

CMSC 626 PRINCIPLES OF COMPUTER SECURITY

PROJECT REPORT

PEER TO PEER FILE SYSTEM

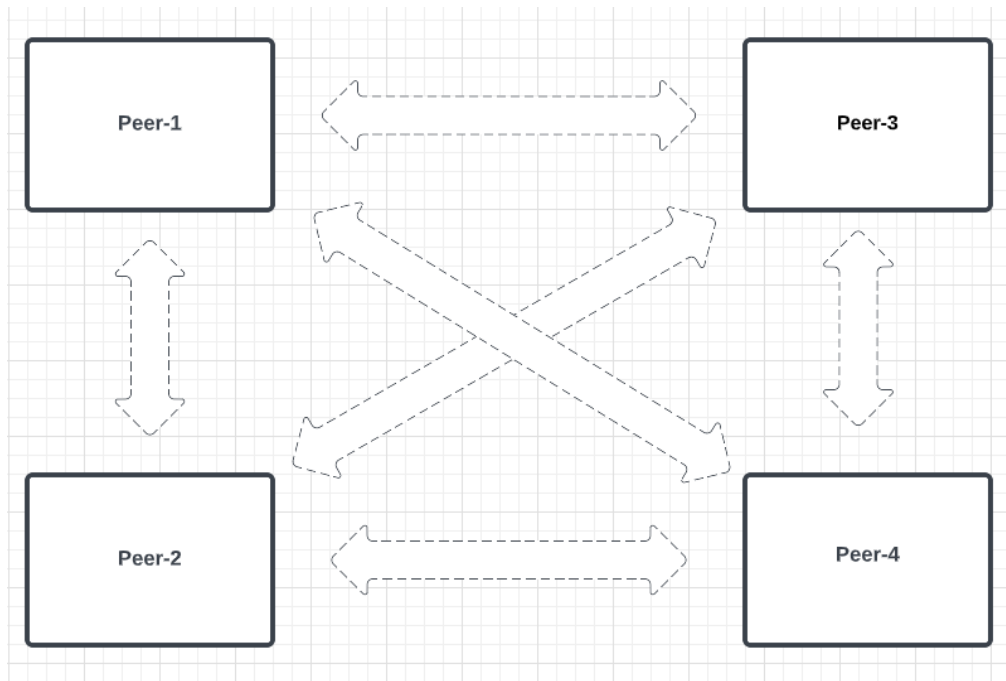
| | | |
|---------------------|---|----------|
| Jaideep Vallapuneni | - | jvallap1 |
| Ummadi Mounika | - | il55454 |
| Sowmith Chakilela | - | zz04733 |
| Dheeraj Reddy Anugu | - | danugu1 |

ABSTRACT:

Besides the client-server paradigm, the peer-to-peer model does not discriminate between clients and servers, rather depending on whether a node is seeking or delivering the services, a node might be either a client or a server. Every node is considered as a peer. Since the file system is encrypted, all data entered the system will also be encrypted before being placed in the files. Authorized users have access to create, remove, read, write, and modify files.

INTRODUCTION:

Our file system supports the same core set of file I/O operations that other file systems do. Files can be created, read, written to, and deleted by users. Along with setting permissions, they may share files and folders with other authorized users. To provide security and secrecy and to safeguard them from unauthorized users and attackers, the file system encrypts all file content and communication between the Peers. Updated Data will be displayed whenever the authorized user wants to access the data.



THREAT MODEL:

Malicious file server -

We have assumed that the user has the permission to read, write, and delete files that are stored on the file system. As the data gets encrypted while getting stored in the file system in case of a Malicious attack, the attackers will view the encrypted format of the content there by notifying the User/Owner will get notified about the attack.

Malicious user -

An authorized user may try to act on a file without having the required privileges, or an unauthorized user may try to access the files and make changes to them. This is partially handled by the ID and password required login feature.

FILE SYSTEM DESIGN:

Java is the programming language that has been utilized in this project to construct a peer-to-peer file system that is both secure and encrypted. To further distribute the load and provide resilience in the event that individual computers go offline, each object is replicated across multiple computers. Individual computer placement and retrieval requirements are more complicated than in client-server architecture. Through a JSON-based protocol, information is sent from one peer to another. We have established a secure and private channel for peer communication. We always encrypt the data, so there is no chance that an unauthorized person could learn specifics of the information that is transferred.

We have implemented a file locking service when multiple clients want to access the same file. For example, if two users want to update the same file the user who requests first will be given permission then system will lock that until the user finishes updating that file. Then the file is unlocked by the system so that the other user can access it. This is how we are handling concurrent read-write operations.

Here the list that the User can see after successful login: -

- 1)List all files
- 2)Create
- 3)Read
- 4)Write
- 5)Delete
- 6)Add User
- 7)Grant
- 8)Revoke
- 9)Create Directory
- 10)Delete Directory

List All Files –

This function will allow users to check all the files that user can Access

Create -

This function helps the user to create a file by a new name.

- Permission - The user who creates the file will have the access to read and write and also he will be the owner for that file that means the owner can give permissions to the created file to other users.

Write -

This functionality will help the user to write data into a new or an existing file in the file system.

