Intel Unnati Industrial Training Program

2024

Problem statement:05

Cryptography Simulation with mbedTLS/OpenSSL Library Usage and User Interaction

Unique Idea Brief (Solution)

- To develop a cryptography simulation that demonstrates secure communication over UDP using OpenSSL/MbedTLS libraries which showcases the use of cryptographic protocols for encryption, confidentiality.
- **Unique Aspects**
  - Security protocol implementations like SIGMA Protocol.
  - Multi-Role simulation
  - Password-protected keys
  - Interactive Simulation
  - Real-Time communication testing

Process Flow:

Exercise #1 – Creating Digital Certificates

Installation of OpenSSL Library and setting up the environment configuration.

1. **Generate a Private Key**:
   - Generated an RSA private key with a size of 3072 bits.

2. **Create a Certificate Signing Request (CSR)**:

   Created a Certificate Signing Request using the generated private key and specified the SHA-384 hashing algorithm

3. **Generate the Self-Signed Certificate**:
   - Created the self-signed root certificate using the CSR and private key.
   - Set the serial number to 01 during the certificate creation process.

Commands used for Task 01: creating digital certificates

1) Generate a 3072-bit RSA private key.

- Command :openssl genpkey -algorithm RSA -out rootCA.key -pkeyopt rsa_keygen_bits:3072

2) Create a Configuration File.

3) Generate the Self-Signed Certificate:

Use the configuration file to create a self-signed root certificate with SHA-384 and a serial number of 01.

- Command : openssl req -x509 -new -nodes -key rootCA.key -sha384 -days 3650 -out rootCA.crt -config openssl.cnf -set_serial 01.

4) Verify the Certificate.

- Command: openssl x509 -in rootCA.crt -text –noout.

(Same has been done for Generating RSA keypair of size 3072 with SHA384 for “Alice” and sign with root CA and set serial number 02
Generate RSA keypair of size 3072 with SHA384 for “Bob” and sign with root CA and set serial number 03)

Images of Digital Certificates

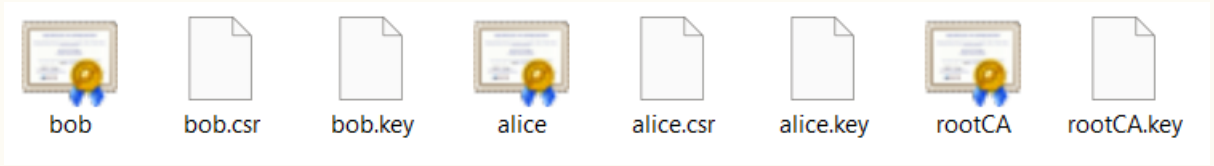
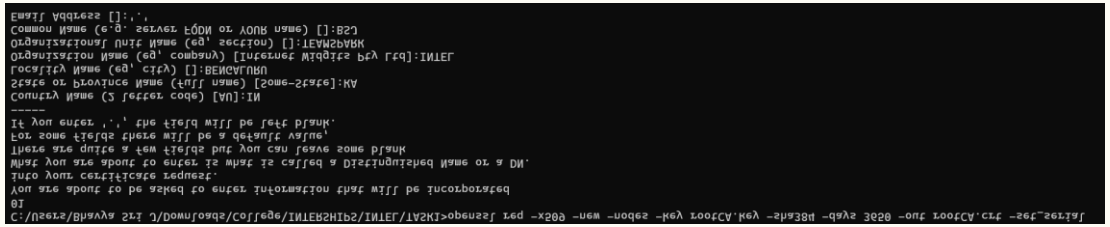
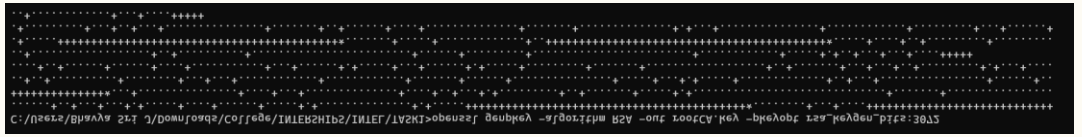


