***Problem Statement 05***

***Cryptography Simulation with mbedTLS / OpenSSL Library Usage and User Interaction***

***Intel Unnati Industrial Training Program 2024***

***Navin Kumar L***

[***navinkumar.l2021@vitstudent.ac.in***](mailto:navinkumar.l2021@vitstudent.ac.in)

***Dr. Shridevi S***

[***shridevi.s@vit.ac.in***](mailto:shridevi.s@vit.ac.in)

***Vellore Institute of Technology, Chennai***

***Problem Statement:-***

Cryptography Simulation with mbedTLS/OpenSSL Library Usage and User Interaction

***Objective:-***

The objective is to develop an interactive tool that simulates cryptographic techniques using OpenSSL, allowing users to experiment with various encryption algorithms and understand key management concepts.

***Project Overview:-***

This project implements a secure custom communication protocol using OpenSSL and C/C++. It demonstrates practical applications of cryptographic concepts including digital certificates, RSA key pairs, symmetric encryption, HMAC, and SIGMA key exchange.

***Overview of Cryptography:-***

The objective is to develop an interactive tool that simulates cryptographic techniques using OpenSSL, allowing users to experiment with various encryption algorithms and understand key management concepts.

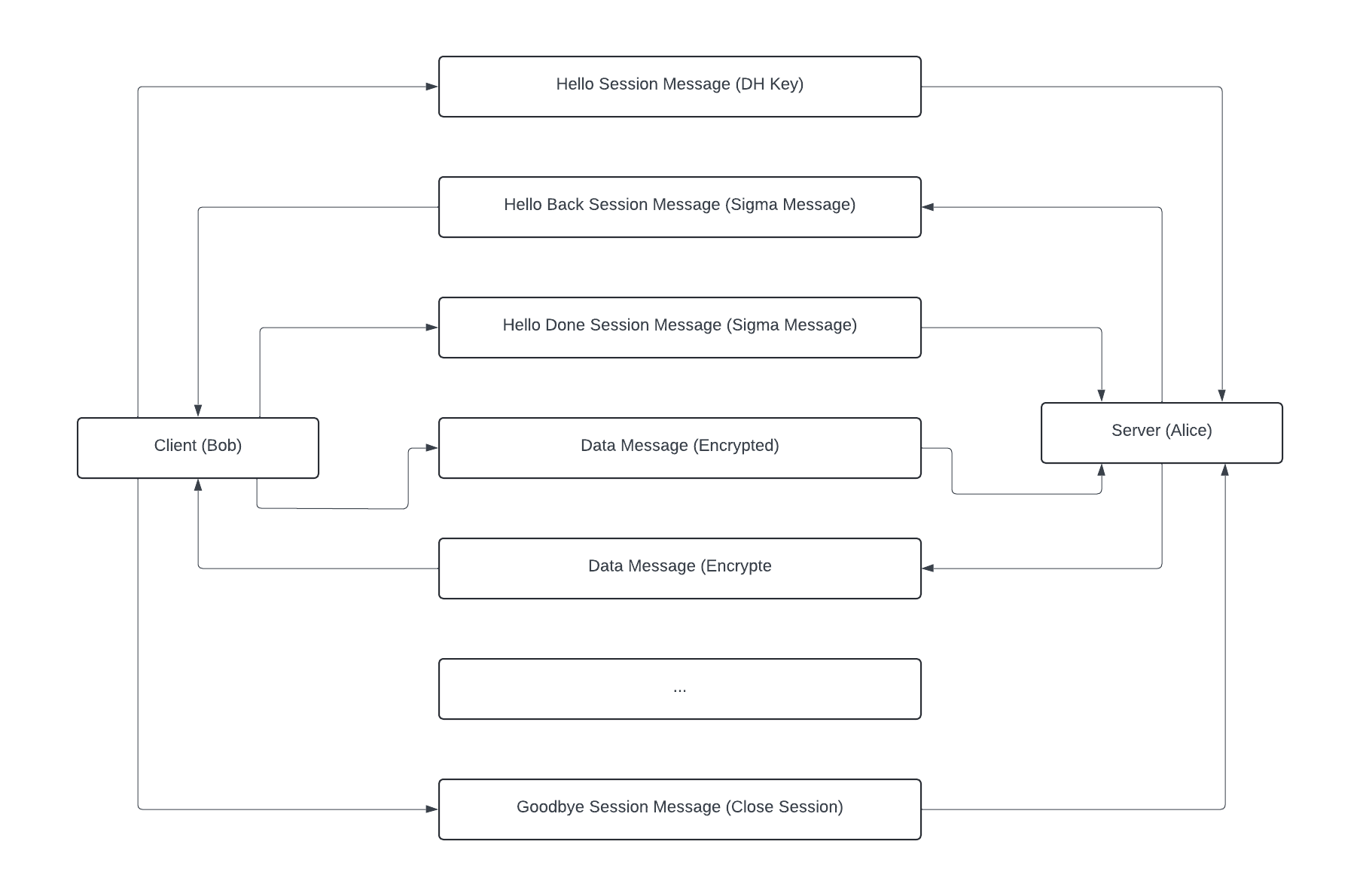
***Challenges in Cryptography:-***

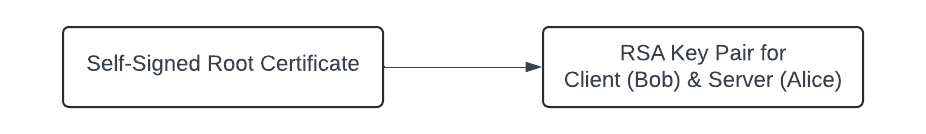
* ***Algorithm Implementation:*** Accurate implementation of RSA, AES, DES, and SHA algorithms using OpenSSL.
* ***Key Generation and Management:*** Simulating key generation, exchange, and management processes.
* One of the main challenges was integrating OpenSSL with the existing codebase and ensuring all cryptographic operations were correctly implemented. We overcame this by careful study of the OpenSSL documentation and extensive testing of each cryptographic function.
* Develop an interactive cryptography simulation using C/C++ and OpenSSL, focusing on securing a custom client-server protocol.

***Key Requirements:-***

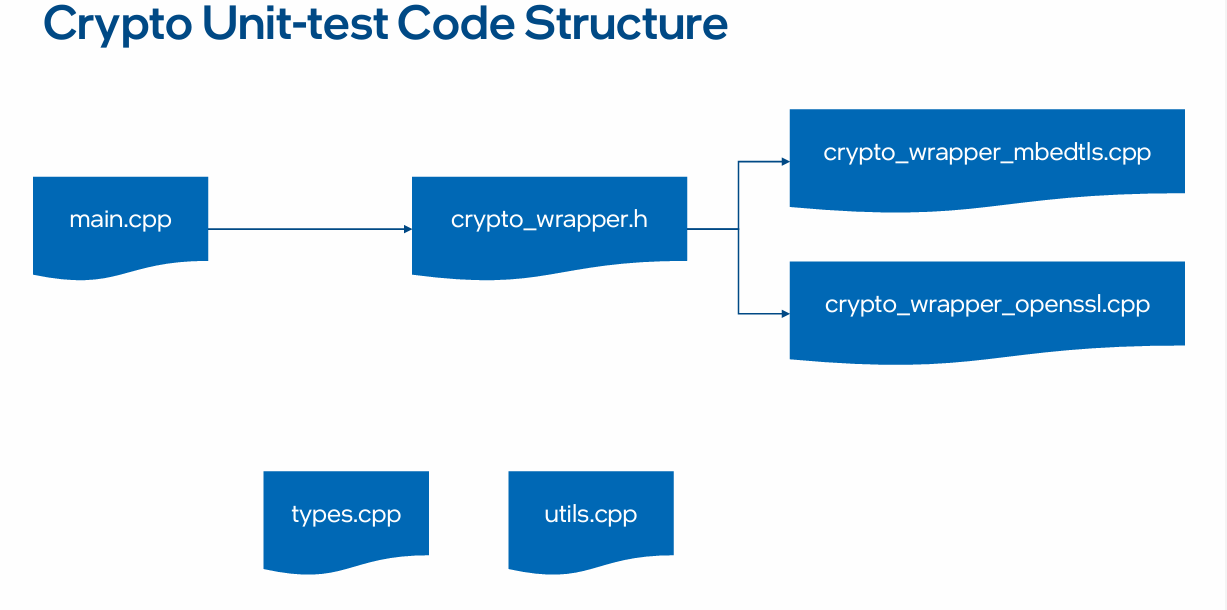
* Implementation of various cryptographic algorithms
* Creation of digital certificates and key pairs
* Secure key exchange and channel establishment
* Message encryption and integrity protection

