SecureVault: Secure File Upload and Encryption System

Siddharth Movaliya, Kawal Ostwal

May 13, 2025

1. Problem Definition

With the rapid adoption of cloud storage, ensuring the confidentiality, integrity, and user control over stored data has become a critical concern. Most cloud providers, while offering some level of server-side encryption, do not give end users fine-grained control over how their data is encrypted or managed. This poses a risk in scenarios involving sensitive data, regulatory compliance, or the need for strong data ownership guarantees.

Background

Cloud storage services like AWS S3 provide highly scalable and available storage solutions, but the default security measures may not satisfy users with advanced privacy needs or compliance constraints (e.g., in healthcare, finance, or government sectors). Standard server-side encryption solutions often lack transparency and leave key management entirely in the provider's hands. This project addresses these limitations by providing a secure file management system where:

- Users authenticate using Google OAuth.
- Upon first login, users configure their own AWS IAM credentials, enabling full control over their cloud storage.
- Files are encrypted client-side or app-server-side before being uploaded.
- Users can choose from multiple encryption methods (SSE-S3, SSE-KMS, or Customer-managed keys).
- The system supports secure file downloads with automatic decryption before delivering to the user.

By shifting control to the user and supporting flexible encryption options, this solution enhances data privacy, aligns with best practices in cloud security, and empowers users to manage their data securely in the cloud.
This project implements a system that allows users to upload files to AWS S3 with encryption, ensuring data security. The system assumes that users have AWS accounts and require a secure interface for file management. The goal is to provide a user-friendly platform for secure file uploads, encryption, and management while adhering to best practices in cloud security.

2. Conceptual Design

The system is composed of the following components:

- **Frontend:** Built using React and Next.js, it provides a user-friendly interface for file uploads, settings, and file management.
- **Backend:** Implements server-side logic using Next.js Server Actions and Prisma ORM for database interactions.
- **Database:** PostgreSQL is used to store user data, AWS configurations, and file metadata.
- **AWS Integration:** AWS S3 is used for file storage, and AWS KMS is used for encryption key management.

Architecture:

- The architecture consists of a three-tier design:
  1. **Presentation Layer:** The frontend interacts with users for file uploads and settings.
  2. **Application Layer:** The backend handles authentication, encryption, and AWS interactions.
  3. **Data Layer:** The database stores user configurations and metadata securely.
- AWS S3 and KMS are integrated into the application layer for secure file storage and encryption.

