Problem Statement

Cryptography Simulation with mbedTLS /OpenSSL Library

Unique Idea Brief (Solution)

Project Overview

This project involves setting up and running a custom UDP protocol with encryption using the Diffie-Hellman key exchange and the SIGMA protocol. The steps include:

1. **Installing Required Software**: Install OpenSSL, Visual Studio, Strawberry Perl, NASM, and Wireshark.
2. **Building the Custom Protocol**: Download the provided custom protocol, unzip it, open the solution file in Visual Studio, and build the solution.
3. **Running the Protocol**: Use command prompt to run the client and server applications with specific parameters for encryption.
4. **Monitoring Traffic**: Use Wireshark to capture and analyze the encrypted UDP traffic.

The project demonstrates secure communication using the Diffie-Hellman key exchange and the SIGMA protocol, providing hands-on experience with network protocol development and analysis.

Features Offered

Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange is a method of securely exchanging cryptographic keys over a public channel. It allows two parties to generate a shared secret key, which can be used for encrypted communication, without having to transmit the key itself. The process involves the following steps:

1. **Public Parameters**: Both parties agree on a large prime number p and a base g (generator), which are public.
2. **Private Keys**: Each party selects a private key (a for Alice and b for Bob) that is kept secret.
3. **Public Keys**: Each party computes their public key by raising the base g to the power of their private key modulo p (i.e., $A = g^a \mod p$ and $B = g^b \mod p$).
4. **Shared Secret**: Each party computes the shared secret key by raising the other party's public key to the power of their own private key modulo p (i.e., $s = B^a \mod p$ for Alice and $s = A^b \mod p$ for Bob). Both parties end up with the same shared secret key s.

SIGMA Protocol

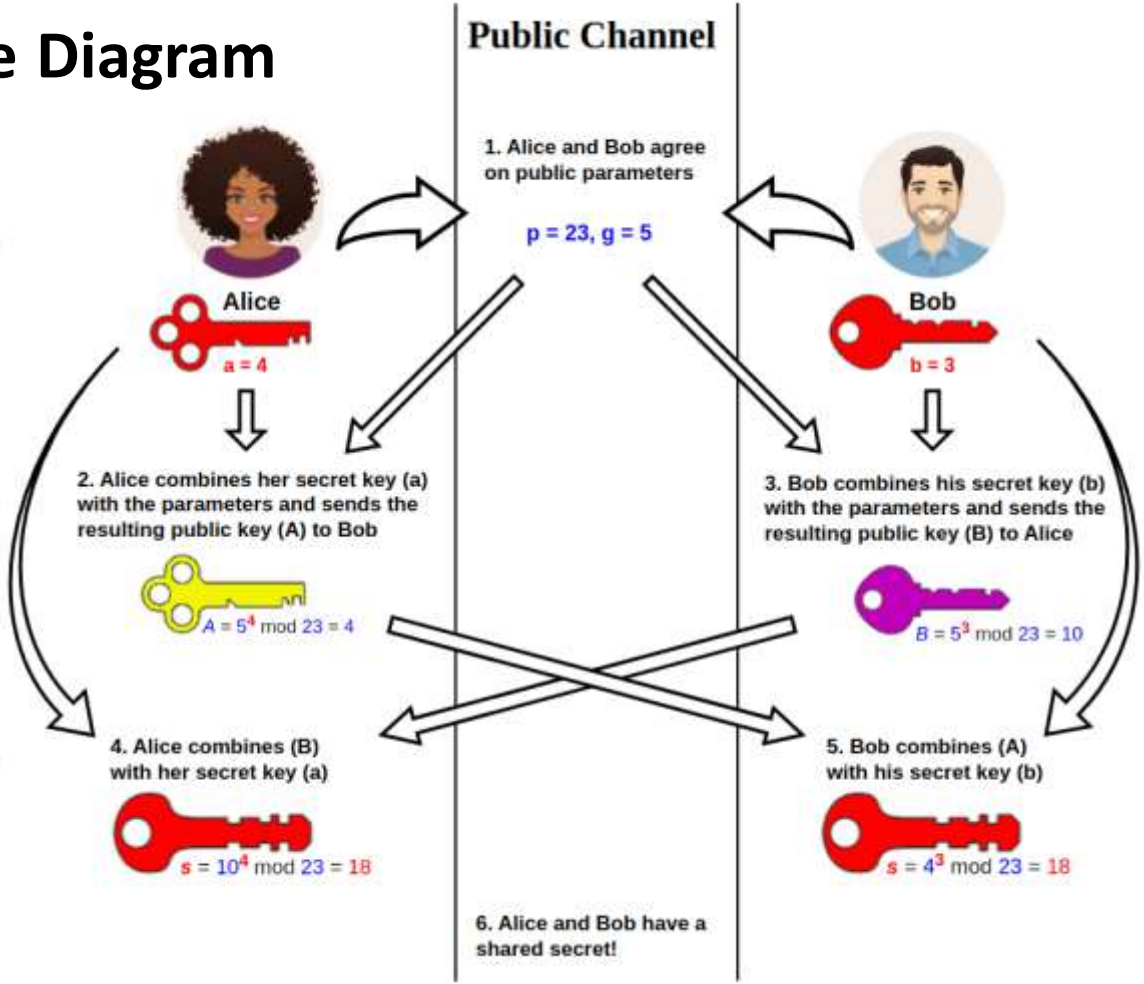
The SIGMA (SIGn and MAc) protocol is used to bind the derived keys (K) with the peer identities. It involves the following steps:

- **Sign the Two Ephemeral Public Keys**: Each party signs the two ephemeral public keys with their own identity.
- **MAC Own Identity**: Each party MACs their own identity with a key derived from the shared secret.

Process flow

1. **Install OpenSSL**: Install the OpenSSL toolkit for SSL/TLS protocols.
2. **Install Visual Studio**: Set up Visual Studio IDE for development.
3. **Install Strawberry Perl**: Install Perl environment for Windows.
4. **Install NASM**: Install the Netwide Assembler for x86 architecture.
5. **Install Wireshark**: Set up Wireshark for network traffic analysis.
6**. Download and Build Custom Protocol**: Download, unzip, and build the custom protocol in Visual Studio.
7.Run Command Prompt Commands: Execute specific commands in two command prompt windows to start the client and server.
8.Monitor Encryption with Wireshark: Use Wireshark to capture and filter encrypted UDP traffic.

Architecture Diagram



Technologies used

OpenSSL

Cryptographic Toolkit: OpenSSL is a robust, full-featured open-source toolkit implementing the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols.

Key Features:

Encryption and Decryption: Supports various encryption algorithms like AES, DES, and RSA.

Certificate Management: Generates and manages SSL/TLS certificates.

Cryptographic Hash Functions: Implements hash functions like SHA-256 and MD5.

Secure Communication: Enables secure communication over networks using SSL/TLS.

Command-Line Tools: Provides a suite of command-line tools for performing cryptographic operations.

Library: Offers a library for developers to integrate cryptographic functions into their applications.

Team members and contribution:

Rohith DR

Contribution:
1)generated key pairs
2)passed all test cases
3)tried to implement it on a custom protocal

Conclusion

In this project, I had the opportunity to dive into the setup and implementation of a custom UDP protocol with encryption, using the Diffie-Hellman key exchange and the SIGMA protocol. I started by installing essential tools like OpenSSL, Visual Studio, Strawberry Perl, NASM, and Wireshark. After that, I downloaded, built, and ran the custom protocol. I also used Wireshark to monitor the process.

I explored the basics of the AES encryption algorithm and the GCD algorithm, gaining a deeper understanding of their roles in cryptography. OpenSSL was particularly crucial in providing the necessary cryptographic toolkit for my project.

This project was a fantastic learning experience, blending theoretical knowledge with hands-on practice. It was a great chance to deepen my understanding of network security, cryptographic protocols, and software development.

Thank you for this wonderful opportunity to learn and grow through this project.