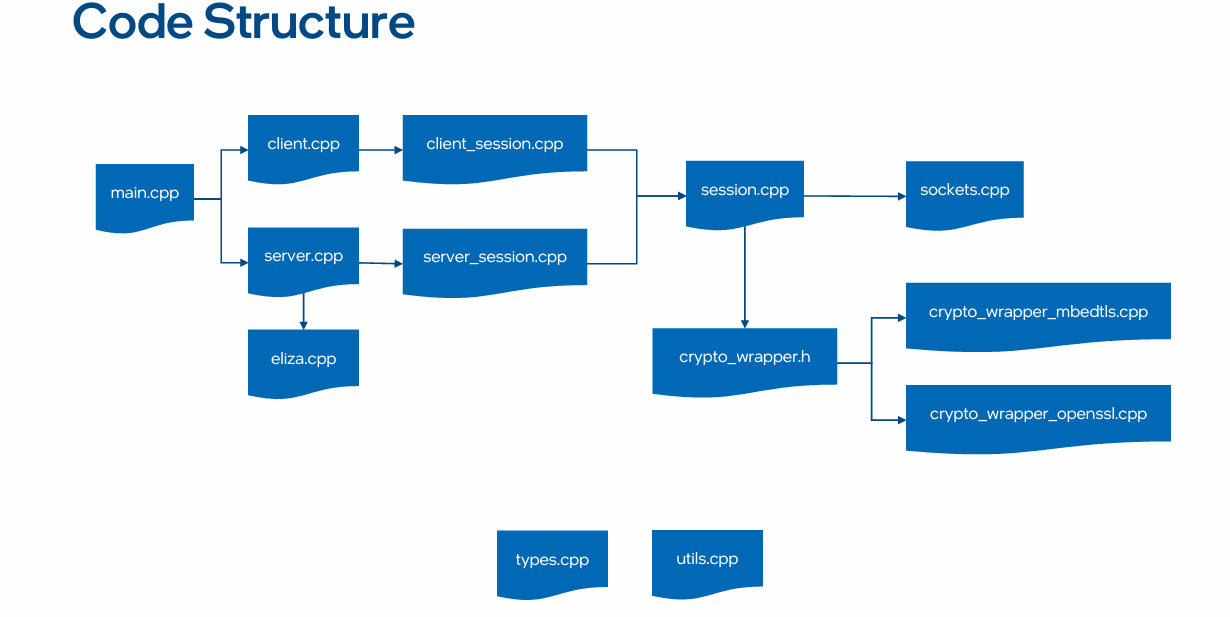
***Process Flow:-***





***Architecture Diagram:-***





## ***Technologies Used:-***

* ***OpenSSL:*** For cryptographic functions and secure communication.
* ***C++:*** Core programming language for implementation.
* ***Visual Studio 2022:*** Integrated Development Environment (IDE).
* ***Wireshark:*** Network protocol analyzer for monitoring.
* ***TreeFrog Framework:*** For future web development

## ***Conclusion:-***

This project successfully demonstrated the practical application of cryptographic concepts by:

* Creating digital certificates and key pairs for client and server communication
* Implementing comprehensive functions in the crypto wrapper
* Securing a custom client-server protocol using the crypto wrapper implementation

The future development of an interactive website will further enhance the educational value of this project, making complex cryptographic concepts more accessible to a wider audience.

