DESIGN DOCUMENT

-DISTRIBUTED SYSTEMS

Objective

The project aims to design a secure file system that allows distributed system nodes to access remote files on remote file servers using RPCs (Remote Procedure Calls). This document will help familiarize the reader with key terminologies, the problem description, workflow, and model characteristics.

1 Introduction

1.1Distributed File Systems (DFS)

A DFS is a file system that is spread over multiple file servers. This method of storing and accessing files is based on a client-server architecture. Several remote file servers can be accessed by several distributed system nodes in the network, granted if they possess the appropriate authorization.

After the distributed system node retrieves a file from the server, the file appears as a normal file on the respective machine, and the user can work with the file in the same ways as if it were stored locally on the device. When the distributed system node finishes working with the file, it is returned over the network to the file server, which keeps the modified file for retrieval later.

1.2Advantages and Disadvantages of DFS

Advantages:

1. Easier to distribute documents to multiple clients.
2. Provide a centralized storage system, so that client machines are not using their resources for file storage.

Disadvantages:

1. It requires that file servers be stateful. Stateful file servers have a distinct disadvantage over stateless file servers in the event of a failure.
2. It is not easy to provide adequate security in distributed systems because the nodes and the connections need to be secured.

2 Problem Description

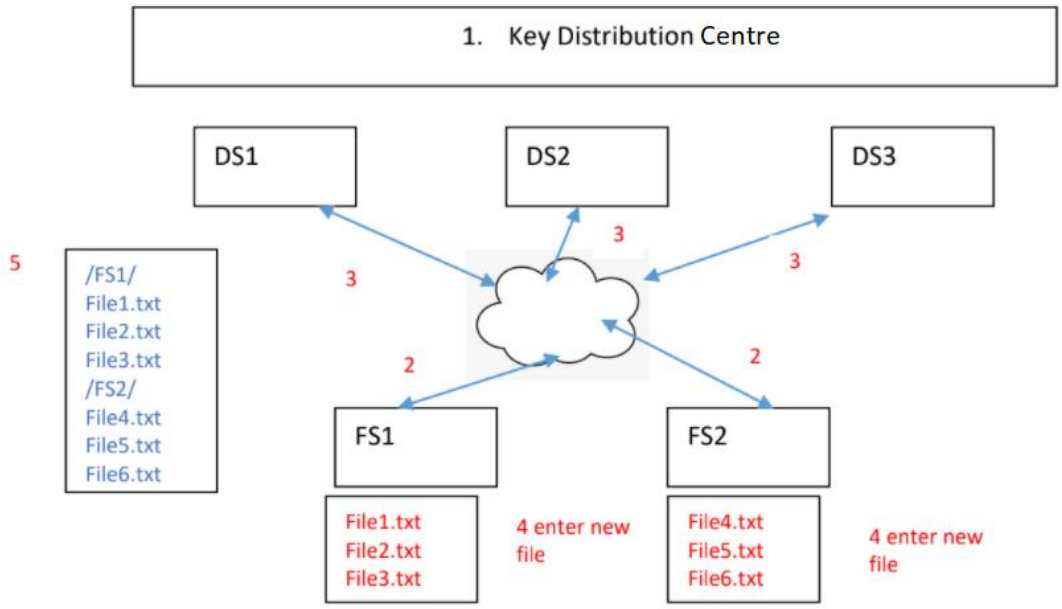


Fig 1. Schematic Diagram of DFS implemented
(DS – Distributed System Node)
(FS – File Server)

2.1 Components

1. Distributed System Node: These represent the "clients" in DFS architecture. They allow users to interact with files stored on File Servers and execute various commands.
2. File Server: These store files in the file system. They allow Distributed System Nodes to interact with the files using Remote Procedure Calls only with proper authentication.
3. Key Distribution Centre: The KDC shares symmetric keys with the nodes in the system.
   The file servers register with the KDC for the files that are stored.
   The KDC grants the distributed system nodes session keys for mutual authentication.

