Secure Personal Cloud

Alpha Cube

November 24, 2018

Abstract

The project is to implement a cloud based file system in which user has complete control over encryption schema of data that is uploaded to the server.

1 Introduction

This project aims to build a fully secured cloud based file sharing system with features like virtual directory like navigation system, real-time synchronization with encryption and decryption taking place only on client-side and periodic sync. The sytem is built on a django backend and uses CSS and Javascript for professional looking frontend. There are three encryption schemes deployed in this model namely : AES, TripleDES, Twofish.

2 Basic Features

2.1 Web Client

Web Client is completely backed by **Django**[1]. It provides a Login and Signup page. Once a user logs in, it is directed to the personal drive, showing the files and folders. Each folder in turn contain sub-folders and files [2]. The prominent features of webclient include deleting a file or a folder, and also downloading or viewing a file. This operation requires the user to provide the **key to decrypt** the file as the server has no copy of it.

2.2 Linux Client

It uses **Rest API**[3] framework of django, which makes it independent from Web Client. The salient feature of Linux Client is the ability to upload files and folder to the database after encrypting the files with the scheme specified by the user.

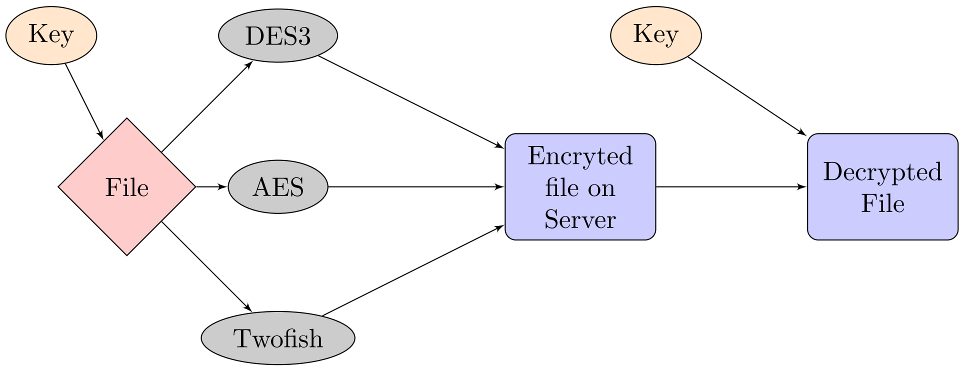
It gives easy access in terms of **spc** command. It has various attributes like Upload, Download, Sync, Observe, Status, etc.

Observe defines the root directory at the client. **Status** give the difference between the root folder and the data of client in the database. Those differences can then be resolved using **sync** command.

2.3 Encryption and Decryption

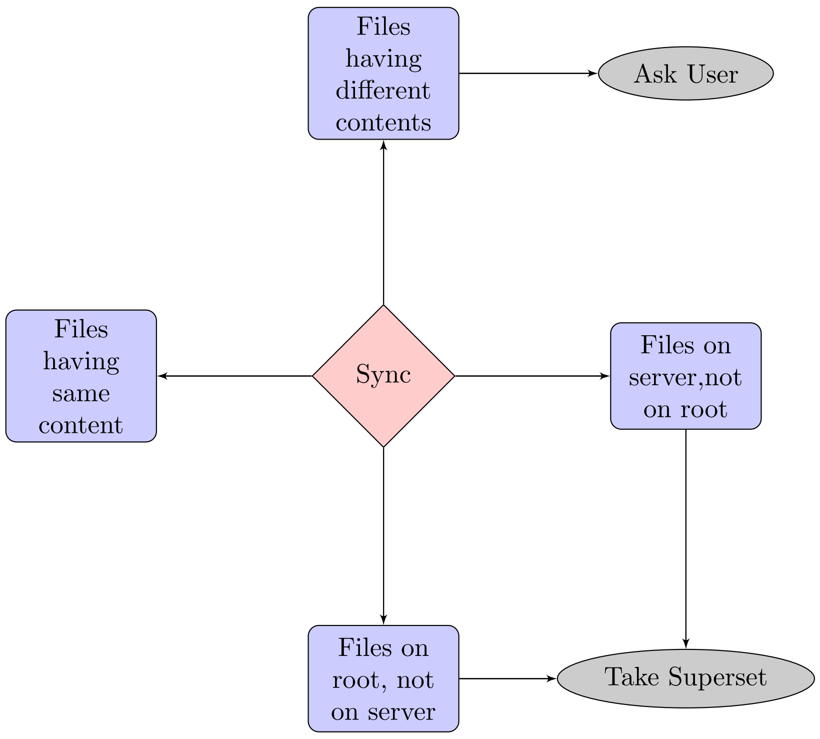
Encryption is done through 3 advanced techniques, namely, **DES3, AES and Twofish**[4]. This is the feature which provides security to the setup as the key isn't stored on server.