- Permission - This can only be done by the user who has read-write permission.

Read -

This functionality will allow users to only read the data from an existing file.

- Permission - The user cannot edit the file as there is no permission to write.

Delete -

Delete is the function used when the user wants to delete the file.

- Permission - The permission for deleting the file is only given to the owner of that file.

Add User –

To add a new user to the file system.

Grant -

This is the functionality that can only be performed by the owners of that file.

- Permission - The owners can give permissions like read, write or can also make another user owner for a particular file.

Revoke -

This is also a functionality which can be performed by owners of that file.

- Permission - The owners can remove permissions from different users.

Create Directory -

The users can create a directory where they want to store all those files.

- Permission – This can be created only by an authorized user.

Delete Directory -

The Users can delete the directory. And the directory should be empty

- Permission – This can only be done by the owner of the directory

KEY ARCHITECTURE:

Our system concentrated primarily on giving users access with security. Everything has been encrypted using the AES encryption technology so that unauthorized users cannot access the data. Here, two distinct keys have been utilized.

One is used to deliver the message to the other Peers, while the Other is used to encrypt file contents. The user permissions will determine how the Peer grants access to the keys.

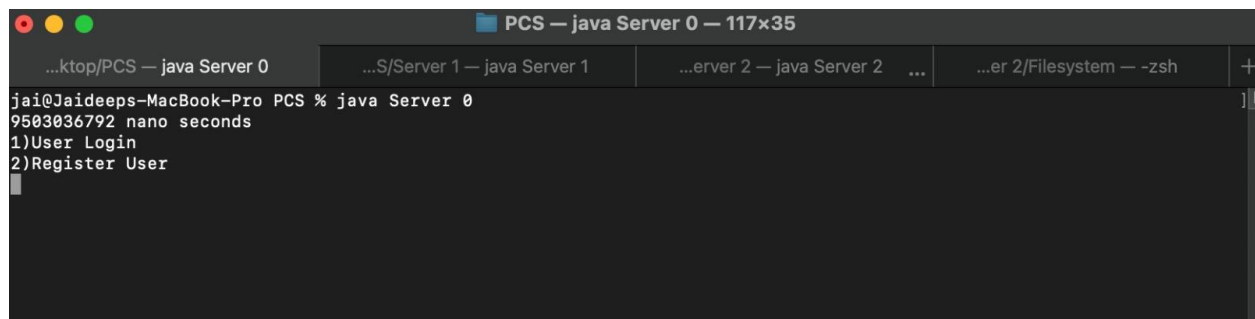
IMPLEMENTATION -

To store the user permissions, we are using meta data, where the permissions object is serialized and stored in the files. Every time the server starts it will retrieve the data from the files. The peer will continuously monitor the file system to find any malicious activity like creating or deleting the files.

When the user selects one of the options from the menu, it performs the operation and simultaneously it sends the operation request to all the peers and the peers will continuously monitor the socket to receive the request from their peers and perform that operation.

EXPERIMENTS&RESULTS:

We ran a **benchmark** test with 100k concurrent read and write requests and it took **9.503 seconds** as shown in the below screenshot.



```
PCS — java Server 0 — 117x35
...ktop/PCS — java Server 0
...S/Server 1 — java Server 1
...erver 2 — java Server 2
...er 2/Filesystem — -zsh
jai@Jaideeps-MacBook-Pro PCS % java Server 0
9503036792 nano seconds
1)User Login
2)Register User
```

This is the screenshot of User who have logged in successfully

```
1)User Login
2)Register User
1
Enter User Name
jai
Enter Password
123
login success
1)List all files
2)Create File
3)Read File
4)Write File
5)Delete File
6)Add User
7)Grant permissions
8)Revoke permissions
9)create directory
10)Delete directory
```

This is the screenshot of user who have created a file successfully

```
1)List all files
2)Create File
3)Read File
4)Write File
5)Delete File
6)Add User
7)Grant permissions
8)Revoke permissions
9)create directory
10)Delete directory
2
Enter the File Name
abc.txt
success file created
1)List all files
2)Create File
3)Read File
4)Write File
5)Delete File
6)Add User
7)Grant permissions
8)Revoke permissions
9)create directory
10)Delete directory
```

CONCLUSION -

The main idea of our project is to provide users a safe and secret way to access the files and perform various functionalities. We have developed a Peer-to-Peer Encrypted Distributed file system by using Java Programming Language. We have created a file system which has High security as there will be User authentication and encryption. As we have created a Decentralized File System, there will be a lot of safety from different server attacks. The encryption algorithms will help in preventing the attacks from Unauthorized users.

REFERENCES:

- [1] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," Proceedings First International Conference on Peer-to-Peer Computing, Linköping, Sweden, 2001, pp.
- [2] J. Sen, "Peer-to-peer networks," 2012 3rd National Conference on Emerging Trends and Applications in Computer Science, Shillong, 2012, pp.
- [3] A. Hac, "A distributed algorithm for performance improvement through file replication, file migration, and process migration" , " IEEE Transactions on Software Engineering" , 2009

GITHUB LINK:

<https://github.com/Jai975/P2P-File-System>