2.2 Working of System

1. All the file servers FS_i and the distributed nodes DS_i must be assigned unique ids and share symmetric keys with a Key Distribution Server (KDC).
2. All the File servers register with the KDC for the files that they store.
3. The Distributed nodes authenticate with the KDC to generate session key using the symmetric key mutual authentication protocol.

4. The DS must mutually authenticate with the file server and get the files which they display in a folder for the file server. Needham-Schroeder protocol will be used for mutual authentication.
5. FS_i must register any new file that is created with KDC.
6. File creation must be communicated to all DS_i as a folder entry seen on the shell.

2.3Functionalities

1. Implementation of RPCs to establish a secure connection between distributed nodes and file servers in the DFS.
2. File Server registration and set-up.
3. Distributed Node registration and authentication with servers.
4. Commands that are listed below:
    - pwd – list the present working directory
    - ls – list contents of the file
    - cp – copy one file to another in the same folder
    - cat – display contents of the file

3 Workflow

3.1 Block Diagram

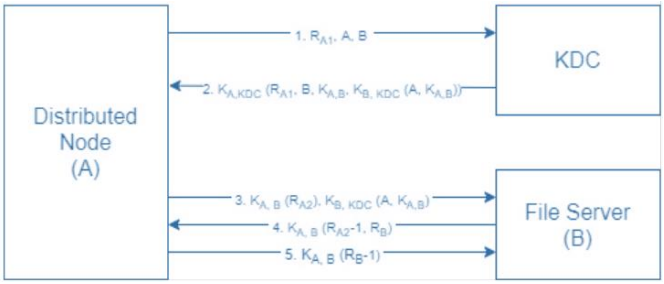
1. Registration of Distributed Node with KDC.

Distributed Node registers with KDC and obtains a unique ID and session key.

2. Registration of File Server with KDC.

File Server registers with KDC for the files it stores.

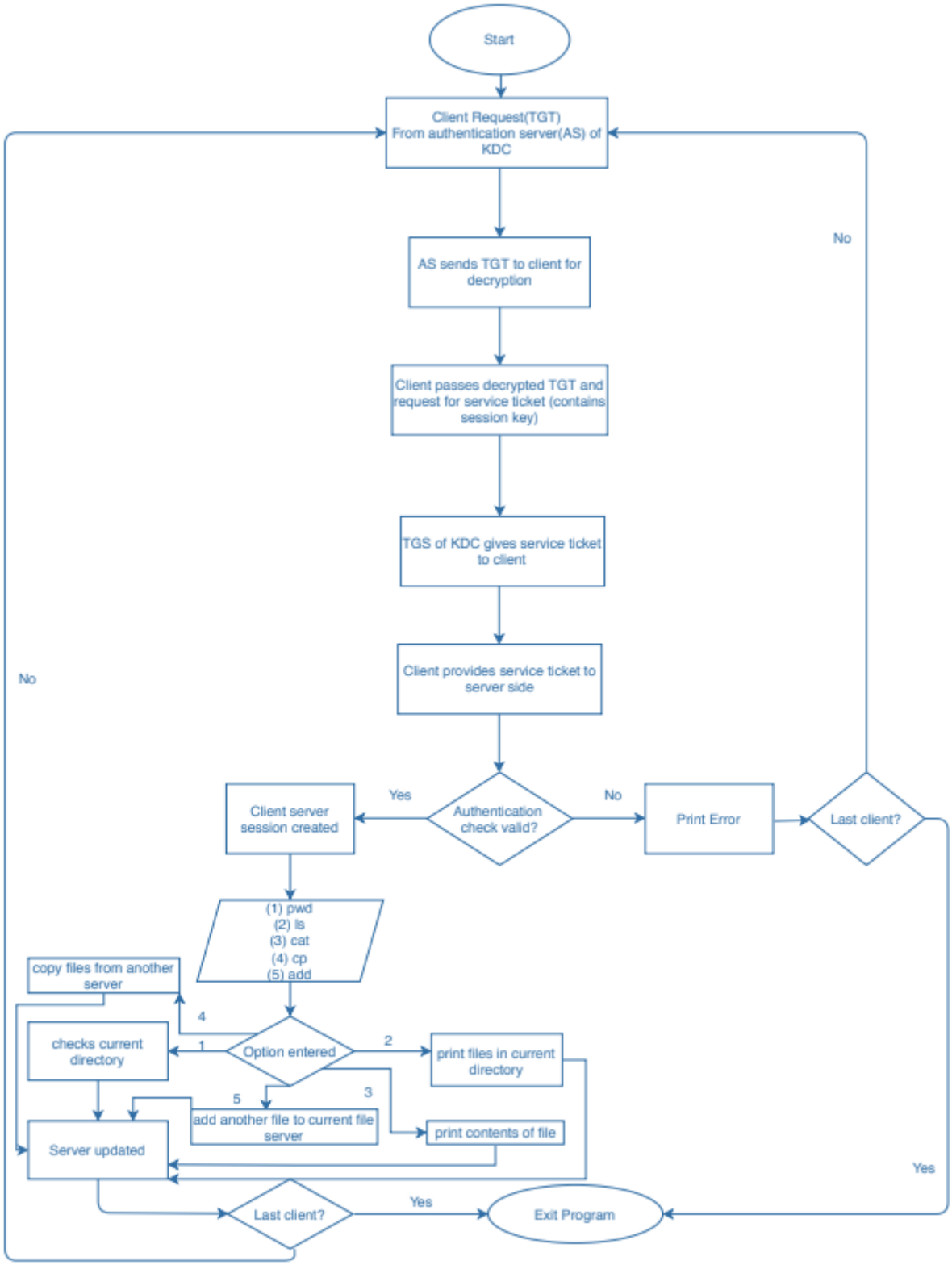
3. Establishment of connection between File Server and Distributed Node with mutual authentication.



