Intel Unnati Training

Summer Internship - 2024

Contents:

Team Name: The Achievers

Team Members:

a. K. Koushik
b. J. Dheeraj Lakshman
c. B. Thanuja

Problem Statement: Cryptography Simulation with mbedTLS/OpenSSL Library Usage and User Interaction.

Objective: The project aims to create an interactive cryptography simulation platform that leverages mbedTLS or OpenSSL libraries. Users will be able to establish a secure connection between the server and the client. 3rd party cannot access the conversation between the server and the client. The conversation can only be decrypted/analyzed by using the respective key.

Technologies Used:

Cryptographic Libraries:

- OpenSSL

Programming Languages:

- C++

Software:

- Visual Studio 2019/2022
- Wire Shark

System Resources:

- Command Prompt

Challenges:

- **Connectivity with OpenSSL:**

  To guarantee compatibility and seamless integration with OpenSSL libraries, a thorough comprehension and management of various APIs and configurations were necessary.

- **Key Management:**

  It was difficult and essential to create a user-friendly, safe key management system in order to create, distribute, and store keys in a secure manner in order to preserve overall security.

- **Encryption and decryption in real-time:**

  It was difficult to implement real-time encryption and decryption while preserving responsiveness and efficiency, particularly when dealing with big data quantities.

- **Establishing a Secure Connection:**

  It took careful attention to detail to develop reliable protocols for establishing secure connections, including managing edge cases and potential security flaws.

- **Handling Errors and Troubleshooting:**

  The task of debugging cryptographic activities has required extensive testing and complex error-handling systems because these problems are frequently non-trivial and can be challenging to replicate.

- **Management of Resources:**

  It was essential to effectively manage system resources (CPU, memory) during the encryption and decryption processes in order to avoid performance snags and guarantee a seamless user experience.

- **Protection of User Data:**

  Robust encryption and secure storage solutions were necessary to protect user data from unauthorized access, particularly when users were undertaking sensitive cryptographic operations.

Learning outcomes:

The integration of OpenSSL libraries, this project offered a thorough education in cryptographic principles and safe communication. The participants gained knowledge about how to handle cryptographic keys, create secure server-client connections, and carry out encryption and decryption operations in real time. They acquired useful abilities in performance optimization, error management, user authentication, and guaranteeing adherence to industry standards. The project culminated in a solid and safe cryptographic simulation platform, emphasizing the value of user-friendly design, safe API development, and keeping up with the most recent developments in cryptography.

Introduction:

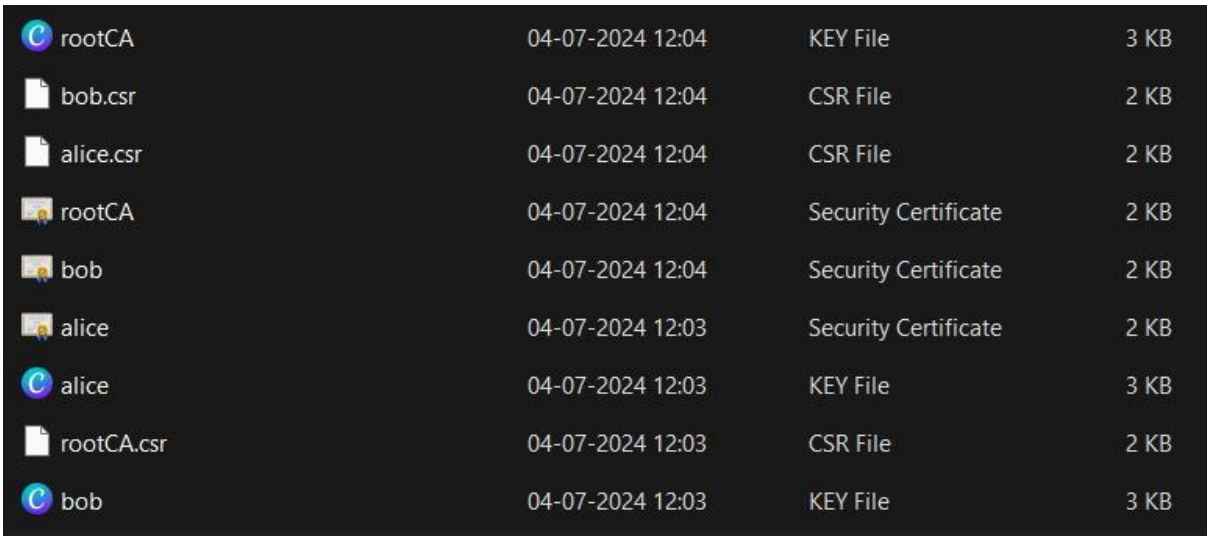
The science of safeguarding information and communication by using codes and ciphers to make sure that only intended recipients can decipher the given data is known as cryptography. It includes methods like encryption, which changes readable data into an unintelligible format, and decryption, which restores the original data. Digital signatures, hashing algorithms, and key exchange techniques are also used in cryptography to ensure non-repudiation, confidentiality, integrity, and authenticity. Modern cryptography, which has its roots in antiquity, uses sophisticated mathematical algorithms and computer science concepts to safeguard digital data in a range of contexts, including private communications, data storage, and safe online transactions. Cryptography, the foundation of cybersecurity, is always changing to combat new threats and weaknesses in the digital environment.

