CMSC 626 PROJECT PROPOSAL

Group 8 Ummadi Mounika, Jaideep Vallapuneni, Dheeraj Reddy Anugu, Sowmith Chakilela

Department of CSE, University of Maryland, Baltimore County

Introduction and Overview

Distributed file systems are becoming more and more popular. They are preferable to traditional centralized file systems because of their fault tolerance, availability, scalability, and performance. We are developing a distributed file system that is safe and encrypted using the Object-Oriented programming language. We can exchange files with numerous users using this file system while yet ensuring that all activities are secure and carried out by authorized users. Our system has a fundamental set of Input Output features, just like all file systems. Authorized users have access to create, delete, read, write files. Concurrent users can also conduct operations on the Distributed File System. We are using AES and RSA to encrypt the requests from client to server and to maintain encryption of the files within the servers.

Client and Server Connection:

Client and server exchange data using RSA keys. As the client's and server's public keys are available at each other respectively the transfer of the data can be done securely using encryption and decryption at their ends.

Server has all the files of all the users. Every file has a random key generated automatically while the file was created. So, the filename and file contents are encrypted using this key. All these keys are stored inside a file(center.txt) which was encrypted using AES, so the data inside center.txt cannot be seen by anyone in the server. This center.txt contains all the data related to users and their access permissions to different files in the server. So, when the user login, server looks inside the center.txt file to get the filenames which the user had permissions. Only the files which the user has certain permissions are visible.

The center.txt file also contains the data about latest operation, user, and timestamp of the respective files. So that malicious file server cannot be able to manipulate the files.

The server runs on multiple machines and all the servers connect to each other and the operations done on different servers converge and latest data is available at every end. This will make sure that client will always see the latest version of the file system.

Format of center.txt:

An object of array data structure will be stored in the file. This object have data about the files. Each object in the array contains filename, List<UsersRead>, List<UsersWrite>, timestamp, owner, last operation by.

Sharing a file through permissions:

When a user gave certain permission to another user, the center.txt file will be updated with that permission details. So, the second user will get the access and can perform read, write, delete operations on the file. The owner of the file can revoke the permissions in future if needed and the same will be reflected in the center.txt.

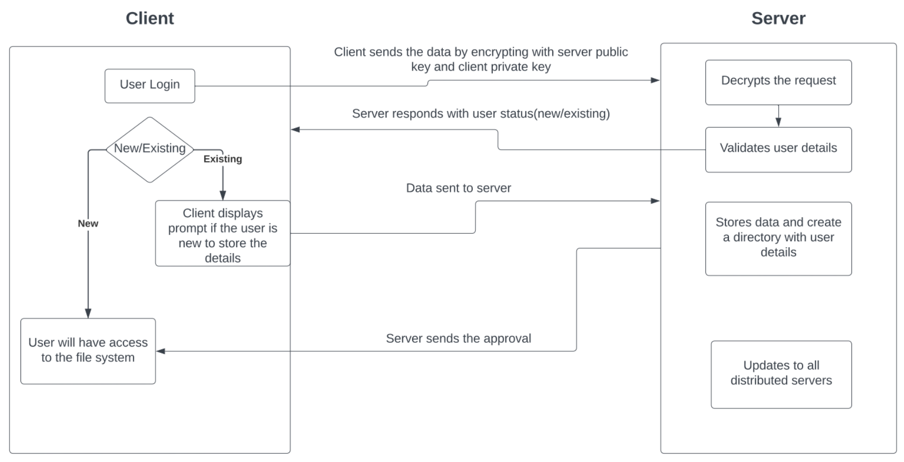
Handling concurrent requests:

To handle the file system concurrently we use shared lock and exclusive lock mechanism. So, the user who wants to read a file will ask for read lock which is shared lock. So, the server grants the lock to user. Shared lock can be requested by multiple users parallelly to read the file until unless it doesn't have any exclusive lock on it.

