# Secure File Sharing System

## 1. Introduction

This document outlines the security measures implemented in the Secure File Sharing System, including encryption techniques, authentication mechanisms, key management, and best practices ensuring data confidentiality and integrity.

---

## 2. Encryption Method

- Algorithm: AES-256-GCM (Advanced Encryption Standard - 256 bit key, Galois/Counter Mode)
- Purpose: Encrypt all files before saving to disk to protect data at rest from unauthorized access.
- Process:
    - A random 16-byte initialization vector (IV) is generated for each file encryption.
    - The plaintext file content is encrypted using AES-256-GCM with the secret key and IV.
    - Authentication tag is generated and stored with the encrypted data for tamper detection.
    - The encrypted file stored on disk contains the IV, authentication tag, and ciphertext concatenated.
- Decryption:
    - Upon download, the stored file is read, IV and authentication tag extracted.
    - The content is decrypted using the same AES key and IV after verifying the authentication tag.
    - This ensures confidentiality and integrity of the file content.

---

## 3. Encryption Key Management

- The AES secret key is stored securely in the server environment variable (`AES_SECRET_KEY`).
- The key is 32 bytes long, fulfilling AES-256 requirements.
- The key is never hardcoded in the source code or exposed to clients.

- In production, the key is managed securely following organizational security policies and rotated periodically.
- Access to the key is restricted to the server environment to prevent leaks.

---

## 4. Authentication and Authorization

- The system uses JWT (JSON Web Tokens) for user authentication and API access control.
- Upon login, a JWT token is generated and signed with a secret (`JWT_SECRET`).
- The client must send this token in the Authorization header (`Bearer <token>`) for protected routes like file upload and download.
- The server verifies token validity and user identity before processing requests.
- JWT tokens have an expiration time (e.g., 1 hour) to reduce risk of token theft.
- Unauthorized or unauthenticated requests receive appropriate HTTP error responses.

---

## 5. Additional Security Measures

- Input Validation and Sanitization:
    - File names are validated to prevent directory traversal attacks (e.g., no `../` sequences).
- Secure Transport (Recommended):
    - Use HTTPS in production to encrypt data in transit between client and server, protecting tokens and files against interception.
- Error Handling:
    - Avoid revealing sensitive information in error messages. Return generic error responses to clients.
- In-memory User Store (Demo Only):
    - User credentials are hashed with `bcrypt` and stored in-memory for simplicity. A real deployment should use a secure database.

---

## 6. Summary

This secure file sharing system protects user files by encrypting them before storage, authenticating and authorizing users with JWT, and managing encryption keys securely in the server environment. Combined with common web security best practices, this approach ensures confidentiality, integrity, and controlled access to user data.