

File Mate

A-level Computing

Coursework (NEA)



File Security Program
Joshua Colclough

Contents

• Analysis	1 - 8
◦ Project Idea	1
◦ Client	1 - 2
◦ Choosing the right algorithms.	3 - 6
◦ Objectives	6 - 8
• Design	9 - 51
◦ Bluetooth	9
▪ Protocol	9/10
▪ Sending files over Bluetooth	10/11
◦ File Storage	11 - 13
▪ Encryption	13
◦ AES	13 - 25
▪ History	13
▪ The Algorithm	14 - 25
▪ Data handling + Before the operation starts.	14 - 16
▪ The operation.	16 - 25
◦ SHA256	25 - 34
▪ The Algorithm	26 - 34
▪ How the data is handled.	26/27
▪ Before the operation starts.	27 - 30
▪ The operation.	30 - 34

◦ BLAKE2b	34 - 40
▪ The Algorithm	34 - 40
▪ How the data is read.	34 - 35
▪ The operation.	35 - 40
◦ Quick Sort	40 - 41
◦ UI Research	42 - 48
▪ Main Program (on PC)	42 - 47
▪ The App	47 - 48
◦ The program as a whole	49 - 51
▪ IPSO diagram	49 - 50
▪ Using a USB stick	51
▪ Storing the files at home	51
• Technical Solution	52 - 143
◦ File structure	52 - 53
◦ Configuration of the program	53 - 56/57
◦ The PC App GUI	57 - 65
▪ Login Screen	57 - 59
▪ Main Screen	60 - 63
▪ The Information Tab	60 - 62
▪ Encryption and Decryption Status	62 - 63
▪ Settings Screen	63/64
▪ Search Bar	64/65
◦ The Mobile App GUI	65 - 69
◦ Key Algorithms	70 - 97/98
▪ AES	70 - 87

▪ AES package	70 - 76
▪ AESfiles package	76 - 82
▪ AESstring package	82 - 84
▪ AEScheckKey package	84 - 85
▪ The main package	85 - 87
▪ SHA256	87 - 90
▪ BLAKE2b	90 - 94
▪ The Sorts	95 - 98
○ The File class	98 - 100
○ PC app GUI code	100 - 131
▪ The root of the GUI	100 - 101
▪ Login classes	101 - 105
▪ Main Screen	106 - 129
▪ Main Screen code	106 - 120
▪ Main Screen Buttons	120 - 122
▪ Main Screen Popups	122 - 128/129
▪ Main Screen Labels	129
▪ Settings Screen	129 - 131
○ Mobile App GUI Code	131 - 143
▪ The root of the GUI	131 - 134
▪ Pad Screen	134 - 137
▪ Main Screen	137
▪ File Selection Screen & Shared Bluetooth functions	137 - 142
▪ SHA	142
▪ What is buildozer.spec?	142 - 143
• Testing	143 - 192

◦ Unit Tests	143 - 152
▪ AES	143 - 150
▪ Testing the core algorithm	144 - 147
▪ Testing AESfiles package	147 - 149
▪ Running the tests	149 - 150
▪ BLAKE2b	151 - 152
◦ Black-box tests	152 - 192
• Evaluation	193 - 196
◦ Evaluation of Objectives	193 - 196
◦ Things I would do to improve the project	196

Analysis

Project Idea:

The idea for my project, is to have a program that acts as a vault for important files. It will encrypt files given, and store them in a specified location. Once they are encrypted, they will only be accessible from within the program, and will only be accessible within the program if you know the encryption key (passphrase) that you set when creating the "vault".

Within the vault, you should be able to easily organise your files, add more to the vault, and remove (decrypt) files from the vault to any location (if possible).

My program would be useful for teachers, as they have to keep documents on student's grades, and any other student details secure. Since this is my use case, I will have to thoroughly test the security and practicality of my program to make sure teachers want to use it, and trust the program with these files. Also, I will add an optional mobile app that the user can download, which lets them connect to the program via Bluetooth to unlock the vault. This would be useful if you are a teacher, as if you leave the room with your phone in your pocket, and it is connected to the vault, if you have forgotten to lock the vault then a student might try to browse through it while you are gone, but with the app, as soon as you disconnect the Bluetooth connection it locks the vault, so if you forgot to close it then it closes itself.

The Bluetooth app should also be able to receive files from the PC app, so that the user can download files that are in the vault onto their mobile device. This would be useful for teachers that do not take their PC home (e.g not a laptop), so they can upload the files from the computer, to their phone so that they can edit the file at home or on the move (with another mobile app).

The program needs to work on both Windows, Linux and MacOS, as then teachers/users have more flexibility with what operating systems they can use it on, so they can easily go from machine to machine and carry their vault with them (on a USB stick for example), and they know that they can reliably use the program on most machines.

The user experience has to be pretty good. Good design practice will have to be used when making the GUI (e.g not putting the delete button next to the decrypt button), as I want my program to be easy to use by a wide range of people, so that even people who are not so good with computers can easily use the program. The way the user is directed around the program has to be logical as to not confuse the user, and adding a panic button to take you back to the main screen may be a good idea.

Client:

An example client for my project could be a teacher/school, as they have to keep files about students secure. For example, pupil details, exam results and other important student details. My program aims to help the teacher/school keep the pupil's files safe, and prevent the files from being accessed if their device is stolen. It will encrypt files given to the program, and be secured by a pin code that is transferred over Bluetooth to the computer from a mobile device. Once the mobile device is unpaired from the computer, the app will lock again. This will prevent someone from having access to the files if the computer is unlocked and is stolen, as the mobile device will go out of range of the computer, so the app on the computer will lock.

I sent a questionnaire to a member of the IT office at my school to ask what regulations there were about keeping a teacher's files safe, and what encryption they would suggest for keeping the files secure.

Hi Josh,

What encryption should I use when encrypting the user's files? [The bare minimum would be 128 bit AES, though 256 bit is recommended.](#)

Are there any standards or laws about what encryption method I should be using for files such as a teacher's student files (one of the clients for this program)? [Data protection laws. The current UK Law is the Data Protection Act 1998. Though as of 25th May, the law will be General Data Protection Regulations \(EU Law regarding all EU Citizens\).](#) This is a very complicated law, that is causing headaches for businesses worldwide. I've attached some links you might find useful regarding GDPR towards the end of this email.

Hope this helps!

Many thanks Mr __

<https://www.eugdpr.org/>

<https://itpeernetwork.intel.com/gdpr-opportunity-rethink-security/>

<https://ico.org.uk/for-organisations/guide-to-the-general-data-protection-regulation-gdpr/>

https://media.datalocker.com/marketing/GDPR_infographic_2017.pdf

<https://www.kingston.com/en/usb/resources/eu-gdpr>

I will be using this information as guidance for what I have to take into consideration. I will keep in mind the data protection laws when I am storing the user's files, and make sure I am within the regulations.

The EU General Data Protection Regulations consist of (As of 25/05/18):

Breach Notification:

If a data breach has been found and it might "result in a risk for the rights and freedoms of individuals", then the person that the data belongs to has to be notified within 72 hours.

Right to Access:

The person who's data it is can at any point ask for confirmation as to whether or not data concerning them is being processed, where it is being processed if it is and for what purpose.

Right to be Forgotten:

The data subject can ask for their data to be erased, and stop the processing of their data. This will be done depending on whether there is public interest in their data (e.g if a politician says something stupid then they can't ask Google to delete it just because it makes them look bad), and if the data is no longer relevant (e.g your cookies from last week that were used for targeted ads).

Data Portability:

The data subject should be allowed to ask to receive the data, and they should also be able to change which company is controlling their data.

Privacy by Design:

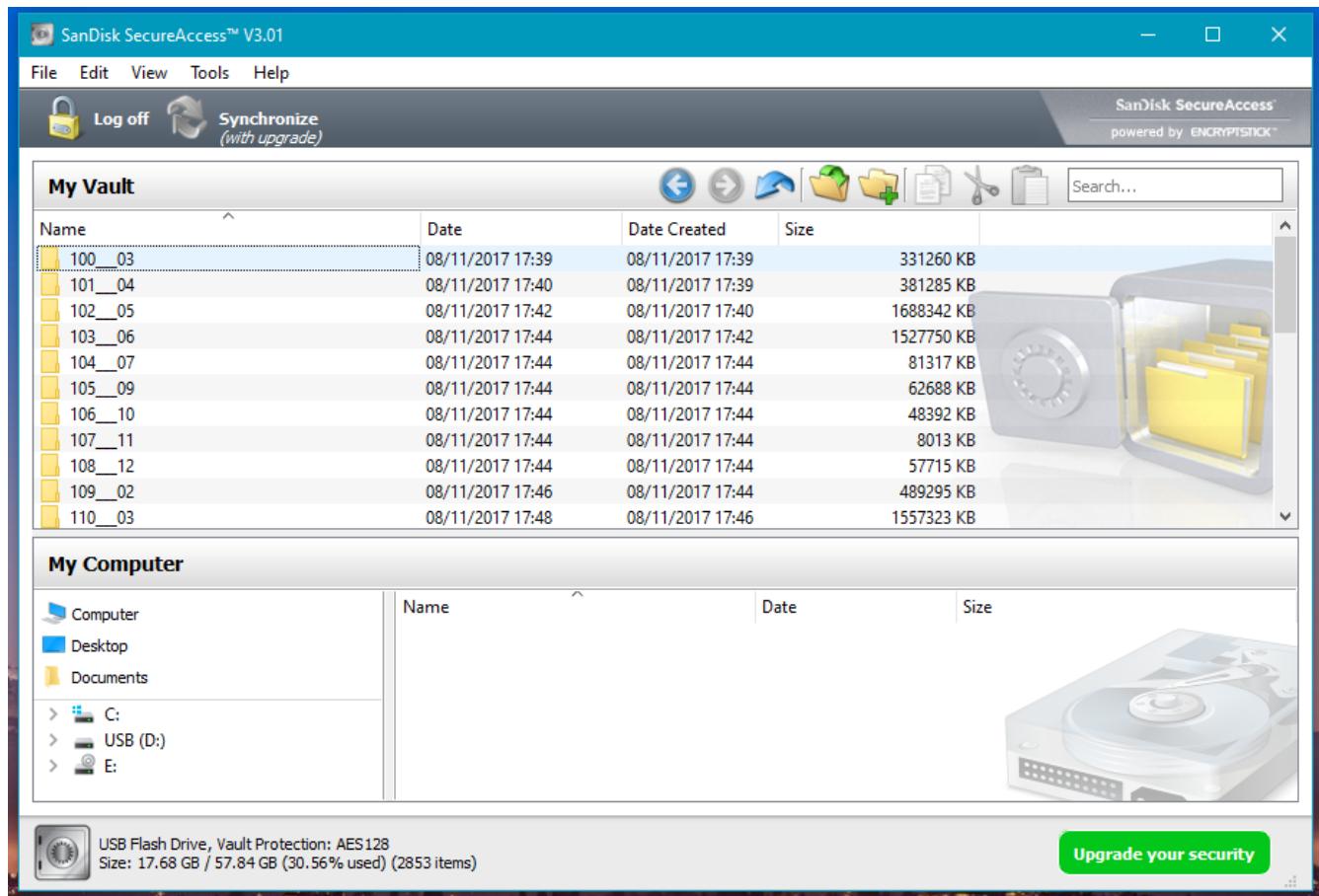
Tells the controllers of the data to only use the data absolutely necessary for the purposes they need it for. For example, an advertisement company might use your cookies to target ads to you, however they can't then use your location unless they are also using that to target ads. Basically don't take more than you need.

For my project, as the user is the data controller, then they already have the right to access, the right to be forgotten and data portability. For the breach notification, they will probably know it has happened as someone needs to have physical access to where the data is stored to breach it. However, with privacy by design, I will not be using any of the user's data for advertising, or any other agenda. I will make this clear to the user when they first use the program. Also the security will be

Another issue could be that if a file is deleted, the contents of the file might still remain. To fully remove the file I may have to use a one way function that ruins the data before deletion so that it cannot be accessed after it is deleted.

Existing Programs

The main program has to be designed to be easy to use, and actions that are used a lot should be easily accessible. I think I will go for a similar layout to a program that already exists, SanDisk Secure Access:



SanDisk Secure Access did inspire this project, however I do not want to make a carbon copy of it. I will take what SanDisk have done right, and improve the areas they lacked on.

SanDisk did these things right:

- The layout is pretty good because all buttons you would need regularly are available, and it doesn't differ too much in design from the Windows file explorer, so it feels familiar to its users.
- Shows the user how much space is left on their device.
- Shows useful information about each file.
- The user can easily sort the list of files however they want.
- More options are hidden unless needed regularly.
- Allows the user to search the vault for a file.
- I can easily drag files in and out of the program.

What I personally think SanDisk did not do too well:

- Looks a bit cluttered with all the extra stuff at the bottom. If I wanted to see other files on my computer I would open my file manager, and if I wanted to add files to the vault I can just drag it in easily.
- Faded pictures in the background are distracting.
- Some buttons are quite small, so may be hard for some users to click.
- Aesthetically alright but could be better.
- Some icons are confusing when first using the program (like the folder with the green arrow inside of it; too much going on).
- Size is displayed in kilobytes, which is alright but is kind of hard to read for files larger than 1 megabyte.

I will take these into account when I come to designing my GUI.

Choosing the right algorithms:

When encrypting, decrypting and hashing data in my program, I want it to be as fast as possible without compromising too much on security.

Hashing:

When hashing the key when it is input, the algorithm has to be very secure, and speed does not matter as much. A member of the SHA2 family of algorithms would be a good algorithm to do this, as it is quite slow, but it is very secure (SHA1 was found to have a lot of hash collisions). Speed does not matter as much for the key, as the input data will only ever be less than 16 bytes. A faster algorithm will only provide a few milliseconds over SHA, so there is no point compromising on security for a negligible time decrease.

For getting the checksum of files, the algorithm has to be very fast, as it will be done on the data in the file before and after the file is opened to check for changes. If this algorithm is slow, then the overall user experience will be much worse if the algorithm takes ages to open and close files. I will test each algorithm I am thinking of using for hashing and compare them using this algorithm (Python):

```
1 import hashlib      # Library of hashing algorithms.
2 from random import randint # Used to generate the data.
3 from time import time    # Used to measure how long the operation takes.
4
5 def generate(times, size): # Generates data, each block of length "size", and "times" number of blocks.
6     data = []
7     for i in range(times):
8         for j in range(size):
9             data.append(randint(0, 255)) # Randomly generate a byte.
10    return bytarray(data)
11
```

```

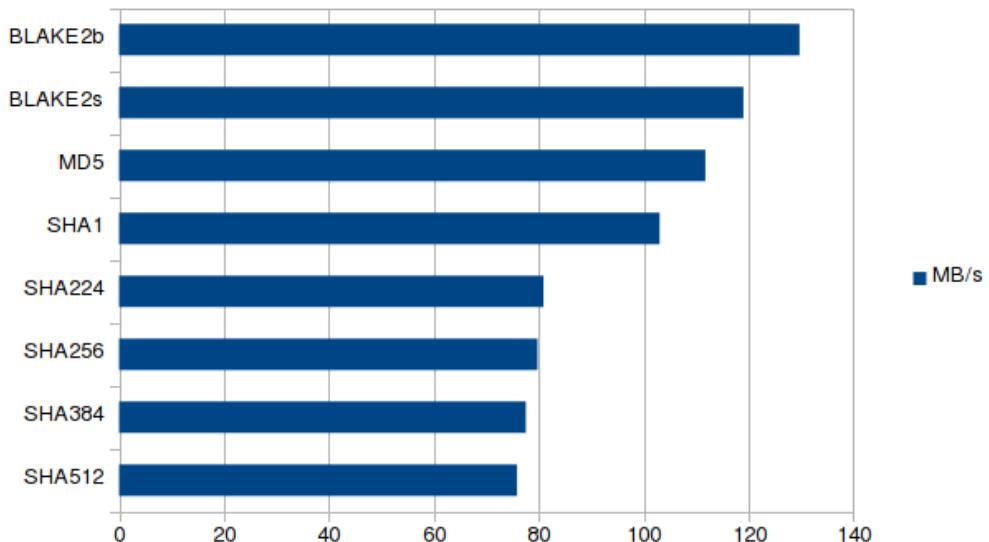
12 def test(times, size):
13     data = generate(times, size)      # Generate the data
14     start = time()                  # Get the start time
15     for i in range(times):
16         hashlib.sha256(data[i*size:(i+1)*size]).hexdigest() # Do the hash (in this case SHA256)
17
18     return (times*size)/(time()-start)    # Return the bytes per second.
19
20 print(test(1000, 128))  # Run the program.

```

I will run this algorithm on the same computer and make sure background tasks are closed, so that the results are not affected by other programs.

Here are the results:

Megabytes per second for each hash function (using 1000 blocks of 128 bytes (128 kilobytes)):



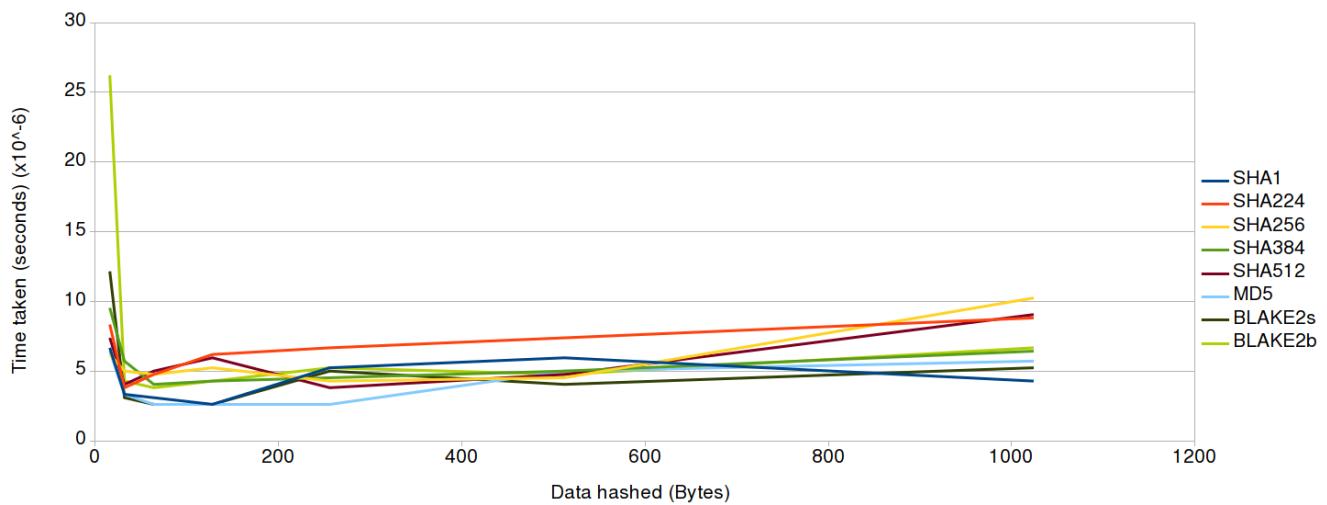
For my next tests, I will do data hashed against time. For this I will be using different sized files that I will make using this function:

```

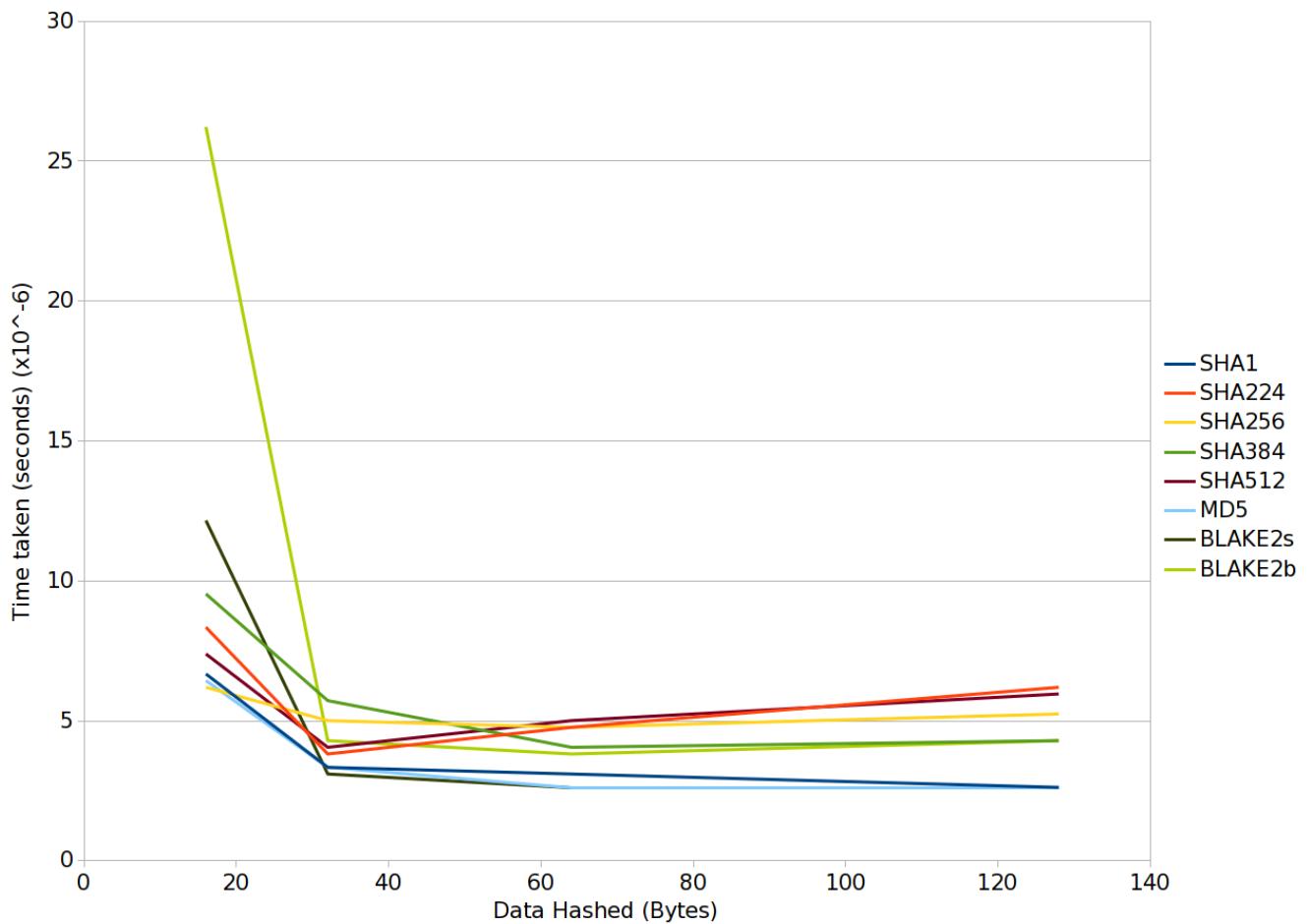
1 def generateFile(name, totalSize):
2     fo = open(name, "wb")
3     a = bytearray()
4     for i in range(totalSize):
5         a.append(randint(0, 255))
6     fo.write(a)
7     fo.close()

```

First I will test each hash function with encrypting very small data (<= 1 KiB). These were the results:

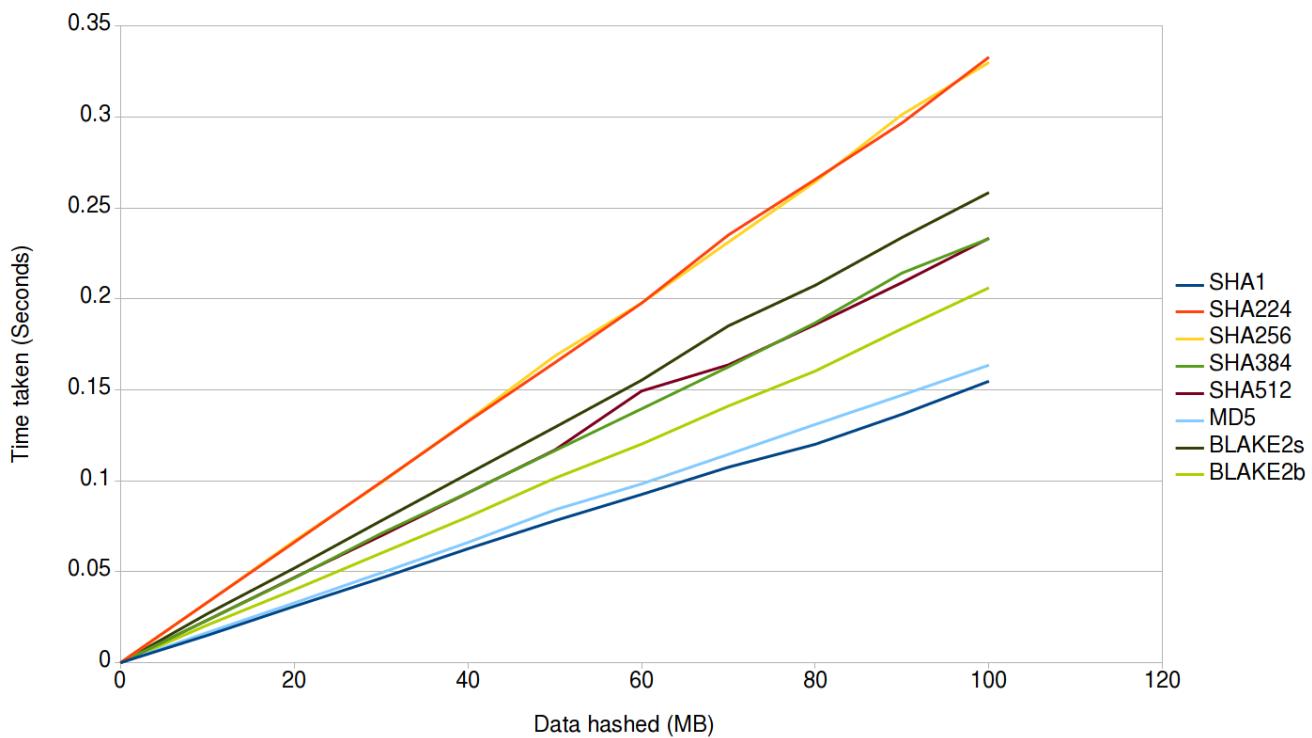


Here is the start of the graph, as that is the most interesting bit:



Here we can see that SHA256 is the fastest at hashing 16 bytes, but is quickly surpassed by most of the algorithms. Both BLAKE algorithms had a bad performance at the start, but after 64 bytes both were doing alright. MD5 is the quickest overall out of the group. From these results I think I will use SHA256 for hashing the key, since the key is 16 bytes in length, and also because SHA is more aimed at security than BLAKE, and MD5 and SHA1 are obsolete in terms of security.

The BLAKE algorithms were designed for big data, which is what I am going to look at next:



In this graph, the gradient (rate of change) of each line is the ratio of seconds to megabytes of each function (so $\frac{x}{y} = \text{megabytes/second}$). So the less steep the line is, the faster the operation.

SHA256 and SHA224 have taken the longest, at almost identical rates. BLAKE2s is quite slow, and this is because BLAKE2s is designed for 32-bit CPU architectures, and my CPU is 64-bit. MD5 and SHA1 are both the fastest, and have similar performance, but have security problems. BLAKE2b was the fastest out of the secure functions, so I will be using BLAKE2b for checksums in the program, as checksums need to be calculated quickly, as discussed before.

Objectives:

1. GUI should:
 - a. Be easy to use:
 - i. Logically laid out.
 - ii. Have simple options.
 - b. Display the files currently stored in the vault, along with the file extension and the size of the file.
 - c. Display the storage space remaining on the storage device the program is running on.
 - d. The user should be able to easily encrypt and decrypt files:
 - i. Using easy to access buttons in the UI.
 - ii. Using drag and drop from outside of the program.
 - iii. Decrypt to a directory specified.
 - e. Have an options menu, including the options to:
 - i. Change from 128 bit security to 256 security. 128 bit is the bare minimum.
 - ii. Change the location of the vault.

- iii. Set the default login method (Bluetooth or no Bluetooth).
- iii. Change if the search in the file browser is recursive or not.
- f. Make it easy to manage the files in the vault (move to other folders in the vault, rename, delete, etc).
- g. Have a secure login screen.
- i. Ask the user to either input the key via their keyboard (no Bluetooth for that session), or connect via the app.
- ii. Tell the user if the key is invalid or not, and smoothly transition into the main program.
- iii. Validate all input.
- h. Look relatively good without being bloated.
- i. Don't be costly on system resources when you are idle.
- ii. Don't overdo animations.
- i. Allow the user to easily read file names, and easily tell folders and files apart.
- j. Let the user preview images without opening them (using thumbnails or an information screen).
- k. Be resizeable, and all items on the screen should look ok.
- l. Allow the user to switch between using Bluetooth and using regular login.
- m. Make it easy for the user to return to the root folder of the Vault in case they get lost (a "panic" button).
- n. Give the user statistics during files being enc/decrypted, including:
 - i. What percentage of the file/folder has been completed. (Visual progress bar to show this too)
 - ii. The current speed of enc/decryption.
 - iii. An estimate of how long it should take to finish enc/decryption.
 - iv. If part of a folder then show the progress of the current file.
- o. Allow the user to sort the files by name or by size.
- p. Allow the user to configure default settings using a configuration file that is easy to use.
- q. Allow the user to search for file names in the vault.
- r. Not break when doing arbitrary tasks such as browsing the files.

2. App should:

- a. Be easy to use.
- b. Connect via Bluetooth to the PC.
- c. Allow the user to input their pin code easily.
- d. Tell the user if the pin code is invalid or not.
- e. Make it easy to recover from mistakes (e.g invalid pin code, or if they make a typo).
- f. Allow the user to see a list of files currently in the vault, and let the user download those files onto their mobile device.

3. File handling:

- a. Store the encrypted contents in the location specified by the user.

- b. Encrypt and decrypt relatively quickly, while still being secure.
 - c. When the Bluetooth device goes out of range or disconnects (if using Bluetooth), encrypt all decrypted files and lock the program until the pin code is input correctly again.
 - e. Have a recycling bin so that the user can recover their files.
 - f. When a file is opened, check for changes once it is closed.
 - g. Files stored in the vault should not be accessible from outside of the app.
 - h. Names of the files stored in the vault should also not be view-able from outside of the app (encrypt the name).
 - i. Allow the files/folders to be decrypted to an external location.
4. Overall system should work on Windows, MacOS and Linux.

Design

Bluetooth:

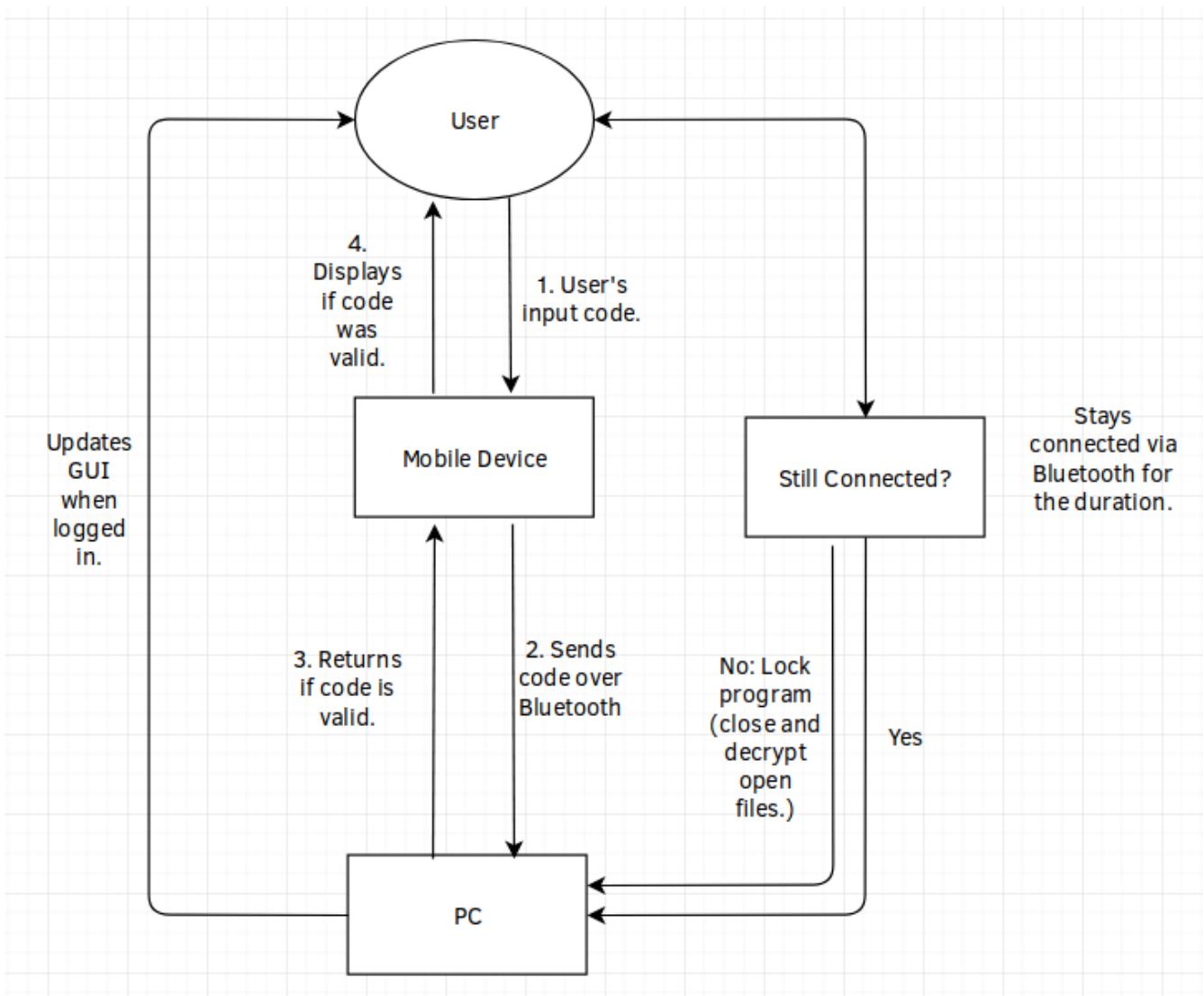
For the file store to be unlocked, I need to send the passcode to the computer via a Bluetooth connection.

For the computer and android device to connect to each other, one device has to be assigned as the server, so it makes sense to me to use the computer as the server, as it will be running for the entire duration that the user wants to use the program.

For the mobile app, I will be using Kivy to program the app. I am using Kivy so that the design is consistent with the design of the PC app. I will be using the android.bluetooth library that is included in the android SDK to transmit the data via Bluetooth.

For the Bluetooth server (on the pc), I will be using Python to receive the pin from the mobile device using PyBluez, check the sent pin, and send a message back saying if the code was valid or not. If the code is not valid, a message will be displayed on the computer that the code is invalid, and the code on the screen of the phone will be erased.

Here is a flow diagram for what Bluetooth will be like:



To send the files, I will need a protocol. A protocol is a set of rules for communicating over a network. A protocol will allow the program to distinguish data that is being sent is a key, file list or a file itself.

Protocol

The protocol rules all have to be strings of bytes that are not likely to appear in a key, file list or a file. This is a necessity because otherwise mid way through sending a key, file list or file, if the program encounters a protocol rule within the key, list or file, then it may cause the program to get confused as to what is being sent, or if the current key, list or file has finished being sent.

For each of the possible items that are going to be sent, each item needs a start header, and an end header.
Start header:

```
1 | !<operation>!
```

End header:

```
1 | ~!END!
```

For operations that do not have any extra data (arguments), then only the start header is sent.

For sending more complex operations, I will use objects that hold the data, pickle them (object sterilisation), and send the object data sandwiched between the `!<operation>!` header (start header) and the `~!END!` header.

For more complex operations that have multiple arguments, a separator is used to separate those arguments:

```
1 | ~~!~~
```

Here is an example with multiple arguments:

```
1 | !<operation>!<argument1>~~!~~<argument2>~!END!
```

This is especially useful for files, as this way I can send the metadata in one big lump, then send the file bit by bit. Here is what a file would look like when it is sent:

```
1 | !FILE!<metadata_object>~~!~~<data>~!END!
```

For the key however, since it will always be small (< 16 bytes), I will just send it with a `#` at the start, and a `~` to finish the message. This is acceptable because when the PC program starts, it doesn't expect any requests from the client, so it is just waiting for the key. The key should also only be made up of numbers.

```
1 | #<key>~
```

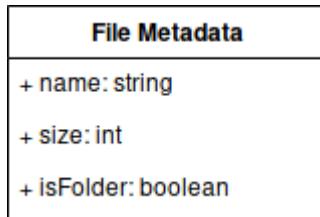
For items such as file metadata, I will use Python pickle to send an object (more of a structure) containing the metadata, rather than using separators, as then it is much easier for me to add information I want to send.

Sending files over Bluetooth:

To send a file from the vault, first it has to be decrypted to a temporary location. I could instead send the data from within AES, so that when a block is decrypted it is sent, however I don't plan on writing AES in Python since speed is essential for AES (and a new Bluetooth socket would have to be set up if using a different language).

Metadata will be sent as an object before sending the file contents, as talked about in the above section.

An example class for file metadata may look like this:



```
1 class fileMetadata:
2     def __init__(self, name, size, isFolder):
3         self.name = name      # The name of the file being sent.
4         self.size = size      # The size of the file being sent.
5         self.isFolder = isFolder # Boolean for if the file is actually a folder.
```

This is more of a structure than an object, as it has no methods, and is just a collection of data.

After the metadata is sent, a separator will have to be sent to separate the metadata from the file data itself. I discuss this in the above section.

For the file itself, I will send the file in chunks, so that

1. I don't use too much memory (since mobile devices usually have a small amount of memory compared to regular computers).
2. The Bluetooth adapter can keep up with the amount being sent.

This reduces the stress on both the mobile device and the PC.

Once the full file is sent, an end header is sent to tell the program that the full file has been transmitted.

File Storage:

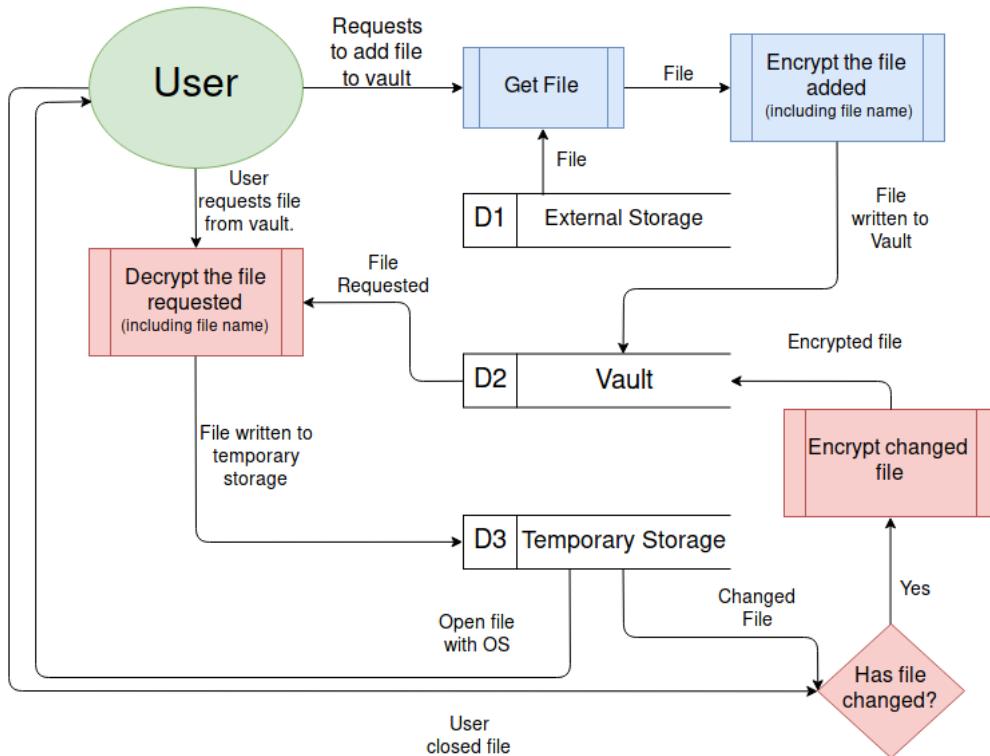
For storing the files, I will store the encrypted files in a directory set by the user (I will call this the 'vault' from now on). The files will be stored just like regular files, however the names of the folders and files within the vault will all be encrypted, and encoded to base 64 (better than encoding to hexadecimal as there is a file name limit on most operating systems, so this allows for longer file names as each byte will be represented by less characters).

The encryption method I will use AES 128 bit, as it will slightly compromise security over using 256 bit, however it will be faster to decrypt files for use, giving the user a better experience, however I can add an option to use 256 in the settings if the user needs more security over performance. For the encryption key, the key will be set up every time a new vault is created (this includes first starting the program). It will tell the user to enter the new key, and then from that moment forwards in that vault, that key will remain the same, and will be used every time a file is encrypted/decrypted in the vault.

When a file is encrypted, the key is appended to the start of the data, and is then encrypted. This is so that when the data is decrypted, only the first block has to be decrypted and compared with the key entered to check if the key entered was correct, rather than decrypting the whole file just to find out that the file can't open, or is corrupted. This will also be used to check the key entered at login, where the login will try to find the first file it can within the vault, decrypt the first block of that file and compare it with the input.

The key will have to be hashed if I send it over Bluetooth, as it may get intercepted, and it is also a good idea to hash it on the computer program as well, as if someone somehow manages to get the key, it will not be the user's original input, so if the user uses it for something else, their other accounts will be fine.

Here is a data flow diagram showing how the data is handled once logged into the program:



The key is also passed to any stages that encrypt or decrypt, as at this point the user should already be logged in.

When a file is edited, the file should be checked to see if any changes have been made, and if there has been changes, remove the version of the file currently in the vault, and encrypt the latest version into the vault. Also, if there are any new files in the temporary folder (for example if the user renames the file), then encrypt them to the vault as well.

To do this, I need a way of getting a checksum of the file before and after it has been opened. I need a fast algorithm so that the user is not waiting too long for the file to open and close, but it also needs to be unlikely that there will be a collision (where if they change the file and the checksum gives an answer that is the same as before the file was changed, that would be a collision). I will discuss which checksum I will be using in the next section.

When viewing the files in my program, I will use an object that holds all of the information I need about the file, and any methods that I need to get that information.

Here is what I expect the class to be like:

File
+ rawSize: int
+ displaySize: string
+ isDir: bool
+ path: string
+ name: string
+ hexPath: string
+ hexName: string
+ getCheckSum(self): string
+ getSize(self): int

Where `getCheckSum` will get the BLAKE2b checksum of the file. The hexPath and the hexName will hold the encrypted path and encrypted name of the file, so that I don't keep encrypting and decrypting the file name.

Encryption:

For encryption, I will definitely be using AES, because it is the standard and has been tested extremely thoroughly by the public. I do not want to compromise on security, and AES is still pretty fast anyway.

I will use 128 bit AES mainly, as it is still proven to be secure from attacks, and may include the option to use 256 bit if desired by the user. The majority of users will not need AES 256 level security, but I will include it for people that may need it.

AES:

History:

In 1997, the encryption standard at the time, DES, was becoming obsolete due to the advancements in the computer industry. This resulted in the National Institute of Standards and Technology in the United States to call for a new advanced encryption standard (AES).

They held a competition that consisted of 15 different algorithms that had been submitted by different teams. The algorithm that won was an algorithm called Rijndael, an algorithm created by two Belgian cryptographers – Vincent Rijmen and Joan Daemen.

One of the reasons AES has been more successful than DES so far is that AES was thoroughly tested by members of the public during the competition, analysing every aspect of the algorithms to find a way to break them. On the other hand, DES was created in secrecy by IBM in the 70s, and the algorithm was only released a few years later.

This open-source approach ended up helping the new Advanced Encryption Standard, as the program could be heavily analysed by people all across the globe.

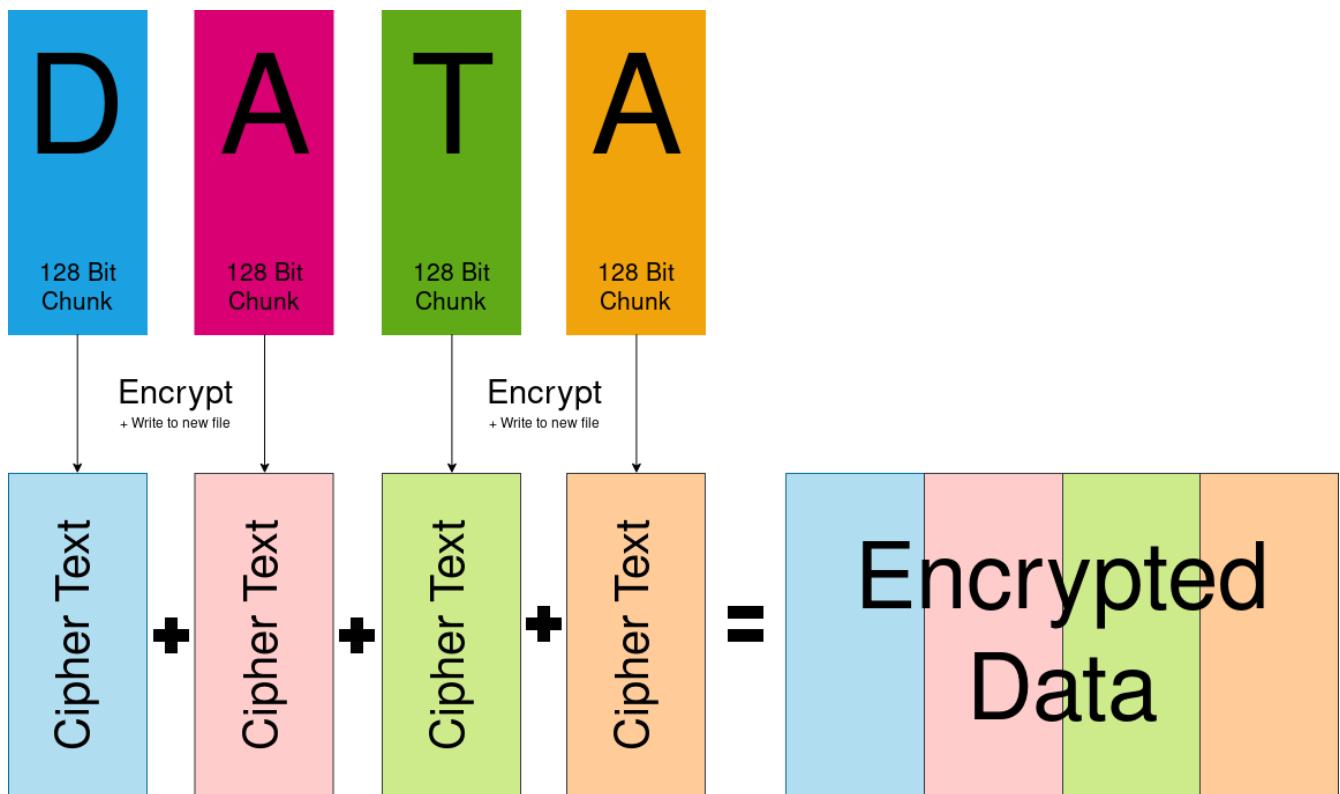
The Algorithm (128 bit AES):

How the data is handled:

AES works by using a block cipher, so it splits the data given into 128 bit, 192 bit or 256 bit chunks depending on what AES you choose (128, 192 or 256). You then use the algorithm on each block to get the cipher text, then you write it to the new file, and move onto the next block.

AES is a symmetric cipher, so only one key is needed to both encrypt and decrypt the data.

Here is an example for 128 bit AES encryption:



Decryption works exactly the same, however the cipher text is split up and decrypted.

Each 128 bit "block" of data can also be called a "state".

Before the operation starts:

First, the data has to be a multiple of 16 in length. If it isn't then more bytes need to be added to the end such that the data is 16 bytes in length (padding).

However, the padding cannot just be 0's at the end, as when we decrypt the block, we have no way of distinguishing these 0's from the rest of the data, or know if they are supposed to be there. To get around this, when we add the padding, we give each byte the value of how many more bytes we need to add to get the length of the block to 16 bytes. This sounds confusing, but here is an example:

Say we had a block that was = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

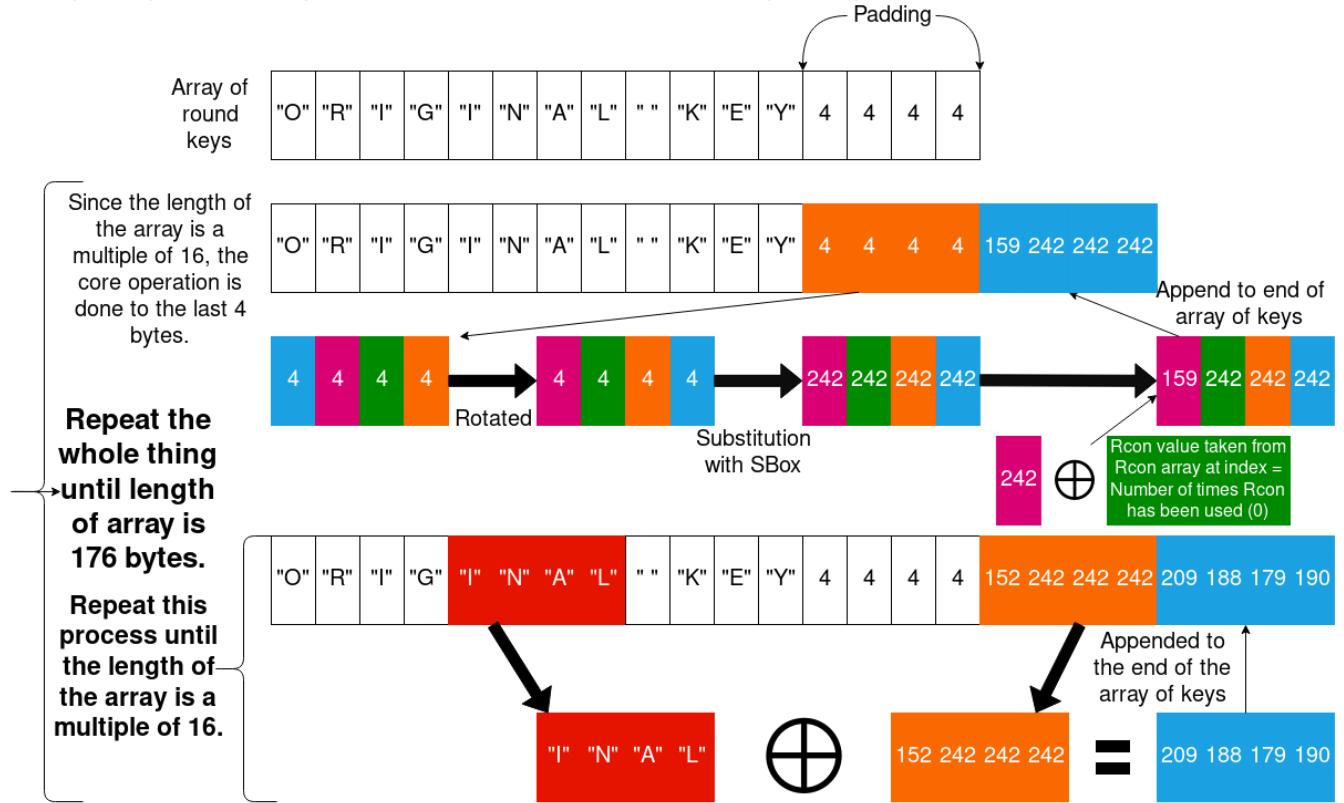
This block is not 16 bytes in length. To pad this block, we need to add 3 lots of the number 3 to the end (since $16 - \text{length of the block} = 3$). The new block would look like this:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 3, 3, 3]

When we go to decrypt this block, we check to see if the value of the last byte in the block is lower than 16, and that if the number occurs the same number of times as the value, then we remove these bytes.

For each round of the encryption, a different key has to be used. To make the cipher decipherable, these keys have to be derived from the original key given. For 128 bit AES (the main one I will be using in the program), the 16 byte key has to be transformed into a 176 byte list of 16 byte keys (11 keys in total, one for every round).

The first 16 bytes are the key, and then from there, the algorithm is started. Here is the algorithm with example: **Figure 1 (A larger version can be found in the "Large Images" section)**



The algorithm in pseudocode:

```

1  function expandKey(inputKey)
2      expanded := inputKey
3      bytesGenerated := 16
4      rconIteration := 1
5      temp := uint8[4]
6
7      while bytesGenerated < 176
8          temp = expanded[bytesGenerated - 4:bytesGenerated]
9
10     if bytesGenerated MOD 16 == 0 then
11         temp[0], temp[1], temp[2], temp[3] = temp[1], temp[2], temp[3], temp[0]
12         temp[0], temp[1], temp[2], temp[3] = sBox[temp[0]], sBox[temp[1]], sBox[temp[2]], sBox[temp[3]]
13
14         temp[0] = temp[0] XOR rcon[rconIteration]
15         rconIteration = rconIteration + 1
16     end if
17
18     for i := 0 to 4

```

```

19     expanded[bytesGenerated] = expanded[bytesGenerated - 16] XOR temp[y]
20     bytesGenerated = bytesGenerated + 1
21   end
22 return expanded
23 end

```

The array of round keys starts off the exact same as the original key. Then if the length of the round key array is a multiple of 16 (which it is), the last 4 bytes of the previous round key (in this case the last 4 bytes of the original key) is:

1. Rotated (The first element of the 4 bytes is put at the end).
2. Substituted (Using the Rijndael Substitution-Box found at: https://en.wikipedia.org/wiki/Rijndael_S-box).
3. First byte of the 4 is XOR-ed with it's corresponding Round Constant (depending on the round number the key will be used in).
4. The result is appended to the array of round keys.

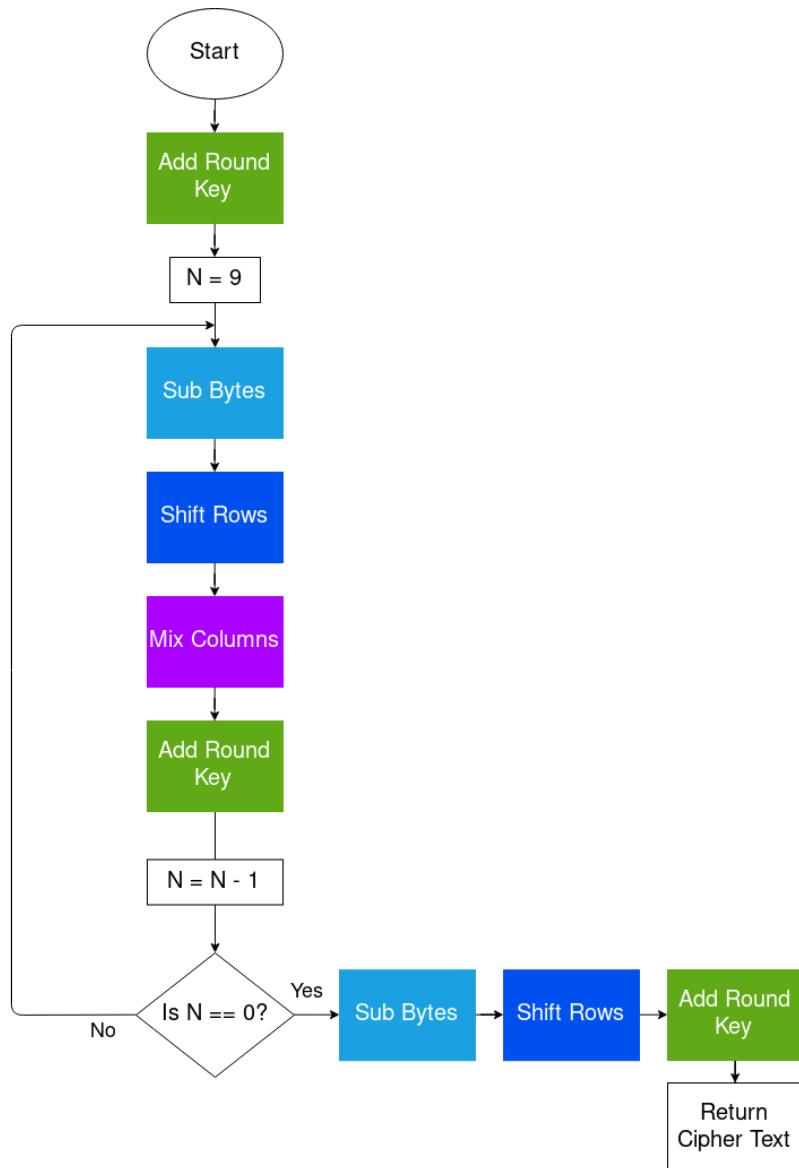
If the length of the round key array is not a multiple of 16, then the last 4 bytes in the array are XOR-ed with 4 bytes of the array that are 16 bytes before hand (shown in **Figure 1**).

This process is repeated until the length of the round key array is 176 bytes, then we will have one 16 byte key for each of the 11 rounds.

And that's all of the preparations done.

The operation:

Here is a diagram of the operation (I will explain each step in detail below):



In total there are 11 rounds (9 regular rounds). For each round, the corresponding round key (that we calculated beforehand) is used in the operation.

The 16 bytes in the state can be represented in a 4x4 grid, to make it easier to visualise what is happening at each stage:

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

Add Round Key:

The Add Round Key step is literally just XOR-ing each byte in the current block of 16 bytes, with each byte in the 16 byte round key, and returning the state.

Here is pseudocode for the **Add Round Key** step:

```
1 | function addRoundKey(state, roundKey)
2 |   for i := 0 to 16
3 |     state[i] = state[i] XOR roundKey[i]
4 |   return state
```

Sub Bytes:

Sub bytes substitutes each byte in the state with it's corresponding value in the Rijndael substitution box:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

When using the sub-box, you have to think of each byte as hexadecimal (0xYZ). Each row of the sub box is the value of the Y value (16s) in the hexadecimal representation of the byte. Each column of the sub box is the value of the Z value (1s) in the hexadecimal representation of the byte.

For example, if I had the hex `0xA1`, it would be substituted by the value: `0xA2`, as it is row "1", column "A".