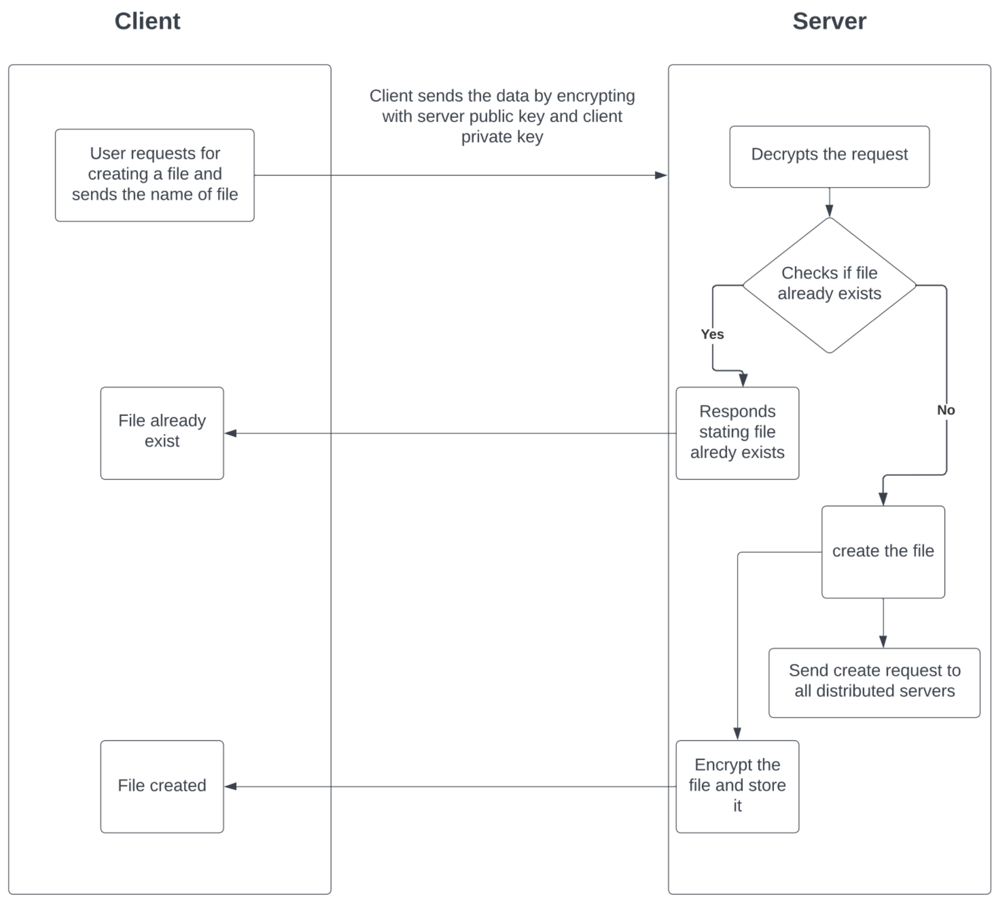
To write to a file user will ask for exclusive lock from server. The server will check if this file has any type of lock on it. If it doesn't have any locks on it the server returns with exclusive lock or else will reject the request saying file is already in use.

Performing Login, create, read, write, and delete operations:

