# ELEC 5150 Project: File Encryption System

## Muthu Ramaswamy

## Erich Wu

## Purpose

*To encrypt a pre-existing file system, generate a key that is newly generated each time a file in the file system is accessed. And to re encrypt that key when done. The key to this file system being completely secure is two-fold:*

1. The key used to encrypt the file system is stored in an encrypted state. The key that encrypts the file system key (The file system key is a simple random number) is generated through means that the attacker should not be able to find.
2. This second key and methods to generating it is stored on a USB drive, offline to the system. This USB drive should only be used when the system this is being performed on is completely offline.

## Methodology

1. The method to generating the key that encrypts files is simple and covered in our previous labs. This key will be denoted as **KFSi**.
2. The key that encrypts the file system key must be generated in a way that introduces entropy - or at least would not be obvious to an attacker. This key will be denoted as **Ki**
   1. Upon encrypting a file system, the user will be prompted for a password and a PIN in addition to a root directory of a file system they would like to encrypt. The user must remember the password and PIN.
   2. This password and pin will be used in several operations in order to generate two keys ( **Ka** and **Kb**) which are resultant from several operations involving other generated components such as two time stamps ( **Ti** and **TP** ) As well as a session number and two 256-bit salts for hashing operations (**SE** and **SU**).
   3. The **Ka** and **Kb** will be used to encrypt a few of the components, notably **Ki** and **Ti**, which will be stored (encrypted) in a table along with the other components including **Ki{KFSi}**. Though these values are stored in plain text, the method of generation is hidden.
3. When the user would like to access a file within the file system, the user must be prompted for the password and pin.
4. Upon password and pin entry, the user will then be prompted by a list of files which they may access within that file tree.
5. Once the user has chosen a file, the file must be decrypted using the **KFSi** which must first be decrypted using the **Ki** which is decrypted from the table, using **Kb**. Once the user is done editing or accessing the file, the file will be re-encrypted using **KFSi**.
6. A new **Ki** is then following the prior steps. Then **KFSi** is then encrypted using the new **Ki** and stored in the previously described table.

## Design

The project was split into several classes:

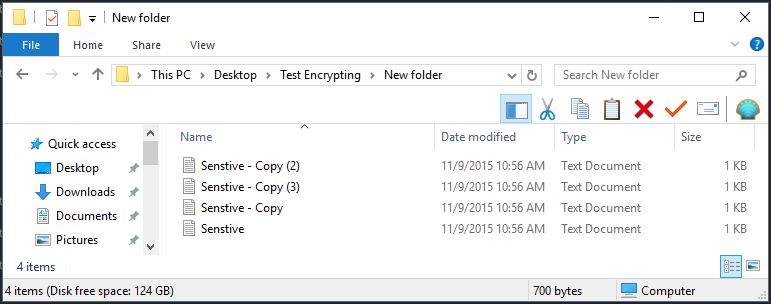
1. *ProjectNew Class:* This class will define the methods to perform the key generation and password and pin operations.
2. *EncryptDecrypt Class:* This class will define the methods to encrypting and decrypting strings, files, etc.
3. *FileHandler Class:* This class will define the methods to perform file checking and handle the file table as well as determining which file paths to access, etc.
4. *Interface Class:* This class will allow the user to actually interact with the file system. e.g. Entering password/pin, encrypting a new file system, accessing a file, etc.
5. *HexIO Class:* This class will handle various hex operations

*An additional batch file will be written to allow the user to easily use the system*

## Usage and Results

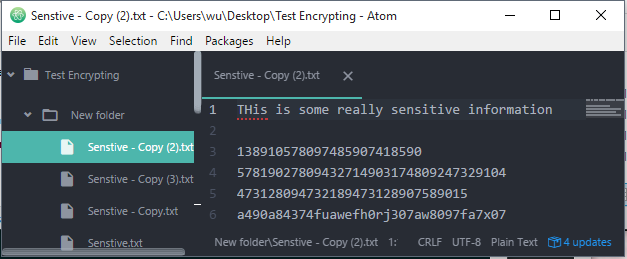
*The project can be run using the RunProjectUSB.bat file. A sample file tree "Test Encrypting" will be used in order to show the results of encrypting/accessing/updating the keys. The FileEncryptionSystem must be stored on a USB. Upon running the batch file, the user will be prompted to insert this USB.*