Here is the pseudocode for the **Sub Bytes** step:

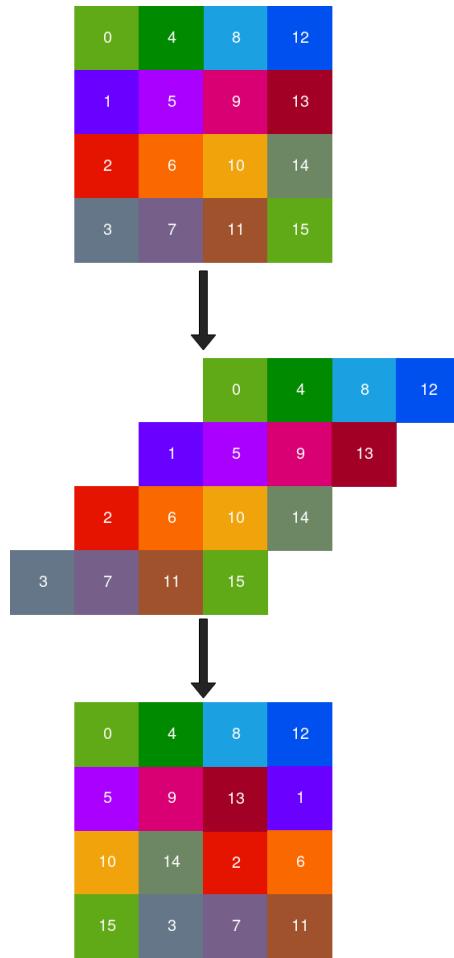
```
1 | function subBytes(state)
2 |   for i := 0 to 16
3 |     state[i] = sBox[state[i]]
4 |   return state
```

It is pretty much the same as **Add Round Key** but instead of XORing you substitute each byte of the state with the corresponding byte in the sub-box (sBox).

Shift Rows:

Shift Rows shifts the rows (really?) left depending on the row number.

For example, the first row is shifted left by 0, second row shifted by 1 and so on:



Here is the algorithm for **Shift Rows**:

```

1  function shiftRows(state)
2      temp := []
3
4      temp[ 0] = state[ 0]
5      temp[ 1] = state[ 5]
6      temp[ 2] = state[10]
7      temp[ 3] = state[15]
8
9      temp[ 4] = state[ 4]
10     temp[ 5] = state[ 9]
11     temp[ 6] = state[14]
12     temp[ 7] = state[ 3]
13
14     temp[ 8] = state[ 8]
15     temp[ 9] = state[13]
16     temp[10] = state[ 2]
17     temp[11] = state[ 7]
18
19     temp[12] = state[12]
20     temp[13] = state[ 1]
21     temp[14] = state[ 6]
22     temp[15] = state[11]
23
24     return temp

```

The array is indexed to correspond to the images above.

Mix Columns:

Mix columns is the most confusing step of AES, so I will try to break it down into small pieces.

The mix columns calculation is this:

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Where \$r_0\$ to \$r_3\$ is the result of the operation, and \$a_0\$ to \$a_3\$ is the 4 bytes that make up the input column.

This is matrix multiplication, but we need to do dot product multiplication. This is where we multiply each corresponding element in each row of the pre-defined matrix (the one with numbers already in it), with the corresponding element in \$a_0\$ to \$a_3\$, and then adds them up MOD2, also known as XOR (so that it is still 1 byte).

One way to represent this is like this:

$$\begin{aligned} r_0 &= (2 \times a_0) \oplus (3 \times a_1) \oplus (1 \times a_2) \oplus (1 \times a_3) \\ r_1 &= (1 \times a_0) \oplus (2 \times a_1) \oplus (3 \times a_2) \oplus (1 \times a_3) \\ r_2 &= (1 \times a_0) \oplus (1 \times a_1) \oplus (2 \times a_2) \oplus (3 \times a_3) \\ r_3 &= (3 \times a_0) \oplus (1 \times a_1) \oplus (1 \times a_2) \oplus (2 \times a_3) \end{aligned}$$

To dot product two binary numbers, they need to be represented using a Galois field.

A number can be represented by using a Galois field. A Galois field is just a way to represent a number as a polynomial, e.g \$5x^2 + 2x+3\$, where \$x^2\$ is \$10^2\$, so the number of 100s in the number (for decimal), while \$x\$ is the number of tens. In this case, this Galois field would represent the number 523, as there are 5 hundreds, 2 tens and 3 ones.

For example, if we wanted to represent the decimal number: 25301 as a Galois field, it would be:

$$2x^4 + 5x^3 + 3x^2 + 1$$

Note that the 0 in 25301 is not included, as \$0x = 0\$.

To represent a binary number, the same logic applies. For example, to represent the binary number `10011011` as a Galois field, it would be:

$$x^7 + x^4 + x^3 + x^1 + 1$$

To get back to decimal, we can replace the \$x\$ with the number 2, as binary is base 2:

$$2^7 + 2^4 + 2^3 + 2^1 + 1 = 155 = 10011011$$

The dot product of two Galois fields is like expanding brackets: \$(x+2)(x+3) = x^2 + 5x + 6\$, which is \$(x \times x) + (2 \times x) + (x \times 3) + (3 \times 2)\$, so we just multiply each item in each bracket together.

Now I will do an example of doing one result (\$r_0\$) of mix columns.

Lets use these values of \$a_0\$ to \$a_3\$ for the example:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} d4 \\ d4 \\ d4 \\ d5 \end{bmatrix}$$

To get \$r_0\$ I have to do:

$$r_0 = (2 \times a_0) \oplus (3 \times a_1) \oplus (1 \times a_2) \oplus (1 \times a_3)$$

which is:

$$r_0 = (2 * d4) \oplus (3 * d4) \oplus (1 * d4) \oplus (1 * d5)$$

in this example.

I am using \$d4, d4, d4, d5\$ as test values as they are test vectors used on this page: https://en.wikipedia.org/wiki/Rijndael_MixColumns, to check that we get the right answer.

Now I need to convert the hex values \$d4\$ and \$d5\$ to binary:

\$d4\$ in binary is \$11010100\$

\$d5\$ in binary is \$11010101\$

Now i need to convert both of these into Galois fields:

$$\begin{aligned} 11010100 &= x^7 + x^6 + x^4 + x^2 \\ 11010101 &= x^7 + x^6 + x^4 + x^2 + 1 \end{aligned}$$

Then I need to multiply them all by their corresponding value in the pre-defined table expressed as a Galois field (e.g. \$2 \equiv x\$):

$$\begin{aligned} (x^7 + x^6 + x^4 + x^2)(x) &= x^8 + x^7 + x^5 + x^3 \\ (x^7 + x^6 + x^4 + x^2)(x+1) &= x^8 + x^7 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 \\ &= x^8 + 2x^7 + x^6 + x^5 + x^4 + x^3 + x^2 \\ (x^7 + x^6 + x^4 + x^2)(1) &= x^7 + x^6 + x^4 + x^2 \\ (x^7 + x^6 + x^4 + x^2 + 1)(1) &= x^7 + x^6 + x^4 + x^2 + 1 \end{aligned}$$

But hang on a second, the answer to \$d4 * 3\$ and \$d4 * 2\$ both have a \$x^8\$ term, which means it's bigger than 255 (since \$2^8 = 256\$), so it is no longer a byte, which means that it no longer fits in with 128 bit AES.

To fix this, we replace all of the \$x^8\$ terms with this pre-determined polynomial (Rijndael's finite field), reducing by MOD2 as we go along:

$$x^8 \equiv x^4 + x^3 + x + 1$$

Let's try this with d4*3 :

$$\begin{aligned}
3d4 &= x^8 + 2x^7 + x^6 + x^5 + x^4 + x^3 + x^2 \\
&= (x^4 + x^3 + x + 1) + 2x^7 + x^6 + x^5 + x^4 + x^3 + x^2 \\
&= 2x^7 + x^6 + x^5 + 2x^4 + 2x^3 + x^2 + x + 1 \\
&= 0x^7 + x^6 + x^5 + 0x^4 + 0x^3 + x^2 + x + 1 \quad \text{Here is where I did MOD2} \\
&= x^6 + x^5 + x^2 + x + 1
\end{aligned}$$

Again with $d4 * 2$:

$$\begin{aligned}
2d4 &= x^8 + x^7 + x^5 + x^3 \\
&= (x^4 + x^3 + x + 1) + x^7 + x^5 + x^3 \\
&= x^7 + x^5 + x^4 + 2x^3 + x + 1 \\
&= x^7 + x^5 + x^4 + x + 1
\end{aligned}$$

Now, with our new values for a_0 to a_3 , we can finally do the equation:

$$\begin{aligned}
r_0 &= (2 \times d4) \oplus (3 \times d4) \oplus (1 \times d4) \oplus (1 \times d5) \\
r_0 &= (x^7 + x^5 + x^4 + x + 1) \oplus (x^6 + x^5 + x^2 + x + 1) \oplus (x^7 + x^6 + x^4 + x^2) \oplus (x^7 + x^6 + x^4 + x^2 + 1) \\
r_0 &= (2^7 + 2^5 + 2^4 + 2 + 1) \oplus (2^6 + 2^5 + 2^2 + 2 + 1) \oplus (2^7 + 2^6 + 2^4 + 2^2) \oplus (2^7 + 2^6 + 2^4 + 2^2 + 1)
\end{aligned}$$

$$\begin{array}{r}
r_0 = 10110011 \\
01100111 \\
11010100 \\
\oplus \quad 11010101 \\
\hline = 11010101
\end{array}$$

$$r_0 = 213(\text{decimal})$$

And, thank god, that is the correct answer for the test vector on this page: https://en.wikipedia.org/wiki/Rijndael_MixColumns.

To get r_1 , r_2 and r_3 , you repeat the process using the equations for each defined at the top of this section.

This whole process has to be done on each column.

On a computer, this would be very demanding on the processor, however since the range of the inputs is 0-255 (since the number has to be represented by 1 byte), you can make a lookup table with all of the 256 possible outputs, for each of the multiplications, for each of the 256 possible inputs. This drastically increases speed, and also makes it easier to program. You would have a table for multiplication by 2 and 3, and for the inverse function of Mix Columns you would need multiplication by 9, 11 and 13.

This trades a few kilobytes of memory for a drastic improvement in speed.

This makes the pseudocode for **Mix Columns** very simple:

```

1 // mul2 and mul3 are the pre-defined tables talked about above.
2 function mixColumns(state)
3     temp := []
4
5     temp[ 0 ] = mul2[state[0]] XOR mul3[state[1]] XOR state[2] XOR state[3]
6     temp[ 1 ] = state[0] XOR mul2[state[1]] XOR mul3[state[2]] XOR state[3]
7     temp[ 2 ] = state[0] XOR state[1] XOR mul2[state[2]] XOR mul3[state[3]]

```

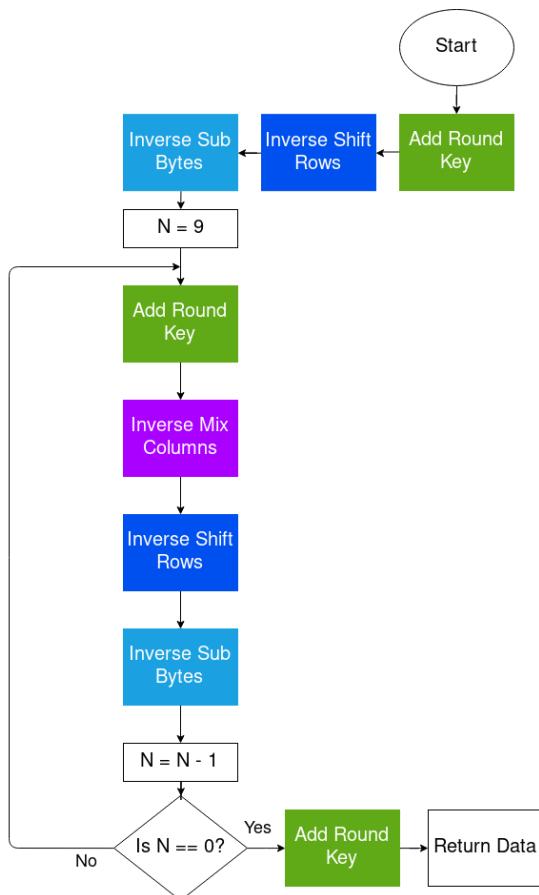
```

8   temp[ 3] = mul3[state[0]] XOR state[1] XOR state[2] XOR mul2[state[3]]
9
10  temp[ 4] = mul2[state[4]] XOR mul3[state[5]] XOR state[6] XOR state[7]
11  temp[ 5] = state[4] XOR mul2[state[5]] XOR mul3[state[6]] XOR state[7]
12  temp[ 6] = state[4] XOR state[5] XOR mul2[state[6]] XOR mul3[state[7]]
13  temp[ 7] = mul3[state[4]] XOR state[5] XOR state[6] XOR mul2[state[7]]
14
15  temp[ 8] = mul2[state[8]] XOR mul3[state[9]] XOR state[10] XOR state[11]
16  temp[ 9] = state[8] XOR mul2[state[9]] XOR mul3[state[10]] XOR state[11]
17  temp[10] = state[8] XOR state[9] XOR mul2[state[10]] XOR mul3[state[11]]
18  temp[11] = mul3[state[8]] XOR state[9] XOR state[10] XOR mul2[state[11]]
19
20  temp[12] = mul2[state[12]] XOR mul3[state[13]] XOR state[14] XOR state[15]
21  temp[13] = state[12] XOR mul2[state[13]] XOR mul3[state[14]] XOR state[15]
22  temp[14] = state[12] XOR state[13] XOR mul2[state[14]] XOR mul3[state[15]]
23  temp[15] = mul3[state[12]] XOR state[13] XOR state[14] XOR mul2[state[15]]
24
25  return temp
26 }
27

```

Decryption

Decryption is just encryption, but in reverse. This uses the inverse functions of each function used to encrypt the data. Here is the algorithm:



It is literally just the encryption algorithm in reverse.

Before decryption, the exact same steps need to be taken as in encryption, apart from the padding because all the blocks should have already been encrypted, so each block should be 16 in length.

Inverse Add Round Key:

Add round key is it's own inverse, as XOR is the same forwards as it is backwards.

Inverse Sub Bytes:

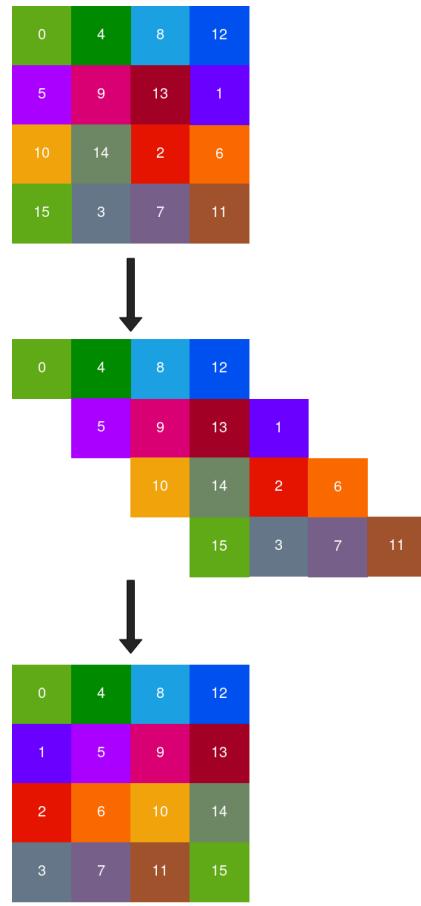
Inverse sub bytes is the same as sub bytes, it just has an inverse of the S-Box.

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Inverse Shift Rows:

Inverse shift rows does what shift rows does, but shifts each row right instead of left.

In the diagram below it takes the shifted data and orders it again.



Inverse Mix Columns:

Inverse mix columns works the same as normal mix columns, but with a different matrix to multiply each element with:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

The a's are the original data, the r's are the encrypted data.

Just like with normal mix columns, you can just use lookup tables for each possible answer to each possible input.

And that's all for AES.

SHA256:

SHA256 (in the Secure Hash Algorithm 2 family) takes an input of 32 bytes (256 bits), and gives a 32 byte output based on the input, but is meaningless. This is useful for passwords, or pin codes like in my program, where you don't want the original password to be known, but for the password to still be unique.

A small difference in the input gives you a drastic change in the output. For example, if I put in:

```
1 | "test string"
```

I get:

```
1 | d5579c46dfcc7f18207013e65b44e4cb4e2c2298f4ac457ba8f82743f31e930b
```

But when I put in:

```
1 | "test strinh"
```

I get:

```
1 | 4e4d20e9fc77e913bf56cc69a2b4685d761a9e44d833198612e80a72cd563f1
```

A vastly different output to the one above. This is important, as there should be no pattern to the output, otherwise the original password could be guessed based off of similar inputs.

Now you might be asking "Why are you using 256 bit SHA, when size key you need for AES is 128 bits?". It is because the more bits you have, the less likely you are to have collisions with other inputs. The security of SHA-1 (128 bit SHA) (measured in bits) is less than 63 bits due to collisions (if it was fully secure it would be the full 128 bits).

What I am doing instead, is taking the output of SHA256, splitting it in half, and XORing each half with each other to get a 128 bit output. This doesn't affect how secure it is, as you still have the extra step of XOR, making it still more secure than SHA-1.

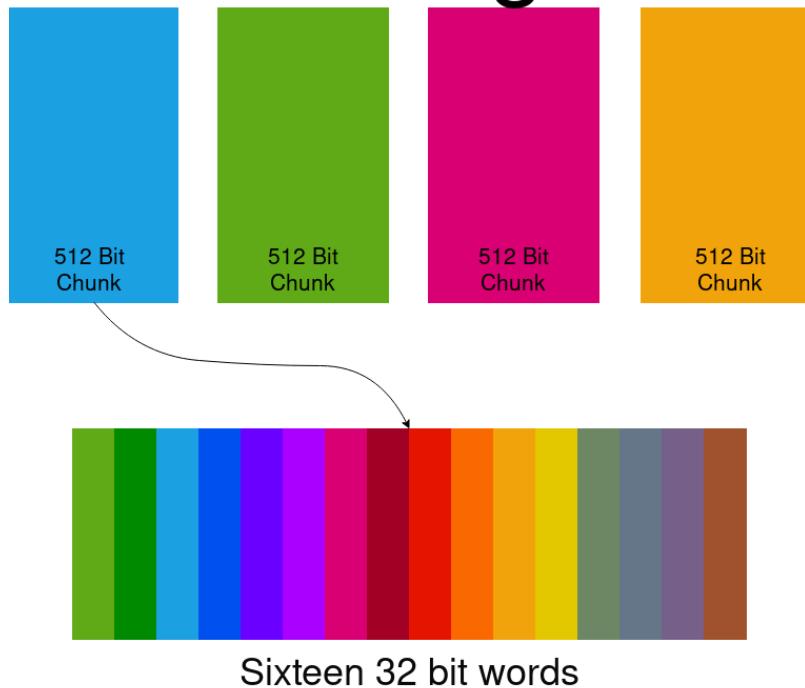
The Algorithm:

Bear in mind that SHA works on a bitwise level, so while I will be explaining it, I will be talking in terms of bits.

How the message is handled:

When doing operations on the data, it will be done in 32 bit words. The message is split into 512 bit blocks, containing sixteen 32 bit words.

Message



SHA is operates on every 32 bit word.

Since the maximum key size for my AES will be 16 bytes (128 bits), I don't need to worry about splitting the message into 512 bit chunks, as the input will only ever be 128 bits as SHA will only ever be used for the AES key. So, for the examples below I won't go into detail on how a message bigger than 512 bits will be handled.

Before the operation starts:

Before we start, we need to **pad the message** \$M\$ so that it is 512 bits in length.

Let \$l\$ = the length of the message \$M\$.

First, we need to append the bit \$1\$ to the end of the message, followed by \$k\$ \$0\$ bits, where \$k\$ is the smallest positive solution to the equation:

$$l + 1 + k \equiv 448 \pmod{512}$$

To get \$k\$, the algorithm would look something like this (I wrote this in Python 3):

```
1 | k = 0
2 | while ((l+1+k) - 448) % 512 != 0:
3 |     k += 1
```

Then, you append the binary representation of the length of the message \$l\$ as a 64 bit binary number. This makes the message 256 bits in length.

Let's do an example: \$M\$ = "i don't know".

$$l = 12 \times 8 = 96$$

Append a "1":

$$M = b"\text{i don't know"} + 1$$

$$448 - (96 + 1) = 351 \text{ Zero Bits}$$

$$M = b"\text{i don't know"} + 1 + 351(0s)$$

$$l = 96 = 01100000$$

Final Padded Message:

$$M = b"\text{i don't know"} + 1 + 351(0s) + 56(0s) + 01100000$$

The message has to be 512 bits in length so that it works with the calculations later.

Then, we also need to **set the initial hash values** for each word in the current block. The initial hash values set by the creators of SHA:

"These words were obtained by taking the first thirty-two bits of the fractional parts of the square roots of the first eight prime numbers. "

$$H_0 = 6a09e667$$

$$H_1 = bb67ae85$$

$$H_2 = 3c6ef372$$

$$H_3 = a54ff53a$$

$$H_4 = 510e527f$$

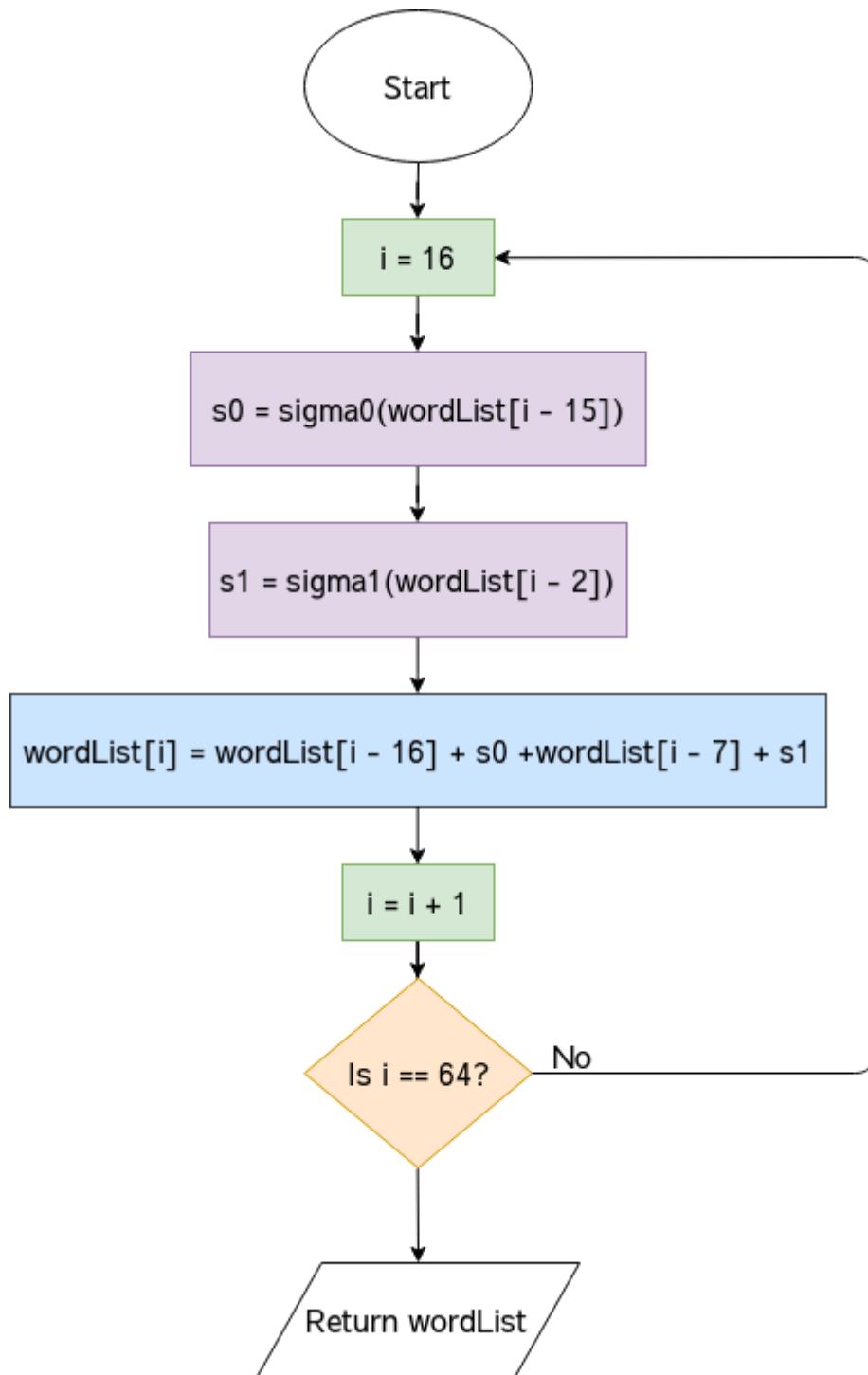
$$H_5 = 9b05688c$$

$$H_6 = 1f83d9ab$$

$$H_7 = 5be0cd19$$

Next, each 32 bit word in the message has to be expanded from 32 bits to 64 bits.

Here is the algorithm:



To do this, we need two functions, sigma 0 σ_0 and sigma 1 σ_1 .

Sigma 0 (Expansion) (σ_0):

Sigma 0 (Expansion) looks like this:

$$\sigma_0(x) = (x >>> 7) \oplus (x >>> 18) \oplus (x >> 3)$$

$>>>$ means that we rotate the 32 bit word x right by the number given (y). What this does is shift the bytes along y places to the right, and wraps them around to the start of x .

I will do an example of $>>>$ with a 4 bit nibble:

$$\begin{aligned}
 f(x) &= x >>> 1 \\
 f(1011) &= 1011 >>> 1 \\
 f(1011) &= 1101
 \end{aligned}$$

As you can see, the \$1\$ bit at the end gets moved to the front, as I shifted it right by 1.

\$>>\$ Means shift the 32 bit word \$x\$ right by the number given (\$y\$). This is different from \$>>>\$, because instead of wrapping the bits around to the beginning of the word again, we just shove a \$0\$ bit at the front instead.

\$\bigoplus\$ is just XOR.

For example:

$$\begin{aligned}
 f(x) &= x >> 1 \\
 f(1011) &= 1011 >> 1 \\
 f(1011) &= 0101
 \end{aligned}$$

$$\begin{aligned}
 g(x) &= x >> 2 \\
 g(0101) &= 0101 >> 2 \\
 g(0101) &= 0001
 \end{aligned}$$

Here the byte is shifted right, and the bytes are removed as they are shifted.

Sigma 1 (Expansion)(\$\sigma_1\$):

Sigma 1(Expansion)(\$\sigma_1\$) is the same as Sigma 0 (Expansion)(\$\sigma_0\$), apart from how much you rotate and shift the word:

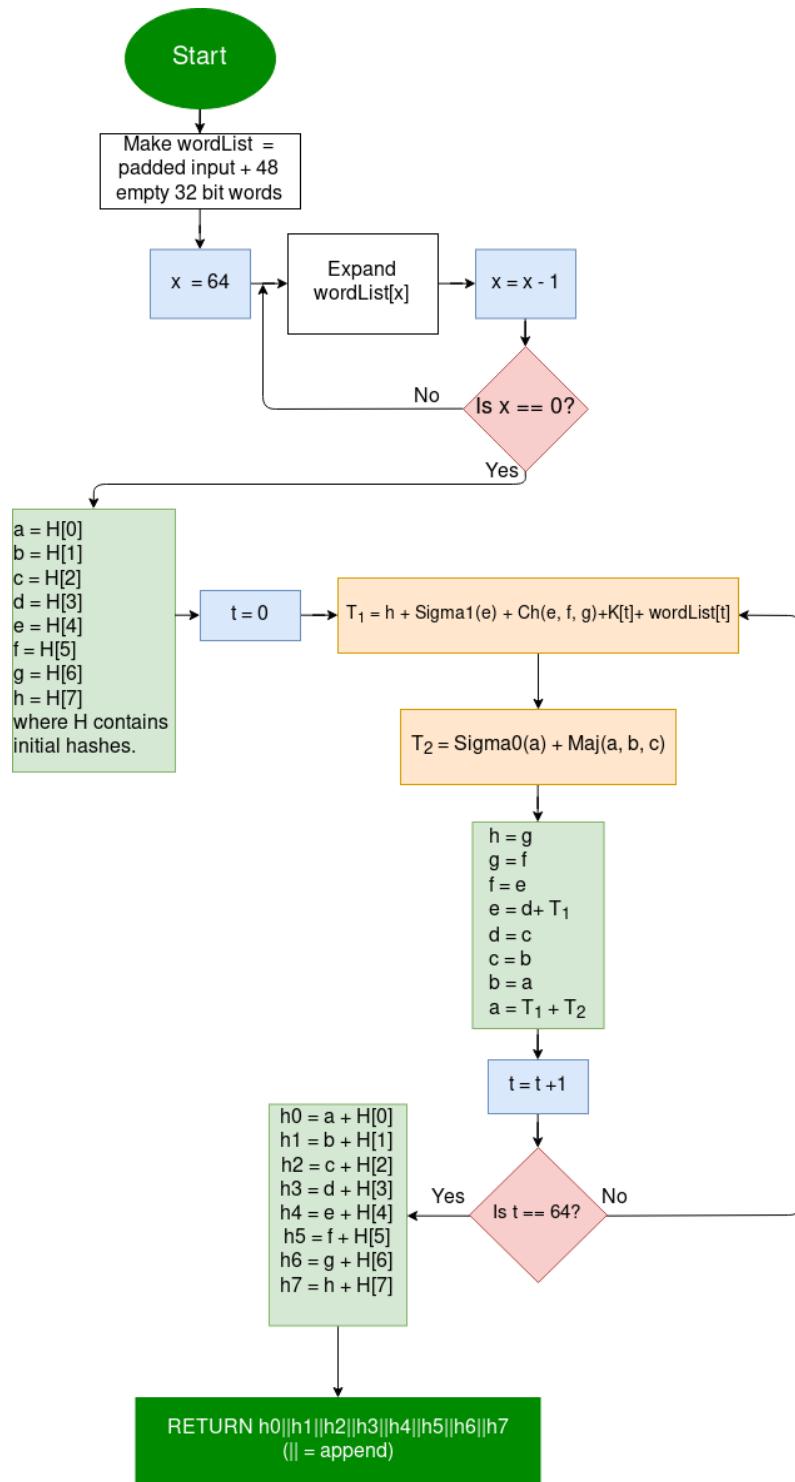
$$\sigma_0(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10)$$

The operation:

All addition is MOD(2^{32}).

Here is the full algorithm:

Figure 2 (Found larger on "Large Images" section)



In the diagram above, H is the array of initial hash values discussed earlier, wordList is a 2D array containing the 32 bit words. `||` means append, so `$h0 || h1 || h2 || ...$` just appends the items together. K is the array with the round constants in (see <https://csrc.nist.gov/csrc/media/publications/fips/180/4/archive/2012-03-06/documents/fips180-4.pdf> section 4.2.2).

The step "Expand wordList[x]" is covered in the section above.

All of the SHA functions operate on 32 bit words, and return a new 32 bit word. I will now explain what the functions Sigma0 (`\$Sigma_0$`), Sigma1 (`\$Sigma_1$`), Ch and Maj.

Sigma 0 (\$\Sigma_0\$):

\$\Sigma_0\$ is this equation:

$$\Sigma_0(x) = (x >>> 2) \oplus (x >>> 13) \oplus (x >>> 22)$$

This looks confusing, but let me break it down.

\$>>>\$ means that we rotate (shift and move displaced numbers to the begining/end of the number) the number right by the number specified.

\$\oplus\$ means that we XOR the items either side with each other.

Here is an example of the rotate function:

$$A = 1001110 \\ A >>> 2 = 1010011 \quad \text{The last two bits are moved to the end.}$$

Let me do an example with a 32 bit word:

$$A = 1001011101101111000110111011101 \\ \Sigma_0 = (1001011101101111000110111011101 >>> 2) \oplus (1001011101101111000110111011101 >>> 13) \oplus \\ \dots (1001011101101111000110111011101 >>> 22) \\ (1001011101101111000110111011101 >>> 2) = 011001011101101111000110111011101 \\ \text{The two bits at the end have been moved to the front one by one.} \\ (1001011101101111000110111011101 >>> 13) = 11110001101110111011001011101101 \\ (1001011101101111000110111011101 >>> 22) = 01110111011001011101101111100011 \\ 01100101110110111100011011101110111011001011101101111100011 \\ = 11100011000001011000101001111001$$

Sorry if that is a bit small.

It isn't too difficult it's just understanding what the \$>>>\$ does.

Sigma 1 (\$\Sigma_1\$):

Sigma 1 (\$\Sigma_1\$) is pretty much the same as \$\Sigma_0\$, the only difference being the amount you rotate by:

$$\Sigma_1(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25)$$

Ch:

The Ch function looks like this:

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$

This also looks a bit confusing, but it really isn't too bad.

The \$\wedge\$ symbol is the bitwise operator AND.

The \$\oplus\$ symbol is the bitwise operator XOR.

The \$\neg\$ symbol is the bitwise operator NOT.

I will do one example run with Ch with three 4 bit nibbles to keep it simple:

$$Ch(1011, 1001, 0011) = (1011 \wedge 1001) \oplus (\neg 1011 \wedge 0011)$$

$$\begin{array}{r} 1011 \\ \hline \wedge 1001 \\ = 1001 \end{array}$$

$$\begin{aligned} Ch(1011, 1001, 0011) &= 1001 \oplus (\neg 1011 \wedge 0011) \\ \neg 1001 &= 0110 \end{aligned}$$

$$\begin{array}{r} 0110 \\ \hline \wedge 0011 \\ = 0010 \end{array}$$

$$Ch(1011, 1001, 0011) = 1001 \oplus 0010$$

$$\begin{array}{r} 1001 \\ \hline \oplus 0010 \\ = 1011 \end{array}$$

$$Ch(1011, 1001, 0011) = 1011$$

Maj:

the Maj function looks like this:

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

You should recognise the symbols in this one, since they appear in the other ones used in SHA that we have covered. Here is an example with three 4 bit nibbles:

$$Maj(1011, 1001, 0011) = (1011 \wedge 1001) \oplus (1011 \wedge 0011) \oplus (1001 \wedge 0011)$$

$$\begin{array}{r} 1011 \\ \wedge 1001 \\ \hline = 1001 \end{array}$$

$$\begin{array}{r} 1011 \\ \wedge 0011 \\ \hline = 0011 \end{array}$$

$$\begin{array}{r} 1001 \\ \wedge 0011 \\ \hline = 0001 \end{array}$$

$$Maj(1011, 1001, 0011) = 1001 \oplus 0011 \oplus 0001$$

$$\begin{array}{r} 1001 \\ 0011 \\ \oplus 0001 \\ \hline = 0101 \end{array}$$

$$Maj(1011, 1001, 0011) = 0101$$

BLAKE 2b:

BLAKE was a finalist in the SHA 3 contest. The SHA 3 contest was announced on November 2nd 2007, as a new hash function was needed, that was very different from the SHA 2 family of hash functions in case a huge issue was found with the SHA 2 family.

BLAKE did not win, as it was too similar to SHA2:

"desire for SHA-3 to complement the existing SHA-2 algorithms ... BLAKE is rather similar to SHA-2."

<https://blake2.net/acns/slides.html>

However, BLAKE was the fastest out of all of the competitors (at 8.4 cycles per byte, cycles being the fetch decode execute cycle of a processor), and was tested to be secure. This meant that even though BLAKE did not win the competition, it is still used in numerous programs. Due to BLAKE's speed, it is ideal for getting the checksum of large data.

No preparations have to be done so lets just jump right into the algorithm.

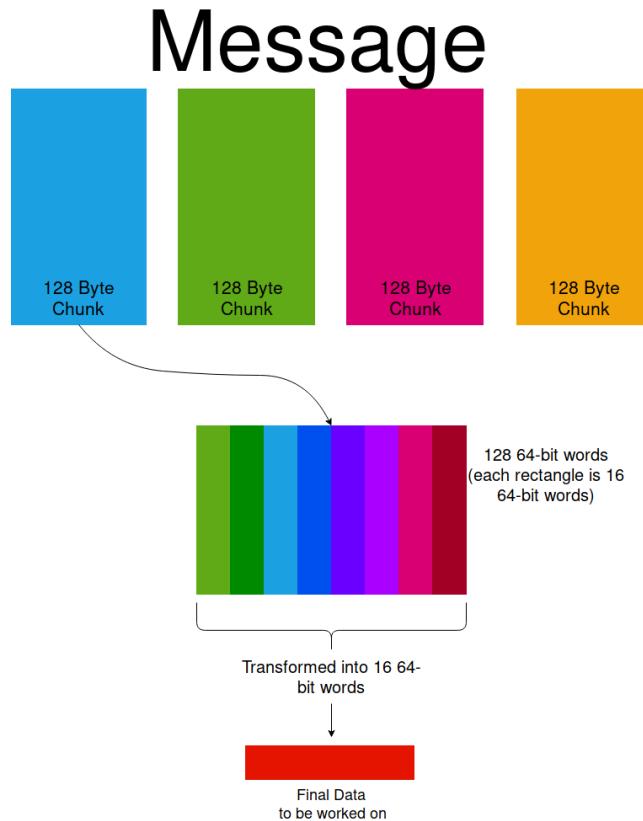
The Algorithm:

How the data is read:

8 initial hash values of size 64-bits are initialised at the start (using pre-defined values), and these are worked on throughout the program.

The data is read in 128 bytes, where each byte is then converted into a 64-bit word (just shove some 0s on the front). Each chunk is operated on using the 8 hash values, creating 8 new hash values. These new hash values are used in computation using the next block and so on.

Here is a diagram showing how the data is converted into data that can be processed:

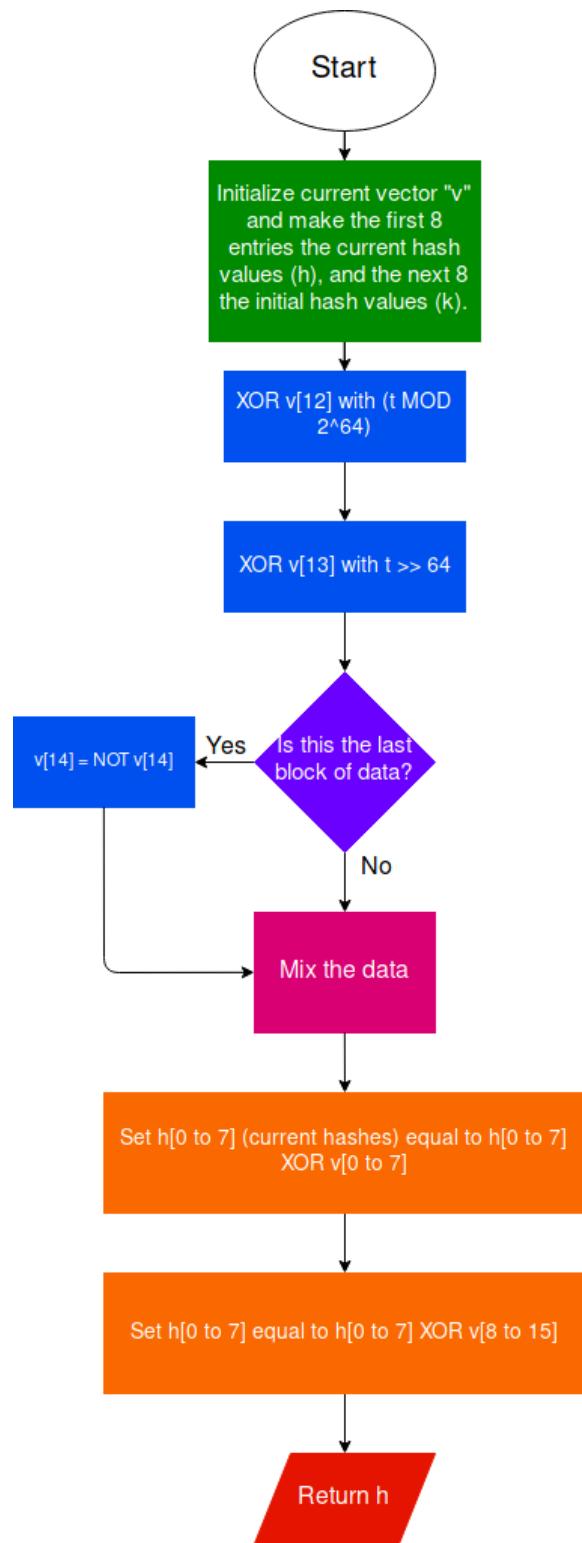


To transform a list of 16 64-bit words into 1 64-bit word, you do this algorithm (where \$a\$ is the list of words):

$$new = a[0] \oplus (a[1] \ll 8) \oplus (a[2] \ll 16) \oplus (a[3] \ll 24) \oplus (a[4] \ll 32) \oplus (a[5] \ll 40) \oplus (a[6] \ll 48) \oplus (a[7] \ll 56)$$

What this does is XOR's the bytes in the array with each other in a way that produces a single word at the end.

The operation:



Each block has to be compressed and returned as 8 hash values. Above is the compression function. \$t\$ is the number of bytes in total that have been compressed so far, \$h\$ is a list of the 8 current hashes, and \$k\$ is the list of 8 initial hash values set here <https://tools.ietf.org/pdf/rfc7693.pdf> section 2.6, the same initial hash values of SHA512.

The operation is quite simple compared to other hash functions like SHA512, as it was built for speed.

The **Mix the data** step looks like this:

```

1  for i := 0 to 12
2    v = mix(v, 0, 4, 8, 12, m[sigma[i][0]], m[sigma[i][1]])
3    v = mix(v, 1, 5, 9, 13, m[sigma[i][2]], m[sigma[i][3]])
4    v = mix(v, 2, 6, 10, 14, m[sigma[i][4]], m[sigma[i][5]])
5    v = mix(v, 3, 7, 11, 15, m[sigma[i][6]], m[sigma[i][7]])
6
7    v = mix(v, 0, 5, 10, 15, m[sigma[i][8]], m[sigma[i][9]])
8    v = mix(v, 1, 6, 11, 12, m[sigma[i][10]], m[sigma[i][11]])
9    v = mix(v, 2, 7, 8, 13, m[sigma[i][12]], m[sigma[i][13]])
10   v = mix(v, 3, 4, 9, 14, m[sigma[i][14]], m[sigma[i][15]])

```

Sigma (σ) is a 2-dimensional array containing some constant values, that determine what index of the current working vector v (a 16 length array of 64-bit words) will be mixed with what other index of v .

Sigma is defined here: <https://tools.ietf.org/pdf/rfc7693.pdf> section 2.7 as:

$$\begin{aligned}
\sigma[0] &= [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] \\
\sigma[1] &= [14, 10, 4, 8, 9, 15, 13, 6, 1, 12, 0, 2, 11, 7, 5, 3] \\
\sigma[2] &= [11, 8, 12, 0, 5, 2, 15, 13, 10, 14, 3, 6, 7, 1, 9, 4] \\
\sigma[3] &= [7, 9, 3, 1, 13, 12, 11, 14, 2, 6, 5, 10, 4, 0, 15, 8] \\
\sigma[4] &= [9, 0, 5, 7, 2, 4, 10, 15, 14, 1, 11, 12, 6, 8, 3, 13] \\
\sigma[5] &= [2, 12, 6, 10, 0, 11, 8, 3, 4, 13, 7, 5, 15, 14, 1, 9] \\
\sigma[6] &= [12, 5, 1, 15, 14, 13, 4, 10, 0, 7, 6, 3, 9, 2, 8, 11] \\
\sigma[7] &= [13, 11, 7, 14, 12, 1, 3, 9, 5, 0, 15, 4, 8, 6, 2, 10] \\
\sigma[8] &= [6, 15, 14, 9, 11, 3, 0, 8, 12, 2, 13, 7, 1, 4, 10, 5] \\
\sigma[9] &= [10, 2, 8, 4, 7, 6, 1, 5, 15, 11, 9, 14, 3, 12, 13, 0]
\end{aligned}$$

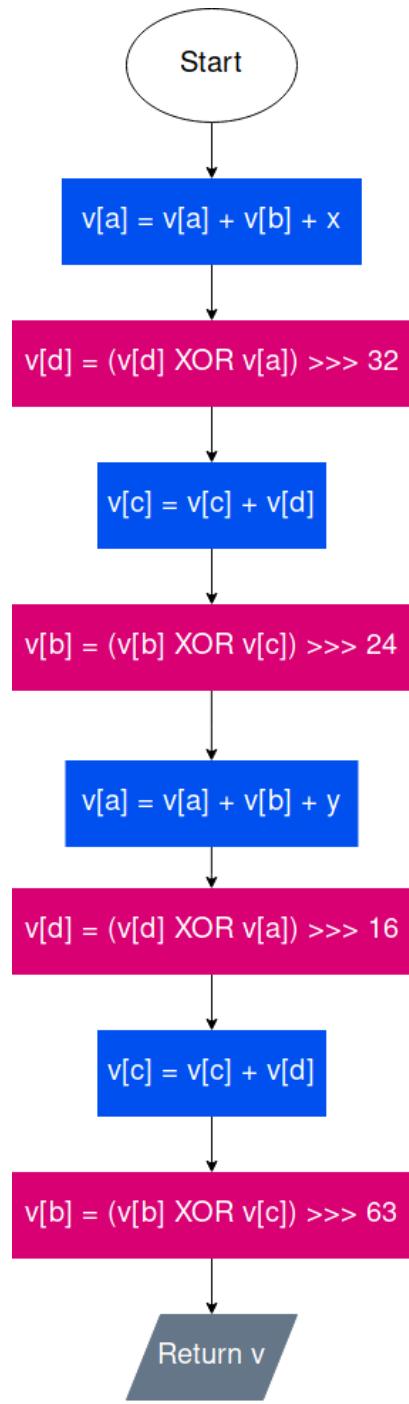
σ is defined for BLAKE2s, and BLAKE2s only has 10 rounds, while BLAKE2b has 12, so σ_0 and σ_1 are repeated again to make the array 12 in length.

Notice that in the first lot of mixing, the vector is mixed row by row normally (with the same indexing as AES), but in the second lot of mixing, the indices change. They shift each column up depending on the column. Column 0 is shifted 0 places, column 1 is shifted 1 place up, column 2 is shifted 2 places up, and column 3 is shifted 3 places up. This is a much better way of shifting each column than doing it before hand.

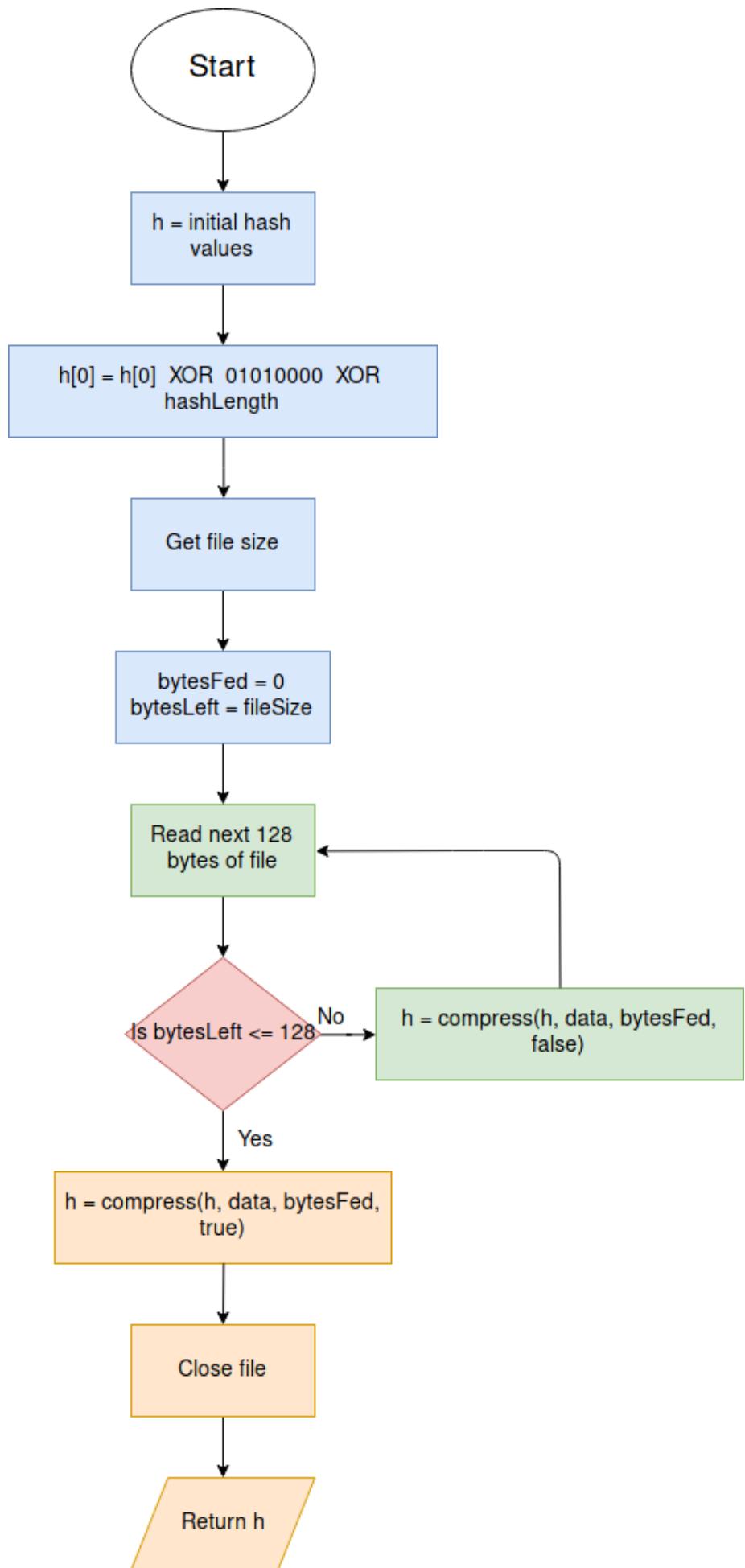
The main mixing function takes the inputs:

$$mix(v, a, b, c, d, x, y)$$

Where v is the current vector (16 64-bit words), a , b , c , d , x , and y are the indices of the working vector you want to work with. Here is the main mixing algorithm:



So all together, this is the BLAKE2b checksum algorithm:



The second step ($h[0] = h[0] \oplus 01010000 \oplus hL$) XORs $h[0]$ with $0101kknn$, where kk is the length of the key (which is optional, so I probably will never use it), and $n nn$ is the hash length desired.

Quick Sort

My program will need a quick sort for sorting the files by:

- Size
- Name

I have chosen quick sort because it is quicker than most sorts (it's in the name!) with a big-O notation of $O(n \log n)$ on average, with the worst case being $O(n^2)$. Merge sort has a big-O notation of $O(n \log n)$, and worst case of $O(n \log n)$, so why am I not using merge sort? Merge sort is supposed to be quicker mathematically, however merge sort has to access the array of items more often, which puts more strain on the memory of the computer, which is considerably slower in comparison to the CPU in the majority of computers. Here is a good video comparing merge sort and quick sort (along with a few other algorithms): <https://youtu.be/ZZuD6iUe3Pc>

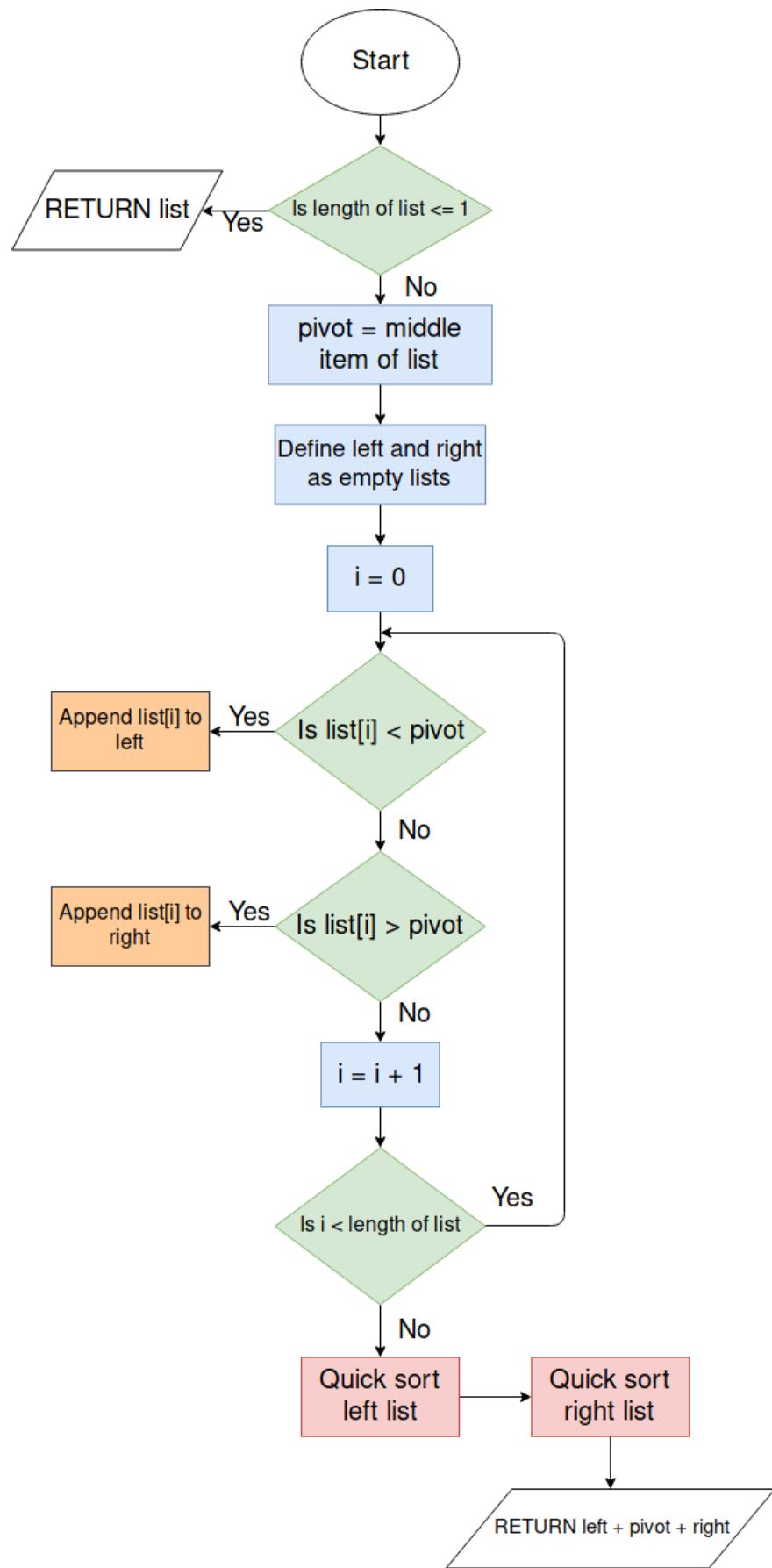
The algorithm goes like this (using a list of items to be sorted):

1. Take the item in the middle of the list. Call this the "pivot".
2. Compare each item either side of the pivot. If the item is bigger than the pivot, add it to a new list called "right", if the item is smaller than the pivot, add the item to a new list called "left".
3. Then repeat this process with the left and right lists (making this algorithm recursive).
4. Once the current left and right lists have been sorted, append the left list and right list with the pivot in the middle.

Here is the pseudocode of the algorithm:

```
1 function quickSort(list)
2     if length(list) <= 1 then
3         return list
4     end
5
6     left  = []
7     middle = []
8     right = []
9     pivot = list[int(length(list)/2)]
10    for i = 0 to length(list) do
11        if list[i] < pivot then
12            left.append(list[i])
13        else if list[i] > pivot then
14            right.append(list[i])
15        else
16            middle.append(list[i])
17        end
18    end
19
20    return quickSort(left)+middle+quickSort(right)
21 end
```

Here is a flow diagram to represent this:

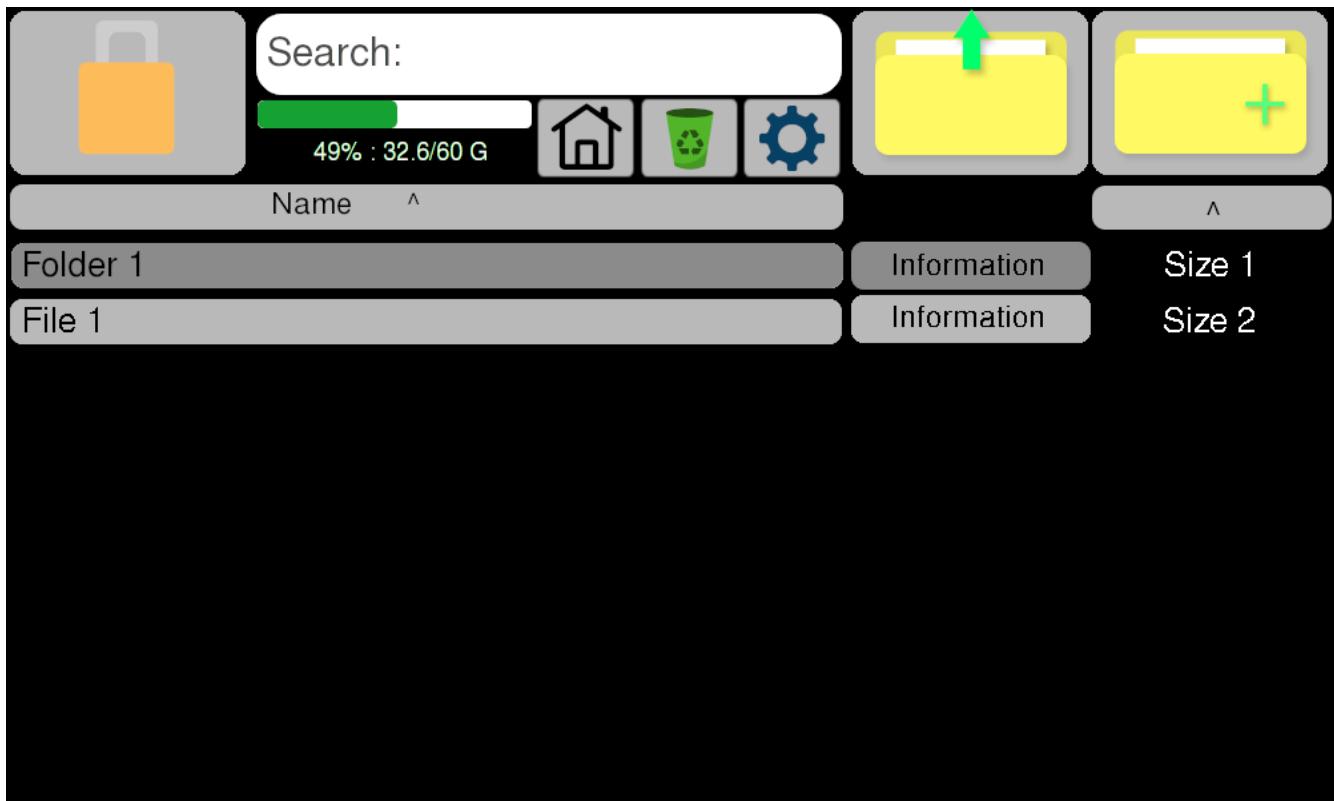


UI Research:

For the UI of both apps, I will use Kivy (a Python module) to make both the mobile app and the PC program. I have chosen Kivy because using it on both the app and the main program means that the design will stay consistent, and Kivy does look quite nice "out of the box".

