National Institute of Technology,Raipur

Department of Computer Science & Engineering

SECURE IP

A Term Project on Network Programming

GitHub Project Link:
https://github.com/sakshya5/Secure-IP

Submitted By:

Roll no: 14115019 Name: Asma Soni

ABSTRACT

In this project, I aimed to implement a secure IP, which will secure the communications by authenticating and encrypting each IP packet of a communication session.

Keeping my aim in mind, I have implemented an algorithm for making the communication channel secure, and for identification of the authentic client. For this, I have used dynamic read and write keys for encryption.

While making the project, my main aims were:

1. Authentication of clients while logging on.
2. Secure communication between client and server.
3. Even if either of read or write keys are comprised, the other key should not be compromised.
4. Clients can communicate only with authenticated clients.

With all my efforts, I was able to successfully implement these points in a JAVA code.

TABLE OF CONTENTS

INTRODUCTION

Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries. Being heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure. The growth of cryptographic technology has raised a number of legal issues in the information age. Cryptography's potential for use as a tool for espionage and sedition has led many governments to classify it as a weapon and to limit or even prohibit its use and export.

In this project, I have demonstrated how a simple cryptographic program can look like. I have used authentication measures for the clients and the concept of private key. The server maintains a list of the private keys of all the clients.

In this project, I have used Caeser Cipher. It is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. The method is named after Julius Caesar, who used it in his private correspondence.

For sending message from the server to the client, the server encrypts the message with the client's private key before putting it in the communication channel. When the client receives the server's message, it decrypts it using its private key.

For sending message from client to server, the client encrypts the message with it's private key before putting it in the communication channel. When the server receives the client's message, it decrypts it using the client's private key.

SERVER SIDE

ACCEPTING CONNECTIONS

At the server side, a ServerSocket accepts the connection requests made by the client, and starts a ServerThread for each connection. This ServerThread maintains and manages the connections with the clients, and exchanges messages between them.

INITIALIZING THE CONNECTION PARAMETERS

The ServerThread reads the identity of the client and the read encryption key. It adds the client to the allClients list. It then initializes a private Connected Clients (connClients) list to null. It also maintains a private Disconnected Clients (dconCients) list, and adds all the exiting clients in allClients to the list. Also, it adds the present client in the dconClients list of all the clients in allClients list.

SETTING THE READ AND WRITE PRIVATE KEYS

As soon as the client logs on, the server generates a write encryption key and sends it to the client.

From now onwards, all the communication will be done in encrypted format.

INITIAL RESPONSE TO THE CLIENT

As soon as a client logs on onto the server, the server displays a list of online clients to the user. If no one is online, the server shows a message indicating the same.

REPONSE TO connect COMMAND

Whenever the client types connect, the server prints a list of disconnected online client, to whom the client can connect. It then prompts the user to enter the client's name o whom they want to connect. When the user enters the name, it checks for the name in the disconnected list. If it finds it, it removes that client from its disconnected list, and adds it to the connected list. Else, it prints a message showing that the client could not be found.

All these conversations are going on in encrypted format, encrypted by the client's read and write keys.

RESPONSE TO exit COMMAND

When the client types exit, the server sends a message to all the clients that this client has left, and their connection (if existing) has been terminated.

All these conversations are going on in encrypted format, encrypted by the client's read and write keys.

RESPONSE TO bye COMMAND

When the client types “bye” commend with the identity of the client it wants to disconnect to, the server removes that client from it’s connected list and adds it to its disconnected list. Also, it removes this client from the other client’s connected list and adds it to the other client’s disconnected list.

All these conversations are going on in encrypted format, encrypted by the client’s read and write keys.

RESPONSE TO MESSAGES

When the server gets the cipher text of the client, which contains the its message suffixed with the name of the receiver, it decrypts the text with the client’s write key, extracts the receiver’s identity and the message to be sent, encrypts it with the receiver’s read key and sends it to the receiver.

CLIENT SIDE

CONNECTING TO SERVER

Client sends a connection request to the server at the specified port. Once connected, it prints the acknowledgement message on the screen.

SETTING THE READ AND WRITE KEYS

Once the connection has been established, the client generates a write key and sends it to the server. Then it receives the read key generated by the server.

COMMAND connect

This is to be used to connect to a new client.

COMMAND exit

This is to be used to log out from the server.

COMMAND bye

This is to be used to disconnect from a particular.

CONCLUSION AND FUTURE WORK

CONCLUSION

Thus, I have implemented a Server and a Client program using Caeser's cryptography.

FUTURE WORK

Much work can be done more on the project, like:

1. Using two-way asymmetric authentication cryptosystems for client.
2. Using more rigorous encrypting and decrypting algorithms.
3. Designing algorithms where even the server could not encrypt the messages between the clients.

REFERENCES

1. www.wikipedia.com
2. www.github.com
3. https://www.schneier.com
4. www.storagecraft.com
5. www.geeksforgeeks.com
6. www.dewheeler.com