***Future Work:-***

While we have successfully implemented the secure custom protocol, there's still room for improvement. Our next step is to develop an interactive website that demonstrates these cryptographic concepts. This will provide a more user-friendly interface for students and professionals to experiment with and learn about cryptography.

## ***Commands to Create Keys and Certificates:-***

We used OpenSSL to create a self-signed root certificate and generate RSA key pairs for two parties, Alice and Bob. The following commands were used:

### *Generate Root Certificate:*

### openssl req -x509 -newkey rsa:3072 -sha384 -nodes -keyout rootCA.key -out rootCA.crt -subj "/CN=RootCA" -days 365 -set\_serial 01

### *Generate Alice's Key and CSR:*

### openssl req -newkey rsa:3072 -sha384 -nodes -keyout alice.key -out alice.csr -subj "/CN=Alice.com"

### *Sign Alice's Certificate:*

### openssl x509 -req -in alice.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out alice.crt -days 365 -sha384 -set\_serial 02

### *Generate Bob's Key and CSR:*

### openssl req -newkey rsa:3072 -sha384 -nodes -keyout bob.key -out bob.csr -subj "/CN=Bob.com"

### *Sign Bob's Certificate:*

### openssl x509 -req -in bob.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial -out bob.crt -days 365 -sha384 -set\_serial 03

### *Verify Certificates:*

### openssl x509 -text -noout -in alice.crt

### openssl x509 -text -noout -in bob.crt

### These commands create a root CA certificate, and individual certificates for Alice and Bob, signed by the root CA.

### *Note:*

### Make sure that all the key files and certificates are available in the below paths. So, all the .csr, .crt and .key files must be copied into the below locations.

### Ex:-

### D:\Navin Files\Projects\Intel Unnati Industrial Training Program 2024\Code\Custom protocol\Custom protocol\udp\_party\x64\Debug

### D:\Navin Files\Projects\Intel Unnati Industrial Training Program 2024\Code\Custom protocol\Custom protocol\udp\_party\crypto\_test

***Crypto Wrapper Implementation:-***

We implemented a crypto wrapper using OpenSSL to handle various cryptographic operations. This wrapper includes functions for:

* Symmetric encryption and decryption
* HMAC generation and verification
* Digital signature creation and verification
* Key exchange using Diffie-Hellman

***Secure Custom Protocol Development:-***

The custom protocol was secured using the SIGMA (SIGn-and-MAc) key exchange protocol. This provides mutual authentication and establishes a secure channel for communication between Alice and Bob.

## ***Running the Project:-***

Open Solution in Visual Studio 2022.

Right Click Each Project and do the below.

Properties -> C/C++ -> Preprocessor -> Preprocessor Definitions -> Add “***OPENSSL;***” at the start.

Properties -> Linker -> General -> Additional Library Directories -> Add OpenSSL Library Locations.

Ex:-

C:\Program Files\OpenSSL-Win64\lib\VC\x64\MDd

C:\Program Files\OpenSSL-Win64\lib\VC\x64\MD

Build the Project by Using Build Command in Visual Studio.

***Wireshark:-***

Choose “***Adapter for loopback traffic capture***” and type in the filter “***udp.srcport == 60000 or udp.dstport == 60000***”

***Running Client and Server:-***

Navigate to Build Directory.

Ex:-

cd D:\Navin Files\Projects\Intel Unnati Industrial Training Program 2024\Code\Custom protocol\Custom protocol\udp\_party\x64\Debug

Run Server.