Main Program (on PC):

Taking all of the points in my **Analysis** into consideration, here is a possible design for the UI of my program:



Everything grey is a clickable button. This helps the user distinguish between buttons and information. The most important buttons are large, as they will be used the most. The user can sort by name or size, and can search the entire vault for a search term.

The information button displays more information, such as:

- The time the file (if it is a file) was added to the vault.
- The full directory path from the vault.
- The size of the file/folder.
- The option to delete the file/folder.

The button with the home picture on it takes the user back to the root directory of the vault. The recycling bin button is for the recycling folder, where the files that have been deleted can be either restored or deleted. The cog wheel button is settings, where all the settings are kept. I gave the settings its separate section to avoid clutter, as most users will probably not need to use it very often.

The user can sort the files by name alphabetically, or they can sort by size. Space remaining on the current device is shown underneath the search bar.

While searching through large folders, the search results should update every so often since it may take a while to search the full file tree.

When using the recycling bin, the program will look exactly the same, but warn the user that they are in the recycling bin "mode", so when they click files, instead of decrypting the file and opening it the file is instead moved back into the vault, recovering it to where it originally came from.

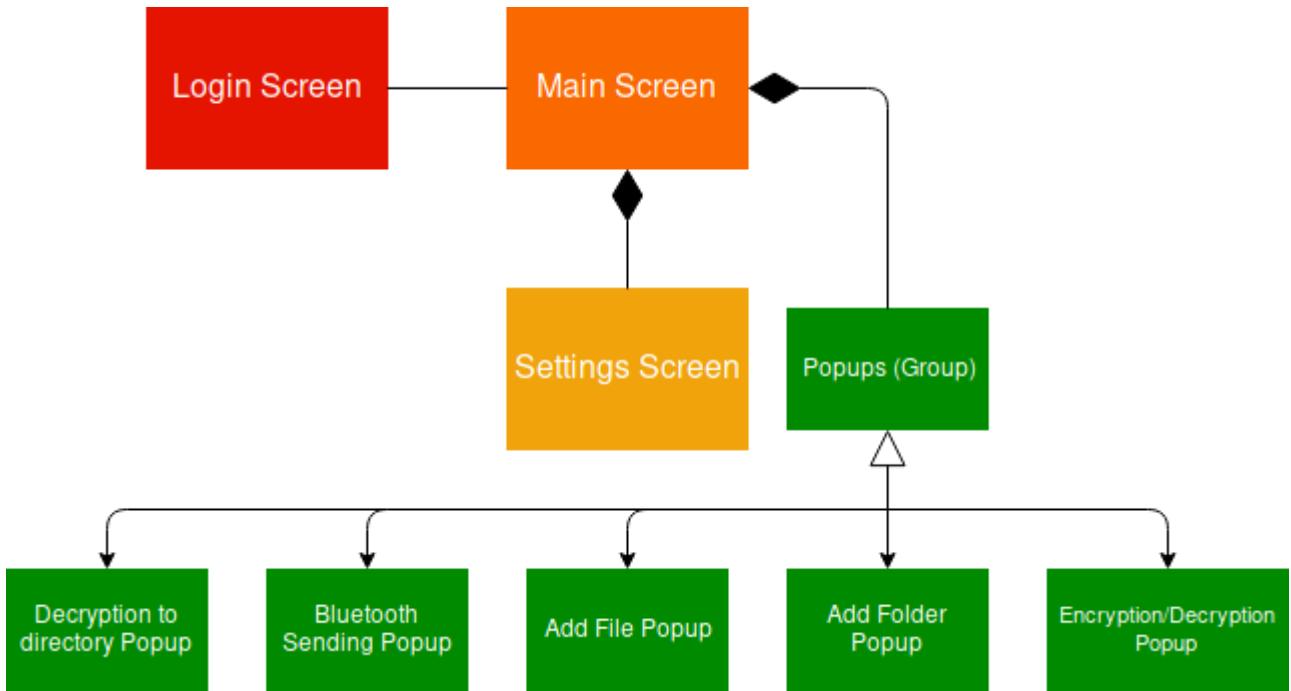
The login screen will have 2 modes:

1. Login without Bluetooth (can't use any Bluetooth functions while logged in).
2. Login with Bluetooth.

I will also make it so that you can easily switch between Bluetooth and non-Bluetooth login, whether that be a button on the login screen, or in the configuration file. Also, when in non-Bluetooth mode, the user will not need to have PyBluez installed, neither will they need Bluetooth on their PC.

When navigating the app, the navigation should be easy and simple so that the user does not get lost. I will have 2 main screens, a login screen and a main screen (to view files and open other functions once logged in), and within the main screen I will have a screen for settings, and a few other popups.

Here is a class diagram to show the relationship between screens and popups:



These are only the custom classes, so regular buttons and labels and such will be left out of this diagram.

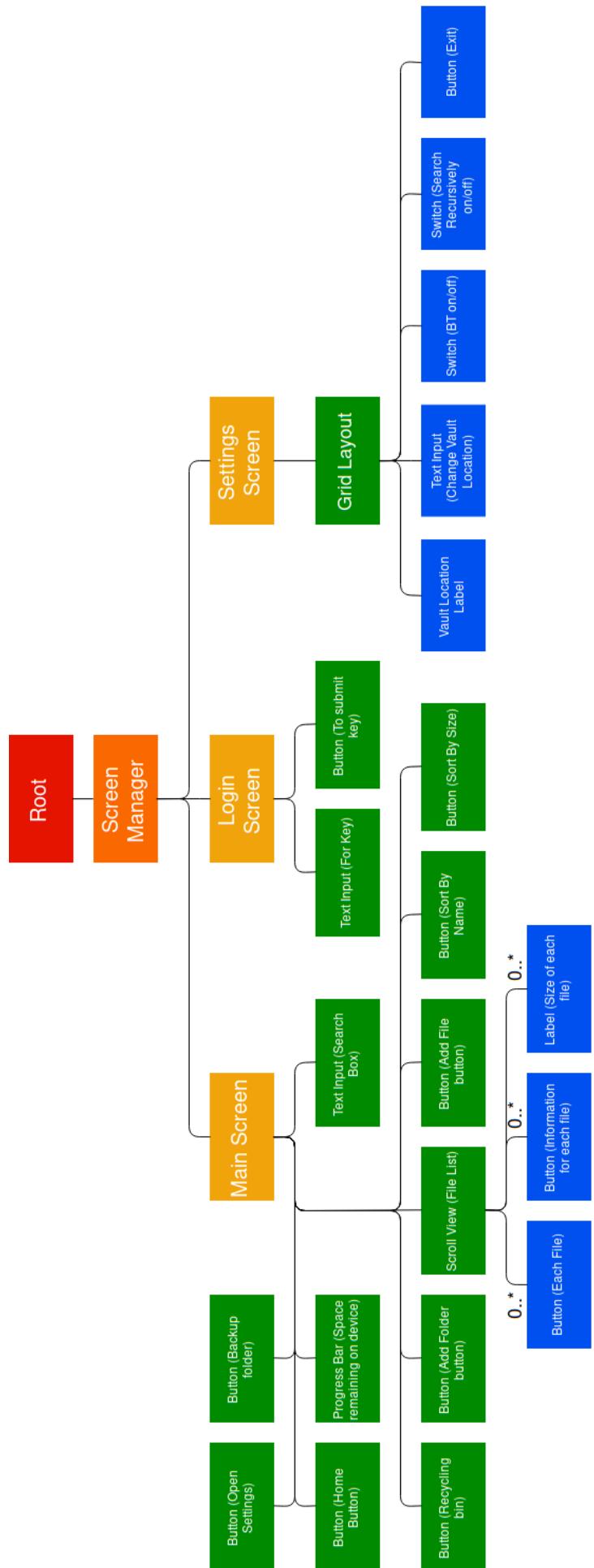
The Encryption/Decryption popup should be opened when the user encrypts/decrypts a file, and should display information including how fast the file is being enc/decrypted (in kb or mb per second), the percentage of the file that has been enc/decrypted so far, and how many items have been done out of the total files to be enc/decrypted. There should also be a progress bar at the bottom, showing the percentage visually.

The Bluetooth sending popup should show the exact same information, but for the current status of the file being sent over Bluetooth.

The add file and add folder popups should both be similar in design, however the add file popup will let the user encrypt a file or folder to the vault, while the add folder popup will allow the user to create a new folder within the vault.

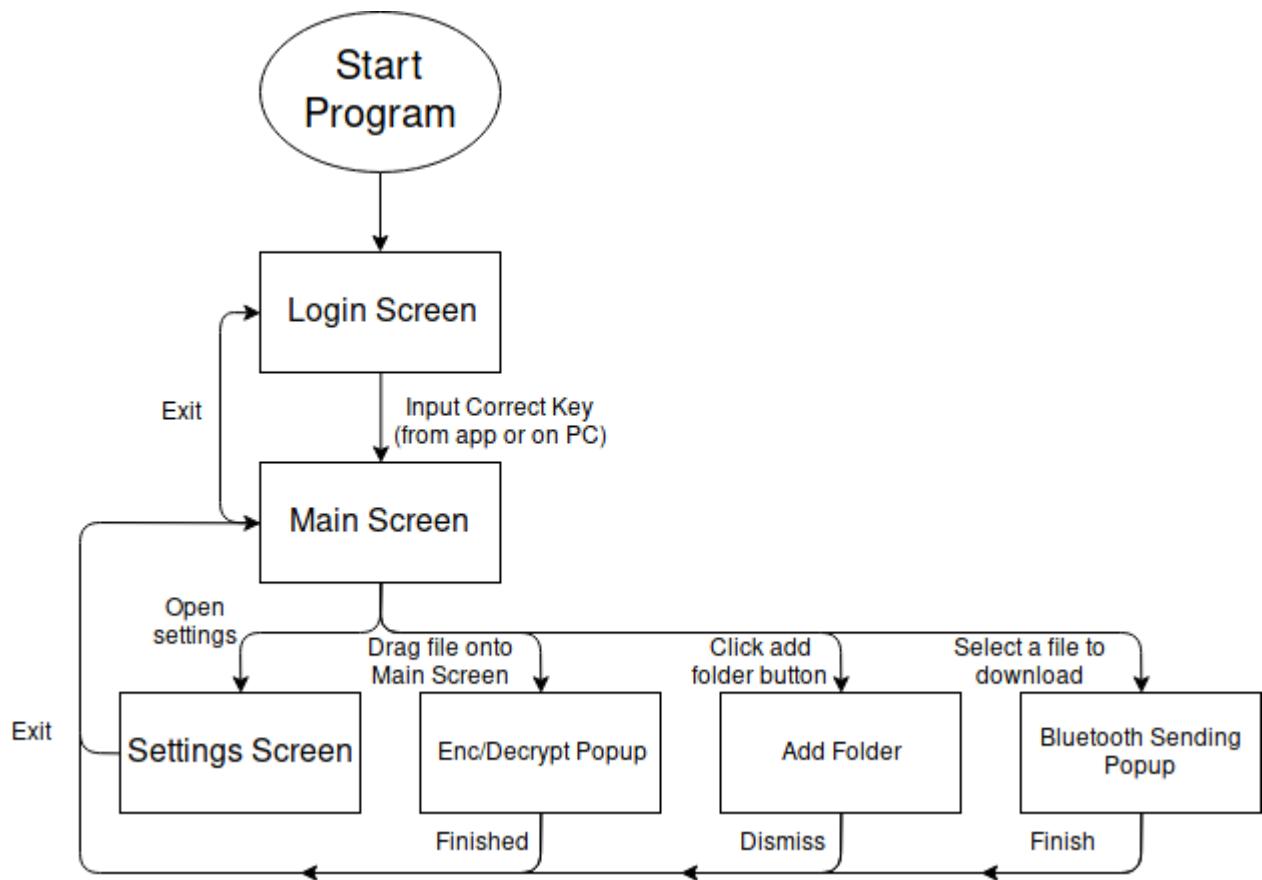
Popups are designed for one purpose only, and are usually used briefly before they are closed again. Screens will be used throughout the program, acting as the base of the GUI, where child widgets can be added to the screen, such as buttons, text inputs and views (such as scroll views). The screens will inherit from Kivy's Screen class, and the popups will inherit from Kivy's Popup class. The screens get managed by a ScreenManager, also a Kivy widget. The ScreenManager is then added to the app's root widget (the base widget of an app).

A hierarchy diagram for the entire GUI would look something like this (since Popups can be added and removed to any widget when needed, I will not include them in this diagram):



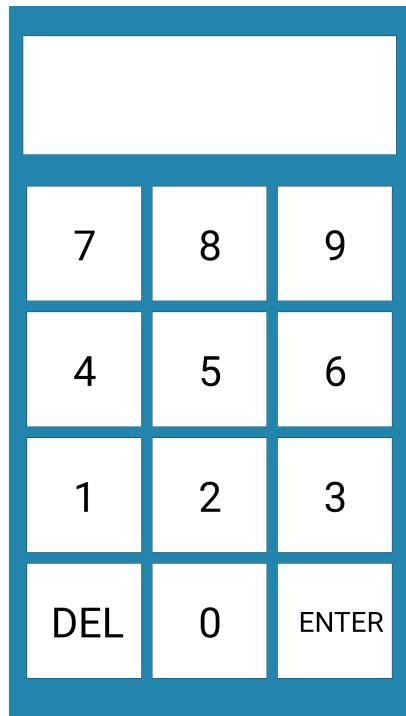
Each layer has its own colour, since I couldn't think of a better way of making this clear without making the image extremely wide. "0..*" means 0 to many of this widget can exist at any time. This shows all of the widgets that will be on each screen at all times (unless obstructed by a popup) as default.

Here is a top-down view of how the GUI will flow while the user is using the program:



The App:

The app's UI design should be very simple, as I do not need to add much. All it needs to be is a number pad with a display, an enter button and a screen to have open while you are connected to the PC, and a file browser similar to the one on the PC app. Here is a prototype I made in Processing (A java based "software sketchbook):



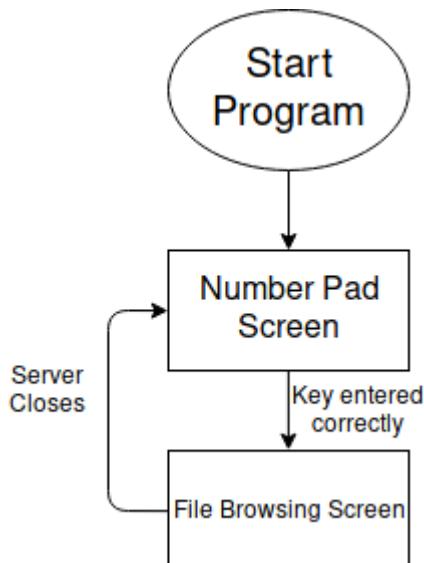
It is very minimal, as I decided to keep it as minimal as possible so that the user doesn't get confused, and to keep clutter at a minimum.

Once the vault is unlocked, the user should be given the option to browse files in the vault from their phone, and select files to download, or instead just minimise the app and continue using their phone. The vault should only close once the user has exited the app, rather than when they minimise the app.

The user should be able to browse the folders independently from the computer program (so both programs can be looking at different folders), browsing the files should be a seamless experience, and when searching for files, the searching work should be done on the computer so that precious phone battery is not wasted, and also because it is quicker in general to just send the search results to the mobile once they are generated.

The app should have a pin-code screen and a file browsing screen. The pin-code screen should only be used when the PC program is logged out.

Here is a top-down diagram of how the GUI will flow while the user is using the program:



The program as a whole:

My program will handle a fair amount of data, so here is a IPSO (Input, Processing, Storage, Output) chart to simplify it a little:

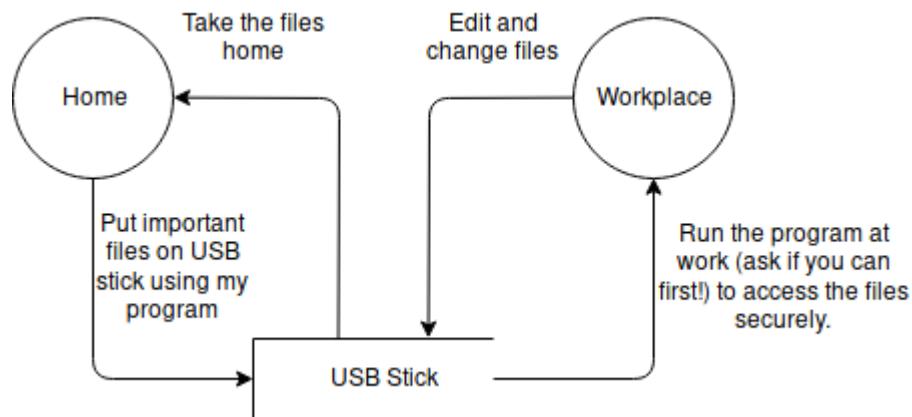
IPSO	Program Section	Item
Input	Login	Key (From user input). Vault directory path. Files in the Vault (for checking the key).
	File Browser (Main Screen)	Vault directory path. Recycling folder path (for when files are deleted). The key (for displaying encrypted file names).
	Search Bar (Main Screen)	Search item entered by user.
	Settings	Configuration file path. Current settings.
	Encryption/Decryption	The file path of the file that is desired to be enc/decrypted. The path to write the new data to. The key.
	Add Folder Popup	The name of the new folder to be created.
	Add File Popup	The path of the file to be encrypted to the vault.
	Recycling bin folder	Recycling folder path. Where each file came from originally.
Processing	Login	Decrypt the first block of the first file you find in the Vault, and check that it is equal to the key entered.
	File Browser (Main Screen)	Getting the sizes of each file. Sorting the files by name or size. Decrypting files when files are clicked. Encrypting files when files/folders are dragged into the window, or if a file/folder is added via the add file popup. Changing directory when a folder is clicked.
	Search Bar (Main Screen)	Search recursively for the file/folder in the Vault, or if recursive search is disabled in settings just search the current directory that the file browser is in.

	Settings	Change settings in the configuration file when changed in the program.
	Encryption/Decryption	Encrypt/Decrypt the file given using the key given.
	Add Folder Popup	Create the new folder in the current directory of the file browser.
	Recycling Bin Folder	Move files selected to original position.
Storage	File Browser (Main Screen)	Read the current files in the current directory that the file path is in.
	Settings	Read from the configuration file, and write to the configuration file when settings are changed.
	Add Folder Popup	Make new directory in the current directory the file browser is in.
	Encryption/Decryption	Read data from the file to be enc/decrypted, and write the enc/decrypted data to the location specified.
	Recycling Bin Folder	Read the file names of the files in the recycling bin.
Output	Login	Change the screen to Main Screen if the key is correct, otherwise create a Popup telling the user that the key is incorrect.
	File Browser (Main Screen)	Display the files in the Vault sorted how the user has specified, along with the size of each file, and a more information button.
	Search Bar	Return the list of closest matches to the search item given.
	Settings	Edit changes to file, and return values of each setting to the main program.
	Add File Popup	Pass the file path of the file to be added to the encryption function, with the path in the Vault where the new data should be written to.

There are many different use cases for my program. Some people may want to travel with the data, some people may just want to use it on one computer. In this section I will outline different ways I intend my program to be used.

Using a USB stick:

People who want to take the data with them to other places, a USB stick is a good idea. All the user has to do is download my program, put it on the USB stick and set the vault directory as a directory on the USB stick. No more setup should be needed. The program should be able to run on Windows, MacOS and Linux so using the USB on most devices should not be an issue. Here is a data flow diagram showing how the user may handle the data:

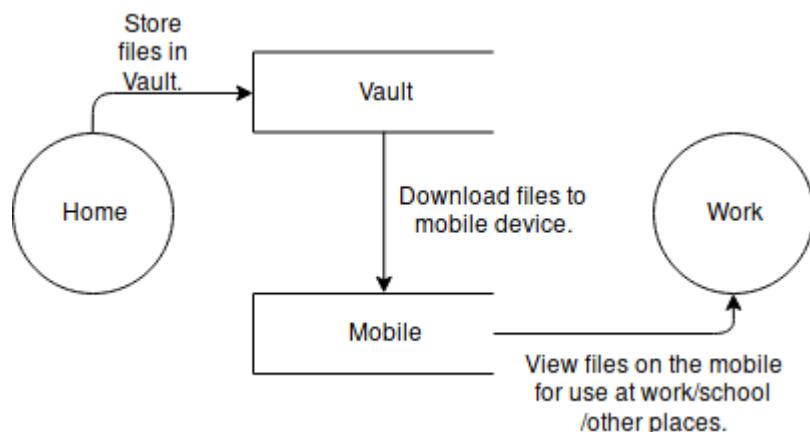


Storing the files at home:

People who may not need to travel as much with their data may just want to store their files at home, however if they want to take files to work/other places I will try to make it as easy as possible to do so.

The user should be able to decrypt the files they need to a folder (possibly on a USB stick), or download files from the Vault to their mobile device. This is worse than just using the whole app on the USB stick as mentioned in the last section, as the files will not be encrypted once they are in the folder or downloaded onto the mobile device. It is not recommended to do this if you want to edit the files while away from home, unless you can edit it on your device, however if not you may as well just put the files onto a USB stick.

A data flow diagram for this use case would look something like this:



If you wanted to edit the files at work without putting the entire program on a USB, you could instead decrypt the file and put it on a USB, take it to work, edit the file, go home and then encrypt it back into the vault, however the file is not encrypted.

Technical Solution

All intensive AES and BLAKE are written in Go, while everything else is written in Python, however the sorts are Cythonized (Python that has been compiled to a C shared object, using a mix of static variables and dynamic variables). Python communicates with Go using stdin and stdout pipes. SHA is written in Python because it is only needed a couple of times during the program, and only ever has to hash small data, so it does not need to be as fast as possible.

File Structure:

Here is the file structure of the code:

```
1  code
2  |   mobile
3  |   |   btShared.py
4  |   |   buildozer.spec
5  |   |   fileSelectionScreen.py
6  |   |   main.py
7  |   |   mainScreen.py
8  |   |   pad.kv
9  |   |   padScreen.py
10  |   |   SHA.py
11  |   python-go
12  |   |   AES
13  |   |   |   AES
14  |   |   |   build.sh
15  |   |   |   main.go
16  |   |   |   src
17  |   |   |   |   AES
18  |   |   |   |   |   AEScheckKey
19  |   |   |   |   |   |   aesCheckKey.go
20  |   |   |   |   |   AESfiles
21  |   |   |   |   |   |   aesFiles.go
22  |   |   |   |   |   |   aesFiles_test.go
23  |   |   |   |   |   aes.go
24  |   |   |   |   |   AESstring
25  |   |   |   |   |   |   aesString.go
26  |   |   |   |   |   aes_test.go
27  |   |   |   |   sorts
28  |   |   |   |   |   sorts.go
29  |   |   |   |   testAES.sh
30  |   |   |   |   testBenchAES.sh
31  |   |   AESWin.exe
32  |   |   BLAKE
33  |   |   |   BLAKE
34  |   |   |   |   BLAKE.test
35  |   |   |   |   build.sh
36  |   |   |   |   main.go
37  |   |   |   |   src
38  |   |   |   |   |   BLAKE
39  |   |   |   |   |   |   blake.go
40  |   |   |   |   |   |   blake_test.go
41  |   |   |   |   |   |   checksum.go
42  |   |   |   |   testBenchBLAKE.sh
43  |   |   |   |   testBLAKE.sh
44  |   |   BLAKEWin.exe
45  |   |   config.cfg
46  |   |   configOperations.py
47  |   |   fileClass.py
48  |   |   KivyStuff
49  |   |   |   kvFiles
50  |   |   |   |   |   loginScBT.kv
51  |   |   |   |   |   loginSc.kv
```

```
52 |     └── mainScButtons.kv
53 |     └── mainSc.kv
54 |     └── mainScLabels.kv
55 |     └── mainScPops.kv
56 |     └── settingsSc.kv
57 |     └── loginClass.py
58 |     └── mainBtNSNew.py
59 |     └── mainBtNS.py
60 |     └── mainScClass.py
61 |     └── mainSmallPops.py
62 |     └── settingsScreen.py
63 |     └── ui.py
64 |     └── SHA.py
65 |     └── start.py
```

I have taken out all of the `__pycache__` folders that Python generates.

This is the output of `tree code` in my projects' `code` directory. You can find my project at <https://github.com/Lytchett-Minster/nea-12Colclough>.

The `code` directory, surprisingly, holds the code for my project. Inside is one folder for the mobile app (`mobile`), and one folder for the PC app (`python-go`). The PC app is started by running `start.py`. `start.py` imports `kivyStuff/ui.py` and runs it. This means that any Python files in `kivyStuff` can import any of the files that are in the same directory as `start.py` (`python-go`), and any Python files in `kivyStuff`. It also makes it easier to find the start script, as it isn't as buried.

The `assets` directory holds all the images needed for the GUI of the PC program (the images on the buttons). Here is a `tree` of the `assets` folder:

```
1 assets/
2   └── exports
3     ├── addFile.png
4     ├── backUpFolder.png
5     ├── folder.png
6     ├── home.png
7     ├── info.png
8     ├── padlock.png
9     ├── recycling.png
10    ├── refresh-icon.png
11    ├── remove file.png
12    ├── search.png
13    └── settings.png
14   └── psd
15     ├── add file.psd
16     ├── back up folder.psd
17     ├── folder.psd
18     ├── info.svg
19     ├── padlock.psd
20     └── remove file.psd
21
22 2 directories, 17 files
```

Some images are taken from the internet, so they do not have `.psd` files (photoshop files).

Configuration of the program:

The program can be configured via the configuration file `config.cfg` located at `code/python-go/config.cfg`, or instead if the user is on Linux, then they can copy the configuration file into `~/.config/FileMate/config`, which is the standard area in Linux where configuration files are stored (and by the way, I called the program "File Mate" because it was the first thing that popped into my head).

The configuration file is edited by the settings menu in the main screen of the app, however if something goes horribly wrong, the user can edit it themselves easily.

The layout of the configuration file looks something like this:

```
1 vaultDir--<file path here>
2 searchRecursively--<True / False>
3 bluetooth--<True / False>
```

`vaultDir` is the path to the Vault that you would like to use to store all encrypted files and folders. The file path can be done as an absolute path from the root of the file system (`/home/user/folder/`, or `C:\\\\Users\\\\user\\\\folder\\\\`), or can instead be done relatively from the folder that the program is in. For example, if the folder the code is stored in is `/home/josh/neal2ColcloughJ/code/python-go/`, then it would remove `neal2-ColcloughJ/code/python-go/` from the file path to get `/home/josh/`, then adds the file path specified in the config file. If the relative path specified was `folder/vault`, then the full path would become `/home/josh/folder/vault/`.

`searchRecursively` determines if the program should search for items recursively, as this may take a long time if you have a lot of files, and some people may just want to search within the current folder.

`bluetooth` determines the default Login Screen to start when the program starts.

I have used `--` to separate the setting name from its set value, as it does not appear at the start of file paths, and should not be needed much in any settings that could possibly be added in the future.

To change the configuration of the program from within the program, `configOperations.py` located at `code/python-go/configOperations.py` has a few functions that can get the configured settings, and write new ones.

Here is the content of `configOperations.py`:

```
1 from os import path as osPath
2 from os import listdir, makedirs
3 from sys import platform
4 from tempfile import gettempdir
5
6 def findConfigFile(startDir, fileSep):
7     config = None
8     if fileSep == "/":
9         try:
10             home = listdir(osPath.expanduser("~/./config/FileMate/"))
11         except:
12             print("No config file in .config")
13         else:
14             if "config" in home:
15                 config = osPath.expanduser("~/./config/FileMate/config")
16
17     if config == None:
18         try:
19             configFile = open(startDir+"config.cfg", "r")
20         except Exception as e:
21             raise FileNotFoundError("No config file found. Refer to the README if you need help.")
22         else:
23             configFile.close()
24             config = startDir+"config.cfg"
25
26     return config
27
28
29 def readConfigFile(configLocation=None, lineNumToRead=None, fSep=None, startDir=None):
30     if fSep == None:
31         fSep = getFileSep()
32     if configLocation == None:
```

```

33     fSep = getFileSep()
34     configLocation = findConfigFile(getStartDir(fSep)[0], fSep)
35
36     configFile = open(configLocation, "r")
37     if lineNumberToRead == None:
38         for line in configFile:
39             lineSplit = line.split("--")
40             lineSplit[1] = lineSplit[1].replace("\n", "")
41             if lineSplit[0] == "vaultDir":
42                 path = lineSplit[1]
43             elif lineSplit[0] == "searchRecursively":
44                 if lineSplit[1] == "True":
45                     recurse = True
46                 elif lineSplit[1] == "False":
47                     recurse = False
48                 else:
49                     raise ValueError("Recursive search settings not set correctly in config file: Not True or False.")
50             elif lineSplit[0] == "bluetooth":
51                 if lineSplit[1] == "True":
52                     bt = True
53                 elif lineSplit[1] == "False":
54                     bt = False
55                 else:
56                     raise ValueError("Bluetooth not configured correctly in config file: Not True or False.")
57
58     configFile.close()
59
60     if path[0] != fSep: # If vaultDir done relatively, then get path relative to the folder the program
61         is in, rather than searching the folder.
62         if startDir == None:
63             startDir = osPath.dirname(osPath.realpath(__file__))+fSep
64             startDir = startDir.split(fSep)
65             path = fSep.join(startDir[:-4])+fSep+path # Removes "" and nea-12ColcloughJ/code/python-go folder
66             names from list, then adds the Vault folder name to the end.
67             if path[-1] != fSep:
68                 path += fSep # End with file separator
69
70     return path, recurse, bt
71
72     else:
73         lineSplit = configFile.readlines()[lineNumberToRead].split("--")
74         lineSplit[1] = lineSplit[1].replace("\n", "")
75     return lineSplit[1]
76
77 def getFileSep():
78     if platform.startswith("win32"): # Find out what operating system is running.
79         return "\\"
80     else:
81         return "/"
82
83 def getStartDir(fileSep=None):
84     if fileSep == None:
85         fileSep = getFileSep()
86     startDir = osPath.dirname(osPath.realpath(__file__))+fileSep
87     tempDir = startDir.split(fileSep)
88     for i in range(2):
89         del tempDir[-2]
90     return startDir, fileSep.join(tempDir)+fileSep+"assets"+fileSep+"exports"+fileSep
91
92 def editConfTerm(term, newContent, config): # Edits a given term in the config.cfg file.
93     with open(config, "r") as conf:
94         confContent = conf.readlines()
95
96     for i in range(len(confContent)):
97         a = confContent[i].split("--")
98         if term == a[0]:
              a[1] = newContent+"\n"

```

```

99         confContent[i] = "--".join(a)
100
101     with open(config, "w") as confW:
102         confW.writelines(confContent)
103
104 def dirInputValid(inp, fileSep):
105     valid = bool((inp[0] == fileSep) and ("\n" not in inp))           #If it starts with the file separator and
106     doesn't contain any new lines, then it is valid for now.
107     inp = inp.split(fileSep)
108     focusIsSlash = False
109     for item in inp:          #Checks for multiple file separators next to each other, as that would be an
110         if item == "":
111             if focusIsSlash:
112                 valid = False
113             focusIsSlash = True
114         else:
115             focusIsSlash = False
116     return valid
117
118 def changeVaultLoc(inp, fileSep, config):      #Sorts out the UI while the vault location is changed.
119     if inp != "":
120         if dirInputValid(inp, fileSep):
121             if osPath.exists(inp) and osPath.isdir(inp):
122                 editConfTerm("vaultDir", inp, config)
123             else:
124                 makedirs(inp)
125                 if inp[-1] != fileSep:
126                     inp += fileSep
127                 editConfTerm("vaultDir", inp, config)
128
129         return True
130
131
132
133 def runConfigOperations():
134     fileSep = getFileSep()
135     osTemp = gettempdir() + fileSep #From tempfile module
136     # Get config settings.
137     startDir, sharedAssets = getStartDir(fileSep)
138
139     configLoc = findConfigFile(startDir, fileSep)
140     path, recurse, bt = readConfigFile(configLoc, fSep=fileSep, startDir=startDir)
141     return fileSep, osTemp, startDir, sharedAssets, path, recurse, bt, configLoc # 8 Outputs in total.

```

`findConfigFile` checks for the configuration file in `~/.config/FileMate/`, and if it does not exist, checks for it in `code/python-go/config.cfg`. Once the configuration file has been found, it returns the path to the file.

`readConfigFile` reads the configuration file, and gets each configured option and returns their value. It can also return the value of a specific line in the config file.

`getFileSep` just gets the file separator of the current system. For Windows this is `\`, but for MacOS and Linux this is `/`.

`getStartDir` gets the path of the current file (located in `code/python-go/`), and the path to the `assets` directory, which is used for the images on the buttons.

`editConfTerm` edits a term in the configuration file. If the term was "bluetooth" then it would find the line that starts with "bluetooth", and change the data after the `:` with the new data specified.

`dirInputValid` checks that a given input is a valid file path (e.g no "/" in a row). This is in here because it is used all over the program, and is used for changing the directory of the Vault.

`changeVaultLoc` changes the location of the Vault using `dirInputValid` to check the input, and `editConfTerm` to update the configuration file.

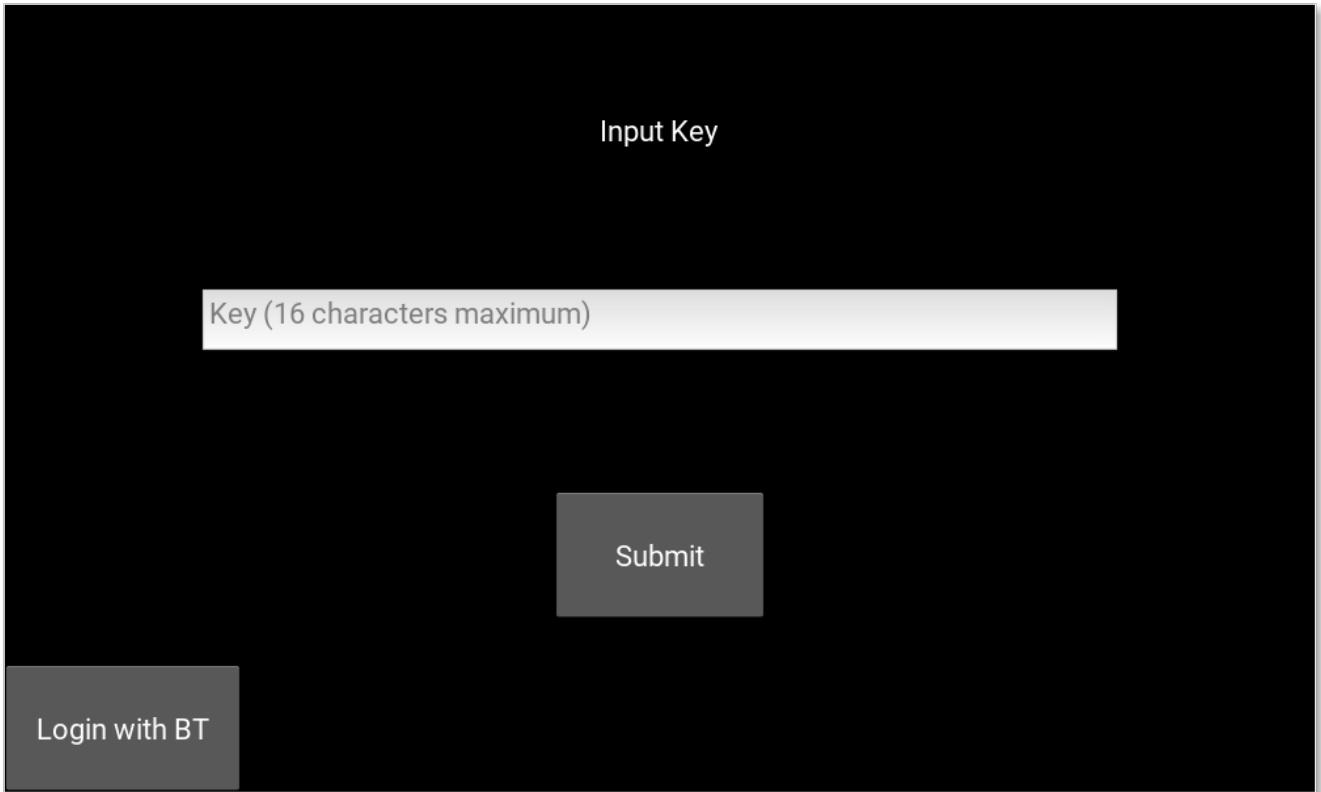
`runConfigOperations` runs all of the operations required for when the program is started, and returns the variables needed by the rest of the program. This is done in `ui.py`, which loads the configuration file, and starts the program.

The PC App GUI:

I will go through the visuals first, and then move onto the code.

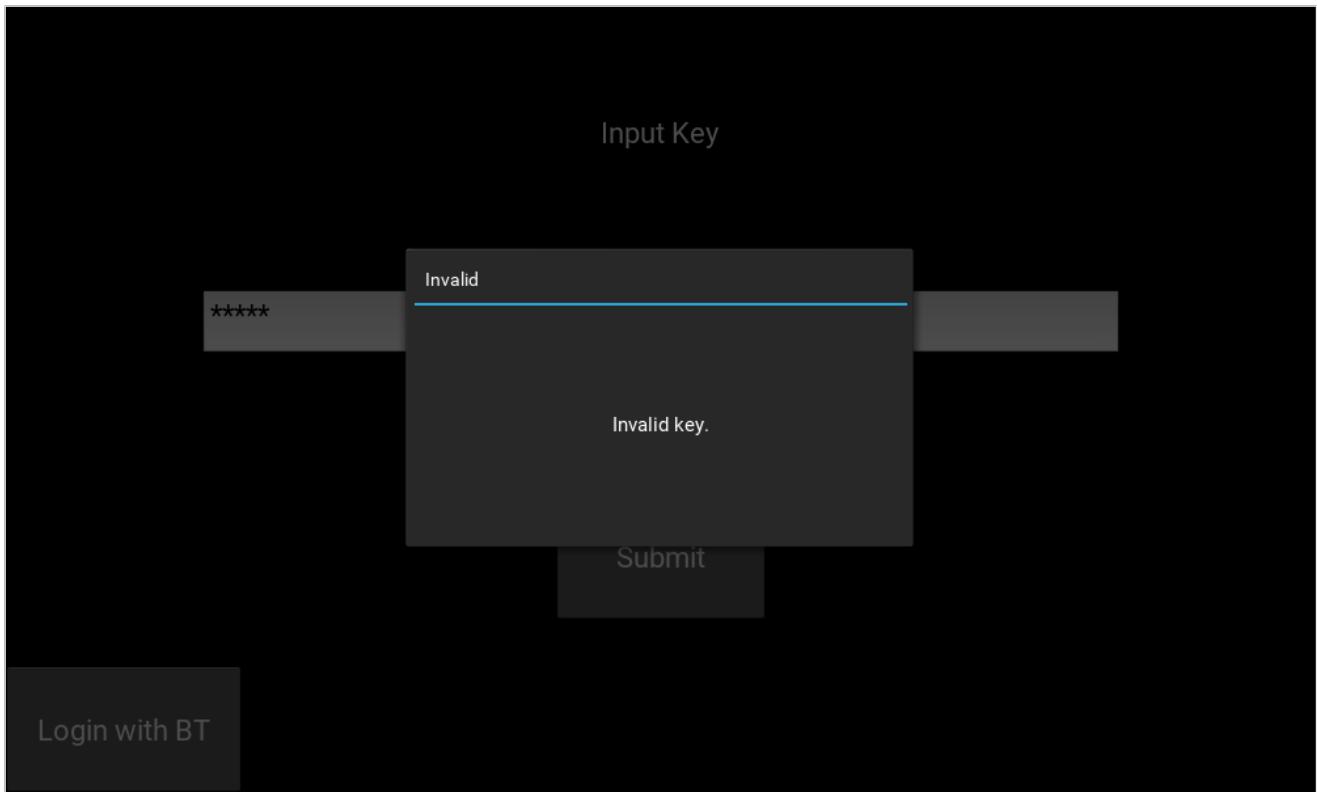
Login Screen

Here is an image of the Login Screen:

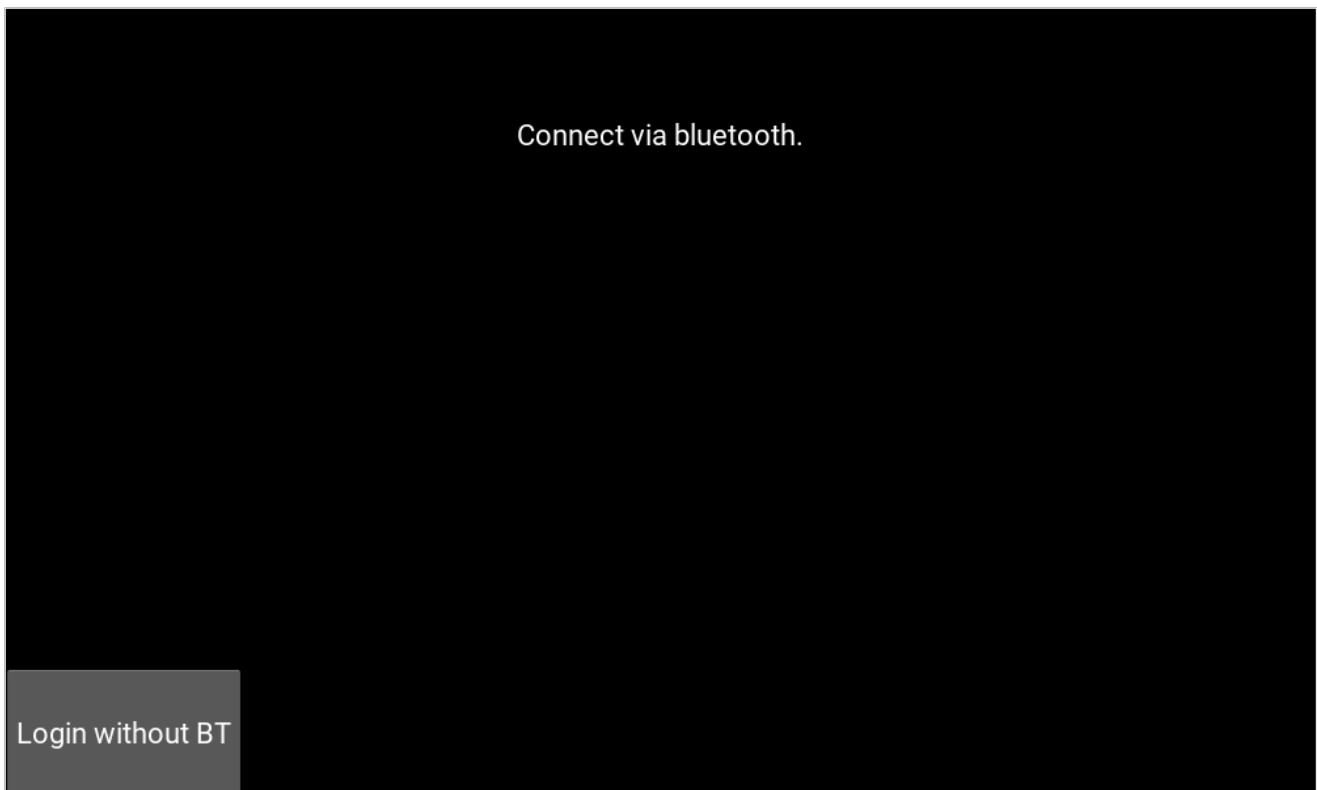


It consists of a key entry text input, a "Submit" button and a button to switch between logging in with Bluetooth and without Bluetooth.

When you enter an incorrect key, a popup tells the user the key is invalid:

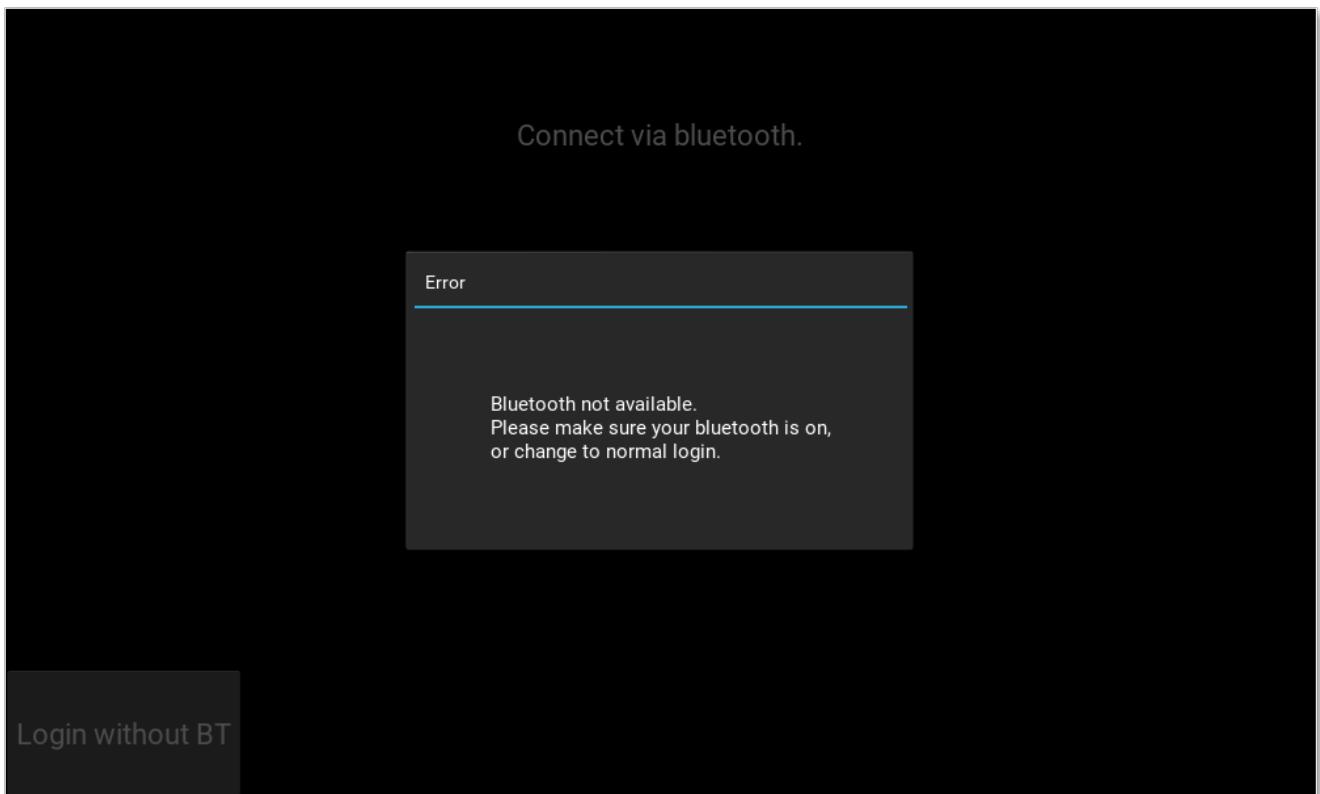


The Bluetooth login screen can be accessed by clicking the "Login with BT" button, changing the configuration file, or changing the settings once logged in. Here is an image of the Login Screen with Bluetooth:



I have tried to keep it as simple and as clutter-free as possible. When a user connects to the BT server, the address of the device connected appears in the middle of the screen, to let the user know that they have connected. The user then proceeds to enter the pin code on the app.

If Bluetooth is not available, or can't start, then a popup appears warning the user that they cannot use the Bluetooth login until Bluetooth becomes available, or they can instead login with regular login:

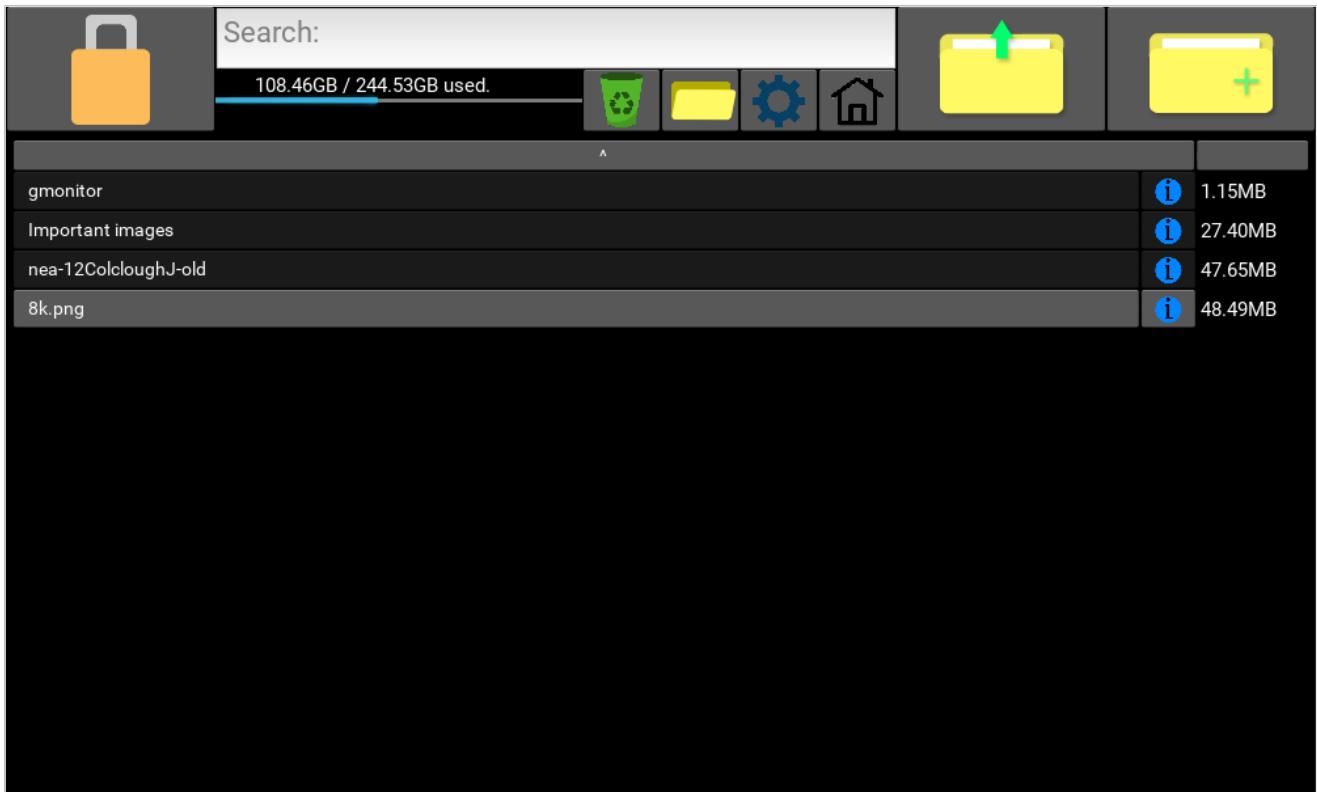


They can then click away from the popup to dismiss it, and do what they want from there.

Once the key has been entered correctly, the screen changes to the Main Screen.

Main Screen

Here is an image of the Main Screen:



I tried to keep it as similar as possible to the design in the **Design** section, however I had to add a button to add a new folder, as I realised that was a necessity. Also, instead of writing "Information" on the information button, I made an image to be put over the button instead, as it looks a bit better. All the buttons in the GUI are darker than in the design, but that is fine.

It is easy to distinguish between files and folders, and doesn't feel cluttered. The progress bar showing the amount of space used on the current storage device, in my opinion looks better than in the design. It is easy to sort by name (click the button above all the files) or to sort by size (click the button above all of the sizes).

When you click a folder, you change directory to that folder, and the contents of that folder are displayed on the screen. If it is a file, it is decrypted to `<systems_temp_folder>/FileMate/<fileName>`, where it is then opened with the system's default application and can be renamed and edited.

When you click to add a new folder, you get the exact same popup as in the **Information Tab** when you decrypt an item to a location.

The Information Tab

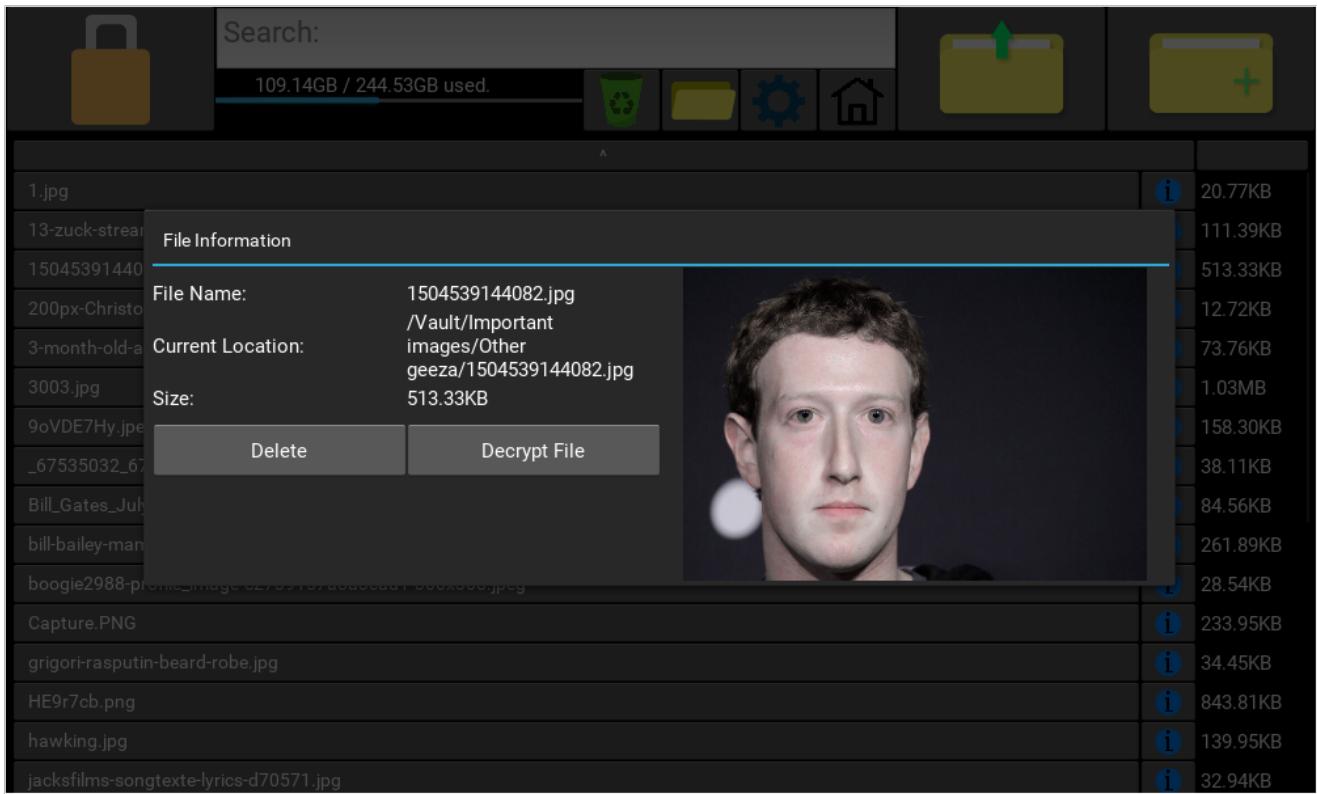
The information tab shows you information about the file:

- The location the file/folder is relative to the Vault.
- The size of the file/folder.
- A thumbnail of the image, if it is a file not a folder, and if the file is an image.