### Test Encryption File Tree



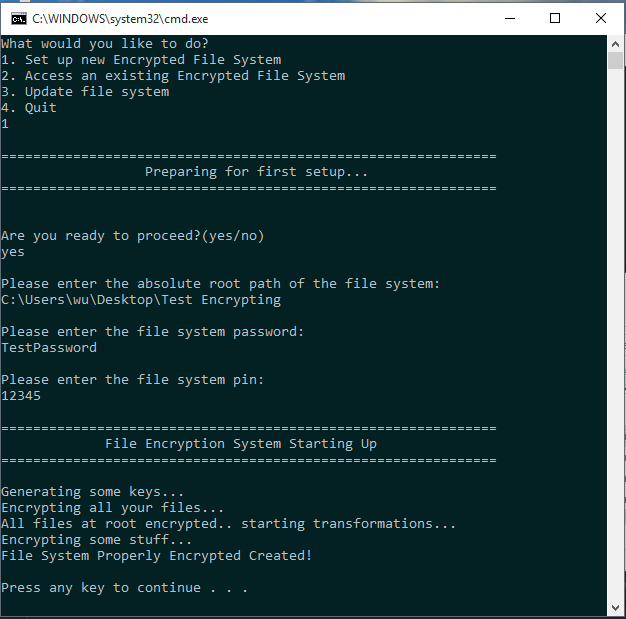
*This is a simple file tree for testing and sample purposes. The program should be able to handle large file trees. There are a total of 4 "New Folders" and within each of these folders is 4 text files.*

### Sample Plain Text Test File

**

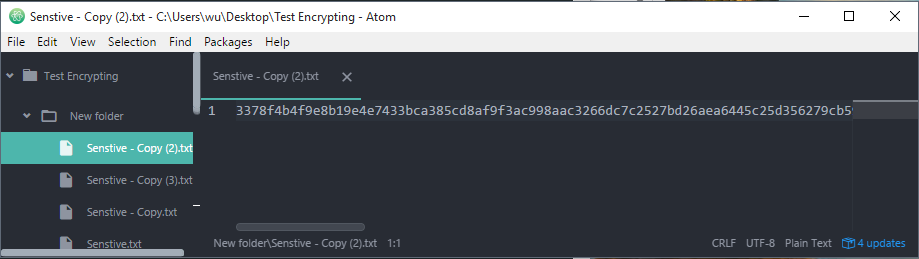
*All the test files in this file tree have the same content for illustration purposes, but the program can handle a variety of file formats and content, provided that no file is particularly large.*

### Sample System Setup Run



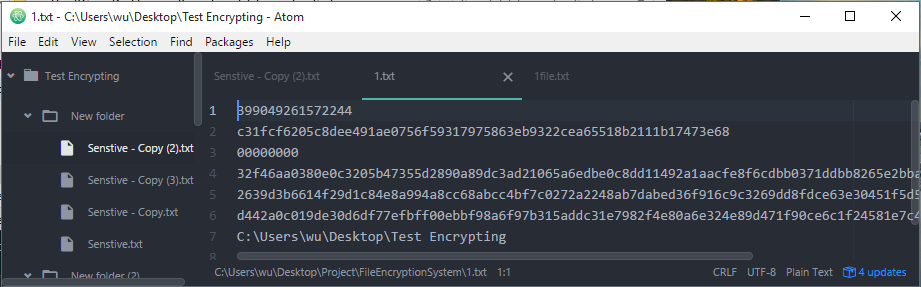
1. The user is prompted for an absolute root path for the file system.
2. The user is prompted for a password and pin.
3. The system handles key creation and storage, and encrypts the files.

