CS 5780 Project Report

Project 1 -Simplified SSL (Secure Socket Layer)

Group – 6

Anirudha Bhaktharahalli Subramanya(403331952)
Shreevathsa Sridhar(403284957)
Yash Divate(403262909)

Project Description:

This project provides a simplified framework for secure server-client communication, encapsulating the core concepts of cryptographic operations, secure sockets, and encrypted data transmission. The design incorporates custom implementations of SSL/TLS protocols, RSA encryption/decryption, one-time key XOR encryption, and basic hashing for ensuring the integrity and confidentiality of the data exchanged between the server and clients. Here's a brief overview of each component's role within the project:

RSA Encryption:

The RSA classes are central to the project's security features, enabling public-key encryption and decryption. They allow secure sharing of encryption keys over an insecure network, ensuring that data can be encrypted by the public key and only decrypted by the corresponding private key. This mechanism is crucial for establishing a secure initial handshake between the client and server, exchanging session keys, and verifying identities.

Hash Function:

The custom Hash class provides data integrity verification, ensuring that data transmitted over the network has not been tampered with. By using a hash function that processes the data alongside a pattern and additional parameters, both the client and the server can generate and verify hashes of the data they send and receive, adding an extra layer of security to their communication.

One-Time Key XOR Encryption:

The OneTimeKey class facilitates the generation of one-time keys and the encryption/decryption of data using the XOR operation. This technique offers a simple yet effective method for encrypting data with a key that is as long as the message, ensuring that the encryption is secure as long as the key remains secret and is used only once.

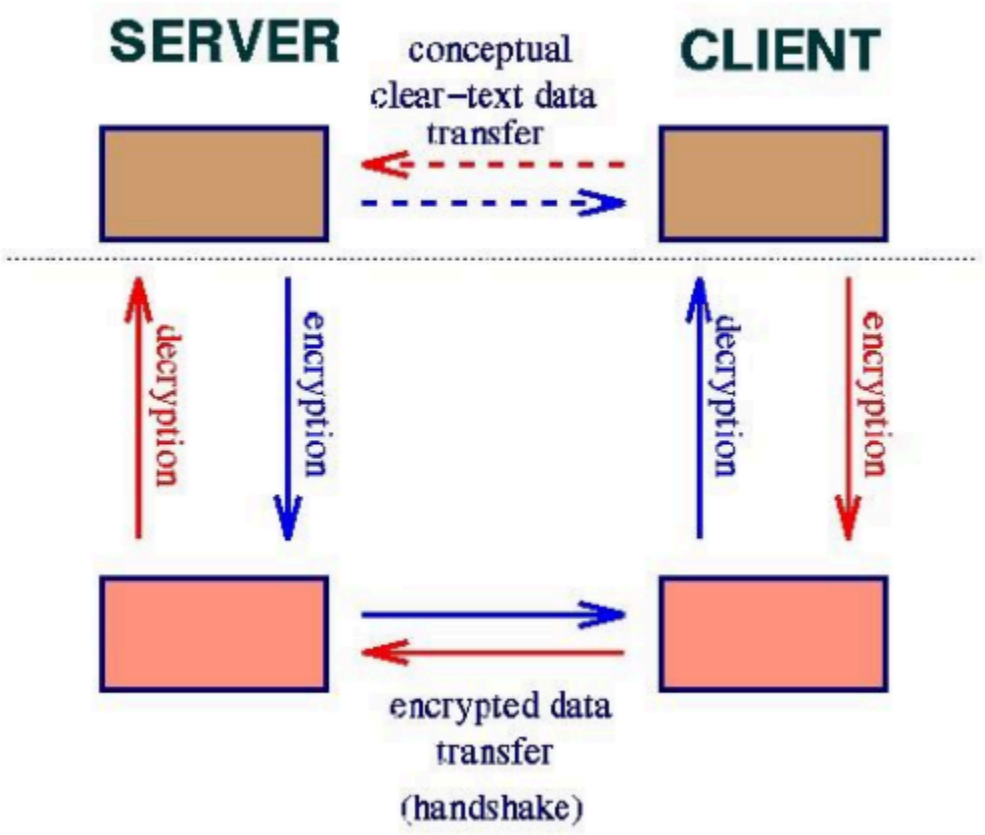
Secure Sockets (SSLSocket and SSLServerSocket):

