Internal File Sharing with Secured Access

Sathwik Gaddi   
Computer Science *11517879*  
University of North TexasDenton, TX  
sathwikgaddi@my.unt.edu  
  
Tanvi Thirunathan  
Computer Science *11521054*  
University of North TexasDenton, TX  
tanvithirunathan@my.unt.edu  
  
Navya Sree Muppala  
Computer Science  *11518129*  
University of North TexasDenton, TX  
navyasreemuppala@my.unt.edu  
  
Sathvik Reddy Gutha  
Computer Science *11619004*  
University of North TexasDenton, TX  
sathvikreddygutha@my.unt.edu

*Abstract*— Humans use technology for a variety of reasons. One of the primary goals of modern technology is to make life easier and safer. People's communication has been enhanced by the usage of social networking sites such as Facebook, as well as equipment such as computers (including related gadgets) and mobile phones.

File sharing is a topic that has sparked economic, moral, and legal debates all around the world. It is convenient, safe, and durable. The files might be shared across two computers, with one storing the files and the other accessing them from their workstation. Instead of a local workstation, the files could be saved on a network file server. Multiple users can access the files at the same time, with varying permissions, such as read-only. In cloud-based setups, the files are also saved on a server that can be accessed over a secure internet connection.

Our project goal is to create a secure way of creating a medium where a system can host files for other systems on the same network without the fear of unwanted systems on the same network accessing the files. The system which needs access to the host system must bypass time-based one-time password authentication to get access.

Keywords— File-sharing, Local Area Network, Sockets, Authentication, One-time password.

# Introduction

1. *File Sharing* -

The act of sharing or providing access to digital resources such as files, multimedia, graphics, software programs, photographs, and e-books is known as file sharing.

There are numerous ways to share files. The most popular techniques for file distribution, storage, and transmission are as follows:

* Removable storage units
* Networks with centralized file-hosting server installations
* Documents with hyperlinks designed for the Internet
* Peer-to-peer distributed networks

1. *Asymmetric Encryption* -

Asymmetric encryption makes use of two independent but correlated keys. The Public Key and Private keys are used for encryption decryption respectively. The Private Key is designed to be private, as the name suggests so that only the authenticated receiver can decode the message.

A cryptographic algorithm is responsible for asymmetric encryption. To generate a key pair, this approach utilizes a key generation protocol. Both the keys (public and private) are linked to one another.

The algorithm has 2 functions namely encryption and decryption. The encryption function encrypts data, and the decryption function decrypts it.

1. *TOTP (Time Based One Time Password)*

Time-based one-time passwords are a series of dynamically generated codes that change according to time. These one-time passwords generally have six-digit digits which depends on the time also these passcodes are temporarily valid and these changes or expire every 30 seconds.

When a user registers, a secret seed password is provided in the form of a QR code or in plaintext. TOTPs (and their seeds) can be used as soft tokens, which are shown on mobile devices, or as hardware security tokens. TOTP decodes a code from the secret using Greenwich Mean Time (GMT).

On top of static password authentication based on shared secrets, TOTPs are utilized for two-factor authentication (2FA) or multi-factor authentication (MFA). A valid TOTP must be submitted in an additional login form after the user has entered their username and password.

# *File Sharing Using Socket Programming -*

# Generally, a server is one that is on a specific port and is in charge of sending and receiving files. The client, on the other side, will attempt to connect to the server and will be able to send any form of a file.

# The socket module specifies how server and client machines can communicate at the hardware level by implementing socket endpoints on top of the operating system. Both connection and connectionless protocols are supported by the socket API.

# Python libraries such as ftplib and httplib, which implement application-level network protocols FTP and HTTP, respectively, provide Higher level support.

# When written a socket creates an instance for the socket class with the port and IP to listen and accept the connections. Generally, there is no need of developing multiple clients in every case as sometimes there might be a case where the communication of the client and server is either important or necessary.