### Sample Cipher Text Test File



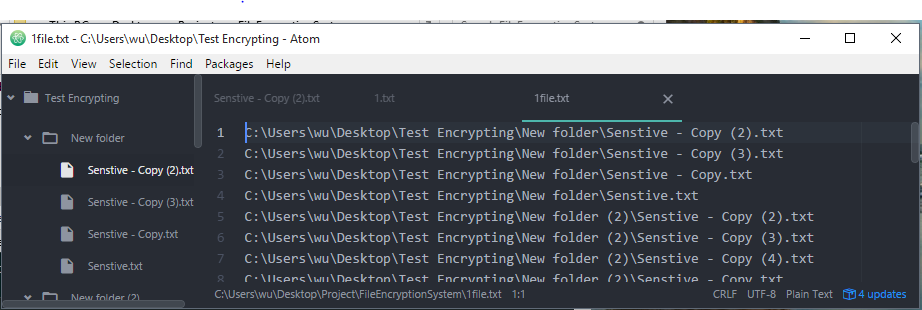
*The files in the file tree are now encrypted with a generated KFSi*

### Sample Key Table

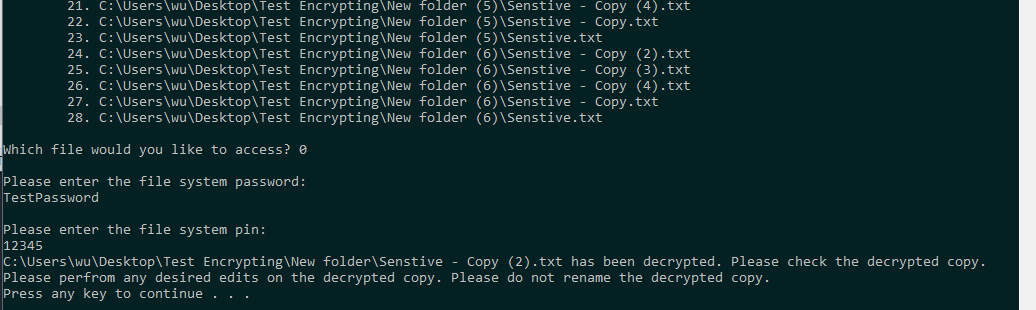
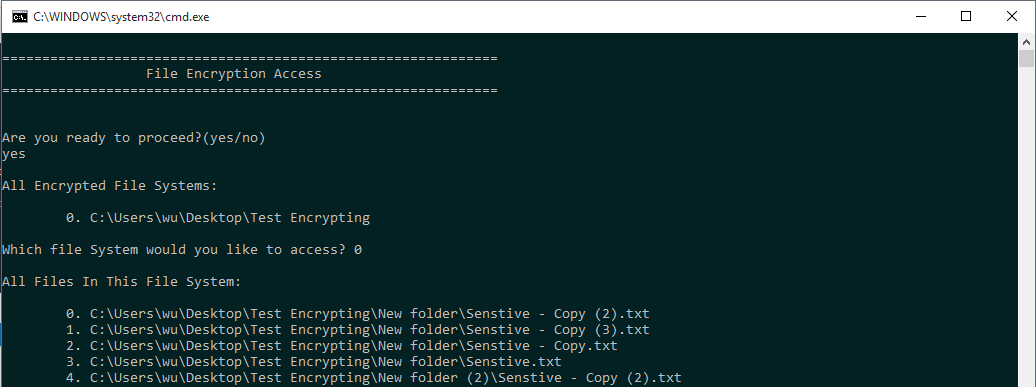
**

*The generated key table for this particular file tree. Note the session number is all 0's as it has not been accessed.*