Fundamentals of Cryptography:

- **Confidentiality** – The data cannot be read by anyone else
- **Authenticity** – The receiver/reader knows that data originated from a trusted source
- **Integrity** – The data hasn't been tampered with in transit/storage
- **Non-repudiation** – The sender cannot dispute its authorship or the validity of data

Exercise 1: Digital Certificates & Keys

1. Create a Self-Signed root certificate(rootCA.crt) with RSA key size of 3072 with SHA384 and set serial number 01
2. Generate RSA keypair of size 3072 with SHA384 for "Alice" and sign with root CA and set serial number 02
3. Generate RSA keypair of size 3072 with SHA384 for "Bob" and sign with root CA and set serial number 03

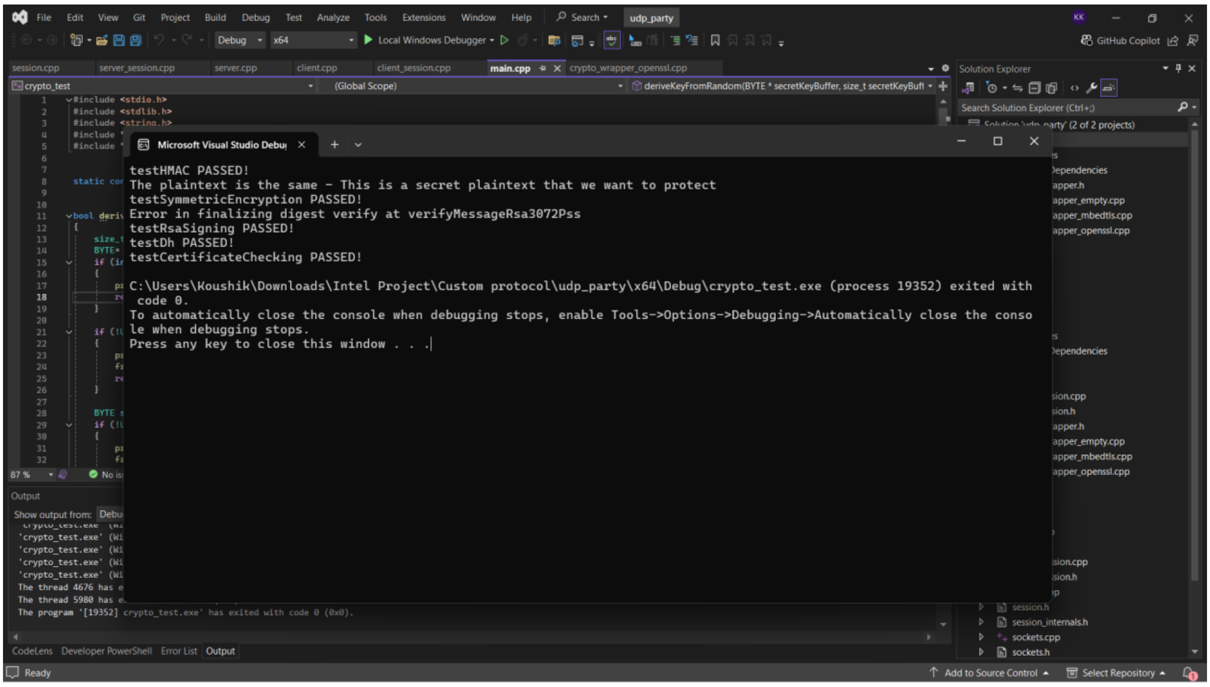


rootCA	04-07-2024 12:04	KEY File	3 KB
bob.csr	04-07-2024 12:04	CSR File	2 KB
alice.csr	04-07-2024 12:04	CSR File	2 KB
rootCA	04-07-2024 12:04	Security Certificate	2 KB
bob	04-07-2024 12:04	Security Certificate	2 KB
alice	04-07-2024 12:03	Security Certificate	2 KB
alice	04-07-2024 12:03	KEY File	3 KB
rootCA.csr	04-07-2024 12:03	CSR File	2 KB
bob	04-07-2024 12:03	KEY File	3 KB