# EXISTING SYSTEM

# Existing systems define the different ways and methods to share files via any medium. There is also a concept of sharing the files or the directory to target systems that are connected to the same network or Local Area Network. Some of the techniques that are generally used for this concept are as follows:

# By using socket programming

# Enabling Windows's file access over the same network.

# Enabling Ubuntu's personal file sharing.

# Usually, almost every operating system has this feature of sharing files with the target systems over the network.

# LIMITATIONS

Given how practical and helpful all of the suggested methods are for accessing the target file system, the access of the file sharing in the present techniques does not ensure any security because any computer linked in the same Local Area Network can access the target file system.

As technology and cybercrime evolve, security is more important than ever. There might be any number of systems that are connected to the network at any point in time and hence these systems will have access to the files hosted by the system.

# PROPOSED SYSTEM

# The intention of the proposed system will be to eliminate the above-mentioned limitation by adding some security authentication between the 2 systems.

# To enhance the file system’s security, the server or target system will authenticate the user or client system using either the secret token or the password. This strategy may be more secure than allowing file system access to every user on the same network.

# When the client attempts to access the target file system, a security step or logic is activated. Individuals are unable to access the target file system unless they have a secret token or are successfully authenticated

# IMPLEMENTATION

#### The implementation of the above-proposed idea is done using the python programming language and the creation of sockets using socket programming.

#### To replicate the system that wants to host the files, we have created a server file in python which is responsible for mocking the machine which in turn is responsible to host the files. To replicate the system that wants to receive the files, we have created a client file in python that is responsible for mocking the machine which is in turn responsible to receive the files or accept files from python, where the newKeys () function creates a new public-private key pair that can be used for encryption and decryption. For the generation of time-based OTPs, we have created a standalone authenticator application in python where it prompts for the QR code of the secret key for the registration of the user and then generated a new OTP whenever a user needs it. The client and server in this application will be communicated or contacted using the sockets.

#### To make sure that the created application is as per the project idea and theme, we have implemented a rule where the server and client can be connected only if they both are on the same network, as the authentication must be done for the file sharing over the network. The communication steps that take place during the secure file transfer process between the client and server along with its implementation details are below:

#### • The server starts hosting its server after prompting and it waits for the client system or the target system to be connected.

#### • Once there is an incoming connection to receive the files, the code prompts the user by showing the IP of the incoming client and asking if it needs to accept it. Once the server system agrees to accept the connection, it creates public and private key pair particular to that client. It sends a message to the client about the successful connection and its public key in the initial handshake.

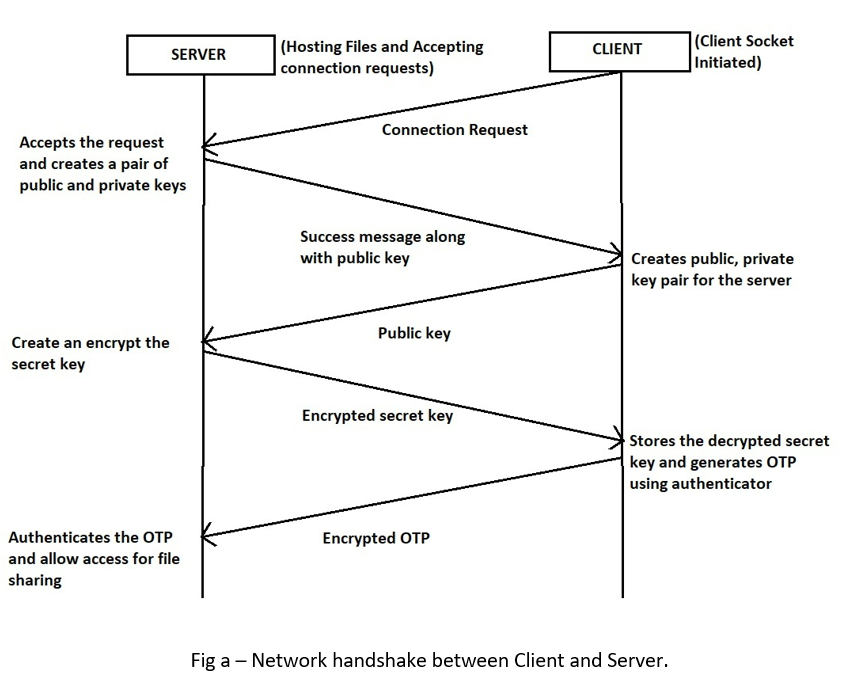
# • The client receives the public key and stores it. Like a server, even a client creates a private-public key pair and sends the public key of the server to the server.