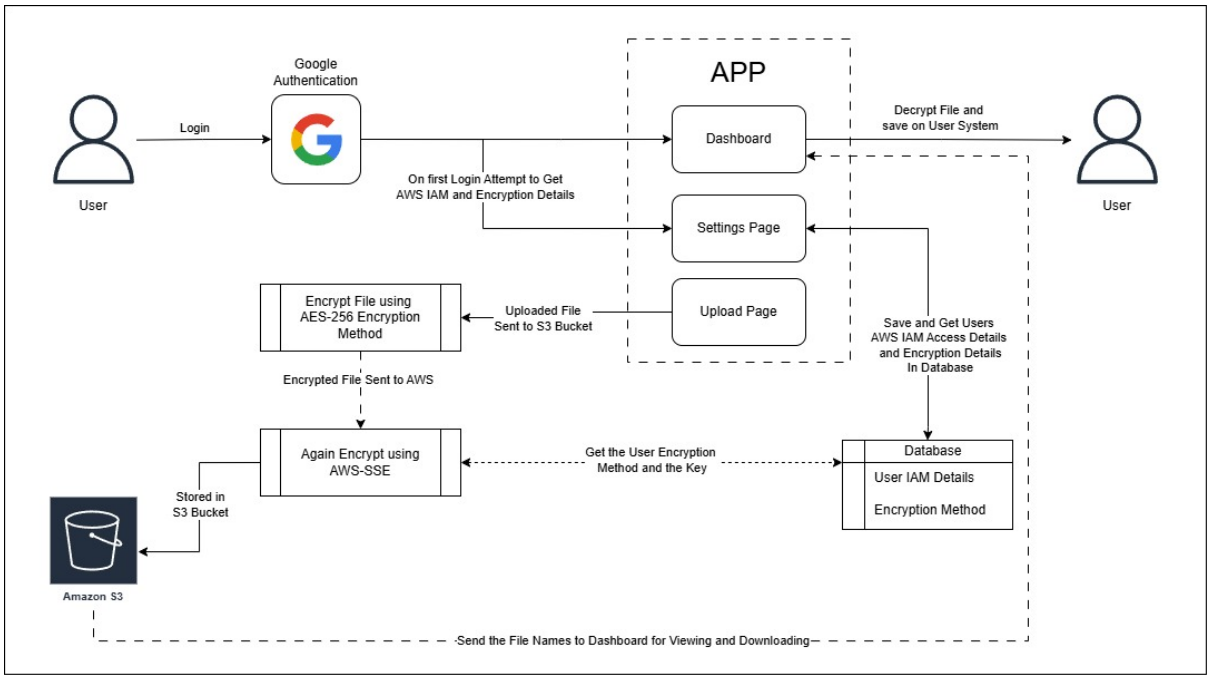

Figure 1: System Architecture Diagram

3. Security Measures

- **Authentication:** Google OAuth is used for secure user authentication.
- **Encryption:** Files are encrypted using AWS-managed keys, KMS, or custom keys before upload.
- **Data Integrity:** File metadata is stored in PostgreSQL with referential integrity enforced.
- **Access Control:** AWS credentials are encrypted and stored securely in the database.
- **Audit Logging:** All critical actions, such as file uploads and AWS configuration changes, are logged for auditing.

4. Implementation Description

Development Platforms/Tools:

- **Frontend:** React, Next.js, Tailwind CSS
- **Backend:** Prisma ORM, AWS SDK
- **Database:** PostgreSQL

Major Data Structures:

- `TUser`: Stores user details.
- `TUserAWSConfig`: Stores AWS configuration for each user.
- `TDecryptedFiles`: Represents decrypted file metadata.

Major Classes/Methods:

- `uploadFileToS3`: Handles file encryption and upload to AWS S3.
- `saveAWSCredentials`: Saves and validates AWS credentials.
- `getUserAWSConfig`: Fetches AWS configuration for a user.
- `saveEncryptionMethod`: Updates encryption settings for a user.

5. User Guide

Installation:

1. Navigate to the project directory and install dependencies using `npm install`.
2. Configure environment variables in `.env` and `.env.local`.
3. Run database migrations using `npx prisma migrate dev`.
4. Start the development server using `npm run dev`.

Usage:

1. Sign in using Google OAuth.
2. Configure AWS credentials and encryption settings in the settings tab.
3. Upload files via the upload tab. Files will be encrypted and stored in AWS S3.
4. Manage files in the dashboard tab.

6. Future Enhancements

- **Multi-file Uploads:** Add support for uploading multiple files simultaneously.
- **File Sharing:** Implement secure file sharing with access control.
- **Versioning:** Enable file versioning to track changes and restore previous versions.
- **Mobile App:** Develop a mobile application for on-the-go file management.
- **Advanced Analytics:** Provide insights into file usage and storage patterns.

7. Self Evaluation

The design goals of secure file upload and encryption have been accomplished. The system provides a seamless user experience for managing encrypted files. AWS integration ensures scalability and reliability. The use of Prisma ORM simplifies database interactions. However, future enhancements could include multi-file uploads and file sharing features.

8. References

1. AWS SDK Documentation: `https://docs.aws.amazon.com/sdk-for-javascript/`
2. Prisma Documentation: `https://www.prisma.io/docs`
3. Next.js Documentation: `https://nextjs.org/docs`
4. Tailwind CSS Documentation: `https://tailwindcss.com/docs`

9. Contribution

- **Siddharth Movaliya:** Led the development of the core web infrastructure and backend logic, and contributed significantly to the report preparation.
- **Kawal Ostwal:** Designed the application's user interface and user experience, and was actively involved in preparing the documentation, presentation materials, and report.