120573588040422658142982802710787280254598193913460
410902264636899703627538492145367475150139312599140178805105458160413309125369825551216095977930277080025335639569807
334015444811887948945694223565269274832143997234279799719238890814122 67,445673669615242333934354966027866753325446358
509576600105740884806232950729389856069376923534224173880290034767080431691728038449432665609051334464411662873281378
890508858739026152518044938565695157529778895824984293797254727096942610955989944889066133464496204239565647295425679
87649064385995073854 33}
KU={23112675281322831091788752027759607119106900839493068642746507882136774236976076519760494196169227278113160467280
142201868373001291237531348912750345481051045051024898779294520257918516146941768597926056589952841470697516272945207
680244125024099091068044429889015909884724584637274624370826369608867 03,445673669615242333934354966027866753325446358
509576600105740884806232950729389856069376923534224173880290034767080431691728038449432665609051334464411662873281378
890508858739026152518044938565695157529778895824984293797254727096942610955989944889066133464496204239565647295425679
87649064385995073854 33}
KU(KR(M))=hello world
KR(KU(M))=hello world

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>
```

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**



C:\WINDOWS\system32\cmd.exe

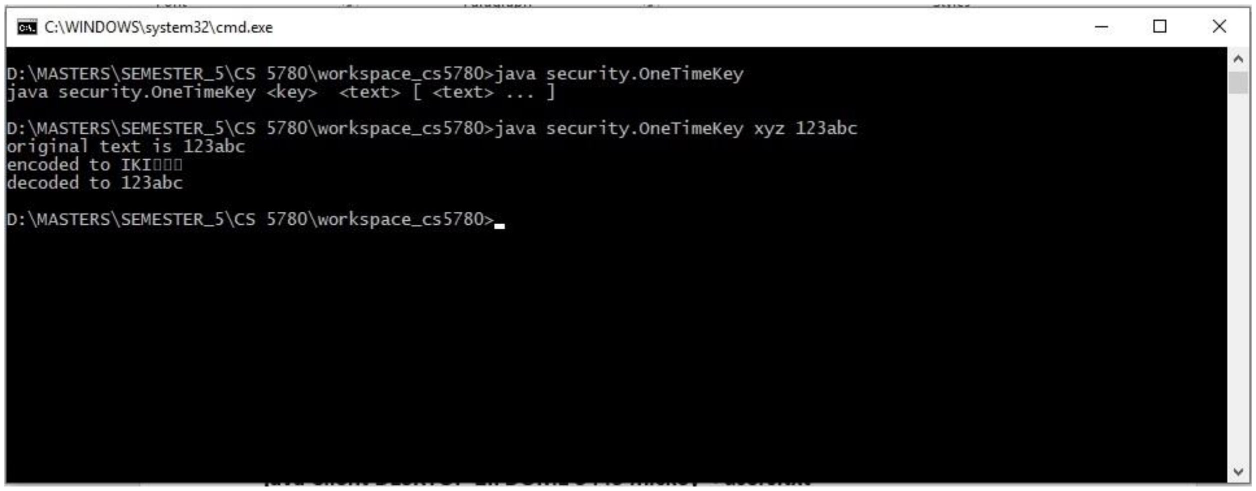
```
D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java security.Hash
java security.Hash <databytes> <checkbytes> <pattern> <k> <text> [ <text> ... ]

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java security.Hash 13 2 131 7 hello
packed Bytes
□hello        □
unpacked Bytes
hello

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>_
```

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**



C:\WINDOWS\system32\cmd.exe

```
D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java security.OneTimeKey
java security.OneTimeKey <key>  <text> [ <text> ... ]

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java security.OneTimeKey xyz 123abc
original text is 123abc
encoded to IKI□□□
decoded to 123abc