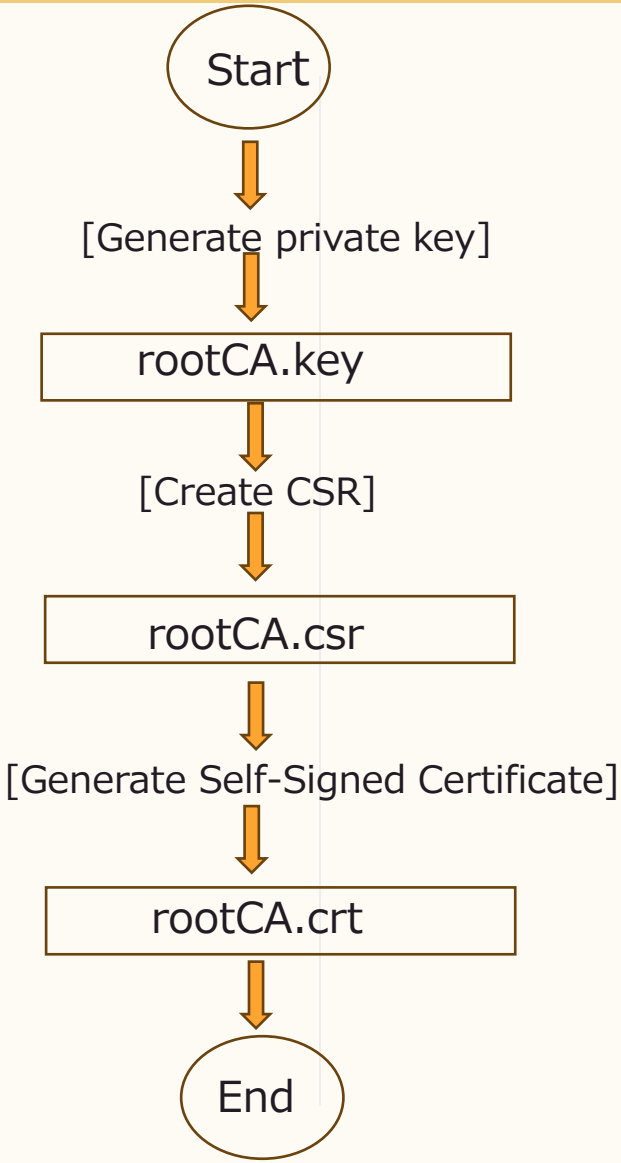
Image of rootCA.crt

```
C:\Users\Bhavya Sri J\Downloads\College\INTERSHIPS\INTEL\TASK1>openssl x509 -in rootCA.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha384WithRSAEncryption
        Issuer: C=IN, ST=KA, L=BENGALURU, O=INTEL, OU=TEAMSPARK, CN=BSJ, emailAddress='.'
        Validity
            Not Before: Jun  3 12:21:11 2024 GMT
            Not After : Jun  1 12:21:11 2034 GMT
        Subject: C=IN, ST=KA, L=BENGALURU, O=INTEL, OU=TEAMSPARK, CN=BSJ, emailAddress='.'
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (3072 bit)
                Modulus:
                    00:db:ed:88:b7:43:0e:78:a1:bf:ca:13:84:8e:f1:
                    6f:62:16:b7:fa:56:f7:16:21:61:f0:4c:eb:30:1f:
                    cf:9c:30:62:62:c7:db:74:d5:c2:24:97:7b:59:dc:
                    30:a4:25:79:16:40:d5:f1:b2:72:53:50:87:fc:d2:
                    22:32:c8:af:cc:4b:79:7b:fc:b5:46:6d:1e:4b:ab:
                    5a:72:08:4c:29:48:46:aa:e1:46:fd:3b:64:48:b4:
                    53:c8:ab:c2:6d:8c:73:74:ac:ce:86:e3:c9:f9:89:
                    dc:ac:bc:21:c0:46:dd:42:20:fa:cc:09:88:e0:55:
                    4f:a1:07:88:5d:5e:c2:84:ad:71:0a:0d:92:2d:e0:
                    4c:66:83:ae:66:1f:28:cf:62:2e:33:f0:ef:c3:e4:
                    67:69:90:42:32:e5:02:41:f0:e1:c6:b6:39:de:f6:
                    09:cb:ec:eb:bf:dd:74:aa:aa:03:8f:ba:b1:5f:46:
                    86:a1:9f:ef:36:b5:92:39:22:36:0c:e6:2c:90:a3:
                    9e:b3:fe:4e:4b:a1:e6:2f:64:f5:dc:93:f1:cf:66:
                    53:3e:de:61:89:e5:af:d2:c5:fc:7e:99:b5:c8:9f:
                    79:96:56:68:58:11:52:a7:1f:7a:62:e7:40:69:18:
                    51:da:ec:84:71:15:69:18:dd:17:f0:22:bd:53:3c:
                    2d:95:c7:83:d2:81:f1:d1:43:b9:0c:c4:5b:bf:83:
                    b1:b8:56:b6:91:46:23:95:2c:fa:85:b1:c5:39:4a:
```





