Secure Data Upload and Retrieval from Shared Storage

CS-699 Project Report

Submitted by

Darshit Bimal Gandhi
(22M0824)

Shamik Kumar De
(22M0822)

Abstract

Nowadays, with the increase in the amount of data and to share that data with multiple users, there arises a need for storing it properly in some shared storage. Here 'properly' will generally mean that a large amount of data should be accurately stored and there should be easy availability of data to multiple users.

But normally, the data is stored in PlainText mode in the shared storage. This leads to the generation of serious concern on data confidentiality in a system.The files should be stored in an encrypted form in the shared storage and only the authenticated users should be able to access/decrypt them. Also, the users should be able to search the files without losing privacy. The users always search their documents through filename (which is in PlainText), which may lead to some privacy breaches of as sometimes file names can also reveal the file contents.

Thus, a solution to this will be storing files in an encrypted form on a system and retrieving and decrypting it as and when required by an authenticated user. Also, the file name should not be in PlainText form.

To implement these functionalities, our model has distributed modules for different schemes leading to an establishment of an integrated and secure system.

Description

Our project comprises a basic software implementation of encryption of files before uploading them on a secure storage server to prevent breach of data and further decryption of the files before downloading them on demand of a desired user.

Technology Stack :

1 Java (Spring Boot)

2 PostgreSQL Database

3 HTML

4 CSS

5 JavaScript

6 Latex

7 Eclipse and VS Code (IDE)

8 Git

Entities :

1. Data Handler :

The data handler is the entity who will upload the data in the shared storage in an encrypted form and also will be able to delete some file from the shared storage when it needs. Also, the file names will be encrypted before storing it in the shared storage so there is no confidentiality breach.

2. User :

The user can retrieve the data files from the shared storage in the encrypted form and decrypt them anytime he/she choses to only if he/she is authenticated. If the user is not authenticated, then he/she will not be able to decrypt it.

3. Shared storage :

It's the standard place to store all the data files provided by the data handler.

Modules :

1. User Registration.

In this module, new user registration is done using various fields like name , email id , password and contact details. All the input validation will be done in this module and the user will not be able to register with invalid data..

2. User Login.

This module enables the pre-registered users to login into their accounts. A list of authenticated users is maintained and this is used when users log in and request services.

3. Encrypting File Name.

Encrypting the file as well as the file name is necessary for privacy as file names might lead to the revelation of file contents and the type. Thus, encrypting the file name is a good option for preventing any data leak.

4. File content encryption and upload.

The contents of data file are encrypted using a symmetric shared key and then will be uploaded to the shared storage.

5. Data Retrieval from the secure storage.

In this module, the user will request for the file from the secure storage. This input is encrypted and compared with the encrypted filenames that are currently present in the secure storage.

6. File Content Decryption.

The contents of the file will be decrypted with the help of shared secret key before being downloaded in the user machine.

7. Viewing Files.

There is an option for the admin to view the files which have been uploaded.

8. Deletion of Files.

There is an option for the admin to delete the files which have been uploaded before.This has been done for the sole discretion of the admin which files are not required anymore.

Flow Of The File Handling

- Data Handler selects the file to be uploaded. Before being stored in the secured storage, the file content and the file name are encrypted with the specified encryption algorithm.
- To retrieve a file , the user enters the filename. This input is encrypted and compared with the encrypted filenames that are currently present in the secure storage.
- When a match occurs, the specified file is decrypted with the specified decryption algorithm and is sent to the user.
- At the user side, corresponding javascript is run and file gets downloaded on the user's machine.

Achievement

With the help of the different modules we have achieved the following:

- Successful signup and login of a user.
- Successful encryption of the file along with its filename and uploading the file on the secure storage by the data handler. Currently, we can upload the file with extensions like txt, sh and pdf to the server.
- Successful decryption of the file along with its filename and downloading the file on the user's demand.
- An option for viewing and deleting files for the data handler.

Possible future work

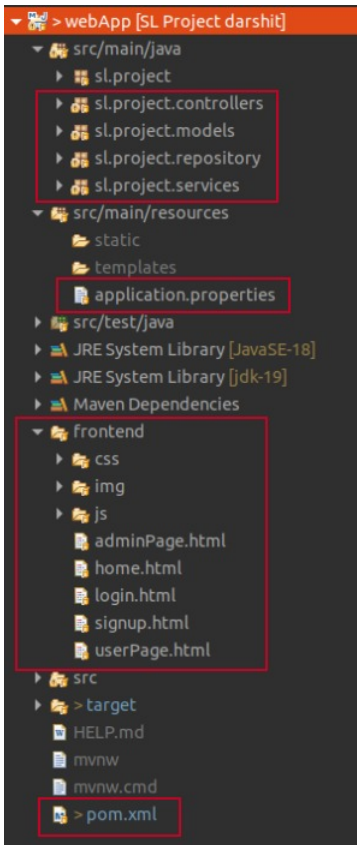
Possible works that can be implemented are as follows:

- We can extend our application so that it can store files with various other extensions like jpg, docx, png, etc.
- The synchronous shared key can be dynamically generated.
- Users may be provided with a reset password link.
- An implementation of data upload for the user can be made .
- Admins may have a management model to grant or revoke permissions to/from the users.

Directories and File System

Four layered structure have been used in the Springboot application.

- **Model** : It is a class-level annotation. It is used to denote a class as a model. We used @model across the application to mark the beans as Spring's managed components.
- **Services**: This encapsulates the main business logic . For instance the LoginSignupService is responsible for creating an account, and performing the required logic in order to register the user or login.
- **Repository**: These were used between the service layer and the model layer. For instance, in the LoginSignupRepository methods were created that contained the code to read/write from the database.
- **Controllers**: This layer acts as a gateway between the input and the domain logic. It mainly decides what to do with the input data and how to output the response data.



```
> webApp [SL Project darshit]
  src/main/java
    sl.project
    sl.project.controllers
    sl.project.models
    sl.project.repository
    sl.project.services
  src/main/resources
    static
    templates
    application.properties
  src/test/java
  JRE System Library [JavaSE-18]
  JRE System Library [jdk-19]
  Maven Dependencies
  frontend
    css
    img
    js
    adminPage.html
    home.html
    login.html
    signup.html
    userPage.html
  src
  > target
  HELP.md
  mvnw
  mvnw.cmd
  > pom.xml
```

Our Directory Structure.

Algorithms Implemented

- Key has been chosen as a random string of bytes which has been hashed using SHA-1. The AES key has been formed from the hashed key. The final product has been used to encrypt and decrypt file content and file names.
- AES-128 encryption and decryption is done with CBC block mode and PKCS5 padding.
- AES-128 is a block cipher with key size as 128 bits , in which the input and output block size is also 128 bits.
- In Cipher Block Chaining, the current cipher text block depends on all the previous blocks. This will ensure that a plainText block will not be encrypted to same CipherText block everytime.
- PKS5 padding obeys the following rules:
  - The number of bytes to be padded equals to ”8 - numberOfBytes(PlainText) mod 8”.
  - All padded bytes have the same value - the number of bytes padded.

Compilation/ Running Instructions

- Run **'sudo systemctl status postgresql'** on the terminal. Check if status is active , otherwise run **'sudo service postgresql restart'** to start PostgreSQL.
- Run the Springboot from your preferred IDE.
- Run the HTML-CSS-JS web application in your browser.