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>_
```

<u>Part C</u>

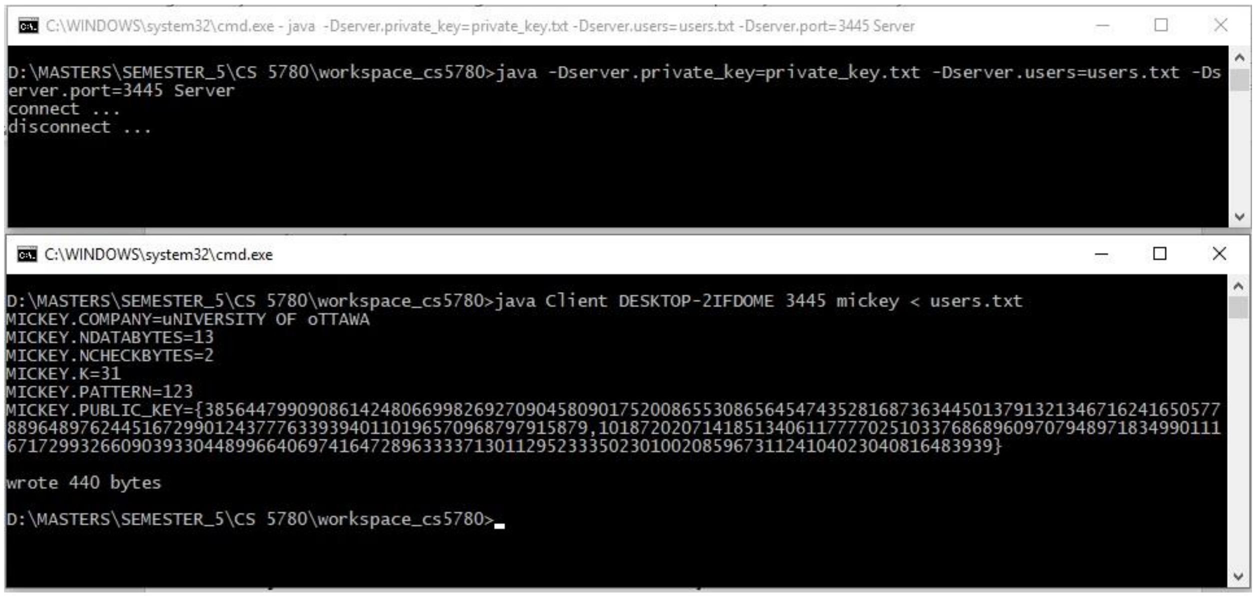
Open up two Command Prompts

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-2IFDOME 3445 mickey < users.txt**



```
C:\WINDOWS\system32\cmd.exe - java  -Dserver.private_key=private_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java -Dserver.private_key=private_key.txt -Dserver.users=users.txt -Ds
erver.port=3445 Server
connect ...
disconnect ...
```

```
C:\WINDOWS\system32\cmd.exe

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>java Client DESKTOP-2IFDOME 3445 mickey < users.txt
MICKEY.COMPANY=uNIVERSITY OF oTTAWA
MICKEY.NDATABYTES=13
MICKEY.NCHECKBYTES=2
MICKEY.K=31
MICKEY.PATTERN=123
MICKEY.PUBLIC_KEY={38564479909086142480669982692709045809017520086553086564547435281687363445013791321346716241650577
88964897624451672990124377763393940110196570968797915879,1018720207141851340611777702510337686896097079489718349901 11
67172993266090393304489966406974164728963333713011295233350230100208596731124104023040816483939}

wrote 440 bytes

D:\MASTERS\SEMESTER_5\CS 5780\workspace_cs5780>
```

CONTRIBUTION OF TEAM

Smitkumar Patel

- ➢ Coded for
  - RSA
  - Hash
  - OneTimekey
  - SSL server Socket
  - Socket
  - Documentation

Riddhiben Patel

- ➢ Coded for
  - Server
  - Client
  - CryptoInputStream
  - CryptoOutputStream
  - Documentation