Exercise 2:

1. Implement a crypto wrapper and make the crypto unit test pass.
2. Secure the protocol using your crypto wrapper implementation.

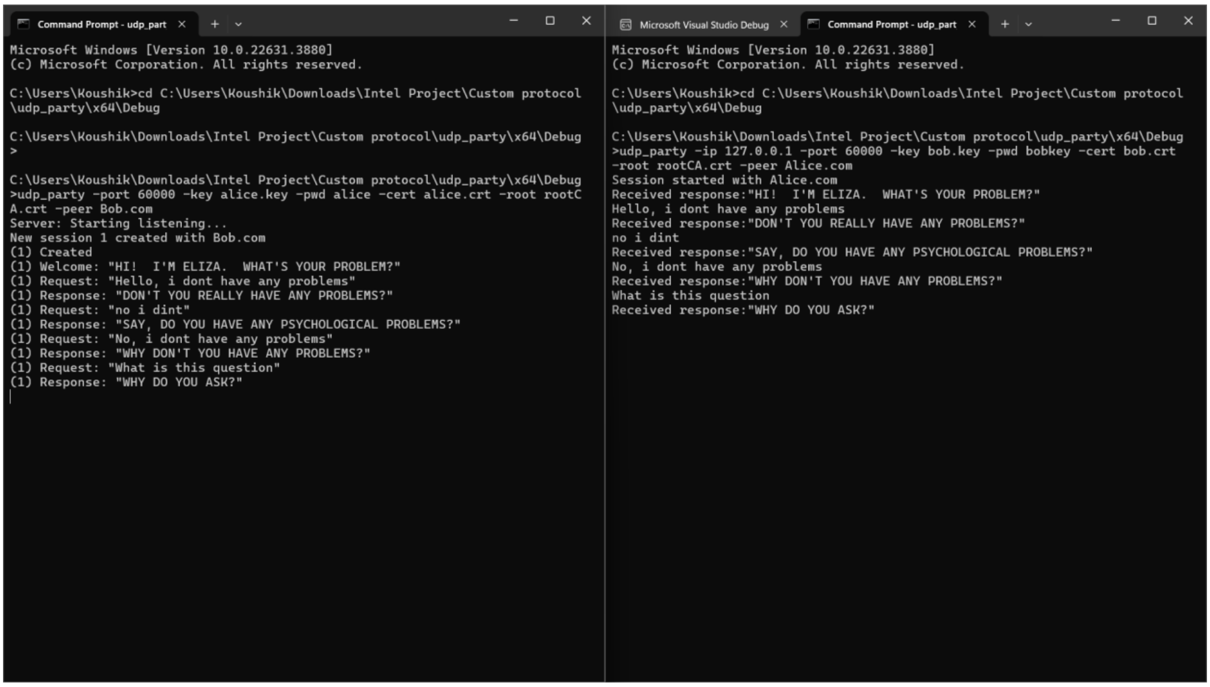
Crypto-wrapper Testcase:


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
udp_party
Debug x64 Local Windows Debugger
session.cpp server_session.cpp server.cpp client.cpp client_session.cpp main.cpp crypto_wrapper_openssl.cpp
crypto_test
(Global Scope)
deriveKeyFromRandom(BYTE * secretKeyBuffer, size_t secretKeyBufl
Solution Explorer
Microsoft Visual Studio Debug
Output
Ready
Add to Source Control
Select Repository

```
testHMAC PASSED!
The plaintext is the same - This is a secret plaintext that we want to protect
testSymmetricEncryption PASSED!
Error in finalizing digest verify at verifyMessageRsa3072Pss
testRsaSigning PASSED!
testDh PASSED!
testCertificateChecking PASSED!

C:\Users\Koushik\Downloads\Intel Project\Custom protocol\udp_party\x64\Debug\crypto_test.exe (process 19352) exited with
 code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
The program '[19352] crypto_test.exe' has exited with code 0 (0x0).
```