# • The server generates a secret key by which the time-based one-time passwords can be generated. It encrypts the secret key using the public key of the client and then sends it to the client where the client receives the encrypted secret key, decrypts it with the help of its private key, and then stores it as a QR code image file which can be later used by the authenticator application.

# • Once the clients use the authenticator app to get the OTP, the client provides the OTP that the program prompts, and this OTP is encrypted using the server’s public key and shared via the network.

# • The encryption of the OTP is key because we assumed that the OTP is the key element for the authentication scenarios and stealing this otp would not make any sense to the main project idea. Also, the plan for the usage of time-based OTP instead of the normal OTPs has been generated from the line as consideration and assuming even if any attack has happened and anyone tries to decrypt the OTP, it takes them more than 30 seconds and the OTP would already be expired.

# • Once the server receives the OTP, it authenticates the OTP and displays the list of files to the client for the file transfer only after the successful OTP validation. The server does not allow any file transfer until and unless the client provides an accurate and valid OTP.



# The authenticator application is a python program that takes input as the QR code for the secret key. This secret is validated and registered for the future generation of passcodes. This secret key is converted to the 32-bit encoded string and used for the passcode generation based on this string. TOTPs are generated based on the standards of HMAC-based one-time passwords. This authenticator application uses the HMAC library to create a new HMAC standard passcode based on hashing of the current time and secret key. Also, this makes sure that this passcode will be same for the 30 seconds.

# Text Description automatically generated

# EVALUATION

# The evaluation of the project can be done by considering 3 major criteria:

# Evaluating if the generated one-time password is valid only for 30 seconds and if these are authenticated correctly. (Authentication).

# Evaluating if the Asymmetric key encryption is accurately and effectively implemented. (Encryption).

# Evaluating if the files are shared completely and successfully. (File sharing)

## AUTHENTICATION:

The authentication part is the heart of the project idea as the key part was to authenticate a user or a target system to access the server file system.

The evaluation of this Authentication is done in 3 steps:

• By checking if the authenticator app is accurately providing the OTP using the QR code generated as a secret key.

• By checking if the secret key is accurately encrypted after being encoded to a 32-bit string and using the public key provided by the server.

• By checking if the OTP is accurately decrypted using the private key and validate the time and code sent for the file access.

## ENCRYPTION:

The Encryption in this project is done using the RSA Asymmetric key encryption. The encryption is included in this project as sending the secret key for Time-based OTP in the network as plain text would not make any sense for authentication.

The evaluation for this encryption is done by checking if the generated unique key pairs can successfully encrypt the plain text to the cipher text and decrypt it back to the plain text.

## FILE SHARING

The important aspect to evaluate in this part is if the server provides access for its files to the target system only after the successful authentication.

After the successful authentication, the evaluation is done to check if only the selected files are sent to the target system and these files should be sent completely and accurately.

# RESULT

# The following results are achieved in the project:

# A successful connection is established by the client to the server only after the approval of the server or the host machine.

Text

Description automatically generated

The client prompts the user for the OTP from the authenticator app and displays a message Invalid OTP if the OTP is invalid.

Text

Description automatically generated

The files are shared successfully only after the successful OTP verification.

Text

Description automatically generated

# RISKS INVOLVED

Although the implemented methodologies have a secured authenticated way of sharing/accessing the files, these have some limitations/problems –

a. User might have to have the authenticator application always registered with the same secret key or the source code running always to get authenticated successfully.

b. As the time-based one-time password generated by the authenticator app has a very less expiry tie, the user has to enter the code within the time interval to get authenticated successfully.

# IMPLEMENTED CODE –

Below is the GitHub link that contains the executable project source code –

<https://github.com/sathwikgaddi/File-Sharing-over-Network-with-Authentication.git>

# References

David M'Raihi, Salah Machani, Mingliang Pei, & Johan Rydell. (2011). *TOTP: Time-Based One-Time Password Algorithm.*

Fruhlinger, J. (2022). *What is cryptography? How algorithms keep information secret and safe.*

Mehta, M. (2020). *What Is Asymmetric Encryption & How Does It Work?*