Architecture Diagram for Exercise #1

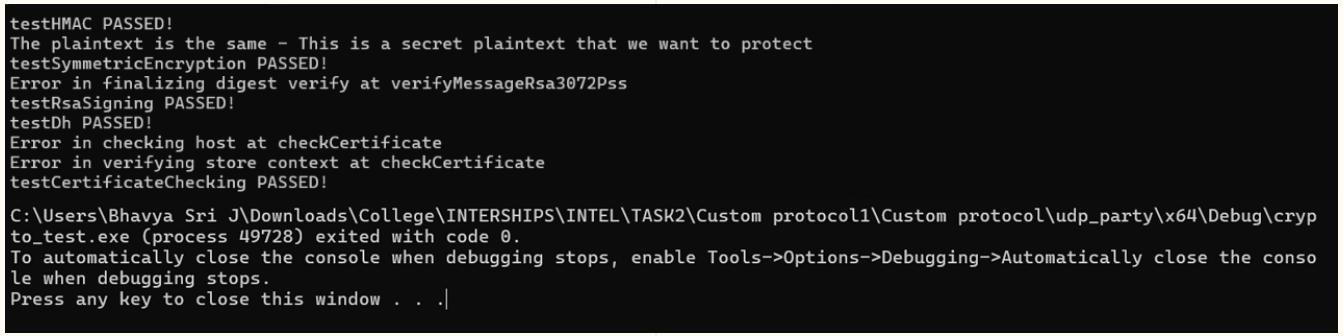


Process flow:

Exercise #2 - Securing a custom protocol

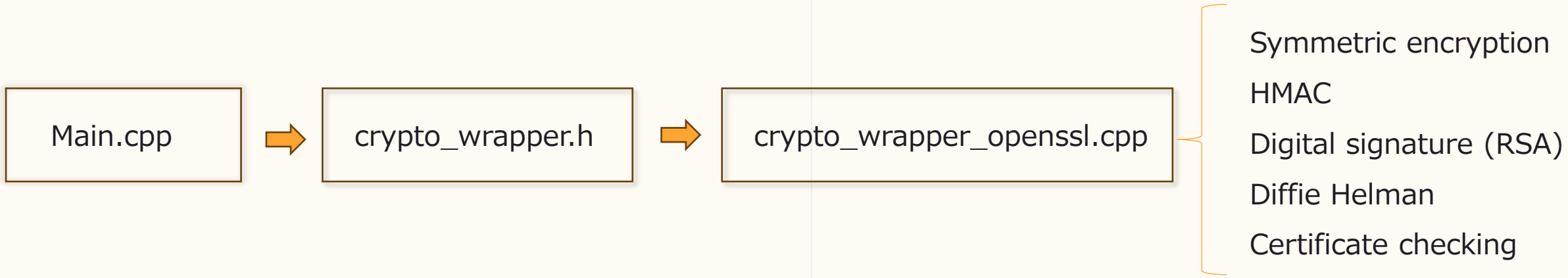
part 1: Implement crypto wrapper and make the crypto unit test pass

- We have chosen OpenSSL crypto library for the crypto wrapper.
- Configured the environment to OpenSSL library.
- Filled the missing functionalities in the crypto wrapper openssl crypto wrapper using the OpenSSL APIs.
- Passed all the test cases in the crypto_test project



Architecture Diagram for Exercise #2

Exercise – part 1:



Process flow:

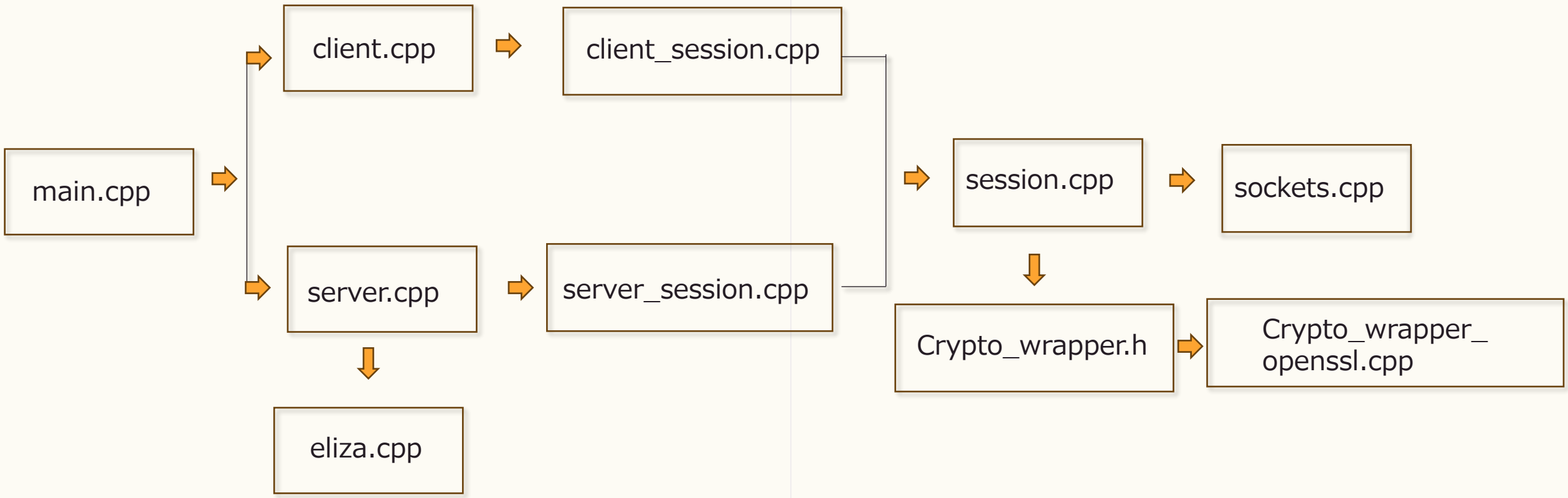
Exercise #2 - Securing a custom protocol

