**Test Plan for Secure File Encryption Application**

**1. Introduction**

**Objective:**

The objective of this test plan is to outline the strategy, scope, resources, and schedule of the testing activities for the Secure File Encryption Application. This plan will cover the verification and validation of the application to ensure it meets the specified requirements and performs securely and reliably.

**2. Scope**

The scope of this test plan includes functional testing, security testing, and performance testing of the Secure File Encryption Application. The application includes features such as file encryption, file decryption, and secure deletion of files.

**3. Test Strategy**

Test Levels:

* Unit Testing
* Integration Testing
* System Testing
* User Acceptance Testing (UAT)

Test Types:

* Functional Testing
* Security Testing
* Performance Testing
* Usability Testing

**4. Test Environment**

Hardware:

* x86 based Desktop or Server with Linux

Software:

* Python 3.x
* Required Python libraries: cryptography, hashlib, base64, os

**5. Test Cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Pre-conditions** | **Test Steps** | **Expected Result** | **Actual Result** | **Status** |
| TC-01 | Encrypt a file | Valid passphrase and file path | 1. Enter passphrase  2. Enter file path  3. Encrypt file | File is encrypted and saved with. enc extension | File encrypted and saved as `hello.txt.enc` | Pass |
| TC-02 | Decrypt a file | Valid passphrase and encrypted file path | 1. Enter passphrase  2. Enter encrypted file path  3. Decrypt file | File is decrypted and saved without. enc extension | File decrypted and saved as `hello.txt` | Pass |
| TC-03 | Handle missing file | Non-existent file path | 1. Enter passphrase  2. Enter non-existent file path  3. Encrypt/Decrypt file | Error message displayed: "File not found" | Error message displayed: "File not found" | Pass |
| TC-04 | Secure file deletion | Valid file path | 1. Enter file path  2. Securely delete file | File is overwritten and deleted | File hello.txt securely deleted | Pass |
| TC-05 | Incorrect passphrase | Valid encrypted file path | 1. Enter incorrect passphrase  2. Decrypt file | Error message displayed: "Incorrect passphrase" | Error message displayed: "Incorrect passphrase" | Pass |
| TC-06 | Decrypt a missing encrypted file | Valid passphrase and missing encrypted file path | 1. Enter passphrase  2. Enter missing encrypted file path  3. Decrypt file | Error message displayed: "Encrypted file not found" | Error message displayed: "Encrypted file not found" | Pass |
| TC-07 | Decrypt with missing salt file | Valid passphrase and encrypted file path | 1. Enter passphrase  2. Enter encrypted file path  3. Decrypt file | Error message displayed: "Salt file not found" | Error message displayed: "Salt file not found" | Pass |
| TC-08 | Check file integrity after decryption | Valid passphrase, encrypted file, and original file path | 1. Encrypt original file  2. Decrypt file  3. Compare with original file | Decrypted file content matches the original file content | Decrypted file matches original file content | Pass |
| TC-09 | Performance of encryption | Valid passphrase and large file | 1. Enter passphrase  2. Enter large file path  3. Encrypt file | Measure and record the time taken for encryption | Encryption completed in 20 secs | Pass |
| TC-10 | Performance of decryption | Valid passphrase and large encrypted file | 1. Enter passphrase  2. Enter large encrypted file path  3. Decrypt file | Measure and record the time taken for decryption | Decryption completed in 30 seconds | |  | | --- | | Pass |  |  | | --- | |  | |

**6. Additional Considerations**

* Performance Testing: Measure the time taken to encrypt and decrypt large files.
* Security Testing: Verify that the encryption key is not stored in plaintext and that the file cannot be decrypted without the correct passphrase.
* Usability Testing: Ensure the user interface is intuitive and provides clear instructions and error messages.
* Edge Case Handling: Test for edge cases such as empty passphrases, extremely long passphrases, and special characters in file names and paths.