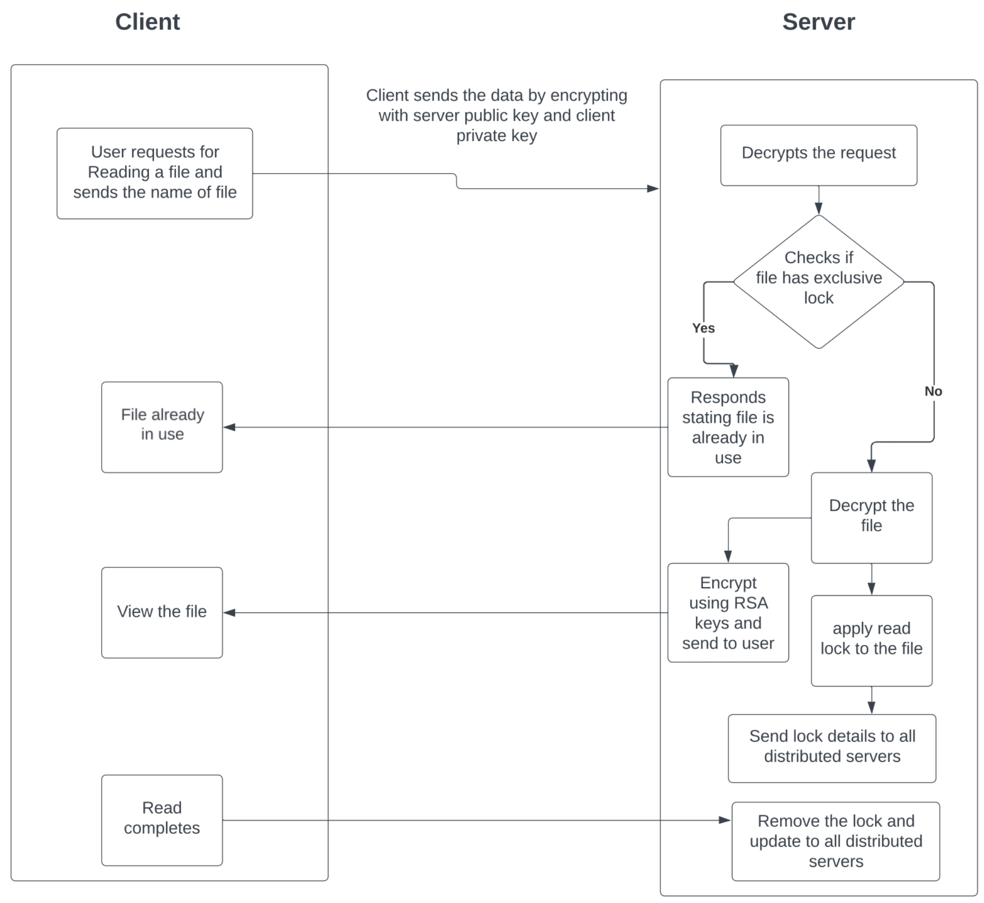
Login: The user details are stored in server. When user enters the login details they were sent to server for validation. So, if the user is an existing one user will have the access to the files which has permission. If the user is new a new directory will be created with the user details has access defined for the user. Update all the servers with the new user details.


Client
Server
User Login
Client sends the data by encrypting with server public key and client private key
Decrypts the request
Server responds with user status(new/existing)
New/Existing
Existing
Validates user details
New
Data sent to server
Client displays prompt if the user is new to store the details
Stores data and create a directory with user details
User will have access to the file system
Server sends the approval
Updates to all distributed servers

