Secure File Encryption/Decryption Script – Full Explanation with Index

This document provides a detailed explanation of a Python script that performs secure file encryption and decryption using AES-GCM and Argon2-based key derivation.

# Table of Contents

1. Overview  
2. Key Features  
3. Modules Used  
4. Core Components  
 4.1 Master Key Derivation  
 4.2 Encryption Functions  
 4.3 Decryption Functions  
 4.4 File Handling  
5. Command Line Interface  
6. Encryption Workflow  
7. Decryption Workflow  
8. Error Handling  
9. Security Practices  
10. Summary

## 1. Overview

This script allows for secure encryption and decryption of files using AES-GCM for encryption and Argon2 for key derivation, ideal for high-security applications.

## 2. Key Features

- AES-GCM encryption for confidentiality and integrity  
- Argon2id key derivation with memory-hard properties  
- Password-protected file key  
- Secure nonce usage and key storage  
- Robust logging and exception handling

## 3. Modules Used

- argparse: for command-line argument parsing  
- secrets: for secure random byte generation  
- getpass: for secure password input  
- logging: for debug/error logging  
- cryptography: for AES-GCM encryption/decryption  
- argon2: for secure password-based key derivation

## 4. Core Components

The script is organized into key functional components including key derivation, data encryption, decryption, and secure file handling.

## 4.1 Master Key Derivation

Uses Argon2id to derive a 256-bit key from the user-provided passphrase and salt. This ensures strong resistance to brute force and GPU cracking.

## 4.2 Encryption Functions

AESGCM is used with a 12-byte nonce to encrypt either plaintext or entire file data. Nonce and ciphertext are returned for secure storage.

## 4.3 Decryption Functions

Decrypts using the same AESGCM class and nonce. Errors in decryption raise specific DecryptionError exceptions.

## 4.4 File Handling

Reads input files securely, encrypts contents, and writes to output with error checking. Handles combined nonce+ciphertext format for simplicity.

## 5. Command Line Interface

Accepts --encrypt or --decrypt flags with input/output file paths, passphrase, and keyfile. Uses argparse for safe and user-friendly parsing.

## 6. Encryption Workflow

- Generate a random file key  
- Encrypt file data  
- Derive master key from passphrase  
- Encrypt file key using master key  
- Store salt + nonce + encrypted key in keyfile  
- Delete original input file

## 7. Decryption Workflow

- Read keyfile (salt + nonce + ciphertext)  
- Derive master key  
- Decrypt file key  
- Use file key to decrypt actual data  
- Save decrypted file to output

## 8. Error Handling

Custom exceptions for encryption and decryption errors. Critical logging with fallback to sys.exit(1) for unrecoverable issues.

## 9. Security Practices

- Uses secrets module for all random data  
- Promotes passphrase input via getpass (avoiding CLI leakage)  
- Deletes original unencrypted files post-encryption  
- AES-GCM ensures both encryption and integrity verification

## 10. Summary

This script provides a complete, secure workflow for encrypting and decrypting files using modern cryptographic standards, designed with both usability and security in mind.