Report: Distributed File System Implementation

The Distributed File System(DFS) is implemented using RESTful Webservices in Python using web.py.

Implemented 4 services/components. They are:

1. Distributed File Access
2. Security Service
3. Directory Service
4. Lock Service

Distributed File Access

The Distributed File Access (File Server), gets the encrypted file path from the client proxy along with the encrypted ticket. File server decrypts the ticket using the Server Encryption Key and session key is retrieved. This is used to decrypt the file path that client sends.

Based on the get and post calls, the file server reads the content of the file path given and writes the file content to the file that client sends respectively. The messages that are sent back from the file server (file content if it is a get call, write success message if it is a post call) are also encrypted using the session key and returned.

Security Service

The Security Service or the Authentication server, is in-charge of the authentication of the client and the generation of the session key. It authenticates the client using login and password. Once the client is logged in, then a session key is generated. This is send back to the client in the token

The response from the Authentication Server is a token with session key and the ticket. The ticket is the encrypted session key using Server Encryption Key. Server Encryption Key is the key shared between all the services and not the client. The token itself is encrypted using the client's password and sends back to the client

Token

Session Key	Ticket

Ticket is the session key encrypted using Server Encryption Key

The Token is encrypted using client's password

Directory Service

Directory Service gets the encrypted file name from the client along with the ticket. The ticket is decrypted using the Server Encryption Key and session key is retrieved. Using the session key, the filename is decrypted. Then it retrieves the path of the file is retrieved along with the port of the file server in which the file resides. This is encrypted using the session key and is then sent back to the client.

Lock Service

Lock service gets the encrypted filename from the client along with the ticket. Ticket is decrypted using Server Encryption Key and session key is retrieved. The session key is used to decrypt the filename. Then it checks if the file is locked or not. If the file is not locked, then it locks the file. If it is locked, it sends back the message saying that the file is locked by another user. The message that is sent back to the client is also encrypted using the session key

Client Proxy

The client interacts with the user and gets the user input. According to the user inputs, the client proxy is in-charge of calling other services. The client proxy takes care of it and user does not know what all calls are happening in the backend. Encryption of the message to be sent and received to the services are done using session key generated using Security Service by the client proxy. Client proxy is implemented as a function inside Client.py. The flow inside client proxy is as follows:

If the operation is Read:

1. Authenticate the user and get the keys (explained the key and encryption in Security Service)
2. Get the filename
3. Encrypt the filename
4. Send the encrypted file name to Directory Service
5. Get the encrypted filepath from the directory service
6. Decrypt the filepath
7. Send encrypted filepath with filename to file server
8. Get the encrypted content of the file as response
9. Decrypt the content
10. Display it

If the operation is Write:

1. Authenticate the user and get the keys (explained the key and encryption in Security Service)
2. Get the file name
3. Encrypt the file name
4. Send the encrypted file name to Directory Service

5. Get the encrypted filepath from the directory service
6. Send the encrypted filename to lock server to check if the file is locked or not
7. If the file is not locked lock server will lock the file and sends the response
8. Send the file path to File server along with content, both encrypted
9. Get the response
10. If the write is success, call the lock server to unlock the file