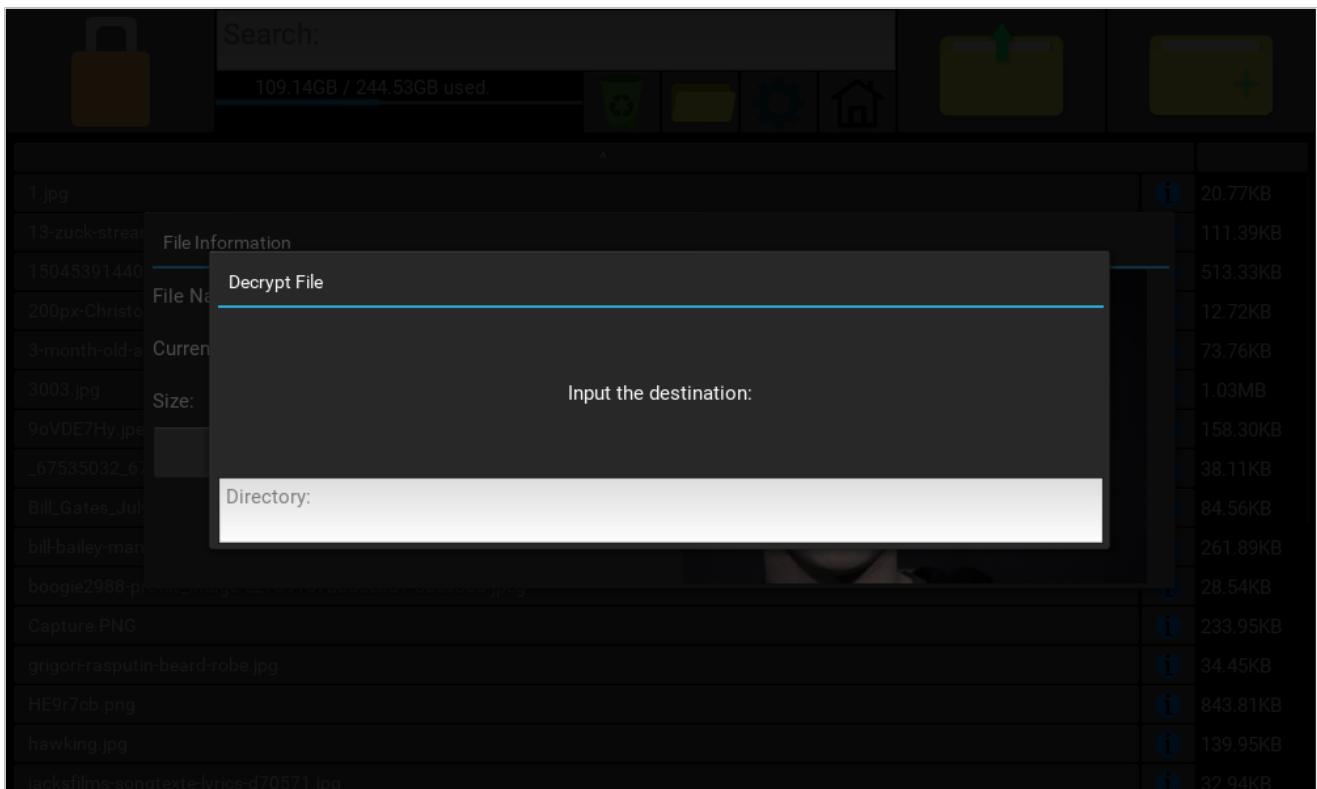
You also have a few options within the information tab to chose from:

- Delete the file/folder.
- Decrypt the file/folder to a specified location.

Here is a screenshot of the information popup:



When you click decrypt file, you are greeted with another popup asking where you would like the file to go:



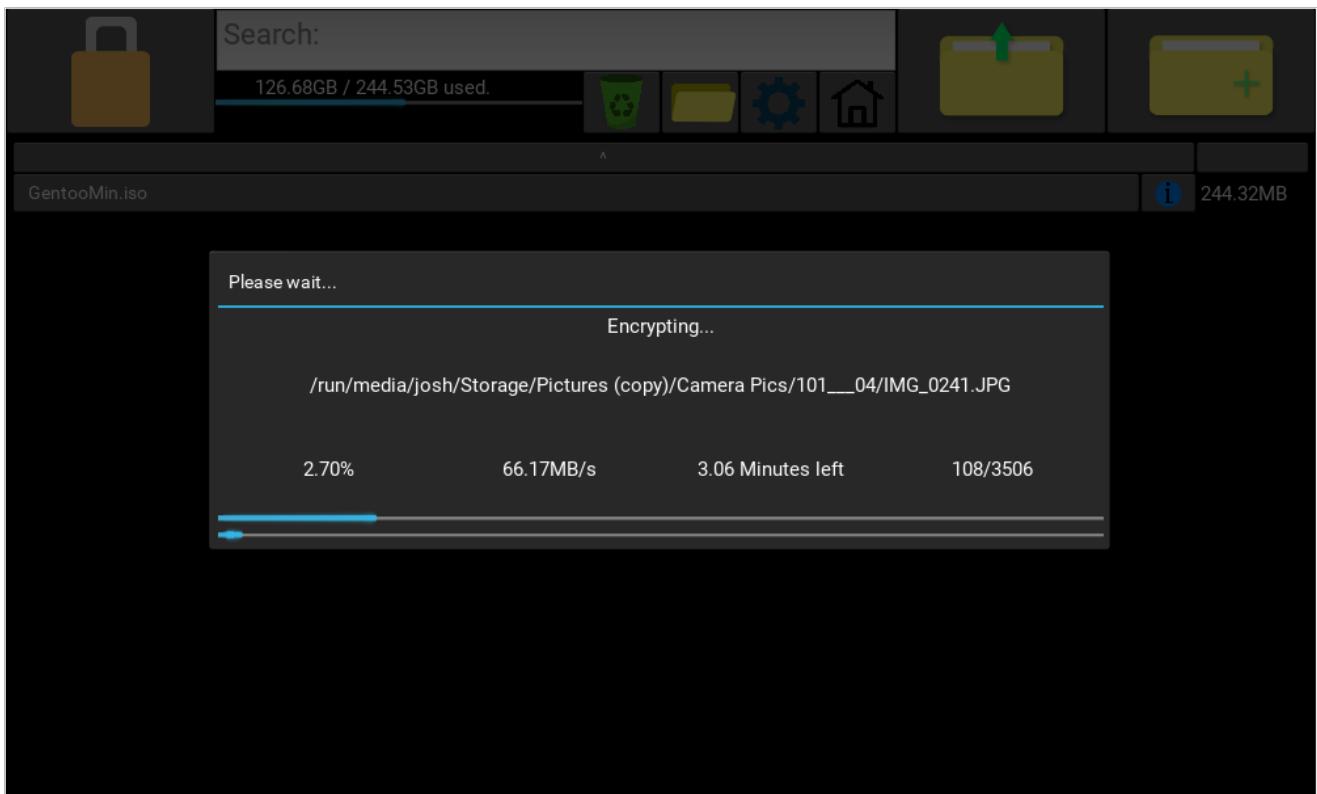
Once you input a correct directory name, it decrypts the file to that directory. If the path ends with the file separator (e.g. "/"), then it will be decrypted to that folder with its original name. If the path does not, then it is saved to that exact location, with that new name. For example, if I wanted to decrypt a file called `Zuckerburg.png`, then I put in `/home/josh/zucc.png`, then it would decrypt the file and would be saved as `zucc.png`. If I instead put in `/home/josh/`, then it would be saved to `/home/josh/Zuckerburg.png`.

When you delete the file, the file moves to the recycling folder located in `Vault/.recycling` (relative to the vault). To recover the file, you click the recycling bin button, and you get put in the recycling folder (with a popup warning the user they are in the recycling bin). Now when you click items in the recycling folder, instead of opening them and decrypting it, it moves the file back into the vault. You can still view information about the file like usual, and search for items. To leave the recycling bin, you click the folder up button on the top right, or the home key below the search bar.

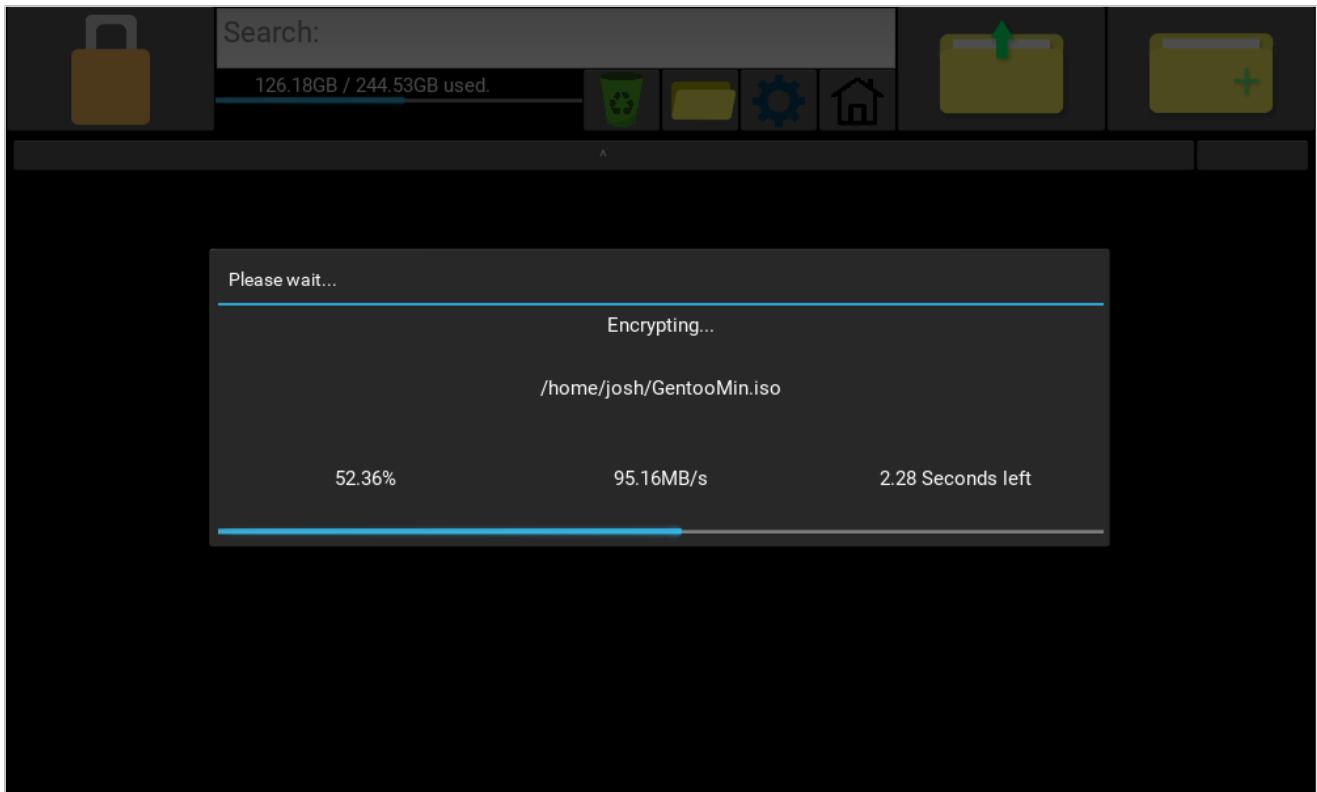
Encryption and Decryption Status

When decrypting a file, a popup opens showing you the percentage of the way through you are in the file and the current speed of decryption. When decrypting a folder, the same information is shown, however there are two progress bars. One for the current file being decrypted, and one for the total progress of decrypting the folder. Also, you are shown how many files in the folder have been encrypted out of the total. This is the exact same for encryption too by the way.

Here is an image of a folder being encrypted (12GB):



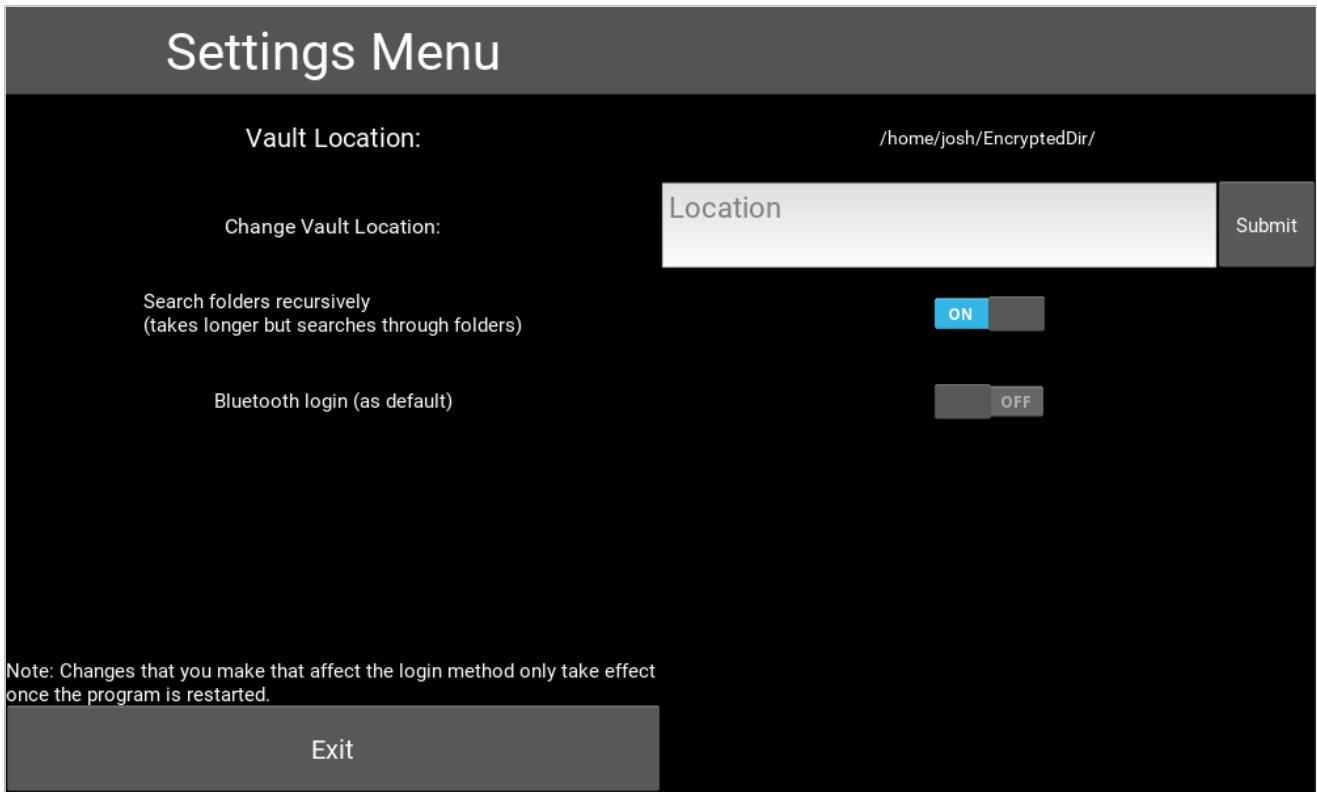
Here is an image of a file being encrypted (244 MB):



As I have said before, when a file is decrypted, it is decrypted to `<systems_temp_folder>/FileMate/<fileName>`, and is then opened using the system's default program for that file type. A checksum is calculated before the file is opened, and after the file is closed using BLAKE2b. The checksum is then compared, and if the checksum is different, then the file is encrypted back into the vault.

Settings Screen

Here is an image of the settings screen:



The current Vault location is displayed at the top of the screen, followed by an input to change the Vault location, followed by a pair of switches to change whether the search is recursive, and which login screen to use as default. When done, the user can click "Exit" to exit to the main screen again.

The Search

The search does a linear search through the unsorted directory, checking if the search term is in the file name, and at what position the item appears in the file name. This data is appended in tuples like this: `(pos, fileName)`. These tuples are then sorted by their `pos` value using a quick sort, and if the search term matched a file in the list exactly, letter for letter, then it will be added to the start of the list.

The list of results is then displayed:

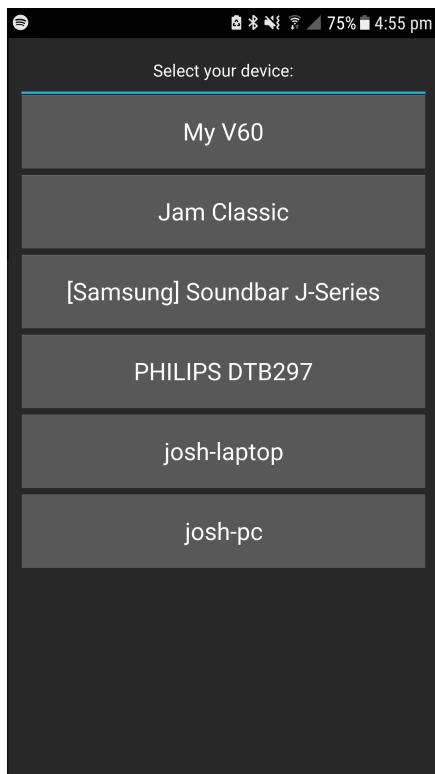


If no results are found, a popup opens saying "No results found for: search item".

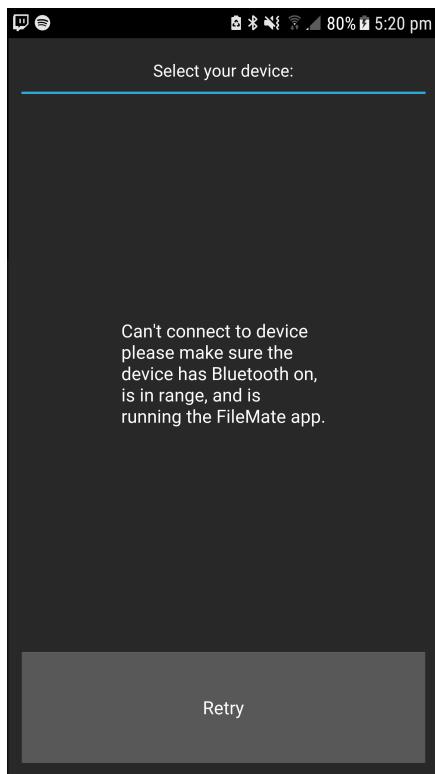
The Mobile App GUI

The mobile app has a basic design, as I wanted to keep it as small as possible, and images were not really needed.

When you first open the app, you are greeted by a scrollable list of devices to connect to (if Bluetooth is on):

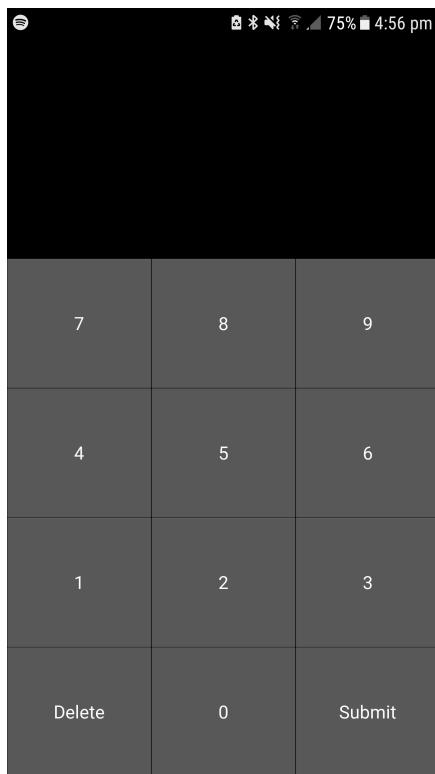


If you can't connect to a device, then you are greeted by this large popup:

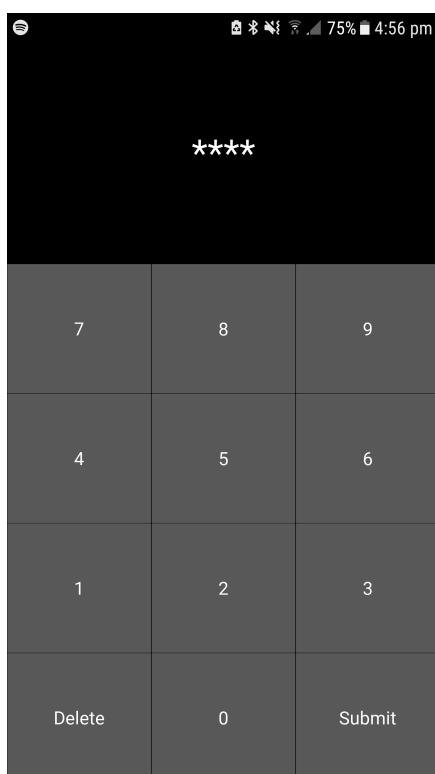


If you click retry, it takes you back to the start screen.

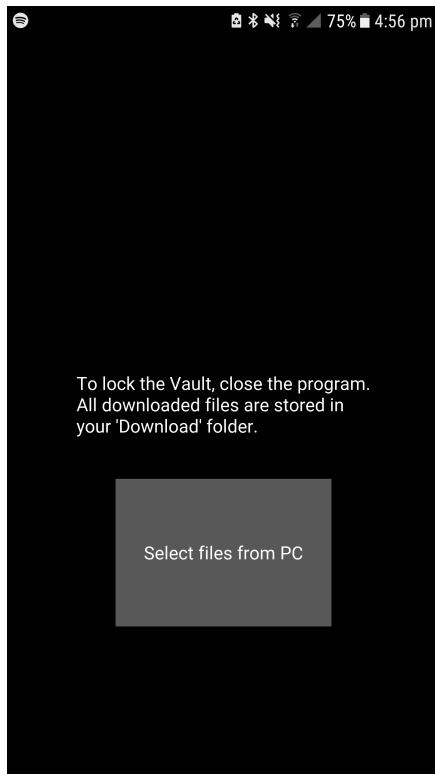
If you connect successfully, you are greeted by this screen:



Where you can then enter your key and submit it. While you are entering it, it looks like this:

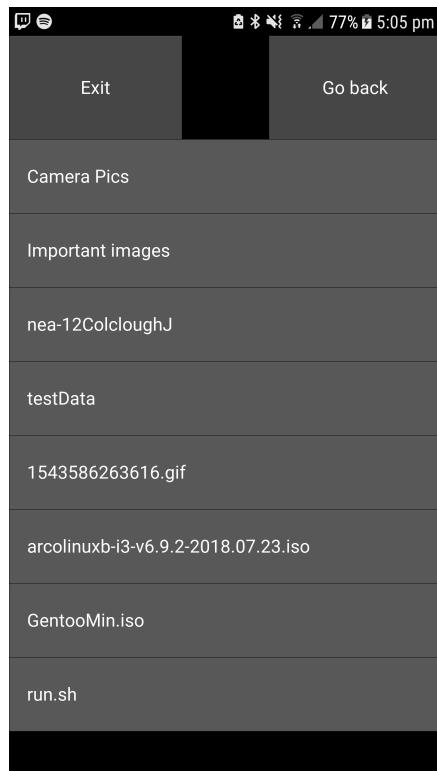


Once you press submit, a popup opens telling you if the key was correct or not, and then you are put into this screen:



From here you can either close the app and leave it running in the background, or browse files. If you close the app completely, the app on the PC locks.

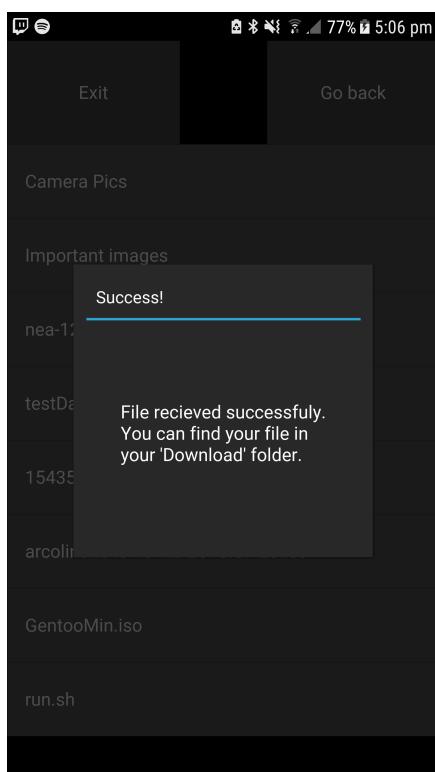
Here is what the app looks like while you are browsing files:



Here you can navigate folders and download files. Here is an image of a navigated folder:



If you want to download a file, a popup opens on the PC app showing the current status of the file using `btTransferPop` ON `MainScreen`, and once the download has finished, a popup opens saying that it has finished, and where you can find it:



Now you can find that file in your devices "Download" folder.

Here is a video of the whole process (type this into your web browser's address bar): <https://youtu.be/FNt5Vsa f4UM>

Key Algorithms

In this section I will explain each algorithm if the comments in the code are not sufficient, and point out any of the bits that have changed or are different in the **Design** section.

AES:

Go packages are like modules in Python, but are created by grouping Go files of the same package in a folder named the same as the package. Packages have to be stored in a source folder, called `src`, and outside of the `src` folder, a main package (usually stored in `main.go`) handles all of the other packages, and the Go project is built from there.

AES is split into several Go packages. One called AES, which holds all the code needed for the core of AES (encryption and decryption of a single block), AESfiles which handles enc/decryption of files, AESstring which handles with enc/decrypting strings (such as file names) and listing directories containing encrypted files, and finally AEScheckKey which handles key checking. They are all tied together with the `main.go` file in `code/python-go/AES/main.go`, which handles input from Python.

Just to recap, the file structure of the AES folder looks like this:

```
1  code/python-go/AES/
2      └── AES
3          ├── build.sh
4          ├── main.go
5          └── src
6              ├── AES
7                  ├── AEScheckKey
8                      └── aesCheckKey.go
9                  ├── AESfiles
10                 └── aesFiles.go
11                 └── aesFiles_test.go
12                 └── aes.go
13                 └── AESstring
14                     └── aesString.go
15                     └── aes_test.go
16                     └── sorts
17                         └── sorts.go
18                 └── testAES.sh
19                 └── testBenchAES.sh
20
21  6 directories, 12 files
```

You will notice a few `.sh` files, these are just used to run tests on the code, and also build the code. I will go a bit more into the testing in the **Testing** section.

In the `code/python-go/AES/src/` folder, there is also a folder called `sorts`, which holds the sorts that are used by `AESstring` to sort lists of files. The `sorts` package is used to sort lists of files.

AES package

Here is the `AES` package in `code/python-go/AES/src/AES/aes.go`:

```
1  package AES
2
3  // Global lookup tables.
4  var sBox = [256]byte {0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
5          0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
6          0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
```

```

7   0x04,0xC7,0x23,0xC3,0x18,0x96,0x05,0x9A,0x07,0x12,0x80,0xE2,0x27,0xB2,0x75,
8   0x09,0x83,0x2C,0x1A,0x1B,0x6E,0x5A,0xA0,0x52,0x3B,0xD6,0xB3,0x29,0xE3,0x2F,0x84,
9   0x53,0xD1,0x00,0xED,0x20,0xFC,0xB1,0x5B,0x6A,0xCB,0xBE,0x39,0x4A,0x4C,0x58,0xCF,
10  0xD0,0xEF,0xAA,0xFB,0x43,0x4D,0x33,0x85,0x45,0xF9,0x02,0x7F,0x50,0x3C,0x9F,0xA8,
11  0x51,0xA3,0x40,0x8F,0x92,0x9D,0x38,0xF5,0xBC,0xB6,0xDA,0x21,0x10,0xFF,0xF3,0xD2,
12  0xCD,0x0C,0x13,0xEC,0x5F,0x97,0x44,0x17,0xC4,0xA7,0x7E,0x3D,0x64,0x5D,0x19,0x73,
13  0x60,0x81,0x4F,0xDC,0x22,0x2A,0x90,0x88,0x46,0xEE,0xB8,0x14,0xDE,0x5E,0x0B,0xDB,
14  0xE0,0x32,0x3A,0xA,0x49,0x06,0x24,0x5C,0xC2,0xD3,0xAC,0x62,0x91,0x95,0xE4,0x79,
15  0xE7,0xC8,0x37,0x6D,0x8D,0x5D,0x4E,0xA9,0x6C,0x56,0xF4,0xEA,0x65,0x7A,0xAE,0x08,
16  0xBA,0x78,0x25,0x2E,0x1C,0xA6,0xB4,0xC6,0xE8,0xDD,0x74,0x1F,0x4B,0xBD,0x8B,0x8A,
17  0x70,0x3E,0xB5,0x66,0x48,0x03,0xF6,0x0E,0x61,0x35,0x57,0xB9,0x86,0xC1,0x1D,0x9E,
18  0xE1,0xF8,0x98,0x11,0x69,0xD9,0x8E,0x94,0x9B,0x1E,0x87,0xE9,0xCE,0x55,0x28,0xDF,
19  0x8C,0xA1,0x89,0x0D,0xBF,0xE6,0x42,0x68,0x41,0x99,0x2D,0x0F,0xB0,0x54,0xBB,0x16}
20

21 var invSBox = [256]byte {0x52,0x09,0x6A,0xD5,0x30,0x36,0xA5,0x38,0xBF,0x40,0xA3,0x9E,0x81,0xF3,0xD7,0xFB,
22   0x7C,0xE3,0x39,0x82,0x9B,0x2F,0xFF,0x87,0x34,0x8E,0x43,0x44,0xC4,0xDE,0xE9,0xCB,
23   0x54,0x7B,0x94,0x32,0xA6,0xC2,0x23,0x3D,0xEE,0x4C,0x95,0x0B,0x42,0xFA,0xC3,0x4E,
24   0x08,0x2E,0xA1,0x66,0x28,0xD9,0x24,0xB2,0x76,0x5B,0xA2,0x49,0x6D,0xBB,0xD1,0x25,
25   0x72,0xF8,0xF6,0x64,0x86,0x68,0x98,0x16,0xD4,0xA4,0x5C,0xCC,0x5D,0x65,0xB6,0x92,
26   0x6C,0x70,0x48,0x50,0xFD,0xED,0xB9,0xDA,0x5E,0x15,0x46,0x57,0xA7,0x8D,0x9D,0x84,
27   0x90,0xD8,0xAB,0x00,0x8C,0xBC,0xD3,0xA,0xF7,0xE4,0x58,0x05,0xB8,0xB3,0x45,0x06,
28   0xD0,0x2C,0x1E,0x8F,0xCA,0x3F,0x0F,0x02,0xC1,0xAF,0xBD,0x03,0x01,0x13,0x8A,0x6B,
29   0x3A,0x91,0x11,0x41,0x4F,0x67,0xDC,0xEA,0x97,0xF2,0xCF,0xCE,0xF0,0xB4,0xE6,0x73,
30   0x96,0xAC,0x74,0x22,0xE7,0xAD,0x35,0x85,0xE2,0xF9,0x37,0xE8,0x1C,0x75,0xDF,0x6E,
31   0x47,0xF1,0x1A,0x71,0x1D,0x29,0xC5,0x89,0x6F,0xB7,0x62,0xE,0xAA,0x18,0xBE,0x1B,
32   0xFC,0x56,0x3E,0x4B,0xC6,0xD2,0x79,0x20,0x9A,0xDB,0xC0,0xFE,0x78,0xCD,0x5A,0xF4,
33   0x1F,0xDD,0xA8,0x33,0x88,0x07,0xC7,0x31,0xB1,0x12,0x10,0x59,0x27,0x80,0xEC,0x5F,
34   0x60,0x51,0x7F,0xA9,0x19,0xB5,0x4A,0x0D,0x2D,0xE5,0x7A,0x9F,0x93,0xC9,0x9C,0xEF,
35   0xA0,0xE0,0x3B,0x4D,0xAE,0x2A,0xF5,0xB0,0xC8,0xEB,0xBB,0x3C,0x83,0x53,0x99,0x61,
36   0x17,0x2B,0x04,0x7E,0xBA,0x77,0xD6,0x26,0xE1,0x69,0x14,0x63,0x55,0x21,0x0C,0x7D}
37

38 var rcon = [256]byte {0x8d,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1b,0x36,0x6c,0xd8,0xab,0x4d,0x9a, // 
39   https://en.wikipedia.org/wiki/Rijndael_key_schedule
40   0x2f,0x5e,0xbc,0x63,0xc6,0x97,0x35,0x6a,0xd4,0xb3,0x7d,0xfa,0xef,0xc5,0x91,0x39,
41   0x72,0xe4,0xd3,0xbd,0x61,0xc2,0xf,0x25,0x4a,0x94,0x33,0x66,0xcc,0x83,0x1d,0x3a,
42   0x74,0xe8,0xcb,0x8d,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1b,0x36,0x6c,0xd8,
43   0xab,0x4d,0x9a,0x2f,0x5e,0xbc,0x63,0xc6,0x97,0x35,0x6a,0xd4,0xb3,0x7d,0xfa,0xef,
44   0xc5,0x91,0x39,0x72,0xe4,0xd3,0xbd,0x61,0xc2,0x9f,0x25,0x4a,0x94,0x33,0x66,0xcc,
45   0x83,0x1d,0x3a,0x74,0xe8,0xcb,0x8d,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1b,
46   0x36,0x6c,0xd8,0xab,0x4d,0x9a,0x2f,0x5e,0xbc,0x63,0xc6,0x97,0x35,0x6a,0xd4,0xb3,
47   0x7d,0xfa,0xef,0xc5,0x91,0x39,0x72,0xe4,0xd3,0xbd,0x61,0xc2,0x9f,0x25,0x4a,0x94,
48   0x33,0x66,0xcc,0x83,0x1d,0x3a,0x74,0xe8,0xcb,0x8d,0x01,0x02,0x04,0x08,0x10,0x20,
49   0x40,0x80,0x1b,0x36,0x6c,0xd8,0xab,0x4d,0x9a,0x2f,0x5e,0xbc,0x63,0xc6,0x97,0x35,
50   0x6a,0xd4,0xb3,0x7d,0xfa,0xef,0xc5,0x91,0x39,0x72,0xe4,0xd3,0xbd,0x61,0xc2,0x9f,
51   0x25,0x4a,0x94,0x33,0x66,0xcc,0x83,0x1d,0x3a,0x74,0xe8,0xcb,0x8d,0x01,0x02,0x04,
52   0x08,0x10,0x20,0x40,0x80,0x1b,0x36,0x6c,0xd8,0xab,0x4d,0x9a,0x2f,0x5e,0xbc,0x63,
53   0xc6,0x97,0x35,0x6a,0xd4,0xb3,0x7d,0xfa,0xef,0xc5,0x91,0x39,0x72,0xe4,0xd3,0xbd,
54   0x61,0xc2,0x9f,0x25,0x4a,0x94,0x33,0x66,0xcc,0x83,0x1d,0x3a,0x74,0xe8,0xcb,0x8d}

55 var mul2 = [256]byte {0x00,0x02,0x04,0x06,0x08,0xA,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e,
56   0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e,
57   0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e,
58   0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e,
59   0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e,
60   0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
61   0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
62   0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
63   0x1b,0x19,0x1f,0x1d,0x13,0x11,0x17,0x15,0x0b,0x09,0x0f,0x0d,0x03,0x01,0x07,0x05,
64   0x3b,0x39,0x3f,0x3d,0x33,0x31,0x37,0x35,0x2b,0x29,0x2f,0x2d,0x23,0x21,0x27,0x25,
65   0x5b,0x59,0x5f,0x5d,0x53,0x51,0x57,0x55,0x4b,0x49,0x4f,0x4d,0x43,0x41,0x47,0x45,
66   0x7b,0x79,0x7f,0x7d,0x73,0x71,0x77,0x75,0x6b,0x69,0x6f,0x6d,0x63,0x61,0x67,0x65,
67   0x9b,0x99,0x9f,0x9d,0x93,0x91,0x97,0x95,0x8b,0x89,0x8f,0x8d,0x83,0x81,0x87,0x85,
68   0xbb,0xb9,0xbf,0xbd,0xb3,0xb1,0xb7,0xb5,0xab,0xa9,0xaf,0xad,0xa3,0xa1,0xa7,0xa5,
69   0xdb,0xd9,0xdf,0xdd,0xd3,0xd1,0xd7,0xd5,0xcb,0xc9,0xcf,0xcd,0xc3,0xc1,0xc7,0xc5,
70   0xfb,0xf9,0xff,0xfd,0xf3,0xf1,0xf7,0xf5,0xeb,0xe9,0xef,0xed,0xe3,0xe1,0xe7,0xe5}

71

72 var mul3 = [256]byte {0x00,0x03,0x06,0x05,0x0c,0x0f,0xA,0x09,0x18,0x1b,0x1e,0x1d,0x14,0x17,0x12,0x11,
73   0x30,0x33,0x36,0x35,0x3c,0x3f,0x3a,0x39,0x28,0x2b,0x2e,0x2d,0x24,0x27,0x22,0x21,
74   0x60,0x63,0x66,0x65,0x6c,0x6f,0x6a,0x69,0x78,0x7b,0x7e,0x7d,0x77,0x72,0x71,
```

```

75    0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
76    0xc0,0xc3,0xc6,0xc5,0xcc,0xcf,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd7,0xd2,0xd1,
77    0xf0,0xf3,0xf6,0xf5,0xfc,0xff,0xfa,0xf9,0xe8,0xeb,0xee,0xed,0xe4,0xe7,0xe2,0xe1,
78    0xa0,0xa3,0xa6,0xa5,0xac,0xaf,0xaa,0xa9,0xb8,0xbb,0xbe,0xbd,0xb4,0xb7,0xb2,0xb1,
79    0x90,0x93,0x96,0x95,0x9c,0x9f,0x9a,0x99,0x88,0x8b,0x8e,0x8d,0x84,0x87,0x82,0x81,
80    0x9b,0x98,0x9d,0x9e,0x97,0x94,0x91,0x92,0x83,0x80,0x85,0x86,0x8f,0x8c,0x89,0x8a,
81    0xab,0xa8,0xad,0xae,0xa7,0xa4,0xa1,0xa2,0xb3,0xb0,0xb5,0xb6,0xbf,0xbc,0xb9,0xba,
82    0xfb,0xf8,0xfd,0xfe,0xf7,0xf4,0xf1,0xf2,0xe3,0xe0,0xe5,0xe6,0xef,0xec,0xe9,0xea,
83    0xcb,0xc8,0xcd,0xce,0xc7,0xc4,0xc1,0xc2,0xd3,0xd0,0xd5,0xd6,0xdf,0xdc,0xd9,0xda,
84    0xb5,0x58,0x5d,0x5e,0x57,0x54,0x51,0x52,0x43,0x40,0x45,0x46,0x4f,0x4c,0x49,0x4a,
85    0x6b,0x68,0x6d,0x6e,0x67,0x64,0x61,0x62,0x73,0x70,0x75,0x76,0x7f,0x7c,0x79,0x7a,
86    0x3b,0x38,0x3d,0x3e,0x37,0x34,0x31,0x32,0x23,0x20,0x25,0x26,0x2f,0x2c,0x29,0x2a,
87    0x0b,0x08,0x0d,0x0e,0x07,0x04,0x01,0x02,0x13,0x10,0x15,0x16,0x1f,0x1c,0x19,0x1a}
88
89 var mul9 = [256]byte {0x00,0x09,0x12,0x1b,0x24,0x2d,0x36,0x3f,0x48,0x41,0x5a,0x53,0x6c,0x65,0x7e,0x77,
90 0x90,0x99,0x82,0x8b,0xb4,0xbd,0xa6,0xaf,0xd8,0xd1,0xca,0xc3,0xfc,0xf5,0xee,0xe7,
91 0x3b,0x32,0x29,0x20,0x1f,0x16,0x0d,0x04,0x73,0x7a,0x61,0x68,0x57,0x5e,0x45,0x4c,
92 0xab,0xa2,0xb9,0xb0,0x8f,0x86,0x9d,0x94,0xe3,0xea,0xf1,0xf8,0xc7,0xce,0xd5,0xdc,
93 0x76,0x7f,0x64,0x6d,0x52,0x5b,0x40,0x49,0x3e,0x37,0x2c,0x25,0x1a,0x13,0x08,0x01,
94 0xe6,0xef,0xf4,0xfd,0xc2,0xcb,0xd0,0xd9,0xae,0xa7,0xbc,0xb5,0x8a,0x83,0x98,0x91,
95 0x4d,0x44,0x5f,0x56,0x69,0x60,0x7b,0x72,0x05,0x0c,0x17,0x1e,0x21,0x28,0x33,0x3a,
96 0xdd,0xd4,0xcf,0xc6,0xf9,0xeb,0xe2,0x95,0x9c,0x87,0x8e,0xb1,0xb8,0xa3,0xaa,
97 0xec,0xe5,0xfe,0xf7,0xc8,0xc1,0xda,0xd3,0xa4,0xad,0xb6,0xbf,0x80,0x89,0x92,0x9b,
98 0x7c,0x75,0x6e,0x67,0x58,0x51,0x4a,0x34,0x3d,0x26,0x2f,0x10,0x19,0x02,0x0b,
99 0xd7,0xde,0xc5,0xcc,0xf3,0xfa,0xe1,0xe8,0x9f,0x96,0x8d,0x84,0xbb,0xb2,0xa9,0xa0,
100 0x47,0x4e,0x55,0x5c,0x63,0x6a,0x71,0x78,0x0f,0x06,0x1d,0x14,0x2b,0x22,0x39,0x30,
101 0x9a,0x93,0x88,0x81,0xbe,0xb7,0xac,0xa5,0xd2,0xdb,0xc0,0xc9,0xf6,0xff,0xe4,0xed,
102 0x0a,0x03,0x18,0x11,0x2e,0x27,0x3c,0x35,0x42,0x4b,0x50,0x59,0x66,0x6f,0x74,0x7d,
103 0xa1,0xa8,0xb3,0xba,0x85,0x8c,0x97,0x9e,0xe9,0xe0,0xfb,0xf2,0xcd,0xc4,0xdf,0xd6,
104 0x31,0x38,0x23,0x2a,0x15,0x1c,0x07,0x0e,0x79,0x70,0x6b,0x62,0x5d,0x54,0x4f,0x46}
105
106 var mul11 = [256]byte {0x00,0x0b,0x16,0x1d,0x2c,0x27,0x3a,0x31,0x58,0x53,0x4e,0x45,0x74,0x7f,0x62,0x69,
107 0xb0,0xbb,0xa6,0xad,0x9c,0x97,0x8a,0x81,0xe8,0xe3,0xfe,0xf5,0xc4,0xcf,0xd2,0xd9,
108 0x7b,0x70,0x6d,0x66,0x57,0x5c,0x41,0x4a,0x23,0x28,0x35,0x3e,0x0f,0x04,0x19,0x12,
109 0xcb,0xc0,0xdd,0xd6,0xe7,0xec,0xf1,0xfa,0x93,0x98,0x85,0x8e,0xbf,0xb4,0xa9,0xa2,
110 0xf6,0xfd,0xe0,0xeb,0xda,0xd1,0xcc,0xc7,0xae,0xa5,0xb8,0xb3,0x82,0x89,0x94,0x9f,
111 0x46,0x4d,0x50,0x5b,0x6a,0x61,0x7c,0x77,0x1e,0x15,0x08,0x03,0x32,0x39,0x24,0x2f,
112 0x8d,0x86,0x9b,0x90,0xa1,0xa2,0xb7,0xbc,0xd5,0xde,0xc3,0xc8,0xf9,0xf2,0xef,0xe4,
113 0x3d,0x36,0x2b,0x20,0x11,0x1a,0x07,0x0c,0x65,0x6e,0x73,0x78,0x49,0x42,0x5f,0x54,
114 0xfc,0xe1,0xea,0xdb,0xd0,0xcd,0xc6,0xaf,0xa4,0xb9,0xb2,0x83,0x88,0x95,0x9e,
115 0x47,0x4c,0x51,0x5a,0x6b,0x60,0x7d,0x76,0x1f,0x14,0x09,0x02,0x33,0x38,0x25,0x2e,
116 0x8c,0x87,0x9a,0x91,0xa0,0xab,0xb6,0xbd,0xd4,0xdf,0xc2,0xc9,0xf8,0xf3,0xee,0xe5,
117 0x3c,0x37,0x2a,0x21,0x10,0x1b,0x06,0x0d,0x64,0x6f,0x72,0x79,0x48,0x43,0x5e,0x55,
118 0x01,0x0a,0x17,0x1c,0x2d,0x26,0x3b,0x30,0x59,0x52,0x4f,0x44,0x75,0x7e,0x63,0x68,
119 0xb1,0xba,0xa7,0xac,0x9d,0x96,0x8b,0x80,0xe9,0xe2,0xff,0xf4,0xc5,0xce,0xd3,0xd8,
120 0x7a,0x71,0x6c,0x67,0x56,0x5d,0x40,0x4b,0x22,0x29,0x34,0x3f,0x0e,0x05,0x18,0x13,
121 0xca,0xc1,0xdc,0xd7,0xe6,0xed,0xf0,0x92,0x99,0x84,0x8f,0xbe,0xb5,0xa8,0xa3}
122
123 var mul13 = [256]byte {0x00,0x0d,0x1a,0x17,0x34,0x39,0x2e,0x23,0x68,0x65,0x72,0x7f,0x5c,0x51,0x46,0x4b,
124 0xd0,0xdd,0xca,0xc7,0xe4,0xe9,0xfe,0xf3,0xb8,0xb5,0xa2,0xaf,0x8c,0x81,0x96,0x9b,
125 0xbb,0xb6,0xa1,0xac,0x8f,0x82,0x95,0x98,0xd3,0xde,0xc9,0xc4,0xe7,0xea,0xfd,0xf0,
126 0x6b,0x66,0x71,0x7c,0x5f,0x52,0x45,0x48,0x03,0x0e,0x19,0x14,0x37,0x3a,0x2d,0x20,
127 0x6d,0x60,0x77,0x7a,0x59,0x54,0x43,0x4e,0x05,0x08,0x1f,0x12,0x31,0x3c,0x2b,0x26,
128 0xbd,0xb0,0xa7,0xaa,0x89,0x84,0x93,0x9e,0xd5,0xd8,0xcf,0xc2,0xe1,0xec,0xfb,0xf6,
129 0xd6,0xdb,0xcc,0xc1,0xe2,0xef,0xf8,0xf5,0xbe,0xb3,0xa4,0xa9,0x8a,0x87,0x90,0x9d,
130 0x06,0x0b,0x1c,0x11,0x32,0x3f,0x28,0x25,0x6e,0x63,0x74,0x79,0x5a,0x57,0x40,0x4d,
131 0xda,0xd7,0xc0,0xcd,0xee,0xe3,0xf4,0xf9,0xb2,0xbf,0xa8,0xa5,0x86,0x8b,0x9c,0x91,
132 0x0a,0x07,0x10,0x1d,0x3e,0x33,0x24,0x29,0x62,0x6f,0x78,0x75,0x56,0x5b,0x4c,0x41,
133 0x61,0x6c,0x7b,0x76,0x55,0x58,0x4f,0x42,0x09,0x04,0x13,0x1e,0x3d,0x30,0x27,0x2a,
134 0xb1,0xbc,0xab,0xa6,0x85,0x88,0x9f,0x92,0xd9,0xd4,0xc3,0xce,0xed,0xe0,0xf7,0xfa,
135 0xb7,0xba,0xad,0xa0,0x83,0x8e,0x99,0x94,0xdf,0xd2,0xc5,0xc8,0xeb,0xe6,0xf1,0xfc,
136 0x67,0x6a,0x7d,0x70,0x53,0x5e,0x49,0x44,0x0f,0x02,0x15,0x18,0x3b,0x36,0x21,0x2c,
137 0x0c,0x01,0x16,0x1b,0x38,0x35,0x22,0x2f,0x64,0x69,0x7e,0x73,0x50,0x5d,0x4a,0x47,
138 0xdc,0xd1,0xc6,0xcb,0xe8,0xe5,0xf2,0xb4,0xb9,0xae,0xa3,0x80,0x8d,0x9a,0x97}
139
140 var mul14 = [256]byte {0x00,0x0e,0x1c,0x12,0x38,0x36,0x24,0x2a,0x7e,0x6c,0x62,0x48,0x46,0x54,0x5a,
141 0xe0,0xee,0xfc,0xf2,0xd8,0xd6,0xc4,0xca,0x90,0x9e,0x8c,0x82,0xa8,0xa6,0xb4,0xba,
142 0xdb,0xd5,0xc7,0xc9,0xe3,0xed,0xff,0xf1,0xab,0xa5,0xb7,0xb9,0x93,0x9d,0x8f,0x81,
143 0x3b,0x35,0x27,0x29,0x03,0x0d,0x1f,0x11,0x4b,0x45,0x59,0x73,0x7d,0x6f,0x61}

```

```

144     0xad,0xa3,0xb1,0xbf,0x95,0x9b,0x89,0x87,0xdd,0xd3,0xc1,0xcf,0xe5,0xeb,0xf9,0xf7,
145     0x4d,0x43,0x51,0x5f,0x75,0x7b,0x69,0x67,0x3d,0x33,0x21,0x2f,0x05,0x0b,0x19,0x17,
146     0x76,0x78,0x6a,0x64,0x4e,0x40,0x52,0x5c,0x06,0x08,0x1a,0x14,0x3e,0x30,0x22,0x2c,
147     0x96,0x98,0x8a,0x84,0xae,0xa0,0xb2,0xbc,0xe6,0xe8,0xfa,0xf4,0xde,0xd0,0xc2,0xcc,
148     0x41,0x4f,0x5d,0x53,0x79,0x77,0x65,0x6b,0x31,0x3f,0x2d,0x23,0x09,0x07,0x15,0x1b,
149     0xa1,0xaf,0xbd,0xb3,0x99,0x97,0x85,0x8b,0xd1,0xdf,0xcd,0xc3,0xe9,0xe7,0xf5,0xfb,
150     0x9a,0x94,0x86,0x88,0xac,0xbe,0xb0,0xea,0xe4,0xf6,0xf8,0xd2,0xdc,0xce,0xc0,
151     0x7a,0x74,0x66,0x68,0x42,0x4c,0x5e,0x50,0xa,0x04,0x16,0x18,0x32,0x3c,0x2e,0x20,
152     0xec,0xe2,0xf0,0xfe,0xd4,0xda,0xc8,0xc6,0x9c,0x92,0x80,0x8e,0xa4,0xaa,0xb8,0xb6,
153     0x0c,0x02,0x10,0x1e,0x34,0x3a,0x28,0x26,0x7c,0x72,0x60,0x6e,0x44,0x4a,0x58,0x56,
154     0x37,0x39,0x2b,0x25,0x0f,0x01,0x13,0x1d,0x47,0x49,0x5b,0x55,0x7f,0x71,0x63,0x6d,
155     0xd7,0xd9,0xcb,0xc5,0xef,0xe1,0xf3,0xa7,0xa9,0xbb,0xb5,0x9f,0x91,0x83,0x8d}
156
157
158 func keyExpansionCore(inp [4]byte, i int) [4]byte {
159     // Shift the inp left by moving the first byte to the end (rotate).
160     inp[0], inp[1], inp[2], inp[3] = inp[1], inp[2], inp[3], inp[0]
161
162     // S-Box the bytes
163     inp[0], inp[1], inp[2], inp[3] = sBox[inp[0]], sBox[inp[1]], sBox[inp[2]], sBox[inp[3]]
164
165     // rcon, the round constant
166     inp[0] ^= rcon[i]
167
168     return inp
169 }
170
171 func ExpandKey(inputKey []byte) [176]byte {
172     var expandedKey [176]byte
173     // first 16 bytes of the expandedKey should be the same 16 as the original key
174     for i := 0; i < 16; i++ {
175         expandedKey[i] = inputKey[i]
176     }
177     var bytesGenerated int = 16 // needs to get to 176 to fill expandedKey with 11 keys, one for every round.
178     var rconIteration int = 1
179     var temp [4]byte
180
181     for bytesGenerated < 176{
182         // Read 4 bytes for use in keyExpansionCore
183         copy(temp[:], expandedKey[bytesGenerated-4:bytesGenerated])
184
185         if bytesGenerated % 16 == 0 {      // Keys are length 16 bytes so every 16 bytes generated, expand.
186             temp = keyExpansionCore(temp, rconIteration)
187             rconIteration += 1
188         }
189
190         for y := 0; y < 4; y++ {
191             expandedKey[bytesGenerated] = expandedKey[bytesGenerated - 16] ^ temp[y] // XOR first 4 bytes of
192             previous key with the temporary list.
193             bytesGenerated += 1
194         }
195
196         return expandedKey
197     }
198
199 func addRoundKey(state []byte, roundKey []byte) {      // Add round key is also it's own inverse
200     state[ 0], state[ 1], state[ 2], state[ 3],
201     state[ 4], state[ 5], state[ 6], state[ 7],
202     state[ 8], state[ 9], state[10], state[11],
203     state[12], state[13], state[14], state[15] =
204
205     state[ 0]^roundKey[ 0], state[ 1]^roundKey[ 1], state[ 2]^roundKey[ 2], state[ 3]^roundKey[ 3],
206     state[ 4]^roundKey[ 4], state[ 5]^roundKey[ 5], state[ 6]^roundKey[ 6], state[ 7]^roundKey[ 7],
207     state[ 8]^roundKey[ 8], state[ 9]^roundKey[ 9], state[10]^roundKey[10], state[11]^roundKey[11],
208     state[12]^roundKey[12], state[13]^roundKey[13], state[14]^roundKey[14], state[15]^roundKey[15]
209 }
210
211 func subBytes(state []byte) {

```

```

212     state[ 0], state[ 1], state[ 2], state[ 3],
213     state[ 4], state[ 5], state[ 6], state[ 7],
214     state[ 8], state[ 9], state[10], state[11],
215     state[12], state[13], state[14], state[15] =
216
217     sBox[state[ 0]], sBox[state[ 1]], sBox[state[ 2]], sBox[state[ 3]],
218     sBox[state[ 4]], sBox[state[ 5]], sBox[state[ 6]], sBox[state[ 7]],
219     sBox[state[ 8]], sBox[state[ 9]], sBox[state[10]], sBox[state[11]],
220     sBox[state[12]], sBox[state[13]], sBox[state[14]], sBox[state[15]]
221 }
222
223 func invSubBytes(state []byte) {
224     state[ 0], state[ 1], state[ 2], state[ 3],
225     state[ 4], state[ 5], state[ 6], state[ 7],
226     state[ 8], state[ 9], state[10], state[11],
227     state[12], state[13], state[14], state[15] =
228
229     invSBox[state[ 0]], invSBox[state[ 1]], invSBox[state[ 2]], invSBox[state[ 3]],
230     invSBox[state[ 4]], invSBox[state[ 5]], invSBox[state[ 6]], invSBox[state[ 7]],
231     invSBox[state[ 8]], invSBox[state[ 9]], invSBox[state[10]], invSBox[state[11]],
232     invSBox[state[12]], invSBox[state[13]], invSBox[state[14]], invSBox[state[15]]
233 }
234
235 func shiftRows(state []byte) {
236     state[ 0], state[ 1], state[ 2], state[ 3],
237     state[ 4], state[ 5], state[ 6], state[ 7],
238     state[ 8], state[ 9], state[10], state[11],
239     state[12], state[13], state[14], state[15] =
240
241     state[ 0], state[ 5], state[10], state[15],
242     state[ 4], state[ 9], state[14], state[ 3],
243     state[ 8], state[13], state[ 2], state[ 7],
244     state[12], state[ 1], state[ 6], state[11]
245 }
246 // Shifts it like this:
247 //
248 // 0 4 8 12      0 4 8 12  Shifted left by 0
249 // 1 5 9 13  ----> 5 9 13 1  Shifted left by 1
250 // 2 6 10 14  ----> 10 14 2 6  Shifted left by 2
251 // 3 7 11 15      15 3 7 11  Shifted left by 3
252
253 func invShiftRows(state []byte) {
254     state[ 0], state[ 1], state[ 2], state[ 3],
255     state[ 4], state[ 5], state[ 6], state[ 7],
256     state[ 8], state[ 9], state[10], state[11],
257     state[12], state[13], state[14], state[15] =
258
259     state[ 0], state[13], state[10], state[ 7],
260     state[ 4], state[ 1], state[14], state[11],
261     state[ 8], state[ 5], state[ 2], state[15],
262     state[12], state[ 9], state[ 6], state[ 3]
263 }
264 // 0 4 8 12      0 4 8 12  Shifted right by 0
265 // 5 9 13 1  ----> 1 5 9 13  Shifted right by 1
266 // 10 14 2 6  ----> 2 6 10 14  Shifted right by 2
267 // 15 3 7 11      3 7 11 15  Shifted right by 3
268
269 func mixColumns(state []byte) {
270     state[ 0], state[ 1], state[ 2], state[ 3],
271     state[ 4], state[ 5], state[ 6], state[ 7],
272     state[ 8], state[ 9], state[10], state[11],
273     state[12], state[13], state[14], state[15] =
274
275     mul2[state[ 0]] ^ mul3[state[ 1]] ^ state[ 2] ^ state[ 3],
276     state[ 0] ^ mul2[state[ 1]] ^ mul3[state[ 2]] ^ state[ 3],
277     state[ 0] ^ state[ 1] ^ mul2[state[ 2]] ^ mul3[state[ 3]],
278     mul3[state[ 0]] ^ state[ 1] ^ state[ 2] ^ mul2[state[ 3]],
279
280     mul2[state[ 4]] ^ mul3[state[ 5]] ^ state[ 6] ^ state[ 7],

```

```

281     state[ 4] ^ mul2[state[ 5]] ^ mul3[state[ 6]] ^ state[ 7],
282     state[ 4] ^ state[ 5] ^ mul2[state[ 6]] ^ mul3[state[ 7]],
283     mul3[state[ 4]] ^ state[ 5] ^ state[ 6] ^ mul2[state[ 7]],
284
285     mul2[state[ 8]] ^ mul3[state[ 9]] ^ state[10] ^ state[11],
286     state[ 8] ^ mul2[state[ 9]] ^ mul3[state[10]] ^ state[11],
287     state[ 8] ^ state[ 9] ^ mul2[state[10]] ^ mul3[state[11]],
288     mul3[state[ 8]] ^ state[ 9] ^ state[10] ^ mul2[state[11]],
289
290     mul2[state[12]] ^ mul3[state[13]] ^ state[14] ^ state[15],
291     state[12] ^ mul2[state[13]] ^ mul3[state[14]] ^ state[15],
292     state[12] ^ state[13] ^ mul2[state[14]] ^ mul3[state[15]],
293     mul3[state[12]] ^ state[13] ^ state[14] ^ mul2[state[15]]
294 }
295
296 func invMixColumns(state []byte) {
297     state[ 0], state[ 1], state[ 2], state[ 3],
298     state[ 4], state[ 5], state[ 6], state[ 7],
299     state[ 8], state[ 9], state[10], state[11],
300     state[12], state[13], state[14], state[15] =
301
302     mul14[state[ 0]] ^ mul11[state[ 1]] ^ mul13[state[ 2]] ^ mul9[state[ 3]],
303     mul9[state[ 0]] ^ mul14[state[ 1]] ^ mul11[state[ 2]] ^ mul13[state[ 3]],
304     mul13[state[ 0]] ^ mul9[state[ 1]] ^ mul14[state[ 2]] ^ mul11[state[ 3]],
305     mul11[state[ 0]] ^ mul13[state[ 1]] ^ mul9[state[ 2]] ^ mul14[state[ 3]],
306
307     mul14[state[ 4]] ^ mul11[state[ 5]] ^ mul13[state[ 6]] ^ mul9[state[ 7]],
308     mul9[state[ 4]] ^ mul14[state[ 5]] ^ mul11[state[ 6]] ^ mul13[state[ 7]],
309     mul13[state[ 4]] ^ mul9[state[ 5]] ^ mul14[state[ 6]] ^ mul11[state[ 7]],
310     mul11[state[ 4]] ^ mul13[state[ 5]] ^ mul9[state[ 6]] ^ mul14[state[ 7]],
311
312     mul14[state[ 8]] ^ mul11[state[ 9]] ^ mul13[state[10]] ^ mul9[state[11]],
313     mul9[state[ 8]] ^ mul14[state[ 9]] ^ mul11[state[10]] ^ mul13[state[11]],
314     mul13[state[ 8]] ^ mul9[state[ 9]] ^ mul14[state[10]] ^ mul11[state[11]],
315     mul11[state[ 8]] ^ mul13[state[ 9]] ^ mul9[state[10]] ^ mul14[state[11]],
316
317     mul14[state[12]] ^ mul11[state[13]] ^ mul13[state[14]] ^ mul9[state[15]],
318     mul9[state[12]] ^ mul14[state[13]] ^ mul11[state[14]] ^ mul13[state[15]],
319     mul13[state[12]] ^ mul9[state[13]] ^ mul14[state[14]] ^ mul11[state[15]],
320     mul11[state[12]] ^ mul13[state[13]] ^ mul9[state[14]] ^ mul14[state[15]]
321 }
322
323 func Encrypt(state []byte, expandedKey *[176]byte) {
324     addRoundKey(state, expandedKey[:16])
325
326     for i := 0; i < 144; i += 16 { // 9 regular rounds * 16 = 144
327         subBytes(state)
328         shiftRows(state)
329         mixColumns(state)
330         addRoundKey(state, expandedKey[i+16:i+32])
331     }
332     // Last round
333     subBytes(state)
334     shiftRows(state)
335     addRoundKey(state, expandedKey[160:])
336 }
337
338 func Decrypt(state []byte, expandedKey *[176]byte) {
339     addRoundKey(state, expandedKey[160:])
340     invShiftRows(state)
341     invSubBytes(state)
342
343     for i := 144; i != 0; i -= 16 {
344         addRoundKey(state, expandedKey[i:i+16])
345         invMixColumns(state)
346         invShiftRows(state)
347         invSubBytes(state)
348     }
349     // Last round

```

```
350     addRoundKey(state, expandedKey[:16])
351 }
```

MixColumns is the same as in **Design**, where I explain how lookup tables can be used towards the end of the **Mix Columns** section.

