

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.

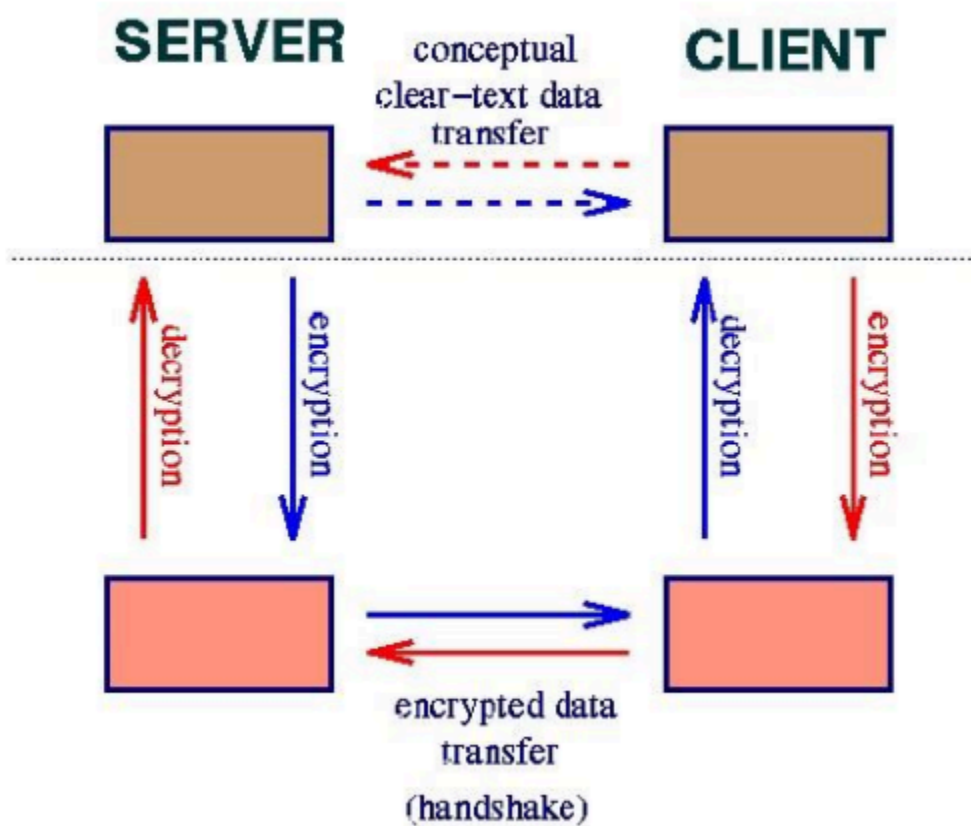
Secure Sockets (SSLSocket and SSLServerSocket):

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.

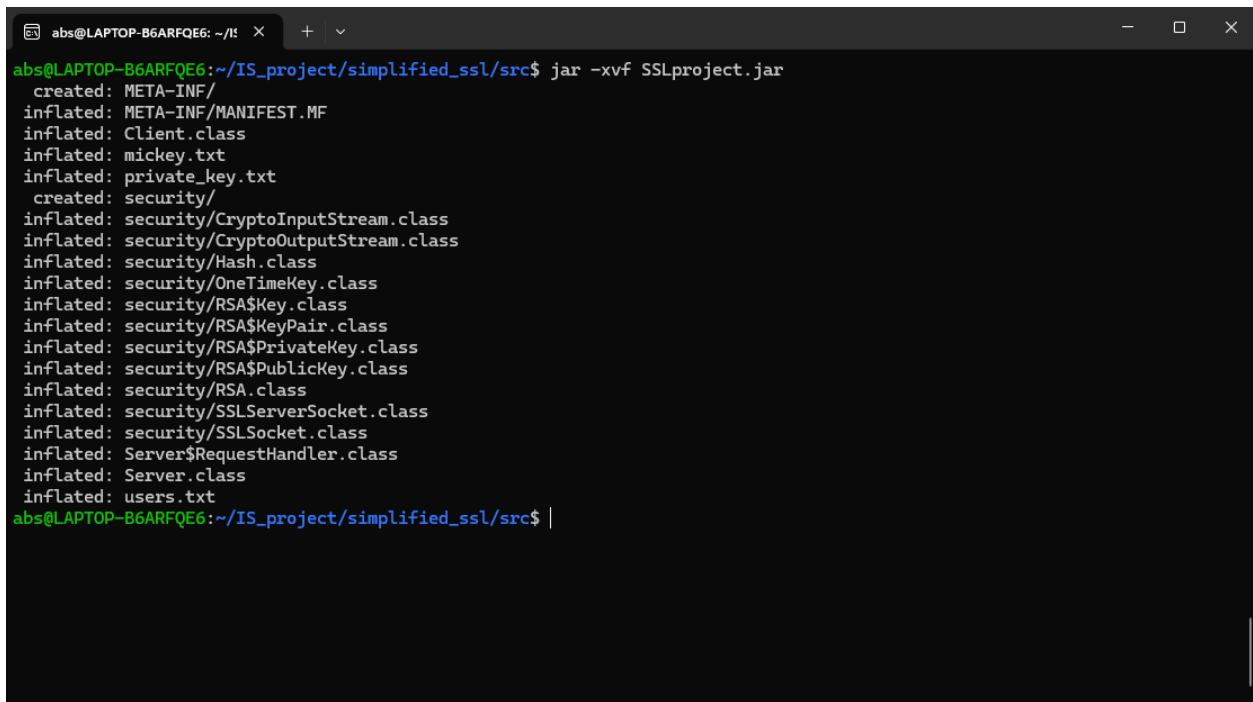
Conclusion:

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`



```
abs@LAPTOP-B6ARFQE6: ~/IS X + v
abs@LAPTOP-B6ARFQE6:~/IS_project/simplified_ssl/src$ 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$RequestHandler.class
inflated: Server.class
inflated: users.txt
abs@LAPTOP-B6ARFQE6:~/IS_project/simplified_ssl/src$ |
```

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"`

```
abs@LAPTOP-B6ARFQE6: ~/! X + v
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ 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)

abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java -Dprime_size=500 security.RSA -gen "hello world"
KR={25598492684405410226698331116681326695055241464059588208669655148433908379043195543284997577305897737859399438863977
170528067953941526595907190959989588863557602451589210259882918771264079797078736516738958706570780567126553714353485698
78290098011397188001782707232832283473359158570902828013880127799, 738879571466112500032040156485188781809771329781908483
368752768526594830109219421198344228345210773436864491683709895383018444597083710579107449308454843792596275351799599303
433397257839202274114644320048075768954030790334088346798949997013857807681590117747850968542835995082846876520311626351
3493939}
KU={11126663269021024863904253633182010312965317285944998969793954566190847632116250593434664732736233431364534105279179
821038606221518227468358891959361449388942143304775071711442041143662665056776646340983995492404211574015727227421056552
48083060761670450653990783137308438916779129103929599445372253679, 738879571466112500032040156485188781809771329781908483
368752768526594830109219421198344228345210773436864491683709895383018444597083710579107449308454843792596275351799599303
433397257839202274114644320048075768954030790334088346798949997013857807681590117747850968542835995082846876520311626351
3493939}
KU(KR(M))=hello world
KR(KU(M))=hello world
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ |
```

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

```
abs@LAPTOP-B6ARFQE6: ~/! X + v
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.Hash
java security.Hash <databytes> <checkbytes> <pattern> <k> <text> [ <text> ... ]
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.Hash 13 2 131 7 hello
packed Bytes
hello
unpacked Bytes
hello
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.OneTimeKey
java security.OneTimeKey <key> <text> [ <text> ... ]
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java security.OneTimeKey xyz 123abc
The Original text is 123abc
Encoded into IKI

ecoded into 123abc
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ |
```

Part C

Open up two Command Prompts

In Command Prompt 1

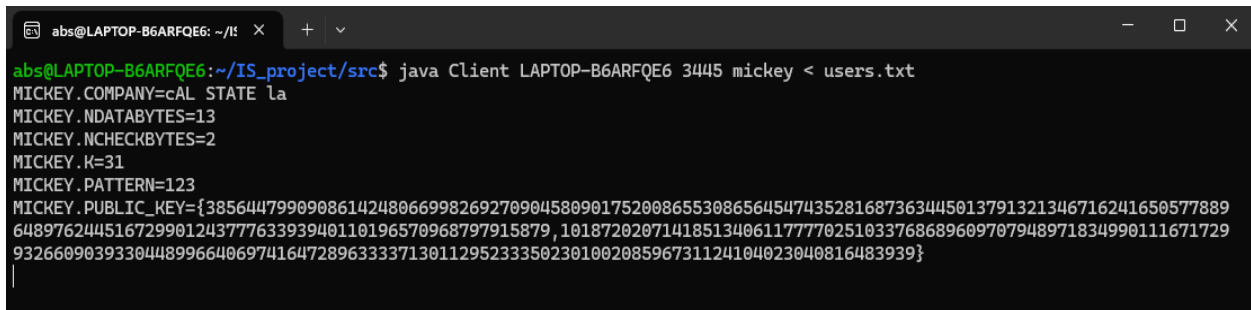
1. `abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java -Dserver.private_key=private_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server`



```
abs@LAPTOP-B6ARFQE6: ~/I  X + v
abs@LAPTOP-B6ARFQE6:~/IS_project/src$ java -Dserver.private_key=private_key.txt -Dserver.users=users.txt -Dserver.port=3445 Server
connect ...
```

In Command Prompt 2

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



```
abs@LAPTOP-B6ARFQE6: ~/I  X + v
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={38564479909086142480669982692709045809017520086553086564547435281687363445013791321346716241650577889
64897624451672990124377763393940110196570968797915879,101872020714185134061177770251033768689609707948971834990111671729
93266090393304489966406974164728963333713011295233350230100208596731124104023040816483939}
```

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