Part 2: Add encryption and integrity protection over the traffic

- After the implementation of the crypto_test project, now we have added encryption to establish protected/secured protocol over the traffic.
- SIGMA (SIGn-and-Mac) protocol was utilized.
- In this protocol Diffie-Hellman was used to establish the shared secret key between the client and server parties.
- This was done in order to prevent man-in-middle attacks, replay attacks. Ensuring integrity and confidentiality.

Architecture Diagram for Exercise #2

Exercise – part 2:



Technologies used

OpenSSL

Is an open-source command line tool that is commonly used to generate private keys, create CSRs, install your SSL/TLS certificate, and identify certificate information.

Visual Studio

Is an integrated development environment (IDE) developed by Microsoft. It is used to develop computer programs including websites, web apps, web services and mobile apps.

Wireshark

Powerful and widely-used network protocol analyzer. It enables users to capture and interactively browse the traffic running on a computer network.

Useful APIs

- EVP_MD_CTX_new
- EVP_PKEY_derive
- EVP_PKEY_derive_init
- EVP_PKEY_CTX_new_id
- EVP_PKEY_CTX_set_hkdf_md
- EVP_PKEY_CTX_set1_hkdf_salt
- EVP_PKEY_CTX_set1_hkdf_key
- EVP_PKEY_CTX_add1_hkdf_info
- EVP_get_digestbyname
- EVP_DigestSignInit
- EVP_DigestSignUpdate
- EVP_DigestSignFinal
- EVP_PKEY_new_raw_private _key
- And many more

Team members and contribution:

[Team Member 1 : Bhavya Sri Jonnala]

Contributions:

- Contributed to CryptoWrapper.h, main.cpp in crypto_test project code implementation.
- Documentation of implementation and results.
- Managed project timelines.

[Team Member 2 : Lalitha Devi I]

Contributions:

- Implementation of OpenSSL command scripts for key and certification generation.
- Contributed to server_session.cpp and main.cpp in udp_party project code implementation.
- Documentation of the scripts/commands of digital certificates.

[Team Member 3 : V Asha Reddy]

Contributions:

- Generation of private key and CSR for the client (Bob).
- Contributed to completing crypto_wrapper_openssl.cpp implementation.
- Documentation of the process for generating client keys and certifications.

Team members and contribution:

[Team Member 4 : Udari Jyothi]

Contributions:

- Contributed to main.cpp and server_session.cpp project code implementation.
- Documentation of integrating the generated certificates and keys in the project.
- Testing the proper functionality and security of the protocol.

[Team Member 5 : Harshini Y L]

Contributions:

- Overlooked the process of generating, signing, and managing keys and certificates.
- Implementation of Wireshark in viewing the packets to ensure successful protocol establishment.
- Documentation of overall crypto_test and udp_party projects process and results.

[Mentor : Rajesh S M]

Contributions:

- Guided the team throughout the project completion.

Conclusion

This project represents a practical approach to understanding and demonstrating cryptography principles and algorithms to secure communication between two parties.

Key Highlights

- Utilizes the SIGMA protocol and hybrid encryption to ensure mutual authentication, data confidentiality, and protection against various attacks like man-in-middle etc.
- Simulates both server and client roles, showcasing the full lifecycle of secure communication setup and management.
- Demonstrates practical application through real-time communication testing.

By working collaboratively and dividing tasks among team members, the project ensures balanced contributions and foster a deeper understanding of cryptographic implementations.

THANK YOU