In Go, names of files and global variables starting with a capital letter are exported in the package, hence why Encrypt, Decrypt and ExpandKey all start with a capital letter, while other names in the file begin with a lower case letter. These names can be accessed when the package is imported like this:

```
1 package main
2
3 import (
4     "package"
5 )
6
7 func main() {
8     package.Name()
9 }
```

The expanded key is passed by reference to Encrypt and Decrypt so it does not keep being copied by functions. This increases speed slightly. The state is passed to each function as a slice, which is an array that can change size. In Go, slices are always passed by reference, so I do not need to worry about dereferencing it or anything, and hence why none of the functions in the `AES` package return any values (apart from for key expansion).

AESfiles package

Here is the `AESfiles` package in `code/python-go/AES/src/AES/AESfiles/aesFiles.go`:

```
1 package AESfiles
2
3 import (
4     "os"          // For opening files
5     "io"          // For reading files
6     "runtime"     // For getting CPU core count
7     "AES"
8     "AES/AEScheckKey"
9 )
10
11 const DEFAULT_BUFFER_SIZE = 65536 // Define the default buffer size for enc/decrypt (is 2^16)
12
13 func check(e error) { // Checks error given
14     if e != nil { panic(e) }
15 }
16
17 func getNumOfCores() int { // Gets the number of cores so the number of workers can be determined.
18     maxProcs := runtime.GOMAXPROCS(0)
19     numCPU := runtime.NumCPU()
20     if maxProcs < numCPU {
21         return maxProcs
22     }
23     return numCPU
24 }
25
26 // For holding the buffer to be worked on and the offset together, so it can be written to the file in the
27 // correct place at the end.
28 type work struct {
29     buff []byte
30     offset int64
31 }
```

```

31
32 func workerEnc(jobs <-chan work, results chan<- work, expandedKey *[176]byte) { // Encrypts a chunk when
33     given (a chunk of length bufferSize)
34     for job := range jobs {
35         for i := 0; i < len(job.buff); i += 16 {
36             AES.Encrypt(job.buff[i:i+16], expandedKey)
37         }
38         results<- job // Return result with encrypted job
39     }
40 }
41
42 func workerDec(jobs <-chan work, results chan<- work, expandedKey *[176]byte, fileSize int) {
43     for job := range jobs {
44         for i := 0; i < len(job.buff); i += 16 {
45             AES.Decrypt(job.buff[i:i+16], expandedKey)
46         }
47         results<- job // Return job with decrypted buffer
48     }
49 }
50
51 func EncryptFile(expandedKey *[176]byte, f, w string) {
52     a, err := os.Open(f) // Open original file to get statistics and read data.
53     check(err)
54     aInfo, err := a.Stat() // Get statistics
55     check(err)
56
57     fileSize := int(aInfo.Size()) // Get size of original file
58
59     if _, err := os.Stat(w); err == nil { // If file already exists, delete it
60         os.Remove(w)
61     }
62
63     var workingWorkers int = 0
64     var workerNum int = getNumOfCores()*2
65
66     jobs := make(chan work, workerNum) // Make two channels for go routines to communicate over.
67     results := make(chan work, workerNum) // Each has a buffer of length workerNum
68
69     for i := 0; i < workerNum; i++ {
70         go workerEnc(jobs, results, expandedKey) // Create the workers
71     }
72     /*
73      Each go routine will be given access to the job channel, where each worker then waits to complete the job.
74      Once the job is completed, the go routine pushes the result onto the result channel, where the result can be
75      received by the main routine. The results are read once all of the go routines are busy, or if the file
76      is completed, then the remaining workers still working are asked for their results.
77     */
78     var bufferSize int = DEFAULT_BUFFER_SIZE
79
80     if fileSize < bufferSize { // If the buffer size is larger than the file size, just read the whole file.
81         bufferSize = fileSize
82     }
83
84     var buffCount int = 0 // Keeps track of how far through the file we are
85
86     e, err := os.OpenFile(w, os.O_CREATE|os.O_WRONLY, 0644) // Open file for writing.
87     check(err) // Check it opened correctly
88
89     // Append key so that when decrypting, the key can be checked before decrypting the whole file.
90     originalKey := make([]byte, 16)
91     for i := 0; i < 16; i++ {
92         originalKey[i] = expandedKey[i]
93     }
94     AES.Encrypt(originalKey, expandedKey)
95     e.Write(originalKey)
96     offset := 16
97
98     for buffCount < fileSize { // Same as a while buffCount < fileSize: in python3
99         if bufferSize > (fileSize - buffCount) {

```

```

99     bufferSize = fileSize - buffCount    // If this is the last block, read the amount of data left in the
100    file.
101
102    buf := make([]byte, bufferSize) // Make a slice the size of the buffer
103    _, err := io.ReadFull(a, buf) // Read the contents of the original file, but only enough to fill the buff
104    array.                      // The "_" tells go to ignore the value returned by io.ReadFull, which in
105    this case is the number of bytes read.
106    check(err)
107
108    if len(buf) % 16 != 0 { // If the buffer is not divisible by 16 (usually the end of a file), then
109        padding needs to be added.
110        var extraNeeded int
111        var l int = len(buf)
112        for l % 16 != 0 {      // extraNeeded holds the value for how much padding the block needs.
113            l++
114            extraNeeded++
115        }
116
117        for i := 0; i < extraNeeded; i++ {           // Add the number of extra bytes needed to the end
118            of the block, if the block is not long enough.
119            buf = append(buf, byte(extraNeeded)) // For example, the array [1, 1, 1, 1, 1, 1, 1, 1] would have
120            the number 8 appended to then end 8 times to make the array 16 in length.
121            } // This is so that when the block is decrypted, the pattern can be recognised, and the correct amount
122            of padding can be removed.
123        }
124
125        jobs <- work{buff: buf, offset: int64(offset)} // Input new work into the jobs channel.
126        workingWorkers++
127
128        if workingWorkers == workerNum { // Once all workers are working, wait for results.
129            workingWorkers = 0
130            for i := 0; i < workerNum; i++ {
131                wk := <-results
132                e.WriteAt(wk.buf, wk.offset) // Write the buffer at the offset specified.
133            }
134
135        if workingWorkers != 0 { // If there are still workers working, then accept the results.
136            for i := 0; i < workingWorkers; i++ {
137                wk := <-results
138                e.WriteAt(wk.buf, wk.offset)
139            }
140        }
141
142        close(jobs) // Close the channels since the file has been finished.
143        close(results)
144
145        a.Close() // Close the files used.
146        e.Close()
147    }
148
149
150    func DecryptFile(expandedKey *[176]byte, f, w string) {
151        a, err := os.Open(f)
152        check(err)
153        aInfo, err := a.Stat()
154        check(err)
155
156        fileSize := int(aInfo.Size())-16 // Take away length of added key for checksum
157
158        if _, err := os.Stat(w); err == nil { // If file exists, delete it
159            os.Remove(w)
160        }

```

```

161
162     var bufferSize int = DEFAULT_BUFFER_SIZE
163
164     var workingWorkers int = 0
165     var workerNum int = getNumOfCores()*2
166
167     jobs := make(chan work, workerNum)      // Make two channels for go routines to communicate over.
168     results := make(chan work, workerNum)   // Each has a buffer of length workerNum
169
170     for i := 0; i < workerNum; i++ {
171         go workerDec(jobs, results, expandedKey, fileSize)
172     }
173
174     if fileSize < bufferSize {
175         bufferSize = fileSize
176     }
177
178     var buffCount int = 0
179
180     e, err := os.OpenFile(w, os.O_CREATE|os.O_WRONLY, 0644) // Open file
181     check(err)
182
183     // Check first block is key
184     firstBlock := make([]byte, 16)
185     _, er := io.ReadFull(a, firstBlock)
186     check(er)
187
188     if AEScheckKey.CheckKey(expandedKey, firstBlock) { // If key is valid
189         offset := 0
190         a.Seek(16, 0) // Move past key in encrypted file
191         for buffCount < fileSize{ // While the data done is less than the fileSize
192             if bufferSize > (fileSize - buffCount) {
193                 bufferSize = fileSize - buffCount
194             }
195
196             buff := make([]byte, bufferSize)
197             _, err := io.ReadFull(a, buff) // Ignore the number of bytes read (_)
198             check(err)
199
200             jobs<- work{buff: buff, offset: int64(offset)}
201             workingWorkers++
202
203             if workingWorkers == workerNum {
204                 workingWorkers = 0
205                 for i := 0; i < workerNum; i++ {
206                     wk := <-results
207                     e.WriteAt(wk(buff, wk.offset)
208                 }
209             }
210
211             offset += bufferSize
212             buffCount += bufferSize
213         }
214
215         if workingWorkers != 0 {
216             for i := 0; i < workingWorkers; i++ { // Collect all but last block
217                 wk := <-results
218                 if int(wk.offset)+bufferSize >= fileSize { // If the offset is the last block in file
219                     wk(buff = checkForPadding(wk(buff))
220                 }
221                 e.WriteAt(wk(buff, wk.offset)
222             }
223         }
224         close(jobs)
225         close(results)
226
227     } else {
228         panic("Invalid Key") // If first block is not equal to the key, then do not bother trying to decrypt the
file.

```

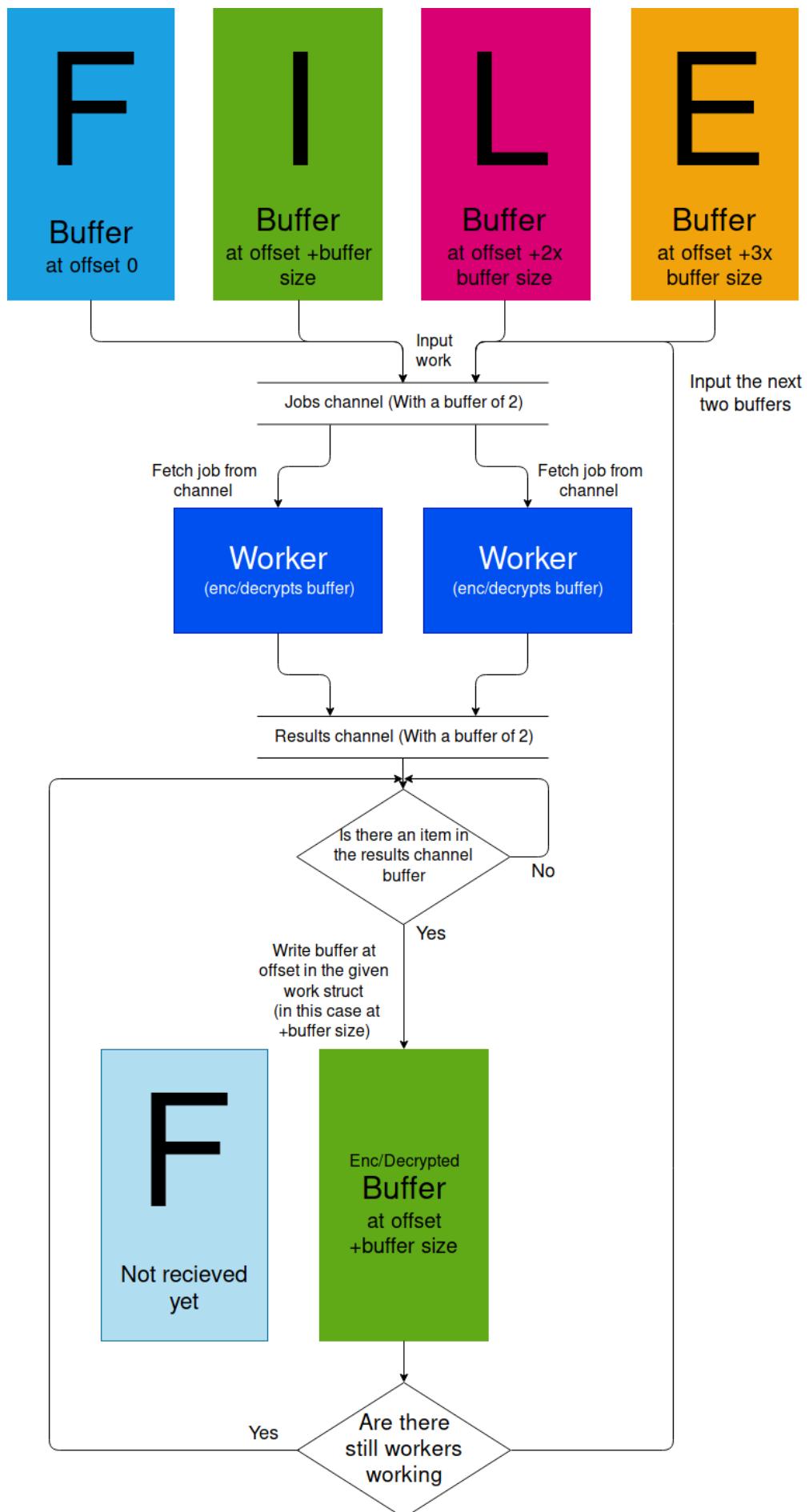
```

229     }
230     a.Close()
231     e.Close()
232 }
233
234 func checkForPadding(buffer []byte) []byte { // Checks a block for padding
235     var focus byte = buffer[len(buffer)-1]
236     var focusCount byte = 0
237     if focus < 16 {
238         for j := 1; (buffer[len(buffer)-j] == focus) && (j <= 16); j++ {
239             if buffer[len(buffer)-j] == focus { focusCount++ }
240         }
241         if focus == focusCount {
242             buffer = buffer[:len(buffer)-int(focusCount)] // If the number of bytes at the end is equal to the
243             // value of each byte, then remove them, as it is padding.
244         }
245     }
246     return buffer
247 }
248
249 // For dealing with directories
250 func EncryptList(expandedKey *[176]byte, fileList []string, targetList []string) { // Encrypts list of files
251     given to the corresponding targets.
252     if len(fileList) != len(targetList) { panic("fileList and targList are different in length") }
253     for i := range fileList {
254         EncryptFile(expandedKey, fileList[i], targetList[i])
255     }
256 }
257
258 func DecryptList(expandedKey *[176]byte, fileList []string, targetList []string) { // Decrypts list of files
259     given to the corresponding targets.
260     if len(fileList) != len(targetList) { panic("fileList and targList are different in length") }
261     for i := range fileList {
262         DecryptFile(expandedKey, fileList[i], targetList[i])
263     }
264 }
```

Enc/decryption of files is done in parallel (using multiple processes/CPU cores) by having 'workers' that accept jobs from a work channel, do work on each job given and return them in a results channel. The workers are really just a function that is run in a **goroutine** (coroutines that are designed to be easy to use and low on memory), and each worker waits for a job on the job channel while the job channel is open. The job channel allows for the transfer of a 'work' struct to the goroutine, containing a full buffer of the original file, and what offset in the file that buffer was read from. The offset is important because each goroutine may finish at a different time, and the data has to be in the correct order, however it does not need to be written to the new file at the same time. As long as it is in order it is fine.

If on the last block of decryption, then the program checks for padding, which works the same as specified in the **Design** section. If the last element of the last block has a value less than 16, it counts how many of that number occur going back through the block, and if the number of occurrences matches the value itself, then the occurrences

Here is a visual representation of this process:



The number of workers is determined by the number of cores, and is then multiplied by 2 because I wanted the CPU usage to be about 85-90%, using a lot of cpu usage but also leaving room for other processes. The `DEFAULT_BUFFER_SIZE` is set to a power of 2 (2^x) as this is a very divisible number, and computers work best with powers of 2.

I did play with the `DEFAULT_BUFFER_SIZE` value on multiple machines, and ended up settling on 2^{16} , as it seemed to perform best on several computers.

`EncryptList` and `DecryptList` take a list of files to enc/decrypt, and a list of targets to decrypt each file in the list to. This is used when enc/decrypting folders. Separating this function from the generation of the lists is a good idea because it makes programming additional features more flexible, as the list does not necessarily need to be created in Go.

The lists are generated in the `AESstring` package, which we will look at next.

AESstring package

The AESstring package contains functions for enc/decrypting strings, and generating lists of directories that need enc/decrypting.

All file names are encoded in base 64 to allow for larger file names to be encrypted (since with hex encoding, the name's size would increase by 2x, since each character is 1 byte). The maximum file name length on most operating systems is 255 in length, and the maximum length for a string to be encoded into base 64 to be within the 255 byte limit is 176 bytes.

Here is the package (`code/python-go/AES/src/AES/AESstring/aesString.go`):

```
1 package AESstring
2
3 import (
4     "os"           // For making new folders
5     "log"          // For debugging
6     "io/ioutil"    // For listing contents of directories
7     b64 "encoding/base64" // For enc/decoding encrypted string
8     "AES"
9     "sorts" // QuickSortAlph made in sorts.go
10 )
11
12 func EncryptFileName(expandedKey *[176]byte, name string) string {
13     byteName := []byte(name)
14
15     for len(byteName) % 16 != 0 { // Pad with 0's
16         byteName = append(byteName, 0)
17     }
18
19     for i := 0; i < len(byteName); i += 16 {
20         AES.Encrypt(byteName[i:i+16], expandedKey) // Done by reference so does not need to be assigned
21     }
22     return b64.URLEncoding.EncodeToString(byteName) // URL encoding used so it is safe for file systems ("/")
23 }
24
25 func DecryptFileName(expandedKey *[176]byte, hexName string) string {
26     byteName, err := b64.URLEncoding.DecodeString(hexName)
27     if err != nil { panic(err) }
28
29     for i := 0; i < len(byteName); i += 16 {
30         AES.Decrypt(byteName[i:i+16], expandedKey)
31     }
32     byteName = checkForPadding(byteName)
33     return string(byteName[:])
34 }
```

```

35 func checkForPadding(input []byte) []byte {
36     var newBytes []byte
37     for _, element := range input {
38         if (element > 31) && (element < 127) { //If a character
39             newBytes = append(newBytes, element)
40         }
41     }
42 }
43 return newBytes
44 }
45
46 func EncryptListOfString(expandedKey *[176]byte, l []string) []string {
47     for i := range l {
48         l[i] = EncryptFileName(expandedKey, l[i])
49     }
50     return l
51 }
52
53 func DecryptListOfString(expandedKey *[176]byte, l []string) []string {
54     var out []string
55     for i := range l {
56         out = append(out, DecryptFileName(expandedKey, l[i]))
57     }
58     return out
59 }
60
61 func GetListsEnc(expandedKey *[176]byte, fileList, targetList []string, folder, target string) ([]string,
62 []string) { // Also makes the folders required
63     os.Mkdir(target, os.ModePerm)
64     list, err := ioutil.ReadDir(folder)
65     if err != nil { panic(err) } // Go has weird error handling
66     for i := range list {
67         if len(list[i].Name()) <= 176 {
68             if list[i].IsDir() {
69                 fileList, targetList = GetListsEnc(expandedKey, fileList, targetList, folder+list[i].Name()+"/",
70 target+EncryptFileName(expandedKey, list[i].Name())+"/") // Recursively go through folders
71             } else {
72                 fileList = append(fileList, folder+list[i].Name())
73                 targetList = append(targetList, target+EncryptFileName(expandedKey, list[i].Name()))
74             }
75         } else {
76             log.Output(0, "File name too long: "+list[i].Name())
77         }
78     }
79     return fileList, targetList
80 }
81
82 func GetListsDec(expandedKey *[176]byte, fileList, targetList []string, folder, target string) ([]string,
83 []string) {
84     os.Mkdir(target, os.ModePerm)
85     list, err := ioutil.ReadDir(folder)
86     if err != nil { panic(err) }
87     for i := range list {
88         if list[i].IsDir() {
89             fileList, targetList = GetListsDec(expandedKey, fileList, targetList, folder+list[i].Name()+"/",
90 target+DecryptFileName(expandedKey, list[i].Name())+"/")
91         } else {
92             fileList = append(fileList, folder+list[i].Name())
93             targetList = append(targetList, target+DecryptFileName(expandedKey, list[i].Name()))
94         }
95     }
96     return fileList, targetList
97 }
98
99 func getSortedFoldersAndFiles(inp []sorts.Tuple) ([]string, []string) {
100    var files []sorts.Tuple
101    var folders []sorts.Tuple
102    for i := 0; i < len(inp); i++ {
103        if inp[i].A.IsDir() {
104

```

```

100     folders = append(folders, inp[i])
101 } else {
102     files = append(files, inp[i])
103 }
104 }
105 foldersSort, filesSort := sorts.QuickSortAlpha(folders), sorts.QuickSortAlpha(files) // Sort the folders and
files.
106 var (
107     encOut []string
108     decOut []string
109 )
110 for x := range foldersSort { // Append folder names to each list.
111     encOut = append(encOut, foldersSort[x].A.Name())
112     decOut = append(decOut, foldersSort[x].B)
113 }
114 for y := range filesSort { // Append file names to each list.
115     encOut = append(encOut, filesSort[y].A.Name())
116     decOut = append(decOut, filesSort[y].B)
117 }
118 return encOut, decOut
119 }
120
121 func GetListOfFiles(expandedKey *[176]byte, dir string) ([]string, []string) { // Decrypts a list of files at
the directory specified, also returning original list
122     list, err := ioutil.ReadDir(dir)
123     if err != nil { panic(err) }
124     l := make([]sorts.Tuple, 0)
125     var listOfNames []string
126     for x := range list {
127         listOfNames = append(listOfNames, list[x].Name())
128     }
129     dec := DecryptListToString(expandedKey, listOfNames)
130
131     for i := range list {
132         l = append(l, sorts.Tuple{A: list[i], B: dec[i]}) // Make a tuple containing the encrypted name and the
decrypted name.
133     } // These need to be paired so the encrypted names can
be returned in order.
134     return getSortedFoldersAndFiles(l)
135 }

```

In `GetListOfFiles`, the directory given is listed, decrypted, and then tuples containing the file info object and decrypted name are sorted.

AEScheckKey package

This package handles key checking. It takes the first 16 byte block of a file, or a block given, and decrypts the block then compares it against the key given to decrypt it with. This is because in encryption, the key is encrypted and written to the first 16 bytes of the file so that it can be checked when decrypting it.

This system works well, and is better than decrypting the whole file just to try and open it and realising that the key was incorrect.

Here is the code for the AEScheckKey package ([code/python-go/AES/src/AES/AEScheckKey/aesCheckKey.go](#)):

```

1 package AEScheckKey
2
3 import (
4     "os"
5     "io"
6     "AES"
7 )
8

```

```

9 func compareSlices(slice1, slice2 []byte) bool {    // Function used for checking first block of a file with
10    the key when decrypting.
11    if len(slice1) != len(slice2) {
12        return false
13    } else {
14        for i := 0; i < len(slice1); i++ {
15            if slice1[i] != slice2[i] {
16                return false
17            }
18        }
19    }
20    return true
21 }
22
23 func CheckKey(expandedKey *[176]byte, block []byte) bool {
24     AES.Decrypt(block, expandedKey)      // Decrypt first block
25     return compareSlices(expandedKey[:16], block) // Compare decrypted first block with the key.
26 }
27
28 func CheckKeyOfFile(key []byte, f string) bool {
29     a, err := os.Open(f)      // Open an encrypted file to check first block against key
30     if err != nil { panic(err) }
31
32     firstBlock := make([]byte, 16)
33     _, er := io.ReadFull(a, firstBlock)    // Fill a slice of length 16 with the first block of 16 bytes in the
34     file.
35     if er != nil { panic(er) }
36     a.Close()
37     expKey := AES.ExpandKey(key)
38     return CheckKey(&expKey, firstBlock)
39 }
```

The expanded key `expKey` is passed by reference, so on line 36, the "&" gets the memory address (pointer) to the `expKey` array.

`CheckKeyOfFile` decrypts the first block of a file and compares it to the key. If the decrypted block is the same as the key, then the key is valid. `CheckKey` does the same, however it takes a 16 byte block and checks the key against it. `compareSlices` is used internally to check the key against the decrypted block, so it does not need to be exported so the first letter of the function is lower case.

The main package

Every Go program that runs by itself requires a main package, with a main function.

Here is the main package (`code/python-go/AES/main.go`):

```

1 package main
2
3 import (
4     "AES"
5     "AES/AEScheckKey"
6     "AES/AESfiles"
7     "AES/AESstring"
8     "fmt"          // For sending output on stout
9     "io/ioutil"   // For reading from stdin
10    "log"
11    "os"         // Gets stdin
12    "sorts"
13    "strconv"   // ^
14    "strings"    // For converting string key to an array of bytes
15 )
16
17 func strToInt(str string) (int, error) { // Used for converting string to integer, as go doesn't have that
18    built in for some reason
19 }
```

```

18    n := strings.Split(str, ".") // Splits by decimal point
19    return strconv.Atoi(n[0])    // Returns integer of whole number
20 }
21
22 func main() {
23     bytes, err := ioutil.ReadAll(os.Stdin)
24     if err != nil {
25         panic(err)
26     }
27     fields := strings.Split(string(bytes), " ")
28     request := string(fields[0])
29     var expandedKey [176]byte
30     var key []byte
31
32     if string(fields[3]) != "" { // If the key has been passed.
33         keyString := strings.Split(string(fields[3]), " ")
34         for i := 0; i < len(keyString); i++ {
35             a, err := strToInt(keyString[i])
36             if err != nil {
37                 panic(err)
38             }
39             key = append(key, byte(a))
40         }
41         expandedKey = AES.ExpandKey(key)
42
43     } else {
44         log.Output(0, "EGG")
45     }
46
47     if request == "y" {
48         AESfiles.EncryptFile(&expandedKey, string(fields[1]), string(fields[2]))
49     } else if request == "n" {
50         AESfiles.DecryptFile(&expandedKey, string(fields[1]), string(fields[2]))
51     } else if request == "yDir" {
52         AESfiles.EncryptList(&expandedKey, strings.Split(string(fields[1]), "\n"), strings.Split(string(fields[2]),
53                                         "\n"))
54     } else if request == "nDir" {
55         AESfiles.DecryptList(&expandedKey, strings.Split(string(fields[1]), "\n"), strings.Split(string(fields[2]),
56                                         "\n"))
57     } else if request == "encString" {
58         fmt.Println(AESstring.EncryptFileName(&expandedKey, string(fields[1])))
59     } else if request == "decString" {
60         fmt.Println(AESstring.DecryptFileName(&expandedKey, string(fields[1])))
61     } else if request == "encList" {
62         fmt.Println(strings.Join(AESstring.EncryptListOfString(&expandedKey, strings.Split(string(fields[1]), "\n")),
63                           ","))
64     } else if request == "decList" {
65         fmt.Println(strings.Join(AESstring.DecryptListOfString(&expandedKey, strings.Split(string(fields[1]), "\n")),
66                           ","))
67     } else if request == "getListsY" {
68         fileList, targList := AESstring.GetListsEnc(&expandedKey, []string{}, []string{}, string(fields[1]),
69                                         string(fields[2]))
70         fmt.Println(strings.Join(fileList, ",") + "--!--")
71         fmt.Println(strings.Join(targList, ","))
72     } else if request == "getListsN" {
73         fileList, targList := AESstring.GetListsDec(&expandedKey, []string{}, []string{}, string(fields[1]),
74                                         string(fields[2]))
75         fmt.Println(strings.Join(fileList, ",") + "--!--")
76         fmt.Println(strings.Join(targList, ","))
77     } else if request == "listDir" {
78         log.Output(0, "Getting List")
79         fs, fsDec := AESstring.GetListOfFiles(&expandedKey, string(fields[1]))
80         fmt.Println(strings.Join(fs, ",") + "--!--")
81         fmt.Println(strings.Join(fsDec, ","))
82     } else if request == "sortSize" {
83         fmt.Println(strings.Join(sorts.UseQuickSortSize(strings.Split(string(fields[1]), "\n)), ","))
84     } else if request == "sortAlpha" {
85         fmt.Println(strings.Join(sorts.UseQuickSortAlpha(strings.Split(string(fields[1]), "\n)), ","))
86     } log.Output(0, "Sorting alpha")

```

```

81 } else if request == "sortSearch" {
82     fmt.Println(strings.Join(sorts.UseQuickSortSearch(strings.Split(string(fields[1]), "\n"),
83     strings.Split(string(fields[2]), "\n)), ","))
84 } else if request == "test" {
85     valid := AEScheckKey.CheckKeyOfFile(key, string(fields[1]))
86     if valid {
87         fmt.Println("-Valid-")
88     } else {
89         fmt.Println("-NotValid-")
90     }
91 } else {
92     panic("Invalid options.")
93 }

```

The `main` function is run when the executable, `code/python-go/AES/AES` or `code/python-go/AESWin.exe` for Windows, is started. My `main` function accepts input from `stdin` (system's way of communicating between programs), which stands for standard input. The results are returned on `stdout` (standard output), which are then received by Python.

The program accepts the fields `<encryptionType>, <field1>, <field2>, <key>`, where `<field1>` is the first argument of the function you want to execute, and `<field2>` is the second argument. If there is no `<field2>` argument, then this can just be set to `0`. The key is input like this: `1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16` (it is hashed first though), where it is then split by the space in-between each number, and each number is converted to a byte, where it can be used in the functions that need it.

If an array is needed to be passed through `field1` or `field2`, then it is joined with a new line `"\n"`, as the fields are separated with `,`. If an array is returned by the program, it is joined with `, , ,`, and if multiple arrays are to be returned then they are separated with `--!--`.

A full command to AES would look like this:

```
1 | 'y', '/home/josh/file.png', '/home/josh/temp', '1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16'
```

This string would get passed to the standard input of the program.

`aes.go` is compiled to `AES` for Linux/MacOS, and `AESWin.exe` for Windows.

`build.sh` just compiles the main code. Here it is:

```

1 |#!/bin/bash
2 |export GOPATH="/home/josh/neal-12ColcloughJ/code/python-go/AES/"
3 |go build

```

It exists because I am too lazy to paste the GOPATH environment variable into the terminal whenever I want to build it. Go requires that you set the GOPATH environment variable saying where the source code is to compile it. This bash script made it easier to go from building AES to building BLAKE.

SHA256:

Here is the code for SHA256 (`code/python-go/SHA.py`, `code/mobile/SHA.py`):

```

1 | k = [0x428a2f98, 0x71374491, 0xb5c0fbef, 0xe9b5dba5,      #Round constants
2 |     0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
3 |     0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
4 |     0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
5 |     0xe49b69c1, 0xefbe4786, 0xfc19dc6, 0x240ca1cc,

```

```

6      0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
7      0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
8      0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
9      0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
10     0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
11     0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
12     0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
13     0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
14     0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
15     0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
16     0x90beffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2]
17
18 def makeBitArray(inp):
19     bitArray = []
20     for element in inp:
21         tempByte = intToBits(element)
22         for bit in tempByte:
23             bitArray.append(bit)
24     return bitArray
25
26 def intToBits(inp, bitLength=8):
27     tempByte = []
28     for x in range(bitLength):
29         tempByte.append(0) #Initialize
30     for i in range(bitLength):
31         tempByte[(bitLength-1)-i] = (inp >> i) & 1 #Goes through bits backwards so append backwards.
32     return tempByte
33
34 def bitsToInt(inp):
35     return int("".join(str(i) for i in inp), 2)
36
37
38 def pad(inpBits): #https://csrc.nist.gov/csrc/media/publications/fips/180/4/archive/2012-03-06/documents/fips180-4.pdf section 5.1
39     l = len(inpBits)
40     if (l % 512 == 0) and l != 0:
41         return inpBits
42     else:
43         inpBits.append(1) #Add one to the end of the message
44         # 448%512 = k + l + 1
45         #k = 448-(l+1)
46         k = 448-(l+1)
47         for i in range(k):
48             inpBits.append(0)
49         #Pad with message length expressed as 64 bit binary number
50         lengthBits = intToBits(l, 64)
51         for x in lengthBits:
52             inpBits.append(x)
53     return inpBits
54
55
56 def checkLessThan32(num): # Used for getting the index to move the element in an array in RotR
57     if num < 32:
58         return num
59     else:
60         return num - 32
61
62 def checkShiftInBounds(word, num): # Similar to checkLessThan32, however it is for shifting.
63     if (num < 0) or (num >= 32):
64         return 0
65     else:
66         return word[num]
67
68
69 def notArray(array, l=32):
70     temp = []
71     for x in range(l):
72         temp.append(0)
73     for i in range(l):

```

```

74     if array[i] == 1:
75         temp[i] = 0
76     else:
77         temp[i] = 1
78     return temp
79
80 def xorArrays(array1, array2): # XORs two arrays
81     temp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
82     for i in range(32):
83         temp[i] = array1[i] ^ array2[i]
84     return temp
85
86 def andBitArrays(array1, array2): # Does AND on two arrays
87     temp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
88     for i in range(32):
89         temp[i] = array1[i] & array2[i]
90     return temp
91
92 def RotR(word, amount):
93     temp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
#32Bits
94
95     for i in range(32):
96         temp[i] = word[checkLessThan32(i-amount)]
97     return temp
98
99 def addMod2W(array1, array2, W=32): # Adds % 2^W two arrays, so that the word does not overflow it's word
length
100    if len(array1) != len(array2):
101        raise IndexError("Arrays not same size - ", array1, array2)
102    return intToBits((bitsToInt(array1) + bitsToInt(array2)) % 2**W, 32)
103
104 def ShR(x, n):
105     temp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
106
107     for i in range(32):
108         temp[i] = checkShiftInBounds(x, i-n)
109     return temp
110
111 def SigExpansion0(x):
112     return xorArrays(xorArrays(RotR(x, 7), RotR(x, 18)), ShR(x, 3))
113
114 def SigExpansion1(x):
115     return xorArrays(xorArrays(RotR(x, 17), RotR(x, 19)), ShR(x, 10))
116
117 def Sig0(x):
118     return xorArrays(xorArrays(RotR(x, 2), RotR(x, 13)), RotR(x, 22))
119
120 def Sig1(x):
121     return xorArrays(xorArrays(RotR(x, 6), RotR(x, 11)), RotR(x, 25))
122
123 def Ch(x, y, z):
124     return xorArrays(andBitArrays(x, y), andBitArrays(notArray(x), z))
125
126 def Maj(x, y, z):
127     return xorArrays(xorArrays(andBitArrays(x, y), andBitArrays(x, z)), andBitArrays(y, z))
128
129 def sha256(inp):
130     #Initial hash values - https://csrc.nist.gov/csrc/media/publications/fips/180/4/archive/2012-03-
06/documents/fips180-4.pdf section 5.3.3
131     hList = [0x6a09e667, # H0
132             0xbb67ae85, # H1
133             0x3c6ef372, # H2
134             0xa54ff53a, # H3
135             0x510e527f, # H4
136             0x9b05688c, # H5
137             0x1f83d9ab, # H6
138             0x5be0cd19] # H7
139

```

```

140     bits = makeBitArray(inp)
141     bits = pad(bits)
142     bits = [bits[x:x+32] for x in range(0, len(bits), 32)] #Split padded message into 32 bit words
143     bits = bits+[[], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], [], []]
144
145     #Main part
146
147     for x in range(16, 64): #Expand current bits to be 64 words
148         bits[x] = addMod2W(addMod2W(addMod2W(bits[x-16], SigExpansion0(bits[x-15])), bits[x-7]),
149         SigExpansion1(bits[x-2]))
150
151         a = intToBits(hList[0], 32)
152         b = intToBits(hList[1], 32)
153         c = intToBits(hList[2], 32)
154         d = intToBits(hList[3], 32)
155         e = intToBits(hList[4], 32)
156         f = intToBits(hList[5], 32)
157         g = intToBits(hList[6], 32)
158         h = intToBits(hList[7], 32)
159
160         for i in range(64):
161             temp1 = addMod2W(addMod2W(addMod2W(h, Sigl(e)), Ch(e, f, g)), intToBits(k[i], 32)), bits[i])
162             S0 = Sig0(a)
163             maj = Maj(a, b, c)
164
165             h = g
166             g = f
167             f = e
168             e = addMod2W(d, temp1)
169             d = c
170             c = b
171             b = a
172             a = addMod2W(temp1, addMod2W(S0, maj))
173
174         resultBits = addMod2W(intToBits(hList[0], 32), a)+addMod2W(intToBits(hList[1], 32),
175         b)+addMod2W(intToBits(hList[2], 32), c)+addMod2W(intToBits(hList[3], 32), d)+addMod2W(intToBits(hList[4], 32),
176         e)+addMod2W(intToBits(hList[5], 32), f)+addMod2W(intToBits(hList[6], 32), g)+addMod2W(intToBits(hList[7], 32),
177         h)
178         # Looks really ugly but works better (otherwise I would have to store each in variables)
179
180     resultBytes = [resultBits[x:x+8] for x in range(0, len(resultBits), 8)] # Makes 2D array of bytes
181     result = []
182     for byte in resultBytes:
183         result.append(bitsToInt(byte)) # Converts each byte into an integer
184     return result
185
186 def getSHA128of16(data):
187     out = sha256(data)
188     return [out[i]^out[i+16] for i in range(16)]

```

Each byte is made into an array of bits. Doing it this way made it easier to debug, however probably made the algorithm much slower than it needed to be. However, I don't really care too much about how fast SHA is, as it is only used a few times in the program, and only ever works on very small amounts of data, so it will probably be unnoticeable for the user.

The file is called `SHA.py`, and is imported by `LoginScreen` (default login without Bluetooth), which is in `code/kivyStuff/loginClass.py`, for use when the key is entered.

BLAKE2b:

Unlike AES, BLAKE does not have any sub-packages (like `AESfiles`). Here is the file structure of BLAKE:

```

1 code/python-go/BLAKE/
2     └── BLAKE
3         ├── BLAKE
4         ├── BLAKE.test
5         ├── build.sh
6         ├── main.go
7         └── src
8             └── BLAKE
9                 ├── blake.go
10                ├── blake_test.go
11                └── checksum.go
12            └── testBenchBLAKE.sh
13            └── testBLAKE.sh
14
15 2 directories, 9 files

```

It consists of two packages, `BLAKE` and `main`. `main` just communicates between Go and Python (like in AES). Here is the content of `code/python-go/BLAKE/main.go`:

```

1 package main
2
3 import (
4     "fmt"
5     "os"
6     "io/ioutil"
7     "BLAKE"
8 )
9
10 func main() {
11     bytes, err := ioutil.ReadAll(os.Stdin) // Read file to hash from stdin
12     if err != nil { panic(err) }
13     f := string(bytes)
14
15     fmt.Printf("%x", BLAKE.GetChecksum(f, 64))
16 }

```

It is a small package, as `BLAKE` only accepts a file location, then returns the checksum of that file. Line 15 `fmt.Printf("%x", BLAKE.GetChecksum(f, 64))` gets the hex representation of the bytes returned by `GetChecksum`, and returns it on `stdout`, back to Python.

Here is the main part of the `BLAKE` package (`code/python-go/BLAKE/src/BLAKE/blake.go`):

```

1 package BLAKE
2
3 // Initial constants.
4 var k = [8]uint64 {0x6A09E667F3BCC908,
5                     0xBB67AE8584CAA73B,
6                     0x3C6EF372FE94F82B,
7                     0xA54FF53A5F1D36F1,
8                     0x510E527FADE682D1,
9                     0x9B05688C2B3E6C1F,
10                    0x1F83D9ABFB41BD6B,
11                    0x5BE0CD19137E2179}
12
13 var sigma = [12][16]uint64 {{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
14                             {14, 10, 4, 8, 9, 15, 13, 6, 1, 12, 0, 2, 11, 7, 5, 3},
15                             {11, 8, 12, 0, 5, 2, 15, 13, 10, 14, 3, 6, 7, 1, 9, 4},
16                             {7, 9, 3, 1, 13, 12, 11, 14, 2, 6, 5, 10, 4, 0, 15, 8},
17                             {9, 0, 5, 7, 2, 4, 10, 15, 14, 1, 11, 12, 6, 8, 3, 13},
18                             {2, 12, 6, 10, 0, 11, 8, 3, 4, 13, 7, 5, 15, 14, 1, 9},
19                             {12, 5, 1, 15, 14, 13, 4, 10, 0, 7, 6, 3, 9, 2, 8, 11},
20                             {13, 11, 7, 14, 12, 1, 3, 9, 5, 0, 15, 4, 8, 6, 2, 10},
21                             {6, 15, 14, 9, 11, 3, 0, 8, 12, 2, 13, 7, 1, 4, 10, 5},
22                             {10, 2, 8, 4, 7, 6, 1, 5, 15, 11, 9, 14, 3, 12, 13, 0},
23                             {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}, // Same as first line
24}

```

```

24           {14, 10, 4, 8, 9, 15, 13, 6, 1, 12, 0, 2, 11, 7, 5, 3} } // Same as second line
25
26
27 func rotR64(in uint64, n int) uint64 { // For 64 bit words
28     return (in >> uint(n)) ^ (in << (64 - uint(n)))
29 }
30
31 func get64(in []uint64) uint64 { // Gets a full 64-bit word from a list of 8 64-bit bytes.
32     return uint64(in[0] ^ (in[1] << 8) ^ (in[2] << 16) ^ (in[3] << 24) ^ (in[4] << 32) ^ (in[5] << 40) ^ (in[6]
33     << 48) ^ (in[7] << 56))
34 }
35
36 func blakeMix(v []uint64, a, b, c, d int, x, y *uint64) {
37     v[a] = v[a] + v[b] + *x
38     v[d] = rotR64((v[d] ^ v[a]), 32)
39
40     v[c] = v[c] + v[d]
41     v[b] = rotR64((v[b] ^ v[c]), 24)
42
43     v[a] = v[a] + v[b] + *y
44     v[d] = rotR64((v[d] ^ v[a]), 16)
45
46     v[c] = v[c] + v[d]
47     v[b] = rotR64((v[b] ^ v[c]), 63)
48 }
49
50 func BlakeCompress(h *[8]uint64, block []uint64, t int, lastBlock bool) { // Compressing function. Takes a
51     block of 128 uint64s
52     v := make([]uint64, 16) // Current vector as a slice. This allows you to pass by reference
53
54     v[ 0], v[ 1], v[ 2], v[ 3],      // Doing this instead of for loop allows for marginal performance increase.
55     v[ 4], v[ 5], v[ 6], v[ 7],
56     v[ 8], v[ 9], v[10], v[11],
57     v[12], v[13], v[14], v[15] =
58
58     h[ 0], h[ 1], h[ 2], h[ 3],
59     h[ 4], h[ 5], h[ 6], h[ 7],
60     k[ 0], k[ 1], k[ 2], k[ 3],
61     k[ 4], k[ 5], k[ 6], k[ 7]
62
62     v[12] ^= uint64(t)
63     v[13] ^= (uint64(t) >> 64)
64
65     if lastBlock {
66         v[14] = ^v[14] // NOT
67     }
68
69     var m [16]uint64
70     for i, z := 0, 0; i < 121; i, z = i+8, z+1 { // Having z prevents having to divide by 8
71         m[z] = get64(block[i:i+8])
72     }
73
74     for i := 0; i < 12; i++ {
75         blakeMix(v, 0, 4, 8, 12, &m[ sigma[i][0]], &m[ sigma[i][1]])
76         blakeMix(v, 1, 5, 9, 13, &m[ sigma[i][2]], &m[ sigma[i][3]])
77         blakeMix(v, 2, 6, 10, 14, &m[ sigma[i][4]], &m[ sigma[i][5]])
78         blakeMix(v, 3, 7, 11, 15, &m[ sigma[i][6]], &m[ sigma[i][7]])
79
80         blakeMix(v, 0, 5, 10, 15, &m[ sigma[i][ 8]], &m[ sigma[i][ 9]]) // Rows have been shifted
81         blakeMix(v, 1, 6, 11, 12, &m[ sigma[i][10]], &m[ sigma[i][11]])
82         blakeMix(v, 2, 7, 8, 13, &m[ sigma[i][12]], &m[ sigma[i][13]])
83         blakeMix(v, 3, 4, 9, 14, &m[ sigma[i][14]], &m[ sigma[i][15]])
84     }
85
86     for i := 0; i < 8; i++ {
87         h[i] ^= v[i]
88         h[i] ^= v[i+8]
89     }
90 }
```

`get64` takes 8 bytes, and sticks them together to return a single 64-bit word. I know `get64` takes an array of 64-bit words `[]uint64`, however these are read from the file as bytes, and are changed into 64-bit words, but they are bytes so none of them are bigger than 255. Here is a small example:

```
1 | 8 bytes: 0xaa 0xbb 0xcc 0xdd 0xee 0xff 0x1e 0x9d
2 | get64: 0xaabbccddeeff1e9d
```

`blakeMix` accepts the reference to the current working vector `v` (since it is a slice it is automatically passed by reference), the elements in `v` to change, and also a reference to tow elements from the message `m`, which is the block that has been converted using the `get64` method explained above.

`h` is the current hash that is being worked on, and is also passed by reference to the compression function. Passing by reference increases speed about twofold compared to not passing by reference, as the function is called a lot it is more efficient to change one copy of the variable, rather than copying it and returning it every time the function is called.

Here is the part of the `BLAKE` package that handles getting the checksum of a file (`code/python-go/BLAKE/src/BLAKE/checksum.go`):

```
1 | package BLAKE
2 |
3 | import (
4 |     "os"
5 |     "io"
6 | )
7 |
8 | func check(e error) {      //Used for checking errors when reading/writing to files.
9 |     if e != nil {
10 |         panic(e)
11 |     }
12 | }
13 |
14 | func GetChecksum(f string, hashL int) [64]byte {
15 |     h := k // Initialize h0-7 with initial values.
16 |     h[0] = h[0] ^ (0x01010000 ^ uint64(hashL)) // Not using a key
17 |
18 |     a, err := os.Open(f)    // Open file
19 |     check(err)
20 |     aInfo, err := a.Stat() // Get statistics of file
21 |     check(err)
22 |
23 |     fileSize := int(aInfo.Size()) // Get size of original file
24 |
25 |     var bufferSize int = 65536
26 |
27 |     if fileSize < bufferSize {    // If the buffer size is larger than the file size, just read the whole file.
28 |         bufferSize = fileSize
29 |     }
30 |
31 |     var buffCount int = 0    // Keeps track of how far through the file we are
32 |     var bytesFed int = 0
33 |     var bytesLeft int = fileSize
34 |
35 |     for buffCount < fileSize {
36 |         if bufferSize > (fileSize - buffCount) {
37 |             bufferSize = fileSize - buffCount
38 |         }
39 |         buff := make([]uint64, bufferSize)
40 |         tempBuff := make([]byte, bufferSize) // Make a slice the size of the buffer
41 |         _, err := io.ReadFull(a, tempBuff) // Read the contents of the original file, but only enough to fill the
buff array.
```

```

42                                     // The "_" tells go to ignore the value returned by io.ReadFull, which in
43                                     this case is the number of bytes read.
44         check(err)
45         for i := range tempBuff {
46             buff[i] = uint64(tempBuff[i])
47         }
48         tempBuff = nil // Delete array
49
50         for len(buff) % 128 != 0 {
51             buff = append(buff, 0) // Append 0s when buffer is not long enough
52         }
53
54         for i := 0; i < bufferSize; i += 128 {
55             if bytesLeft <= 128 {
56                 BlakeCompress(&h, buff[i:i+128], bytesFed+bytesLeft, true)
57             } else {
58                 bytesFed += 128
59                 BlakeCompress(&h, buff[i:i+128], bytesFed, false)
60             }
61             bytesLeft -= 128
62         }
63
64         buffCount += bufferSize
65     }
66     a.Close()
67
68     return getLittleEndian(h)
69 }
70
71 func getLittleEndian(h [8]uint64) [64]byte {
72     var out [64]byte
73     for i := 0; i < 8; i++ {
74         for j := 8; j != 0; j-- {
75             out[i*8+(j-1)] = byte(((h[i] << uint64(64 - uint64((j)*8))) & 0xFFFFFFFFFFFFFF) >> 56)
76         }
77     }
78     return out
79 }
```

The way that the `GetChecksum` function goes through the file is very similar to AES, however it cannot be easily parallelised like AES, as the checksum takes into account the order of the data. There is a way to make BLAKE parallel but I didn't really have the time to look into it.

`getLittleEndian` turns the array `h`, which contains 8 64-bit words, into an array of 64 bytes, that can then be turned into a hex output that is a bit more readable. The way it works is it generates a little-endian interpretation of the 64-bit words as bytes. So if I had the word \$0D4D1C983FA580BA\$, the output of the function would return \$BA 80 A5 3F 98 1C 4D 0D\$. The function `getLittleEndian` uses bit masking (shifting the bits in the word around to leave the bits you want to change exposed) to get each byte of the 64-bit word, then appends the byte to the list in reverse order (since it is little-endian). Little-endian is just a way to store a number larger than a byte. Little-endian and big-endian are needed in computer systems because in memory, each address can only store a single byte, so if a number is bigger than that then the number needs to be split into separate bytes. For example, if I had the number 354, then I would first convert that into binary: \$2+32+64+256 = 354\$, = \$101100010\$, however this is larger than 8 bits, so split it into two:

\$00000001\$ and \$01100010\$, where the first byte starts at \$2^8\$. Little-endian arranges these bytes in memory like this:

Address1: \$01100010\$, Address2: \$00000001\$

It is called little-endian because the smaller part of the number (little) number is stored in the first address (the end). Big-endian is just the opposite way around.

`build.sh` just compiles the main code. Here it is:

```
1  #!/bin/bash
2  export GOPATH="/home/josh/nea-12ColcloughJ/code/python-go/BLAKE/"
3  go build
```

The Sorts:

The `sorts` package is in the AES folder, as it is used a lot by AES, and the sorts are managed from AES. Here is the code (`code/python-go/AES/src/sorts/sorts.go`):

```
1  package sorts
2
3  import (
4      "fmt"
5      "os"
6      "strconv"
7  )
8
9  // For keeping the decrypted name with their encrypted name
10 type Tuple struct {
11     A os.FileInfo
12     B string
13 }
14
15 type SearchTuple struct {
16     pos int
17     name string
18 }
19
20 func UseQuickSortSize(inp []string) []string { // Converts inputs so that QuickSortSize can be used.
21     var nums []int64
22     var out []string
23     for i := 0; i < len(inp); i++ {
24         int, err := strconv.ParseInt(inp[i], 10, 64)
25         if err != nil { panic(err) }
26         nums = append(nums, int)
27     }
28     nums = quickSort(nums)
29     for i := 0; i < len(nums); i++ {
30         out = append(out, strconv.FormatInt(nums[i], 10))
31     }
32     return out
33 }
34
35 func UseQuickSortAlph(inp []string) []string { // For sorting a list that has no encrypted name
36     var inpToAlph []Tuple
37     var out []string
38     for i := 0; i < len(inp); i++ {
39         inpToAlph = append(inpToAlph, Tuple{A: nil, B: inp[i]})
40     }
41
42     inpToAlph = QuickSortAlph(inpToAlph)
43     for i := 0; i < len(inpToAlph); i++ {
44         out = append(out, inpToAlph[i].B)
45     }
46     return out
47 }
48
49 func UseQuickSortSearch(posList, nameList []string) []string { // Returns names in order.
50     var inpToSort []SearchTuple
51     var out []string
52     if len(posList) != len(nameList) {
53         panic("Search result lists are different sizes.")
54     }
```

```

55     for i := 0; i < len(posList); i++ {
56         intPos, err := strconv.Atoi(posList[i])
57         if err != nil { panic(err) }
58         inpToSort = append(inpToSort, SearchTuple{pos: intPos, name: nameList[i]})  

59     }
60     inpToSort = quickSortSearch(inpToSort)
61     for i := 0; i < len(inpToSort); i++ {
62         out = append(out, inpToSort[i].name)
63     }
64     return out
65 }
66
67 func quickSort(inp []int64) []int64 {
68     if len(inp) < 2 {
69         return inp
70     }
71     var pivot int64 = inp[int(len(inp)/2)]
72     var left []int64
73     var middle []int64
74     var right []int64
75     for i := 0; i < len(inp); i++ {
76         if inp[i] < pivot {
77             left = append(left, inp[i])
78         } else if inp[i] > pivot {
79             right = append(right, inp[i])
80         } else {
81             middle = append(middle, inp[i])
82         }
83     }
84     left = quickSort(left)
85     right = quickSort(right)
86     return append(append(left, middle...), right...)
87 }
88
89 func getLower(inp []byte) []byte { // .lower() in python
90     var out []byte
91     for i := range inp {
92         out = append(out, inp[i])
93         if out[i] >= 65 && out[i] <= 90 {
94             out[i] += 32 // Using ASCII table
95         }
96     }
97     return out
98 }
99
100 func QuickSortAlph(inp []Tuple) []Tuple { // Only used internally by AES
101     if len(inp) < 2 {
102         return inp
103     }
104     var pivot Tuple = inp[int(len(inp)/2)]
105     var left []Tuple
106     var middle []Tuple
107     var right []Tuple
108     for i := 0; i < len(inp); i++ {
109         result := compareStrings(pivot.B, inp[i].B)
110         if result == 0 {
111             right = append(right, inp[i])
112         } else if result == 1 {
113             left = append(left, inp[i])
114         } else if result == 2 {
115             middle = append(middle, inp[i])
116         }
117     }
118     left = QuickSortAlph(left)
119     right = QuickSortAlph(right)
120     return append(append(left, middle...), right...)
121 }
122
123 func quickSortSearch(inp []SearchTuple) []SearchTuple { // Sorts search results

```

```

124     if len(inp) < 2 {
125         return inp
126     }
127     var pivot int = inp[int(len(inp)/2)].pos
128     var left []SearchTuple
129     var middle []SearchTuple
130     var right []SearchTuple
131     for i := 0; i < len(inp); i++ {
132         if inp[i].pos < pivot {
133             left = append(left, inp[i])
134         } else if inp[i].pos > pivot {
135             right = append(right, inp[i])
136         } else {
137             middle = append(middle, inp[i])
138         }
139     }
140     left = quickSortSearch(left)
141     right = quickSortSearch(right)
142     return append	append(left, middle...), right...
143 }
144
145 func compareStrings(string1, string2 string) int {
146     if string1 == string2 {
147         return 2
148     }
149     string1b, string2b := []byte(string1), []byte(string2) // Get the ascii values in bytes
150     string1bLower, string2bLower := getLower(string1b), getLower(string2b) // Get each string as lower case
151
152     for i := 0; i < len(string1) && i < len(string2); i++ {
153         if string2bLower[i] < string1bLower[i] {
154             return 1
155         } else if string2bLower[i] > string1bLower[i] {
156             return 0
157         } else { // If the characters are both the same, then compare if they
158             are lower case or upper case.
159             if string2b[i] < string1b[i] {
160                 return 1
161             } else if string2b[i] > string1b[i] {
162                 return 0
163             }
164         }
165         if len(string1) > len(string2) { // If they are the exact same to a certain point, then compare their
166             lengths
167             return 1
168         } else if len(string1) < len(string2) {
169             return 0
170         } else {
171             panic("Strings are the exact same!")
172         }
173     }
174     func main() {
175         fmt.Println("a")
176     }

```

The `UseQuickSort` functions are used for formatting input from `stdin` that Python has given us.

`QuickSortAlpha` is used by `AESstring`

`quickSortSearch` is used for sorting search results, as search results are collected along with the position that the search item was found in the word. For example, if I searched for "b" in a folder, and there was a file called "brian.png", then the search result would be (0, "brian.png"). `quickSortSearch` then sorts these results by the number. I need to use a tuple so that I know what string belongs to which number.