Linux client asks for the key only once and then store it locally with the client, for subsequent use till the user logs out. Web Client asks for the key to view or download a file.



2.4 Synchronization

A root folder is defined by the client at client's system using **Observe**. Whenever **Sync** is called, it ensures that the server and the root folder have same **set of files and content**.

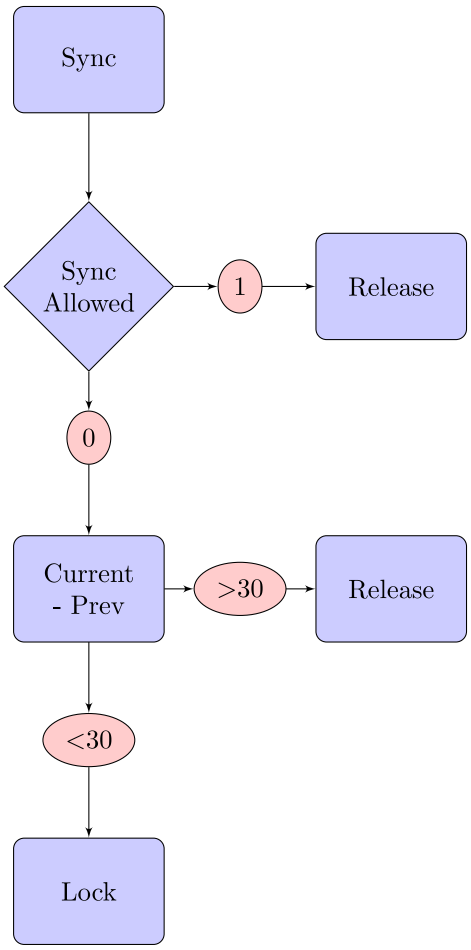

Files having different contents
Ask User
Files having same content
Sync
Files on server,not on root
Files on root, not on server
Take Superset

Server contains **md5sum** of each decrypted file. If there's a file which was edited, it'll give a different md5sum for the same. The client is then prompted asking for which file to keep.

2.5 Race Conditions

When synchronization of a client A is going on, other clients are not allowed to sync from same user i.e. the sync operations is **locked** for the period. A special case occurs when syncing of A die due to some reason, hence a condition of **deadlock** arises. To handle this, we have allowed locking of sync operation only for a fixed upper limit(viz. 30 seconds here).

This is implemented through a Model named 'Time' which has two attributes, a boolean variable to store if sync is allowed at that point of time, and a timestamp for the beginning of last sync operation.



3 Additional Features

The database acts as a virtual directory with **ls** and **cd** and **pwd** commands. Periodic sync of files using **Linux Daemons**. Automatically sync between client and server periodically with time period 60 seconds and if files differ a popup is shown.

References

[1] https://tutorial.djangogirls.org/en/

[2] https://www.w3schools.com/

[3] https://www.django-rest-framework.org/

[4] https://github.com/ricmoo/aes-js