Ex:-

./udp\_party -port 60000 -key alice.key -pwd alice -cert alice.crt -root rootCA.crt -peer Bob.com

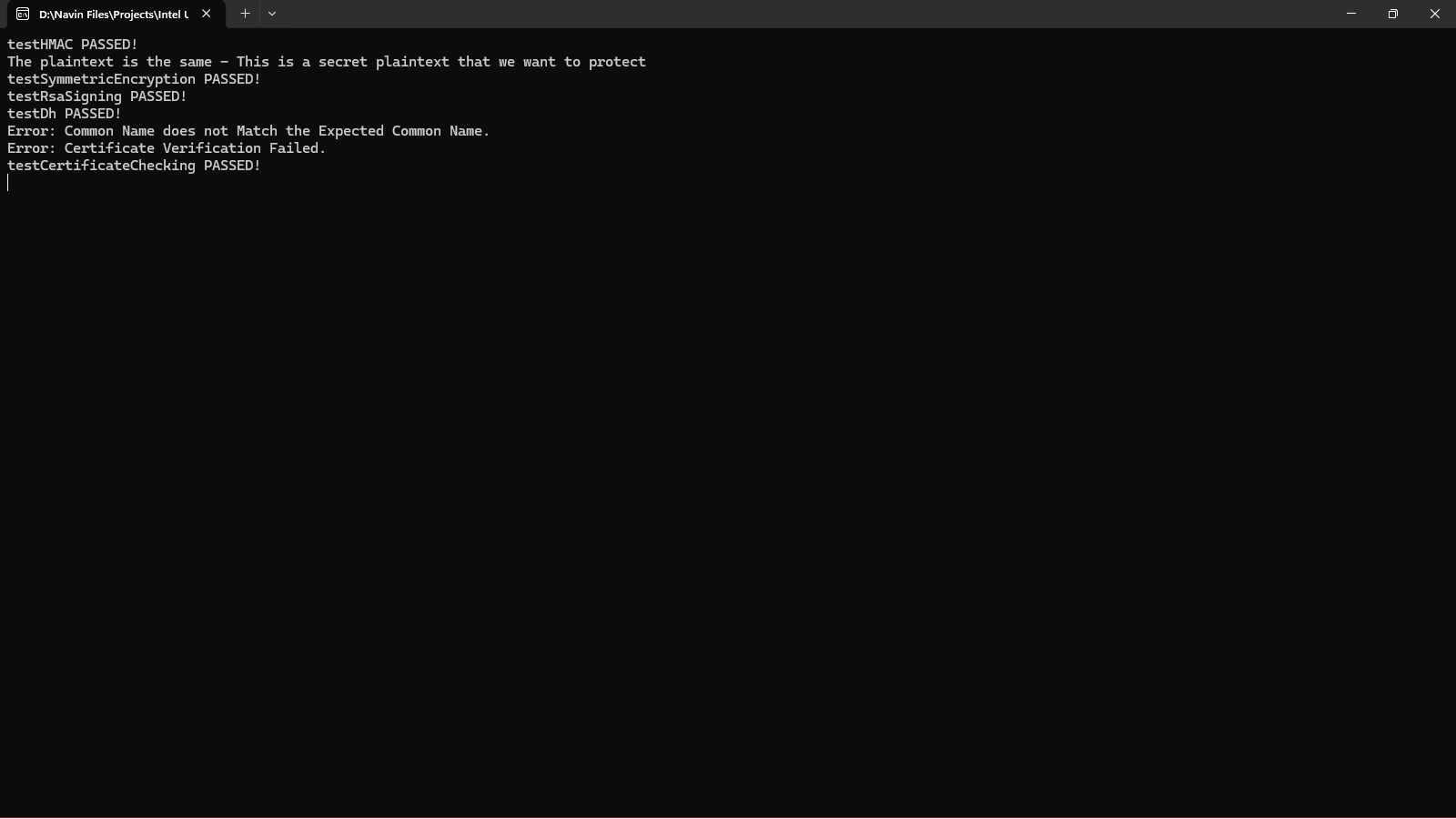
Run Client.