`compareStrings` starts at the first character of each string, compares the characters using `ord()` to get their ASCII value, if character1 has a bigger ASCII value than character2, then the function will return `1`, if character1 is less than character2, then the function will return `0`. If they are both the same, then the function moves onto the next pair of characters. If both strings turn out to be exactly the same, then the function returns `2`.

`QuickSortAlpha` uses this output to determine which side of the pivot the item should be added to. If the output of the function was `2`, then the item is the search item, so add it to the middle. If the output of the function was `1` then append it to the left side of the list, and if the returned value was `0` then append it to the right. All of the quick sorts always sort in ascending order, and then if the program wants it in descending order, then all you have to do is reverse the list (`list = list[::-1]`).

The File class:

Here is the code for the File class (`code/python-go/fileClass.py`), often assigned the variable name `fileObj` in the rest of the program:

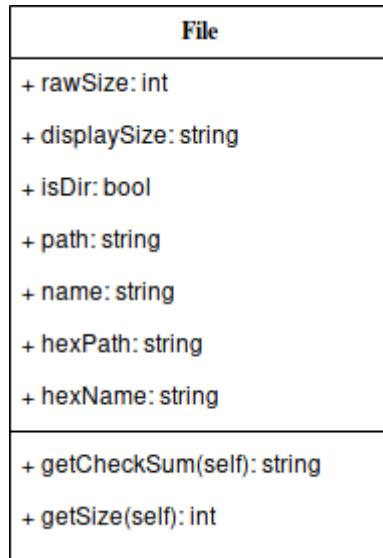
```
1  from os import path as osPath
2  from os import listdir
3  from subprocess import Popen, PIPE
4
5  class File:
6
7      def __init__(self, screen, hexPath, hexName, fileSep, extension=None, isDir=False, name=None, path=None):
8          self.outerScreen = screen
9          self._totalSize = 0
10         self.hexPath, self.hexName, self.isDir, self.fileSep, self.extension = hexPath, hexName, isDir,
11         fileSep, extension
12         self.thumbDir = ""
13         self.checkSum = None
14         self.rawSize = self.__getFileSize()
15         self.size = self.outerScreen.getGoodUnit(self.rawSize)
16         self.isDirectory = isDir
17         if name == None:
18             self.name = self.outerScreen.decString(self.hexName)
19         else:
20             self.name = name
21         if path == None:
22             self.path = self.__getNormDir(self.hexPath)
23         else:
24             self.path = path
25         if extension == None:
26             extension = self.path.split(".")
27             self.extension = extension[-1].lower()
28         else:
29             extension = extension.lower()
30
31         if self.isDirectory:
32             self.hexPath += self.fileSep
33             self.path += self.fileSep
34
35         self.relPath = self.hexPath.replace(self.outerScreen.path, "") # Encrypted path relative to root
36         folder of Vault
37
38     def __getNormDir(self, hexDir):          # Private functions as they are usually only needed once and
39         should only be callable from within the class
40         dir = self.fileSep.join(self.outerScreen.decListString(hexDir.replace(self.outerScreen.path,
41             "").split(self.fileSep)[: -1])) + self.name
42         if self.isDirectory:
43             dir += self.fileSep
44         return dir
```

```

42     def __getFileSize(self, recurse=True):
43         if self.isDir:
44             if recurse:
45                 self._totalSize = 0
46                 self.__recursiveSize(self.hexPath)
47                 size = self._totalSize
48                 return size
49             else:
50                 return " -"
51         else:
52             try:
53                 size = osPath.getsize(self.hexPath) # Imported from os module
54                 return size
55             except Exception as e:
56                 print(e, "couldn't get size.")
57                 return " -"
58
59     def __recursiveSize(self, f): #Get size of folders.
60         fs = listdir(f)
61         for item in fs:
62             if osPath.isdir(f+self.fileSep+item):
63                 try:
64                     self.__recursiveSize(f+self.fileSep+item)
65                 except OSError:
66                     pass
67             else:
68                 try:
69                     self._totalSize += osPath.getsize(f+self.fileSep+item)
70                 except PermissionError: #Thrown when the file is owned by another user/administrator.
71                     pass
72
73     def decryptRelPath(self):      # Gets relative path from root of Vault in human form
74         splitPath = self.relPath.split(self.fileSep)
75         return self.fileSep.join(self.outerScreen.decListString(splitPath))

```

This is the File class talked about in the **File Storage** section of the design. As a recap, here is the class diagram I made for this class:



Most of the variables have been kept the same, however `extension` was added for when I get the thumbnail of the file, as if the file is not a png or a jpg, then a thumbnail cannot be shown (since it isn't an image). I also have the variable `outerScreen` that holds a reference to the Kivy Screen object that created it, so it can access functions and variables from the Screen if it needs to.

There are a few new functions too. `__getNormDir` gets the normal file path if path is `None` (it is also a private function (`_`), as it is only needed once by the object, and shouldn't be used again by anything else).

`__getFileSize` gets the total size of the File object. If it is a folder (isDir) then `__recursiveSize` is called to handle it. If the size can not be read, then the function returns " - ", which will display nicely in the GUI. `decryptRelPath` decrypts the path of the file relative to the Vault's root directory.

`decryptRelPath` decrypts the relative path from the Vault so it can be read by humans.

PC app GUI Code

In this section, I will go through the code for the entire GUI (basically anything in the `code/python-go/kivyStuff` folder).

The root of the GUI

The GUI is started once `code/python-go/kivyStuff/ui.py`'s `runUI()` function is called from `code/python-go/start.py`. Here is the code for `ui.py`:

```
1  from tempfile import gettempdir
2  from shutil import rmtree
3
4  from kivy.config import Config
5  Config.set("graphics", "resizable", True)
6  Config.set("graphics", "width", "1000") # Set default window size
7  Config.set("graphics", "height", "600")
8  Config.set("input", "mouse", "mouse,disable_multitouch") # Disable multitouch features used on mobile apps.
9  Config.write()
10
11 from kivy.app import App
12 from kivy.uix.screenmanager import ScreenManager, Screen, FadeTransition
13 from kivy.lang import Builder
14
15 #####Import personal classes#####
16 from mainScClass import MainScreen
17 from loginClass import LoginScreen, LoginScreenBT
18 from settingsScreen import SettingsScreen
19
20 #####Import config functions#####
21 import configOperations
22
23
24 def runUI():
25     ui = uiApp(title="FileMate")
26     ui.run()
27
28     # When program closes:
29     print("Deleting temp files.")
30     try:
31         fSep = configOperations.getFileSep()
32         rmtree(gettempdir()+fSep+"FileMate"+fSep) # Remove all temporary files.
33     except FileNotFoundError:
34         print("No temp files.")
35     print("App closed.")
36
37 class uiApp(App):
38
39     def build(self):
40         sm = ScreenManager()
41
42         sm.transition = FadeTransition() # Set transition animation when changing screens.
```

```

43     fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc =
44     configOperations.runConfigOperations()
45     # Load kv files for each screen.
46     Builder.load_file(startDir+"kivyStuff/kvFiles/mainSc.kv")      # MainScreen styling.
47     Builder.load_file(startDir+"kivyStuff/kvFiles/settingsSc.kv") # SettingsScreen styling.
48
49     if useBT:
50         Builder.load_file(startDir+"kivyStuff/kvFiles/loginScBT.kv")
51         sm.add_widget(LoginScreenBT(fileSep, path, startDir, name="Login"))
52     else:
53         Builder.load_file(startDir+"kivyStuff/kvFiles/loginSc.kv")
54         sm.add_widget(LoginScreen(fileSep, path, startDir, name="Login"))
55
56     sm.add_widget(MainScreen(fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc,
57     name="Main")) # fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, **kwargs
58     sm.add_widget(SettingsScreen(sm.get_screen("Main"), configLoc, name="Settings"))
59     sm.current = "Login"
60
61     return sm

```

This is the program that runs the app itself. All it does is create the root App, and add the ScreenManager as the root widget, where then child widgets (in this case screens) can be added.

This program is called from `code/python-go/start.py`, which is used to start the program. This file is also handy because it means that when I call `import <module>` in the `code/python-go/kivyStuff/` folder, the Python programs can import Python programs from `code/python-go/`, which is where everything non-gui related is kept.

Login classes

Here is the code for both the regular login (LoginScreen), and the Bluetooth login screen (LoginScreenBT) (at `code/python-go/kivyStuff/loginClass.py`):

```

1  from os import listdir
2  from os.path import isdir as osIsDir
3  from subprocess import Popen, PIPE
4
5  from kivy.uix.screenmanager import Screen
6  from kivy.lang.builder import Builder
7  from kivy.uix.popup import Popup
8  from kivy.uix.label import Label
9
10 from kivy.clock import Clock
11 from threading import Thread
12
13 import SHA
14
15 # Try importing the BT module, if it isn't available then they just can't use BT. Imported in case the user
16 # wants to switch from normal login to Bluetooth login.
17 try:
18     from bluetooth import *
19 except:
20     pass
21
22 class LoginScreen(Screen):
23
24     def __init__(self, fileSep, path, startDir, **kwargs):
25         self.fileSep, self.path, self.startDir = fileSep, path, startDir # Start dir is location of running
26         program, path is path of vault
27         super(Screen, self).__init__(**kwargs) # Run kivy's Screen.__init__ function with the key word
28         arguments (such as size or position)
29         self.key = ""

```

```

28     def cancel(self):
29         self.manager.get_screen("Main").useBT = True # Am now using BT
30         Builder.load_file(self.startDir+"kivyStuff/kvFiles/loginScBT.kv") # Load the styling file for BT login
31         screen
32             self.manager.add_widget(LoginScreenBT(self.fileSep, self.path, self.startDir, name="Login")) # Create
33             the new screen
34             self.name = "Dead" # To prevent clash with new login screen.
35             self.manager.current = "Login" # Change to Login
36             self.manager.remove_widget(self) # Remove self from the app
37             self = None # Kill self
38
39     def findFile(self, dir): # For finding a file to decrypt first block and compare it with key given.
40         fs = listdir(dir)
41         for item in fs:
42             if osIsDir(dir+item+"/"):
43                 if self.count == 0:
44                     self.findFile(dir+item+"/")
45                 else:
46                     return
47             else:
48                 self.decryptTestFile = dir+item
49                 self.count += 1
50             return
51
52     def passToTerm(self, key, d): # Makes a pipe to communicate with AES
53         if self.fileSep == "\\":
54             programe = "AESWin"
55         else:
56             programe = "AES/AES"
57         goproc = Popen(self.startDir+programe, stdin=PIPE, stdout=PIPE)
58         out, err = goproc.communicate(("test, "+d+", 0, ").encode()+key.encode())
59         return out
60
61     def getIfValidKey(self, inputKey): # Gets the output of the AES key checker.
62         if len(listdir(self.path)) > 1:
63             self.decryptTestFile = ""
64             self.count = 0
65             self.findFile(self.path)
66             diditwork = self.passToTerm(inputKey, self.decryptTestFile)
67             if diditwork == b"-Valid-": #The go program prints "-Valid-" or "-Invalid-" once it is done
68             checking the key.
69                 return True
70             else:
71                 return False
72         else:
73             return True
74
75     def checkKey(self, inputKey): # Handles the GUI while the key is checked, and passes key to functions to
76     check it.
77         if len(inputKey) < 1:
78             Popup(title="Invalid", content=Label(text="Invalid key, valid key\ncontains at least 1 digit."),
79             pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
80             return "Login"
81         try:
82             int(inputKey)
83         except:
84             Popup(title="Invalid", content=Label(text="Invalid key, valid key\ncontains no letters."),
85             pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
86             return "Login"
87         else:
88             if len(str(inputKey)) > 16:
89                 Popup(title="Invalid", content=Label(text="Invalid key, longer than\n 16 characters."),
90                 pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
91                 return "Login"
92             else:
93                 inputKeyTemp = []
94                 for i in range(len(inputKey)):
95                     inputKeyTemp.append(int(inputKey[i]))
96                 inputKey = inputKeyTemp

```

```

90         inputKey = SHA.getSHA128of16(inputKey)
91         key = " ".join(str(i) for i in inputKey)
92         valid = self.getIfValidKey(key)
93         if valid:
94             self.ids.keyInput.text = "" #reset key input if valid
95             self.key = key
96             return "Main"
97         else:
98             Popup(title="Invalid", content=Label(text="Invalid key."), pos_hint={"x_center": .5,
99 "y_center": .5}, size_hint=(.4, .4)).open()
100            return "Login"
101
102     def needToSetKey(self):      # Gets text to tell the user if they need to set a key.
103         if len(listdir(self.path)) == 0:  # If there are no files in the vault, then the key hasn't been set
104             yet.
105             return "Input New Key (Write this down if you have to)"
106         else:
107             return "Input Key"
108
109
110 class LoginScreenBT(LoginScreen, Screen):      #Has the same methods as LoginScreen, but some overwritten with
111     bluetooth.
112
113     def __init__(self, fileSep, path, startDir, **kwargs):
114         self.fileSep, self.path, self.startDir = fileSep, path, startDir
115         super(Screen, self).__init__(**kwargs)
116         self.key = ""
117
118     def on_enter(self):
119         self.serv = None
120         self.startSrv = Clock.schedule_once(self.startSrv, 0.5) # Use the clock to allow the screen to be
121         rendered. (Waits 0.5 seconds for screen to be loaded.)
122
123     def checkKey(self, inputKey):
124         inputKey = inputKey.split(",")
125         inputKey = inputKey[:-1]
126         key = " ".join(str(i) for i in inputKey)      #Formatting for AES
127         valid = self.getIfValidKey(key)
128         if valid:
129             self.key = key
130             self.manager.get_screen("Main").key = key
131             return True
132         else:
133             return False
134
135     def cancel(self):
136         if self.serv != None:
137             self.manager.get_screen("Main").serverSock.close() # Close the BT server
138             self.serv.join() # Close the thread that runs the server (in LoginScreenBT)
139             try:
140                 self.manager.get_screen("Main").clientSock.close()
141             except AttributeError: # clientSock will not be initialized if there are no clients.
142                 pass
143             else:
144                 self.startSrv.cancel() # Cancels scheduled task to start server, as we are switching screens
145             anyway.
146
147             print("Server closed.")
148             self.manager.get_screen("Main").useBT = False
149             Builder.load_file(self.startDir+"kivyStuff/kvFiles/loginSc.kv")
150             self.manager.add_widget(LoginScreen(self.fileSep, self.path, self.startDir, name="Login"))
151             self.name = "Dead"      # To prevent name clash with other login screen.
152             self.manager.current = "Login"
153             self.manager.remove_widget(self)
154             self = None
155
156     def startSrv(self, dt=None):
157         self.serv = Thread(target=self.manager.get_screen("Main").startBT, daemon=True) # Runs the function
158         in MainScreen, which prevents segmentation, so I don't have to shutdown server when screen is switched

```

```
153 |     self.serv.start()
```

`LoginScreenBT.startSrv` starts the Bluetooth server, however the server is run inside of the `MainScreen` class, which displays the files. This is important because when you change screen and have a thread running, the thread gets cut off and a `Segmentation Fault` is thrown. Running the server in `MainScreen` also means that the server does not need to be closed and opened again (which is what I was doing before I did this).

The function `cancel` is called when the user wants to switch between login screens. What it does is rename the kivy Screen to something other than "Login", load the `.kv` file for the new login screen, then create the new Login screen, and change screen to that one.

`.kv` files are for Kivy's styling language, layed out similar to css. `.kv` files work like this:

```
1 | RootWidget:           # Example: ScreenManager:  
2 |     ChildWidget:  
3 |         key_word_argument: value  
4 |  
5 | <CustomClass@KivyClass>:    # Example: <LoginScreen@Screen>  
6 |     ...
```

For custom classes, you do not always have to specify a `KivyClass`.

Here is the style sheet `kv` file for both login screens:

`LoginScreen` ([code/python-go/kivyStuff/kvFiles/loginSc.kv](#)):

```
1 | <LoginScreen>:  
2 |     RelativeLayout:           #Adds background in case fade transition breaks.  
3 |         canvas.before:  
4 |             Color:  
5 |                 rgba: 0,0,0,1  
6 |             Rectangle:  
7 |                 pos: self.pos  
8 |                 size: self.size  
9 |  
10 |         Label:  
11 |             id: labelLogin  
12 |             size_hint: .46, .08  
13 |             text: root.needToSetKey()  
14 |             font_size: 22  
15 |             pos_hint: {"center_x": 0.5, "y": 0.8}  
16 |  
17 |         TextInput:  
18 |             id: keyInput  
19 |             size_hint: .7, .08  
20 |             font_size: 22  
21 |             hint_text: "Key (16 characters maximum)"  
22 |             pos_hint: {"center_x": 0.5, "center_y": 0.6}  
23 |             password: True  
24 |             multiline: False  
25 |             on_text_validate: root.manager.current = root.checkKey(keyInput.text)  
26 |  
27 |         Button:  
28 |             id: submitKey  
29 |             size_hint: .16, .16  
30 |             font_size: 22  
31 |             text: "Submit"  
32 |             pos_hint: {"center_x": 0.5, "center_y": 0.3}  
33 |             on_release: root.manager.current = root.checkKey(keyInput.text)  
34 |  
35 |         Button:  
36 |             size_hint: .18, .16  
37 |             pos_hint: {"x": 0, "bottom": 1}
```

```

38     text: "Login with BT"
39     font_size: 22
40     on_release: root.cancel()

```

The syntax highlighting on this document may be a bit off, as the closest highlighting language is `cs`. Comments are done using `#`, however in `cs` they are `///`.

LoginScreenBT:

```

1  <LoginScreenBT>:
2      RelativeLayout:           #Adds background in case fade transition breaks.
3          canvas.before:
4              Color:
5                  rgba: 0,0,0,1
6              Rectangle:
7                  pos: self.pos
8                  size: self.size
9
10     Button:
11         size_hint: .18, .16
12         pos_hint: {"x": 0, "bottom": 1}
13         text: "Login without BT"
14         font_size: 22
15         on_release: root.cancel()
16
17     Label:
18         id: labelLogin
19         size_hint: .46, .08
20         text: "Connect via bluetooth."
21         font_size: 22
22         pos_hint: {"center_x": 0.5, "y": 0.8}
23
24     Label:
25         id: clientLabel
26         pos_hint: {"center_x": 0.5, "center_y": 0.5}

```

The `id` field of some of the widgets is used to access that widget from within the Python code. For example, to access the Label with the text "Connect via bluetooth." in LoginScreenBT, you would have to do

```

1  class LoginScreenBT:
2      ...
3      self.ids.labelLogin
4      ...

```

and then from there you can change any attribute of that Label, such as the text (`self.ids.labelLogin.text = "blah"`).

The `RelativeLayout` at the top of each class is only for setting the background to black for both screens. None of the other widgets are children of the `RelativeLayout`. Just thought I should clarify that before moving onto the different types of positioning.

Positioning widgets in Kivy can work using relative positioning, or exact positioning, where exact positioning requires that you put the exact pixel coordinates as the position, while relative positioning takes the width and height of the window and translates it onto a 0 to 1 scale. `x` goes from left to right, 0 to 1, and `y` goes from bottom to top 0 to 1. When setting the `pos_hint` of each widget, there are a few other options than `"x"` and `"y"`, such as `"center_(x/y)"`, which sets the position of the widget relative to its centre, `"top/bottom"` which sets the position relative to the top or bottom of the screen, `"left/right"` which sets the position relative to the left or right of the screen.

`size_hint` also uses the relative layout system (not actual `RelativeLayout`, there are others like `FloatLayout`) to set the size of a widget depending on the size of the screen.

Also, when I reference `root.something`, `root` is the root widget of this child widget, so in this case either `LoginScreen` OR `LoginScreenBT`. `self` refers to the widget itself.

Main Screen

Here is the code for the MainScreen class (`code/python-go/kivyStuff/mainScClass.py`):

```
1 import os
2 from shutil import move, disk_usage, rmtree
3 from threading import Thread
4 from functools import partial # For passing in functions with multiple arguments to widgets/threads
5 from subprocess import Popen, PIPE
6 from time import sleep
7
8 from kivy.uix.scrollview import ScrollView
9 from kivy.uix.gridlayout import GridLayout
10 from kivy.uix.boxlayout import BoxLayout
11 from kivy.uix.label import Label
12 from kivy.clock import Clock
13 from kivy.clock import mainthread
14 from kivy.core.window import Window
15 from kivy.uix.button import Button
16 from kivy.uix.progressbar import ProgressBar
17 from kivy.uix.popup import Popup
18 from kivy.uix.image import Image
19 from kivy.uix.screenmanager import Screen
20
21 from fileClass import File
22 # Own kivy classes
23 import mainBtns
24 from settingsScreen import SettingsScreen
25 import mainSmallPops as mainSPops
26
27 try:
28     from bluetooth import *
29 except:
30     pass
31
32 class MainScreen(Screen):
33
34     class infoLabel(Label): # Not a popup or button so only suitable place.
35         pass
36
37     def __init__(self, fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc,
38      *kwargs):
39         self.fileSep, self.osTemp, self.startDir, self.assetsPath, self.path, self.searchRecursively,
40         self.useBT, self.configLoc = fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc
41         super(Screen, self).__init__(**kwargs)
42         self.ascending = True # Sort order
43         self.key = ""
44         self.encPop = None
45         self.entered = False
46         self.validBTKey = False
47         self.useBTTemp = self.useBT
48         self.previousDir = None
49         self.lastPathSent = ""
50         self.recycleFolder = ""
51         self.recycleName = ""
52         self.thumbsName = ""
53
54         Window.bind(on_dropfile=self.onFileDrop) #Binding the function to execute when a file is dropped
55         into the window.
```

```

53     self.currentDir = self.path
54
55
56     def on_enter(self): # When the screen is started.
57         self.key = self.manager.get_screen("Login").key # Fetch the key from the Login Screen.
58         if not self.entered:
59             self.setupSortButtons() #Put sort buttons in place.
60             [self.recycleName, self.thumbsName] = self.enclistString([".$recycling", ".$thumbs"]) # Prepare
61             recycling and thumbnail folder names for use in the program.
62             self.recycleFolder = self.path+self.recycleName+self.fileSep
63
64             if not os.path.exists(self.recycleFolder):
65                 print("Recycling folder not found in directory, making one now.")
66                 os.makedirs(self.recycleFolder)
67
68             self.entered = True
69
70             if self.recycleFolder in self.currentDir:
71                 self.createButtons(self.List(self.path), sort=False) # Don't want to log into the recycling
72                 bin, as the user might get confused.
73             else:
74                 self.createButtons(self.List(self.currentDir), sort=False) # Loads previous directory.
75
76     def on_leave(self): # Kept separate from lock because i may want to add more screens that need the
77     key, and do not log the user out.
78         if self.useBT: # Popups that are open block the lock button, but if BT is lost, the popups stay
79         open.
80             try: # Try to close any popups that may be open.
81                 self.largePop.dismiss()
82                 self.remove_widget(self.largePop)
83             except Exception as e:
84                 print(e, "Already closed?")
85             try:
86                 self.smallPop.dismiss()
87                 self.remove_widget(self.smallPop)
88             except Exception as e:
89                 print(e, "Already closed?")
90             try:
91                 self.encPop.dismiss()
92                 self.remove_widget(self.encPop)
93             except Exception as e:
94                 print(e, "Already closed?")
95
96             self.removeButtons()
97
98     def lock(self, fromRunServ=False): # Procedure for when the program is locked. If it has been called from
99     runServMain, then we might still be on login screen, so don't change screen to login, and restart the server.
100         self.clearUpTempFiles() # Delete all temporary files (decrypted files ready for use).
101         if self.useBT:
102             self.manager.get_screen("Login").ids.clientLabel.text = ""
103
104         if fromRunServ and self.validBTKey == False:
105             self.runServMain()
106         else:
107             self.validBTKey = False
108             return mainthread(self.changeToLogin()) #Change screen to the login screen. Ran on mainthread
109             in case it was called in
110
111     def runServMain(self):
112         self.serverSock = BluetoothSocket( RFCOMM )
113         self.serverSock.bind(("" ,PORT_ANY))
114         self.serverSock.listen(1)
115
116         uuid = "80677070-a2f5-11e8-b568-0800200c9a66"
117
118         try:
119             advertise_service(self.serverSock, "FileMateServer",
120                             service_id = uuid,
121                             service_classes = [ uuid, SERIAL_PORT_CLASS ],

```

```

116         profiles = [ SERIAL_PORT_PROFILE ])
117     except BluetoothError as e:
118         Popup(title="Error", content=Label(text="Bluetooth not available.\nPlease make sure your bluetooth
119 is on,\nor change to normal login.\n\nReason: "+str(e)), size_hint=(.4, .4), auto_dismiss=True).open()
120     return
121
122     print("[BT]: Waiting for connection on RFCOMM channel", self.serverSock.getsockname()[1])
123
124     self.clientSock, self.clientInfo = self.serverSock.accept() # Wait for a connection
125     print("[BT]: Accepted connection from ", self.clientInfo)
126     self.manager.get_screen("Login").ids.clientLabel.text = "Connected to: "+str(self.clientInfo[0])
127
128     numbers = []
129     data = ""
130     buff = []
131     backCommand = [33, 66, 65, 67, 75, 33] # !BACK!
132     fileSelectCommand = [33, 70, 73, 76, 69, 83, 69, 76, 69, 67, 84, 33] # !FILESELECT!
133     endHeader = [126, 33, 69, 78, 68, 83, 69, 76, 69, 67, 84, 33] # ~!ENDSELECT!
134
135     try:
136         while len(data) > -1:
137             data = self.clientSock.recv(1024) # Recieve 1kb of data
138             print("[BT]: Received data.")
139             if not self.validBTKey: # If the key is not valid yet, BT server has to wait
140                 for key
141                     numbers.append(str(data, "utf-8"))
142                     if b"~" in data: # End of key message
143                         append = False
144                         tempNums = "".join(numbers)
145                         tempNums = tempNums.replace("#", "")
146                         tempNums = tempNums.replace("~", "")
147                         if self.manager.get_screen("Login").checkKey(tempNums): # Check the key in login.
148                             numbers = []
149                             self.clientSock.send("1")
150                             print("[BT]: Send true.")
151                             self.validBTKey = True
152                             [self.recycleName, self.thumbsName] = self.encListString([".$recycling",
153 ".thumbnails"]) # Set so that file list can be sent
154                             self.sendFileList(self.getListForSend(self.path))
155                             mainthread(self.changeToMain()) # Exit thread and change screen to main.
156                         else:
157                             numbers = []
158                             self.clientSock.send("0")
159                             print("[BT]: Send false.")
160                             self.validBTKey = False
161
162                         else:
163                             for i in data:
164                                 buff.append(i)
165
166                             if buff[:6] == backCommand: # Buffer is reset every time a header is found
167                                 pathBack = self.getPathBack(self.lastPathSent)
168                                 if (not pathBack) or (pathBack.replace(self.path, "") == pathBack): # If you can't
169 go further back (if pathBack has less than path, then remove returns the original string).
170                                 print("[BT]: Can't go further back.")
171                                 self.clientSock.send("!ENDOFTREE!")
172                             else:
173                                 self.sendFileList(self.getListForSend(pathBack))
174                             buff = []
175
176                             elif buff[:12] == fileSelectCommand: # If the command is fileSelect
177                                 commandParams = buff[12:] # Get parameters (buffer will not be reset)
178                                 if commandParams[-12:] == endHeader: # If end of the buffer is the endHeader, then
179 proceed.
180                                 fileWantedList = commandParams[:-12]
181                                 fileWanted = ""
182                                 for letter in fileWantedList:
183                                     fileWanted += chr(letter)

```

```

180             print("[BT]:", fileWanted, "fileWanted")
181             buff = []
182             filesInPath = self.List(self.lastPathSent) # Get list of files at directory
183             requested.
184
185             f = 0
186             fileObj = None
187             while (f < len(filesInPath)) and (fileObj == None): # Searches for the file in the
188                 path
189                     if filesInPath[f].name == fileWanted:
190                         fileObj = filesInPath[f]
191                         f += 1
192
193                     if fileObj != None: # If the file was found, then send it
194                         if fileObj.isdir: # If it was a directory then send the list of files in that
195                             directory.
196
197                             self.sendFileList(self.getListForSend(fileObj.hexPath))
198                         else:
199                             self.makeSendFile(fileObj) # Otherwise send the file.
200
201                     else:
202                         print("[BT]: Couldn't find that file :/")
203                         self.clientSock.send("!NOTFOUND!")
204
205
206             except IOError as e:
207                 print(e) # Will be caused when app on mobile closes.
208
209             print("[BT]: Closed.")
210
211             self.clientSock.close()
212             self.serverSock.close()
213             self.lock(fromRunServ=True)
214
215     def sendFileList(self, fileList):
216         # File list sent like: !FILELIST!--filename1--filename2~!!ENDLIST!
217         self.clientSock.send("!FILELIST!")
218         print("[BT]: Sent !FILELIST!")
219
220         for i in fileList:
221             self.clientSock.send("--{}".format(i))
222
223         print("[BT]: Sent full list, now sent end.")
224         self.clientSock.send("~!!ENDLIST!")
225
226
227     def getListForSend(self, path):
228         if not path:
229             return False
230         else:
231             _, fsDec = self.listdir(path)
232             self.lastPathSent = path
233             return [i for i in fsDec if i != ".thumbnails" and i != ".recycling"]
234             # Cheeky bit of list comprehension so that these two folders are not sent.
235
236
237     ##Functions for changing screen within threads (used to prevent segmentation faults)
238     @mainthread
239     def changeToMain(self): # Has to be defined in a function because threads need a target function.
240         self.manager.current = "Main"
241
242     @mainthread
243     def changeToLogin(self): # Only used for checkServerStatus because you can only return a function or
244     variable, and if i execute this within the thread then it causes a segmentation fault.
245         self.manager.current = "Login"

```

```

245 #####
246
247     def startBT(self):
248         self.serverThread = Thread(target=self.runServMain, daemon=True)      #Start BT server as thread so
the screen still renders.
249         self.serverThread.start()
250
251     def setupSortButtons(self):
252         self.sortsGrid = GridLayout(cols=2, size_hint=(.99, .04), pos_hint={"x": .005, "y": .79})      #Make a
grid of 1 row (columns=2 and i am only adding 2 widgets) to hold sort buttons.
253         self.nameSort = mainBtns.nameSortButton(self, text="^")  # Default starts with Alphabetical sort
ascending.
254         self.sizeSort = mainBtns.sizeSortButton(self)
255         self.sortsGrid.add_widget(self.nameSort)
256         self.sortsGrid.add_widget(self.sizeSort)
257         self.add_widget(self.sortsGrid) #Add the sort buttons grid to the float layout of MainScreen.
258
259     def getGoodUnit(self, bytes):      #Get a good unit for displaying the sizes of files.
260         if bytes == " -":
261             return " -"
262         else:
263             divCount = 0
264             divisions = {0: "B", 1: "KB", 2: "MB", 3: "GB", 4: "TB", 5: "PB"}
265             while bytes > 1000:
266                 bytes = bytes/1000
267                 divCount += 1
268
269             return ("%.2f" % bytes) + divisions[divCount]
270
271     def getSortedFoldersAndFiles(self, fileObjects, inverse=False): # Sorts list of fileObjects by folder/file
and name
272         folders = []
273         files = []
274         for i in range(len(fileObjects)):    #Separate into folders and files
275             if fileObjects[i].isDir:
276                 folders.append(fileObjects[i])
277             else:
278                 files.append(fileObjects[i])
279
280         foldersSort = self.sortAlph(folders)    #Quick sort the list of folders and the list of files.
281         filesSort = self.sortAlph(files)
282
283         if inverse: #If inverse
284             foldersSort = foldersSort[::-1] #Invert the array
285             filesSort = filesSort[::-1]
286
287         return foldersSort+filesSort
288
289     def openRecycling(self): # Open the recycling folder.
290         if not os.path.exists(self.recycleFolder):
291             print("Recycling folder doesn't exist, making one now.")
292             makedirs(self.recycleFolder)
293
294             Popup(title="Changed Mode",
295                  content=Label(text="You are now in the\nrecycling folder.\nClick files to restore, and \nenter
the INFO menu\n to see more information,\nor delete the file permanently."),
296                  pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
297             self.currentDir = self.recycleFolder
298             self.removeButtons()
299             print(self.currentDir, "current dir")
300             self.createButtons(self.List(self.currentDir))
301
302
303 #####Button Creation and button functions#####
304     def createButtonsCore(self, array): # Makes each file button with it's information and adds it to the
scroll view.
305         self.currentList = array
306         for item in array:

```

```

307         if item.name != ".$recycling" and item.name != ".$thumbs": # If the folder is the recycling folder
308             or thumbnail temporary folder, don't draw it.
309             back = (1, 1, 1, 1)
310             if item.isDir: # Colour folders darker than files
311                 back = (0.3, 0.3, 0.3, 1) # Works as a tint rather than a colour.
312
313             btn = mainBtns.listButton(self, item, text="    "+item.name), background_color=back)
314             info = mainBtns.infoButton(self, item, background_color=back)
315
316             btn.bind(size=btn.setter("text_size")) # Set the text to wrap within the button
317             info.bind(size=info.setter("text_size"))
318             fileS = Label(text=" "+str(item.size), size_hint=(.1, 1), halign="left", valign="middle")
319             fileS.bind(size=fileS.setter("text_size")) # Wrap text in label
320             self.grid.add_widget(btn)
321             self.grid.add_widget(info)
322             self.grid.add_widget(fileS)
323
323     def createButtons(self, fileObjects, sort=True):
324         self.currentList = []
325         if sort:
326             print("sorting")
327             fileObjects = self.getSortedFoldersAndFiles(fileObjects) #Sort the list of files.
328
329             self.grid = GridLayout(cols=3, size_hint_y=None)
330             self.grid.bind(minimum_height=self.grid.setter("height"))
331             self.scroll = ScrollView(size_hint=(.99, .79), pos_hint={"x": .005, "y": 0}) #Grid is added to the
332             scroll view.
333             self.scroll.add_widget(self.grid)
334
334     self.add_widget(self.scroll) #Scroll view is added to the float layout of MainScreen.
335     self.createButtonsCore(fileObjects)
336
337     def traverseButton(self, fileObj): # Function when file is clicked.
338         if self.recycleFolder not in self.currentDir:
339             if fileObj.isDirectory: #If is a folder, then display files within that folder.
340                 self.previousDir = self.currentDir
341                 self.currentDir = fileObj.hexPath
342                 self.ascending = True
343                 self.resetButtons()
344             else: # If is a file, decrypt the file and open it.
345                 self.decrypt(fileObj)
346         else:
347             print("Recovering this file to path:", fileObj.name)
348             move(fileObj.hexPath, self.path) # Imported from shutil
349             self.refreshFiles()
350
351     def openAddFilePop(self): # Needs to be assigned to self.smallPop because if the screen is closed with
352     the popup open (only possible when using Bluetooth), all crucial popups need to be closed.
353         self.smallPop = mainSPops.addFilePop(self)
354         self.smallPop.open()
355
355     def openAddFolderPop(self):
356         self.smallPop = mainSPops.addNewFolderPop(self)
357         self.smallPop.open()
358
359     def onFileInfoClose(self, fileObj, _): # _ is me discarding the popup object.
360         if os.path.exists(fileObj.thumbDir): # Remove temporary thumbnail directory once done with thumbnail
361             os.remove(fileObj.thumbDir)
362
363     def getFileInfo(self, fileObj): #Get information about a file/folder.
364         size = (.7, .4) # Size of popup
365         if fileObj.extension == "png" or fileObj.extension == "jpg":
366             thumb = self.getThumbnail(fileObj)
367             size = (.8, .5) # Increase size of popup to display image preview.
368
369         # Works as: internalLayout -> scrollView + (Image?)
370         # scrollView contains infoGrid with all of the file's information.
371         internalLayout = BoxLayout(orientation="horizontal", size_hint=(1, 1))
372         scrollView = ScrollView()

```

```

373     self.infoPopup = Popup(title="File Information", content=internalLayout, pos_hint={"center_x": .5,
374 "center_y": .5}, size_hint=size)
375     self.infoPopup.bind(on_dismiss=partial(self.onFileInfoClose, fileObj, ))
376
377     infoGrid = GridLayout(cols=2, size_hint_y=None, row_default_height=40)
378     scrollView.add_widget(infoGrid)
379     internalLayout.add_widget(scrollView)
380
381     if fileObj.extension == "png" or fileObj.extension == "jpg":
382         internalLayout.add_widget(thumb)
383
384     infoGrid.add_widget(self.infoLabel(text="File Name:", halign="left", valign="middle"))
385     infoGrid.add_widget(self.infoLabel(text=fileObj.name, halign="left", valign="middle"))
386
387     infoGrid.add_widget(self.infoLabel(text="Current Location:", halign="left", valign="middle"))
388     infoGrid.add_widget(self.infoLabel(text="/Vault/" + fileObj.decryptRelPath(), halign="left",
389                                     valign="middle"))
390
391     infoGrid.add_widget(self.infoLabel(text="Size:", halign="left", valign="middle"))
392     infoGrid.add_widget(self.infoLabel(text=str(fileObj.size), halign="left", valign="middle"))
393
394     delText = "Delete"
395     if self.recycleFolder in self.currentDir:    # If in the recycling folder, then delete the item
396     permanently.
397     delText = "Delete Permanently"
398
399     infoGrid.add_widget(mainBtns.deleteButton(self, fileObj, text=delText))
400
401     decBtnText = "Decrypt File"
402     if fileObj.isDir:
403         decBtnText = "Decrypt Folder"
404
405     if fileObj.rawSize > 0:
406         decBtn = Button(text=decBtnText, halign="left", valign="middle")
407         decBtn.bind(on_release=partial(self.decryptFileToLoc, fileObj))
408         infoGrid.add_widget(decBtn)
409
410     self.infoPopup.open()
411
412     def makeSendFile(self, fileObj, buttonInstance=None):
413         self.sendFile = mainSPops.btTransferPop(self, fileObj)
414         self.sendFile.open()
415
416     def moveFileToRecycling(self, fileObj):
417         print("Moving", fileObj.hexPath)
418         if os.path.exists(fileObj.hexPath):
419             move(fileObj.hexPath, self.recycleFolder) # Imported from shutil
420         else:
421             raise FileNotFoundError(fileObj.hexPath, "Not a file, can't move to recycling.") # Doesn't exist,
so issue with code somewhere.
422
423     def deleteFile(self, fileObj):
424         if os.path.exists(fileObj.hexPath): #Checks file actually exists before trying to delete it.
425             if self.recycleFolder not in self.currentDir: # If outside of recycling bin.
426                 print("Moving", fileObj.hexPath)
427                 if os.path.exists(self.recycleFolder+fileObj.hexName):
428                     if os.path.isdir(self.recycleFolder+fileObj.hexName):
429                         rmtree(self.recycleFolder+fileObj.hexName)
430                     else:
431                         os.remove(self.recycleFolder+fileObj.hexName)
432                     move(fileObj.hexPath, self.recycleFolder) # Imported from shutil
433                 else:
434                     print("Deleting:", fileObj.hexPath, "and checking temp.")
435                     if os.path.exists(self.osTemp+"FileMate"+self.fileSep+fileObj.name): # If removing
permanently, check that the file is not decrypted in <system_temp>.
436                         os.remove(self.osTemp+"FileMate"+self.fileSep+fileObj.name)
437                     if fileObj.isDir: # Delete the file/folder
438                         rmtree(fileObj.hexPath) # Imported from shutil
439                     else:

```

```

437         os.remove(fileObj.hexPath)
438         self.refreshFiles()
439         self.infoPopup.dismiss()
440
441     else:
442         raise FileNotFoundError(fileObj.hexPath, "Not a file, can't delete.")
443
444 def goBackFolder(self):      #Go up a folder.
445     if self.currentDir != self.path:    #Can't go further past the vault dir.
446         self.previousDir = self.currentDir
447         if self.recycleFolder in self.currentDir:
448             self.goHome()
449         else:
450             self.currentDir = self.getPathBack(self.currentDir)
451             self.resetButtons()
452     else:
453         print("Can't go further up.")
454         return False
455
456 def getPathForButton(self, item):  # Get the path to the asset for each button.
457     return self.assetsPath+item
458
459 def removeButtons(self):    # Remove the list of files.
460     self.grid.clear_widgets()
461     self.scroll.clear_widgets()
462     self.remove_widget(self.scroll)
463
464 def resetButtons(self): # Goes back to self.currentDir, different to refresh.
465     self.removeButtons()
466     self.nameSort.text = "˄"
467     self.sizeSort.text = ""
468     self.createButtons(self.List(self.currentDir), sort=False)
469
470 def refreshFiles(self):   # Refreshes the files in the current directory
471     self.removeButtons()
472     self.createButtons(self.List(self.currentDir), sort=False)
473
474 def refreshButtons(self): # Refreshes file list buttons currently displayed.
475     self.removeButtons()
476     self.createButtons(self.currentList, sort=False)
477
478 def goHome(self):    #Takes the user back to the vault dir.
479     self.currentDir = self.path
480     self.refreshFiles()
481
482 def List(self, dir):    # Lists a directory, returning File objects.
483     fs, fsDec = self.listDir(dir)  # Lists encrypted names and decrypted names.
484     listOfFolders = []
485     listOfFiles = []
486     for i in range(len(fs)):
487         if os.path.isdir(dir+fs[i]):
488             listOfFolders.append(File(self, dir+fs[i], fs[i], self.fileSep, name=fsDec[i], isDir=True))
489         else:
490             listOfFiles.append(File(self, dir+fs[i], fs[i], self.fileSep, name=fsDec[i]))
491
492     return listOfFolders+listOfFiles
493
494 def getPathBack(self, origPath): # Gets the path above the current folder.
495     tempDir = origPath.split(self.fileSep)
496     del tempDir[-2]
497     tempDir = self.fileSep.join(tempDir)
498     return tempDir
499
500 #####Searches#####
501 def findAndSortCore(self, dirName, item):
502     files = self.List(dirName)
503     if len(files) == 0:
504         return
505     for fileObj in files:

```

```

506         loc = fileObj.name.find(item) # Find where in the word the item is found, if it is a substring of
507         the word
508
509         if fileObj.name == item:
510             self.searchResults = [fileObj] + self.searchResults
511         elif loc != -1: # If the search term is a substring of the current word
512             self.unsorted.append((loc, fileObj)) #Adds loc found in word, so that it can be sorted by
513             where it is found
514
515             if (fileObj.isDir and self.searchRecursively) and (fileObj.hexPath != self.recycleFolder) and
516             (fileObj.hexName != self.thumbsName):
517                 self.findAndSortCore(fileObj.hexPath, item) # Search folder if recursive and not recycle
518             folder or thumbnail folder.
519
520
521     def findAndSort(self, item):      #Main search function.
522         self.unsorted = []
523         self.findAndSortCore(self.currentDir, item)
524
525         if len(self.unsorted)+len(self.searchResults) > 0:
526             sorted = self.sortSearch(self.unsorted)
527             for i in sorted:
528                 self.searchResults.append(i)
529             mainthread(self.removeButtons())
530             return mainthread(self.createButtons(self.searchResults, False))
531
532
533     def searchForItem(self, item):
534         self.searchResults = []
535         Thread(target=self.findAndSort, args=(item,), daemon=True).start()
536
537
538 #####Progress Bar Information#####
539     def values(self, st):  #Information for space left on device.
540         values = disk_usage(self.path) # Imported from shutil
541         if st:
542             return self.getGoodUnit(int(values[1]))+ " / " + self.getGoodUnit(int(values[0])) + " used."
543         else:
544             return [values[0], values[1]]
545
546
547 #####Encryption Stuff + opening decrypted files + interface with go#####
548     def encDec(self, encType, d, targetLoc, newName=None, op=True): # Encrypt and decrypt. A wrapper for the
549     thread that will do it.
550         if self.encPop != None:
551             self.encPop.dismiss()
552             self.encPop = None
553
554         def encThread(encType, d, targetLoc, op=True):
555             if os.path.isdir(d):
556                 if not os.path.exists(targetLoc):
557                     os.makedirs(targetLoc)
558                 fileList, locList = self.getDirLists(encType, d, targetLoc)
559                 self.encPop = mainSPops.encDecPop(self, encType, fileList, locList, op=op)
560                 mainthread(self.encPop.open())
561                 self.passToPipe(encType+"Dir", "\n".join(fileList), "\n".join(locList))
562             else:
563                 size = os.path.getsize(d)
564                 if size > 10000: # If larger than 10 kb, otherwise don't bother
565                     self.encPop = mainSPops.encDecPop(self, encType, [d], [targetLoc] , op=op)
566                     mainthread(self.encPop.open())
567                     self.passToPipe(encType, d, targetLoc)
568
569         if self.encPop != None:

```

```

569         mainthread(self.encPop.dismiss())
570         self.encPop = None
571
572     if encType == "y":
573         self.resetButtons()
574
575     if op and encType == "n":
576         self.openFileTh(targetLoc, d)
577
578
579     return Thread(target=encThread, args=(encType, d, targetLoc, op,)).start()
580
581
582 def passToPipe(self, type, d, targetLoc):      #Passes parameters to AES written in go.
583     if self.fileSep == "\\\":
584         progname = "AESWin.exe"
585     else:
586         progname = "AES/AES"
587
588     goproc = Popen(self.startDir+progname, stdin=PIPE, stdout=PIPE)
589     out, _ = goproc.communicate((type+, "+d+", "+targetLoc+", "+self.key).encode()) # Send parameters to
AES
590     return out
591
592 ## Utilising passToPipe function
593 def getDirLists(self, encType, root, targ): # Communicates with AES to get list of files in a folder, and
their target.
594     out = self.passToPipe("getLists"+encType, root, targ).decode()
595     out = out.split("--!--") # Separator
596     return out[0].split(",,"), out[1].split(",,")

597
598 def listDir(self, location):
599     out = self.passToPipe("listDir", location, "").decode()
600     out = out.split("--!--") # Separator
601     out = [out[0].split(",,"), out[1].split(",,")]
602     if out[0] == []:
603         out[0] = []
604     if out[1] == []:
605         out[1] = []
606     return out[0], out[1]
607
608 def encString(self, string):
609     out = self.passToPipe("encString", string, "").decode()
610     return out
611
612 def decString(self, string):
613     out = self.passToPipe("decString", string, "").decode()
614     return out
615
616 def encListString(self, list):
617     out = self.passToPipe("encList", "\n".join(list), "").decode()
618     return out.split(",,")

619
620 def sortSize(self, fileObjects):
621     out = self.passToPipe("sortSize", "\n".join([str(i.rawSize) for i in fileObjects]), "").decode()
622     out = [int(i) for i in out.split(",")]
623     if len(out) != len(fileObjects):
624         raise ValueError("Length of sizes not the same as original:", len(out), len(fileObjects))
625     outList = []
626     for i in range(len(fileObjects)):
627         outList.append(-1) # Initialize
628
629     for i in fileObjects:
630         outList[out.index(i.rawSize)] = i # Insert the file object on the place in outList that
corresponds to where it's size is in the 'out' list.
631
632     return [i for i in outList if i != -1] # In case of any left-overs
633
634 def sortAlph(self, fileObjects):

```

```

635     out = self.passToPipe("sortAlpha", "\n".join([str(i.name) for i in fileObjects]), "").decode()
636     out = [str(i) for i in out.split(",")]
637
638     return self.matchFileObjToName(fileObjects, out)
639
640     def sortSearch(self, searchResults):
641         out = self.passToPipe("sortSearch", "\n".join([str(i[0]) for i in searchResults]),
642 "\n".join([i[1].name for i in searchResults])).decode()
643         out = [str(i) for i in out.split(",")]
644
645     return self.matchFileObjToName([i[1] for i in searchResults], out)
646
647     def decListString(self, list):
648         out = self.passToPipe("decList", "\n".join(list), "").decode()
649         return out.split(",")
650
651     def matchFileObjToName(self, fileObjects, listOfNames):    # Used when sorting file objects by name
652         if listOfNames == []:
653             return []
654         if len(listOfNames) != len(fileObjects):
655             print(listOfNames)
656             print(fileObjects)
657             raise ValueError("Length of names not the same as original:", len(listOfNames), len(fileObjects))
658         outList = []
659         for i in range(len(fileObjects)):
660             outList.append(-1)    # Initialize
661
662         for i in fileObjects:
663             outList[listOfNames.index(i.name)] = i    # Insert the file object on the place in outList that
corresponds to where it's size is in the 'out' list.
664
665         return [i for i in outList if i != -1] # In case of any left-overs
666
667     def getCheckSum(self, location):    # Communicates to BLAKE to get checksum.
668         if self.fileSep == "\\":        # If on windows
669             goproc = Popen(self.startDir+"BLAKEWin.exe", stdin=PIPE, stdout=PIPE)
670         elif self.fileSep == "/":
671             goproc = Popen(self.startDir+"BLAKE/BLAKE", stdin=PIPE, stdout=PIPE)
672
673         out, err = goproc.communicate((location).encode())
674         if err != None:
675             raise ValueError(err)
676
677         return out.decode()
678
679     def getThumbnail(self, fileObj):
680         if self.thumbsName not in self.currentDir:    # Only check this when not in the thumbnail folder
681             if self.thumbsName not in os.listdir(self.currentDir): # Checks that there is a thumbnail folder
in this directory.
682                 os.makedirs(self.currentDir+self.thumbsName)
683                 print("Made thumbnail directory since it wasn't there")
684
685         fileObj.thumbDir = self.currentDir+self.thumbsName+self.fileSep+fileObj.hexName
686         self.passToPipe("\n", fileObj.hexPath, fileObj.thumbDir) # Decrypts thumbnail temporarily. Is deleted
once program is finished displaying it.
687         thumb = Image(source=fileObj.thumbDir)
688         return thumb
689
690     def openFileTh(self, fileLoc, startLoc):    # Creates a thread to open a file (stops program locking up)
691         Thread(target=self.openFile, args=(fileLoc, startLoc,), daemon=True).start()
692
693     def openFile(self, location, startLoc):
694         locationFolder = location.split(self.fileSep)
695         nameOfOriginal = locationFolder[-1]
696         locationFolder = self.fileSep.join(locationFolder[:-1])
697         if self.fileSep == "\\":
698             location = location.split("\\\\")
699             location = "/".join(location) # Windows actually accepts forward slashes in terminal

```

```

700         command = "START /wait "+'""'+location+'"'
701     else:
702         command = "xdg-open "+'""'+location+'"'      # Quotation marks for if the dir has spaces in it
703
704     startCheckSum = self.getCheckSum(location) # Gets checksum of file before opening.
705     os.system(command)# Using the same for both instead of os.startfile because os.startfile doesn't wait
706     for file to close
707         # After this line, the file has been closed.
708     if os.path.exists(locationFolder) and nameOfOriginal in os.listdir(locationFolder):
709         print("Original still here")
710         if self.getCheckSum(location) != startCheckSum:
711             print("Original file has changed.")
712             self.encDec("y", location, startLoc)
713
714     def onFileDrop(self, window, filePath): # For draging + dropping files into the window.
715         self.checkCanEncrypt(filePath.decode())
716         return "Done"
717
718     def decrypt(self, fileObj, op=True):           # Default decrypt for file. Manages opening files etc.
719         if not os.path.isdir(self.osTemp+FileMate+self.fileSep):
720             os.makedirs(self.osTemp+FileMate+self.fileSep)
721         fileLoc = self.osTemp+FileMate+self.fileSep+fileObj.name #Place in temporary files where it is
722         going to be stored.
723         if os.path.exists(fileLoc) and op:          #Checks file exists already in temp files, so it doesn't
724             have to decrypt again.
725             self.openFileTh(fileLoc, fileObj.hexPath)
726         else:
727             self.encDec("n", fileObj.hexPath, fileLoc, newName=fileObj.name, op=op)
728
729     def decryptFileToLoc(self, fileObj, button):   # Decrypt a file/folder to a location (just handles the
730     input)
731     mainSPops.decryptFileToLocPop(self, fileObj).open()
732
733     def checkCanEncryptCore(self, inp): # Used for adding new files to the vault by the user.
734         if self.checkDirExists(inp):
735             inpSplit = inp.split(self.fileSep)
736
737             if os.path.isdir(inp):
738                 while inpSplit[-1] == "":          # Removes excess "/" from input
739                     inpSplit = inpSplit[:-1]
740                 targ = self.currentDir+self.encString(inpSplit[-1])+self.fileSep
741                 inp = self.fileSep.join(inpSplit)+self.fileSep
742             else:
743                 targ = self.currentDir+self.encString(inpSplit[-1])
744                 inp = self.fileSep.join(inpSplit)
745                 self.encDec("y", inp, targ)
746
747     def checkCanEncrypt(self, inp): # Used for adding new files to the vault by the user.
748         if "--" in inp: # Multiple files/folders input.
749             inp = inp.split("--")
750             for d in inp:
751                 self.checkCanEncryptCore(d) # Actally encrypt/decrypt it.
752             else:
753                 self.checkCanEncryptCore(inp)
754
755     def checkDirExists(self, dir): #Handles UI for checking directory exits when file added.
756         if os.path.exists(dir):
757             return True
758         else:
759             self.popup = Popup(title="Invalid", content=Label(text=dir+" - Not a valid directory."), pos_hint=
{ "center_x": .5, "center_y": .5}, size_hint=(.4, .4))
760             self.popup.open()
761             return False
762
763     def clearUpTempFiles(self):    # Deletes temp files when the program is locked.
764         print("Deleting temp files.")
765         try:
766             rmtree(self.osTemp+FileMate+self.fileSep) # Imported from shutil

```

```
764     except:
765         print("No temp files.")
```

`passToPipe` is used to pass commands through `stdin` to AES, and receive results through `stdout`. There are many functions just below it dedicated to formatting data to be sent through the pipe, for sorting files and many other functions. `encDec` is the function used in the program for encryption and decryption, while `decrypt` is a function used as a default operation to perform when a file is decrypted (open it, make a temporary folder etc).

The temporary files are removed when the program is locked using `clearUpTempFiles`, called by `lock` when the BT server is closed, or when the lock button is pressed.

`findAndSort` doesn't return anything, but instead edits `self.searchResults`. This is because it was easier to do it this way than pass it in every time, when it is needed by the rest of the program anyway. I just need to make sure I set it to `self.searchResults = []` before getting new results.

The `File` class (defined in `code/python-go/fileClass.py`) is an integral part of the program, as they are used throughout to handle the file's encrypted path, decrypted name and a host of other important information needed throughout the program. The `File` class has made programming other functions much easier and cleaner, due to the data already being there.

The `values` function returns the values required by the progress bar showing the amount of storage space left on the current device, and can optionally return this in string form for the text above the status bar.

Most custom child widgets that the MainScreen needs are defined in `code/python-go/kivyStuff/mainBtns.py` and `code/python-go/kivyStuff/mainSmallPops.py`, and their styling which is in `code/python-go/kivyStuff/kvFiles/mainScClasses.kv`, but I will talk more about those after I go through the styling for the MainScreen class.

Here is the `.kv` file for MainScreen (`code/python-go/kivyStuff/kvFiles/mainSc.kv`):

```
1 #:include kivyStuff/kvFiles/mainScButtons.kv
2 #:include kivyStuff/kvFiles/mainScPops.kv
3 #:include kivyStuff/kvFiles/mainScLabels.kv
4
5 <MainScreen>:
6     addFile: addFile
7     RelativeLayout: #Adds background in case fade transition breaks.
8         canvas.before:
9             Color:
10                 rgba: 0,0,0,2
11                 Rectangle:
12                     pos: self.pos
13                     size: self.size
14
15         FloatLayout:
16             id: MainLayout
17
18             TextInput:
19                 id: Search
20                 size_hint: .52, .08
21                 font_size: 22
22                 hint_text: "Search:"
23                 pos_hint: {"x": 0.16, "top": 1}
24                 multiline: False
25                 on_text_validate: root.searchForItem(self.text)
26
27             Button:
28                 id: home
29                 size_hint: .06, .08
```

```

30     pos_hint: {"x": 0.62, "y": 0.84}
31     on_release: root.goHome()
32     Image:
33         source: root.getPathForButton("home.png")
34         center_x: self.parent.center_x
35         center_y: self.parent.center_y
36         size: 40, 40
37         allow_stretch: True
38
39     Button:
40         id: settings
41         size_hint: .06, .08
42         pos_hint: {"x": 0.56, "y": 0.84}
43         on_release: root.manager.current = "Settings"
44         Image:
45             source: root.getPathForButton("settings.png")
46             center_x: self.parent.center_x
47             center_y: self.parent.center_y
48             size: 40, 40
49             allow_stretch: True
50
51     Button:
52         id: addFolder
53         size_hint: .06, .08
54         pos_hint: {"x": 0.5, "y": 0.84}
55         text_size: self.size
56         halign: "center"
57         valign: "center"
58         on_release: root.openAddFolderPop()
59         Image:
60             source: root.getPathForButton("newFolder.png")
61             center_x: self.parent.center_x+3 # Drop shadow
62             center_y: self.parent.center_y
63             size: 50, 50
64
65     Button:
66         id: recyclingBin
67         size_hint: .06, .08
68         pos_hint: {"x": 0.44, "y": 0.84}
69         on_release: root.openRecycling()
70         Image:
71             source: root.getPathForButton("recycling.png")
72             center_x: self.parent.center_x
73             center_y: self.parent.center_y+3
74             size: 50, 50
75
76     ProgressBar:
77         id: pbl
78         min: 0
79         max: root.values(False)[0]
80         value: root.values(False)[1]
81         size_hint: .28, .08
82         pos_hint: {"x": 0.16, "y": 0.84}
83
84     Label:
85         pos_hint: {"x": 0.2, "y": 0.82}
86         size_hint: .16, .16
87         text: root.values(True)
88
89     Button: #Padlock
90         id: LogOut
91         size_hint: .16, .16
92         pos_hint: {"x": 0, "top": 1}
93         on_release: root.lock()
94         Image:
95             source: root.getPathForButton("padlock.png")
96             center_x: self.parent.center_x
97             center_y: self.parent.center_y
98             size: 60, self.size[1]

```

```

99
100    Button:
101        id: addFile
102        size_hint: .16, .16
103        pos_hint: {"right": 1, "top": 1}
104        on_release: root.openAddFilePop()
105        Image:
106            source: root.getPathForButton("addFile.png")
107            center_x: self.parent.center_x+2 # Drop shadow
108            center_y: self.parent.center_y-5
109            size: 100, 133.54
110            allow_stretch: False
111
112    Button:
113        id: goBackFolder
114        size_hint: .16, .16
115        pos_hint: {"x": 0.68, "top": 1}
116        on_release: root.goBackFolder()
117        Image:
118            source: root.getPathForButton("backUpFolder.png")
119            center_x: self.parent.center_x+2 # Drop shadow
120            center_y: self.parent.center_y
121            size: 100, 133.54
122            allow_stretch: False
123

```

The first three lines import the styling for the custom child widgets, which I will go through next.

