PROJECT REPORT

# Secure File Storage System with AES

## Abstract

This project focuses on developing a secure file encryption and decryption system using the Advanced Encryption Standard (AES) algorithm for both data.Develop a local command-line tool that securely encrypts and decrypts files using AES-256, ensuring both confidentiality and integrity. AES is a symmetric encryption technique known for its speed, efficiency, and strong security. It outperforms older algorithms like DES and RSA in both performance and reliability. The system proposed in this project employs AES to encrypt and decrypt the files, ensuring confidentiality and integrity during transmission using SHA algorithm.. Encryption converts plaintext into unreadable ciphertext using a secret key, while decryption reverses the process with the same key, making it accessible only to authorized users.The system is implemented in python and features a user-friendly interface for securely encrypting data.

## Introduction

In the digital era, data security has become a major concern for individuals, businesses, and organizations worldwide. With the rise in cyber threats such as hacking, phishing, ransomware, and unauthorized data breaches, protecting this information has become more important than ever. File encryption and decryption play a crucial role in ensuring the confidentiality, integrity, and security of such data. Encryption is the process of converting readable data, known as plaintext, into an encoded format called ciphertext. This transformation is achieved using complex cryptographic algorithms and an encryption key. The primary purpose of encryption is to prevent unauthorized access to the data, ensuring that only authorized users with the correct decryption key can revert the ciphertext back to its original readable format. Decryption is the reverse of encryption, where the encrypted data is converted back to its original form using a predefined key and algorithm.

### Hash Functions

Another crucial aspect of cryptography is the use of hash functions. Hash functions play a vital role in ensuring data integrity and authentication. Hash functions take an input (or 'message') and generate a fixed-length output known as the hash or digest. Hash functions are extensively used in digital signatures and message authentication codes (MACs) to ensure that data has not been tampered with during transmission. One of the most widely used hash functions is SHA-256, which is part of the SHA-2 family and is utilized in blockchain technology, secure file sharing, and digital certificate verification.

## Tools Used

1. Python
2. cryptography library (for AES/Fernet)
3. hashlib (for SHA-256 hashing)
4. os, json, datetime,CLI

## Steps Involved in Building the Project

1. Use AES-256 for authenticated encryption.

   File Encryption Workflow

- Take the input file to be encrypted.
- Generate a **random 12-byte IV**
- Read file content into memory.
- Apply AES-256-GCM encryption.
- Save encrypted output as **filename.txt.enc**

2. Store metadata

Store all metadata entries in a Python list
Convert metadata list to JSON string.
Encrypt JSON using AES
Save encrypted blob as metadata.enc.

Metadata structure

```
json
{
 "original_file": "...",
 "encrypted_file": "...",
 "timestamp": "...",
 "sha256": "..."
}
```

3. Compute SHA-256 hash of encrypted file
   Append encrypted metadata entry to metadata.enc

4. File Decryption Process
   - Read the encrypted file
   - Extract IV and ciphertext
   - Decrypt with same key
   - Verify authentication tag
   - Save the original file
5. Command Line Interface (CLI) Implementation

```
Generating keys - python progam.py genkey
Encrypting - python pogram.py encrypt sample_fille.txt
Verifying integrity - python program.py verify sample_file.txt.enc
Decryting - python program.py decrypt sample_file.txt.enc
```

## Conclusion

The Secure File Storage System using AES-256 successfully demonstrates the application of modern cryptography to protect sensitive information. By integrating AES-256-GCM encryption, SHA-256 hashing, and secure metadata handling, the system ensures both confidentiality and integrity of stored files. The solution is lightweight, efficient, and practical for real-world usage. It provides a strong foundation for secure data storage and can be extended with features such as cloud integration, multi-user support, advanced key management, or full graphical interfaces. Overall, the project highlights the importance of encryption in safeguarding digital assets and serves as a valuable learning experience in cybersecurity and secure software development.