Ex:-

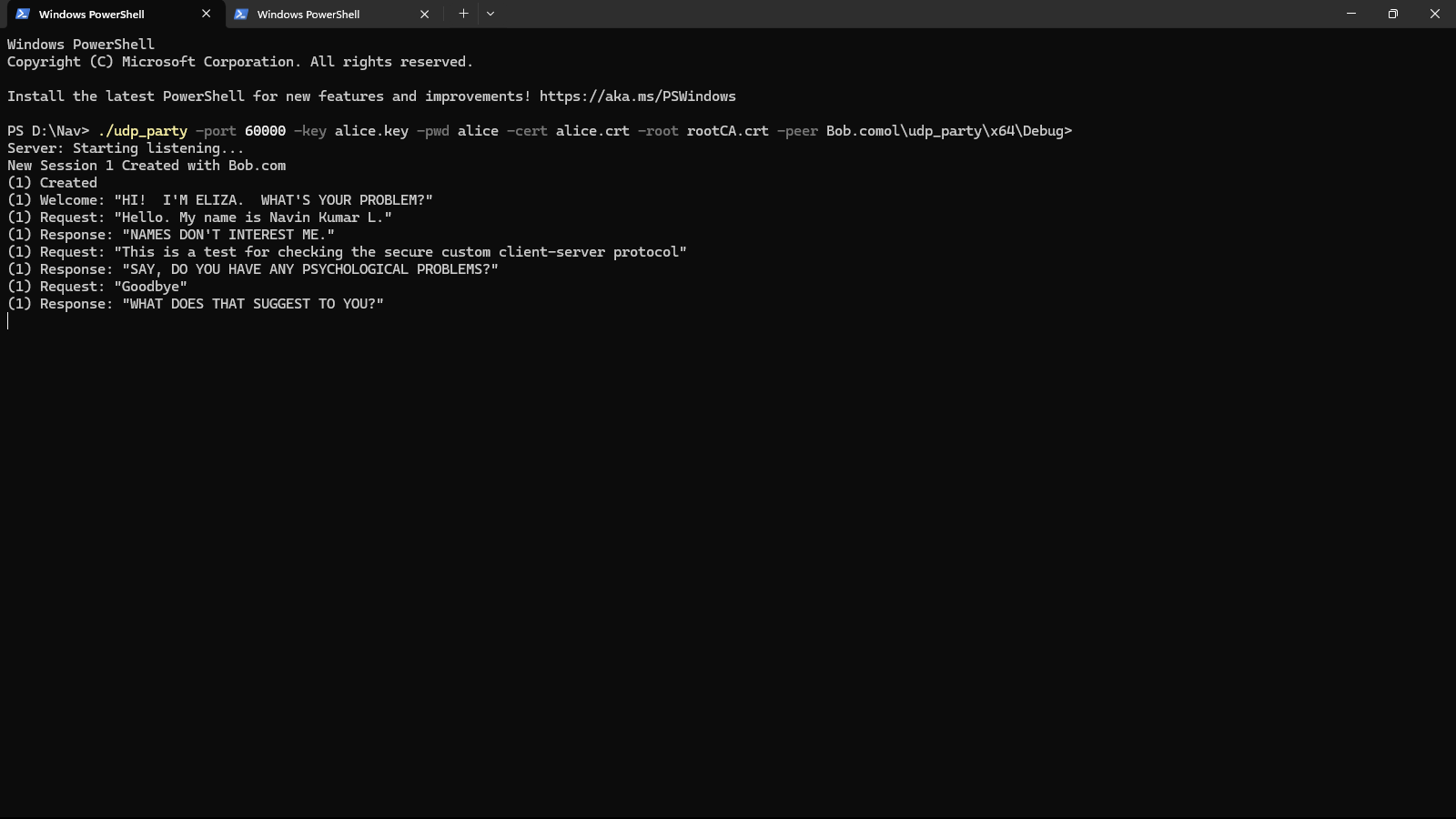
./udp\_party -ip 127.0.0.1 -port 60000 -key bob.key -pwd bob -cert bob.crt -root rootCA.cr