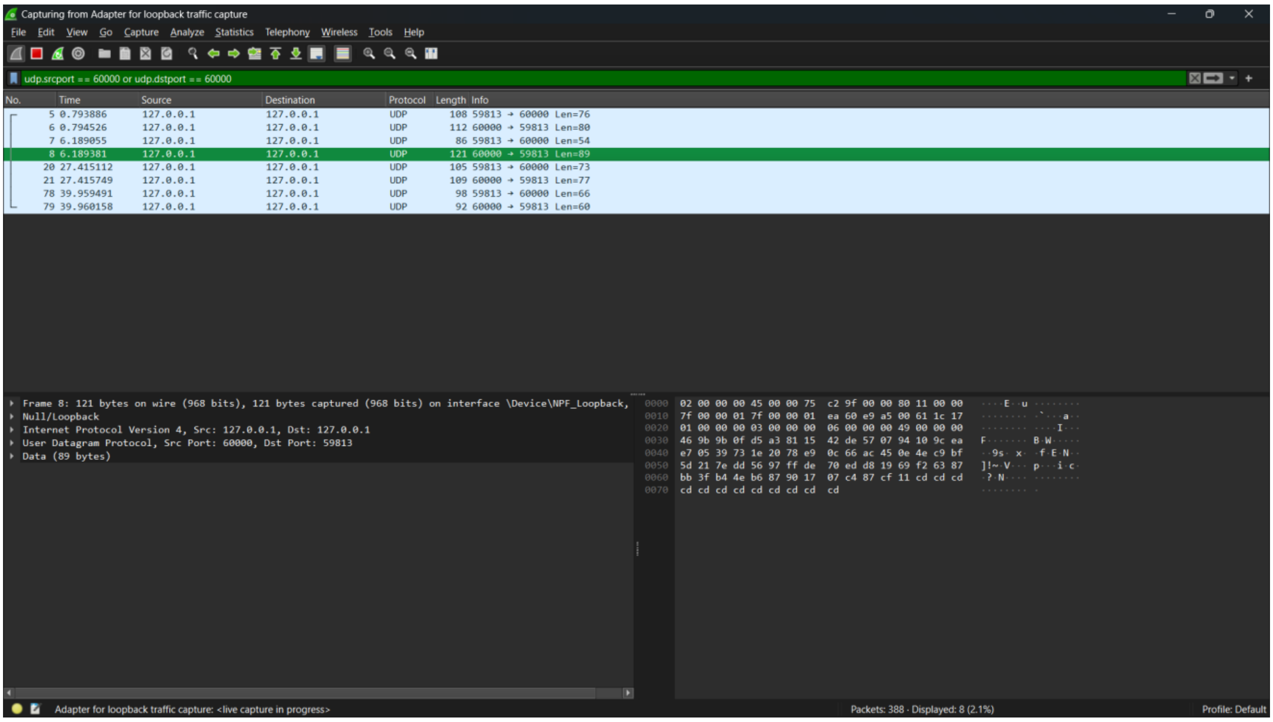
Client and Server connection:


Command Prompt - udp_part
Microsoft Visual Studio Debug
Command Prompt - udp_part