The custom SSL sockets (SSLSocket and SSLServerSocket) simulate SSL/TLS functionality, providing a secure channel for data transmission over the network. These classes wrap standard Java sockets, adding layers of encryption (via RSA and one-time keys) and data integrity (via hashing) to the data being transmitted. They handle the encryption and decryption of data streams transparently, allowing the client and server to communicate securely without worrying about the underlying cryptographic operations.

Server:

The Server class sets up a secure server that listens for incoming SSL connections from clients. It performs a handshake to securely exchange cryptographic parameters and establishes an

encrypted communication channel with each client. The server handles multiple client connections concurrently, processing and responding to client requests securely.



Client:

The Client class demonstrates how a client can establish a secure connection with the server, performing a handshake to exchange cryptographic parameters and establish an encrypted communication channel. The client can then securely send and receive data to and from the server.

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 onetime key.

Secure Communication

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

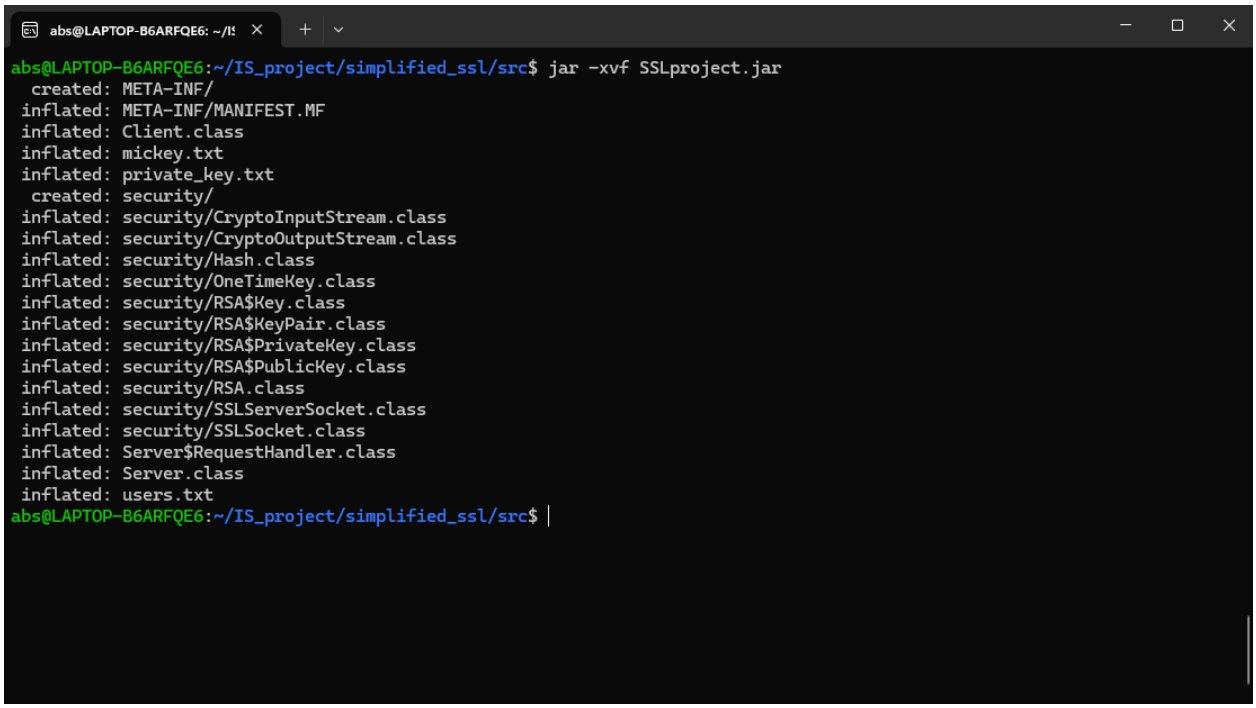
Conclusion:

In conclusion, this project has successfully implemented a secure communication system that mirrors the complexities and safeguards of real-world secure messaging applications. By integrating RSA encryption for secure key exchange, employing hash functions for message integrity, utilizing one-time keys for message encryption, and setting up secure SSL sockets for encrypted data transmission, we've created a comprehensive framework that ensures confidentiality, integrity, and authentication in communications between a server and its clients.