### Sample File Table

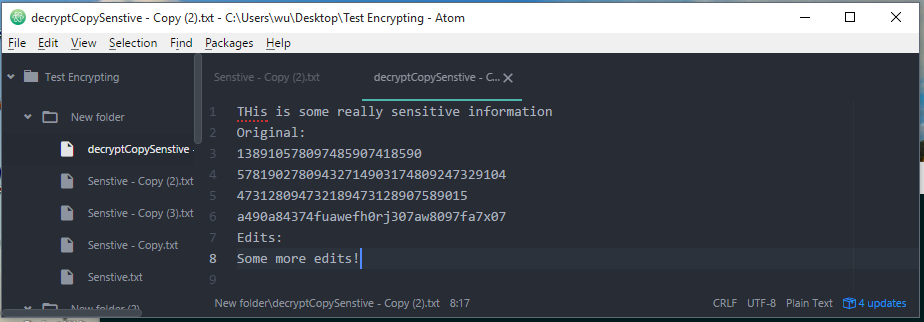


### Sample Access Run

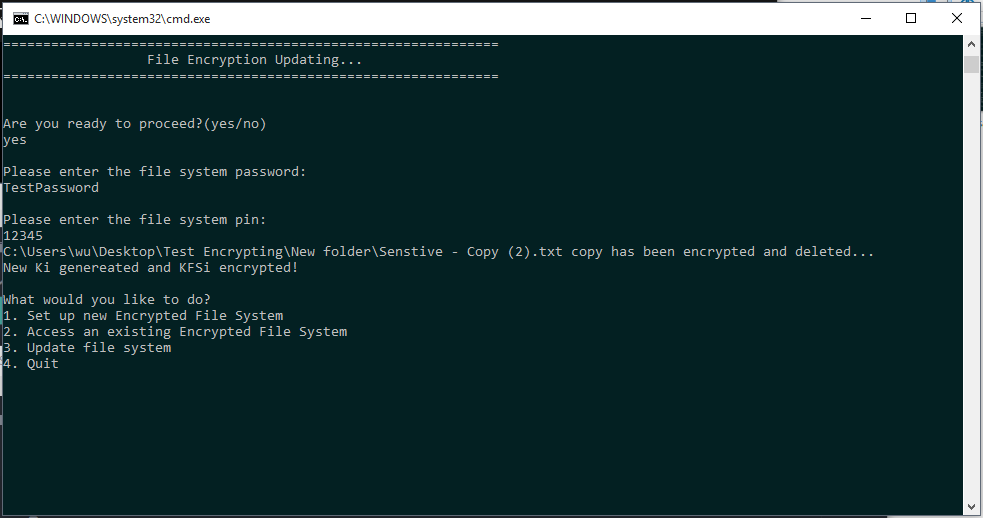


1. The user is prompted for which file system they would like to access.
2. The user is prompted for which file within the file system they would like to access.
3. The user is prompted for a password and pin.
4. The system attempts to decrypt the cipher text into a plain text copy.
5. The user is allowed to access the file, but the user **should not** rename the file.

*Here are some sample edits to the plain text:*

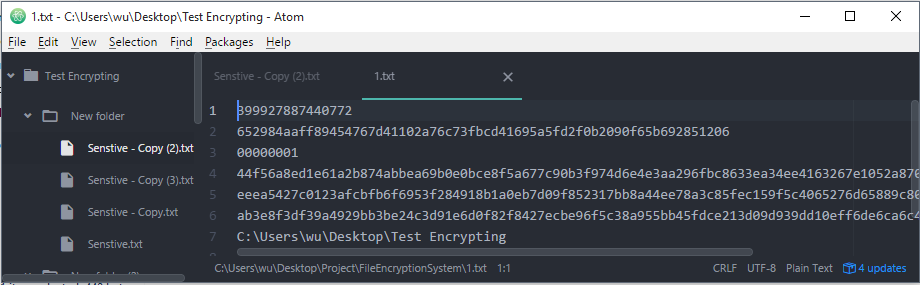


### Sample Update Run



* If the user has previously accessed a file the user is prompted for the password and pin of that file tree. The file is encrypted over into the original file, and then the keys are updated.
* If the user has not previously accessed a file the user is prompted for what file tree the would like to update, then the key file is simply updated with a fresh set of keys.

### Updated Key Table



*The new key table. Note how every key in the table has changed and the session number has been incremented*

### Updated Cipher Text

*Note that the differences between this cipher text and the prior one is further in the text, as AES-GCM-256 cipher is not dependent on prior blocks*

## Possible Updates, Additions:

* User experience? A GUI? Anything to improve user experience. As of now it takes a bit of computer savvy to be able to use the system.
* Reading large files? Major limitation right now is the inability to accurately parse large files. This becomes really difficult as after encryption, Java AES-GCM-256 writes the tag at the end of the file, meaning the complete file must be parsed before beginning decryption.

## Conclusion

This project provided an excellent review for encryption and key generation. It also provided an in-depth method for introducing randomness into encryption, in addition to building a usable application involving a mildly complex structure.