***Output Screen Shots:-***

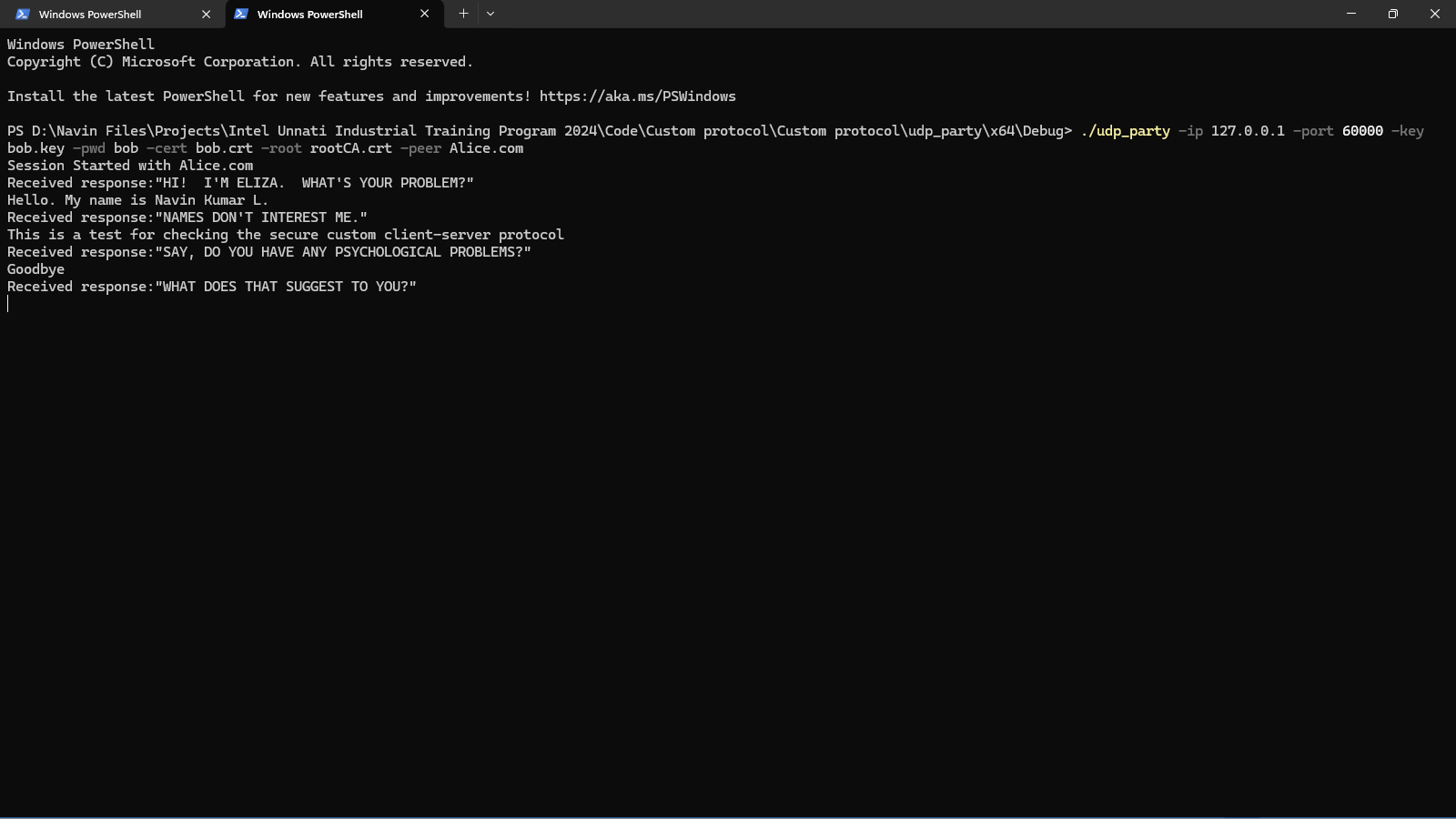
***Crypto Wrapper Unit Test Cases***



***Server***



***Client***



***Wireshark***