How To Run Program and outputs:

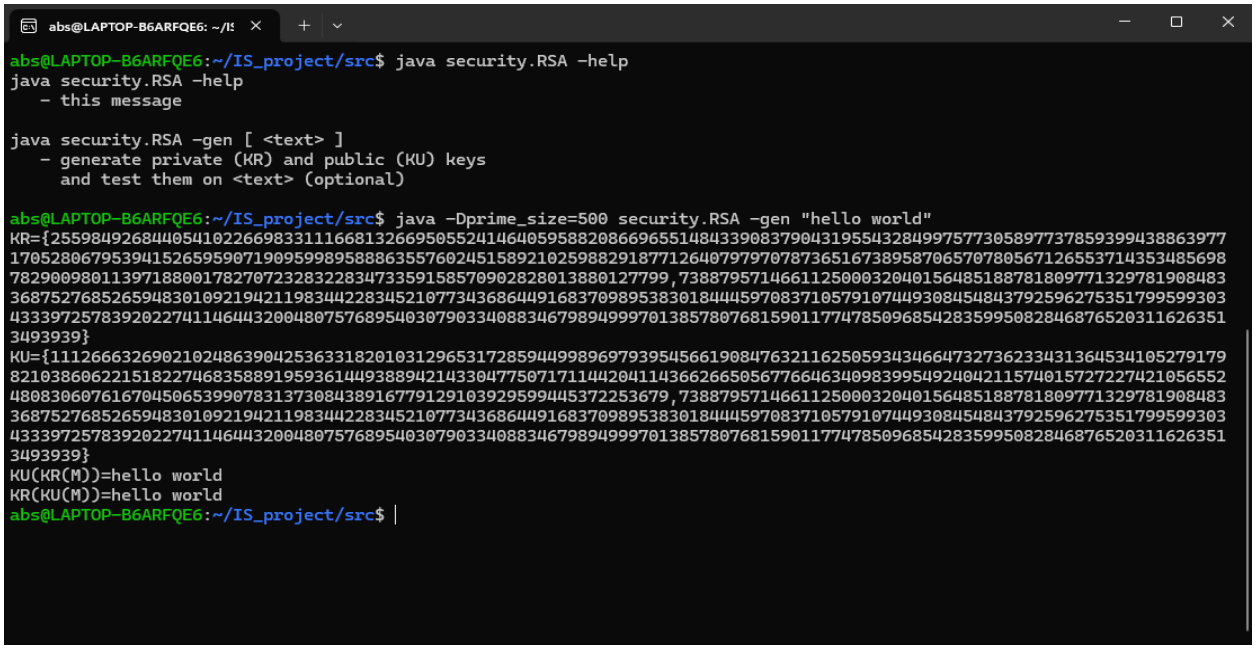
Open Command prompt at workshop directory

1. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ jar -xvf SSLproject.jar



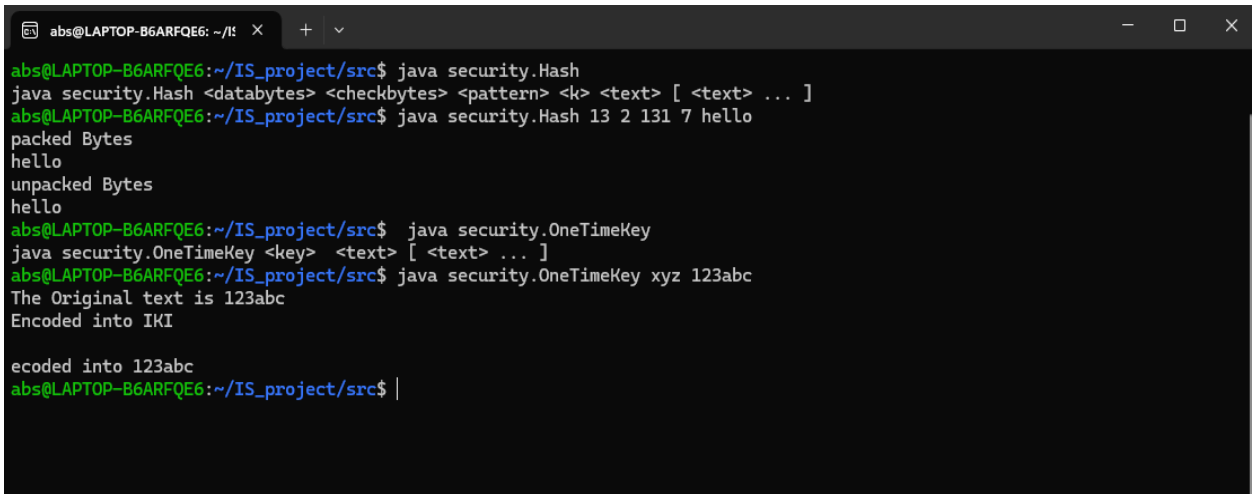
Part A

1. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.RSA -help
2. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java -Dprime_size=500 security.RSA -gen "hello world"



Part B

1. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.Hash
2. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.Hash 13 2 131 7 hello
3. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.OneTimeKey
4. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.OneTimeKey xyz 123abc



Part C

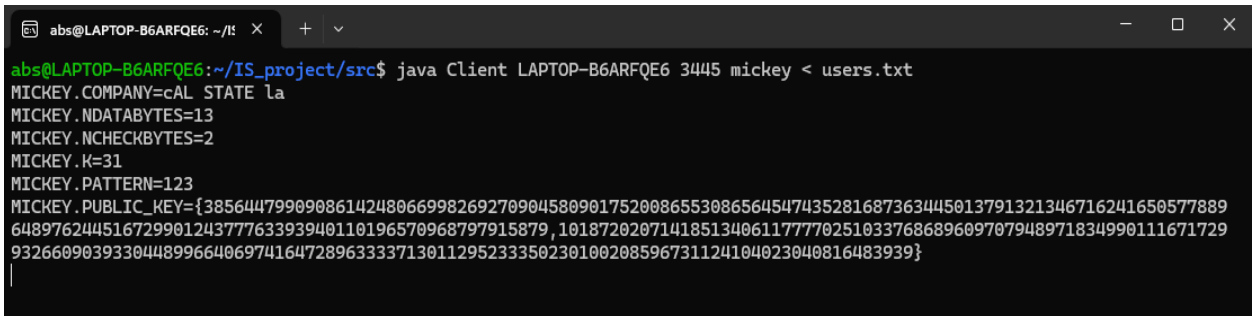
Open up two Command Prompts

In Command Prompt 1

1. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java -Dserver.private_key=private_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server

In Command Prompt 2

2. abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java Client DESKTOP2IFDOME 3445 mickey < users.txt


abs@LAPTOP-B6ARFQE6: ~/IS
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java Client LAPTOP-B6ARFQE6 3445 mickey < users.txt
MICKEY.COMPANY=cAL STATE la
MICKEY.NDATABYTES=13
MICKEY.NCHECKBYTES=2
MICKEY.K=31
MICKEY.PATTERN=123
MICKEY.PUBLIC_KEY={385644799090861424806699826927090458090175200865530865645474352816873634450137913213467162416505778896489762445167299012437776339394011019657096879791587 9,101872020714185134061177770251033768689609707948971834990111671729932660903933044899664069741647289633337130112952333502301002085967311241040230408164839 39}

Member Contributions:

Anirudha Bhatharahalli Subramanya – 33.3%

- RSA Key Generation and Ciphering
- Testing and Documenting

Shreevathsa Sridhar – 33.3%

- Hash Function and One-Time Key Encryption
- Testing and Documenting

Yash Divate – 33.3%

- SSL Layer Implementation and Application Demonstration
- Testing and Documenting