i. Node A requests the KDC for a connection establishment with B. A sends token R_{A1}.
ii. KDC responds to this request by sending a message which is a result of the following:
    a. $K_{B, KDC}$ encrypts A, $K_{A, B}$
    b. $K_{A, KDC}$ encrypts R_{A1}, B, $K_{A, B,}$ and the result of step (a).
iii. A sends a message to B, which results from encrypting token R_{A2} and the message received from KDC in step (ii).
iv. B performs decryption on the received message using $K_{B, KDC}$ to obtain $K_{A, B}$ for communication between the two nodes. B then sends R_{A2}-1 and R_B to A as proof that it can decrypt $K_{B, KDC}$.
v. A reciprocates by sending a message which contains R_B-1 encrypted by $K_{A, B}$. This is done to prove A possesses $K_{A, B}$.

3.2Flowchart

The following flowchart describes the process, including the Client Request procedure to the Authentication check. Once the client-server is created, the commands – pwd, ls, cat, cp, and add can be run. At each turn, the server is updated up until the last client is running. Once that is complete, the program is exited!



4 Principal Entities

Client: The client acts on behalf of the user and initiates communication for a service request.

Server: The server hosts the service the user wants to access.

Key Distribution Center (KDC): the authentication server is separated into three parts – database (db), Authentication Server (AS), and Ticket Granting Server (TGS).

Authentication Server (AS): The AS performs the desired client authentication. If the authentication happens successfully, the AS issues the client a ticket called TGT. This ticket assures the other servers that the client is authenticated.

Ticket Granting Server (TGS): The TGS is an application server that issues service tickets as a service.

5 Protocols and Libraries

1. Socket - The Python interface is a straightforward transliteration of the Unix system call and library interface for sockets to Python's object-oriented style: the socket() function returns a *socket object* whose methods implement the various socket system calls.
2. Fernet - Python supports a cryptography package that helps us encrypt and decrypt data. The fernet module of the cryptography package has inbuilt functions for the generation of the key, encryption of plaintext into ciphertext, and decryption of ciphertext into plaintext using the encrypt and decrypt methods, respectively.
3. Pickle - The pickle module implements binary protocols for serializing and de-serializing a Python object structure.

6 Platform

This project was carried out on Linux Ubuntu virtual machine (ver 20.04).