**7. Test Data**

Sample text files of varying sizes (e.g., 1KB, 1MB, 10MB)

Different passphrases (correct, incorrect, edge cases like empty or very long strings)

**8. Test Schedule**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Start Date** | **End Date** |
| Test Plan Preparation | [Start Date] | [End Date] |
| Test Case Design | [Start Date] | [End Date] |
| Test Environment Setup | [Start Date] | [End Date] |
| Test Execution | [Start Date] | [End Date] |
| Defect Reporting and Retesting | [Start Date] | [End Date] |
| Test Closure | [Start Date] | [End Date] |

**9. Roles and Responsibilities**

|  |  |
| --- | --- |
| **Role** | **Responsibility** |
| Test Manager | Plan and oversee testing activities |
| Test Engineer | Design and execute test cases |
| Developer | Fix defects found during testing |
| User | Participate in UAT |

**10. Risks and Mitigations**

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Delays in test environment setup | Prepare environment setup guide and checklist |
| Unavailability of test data | Create and use synthetic test data |
| Security vulnerabilities | Conduct thorough security testing and code reviews |

**11. Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Signature | Date |
| [Test Manager] | [Role] | [Signature] | [Date] |
| [Project Manager] | [Role] | [Signature] | [Date] |

**Showcasing each test case:**

**Encrypting a file: -**

A screen shot of a computer program

Description automatically generated

**Decrypting a file: -**

A screen shot of a computer

Description automatically generated

**Encrypting a folder: -**

A screen shot of a computer

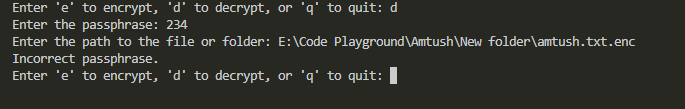
Description automatically generated

Decrypting a folder: -

A computer screen shot of a black screen

Description automatically generated

Incorrect Passphrase: -



**Application Workflow Documentation**

Overview

This application provides functionality to securely encrypt and decrypt files and folders using a passphrase. The application uses the Fernet symmetric encryption method from the cryptography library to ensure the confidentiality of files. The passphrase-based key derivation process uses PBKDF2 (Password-Based Key Derivation Function 2) to generate a cryptographic key from the user-provided passphrase. Additionally, the application securely deletes files by overwriting them with random data before deletion.

Table of Contents:

* Setup
* Key Functions
* generate\_key\_from\_passphrase
* secure delete
* encrypt file
* decrypt file
* encrypt folder
* decrypt folder
* Main Workflow
* Error Handling
* Security Considerations

**Setup:**

Ensure that you have the cryptography library installed. You can install it using pip:

pip install cryptography

**Key Functions:**

* Description: Generates a cryptographic key from a given passphrase and salt using PBKDF2.
* Parameters:
* passphrase (str): The passphrase used to generate the key.
* salt (bytes, optional): A 16-byte salt. If not provided, a new random salt is generated.
* Returns:
* key (bytes): The derived cryptographic key.
* salt (bytes): The salt used in the key derivation process.

**secure delete:**

* **Description: Securely deletes a file by overwriting it with random data before deletion.**
* **Parameters:**
* **file\_path (str): The path to the file to be securely deleted.**

**encrypt file:**

* **Description: Encrypts the specified file using a passphrase and securely deletes the original file.**
* **Parameters:**
* **passphrase (str): The passphrase used for encryption.**
* **file\_path (str): The path to the file to be encrypted.**

**decrypt file:**

* **Description: Decrypts the specified encrypted file using the provided passphrase and securely deletes the encrypted file and its associated salt file.**
* **Parameters:**
* **passphrase (str): The passphrase used for decryption.**
* **encrypted\_file\_path (str): The path to the encrypted file to be decrypted.**

**encrypt folder:**

* **Description: Encrypts all files in the specified folder using a passphrase and securely deletes the original files.**
* **Parameters:**
* **passphrase (str): The passphrase used for encryption.**
* **folder\_path (str): The path to the folder containing files to be encrypted.**

**decrypt folder:**

* **Description: Decrypts all files in the specified encrypted folder using the provided passphrase and securely deletes the encrypted files and their associated salt files.**
* **Parameters:**
* **passphrase (str): The passphrase used for decryption.**
* **encrypted\_folder\_path (str): The path to the folder containing encrypted files to be decrypted.**

**Main Workflow:**

**Error Handling**

* The application includes specific error handling to manage various exceptions:
* FileNotFoundError: Triggered when the specified file or directory is not found.
* PermissionError: Triggered when there are permission issues accessing the file or directory.
* IsADirectoryError: Triggered when a directory is encountered where a file is expected.
* InvalidToken: Triggered during decryption when the passphrase is incorrect.
* Exception: A general catch-all for any unexpected errors.

**Security Considerations**

* Passphrase Security: The security of the encrypted files relies on the strength of the passphrase. Users should choose a strong, unique passphrase.
* Salt Usage: A unique salt is used for each file encryption process, enhancing security by ensuring that identical passphrases do not produce identical keys.
* Secure Deletion: Files are securely deleted by overwriting them with random data before deletion, minimizing the risk of data recovery.