Secure File System Application
Chris Middleton
CSC 321

The secure file system application created by me, Chris Middleton, is designed to permit encrypted password-protected data storage, which can be kept on a shared local system, or sent to others through insecure channels as a layer of security to provide data anonymity and integrity. The application utilizes CBC encryption with an AES192 cipher, with a hash determined by the password entered upon creation of the encrypted file. It also features data integrity checks through a MAC.

Usage:

Append a file to the secure file system:

```
./securefs.exe -write [name of encrypted file] [name to store data as] [filename of data you wish to store]
```

Read data from the secure file system:

```
./securefs.exe -read [name of encrypted file] [name of data to access]
```

List the data contained in the secure file system:

```
./securefs.exe -list [name of encrypted file]
```

Open the secure file system for interactive viewing & editing:

```
./securefs.exe -open [name of encrypted file]
```

Each of the listed commands will prompt for a password to be entered. If the password is not correct, an error message will be shown and the application will exit. If the encrypted file does not yet exist, it will be created, and any password you enter will be bound to the file.

Passwords:

The secure file system application has no password validation. It is up to the user how secure they wish their password to be. Additionally, any character accepted by the command line (besides \r, \n, and \0) are usable in the password. Because of this, emojis and non-latin letters can be used in passwords, which tend to be 4 bytes each. Because aes 192 is used, the strongest passwords will be at least 16 bytes long. My personal recommendation for secure passwords is choosing 4-5 random obscure emojis and using those. This is especially secure from brute force attacks for the time being, since attackers usually do not include emoji in their alphabets.

Limitations:

1. The secure file system application is not capable of deleting or modifying items. You could delete the entire file and create a new one if you want to change data, but this is inconvenient. Thus, as of now, the application mainly only works well for archiving sensitive data. It would not be too difficult to program these features in, however this wouldn't change much about the actual security aspect of the assignment so I decided against spending time on this which is outside the scope of this class. 2. Subdirectories are not technically a part of the filesystem, however slash characters are completely valid in the names to store encrypted data as. So, you could just follow a typical file system naming convention to get a similar effect.