Main Screen Buttons

All of the custom buttons for MainScreen are defined in `code/python-go/kivyStuff/mainBtns.py`. Here is the code:

```

1  from kivy.uix.button import Button
2  from kivy.uix.image import Image
3
4  from sortsCy import quickSortSize
5
6  class listButton(Button):          #File button when using main screen.
7
8      def __init__(self, fileObj, **kwargs):
9          super(Button, self).__init__(**kwargs)
10         self.fileObj = fileObj           #The file the button corresponds to.
11
12 class nameSortButton(Button):      #Sorts the listButtons alphabetically and by folders/files.
13
14     def __init__(self, mainScreen, **kwargs):
15         super(Button, self).__init__(**kwargs)  # Run kivy Button.__init__ class with it's key word arguments.
16         self.outerScreen = mainScreen
17
18     def changeSortOrder(self):
19         self.outerScreenascending = not self.outerScreenascending
20         if self.outerScreenascending:
21             self.text = "^"
22             self.outerScreen.removeButtons()
23             self.outerScreen.createButtons(self.outerScreen.currentList, True)
24         else:
25             self.text = "v"
26             self.outerScreen.removeButtons()
27             self.outerScreen.createButtons(self.outerScreen.currentList[::-1], False)
28
29 class sizeSortButton(Button):      #Sorts the files/folders by size
30
31     def __init__(self, mainScreen, **kwargs):
32         super(Button, self).__init__(**kwargs)
33         self.outerScreen = mainScreen
34         self.ascending = True

```

```

35     self.sortList = []
36
37
38     def sortBySize(self):
39         self.sortList = quickSortSize(self.outerScreen.currentList)
40         if not selfascending:
41             self.sortList = self.sortList[::-1]      # Reverse sorted list.
42
43         self.outerScreen.removeButtons()
44         self.outerScreen.createButtons(self.sortList, False)
45
46     def changeSizeOrder(self):
47         self.ascending = not self.ascending
48         if self.ascending:
49             self.text = "v"
50         else:
51             self.text = "^"
52
53         if (self.sortList) and (self.outerScreen.previousDir == self.outerScreen.currentDir):    # Checking that
the sortList is for the current directory and we haven't moved.
54             self.sortList = self.sortList[::-1]
55             self.outerScreen.currentList = self.sortList
56             self.outerScreen.removeButtons()
57             self.outerScreen.createButtons(self.sortList, False)
58         else:
59             self.outerScreen.previousDir = self.outerScreen.currentDir
60             self.sortBySize()
61
62     class infoButton(Button):      #The button that displays information about the file.
63
64         def __init__(self, mainScreen, fileObj, **kwargs):
65             self.outerScreen = mainScreen
66             super(Button, self).__init__(**kwargs)
67             self.fileObj = fileObj
68
69
70     class deleteButton(Button):
71
72         def __init__(self, mainScreen, fileObj, **kwargs):
73             super(Button, self).__init__(**kwargs)
74             self.outerScreen = mainScreen
75             self.fileObj = fileObj

```

The `listButton` class is just a regular `Button`, but can take a `File` class (as `fileObj`), and has a reference to the current `MainScreen` object. `listButton` has no custom methods.

The `nameSortButton` class is a regular `Button`, however closely interacts with the `MainScreen` class, to change the order of items in the `ScrollView` that contains the `listButtons` on `MainScreen`. It basically handles `MainScreen`.

The `sizeSortButton` class works similarly to `nameSortButton`, however it has to handle sorting the list by size.

The `infoButton` and `deleteButton` classes are both similar to `listButton` in that they just have a few extra attributes.

Here is the `kv` file for the custom buttons (`code/python-go/kivyStuff/kvFiles/mainScButtons.kv`):

```

1 #: import Window kivy.core.window.Window
2
3 <listButton@Button>:
4     font_size: 14
5     halign: "left"
6     valign: "middle"
7     height: 30
8     size_hint: 1, None
9     on_release: root.outerScreen.traverseButton(self.fileObj)

```

```

10 <nameSortButton@Button>:
11     font_size: 14
12     size_hint: 10.4, 1
13     on_release: root.changeSortOrder()
14
15 <sizeSortButton@Button>:
16     font_size: 14
17     on_release: root.changeSizeOrder()
18
19 <infoButton@Button>:
20     font_size: 14
21     size_hint: .05, 1
22     halign: "left"
23     valign: "middle"
24     on_release: root.outerScreen.getFileInfo(self.fileObj)
25     Image:
26         source: self.parent.outerScreen.assetsPath+"info.png"
27         center_x: self.parent.center_x
28         center_y: self.parent.center_y
29         size: 20, 20
30
31 <deleteButton@Button>:
32     on_release: root.outerScreen.deleteFile(self.fileObj)

```

The first line imports `Window` from `kivy.core.window`. The equivalent in python would be:

```
1 | from kivy.core.window import Window
```

Main Screen Popups

Here is the code for the MainScreen popups ([code/python-go/kivyStuff/mainSmallPops.py](#)):

```

1 import os
2 from threading import Thread
3 from time import time, sleep
4 from random import uniform as randUniform
5
6 from kivy.uix.scrollview import ScrollView
7 from kivy.uix.gridlayout import GridLayout
8 from kivy.uix.label import Label
9 from kivy.clock import Clock
10 from kivy.clock import mainthread
11 from kivy.core.window import Window
12 from kivy.uix.button import Button
13 from kivy.uix.progressbar import ProgressBar
14 from kivy.uix.popup import Popup
15
16 from configOperations import dirInputValid
17
18 class encDecPop(Popup): #For single files
19
20     def __init__(self, outerScreen, encType, fileList, locList, op=True, **kwargs):
21         super(Popup, self).__init__(**kwargs)
22         self.outerScreen = outerScreen
23         self.fileList = fileList
24         self.locList = locList
25
26         # Kivy stuff
27         self.title = "Please wait..."
28         self.pos_hint = {"center_x": .5, "center_y": .5}
29         self.size_hint = (.7, .4)
30         self.auto_dismiss = False
31
32         self.grid = GridLayout(cols=1)

```

```

33     self.subGrid = GridLayout(cols=4)
34     self.currFile = Label(text="", halign="center", valign="center")
35     self.currFile.bind(size=self.currFile.setter("text_size")) # Wrap text inside label
36     self.per = Label(text="")
37     self.spd = Label(text="")
38     self.tim = Label(text="")
39     self.outOf = Label(text="")
40     self.pb = ProgressBar(value=0, max=os.path.getsize(self.fileList[0]), size_hint=(.9, .2))
41     self.wholePb = ProgressBar(value=0, max=self.__getTotalSize(), size_hint=(.9, .2))
42     labText = "Encrypting..."
43     if encType == "n":
44         labText = "Decrypting..."
45     self.grid.add_widget(Label(text=labText, size_hint=(1, .4)))
46     self.grid.add_widget(self.currFile)
47     self.subGrid.add_widget(self.per)
48     self.subGrid.add_widget(self.spd)
49     self.subGrid.add_widget(self.tim)
50     self.grid.add_widget(self.subGrid)
51     if len(self.fileList) > 1: # Don't bother showing 2 progress bars if the user is only doing 1 file.
52         self.grid.add_widget(self.pb)
53         self.subGrid.add_widget(self.outOf)
54     self.grid.add_widget(self.wholePb)
55     self.content = self.grid
56
57     self.checkThread = Thread(target=self.encDec, args=(encType, op,), daemon=True)
58     self.checkThread.start()
59
60     def __getTotalSize(self):
61         total = 0
62         for file in self.fileList:
63             total += os.path.getsize(file)
64         return total
65
66     def getGoodUnit(self, bps):
67         divCount = 0
68         divisions = {0: "B/s", 1: "KB/s", 2: "MB/s", 3: "GB/s", 4: "TB/s"}
69         while bps > 1000:
70             bps = bps/1000
71             divCount += 1
72
73         return ("%.2f" % bps) + " " + divisions[divCount]
74
75     def __getGoodUnitTime(self, time):
76         divCount = 0
77         times = [(0.001, "Miliseconds"), (1, "Seconds"), (60, "Minutes"), (3600, "Hours"), (86400, "Days"),
78 (604800, "Weeks"), (2419200, "Months"), (31557600, "Years")] # 1 second, 1 minute, 1 hour, 1 day, 1 week, 1
month, 1 year in seconds
79         i = 0
80         while i < len(times): # Is broken when return is found
81             if time > times[i][0]:
82                 i += 1
83             else:
84                 return ("%.2f" % float(time/times[i-1][0])) + " " + times[i-1][1] + " left"
85
86         return "A lot of time left."
87
88     def __getRelPathDec(self, path): # Similar to decryptRelPath in fileClass
89         splitPath = (path.replace(self.outerScreen.path, "")).split(self.outerScreen.fileSep)
90         return "/Vault/" + self.outerScreen.fileSep.join(self.outerScreen.decListString(splitPath))
91
92     def __getMeanOfList(self, l):
93         out = 0
94         for i in l:
95             out += i
96         return out/len(l)
97
98     def encDec(self, encType, op):
99         total = 0
100        totalPer = 0

```

```

100     factor = 0.5
101     timeLast = 0
102     lastSize = 0
103     timeDelta = 0
104     perDelta = 0
105     per = 0
106     prevPer = 0
107     lastPerDeltas = [] # Stores 8 of the last percentage deltas so that a mean can be obtained.
108     for i in range(len(self.fileList)):
109         done = False
110         self.pb.value = 0
111         self.pb.max = os.path.getsize(self.fileList[i])
112
113         self.outOf.text = str(i)+"/"+str(len(self.fileList))
114         if encType == "n":
115             self.currFile.text = self.__getRelPathDec(self.fileList[i])
116         else:
117             self.currFile.text = self.fileList[i]
118
119         while not done: # Padding can cause issues as original size is not known.
120             if os.path.exists(self.locList[i]):
121                 self.pb.value = os.path.getsize(self.locList[i])
122                 self.wholePb.value = total + self.pb.value
123                 per = self.wholePb.value_normalized*100
124
125                 a = time() # Temporary variable to hold the time
126                 timeDelta = a - timeLast # Get time difference
127                 if timeDelta >= 0.5: # Update every 0.5 seconds
128                     perDelta = per - prevPer # Change in percentage in that time.
129                     if len(lastPerDeltas) == 8:
130                         lastPerDeltas = lastPerDeltas[1:]
131                         lastPerDeltas.append(perDelta)
132                         perDelta = self.__getMeanOfList(lastPerDeltas)
133
134                     timeLast = a
135                     sizeDelta = self.wholePb.value - lastSize # Get change in size of the file being
136                     encrypted
137                     speed = sizeDelta/timeDelta # Get speed of encryption in bytes/second
138
139                     if speed != 0:
140                         self.tim.text = self.__getGoodUnitTime((100 -
141 (self.wholePb.value_normalized*100))/(perDelta/timeDelta))
142                         self.spd.text = self.getGoodUnit(speed)
143
144                     lastSize = self.wholePb.value
145                     prevPer = per
146
147                     self.per.text = "{0:.2f}%".format(per)
148
149                     if self.pb.value >= self.pb.max-64: # -64 is due to padding and key.
150                         done = True
151                     else:
152                         sleep(0.01) # Reduces the rate the file is checked, so python doesn't use too much CPU.
153                         AES will still run the same regardless, the file just doesn't need to be checked as soon as possible.
154
155                     self.pb.value = self.pb.max
156                     totalPer += 100
157                     total += self.pb.max
158
159     class btTransferPop(encDecPop):
160
161         def __init__(self, mainScreen, fileObjTmp, **kwargs):
162             super(Popup, self).__init__(**kwargs)
163             self.outerScreen = mainScreen
164             self.title = "Please wait..."
165             self.size_hint = (.7, .4)
166             self.pos_hint = {"center_x": .5, "center_y": .5}
167             self.auto_dismiss = False

```

```

166     self.grid = GridLayout(cols=1)
167     self.subGrid = GridLayout(cols=3)
168     self.currFile = Label(text=fileObjTmp.path)
169     self.per = Label(text="")
170     self.spd = Label(text="")
171     self.tim = Label(text="")
172     self.pb = ProgressBar(value=0, max=1, size_hint=(.9, .2))
173     self.grid.add_widget(Label(text="Sending..."))
174     self.grid.add_widget(self.currFile)
175     self.subGrid.add_widget(self.per)
176     self.subGrid.add_widget(self.spd)
177     self.subGrid.add_widget(self.tim)
178     self.grid.add_widget(self.subGrid)
179     self.grid.add_widget(self.pb)
180     self.content = self.grid
181
182     self.sendThread = Thread(target=self.sendFile, args=(fileObjTmp,), daemon=True) # can be cancelled mid
way through
183     self.sendThread.start()
184
185 def sendFile(self, fileObj):
186     # File name is sent with !NAME!#!!<name here>!~
187     # File data is sent right afterwards, ending with ~!!ENDF!
188     # Overall, it is sent as: !NAME!#!!<name here>!~<datahere>~!!ENDF!
189     self.outerScreen.clientSock.send("!NAME!{}~~!~".format(fileObj.name))
190     #print("!NAME!{}~~!~".format(fileObj.name), "Sent")
191
192     newLoc = self.outerScreen.osTemp+"FileMate"+self.outerScreen.fileSep+fileObj.name
193     if not os.path.isdir(self.outerScreen.osTemp+"FileMate"+self.outerScreen.fileSep):
194         os.makedirs(self.outerScreen.osTemp+"FileMate"+self.outerScreen.fileSep)
195
196     self.outerScreen.passToPipe("\n", fileObj.hexPath, newLoc)
197
198     bufferSize = 1024
199     buff = []
200     fr = open(newLoc, "rb")
201     buff = fr.read(bufferSize)      #Read 1Kb of data
202     buffCount = 0
203     self.per.text = "{0:.2f}%".format(0)
204
205     start = time()
206     #Send data
207     while buff:
208         self.outerScreen.clientSock.send(buff)
209         buffCount += bufferSize
210         buff = fr.read(bufferSize)
211
212         self.pb.value = buffCount/fileObj.rawSize
213         self.per.text = "{0:.2f}%".format(self.pb.value*100)
214         self.spd.text = self.getGoodUnit(buffCount/(time() - start))
215
216     self.outerScreen.clientSock.send("~!ENDFILE!")
217     self.dismiss()
218
219
220 class decryptFileDialog(Popup): # Input box for location of where directory is to be saved.
221
222     def __init__(self, mainScreen, fileObj, **kwargs):
223         self.outerScreen = mainScreen
224         self.fileObj = fileObj
225         super(Popup, self).__init__(**kwargs)
226
227     def makeDirs(self, dir):
228         try:
229             os.makedirs(dir)
230         except OSError as e:
231             if "[Errno 13]" in str(e): # OSError doesn't store the error code.
232                 Popup(title="Invalid", content=Label(text="Can't decrypt here.", halign="center"), size_hint=(.3, .3), pos_hint={"x_center": .5, "y_center": .5}).open()

```

```

233         return False
234     elif "[Errno 36]" in str(e):
235         Popup(title="Invalid", content=Label(text="File name too long.", halign="center"), size_hint=(.3, .3), pos_hint={"x_center": .5, "y_center": .5}).open()
236         return False
237     else:
238         return True
239
240 def checkCanDec(self, inp):
241     valid = True
242     if dirInputValid(inp, self.outerScreen.fileSep): # Re-use from settings pop, setting self as None
because it isn't even used in the function, but is needed to run from within SettingsPop.
243     if self.fileObj.isDir:
244         if not os.path.exists(inp):
245             valid = self.makeDirs(inp)
246         if inp[-1] != self.outerScreen.fileSep: inp += self.outerScreen.fileSep
247
248     if valid:
249         self.outerScreen.encDec("n", self.fileObj.hexPath, inp, op=False)
250     else:
251         if inp[-1] == self.outerScreen.fileSep: # If ends with "/", then decrypt with it's file
name.
252         if not os.path.exists(inp):
253             valid = self.makeDirs(inp)
254             inp += self.fileObj.name
255
256     if valid:
257         self.outerScreen.encDec("n", self.fileObj.hexPath, inp, op=False)
258     else:
259         Popup(title="Invalid", content=Label(text="Can't decrypt here, path is invalid.", halign="center"), size_hint=(.3, .3), pos_hint={"x_center": .5, "y_center": .5}).open()
260
261 def getTitle(self):
262     if self.fileObj.isDir:
263         return "Decrypt Folder"
264     else:
265         return "Decrypt File"
266
267
268 class addNewFolderPop(Popup):
269
270     def __init__(self, mainScreen, **kwargs):
271         super(Popup, self).__init__(**kwargs)
272         self.outerScreen = mainScreen
273
274     def makeFolder(self, text):
275         if dirInputValid(self.outerScreen.currentDir+text, self.outerScreen.fileSep):
276             try:
277                 os.makedirs(self.outerScreen.currentDir+self.outerScreen.encString(text))
278             except OSError as e:
279                 if "[Errno 36]" in str(e): #OSError doesn't store the error code for some reason.
280                     Popup(title="Invalid Folder Name", content=Label(text="Folder name too long.", halign="center"), size_hint=(.3, .3), pos_hint={"x_center": .5, "y_center": .5}).open()
281
282             self.outerScreen.refreshFiles()
283             self.dismiss()
284         else:
285             Popup(title="Invalid", content=Label(text="Invalid folder name.", halign="center"), size_hint=(.3, .3), pos_hint={"x_center": .5, "y_center": .5}).open()
286
287 class addFilePop(Popup): #The screen (it's actually a Popup) for adding folders/files to the vault.
288
289     def __init__(self, mainScreen, **kwargs):
290         super(Popup, self).__init__(**kwargs)
291         self.outerScreen = mainScreen
292
293     class ConfirmationPopup(Popup): #Popup for confirming encryption.
294
295         def __init__(self, fileScreen, input, **kwargs):

```

```

296     super(Popup, self).__init__(**kwargs)
297     self.fileScreen = fileScreen
298     self.inputText = input
299
300
301     def checkIfSure(self, input):
302         self.ConfirmationPopup(self, input).open()

```

I have not got any styling for `encDecPop` or `btTransferPop` because I have to access the GUI elements a lot. `btTransferPop` inherits from `encDecPop` to get useful methods that need to be used in both classes.

I will break down the way `encDecPop` gets the statistics it needs:

1. The `encDecPop` is created, and in `__init__` it starts a new thread which executes `self.encDec`.
2. `encDec` goes through the list of files given (`self fileList` and `self locList`) (which contains the locations to enc/decrypt to), while Go encrypts each file separately. As Go is encrypting the files, `encDec` monitors the size of the current file using `os.path.getsize(<file>)`, and monitors it while its size is less than the size of the original file.
3. The maximum value of the first status bar is set based on the size of the original file, while if enc/decrypting a folder, the second bar is based on the total size of all of the files in `fileList`.
4. Every loop the values of the status bars are updated, while every 0.5 seconds, the statistics are updated using the time difference from the last update.
5. The speed of the operation is obtained by getting the difference in size of the file in the time difference, divided by the time difference (or `sizeDelta/timeDelta`).
6. The time remaining of the operation is obtained by dividing the percentage left of all the files by the time it takes to do 1 percent (`(100 - (self.wholePb.value_normalized*100))/(perDelta/timeDelta)`). The percentage delta (`perDelta`) is calculated by taking a mean of 8 of the last percentage deltas, as this smooths the time prediction so it is not as erratic. This makes for a more accurate time prediction reading.
7. Both of these values have a function to get a nice human-readable output (`__getGoodUnit` for the speed, and `__getGoodUnitTime` for the time remaining).
8. The whole loop sleeps for 0.01 seconds to not max out the CPU, as it doesn't need to be updated that often (reduces usage to about 3 to 7%).

`btTransferPop` does work very similar to this, however since it is sending the data itself, it can get 100% accurate measurements of the statistics. Every time another batch of data is sent, the popup is updated.

`decryptFileToLocPop` is just a text input popup that is opened when the user wants to decrypt a file or folder to a certain location. It validates the input is a correct file path, and then creates a new `encDecPop` to handle the rest of the operation.

`addFilePop` is very similar to `decryptFileToLocPop`, however it has a bit more information, and is for encrypting a new file into the Vault (at `MainScreen.currentDir`). It has its own child popup that asks for confirmation.

I named the file `mainSmallPops` incase I added large popups that filled the entire screen, in which case they would have their own file called `mainLargePops`, just to help me distinguish between the two.

Here is the styling for `mainSmallPops` (`code/python-go/kivyStuff/kvFiles/mainScPops.kv`):

```

1 #: import Clock kivy.clock.Clock
2
3 <addNewFolderPop@Popup>:
4     title: "Add New Folder"
5     size_hint: .7, .4
6     pos_hint: {"center_x": .5, "center_y": .5}
7     auto_dismiss: True
8     GridLayout:

```

```

9     cols: 1
10    Label:
11        text: "New Folder Name:"
12    TextInput:
13        id: folderNameInput
14        size_hint: .7, .4
15        multiline: False
16        hint_text: "Name"
17        on_text_validate: root.makeFolder(self.text)
18
19 <decryptFileToLocPop@Popup>:
20     title: root.getTitle()
21     size_hint: .7, .4
22     pos_hint: {"center_x": .5, "center_y": .5}
23     auto_dismiss: True
24     GridLayout:
25         cols: 1
26         Label:
27             text: "Input the destination:"
28         TextInput:
29             id: decDirInput
30             size_hint: .7, .4
31             multiline: False
32             hint_text: "Directory:"
33             on_text_validate: root.checkCanDec(self.text)
34
35
36 <addFilePop@Popup>:
37     id: addFile
38     submitDirs: submitDirs
39     size_hint: .7, .7
40     title: "Add File"
41     FloatLayout:
42         Label:
43             size_hint: .46, .08
44             font_size: 18
45             text: "Enter the directories what files you would like to encrypt (can be a folder).\nYou can
seperate each directory with '--' if you want to do multiple locations."
46             pos_hint: {"center_x": 0.5, "y": 0.8}
47
48         Button:
49             size_hint: .16, .16
50             pos_hint: {"center_x": .5, "center_y": .2}
51             text: "Close"
52             on_release: root.dismiss()
53
54         TextInput:
55             size_hint: .9, .12
56             font_size: 22
57             hint_text: "Directories"
58             id: dirInp
59             pos_hint: {"center_x": 0.5, "center_y": 0.7}
60             multiline: False
61
62         Button:
63             id: submitDirs
64             size_hint: .16, .16
65             pos_hint: {"center_x": .5, "center_y": .4}
66             text: "Submit"
67             on_release: root.checkIfSure(root.ids.dirInp.text)
68
69
70 <ConfirmationPop@Popup>:
71     title: "Confirmation"
72     size_hint: .4, .4
73     pos_hint: {"center_x": .5, "center_y": .5}
74     auto_dismiss: False
75     FloatLayout:
76         Label:

```

```

77     text: "Are you sure?"
78     pos_hint: {"center_x": .5, "center_y": .7}
79     Button:
80         text: "No!!!"
81         pos_hint: {"center_x": .25, "center_y": .25}
82         size_hint: .4, .3
83         on_release: Clock.schedule_once(root.dismiss, -1)
84
85     Button:
86         text: "Yes"
87         pos_hint: {"center_x": .75, "center_y": .25}
88         size_hint: .4, .3
89         on_release: Clock.schedule_once(root.dismiss,
-1);root.fileScreen.outerScreen.checkCanEncrypt(root.inputText)

```

Main Screen Labels

There is only one custom Label in my program, so this should be short.

It is defined in the `MainScreen` class:

```

1 class MainScreen(Screen):
2
3     class infoLabel(Label):
4         pass

```

Here is the styling for the class (`code/python-go/kivyStuff/kvFiles/mainScLabels.kv`):

```

1 <infoLabel@Label>:
2     text_size: self.width, None
3     height: self.texture_size[1]

```

What this does is wrap the text and constrain it within its area. `infoLabel` is used for the information in the information popup, so it has to be constrained into its area. The `text_size` is set to `self.width, None` because it is in a grid layout, so the grid should change its size.

And that is everything in `MainScreen`.

Settings Screen

Here is the code for `SettingsScreen` (`code/python-go/kivyStuff/settingsScreen.py`):

```

1 from os import path, makedirs
2 from kivy.uix.popup import Popup
3 from kivy.uix.screenmanager import Screen
4
5 from configOperations import changeVaultLoc, editConfTerm
6
7 class SettingsScreen(Screen):
8
9     def __init__(self, mainScreen, configLoc, **kwargs):
10         self.outerScreen = mainScreen
11         self.config = configLoc
12         super(Screen, self).__init__(**kwargs) # Done after so that when Screen.__init__ runs, it already has those attributes.
13
14         self.ids.searchSwitch.bind(active=self.searchSwitchCallback)
15         self.ids.btSwitch.bind(active=self.btSwitchCallback)
16
17     def searchSwitchCallback(self, switch, value):

```

```

18     self.outerScreen.searchRecursively = not self.outerScreen.searchRecursively
19     return editConfTerm("searchRecursively", str(value), self.config)
20
21 def btSwitchCallback(self, switch, value):
22     self.outerScreen.useBTTemp = not self.outerScreen.useBTTemp
23     return editConfTerm("bluetooth", str(value), self.config)
24
25 def changeVault(self, inp):
26     if inp[0] != self.outerScreen.fileSep: # Relative path
27         temp = self.outerScreen.startDir.split(self.outerScreen.fileSep)
28         inp = self.outerScreen.fileSep.join(temp[:-4])+self.outerScreen.fileSep+inp
29     if inp[-1] != self.outerScreen.fileSep:
30         inp += self.outerScreen.fileSep # End with file separator
31
32
33     if inp[len(inp)-1] != self.outerScreen.fileSep:
34         inp += self.outerScreen.fileSep
35     try:
36         worked = changeVaultLoc(inp, self.outerScreen.fileSep, self.config)
37     except FileNotFoundError:
38         Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Directory not valid:\n"+inp),
39               size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()
40     except PermissionError as e:
41         Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Can't make a folder here:\n"+inp),
42               size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()
43     except Exception as e:
44         print(e)
45         Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Can't make a folder here:\n"+inp),
46               size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()
47     else:
48         if worked:
49             done = Popup(title="Done", content=self.outerScreen.infoLabel(text="Changed Vault Location
50 to:\n"+inp), size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5})
51             self.outerScreen.path = inp
52             self.outerScreen.currentDir = inp
53             self.outerScreen.recycleFolder =
54             self.outerScreen.path+self.outerScreen.recycleName+self.outerScreen.fileSep
55             done.open()
56         else:
57             Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Directory not valid:\n"+inp),
58                   size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()

```

Here is the styling for this screen:

```

1 <SettingsScreen>:
2     auto_dismiss: False
3     newLoc: newLoc
4     title: "Settings"
5     GridLayout:
6         cols: 2
7         Label:
8             canvas.before:
9                 Color:
10                    rgba: 0.33, 0.33, 0.33, 1      # Aproximately the same as kivy's default button colour.
11                 Rectangle:
12                     pos: self.pos
13                     size: self.size
14
15             text: "Settings Menu"
16             font_size: 40
17         Label:
18             canvas.before:
19                 Color:
20                    rgba: 0.33, 0.33, 0.33, 1
21                 Rectangle:
22                     pos: self.pos
23                     size: self.size

```

```

24
25
26     Label:
27         text: "Vault Location:"
28         font_size: 20
29     Label:
30         text: root.outerScreen.path
31         font_size: 14
32
33     Label:
34         text: "Change Vault Location:"
35         font_size: 16
36     GridLayout:
37         cols: 2
38         TextInput:
39             id: newLoc
40             size_hint_x: .85
41             font_size: 22
42             multiline: False
43             hint_text: "Location"
44         Button:
45             size_hint_x: .15
46             text: "Submit"
47             on_release: root.changeVault(newLoc.text)
48
49
50     Label:
51         text: "Search folders recursively\n(takes longer but searches through folders)"
52     Switch:
53         id: searchSwitch
54         active: root.outerScreen.searchRecursively
55
56     Label:
57         text: "Bluetooth login (as default)"
58     Switch:
59         id: btSwitch
60         active: root.outerScreen.useBTTemp
61
62     Label:
63     Label:
64
65     Label:
66     Label:
67
68     Label:
69         text: "Note: Changes that you make that affect the login method only take effect once the program
70         is restarted."
71         text_size: self.size
72     Label:
73
74     Button:
75         text: "Exit"
76         font_size: 20
77
78         on_release: root.manager.current = "Main"
79

```

There are a few blank labels to fill the space in-between the exit button, and the rest of the settings.

This screen is quite simple, and most of the code is managing the GUI (boring).

Mobile App GUI Code

All of the mobile code is in Python 2, other than SHA, which is Python 3.

The root of the GUI

The main file of the program is `main.py` in `code/mobile/main.py`. Here is the code of `main.py`:

```
1 from kivy.app import App
2 from kivy.uix.screenmanager import ScreenManager, Screen, FadeTransition
3 from kivy.lang import Builder
4
5 from padScreen import PadScreen
6 from mainScreen import MainScreen
7 from fileSelectionScreen import FileSelectionScreen
8
9
10 class ScreenManagement(ScreenManager):
11     pass
12
13 presentation = Builder.load_file(u"pad.kv")
14
15 class uiApp(App):
16
17     def build(self):
18         return presentation
19
20     def runUI():
21         ui = uiApp()
22         ui.run()
23
24
25 if __name__ == u"__main__":
26     runUI()
27
```

All this does it load the `pad.kv` file, which contains the styling for the entire program, and also import all of the screens.

Here is `pad.kv` (`code/mobile/pad.kv`):

```
1 #: import FadeTransition kivy.uix.screenmanager.FadeTransition
2 #: import Window kivy.core.window.Window
3 #: import Clock kivy.clock.Clock
4
5 ScreenManagement:
6     id: screenmanager
7     transition: FadeTransition()
8     PadScreen:
9         name: "Pad"
10        id: Pad
11        manager: screenmanager
12     MainScreen:
13         name: "Main"
14         id: Main
15         manager: screenmanager
16     FileSelectionScreen:
17         name: "Select"
18         id: Select
19         manager: screenmanager
20
21 <PadNum@Button>:
22     font_size: 30
23     on_release: root.root.addNum(self.text)    # = PadScreen.addnNum(self.text)
24
25 <DeviceButton@Button>:
```

```

26     font_size: 80
27     on_release: self.devicePop.setupBT(self.text)
28
29 <PadScreen>:
30     display: display
31     FloatLayout:
32         GridLayout:
33             size_hint_y: .7
34             cols: 3
35             Button:
36                 text: "7"
37                 on_release: root.addNum(self.text)
38             Button:
39                 text: "8"
40                 on_release: root.addNum(self.text)
41             Button:
42                 text: "9"
43                 on_release: root.addNum(self.text)
44
45             Button:
46                 text: "4"
47                 on_release: root.addNum(self.text)
48             Button:
49                 text: "5"
50                 on_release: root.addNum(self.text)
51             Button:
52                 text: "6"
53                 on_release: root.addNum(self.text)
54
55             Button:
56                 text: "1"
57                 on_release: root.addNum(self.text)
58             Button:
59                 text: "2"
60                 on_release: root.addNum(self.text)
61             Button:
62                 text: "3"
63                 on_release: root.addNum(self.text)
64
65
66             Button:
67                 text: "Delete"
68                 on_release: root.backSpace()
69             Button:
70                 text: "0"
71                 on_release: root.addNum(self.text)
72             Button:
73                 text: "Submit"
74                 on_release: root.confirm()
75
76     Label:
77         id: display
78         text: root.numsString
79         font_size: Window.height/20
80         pos_hint: {"center_x": .5, "y": .35}
81
82 <MainScreen>:
83     FloatLayout:
84         Label:
85             text: "To lock the Vault, close the program.\nAll downloaded files are stored in\nyour 'Download' folder."
86
87         Button:
88             pos_hint: {"x": .25, "y": .2}
89             size_hint: .5, .2
90             text: "Select files from PC"
91             on_release: root.manager.current = "Select"
92
93 <listButton@Button>:

```

```

94     size_hint: 1, None
95     on_release: root.outerScreen.selectFile(self.fileName)
96
97 <FileSelectionScreen>:
98     FloatLayout:
99         Button:
100            text: "Exit"
101            size_hint: .4, .14
102            background_color: (0.8, 0.8, 0.8, 1)
103            pos_hint: {"top": 1, "left": 1}
104            on_release: root.exit()
105
106         Button:
107            text: "Go back"
108            size_hint: .4, .14
109            background_color: (0.8, 0.8, 0.8, 1)
110            pos_hint: {"top": 1, "right": 1}
111            on_release: root.getBackDir()

```

The way the app is built is slightly different than my PC app, as the mobile app is more built around the main `kv` file. The ScreenManager (`ScreenManagement`) is set as the root widget of the `kv` file, and then the app is built from the `kv` file.

Pad Screen

Here is the code for the `PadScreen`, which also contains the 'screen' (actually a popup) that lets you select a device:

```

1  from kivy.uix.gridlayout import GridLayout
2  from kivy.uix.label import Label
3  from kivy.uix.floatlayout import FloatLayout
4  from kivy.uix.button import Button
5  from kivy.uix.popup import Popup
6  from kivy.core.window import Window
7  from kivy.uix.scrollview import ScrollView
8  from kivy.clock import Clock
9  from kivy.uix.screenmanager import Screen
10 from jnius import autoclass
11
12 from btShared import recieveFileList
13 import SHA
14
15 # Import java Bluetooth classes.
16 BluetoothAdapter = autoclass("android.bluetooth.BluetoothAdapter")
17 BluetoothDevice = autoclass("android.bluetooth.BluetoothDevice")
18 BluetoothSocket = autoclass("android.bluetooth.BluetoothSocket")
19 UUID = autoclass("java.util.UUID")
20
21 def createSocketStream(self, devName):
22     pairedDevs = BluetoothAdapter.getDefaultAdapter().getBondedDevices().toArray()
23     socket = None
24     found = False
25     for dev in pairedDevs:
26         if dev.getName() == devName:
27             socket = dev.createRfcommSocketToServiceRecord(UUID.fromString("80677070-a2f5-11e8-b568-
28 0800200c9a66")) #Random UUID from https://www.famkruithof.net/uuid/uuidgen
29             rStream = socket.getInputStream() # Stream for recieving data
30             sStream = socket.getOutputStream() #Stream for sending data
31             self.devName = devName
32             found = True
33             break #Stop when device found
34     if found:
35         socket.connect()
36         return rStream, sStream
37     else:

```

```

37         raise ConnectionAbortedError(u"Couldn't find + connect to device.")
38
39 class PadScreen(Screen, FloatLayout):
40
41     class DeviceSelectionPopup(Popup):
42
43         class DeviceButton(Button):
44
45             def __init__(self, devPopup, **kwargs):
46                 self.devicePop = devPopup
47                 super(Button, self).__init__(**kwargs)
48                 self.outerScreen = self.devicePop.outerScreen
49
50             def __init__(self, padScreen, **kwargs):
51                 self.outerScreen = padScreen
52                 super(Popup, self).__init__(**kwargs)
53                 self.devName = ""
54                 self.connected = False
55                 self.setupAll()
56
57             def setupAll(self, instance=None):
58                 paired = self.getDeviceList()
59                 if paired: # If there are paired devices.
60                     self.setupDevButtons(paired)
61                 else:
62                     grid = GridLayout(cols=1)
63                     info = Label(text="No paired devices found.\nPlease make sure your Bluetooth\\nis on, you are
in range of\\nyour device, and you are paired\\nto your device.")
64                     btn = Button(text="Retry", size_hint_y=.2)
65                     btn.bind(on_release=self.setupAll)
66                     self.content = grid # Change content of popup to the grid
67
68             def setupDevButtons(self, listOfDevs): # Similar to `createButtons` in `MainScreen` on PC app
69                 self.layout = GridLayout(cols=1, spacing=20, size_hint_y=None)
70                 self.layout.bind(minimum_height=self.layout.setter("height")) # Set due to ScrollView
71
72                 for devName in listOfDevs:
73                     btn = self.DeviceButton(self, text=devName, size_hint_y=None, height=Window.height/10,
74 halign="left", valign="middle")
75                     self.layout.add_widget(btn)
76
76                 self.view = ScrollView(size_hint=(1, 1))
77                 self.view.add_widget(self.layout)
78                 self.content = self.view
79
80             def getDeviceList(self):
81                 result = []
82                 pairedDevs = BluetoothAdapter.getDefaultAdapter().getBondedDevices().toArray()
83                 for dev in pairedDevs:
84                     result.append(dev.getName()) # Get the names of each device.
85
86             return result
87
88
89             def changeToDeviceList(self, instance=None): # has to be a function as it is bound to a button
90                 self.content = self.view
91
92             def setupBT(self, devName):
93                 try:
94                     self.outerScreen.rStream, self.outerScreen.sStream = createSocketStream(self, devName) # Create the two streams for communicating with the PC app
95                 except Exception, e:
96                     print u"Can't connect to device."
97                     self.connected = False
98                     grid = GridLayout(cols=1)
99                     info = Label(text="Can't connect to device\\nplease make sure the\\ndevice has Bluetooth on,\\nis
in range, and is\\nrunning the FileMate app.")
100                    btn = Button(text="Retry", size_hint_y=.2)
101                    btn.bind(on_press=self.changeToDeviceList)

```

```

102         grid.add_widget(info)
103         grid.add_widget(btn)
104         self.content = grid
105     else:
106         print u"Connected to:", devName
107         self.connected = True
108         self.dismiss()
109
110
111 def __init__(self, **kwargs):
112     super(PadScreen, self).__init__(**kwargs)
113     self.nums = []
114     self.numsString = u""
115     self.rStream = None
116     self.sStream = None
117     self.deviceSelection = self.DeviceSelectionPopup(self, title="Select your device:",
118     title_align="center", size_hint=(1, 1), pos_hint={"x_center": .5, "y_center": .5}, auto_dismiss=False)
119     Clock.schedule_once(self.deviceSelection.open, 0.5)
120
121
122 def addNum(self, num):
123     if len(self.nums) < 16:      # Maximum length of key is 16
124         self.nums.append(int(num))
125         self.numsString += "*"
126         self.updateDisplay()
127
128
129 def updateDisplay(self):      # Updates the Label at the top of PadScreen
130     self.ids.display.text = self.numsString    # self.numsString just contains asterisks "*"
131
132
133 def backSpace(self):        # For deleting the input
134     if len(self.nums) != 0:
135         del self.nums[-1]
136         self.numsString = self.numsString[:len(self.nums)]
137         self.updateDisplay()
138
139
140 def confirm(self):
141     pop = Popup(title="Please Wait...", content=Label(text="Waiting for confirmation."), size_hint=(1, 1),
142     pos_hint={"x_center": .5, "y_center": .5}, auto_dismiss=False)
143     if self.rStream != None and self.sStream != None: # if the connection is still up
144         self.sStream.write("{}".format("#"))    # Key sent as #<key>-
145         self.nums = SHA.getSHA128of16(self.nums)
146         for num in self.nums:
147             self.sStream.write("{},".format(num))
148             self.sStream.write("{}.".format("~"))
149             self.sStream.flush()
150             print u"Numbers sent."
151             pop.open()
152
153             data = self.rStream.read()
154             while len(str(data)) == 0:
155                 try:
156                     data = self.rStream.read()
157                 except Exception as e:
158                     print e, u"Couldn't recieve data."
159
160             print u"Out of while loop"
161             print data, u"Response"
162             if data == 49: # Response sent by the PC if the key is valid.
163                 pop.dismiss()
164                 print u"Valid"
165
166             corPop = Popup(title="Valid.", content=Label(text="Valid passcode!\nPlease leave the app open
in the background\notherwise the vault will lock."), size_hint=(.9, .5), pos_hint={"x_center": .5, "y_center": .5})
167             Clock.schedule_once(corPop.open, -1)
168
169             # Time to recieve file names of current directory
170             listOfFiles = recieveFileList(self.rStream)

```

```

167         self.manager.get_screen("Main").sStream, self.manager.get_screen("Main").rStream =
168         self.sStream, self.rStream # Hand over the streams to the other screens
169
170         self.manager.get_screen("Select").sStream, self.manager.get_screen("Select").rStream =
171         self.sStream, self.rStream
172         self.manager.get_screen("Select").fileList = listOffFiles
173
174         self.manager.current = "Main"
175
176     elif data == 48: # Response sent by the PC if code is invalid.
177         print u"Invalid."
178         pop.dismiss()
179         invPop = Popup(title="Invalid.", content=Label(text="Invalid passcode, please try again."),
180         size_hint=(.9, .5), pos_hint={"x_center": .5, "y_center": .5})
181         self.nums = []
182         self.numsString = u""
183         self.updateDisplay()
184         invPop.open()
185     else:
186         print type(data), "data was not either 49 or 48..."
187     else:
188         print u"Can't connect to device."
189         Popup(self, title="Can't connect.",
190             content=Label(text="Can't connect to device\nplease make sure the\ndevice has Bluetooth
191             on,\nis in range, and is\nrunning the FileMate program."),
192             title_align="center",
193             size_hint=(.6, .6),
194             pos_hint={"x_center": .5, "y_center": .5},
195             auto_dismiss=True).open()
196
197         self.deviceSelection.open()

```

The class `DeviceSelectionPopup` is inside of `PadScreen`, because it is only ever needed by `PadScreen`, and shouldn't exist unless `PadScreen` exists. The same thing applies with `DeviceButton`.

As soon as the screen opens, the `DeviceSelectionPopup` is opened, as it is needed at start up.

Most of this is just GUI.

Main Screen

This is the home screen of the app, although it is extremely basic (literally a Label and a Button).

Here is the code (`code/mobile/mainScreen.py`):

```

1  from kivy.uix.label import Label
2  from kivy.uix.floatlayout import FloatLayout
3  from kivy.uix.popup import Popup
4  from kivy.uix.screenmanager import Screen
5  from time import sleep
6
7  from btShared import recieveFile
8
9  class MainScreen(Screen, FloatLayout):
10
11     def __init__(self, **kwargs):
12         super(MainScreen, self).__init__(**kwargs)
13         self.sStream = None
14         self.rStream = None

```

Modules used in the `.kv` file, that are not from Kivy, have to be imported here.

File Selection Screen & Shared Bluetooth functions

This screen is where the can browse the folders in the Vault, or download files from it instead.

I have used the protocol in the **Design** section to recieve both the list of files and to download files.

Here is the code for recieving a list of files, and for recieving a file (`code/mobile/btShared.py`):

```
1  from plyer import storagepath
2  from os import remove
3  from kivy.uix.label import Label
4  from kivy.uix.popup import Popup
5
6  #Shared methods
7  def recieveFileList(rStream, buffAlreadyKnown=[]):
8      buff = buffAlreadyKnown    # If called from other places, some of they data may already
9      data = ""
10
11     endList = [126, 33, 33, 69, 78, 68, 76, 73, 83, 84, 33]           #~!!ENDLIST!
12
13     while buff[-11:] != endList:      # If last 11 elements of the buffer is ~!!ENDLIST!
14         try:
15             data = rStream.read()
16         except Exception as e:
17             print e, "Failed while getting file list."
18             break
19         else:
20             buff.append(data)
21
22     buff = buff[10:-11] # Get the actuall list of files from the buffer
23
24     listOfFiles = "".join([chr(i) for i in buff]) # Join input into a string
25     listOfFiles = listOfFiles.split("--")[1:] # First element will be "" due to first part of string being "--"
26
27     print "List of files given:", listOfFiles
28     return listOfFiles
29
30 def recieveFile(rStream, buffAlreadyKnown=[]):
31     print("Recieve file has been called.")
32     # File is sent with !NAME!<name here>~~!~<data>~!!ENDF! like a data sandwich.
33     # To do: make dictionary with each nameInstruction, startHeader etc, so they can be
34     # easily identified.
35     downloadsDir = storagepath.get_downloads_dir()
36
37     buff = buffAlreadyKnown
38     data = ""
39     nameInstruction = [33, 78, 65, 77, 69, 33]           # !NAME!
40     endFile      = [126, 33, 69, 78, 68, 70, 73, 76, 69, 33] # ~!ENDFILE!
41     separator     = [126, 126, 33, 126, 126]            # ~~!~
42     nameFound = False
43     name = []
44     fo, fw = None, None
45     fileName = ""
46     bufferSize = 1024
47     buffCount = 0
48
49     while len(str(data)) > -1: # While connection is open
50         try:
51             data = rStream.read() # Read from the recieving stream
52         except Exception as e:
53             print e, u"Failed recieving file."
54             if buffCount > 0: # Clean up the file if it has been edited
55                 fo.close()
56                 remove(downloadsDir+"/"+fileName) # Remove incomplete file. (from os module)
57             return False
58         else:
59             buff.append(data)
60
```

```

61     if not nameFound:
62         name = []
63         for i in range(len(buff)-6): # -6 because that is the length of nameInstruction (scan is 6 wide)
64             if buff[i:i+6] == nameInstruction: # Are these 6 items the same as nameInstruction
65                 z = i+6 # Move past nameInstruction
66                 while (buff[z:z+5] != separator) and (z+5 < len(buff)): # Scans current buffer for the
67                     name.append(buff[z])
68                     z += 1
69
70             if buff[z:z+5] == separator: # Once you get to the separator, then you know the name has
71                 been received.
72                 nameFound = True # Name has been found
73                 buff[i:z+5] = [] # Clear name + separator
74
75                 for letter in name:
76                     fileName += chr(letter)
77
78                 fo = open(downloadsDir+"/"+fileName, "wb") # Open for writing
79
80             elif ((len(buff) > bufferSize+10) or (buff[-10:] == endFile)): # If end of file header found
81                 if buff[-10:] == endFile:
82                     buff[-10:] = []
83                     print u"End found"
84                     fo.write(bytarray(buff))
85                     fo.close()
86
87                     pop = Popup(title="Success!", content=Label(text="File received successfully.\nYou can find your
88 file in\nyour 'Download' folder."), pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.7, .4))
89                     pop.open()
90                     return True
91
92             else:
93                 fo.write(bytarray(buff[:bufferSize]))
94                 buff[:bufferSize] = []
95                 buffCount += bufferSize

```

`recieveFileList` is called when `!FILELIST!` is received while changing directory, or when first entering the screen. It waits for `~!ENDLIST!`, and then joins the contents of the data received (that is not the start header) into a string, then splits the string by "--" to get the list of files. The first element of this list will be `[""]`, because the file list looks like this: `--item1--item2`.

`recieveFile` is called when `!NAME!` is received while waiting for a response from `!FILESELECT!` command (sent when a file/folder is selected while browsing the files), and the response will be `!FILELIST!` if the item was a folder (containing list of items in that folder), or `!NAME!` followed by the file's data (if the item was a file).

Moving on to the file selection screen (`FileSelectionScreen` at `code/mobile/fileSelectionScreen.py`).

Here is the code:

```

1  from kivy.uix.gridlayout import GridLayout
2  from kivy.uix.floatlayout import FloatLayout
3  from kivy.uix.button import Button
4  from kivy.core.window import Window
5  from kivy.uix.scrollview import ScrollView
6  from kivy.uix.screenmanager import Screen
7
8  from btShared import recieveFileList, recieveFile
9
10 class FileSelectionScreen(Screen, FloatLayout):
11
12     class listButton(Button):

```

```

14     def __init__(self, mainScreen, fileName, **kwargs):
15         super(Button, self).__init__(**kwargs)
16         self.outerScreen = mainScreen
17         self.fileName = fileName
18
19
20     def __init__(self, **kwargs):
21         super(FileSelectionScreen, self).__init__(**kwargs)
22         self.sStream = None
23         self.rStream = None
24         self.fileList = []
25
26         # List of possible responses
27         self.endOfTreeResponse = [33, 69, 78, 68, 79, 70, 84, 82, 69, 69, 33] # !ENDOFTREE!
28         self.startList = [33, 70, 73, 76, 69, 76, 73, 83, 84, 33] # !FILELIST!
29         self.nameInstruction = [33, 78, 65, 77, 69, 33] # !NAME! --Start of a file
30         self.fileNotFound = [33, 78, 79, 84, 70, 79, 85, 78, 68, 33] # !NOTFOUND! --Response to
file selection
31
32
33     def on_enter(self):
34         self.createButtons(self.fileList)
35
36     def on_leave(self):
37         self.removeButtons()
38
39     def removeButtons(self): # Clears all the widgets off the screen
40         self.grid.clear_widgets()
41         self.scroll.clear_widgets()
42         self.grid = 0
43         try:
44             self.remove_widget(self.scroll)
45         except Exception as e:
46             print e, u"Already removed?"
47         self.scroll = 0
48
49     def createButtons(self, array): # Similar to createButtons on the PC app
50         self.grid = GridLayout(cols=1, size_hint_y=None) # Added in case I need to add more columns in the
future (file size etc)
51         self.grid.bind(minimum_height=self.grid.setter("height"))
52         for item in array:
53             btn = self.listButton(self, item, text=" "+str(item)), height=Window.height/10, halign="left",
valign="middle")
54             btn.bind(size=btn.setter("text_size"))
55             self.grid.add_widget(btn)
56
57         self.scroll = ScrollView(size_hint=(1, .86))
58         self.scroll.add_widget(self.grid)
59         self.add_widget(self.scroll)
60
61     def recreateButtons(self, array):
62         self.removeButtons()
63         self.createButtons(array)
64
65     def selectFile(self, fileName):
66         # File request looks like: !FILESELECT!<name here>~!ENDSELECT!
67         msg = [33, 70, 73, 76, 69, 83, 69, 76, 69, 67, 84, 33] # !FILESELECT!
68
69         for letter in fileName:
70             msg.append(ord(letter))
71
72         msg += [126, 33, 69, 78, 68, 83, 69, 76, 69, 67, 84, 33] # End header: ~!ENDSELECT!
73
74         self.sStream.flush() # Clear write buffer on data stream.
75
76         for i in msg:
77             self.sStream.write(i)
78
79         # Get response

```

```

80     buff = []
81     data = ""
82     responseFound = False
83     print u"Waiting for response"
84
85     while not responseFound:
86         try:
87             data = self.rStream.read()
88         except Exception as e:
89             print e, "Failed receiving response to select file."
90             return False
91         else:
92             buff.append(data)
93
94         if (buff[:6] == self.nameInstruction) and (len(buff) >= 6): # If the response is !NAME!, then it
95             was a file and will be sent to this program
96             print u"Is name instruction"
97             recieveFile(self.rStream, buff)
98             responseFound = True
99             buff = []
100
101        elif (buff[:10] == self.fileNotFound) and (len(buff) >= 10): # If the response was !NOTFOUND! then
102            the host couldn't find the item we wanted
103            print u"Response is fileNotFound."
104            raise ValueError("File was not found by host.")
105            responseFound = True
106            buff = []
107
108        elif (buff[:10] == self.startList) and (len(buff) >= 10): # If the response was !FILELIST! then
109            it was a folder, so prepare to receive the list of files in that folder.
110            print u"Response is a file list."
111            self.fileList = recieveFileList(self.rStream, buff)
112            responseFound = True
113            self.recreateButtons(self.fileList)
114
115
116    elif len(buff) >= 13: # Reset the buffer and wait for next command
117        print u"Didn't get response :(", buff
118        buff = []
119
120
121    def getBackDir(self):
122        # Back dir request looks like: !BACK!
123        backCommand = [33, 66, 65, 67, 75, 33]
124        self.sStream.flush() # Clear the buffer for sending
125        for i in backCommand:
126            self.sStream.write(i)
127
128        data = ""
129        buff = []
130        responseFound = False
131        while not responseFound:
132            try:
133                data = self.rStream.read()
134            except Exception as e:
135                print e, "Failed receiving server response to BACK request."
136                return False
137            else:
138                buff.append(data)
139
140            if (buff[:11] == self.endOfTreeResponse) and (len(buff) >= 11): # If you cannot go further back
141                in the directory, !ENDOTREE! is sent from the PC
142                print "END OF TREE"
143                buff = []
144                responseFound = True
145
146            elif (buff[:10] == self.startList) and (len(buff) >= 10): # Otherwise, it should be a list of
147                new file names
148                responseFound = True
149                self.fileList = recieveFileList(self.rStream, buff)

```

```

144         self.recreateButtons(self.fileList)
145
146     elif len(buff) >= 11:
147         print "start header not found yet :(", buff
148         buff = []

```

Again like `PadScreen`, this screen has an embedded class. `listButton`'s are displayed showing the contents of the current directory.

`selectFile` is called when a `listButton` is pressed, and requests to navigate that button.

`getBackDir` is called when the back button is pressed, and requests the list of items that are in the directory above the one we are currently in.

SHA

SHA is the exact same on the mobile app, as it is on the PC program.

What is buildozer.spec?

`buildozer.spec` is used for building the app and putting it onto the mobile device, and is unrelated to the rest of the code.

For transparency, here is the contents:

```

1 [app]
2
3 # (str) Title of your application
4 title = FM Pad
5 # (str) Package name
6 package.name = PadApp
7
8 # (str) Package domain (needed for android/ios packaging)
9 package.domain = org.test
10
11 # (str) Source code where the main.py live
12 source.dir = .
13
14 # (list) Source files to include (let empty to include all the files)
15 source.include_exts = py,png,jpg,kv,atlas
16
17 # (str) Application versioning (method 1)
18 version = 0.1
19
20 # (str) Application versioning (method 2)
21 # version.regex = __version__ = ['"'](.*)['"]"
22 # version.filename = %(source.dir)s/main.py
23
24 # (list) Application requirements
25 # comma separated e.g. requirements = sqlite3,kivy
26 requirements = python2,android,plyer,pyjnius,kivy
27
28 # (str) Custom source folders for requirements
29 # Sets custom source for any requirements with recipes
30 # requirements.source.kivy = ../../kivy
31
32 # (list) Garden requirements
33 #garden_requirements =
34
35 # (str) Presplash of the application
36 #presplash.filename = %(source.dir)s/data/presplash.png
37
38 # (str) Icon of the application

```

```

39 #icon.filename = %(source.dir)s/data/icon.png
40
41 # (str) Supported orientation (one of landscape, portrait or all)
42 orientation = portrait
43
44 # (list) List of service to declare
45 #services = NAME:ENTRYPOINT_TO_PY,NAME2:ENTRYPOINT2_TO_PY
46
47 #
48 # OSX Specific
49 #
50
51 #
52 # author = © Copyright Info
53
54 # change the major version of python used by the app
55 osx.python_version = 3
56
57 # Kivy version to use
58 osx.kivy_version = 1.10.1
59
60 #
61 # Android specific
62 #
63
64 # (bool) Indicate if the application should be fullscreen or not
65 fullscreen = 0
66
67 # (string) Presplash background color (for new android toolchain)
68 # Supported formats are: #RRGGBB #AARRGGBB or one of the following names:
69 # red, blue, green, black, white, gray, cyan, magenta, yellow, lightgray,
70 # darkgray, grey, lightgrey, darkgrey, aqua, fuchsia, lime, maroon, navy,
71 # olive, purple, silver, teal.
72 #android.presplash_color = #FFFFFF
73
74 # (list) Permissions
75 android.permissions = BLUETOOTH,BLUETOOTH_ADMIN,BLUETOOTH_PRIVILEGED
76
77 [buildozer]
78
79 # (int) Log level (0 = error only, 1 = info, 2 = debug (with command output))
80 log_level = 2
81
82 # (int) Display warning if buildozer is run as root (0 = False, 1 = True)
83 warn_on_root = 1
84
85 # (str) Path to build artifact storage, absolute or relative to spec file
86 build_dir = /home/kivy/VMOUT/

```

And that is all of the code.

Testing

For testing my program, I will mostly be doing black-box testing, with some unit testing of important functions.

Unit Tests

The unit tests are only for the Go programs, as the `"testing"` package for Go makes testing very easy. Testing files in Go are made with the file name `<name-here>.test.go`. You may have noticed this in the **File Structure** section.

AES

To test AES, there are multiple `*_test.go` files, which I will go through individually. I will go through the results of the tests once I have gone through how I am doing them.

Testing the core algorithm

To test the core algorithm, I made tests for each of the functions in `code/python-go/AES/src/AES/aes.go` using test vectors that I worked out by hand (easy ones like shift rows), and some from official documentation or Wikipedia (The American government is shut down at the moment so the NIST, the National Institute of Standards and Technology (they do all the SHA contests and AES contests), website is down, so official test vectors for AES were not available, apart from the MixColumns step which has some on Wikipedia (https://en.wikipedia.org/wiki/Rijndael_MixColumns#TestVectors)).