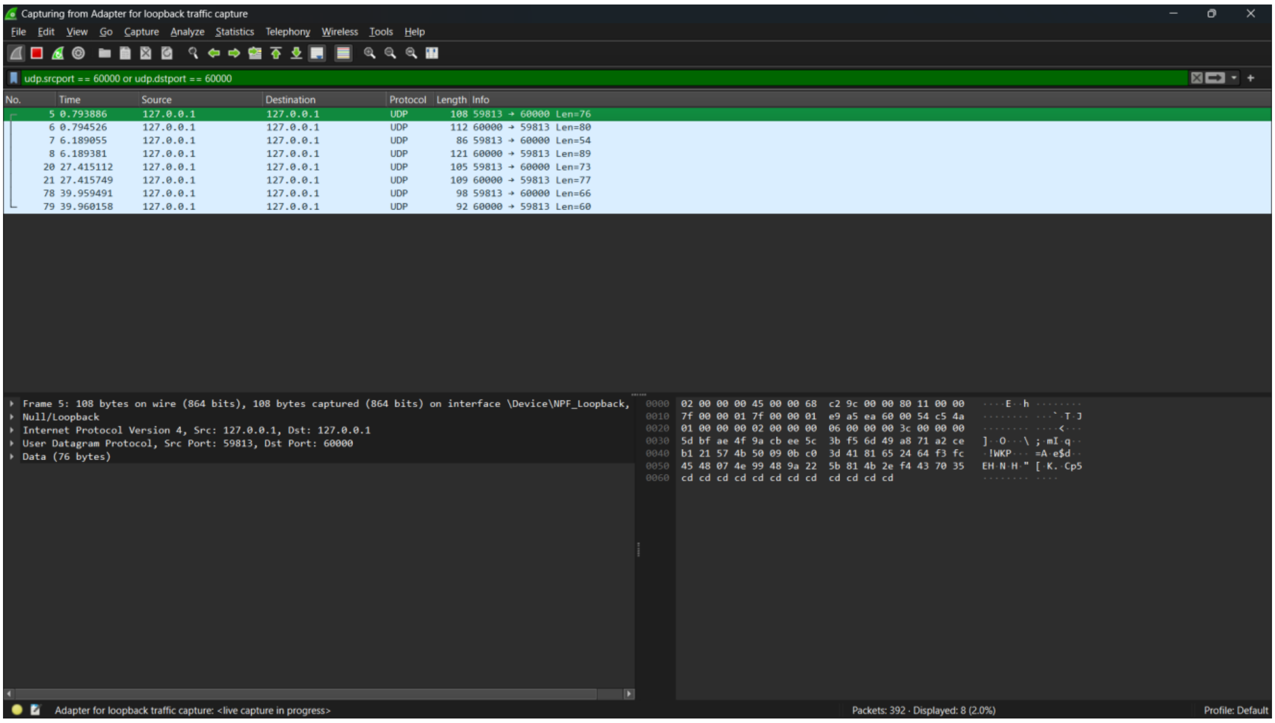
```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Koushik>cd C:\Users\Koushik\Downloads\Intel Project\Custom protocol
\udp_party\x64\Debug

C:\Users\Koushik\Downloads\Intel Project\Custom protocol\udp_party\x64\Debug
>

C:\Users\Koushik\Downloads\Intel Project\Custom protocol\udp_party\x64\Debug
>udp_party -port 60000 -key alice.key -pwd alice -cert alice.crt -root rootC
A.crt -peer Bob.com
Server: Starting listening...
New session 1 created with Bob.com
(1) Created
(1) Welcome: "HI!  I'M ELIZA.  WHAT'S YOUR PROBLEM?"
(1) Request: "Hello, i dont have any problems"
(1) Response: "DON'T YOU REALLY HAVE ANY PROBLEMS?"
(1) Request: "no i dint"
(1) Response: "SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?"
(1) Request: "No, i dont have any problems"
(1) Response: "WHY DON'T YOU HAVE ANY PROBLEMS?"
(1) Request: "What is this question"
(1) Response: "WHY DO YOU ASK?"
```

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Koushik>cd C:\Users\Koushik\Downloads\Intel Project\Custom protocol
\udp_party\x64\Debug

C:\Users\Koushik\Downloads\Intel Project\Custom protocol\udp_party\x64\Debug
>udp_party -ip 127.0.0.1 -port 60000 -key bob.key -pwd bobkey -cert bob.crt
-root rootCA.crt -peer Alice.com
Session started with Alice.com
Received response:"HI!  I'M ELIZA.  WHAT'S YOUR PROBLEM?"
Hello, i dont have any problems
Received response:"DON'T YOU REALLY HAVE ANY PROBLEMS?"
no i dint
Received response:"SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?"
No, i dont have any problems
Received response:"WHY DON'T YOU HAVE ANY PROBLEMS?"
What is this question
Received response:"WHY DO YOU ASK?"
```

Encrypted Outputs:



```
udp.srcport == 60000 or udp.dstport == 60000

No.  Time          Source      Destination  Protocol  Length  Info
   5 0.793886      127.0.0.1   127.0.0.1    UDP        108 59813 → 60000 Len=76
   6 0.794526      127.0.0.1   127.0.0.1    UDP        112 60000 → 59813 Len=80
   7 6.189055      127.0.0.1   127.0.0.1    UDP         86 59813 → 60000 Len=54
   8 6.189381      127.0.0.1   127.0.0.1    UDP        121 60000 → 59813 Len=89
  20 27.415112     127.0.0.1   127.0.0.1    UDP        105 59813 → 60000 Len=73
  21 27.415749     127.0.0.1   127.0.0.1    UDP        109 60000 → 59813 Len=77
  78 39.959491     127.0.0.1   127.0.0.1    UDP         98 59813 → 60000 Len=66
  79 39.960158     127.0.0.1   127.0.0.1    UDP         92 60000 → 59813 Len=60