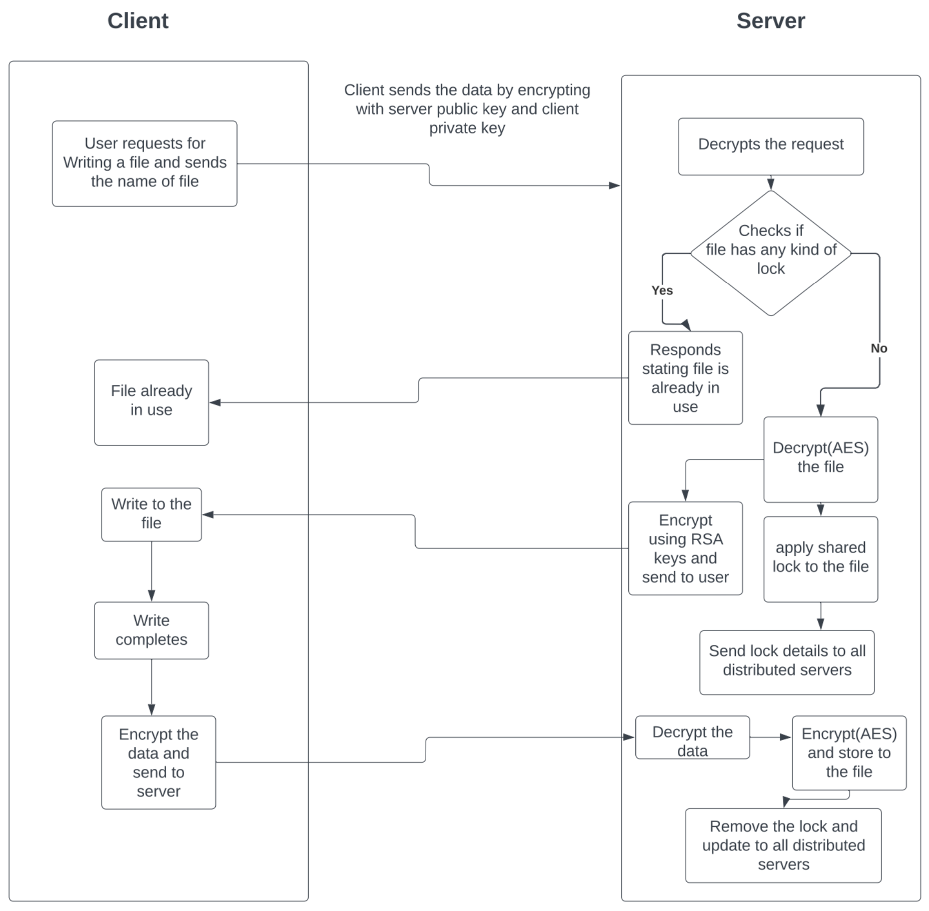
Create: user will give a command “touch filename.txt” at client’s command prompt. This command data will be encrypted using RSA mechanism (first encrypt with client private and then again encrypt with server public key) and sent to server. This request from client is decrypted at server and will create a file. While creating the file a random key is generated and encrypts the filename. This key is stored in center.txt along with filename and user created along with timestamp.


Client
Server
User requests for creating a file and sends the name of file
Client sends the data by encrypting with server public key and client private key
Decrypts the request
Checks if file already exists
Yes
No
Responds stating file alredy exists
File already exist
create the file
Send create request to all distributed servers
Encrypt the file and store it
File created