Frame 8: 121 bytes on wire (968 bits), 121 bytes captured (968 bits) on interface \Device\NPF_Loopback,
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 60000, Dst Port: 59813
Data (89 bytes)

0000  02 00 00 00 45 00 00 75  c2 9f 00 00 80 11 00 00   ····E··u ········
0010  7f 00 00 01 7f 00 00 01  ea 60 e9 a5 00 61 1c 17   ········ ·`···a··
0020  01 00 00 00 03 00 00 00  06 00 00 00 49 00 00 00   ········ ····I···
0030  46 9b 9b 0f d5 a3 81 15  42 de 57 07 94 10 9c ea   F······· B·W·····
0040  e7 05 39 73 1e 20 78 e9  0c 66 ac 45 0e 4e c9 bf   ··9s· x· ·f·E·N··
0050  5d 21 7e dd 56 97 ff de  70 ed d8 19 69 f2 63 87   ]!~·V··· p···i·c·
0060  bb 3f b4 4e b6 87 90 17  07 c4 87 cf 11 cd cd cd   ·?·N···· ········
0070  cd cd cd cd cd cd cd cd  cd                        ········ ·

Adapter for loopback traffic capture: <live capture in progress>    Packets: 388 · Displayed: 8 (2.1%)    Profile: Default
```



```
udp.srcport == 60000 or udp.dstport == 60000

No.  Time          Source      Destination  Protocol  Length  Info
   5 0.793886      127.0.0.1   127.0.0.1    UDP        108 59813 → 60000 Len=76
   6 0.794526      127.0.0.1   127.0.0.1    UDP        112 60000 → 59813 Len=80
   7 6.189055      127.0.0.1   127.0.0.1    UDP         86 59813 → 60000 Len=54
   8 6.189381      127.0.0.1   127.0.0.1    UDP        121 60000 → 59813 Len=89
  20 27.415112     127.0.0.1   127.0.0.1    UDP        105 59813 → 60000 Len=73
  21 27.415749     127.0.0.1   127.0.0.1    UDP        109 60000 → 59813 Len=77
  78 39.959491     127.0.0.1   127.0.0.1    UDP         98 59813 → 60000 Len=66
  79 39.960158     127.0.0.1   127.0.0.1    UDP         92 60000 → 59813 Len=60

Frame 5: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface \Device\NPF_Loopback,
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 59813, Dst Port: 60000
Data (76 bytes)

0000  02 00 00 00 45 00 00 68  c2 9c 00 00 80 11 00 00   ····E··h ········
0010  7f 00 00 01 7f 00 00 01  e9 a5 ea 60 00 54 c5 4a   ········ ···`·T·J
0020  01 00 00 00 02 00 00 00  06 00 00 00 3c 00 00 00   ········ ····<···
0030  5d bf ae 4f 9a cb ee 5c  3b f5 6d 49 a8 71 a2 ce   ]··O···\ ;·mI·q··
0040  b1 21 57 4b 50 09 0b c0  3d 41 81 65 24 64 f3 fc   ·!WKP··· =A·e$d··
0050  45 48 07 4e 99 48 9a 22  5b 81 4b 2e f4 43 70 35   EH·N·H·" [·K.·Cp5
0060  cd cd cd cd cd cd cd cd  cd cd cd cd               ········ ····

Adapter for loopback traffic capture: <live capture in progress>    Packets: 392 · Displayed: 8 (2.0%)    Profile: Default
```

Applications:

- Secure communication
- Digital signatures
- Data encryption
- Secure email
- Blockchain and cryptocurrencies
- Secure file storage
- Authentication mechanisms
- Digital certificates
- Secure voting systems
- Secure messaging apps
- Access control systems
- Payment systems and online banking
- Virtual Private Networks (VPNs)
- Secure remote access
- Intellectual property protection

Conclusion:

Using OpenSSL or mbedTLS libraries, the interactive cryptography simulation platform provides a reliable and workable way to create secure connections between a client and a server. This guarantees that all interactions are encrypted, shielded from prying eyes, and can only be decrypted with the appropriate keys. This platform offers users priceless practical experience and profound insights into the critical role that cryptography plays in protecting digital communications, based on our extensive knowledge in cryptographic implementations and secure communication protocols.