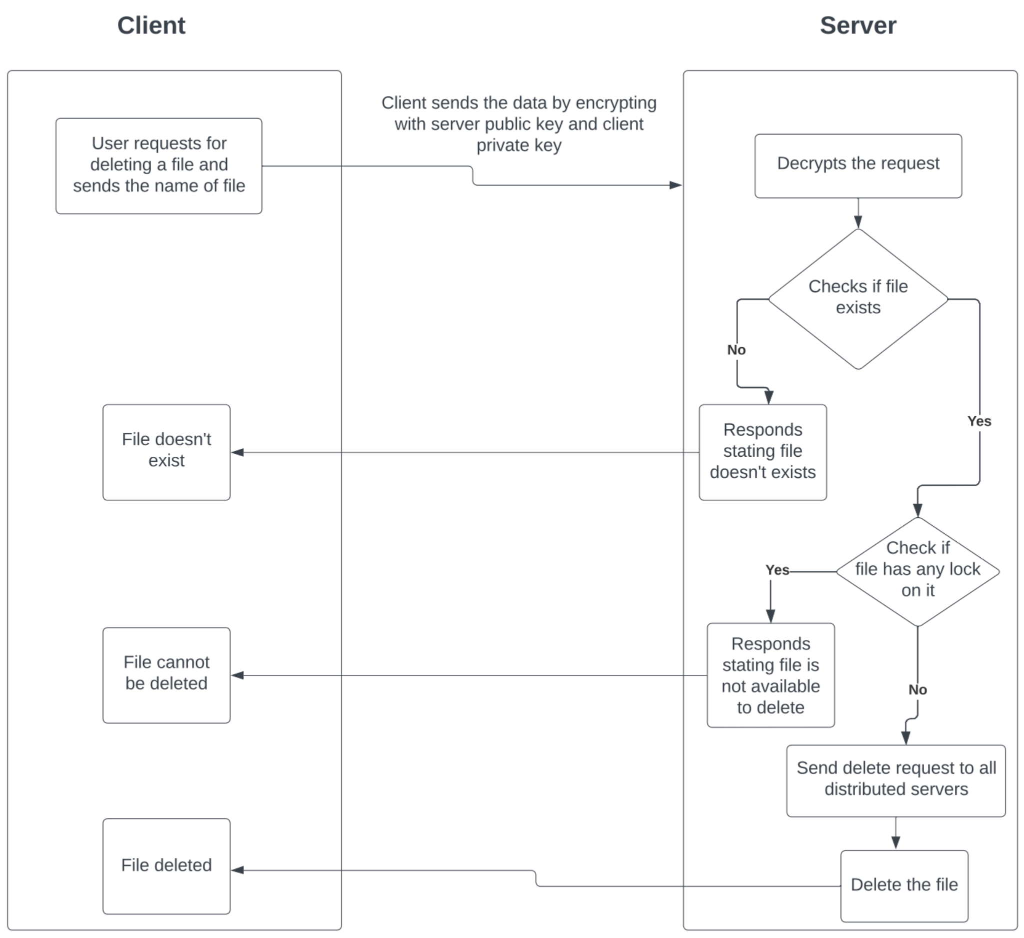
Read: user will give a command “cat filename.txt” at client’s command prompt. This command data encrypted and sent to server using same RSA mechanism. At server first the filename will be searched inside the center.txt and permissions are validated for user. If user has read permission, then the respective key of the file is used to decrypt the data. Then this data is sent to client using RSA encryption. And the lock details are updated to all the servers.


Client
Server
Client sends the data by encrypting with server public key and client private key
User requests for Reading a file and sends the name of file
Decrypts the request
Checks if file has exclusive lock
Yes
No
Responds stating file is already in use
File already in use
Decrypt the file
Encrypt using RSA keys and send to user
View the file
apply read lock to the file
Send lock details to all distributed servers
Read completes
Remove the lock and update to all distributed servers

Write: user will give a command 'echo "some text to write in file" > filename.txt ' at client's command prompt. This command data encrypted and sent to server using RSA mechanism. At server filename is searched in the center.txt and if the user had permission to write data, and check if it has any lock on it. If there is no lock update the lock on file then send lock details to all servers then key will be retrieved from center.txt and the data will be written to the file. After writing the data again file will be encrypted and the center.txt will be updated with timestamp and operation. Remove the lock and update write details and lock to all servers


Client
Server
Client sends the data by encrypting with server public key and client private key
User requests for Writing a file and sends the name of file
Decrypts the request
Checks if file has any kind of lock
Yes
No
Responds stating file is already in use
File already in use
Decrypt(AES) the file
Encrypt using RSA keys and send to user
apply shared lock to the file
Write to the file
Write completes
Send lock details to all distributed servers
Encrypt the data and send to server
Decrypt the data
Encrypt(AES) and store to the file
Remove the lock and update to all distributed servers

Delete: user will give a command "rm filename.txt" at client's command prompt. This command data encrypted and sent to server using RSA mechanism. At server filename is searched in the center.txt and if the user had permission to delete file, then key will be retrieved from center.txt and the file will be deleted.


Client
Server
User requests for deleting a file and sends the name of file
Client sends the data by encrypting with server public key and client private key
Decrypts the request
Checks if file exists
No
Yes
Responds stating file doesn't exists
File doesn't exist
Check if file has any lock on it
Yes
No
Responds stating file is not available to delete
File cannot be deleted
Send delete request to all distributed servers
Delete the file
File deleted

While performing all these above operations all the connected servers will be updated with latest changes.