Anyway, here is `aes_test.go` in `code/python-go/AES/src/AES/aes_test.go`:

```
1 package AES
2
3 import (
4     "fmt"
5     "testing"
6 )
7
8 func TestSubBytes(t *testing.T) {
9     testVector := []byte{ 76, 201, 25, 213,
10                         255, 41, 13, 94,
11                         44, 14, 215, 104,
12                         168, 101, 19, 89} // Randomly generated by Python
13
14     expectedOutcome := fmt.Sprintf("%x", []byte{41, 221, 212, 3,
15                                         22, 165, 215, 88,
16                                         113, 171, 14, 69,
17                                         194, 77, 125, 203})
18     subBytes(testVector)
19     actualOutcome := fmt.Sprintf("%x", testVector)
20     if actualOutcome != expectedOutcome {
21         t.Fatalf("Expected %s but got %s", expectedOutcome, actualOutcome)
22     }
23 }
24
25 func TestInvSubBytes(t *testing.T) {
26     testVector := []byte{41, 221, 212, 3,
27                         22, 165, 215, 88,
28                         113, 171, 14, 69,
29                         194, 77, 125, 203}
30
31     expectedOutcome := fmt.Sprintf("%x", []byte{ 76, 201, 25, 213,
32                                         255, 41, 13, 94,
33                                         44, 14, 215, 104,
34                                         168, 101, 19, 89})
35     invSubBytes(testVector)
36     actualOutcome := fmt.Sprintf("%x", testVector)
37     if actualOutcome != expectedOutcome {
38         t.Fatalf("Expected %s but got %s", expectedOutcome, actualOutcome)
39     }
40 }
41
42 func TestShiftRows(t *testing.T) {
43     testVector := []byte{ 76, 201, 25, 213, // Taken from TestSubBytes
44                         255, 41, 13, 94,
45                         44, 14, 215, 104,
46                         168, 101, 19, 89}
47 }
```

```

48     expectedOutcome := fmt.Sprintf("%x", []byte{ 76, 41, 215, 89,
49                                         255, 14, 19, 213,
50                                         44, 101, 25, 94,
51                                         168, 201, 13, 104})
52
53     shiftRows(testVector) // Done by reference
54     actualOutcome := fmt.Sprintf("%x", testVector)
55     if actualOutcome != expectedOutcome {
56         t.Fatalf("Expected %s but got %s", expectedOutcome, actualOutcome)
57     }
58 }
59
60 func TestInvShiftRows(t *testing.T) {
61     testVector := []byte{ 76, 41, 215, 89, // Taken from TestSubBytes
62                         255, 14, 19, 213,
63                         44, 101, 25, 94,
64                         168, 201, 13, 104}
65
66     expectedOutcome := fmt.Sprintf("%x", []byte{ 76, 201, 25, 213,
67                                         255, 41, 13, 94,
68                                         44, 14, 215, 104,
69                                         168, 101, 19, 89})
70     invShiftRows(testVector)
71     actualOutcome := fmt.Sprintf("%x", testVector)
72     if actualOutcome != expectedOutcome {
73         t.Fatalf("Expected %s but got %s", expectedOutcome, actualOutcome)
74     }
75 }
76
77 func TestMixColumns(t *testing.T) {
78     testVector := []byte{0xdb, 0x13, 0x53, 0x45, // Different test vectors from
79                         // https://en.wikipedia.org/wiki/Rijndael_MixColumns
80                         0xf2, 0xa, 0x22, 0x5c,
81                         0x01, 0x01, 0x01, 0x01,
82                         0x2d, 0x26, 0x31, 0x4c}
83
84     expectedOutcome := fmt.Sprintf("%x", []byte{0x8e, 0x4d, 0xa1, 0xbc,
85                                         0x9f, 0xdc, 0x58, 0x9d,
86                                         0x01, 0x01, 0x01, 0x01,
87                                         0x4d, 0x7e, 0xbd, 0xf8})
88
89     mixColumns(testVector)
90     actualOutcome := fmt.Sprintf("%x", testVector)
91     if actualOutcome != expectedOutcome {
92         t.Fatalf("Expected %s but got %s", expectedOutcome, actualOutcome)
93     }
94 }
95
96 func testInvMixColumns(t *testing.T) {
97     testVector := []byte{0x8e, 0x4d, 0xa1, 0xbc,
98                         0x9f, 0xdc, 0x58, 0x9d,
99                         0x01, 0x01, 0x01, 0x01,
100                        0x4d, 0x7e, 0xbd, 0xf8}
101
102    expectedOutcome := fmt.Sprintf("%x", []byte{0xdb, 0x13, 0x53, 0x45,
103                                         0xf2, 0xa, 0x22, 0x5c,
104                                         0x01, 0x01, 0x01, 0x01,
105                                         0x2d, 0x26, 0x31, 0x4c})
106
107    invMixColumns(testVector)
108    actualOutcome := fmt.Sprintf("%x", testVector)
109    if actualOutcome != expectedOutcome {
110        t.Fatalf("Expected %s but got %s", expectedOutcome, actualOutcome)
111    }
112 }
113
114 func TestKeyExpansionCore(t *testing.T) {
115     testVector := [4]byte{1, 2, 3, 4}
116     // 2, 3, 4, 1
117     // sBox[2], sBox[3], sBox[4], sBox[1]

```



```

159 0xf3, 0x37, 0x7f, 0xa2, 0x59, 0xba, 0x25,
160 0xa9, 0x80, 0x19, 0xc0, 0xf0, 0x3a, 0x3c,
161 0x1b, 0xbe, 0xf5, 0x61, 0xeb, 0x84, 0xc9,
162 0x1e, 0xd4, 0x7c, 0xb2, 0xf5, 0x50, 0xb5,
163 0xff, 0x3f, 0xa7, 0x04, 0x0a, 0x6f, 0x12,
164 0x52, 0x9c, 0xdc, 0x27, 0x58, 0xf3, 0xce,
165 0xbe, 0xfc, 0xcb, 0x10, 0xe6, 0x0f, 0x05,
166 0x65, 0x34, 0x7a, 0x0a, 0x83, 0x3b, 0x7f})
167
168 outcome00 := fmt.Sprintf("%x", ExpandKey(testVector00))
169 if outcome00 != expectedOutcome00 {
170     t.Fatalf("00 Expected %s but got %s", expectedOutcome00, outcome00)
171 }
172
173 outcome1234 := fmt.Sprintf("%x", ExpandKey(testVector1234))
174 if outcome1234 != expectedOutcome1234 {
175     t.Fatalf("1234 Expected %s but got %s", expectedOutcome1234, outcome1234)
176 }
177
178 outcomeff := fmt.Sprintf("%x", ExpandKey(testVectorff))
179 if outcomeff != expectedOutcomeff {
180     t.Fatalf("ff Expected %s but got %s", expectedOutcomeff, outcomeff)
181 }
182 }
```

Benchmarks are run multiple times, and get the time taken for each run, and the amount of bytes processed and the number of variable allocations made.

The test vectors for `ExpandKey` were calculated using an old version of AES I made in Python while I was making my project, and I have verified it using this website <http://aes.online-domain-tools.com/>, just to check it encrypts data correctly (it was fine).

Testing AESfiles

Here is the test file for the `AESfiles` package (`code/python-go/AES/src/AES/AESfiles/aesFiles_test.go`):

```

1 package AESfiles
2
3 import (
4     "fmt"
5     "log"
6     "testing"
7     "os/exec"
8     "strings"
9     "AES"
10 )
11
12 const (
13     largeFile = "/home/josh/GentooMin.iso"
14     largeFileTemp = "/home/josh/temp"
15     largeFileDec = "/home/josh/decTemp.iso"
16
17     mediumFile = "/home/josh/8k.png"
18     mediumFileTemp = "/home/josh/8kTemp"
19     mediumFileDec = "/home/josh/8kDec.png"
20 )
```

```

21     smallFile = "/home/josh/a.txt"
22     smallFileTemp = "/home/josh/temp2"
23     smallFileDec = "/home/josh/smallDec.txt"
24 )
25
26 var (
27     expandedKey = AES.ExpandKey([]byte{0x00, 0x0b, 0x16, 0x1d, 0x2c, 0x27, 0x3a, 0x31, 0x58, 0x53, 0x4e, 0x45,
28     0x74, 0x7f, 0x62, 0x69})
29 )
30
31 func BenchmarkEncryptFileLarge(b *testing.B) {
32     for n := 0; n < b.N; n++ {
33         EncryptFile(&expandedKey, largeFile, largeFileTemp)
34     }
35 }
36
37 func BenchmarkDecryptFileLarge(b *testing.B) {
38     for n := 0; n < b.N; n++ {
39         DecryptFile(&expandedKey, largeFileTemp, largeFileDec)
40     }
41 }
42
43 func BenchmarkEncryptFileMedium(b *testing.B) {
44     for n := 0; n < b.N; n++ {
45         EncryptFile(&expandedKey, mediumFile, mediumFileTemp)
46     }
47 }
48
49 func BenchmarkDecryptFileMedium(b *testing.B) {
50     for n := 0; n < b.N; n++ {
51         DecryptFile(&expandedKey, mediumFileTemp, mediumFileDec)
52     }
53 }
54
55 func TestEncDecMediumFile(t *testing.T) {
56     out, err := exec.Command("/bin/bash", "-c", "b2sum '"+mediumFile+"'").Output() // Gets hash of original file
57     using the b2sum utility in the GNU core utils
58     if err != nil { panic(err) }
59
60     initialHash := strings.Replace(fmt.Sprintf("%s", out), mediumFile, "", -1) // b2sum outputs the dir after the
61     checksum is output, so remove the dir.
62     log.Output(0, fmt.Sprintf("initialHash: %s", initialHash))
63
64     EncryptFile(&expandedKey, mediumFile, mediumFileTemp)
65     DecryptFile(&expandedKey, mediumFileTemp, mediumFileDec)
66
67     out, err = exec.Command("/bin/bash", "-c", "b2sum '"+mediumFileDec+"'").Output()
68     finalHash := strings.Replace(fmt.Sprintf("%s", out), mediumFileDec, "", -1)
69     log.Output(0, fmt.Sprintf("finalHash: %s", finalHash))
70
71     if finalHash != initialHash {
72         t.Fatalf("Expected %s but got %s", initialHash, finalHash)
73     }
74 }
75
76
77 func TestEncDecSmallFile(t *testing.T) {
78     out, err := exec.Command("/bin/bash", "-c", "b2sum '"+smallFile+"'").Output() // Gets hash of original file
79     using the b2sum utility in the GNU core utils
80     if err != nil { panic(err) }
81
82     initialHash := strings.Replace(fmt.Sprintf("%s", out), smallFile, "", -1) // b2sum outputs the dir after the
83     checksum is output, so remove the dir.
84     log.Output(0, fmt.Sprintf("initialHash: %s", initialHash))
85
86     EncryptFile(&expandedKey, smallFile, smallFileTemp)
87     DecryptFile(&expandedKey, smallFileTemp, smallFileDec)
88
89     out, err = exec.Command("/bin/bash", "-c", "b2sum '"+smallFileDec+"'").Output()

```

```

85     finalHash := strings.Replace(fmt.Sprintf("%s", out), smallFileDec, "", -1)
86     log.Output(0, fmt.Sprintf("finalHash: %s", finalHash))
87
88     if finalHash != initialHash {
89         t.Fatalf("Expected %s but got %s", initialHash, finalHash)
90     }
91 }
```

When testing enc/decryption of files, I will get the blake checksum of the file before and after encryption and decryption (one after the other, not separate) using the `b2sum` utility included in the GNU core utilities on Linux. That is what line 55, 64, 74 and 83 are for. The output of `b2sum` looks like `<checksum here> <file name done>`, so the last part has to be removed (as the file will be decrypted to a different file), which is what `strings` and `fmt` are needed for.

The checksums are also printed to the terminal using `log.Output`, so I can compare them myself too, in case of a bug with the test.

After decryption of the encrypted file, the checksums are compared, and if they are different then the test has failed.

That's it for the code of the AES unit tests.

Running the tests

To run the tests I have made a few bash scripts that just run a set of commands so that I don't have to remember them. These are `testAES.sh` and `testBenchAES.sh`, both in `code/python-go/AES/`.

Here is `testAES.sh`:

```

1  #!/bin/bash
2  export GOPATH="/home/josh/nea-12ColcloughJ/code/python-go/AES/"
3  cd src/AES && go test -v
4  cd AESfiles && go test -v
```

`-v` means verbose output, so Go shows each of the tests individually and what it is doing. The last two lines just change directories to where the tests are relative to `code/python-go/AES/`, as otherwise Go can't find the test files.

`testBenchAES.sh` is the same, but it runs benchmarks after doing the tests. Here it is:

```

1  #!/bin/bash
2  export GOPATH="/home/josh/nea-12ColcloughJ/code/python-go/AES/"
3  cd src/AES && go test -v -bench=. -benchmem
4  cd AESfiles && go test -v -bench=. -benchmem
```

`-bench=.` tells Go to run all benchmarks that match a regular expression. A `.` just says that it can be anything. `-benchmem` outputs how many bytes were processed during the operation, and how many variable allocations were made.

Here are the results of the tests (running `testBenchAES.sh`):

```

1  === RUN   TestSubBytes
2  --- PASS: TestSubBytes (0.00s)
3  === RUN   TestInvSubBytes
4  --- PASS: TestInvSubBytes (0.00s)
5  === RUN   TestShiftRows
6  --- PASS: TestShiftRows (0.00s)
```

```

7  === RUN  TestInvShiftRows
8  --- PASS: TestInvShiftRows (0.00s)
9  === RUN  TestMixColumns
10 --- PASS: TestMixColumns (0.00s)
11 === RUN  TestKeyExpansionCore
12 --- PASS: TestKeyExpansionCore (0.00s)
13 === RUN  TestKeyExpansion
14 --- PASS: TestKeyExpansion (0.00s)
15 PASS
16 ok    AES 0.001s
17 === RUN  TestEncDecMediumFile
18 2019/01/04 12:06:00 initialHash:
9336f1fc79ff254c8644f1df83fccfd576388c5075754ae9142c2640dd9530c5e08403937296afc7ca8138cedaf0794117e78d247ce6949
ee40304284c39b7ed
19 2019/01/04 12:06:01 finalHash:
9336f1fc79ff254c8644f1df83fccfd576388c5075754ae9142c2640dd9530c5e08403937296afc7ca8138cedaf0794117e78d247ce6949
ee40304284c39b7ed
20 --- PASS: TestEncDecMediumFile (0.91s)
21 === RUN  TestEncDecSmallFile
22 2019/01/04 12:06:01 initialHash:
0c8b815719e57e1e3529a20df69c31d8cabab90212bce1b7a3296c8f70f307d1a7e8e64ed5bb17fc7b7f3b9cc3150e4f59c4a70922a0d3b
46bfda1652e735588
23 2019/01/04 12:06:01 finalHash:
0c8b815719e57e1e3529a20df69c31d8cabab90212bce1b7a3296c8f70f307d1a7e8e64ed5bb17fc7b7f3b9cc3150e4f59c4a70922a0d3b
46bfda1652e735588
24 --- PASS: TestEncDecSmallFile (0.01s)
25 goos: linux
26 goarch: amd64
27 pkg: AES/AESfiles
28 BenchmarkEncryptFileLarge-4          1  1871293107 ns/op  244325952 B/op      3790 allocs/op
29 BenchmarkDecryptFileLarge-4         1  2042360853 ns/op  244320832 B/op      3757 allocs/op
30 BenchmarkEncryptFileMedium-4        3  378539062 ns/op   48565877 B/op      769 allocs/op
31 BenchmarkDecryptFileMedium-4        3  417413577 ns/op   48491632 B/op      768 allocs/op
32 PASS
33 ok    AES/AESfiles  9.640s

```

Every test passed, and I can now calculate the speed of encryption and decryption using the time taken on average for each operation (ns/op). I made a small Python script to do this (it isn't part of my project):

```

1 def getGoodUnit(bytes):      #Get a good unit for displaying the sizes of files.
2     if bytes == " -":
3         return " -"
4     else:
5         divCount = 0
6         divisions = {0: "B", 1: "KB", 2: "MB", 3: "GB", 4: "TB", 5: "PB"}
7         while bytes > 1000:
8             bytes = bytes/1000
9             divCount += 1
10
11     return ("%.2f" % bytes) + " " + divisions[divCount]
12
13 def calc(time, data):
14     time = time * (10**-9) # 10^-9 because the time is in nano seconds.
15     datTime = data/time
16     return getGoodUnit(datTime)
17
18 print(calc(float(input("Time taken: ")), int(input("Num of bytes: "))) + "/s")

```

It steals the `getGoodUnit` function from my project. There is no error checking because I am the only person who will be using it, and also I am lazy.

The result (on an i5-6600k - 4 cores) is that when encrypting, the speed was `129.97 MB/s`, and when decrypting the speed was `119.63 MB/s` which is slightly slower (`10.34 MB/s`), probably because it has to check for padding. Before I changed my code to pass by reference, and also before I made AES parallel using goroutines, it had a performance of `18.92 MB/s` when encrypting. If I compare the single core performance (by dividing `129.97` by 4 (4 cores)), the single core performance of the new AES is `32.49 MB/s`, which is 1.72 times as fast, just by passing by reference.

BLAKE

All of the bash files for BLAKE are the same as AES with minor changes.

Like AES, BLAKE also has some unit tests. There is only one test file (`code/python-go/BLAKE/src/BLAKE/blake_test.go`):

```

1 package BLAKE
2
3 import (
4     "fmt"
5     "testing"
6 )
7
8 const largeFile = "/home/josh/GentooMin.iso"
9 const smallFile = "/home/josh/a.txt"
10
11 func BenchmarkBLAKELarge(b *testing.B) {
12     for n := 0; n < b.N; n++ {
13         GetChecksum(largeFile, 64)
14     }
15 }
16
17 func BenchmarkBLAKESmall(b *testing.B) {
18     for n := 0; n < b.N; n++ {
19         GetChecksum(smallFile, 64)
20     }
21 }
22
23 func TestBLAKELarge(t *testing.T) {
24     actualResult := fmt.Sprintf("%x", GetChecksum(largeFile, 64))
25     expectedResult :=
26 "640ac216c91f85d69b450b070828b0f2f54db51af3ecf1daffead657ae1a8d4e5732b4594f936c9d2d853ee12a1df58e6fa63535c1ed
27 3e170e9578da740e5d" // Obtained using b2sum tool in GNU core utilities
28
29     if actualResult != expectedResult {
30         t.Fatalf("Expected %s but got %s", expectedResult, actualResult)
31     }
32 }
33
34 func TestBLAKESmall(t *testing.T) {
35     actualResult := fmt.Sprintf("%x", GetChecksum(smallFile, 64))
36     expectedResult :=
37 "f545377b0ab74d283ff65ec5518bc00633d46125ec28bbd11f417da16949e8937759d8f1aa97556845e24edc676d8f288d49aae1bb195a
38 12e5595525713427c4"
39
40     if actualResult != expectedResult {
41         t.Fatalf("Expected %s but got %s", expectedResult, actualResult)
42     }
43 }
44
45 func TestBlakeMix(t *testing.T) {
46     arr := []uint64{0x70f3abeaaf82f2d2, 0x48221449c090c5cb, 0x1631bc17f31ed4ef, 0xdf4a6c4edfb6012e,
47                 0x4011c31f6e4fbe98, 0xec6fe4ecfb235d, 0xc732f6aacdfed23, 0x6c59d3af929a71a7,
48                 0x726e4d94d076d220, 0xd5261f05e988e99, 0x47c5fed6073ffff6f, 0xafdc6ec84b5122fd,
49                 0x828789a16ddf8fcc, 0x9219a1cce5eaceb3, 0x81c389d523f73c81, 0xc518f0411804f255} // Randomly
50     generated in python.
51 }
```

```

47 // Tested using working version
48 expectedResult := []uint64{0x24f71c9177a10424, 0x48221449c090c5cb, 0x1631bc17f31ed4ef, 0xdf4a6c4edfb6012e,
49                                         0x216426a085db4092, 0xecabfe4ecfb235d, 0xc732f6aacdfed23, 0x6c59d3af929a71a7,
50                                         0xaa69fe02ab573a54, 0xd55261f05e988e99, 0x47c5fed6073fff6f, 0xafdc6ec84b5122fd,
51                                         0x9771847d3761d4df, 0x9219a1cce5eaceb3, 0x81c389d523f73c81, 0xc518f0411804f255}
52
53 blakeMix(arr, 0, 4, 8, 12, &arr[0], &arr[1]) // Simulates first mix of the compression function.
54
55 if fmt.Sprintf("%x", arr) != fmt.Sprintf("%x", expectedResult) {
56     t.Fatalf("Expected %x but got %x", expectedResult, arr)
57 }
58 }
```

For most of the tests, the result of the command `b2sum` is hard coded rather than retrieved from the terminal, as I was having issues comparing the output of BLAKE and the `b2sum` command.

Here are the test results:

```

1 === RUN   TestBLAKELarge
2 --- PASS: TestBLAKELarge (1.68s)
3 === RUN   TestBLAKESmall
4 --- PASS: TestBLAKESmall (0.00s)
5 === RUN   TestBlakeMix
6 --- PASS: TestBlakeMix (0.00s)
7 goos: linux
8 goarch: amd64
9 pkg: BLAKE
10 BenchmarkBLAKELarge-4          1  1676409286 ns/op  2198866712 B/op    7476 allocs/op
11 BenchmarkBLAKESmall-4         300000      5282 ns/op      3264 B/op    10 allocs/op
12 PASS
13 ok    BLAKE 5.004s
```

All tests passed, and using the same Python script as in AES, the speed was `145.74 MB/s`. The speed is ok, however if you open and edit a very large file (> 145 MB), it will take more than 1 second to open and close, after decryption and before encryption. This is not much of an issue however, since not many people will be editing files that are that large it shouldn't be an issue for most people, since the checksum is not calculated when merely adding and removing files from the Vault.

Here is a summary of the unit tests:

BLAKE:

Test Number	Name	Description	Result
1	TestBlakeLarge	Tests BLAKE on a large file (244.3 MB disk image file). Expected result has been calculated using the b2sum utility.	Pass
2	TestBlakeSmall	Tests BLAKE on a very small file (txt file containing the string “hello abcdefghijklmnop” = 23 bytes).	Pass
3	TestBlakeMix	Tests BLAKE’s mixing function (used in compression). Uses values obtained from a working version of the program (checked against official test vectors).	Pass
4	BenchmarkBLAKELarge	Runs test number 1 multiple times (without checking checksum), and calculates the time taken on average for each run.	Time per operation: 1676409286 ns Speed: 145.74 MB/s
5	BenchmarkBLAKESmall	The same as test number 4, however it runs test number 2.	Time per operation: 5282 ns Speed: 4.35 MB/s

AES:

Test Number	Name	Description	Result
1	TestSubBytes	Tests the SubBytes step of AES. Input is some numbers I randomly generated in Python, and the expected result I had to do by hand.	Pass
2	TestInvSubBytes	Tests Inverse SubBytes, using the values in test 1, but inverted (expected values = test values, test values = expected values).	Pass

3	TestShiftRows	Tests the ShiftRows step of AES. Uses the same input values as test 1, and I did the expected result by hand.	Pass
4	TestInvShiftRows	Tests the Inverse ShiftRows step of AES. Uses the same input values as test 3, but inverted.	Pass
5	TestMixColumns	Tests the MixColumns step of AES. Uses test vectors defined here: https://en.wikipedia.org/wiki/Rijndael_MixColumns	Pass
6	TestInvMixColumns	Tests the MixColumns step of AES. Uses the same test vectors in test 5.	Pass
7	TestKeyExpansionCore	Tests the core part of the Key Expansion step at the start of AES. Test vectors calculated by hand again.	Pass
8	TestKeyExpansion	Tests the core Key Expansion step at the start of AES. Test vectors calculated using working version (result of working version was tested against official test vectors).	Pass
9	TestEncDecMediumFile	Encrypts a medium sized file (48.5 MB), then decrypts it again. A checksum of the original file is calculated using the “b2sum” command, and is compared with the checksum of the decrypted file. Both should be the same otherwise there is something wrong with enc/decryption.	Pass
10	TestEncDecSmallFile	Same as test 9 but with a smaller file (A 23 byte txt file containing “hello abcdefghijklmnop”).	Pass
11	BenchmarkEncryptFileLarge	Encrypts a large file (244.3 MB disk image file) multiple times, and gets the average time for	Time taken: 1871293107 ns Speed:

		each run.	130.56 MB/s
12	BenchmarkDecry ptFileLarge	Same as test 11, but decrypts instead.	Time taken: 2042360853 ns Speed: 119.63 MB/s
13	BenchmarkEncry ptFileMedium	Encrypts a medium file (48.5 MB image) multiple times, and gets the average time for each run.	Time taken: 378539062 ns Speed: 128.09 MB/s
14	BenchmarkDecry ptFileMedium	Same as test 13, but decrypts.	Time taken: 417413577 ns Speed: 116.16 MB/s

Black-box tests

The key for the different types of data (where applicable) will be:

- T = Typical Data
- E = Erroneous Data
- B = Boundary Data

Test Number	1
Relevant Objective	1.g
Description	Input various data into the key input of the regular login screen. The valid key I will use will be 1234.
Purpose	To ensure that the login screen is both secure and works as intended.
Test Data	<p>T: [1]: 1234 [2]: 222333 [3]: 2001</p> <p>E: [4]: My name is jeff [5]: 12345678901234567 [6]: 1a2b3c4ddeed [7]: abcdefghijklmnopq</p> <p>B: [8]: 9999999999999999 [9]: 0000000000000000 [10]: 0000000000000000 [11]: (nothing)</p>
Expected Outcome	<p>[1]: Should be accepted as the correct key and main screen should open.</p> <p>[2]: Program should tell the user that the key is invalid, as it is not the key I set (1234).</p> <p>[3]: Program should tell the user that the key is invalid, as it is not the key I set (1234).</p> <p>[4]: Program should tell the user that the key is invalid, as it contains characters.</p> <p>[5]: Program should tell the user that the key is invalid, as it is longer than 16 in length.</p> <p>[6]: Program should tell the user that the key is invalid, as it contains characters.</p> <p>[7]: Program should tell the user that the key is invalid, as it is longer than 16 in length.</p> <p>[8]: Program should tell the user that the key is invalid, as it is not the key I set (1234) ([8] is 16 in length).</p> <p>[9]: Program should tell the user that the key is invalid, as it is not the key I set (1234) ([9] is 16 in length).</p> <p>[10]: Program should tell the user that the key is invalid, as it is longer than 16 in length.</p> <p>[11]: Program should tell the user that the key is invalid, as the key has to at least be 1 in length.</p>
Actual Outcome	[All]: Pass

Evidence:

[1]: Opened



[2, 3]: Invalid

Input Key

Invalid

Invalid key.

Submit

Login with BT

This screenshot shows a dark-themed login form. At the top is a text input field labeled "Input Key" containing five asterisks ("*****"). A red horizontal bar spans the width of the input field, with the word "Invalid" written in white at its center. Below the input field, a dark gray rectangular box contains the text "Invalid key." A "Submit" button is located at the bottom of this box. In the bottom left corner, there is a "Login with BT" button.

Input Key

Invalid

Invalid key.

Submit

Login with BT

This screenshot shows a dark-themed login form. At the top is a text input field labeled "Input Key" containing three asterisks ("***"). A red horizontal bar spans the width of the input field, with the word "Invalid" written in white at its center. Below the input field, a dark gray rectangular box contains the text "Invalid key." A "Submit" button is located at the bottom of this box. In the bottom left corner, there is a "Login with BT" button.

[4, 6]: No characters:

Input Key

Invalid

Invalid key, valid key contains no letters.

Submit

Login with BT

This screenshot shows a dark-themed login form. At the top is a text input field labeled "Input Key" containing five asterisks ("*****"). A red horizontal bar spans the width of the input field, with the word "Invalid" written in white at its center. Below the input field, a dark gray rectangular box contains the text "Invalid key, valid key contains no letters." A "Submit" button is located at the bottom of this box. In the bottom left corner, there is a "Login with BT" button.

Input Key

Invalid

Invalid key, valid key contains no letters.

Submit

Login with BT

This screenshot shows a dark-themed login form. At the top is a text input field labeled "Input Key" containing five asterisks ("*****"). A red horizontal bar spans the width of the input field, with the word "Invalid" written in white at its center. Below the input field, a dark gray rectangular box contains the text "Invalid key, valid key contains no letters." A "Submit" button is located at the bottom of this box. In the bottom left corner, there is a "Login with BT" button.

[5, 7, 10]: Input too long:

Input Key

Invalid

Invalid key, longer than 16 characters.

Submit

Login with BT

This screenshot shows a dark-themed login form. At the top is a text input field labeled "Input Key" containing five asterisks ("*****"). A red horizontal bar spans the width of the input field, with the word "Invalid" written in white at its center. Below the input field, a dark gray rectangular box contains the text "Invalid key, longer than 16 characters." A "Submit" button is located at the bottom of this box. In the bottom left corner, there is a "Login with BT" button.

Input Key

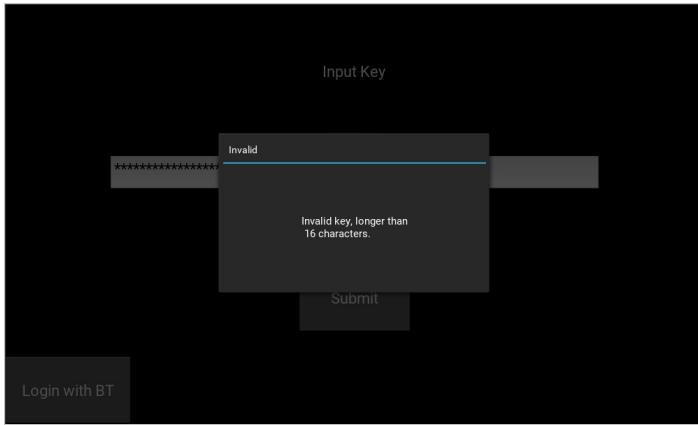
Invalid

Invalid key, longer than 16 characters.

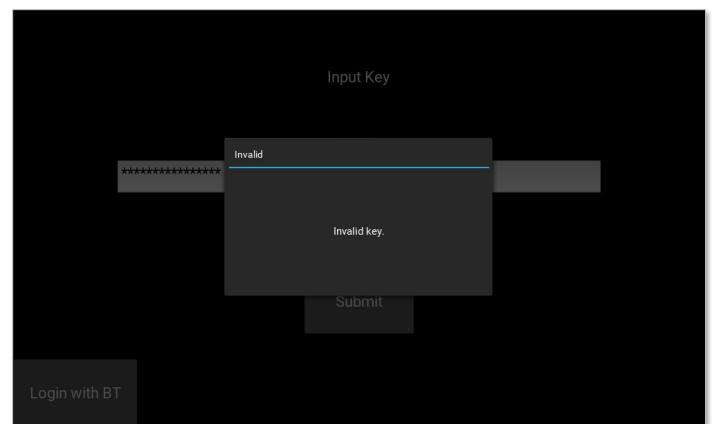
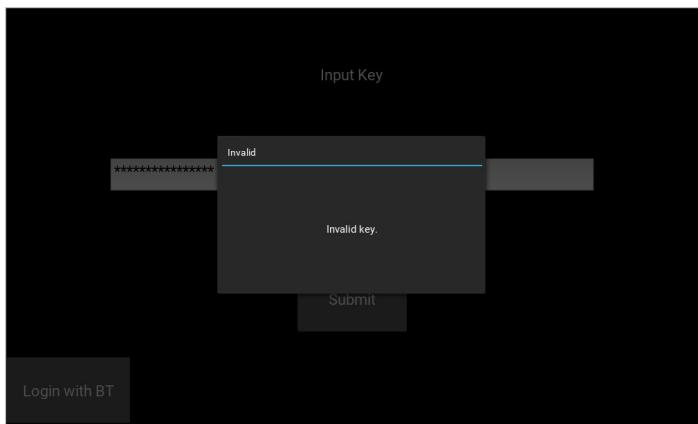
Submit

Login with BT

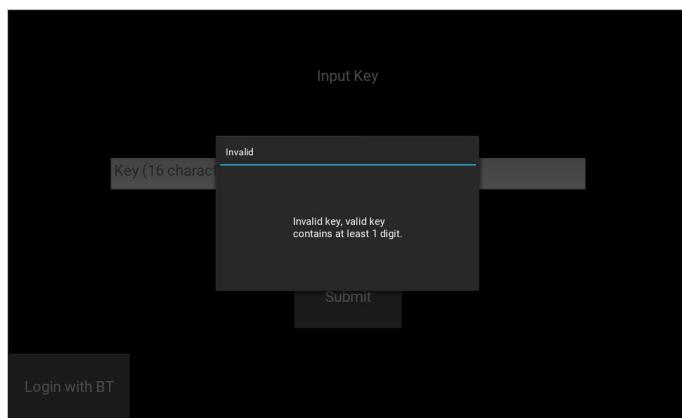
This screenshot shows a dark-themed login form. At the top is a text input field labeled "Input Key" containing five asterisks ("*****"). A red horizontal bar spans the width of the input field, with the word "Invalid" written in white at its center. Below the input field, a dark gray rectangular box contains the text "Invalid key, longer than 16 characters." A "Submit" button is located at the bottom of this box. In the bottom left corner, there is a "Login with BT" button.



[8, 9]: Invalid key



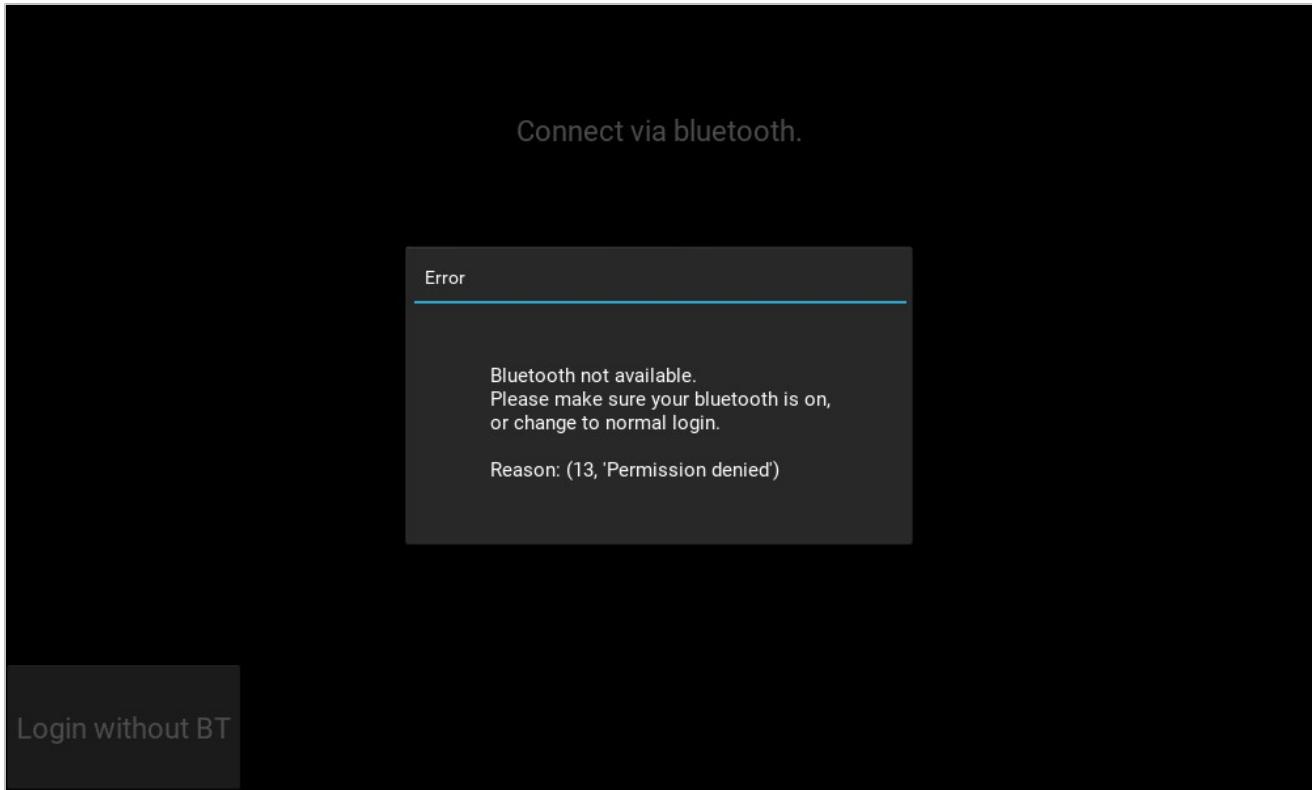
[11]: Invalid key, has to be at least 1 in length.



Test Number	2
Relevant Objective	1.L
Description	Switch login screen to Bluetooth login screen from regular login screen with Bluetooth disabled.
Purpose	To check that it will warn the user that Bluetooth is not available.

Test Data	Only one available input: (T) Click "Login with BT" button.
Expected Outcome	A popup should appear telling the user that Bluetooth is not available, and why (for the user to debug their Bluetooth problem).
Actual Outcome	Pass

Evidence:



Test Number	3
Relevant Objective	2.b
Description	Connect to PC from mobile device.
Purpose	To check that the mobile device can actually connect, and if not displays the appropriate message.
Test Data	<p>T: [1]: Click "josh-pc" to connect to my PC. E: [2]: Click "josh-pc" when the program is not running on the PC. [3]: Click "josh-laptop" when the program is running on the PC. B: [4]: Click "josh-pc" when the program is running on both my laptop and the PC. [5]: Click "josh-laptop" in the same situation as [4].</p>
Expected Outcome	<p>[1]: Connects successfully and changes screen to the pad screen. [2]: Should not connect and asks user to retry. [3]: Should not connect and asks user to retry. [4]: Should connect to "josh-pc" successfully and change screen on the PC. [5]: Should connect to "josh-laptop" successfully and change screen on the PC.</p>

Actual Outcome	[All]: Pass
----------------	-------------

Evidence:

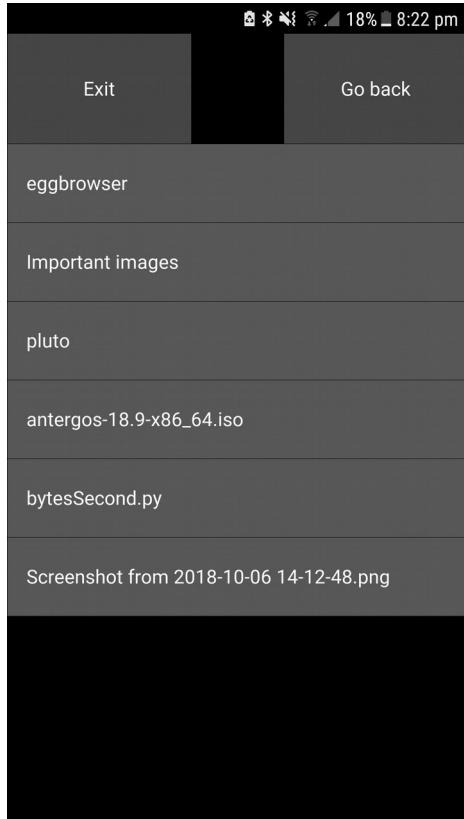
Test Number	4
Relevant Objective	1.g.ii, 2.b, 2.d
Description	Enter the key from the mobile app.
Purpose	Test that the key is either accepted, or declined by the server.
Test Data	(The app doesn't let you enter more than 16 numbers.) T: [1]: 1234 (The set key) [2]: 12340 [3]: 2001 B: [4]: 9876543210123456 [5]: 7389123612645815 [6]: 0000000000000000 [7]: (nothing)
Expected Outcome	[1]: PC program should log in and change screen to main screen. [2-7]: App should say that the key is invalid.
Actual Outcome	[All]: Pass

Evidence:

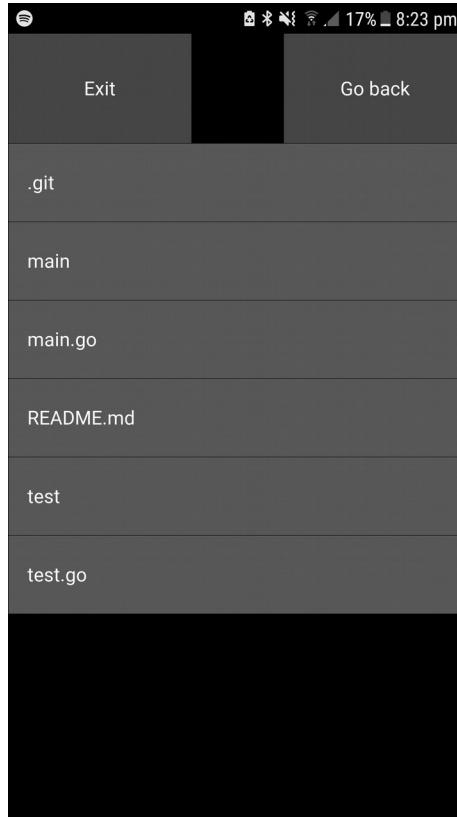
Test Number	5
Relevant Objective	2.f
Description	Browse files + download files within the mobile app.
Purpose	To test that the user can browse the files in the Vault and also download.
Test Data	T: [1]: Click to open the file browser. [2]: Click a folder. [3]: Click a file. [4]: Click to go back up one directory. [5]: Exit the file browser.
Expected Outcome	[1]: File browser should open. [2]: The file browser should navigate to that folder. [3]: The file should download to the users downloads folder. A popup should open on the PC showing the current status of the transfer, and once complete, the file should be found in the users "Download" folder. [4]: The file browser should display the contents of the folder above the current folder. [5]: Should return to main screen.
Actual Outcome	[All]: Pass

Evidence:

[1] Opening the file browser:

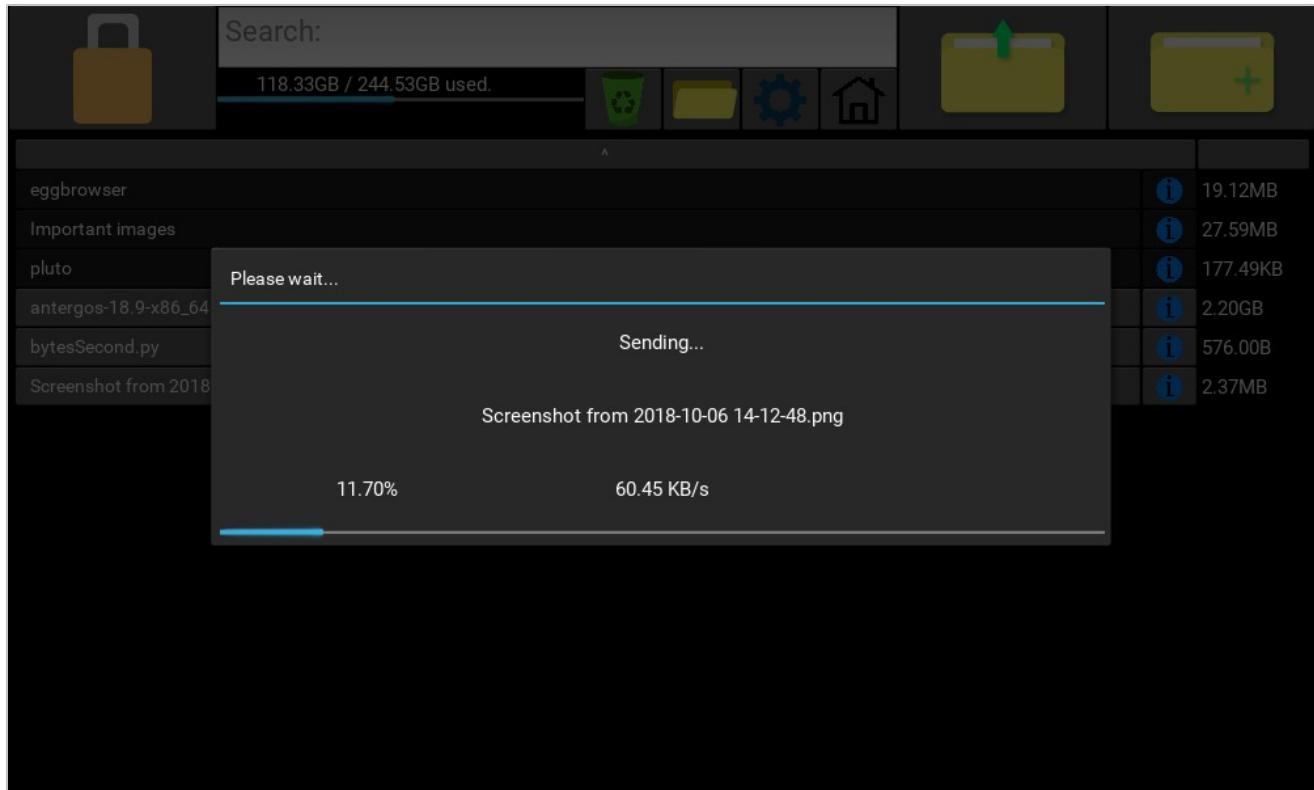


[2] Opening a folder:

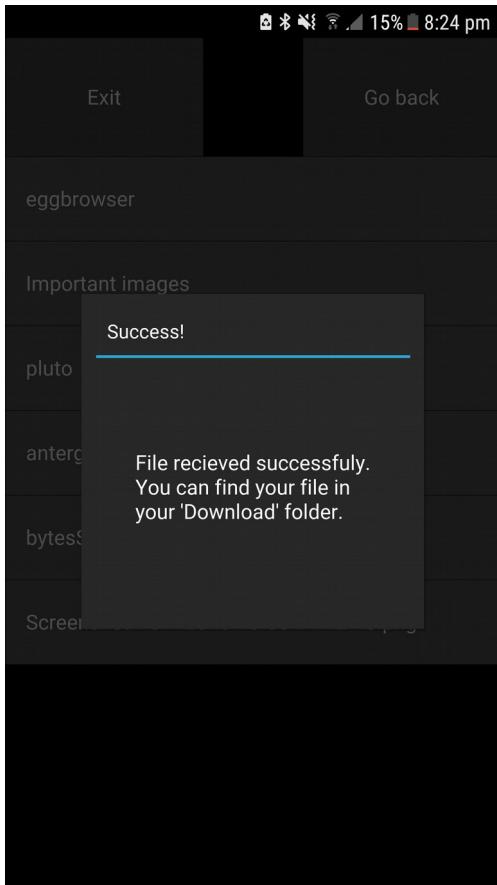


[3] Clicking a file ("Screenshot from 2018-10-06 14-12-48"):

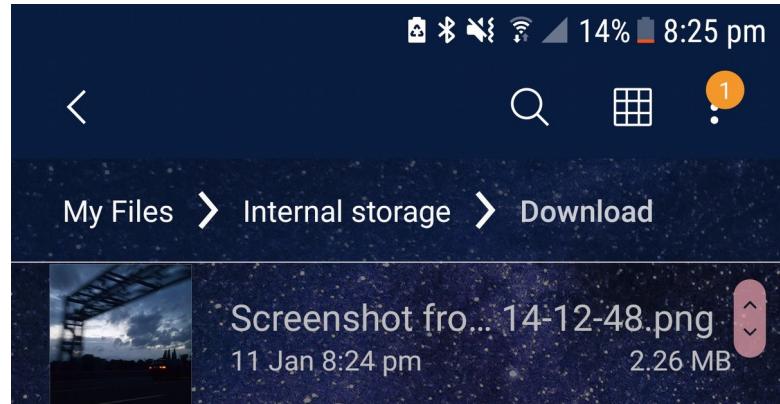
View on PC program:



When done on phone:



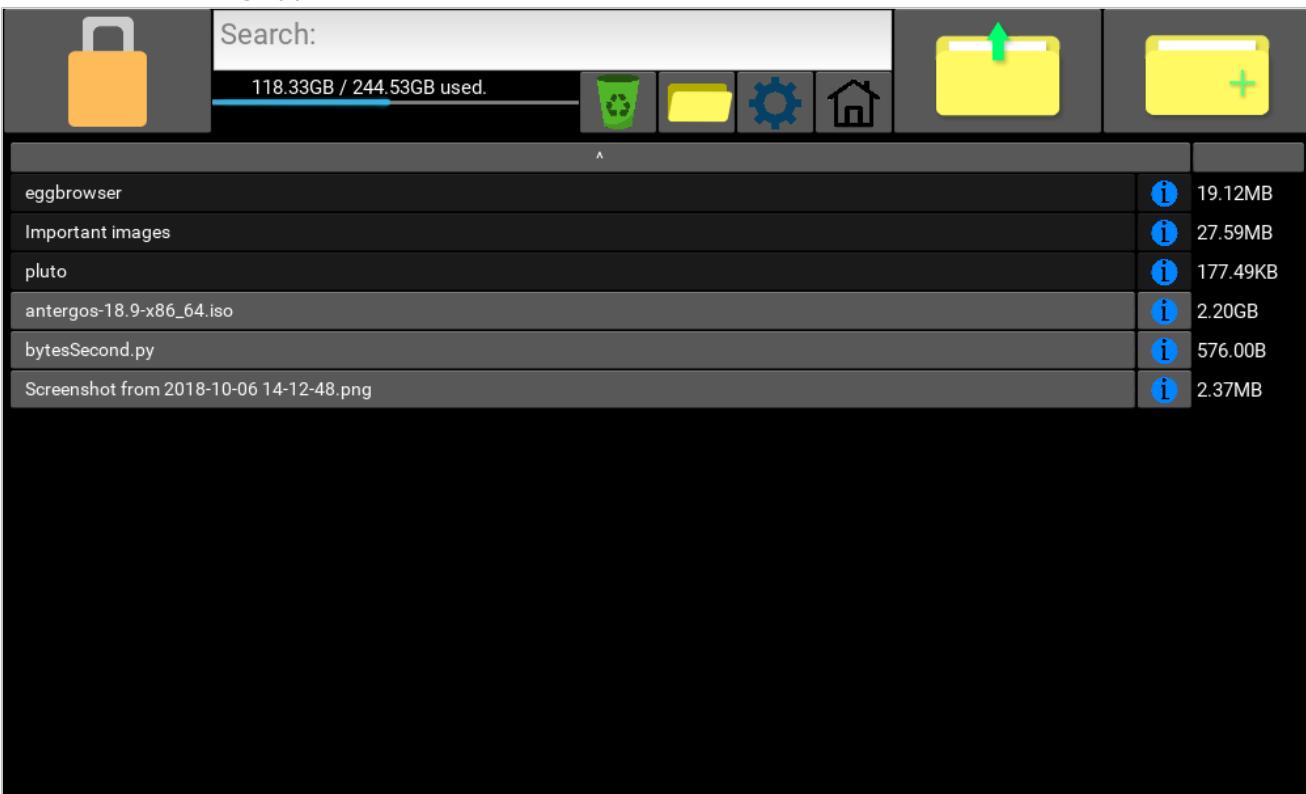
Download folder:



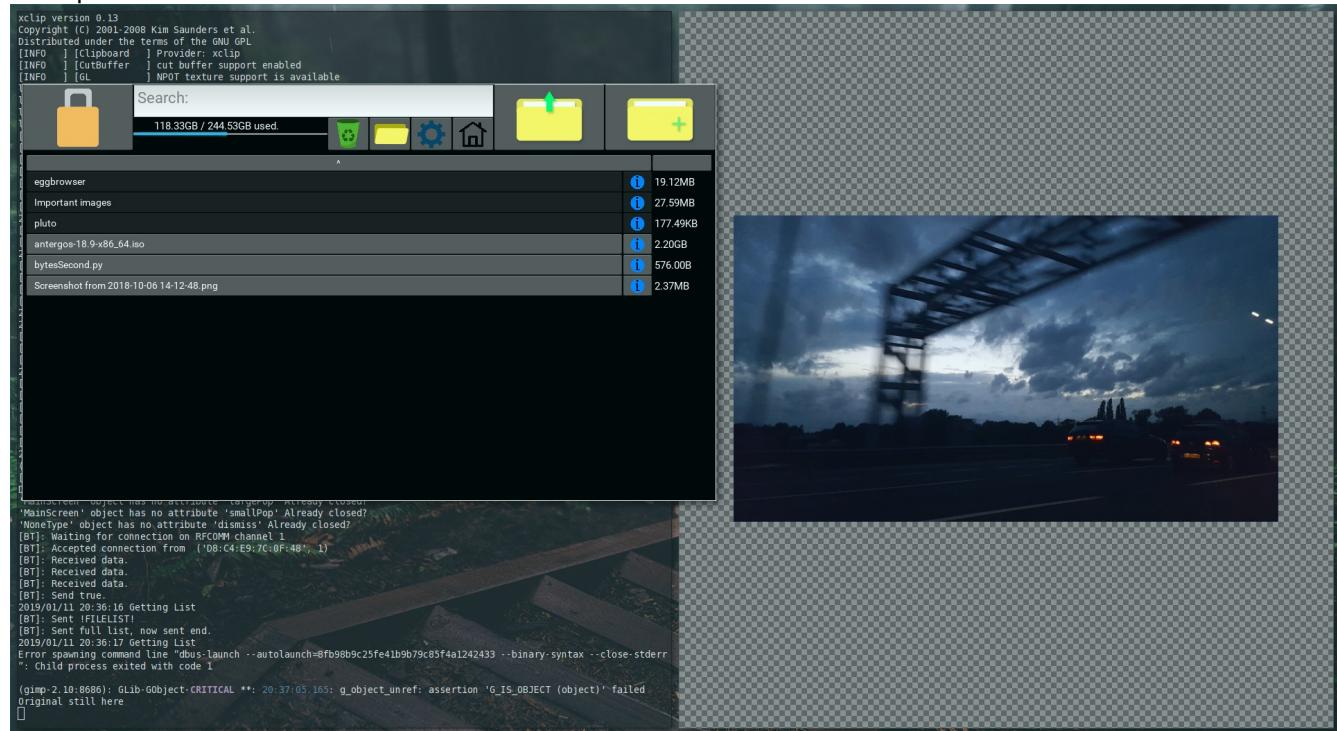
Test Number	6
Relevant Objective	3.c
Description	Disconnect from the Bluetooth server. (Close the mobile app)
Purpose	To check that temporary files are deleted and that the program is locked once connection is lost to the mobile device.
Test Data	T: [1]: Log into the program using the mobile device, open a file, close the file and disconnect from the program.
Expected Outcome	[1]: The temporary files should be deleted and the program should be locked.
Actual Outcome	[All]: Pass

Evidence:

Vault unlocked using app:



File opened:



Disconnected from vault, and list /tmp directory to make sure temporary files have been deleted:

Connect via bluetooth.

```

user:      josh
os:       archlabs
kern:     4.20.0
wm:      i3-gaps
sh:       fish
pkgs:    1491
term:    termite

▲ ▼ ▲ ▼ ▲ ▼ ▲
~ $ ls /tmp | grep "FileMate"
~ $ 

```

Login without BT

FILEMATE (filemate) - ALREADY CLOSED

Test Number	7
Relevant Objective	3.e
Description	Delete a file, open the recycling folder and recover it.
Purpose	To check that when a file is deleted and recovered to the correct location.
Test Data	Delete a file in the Vault, go into the recycling folder, click the file, exit the recycling folder and see if it has been recovered.
Expected Outcome	The file should be in the recycling folder when the file has been deleted, and should return to the Vault where it was deleted from once you recover it.
Actual Outcome	The file is recovered to the root folder of the Vault, instead of the place it was deleted from.

Evidence:

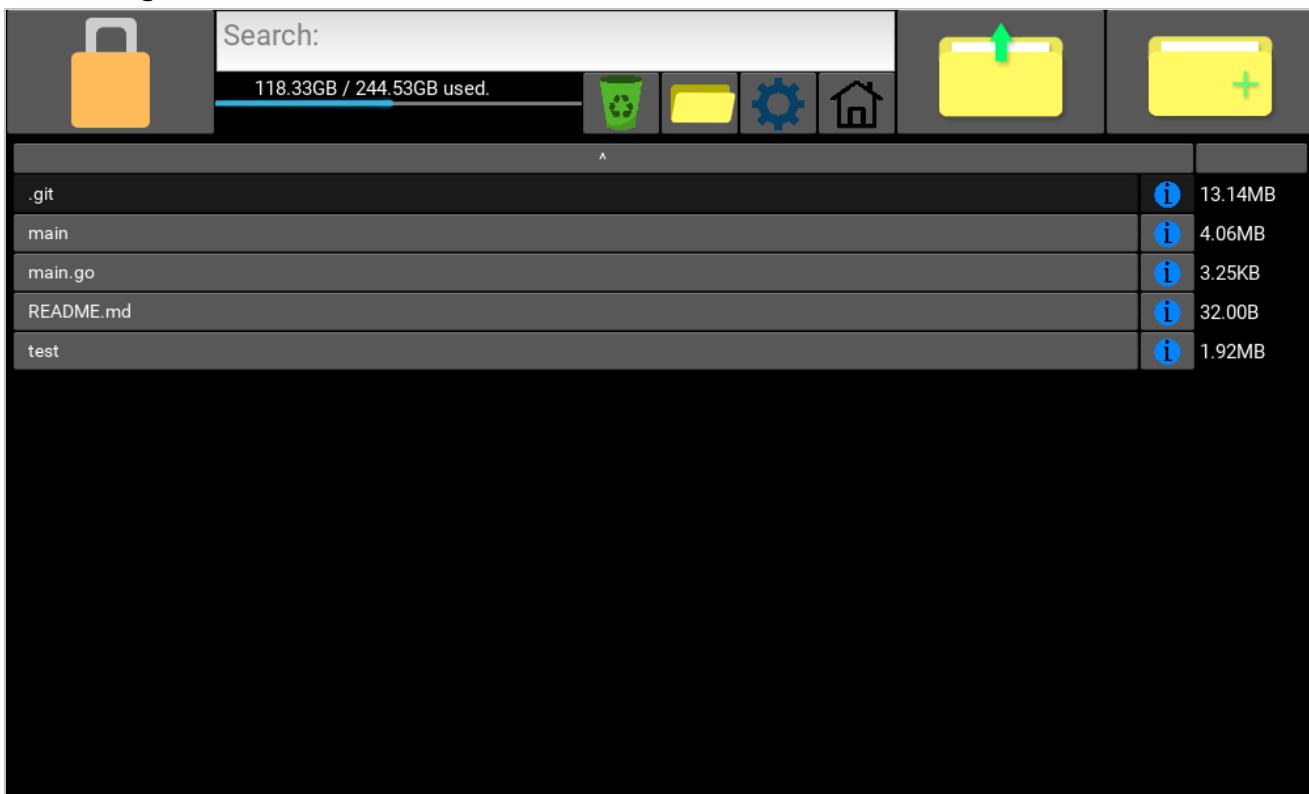
Files in Vault/eggbrowser/:



The screenshot shows the 'Vault' application interface with the 'eggbrowser' directory selected. The top bar includes a search field, a progress bar indicating '118.33GB / 244.53GB used.', and several icons for file operations like trash, folder, settings, and home.

File/Folder	Size
.git	13.14MB
main	4.06MB
main.go	3.25KB
README.md	32.00B
test	1.92MB
test.go	176.00B

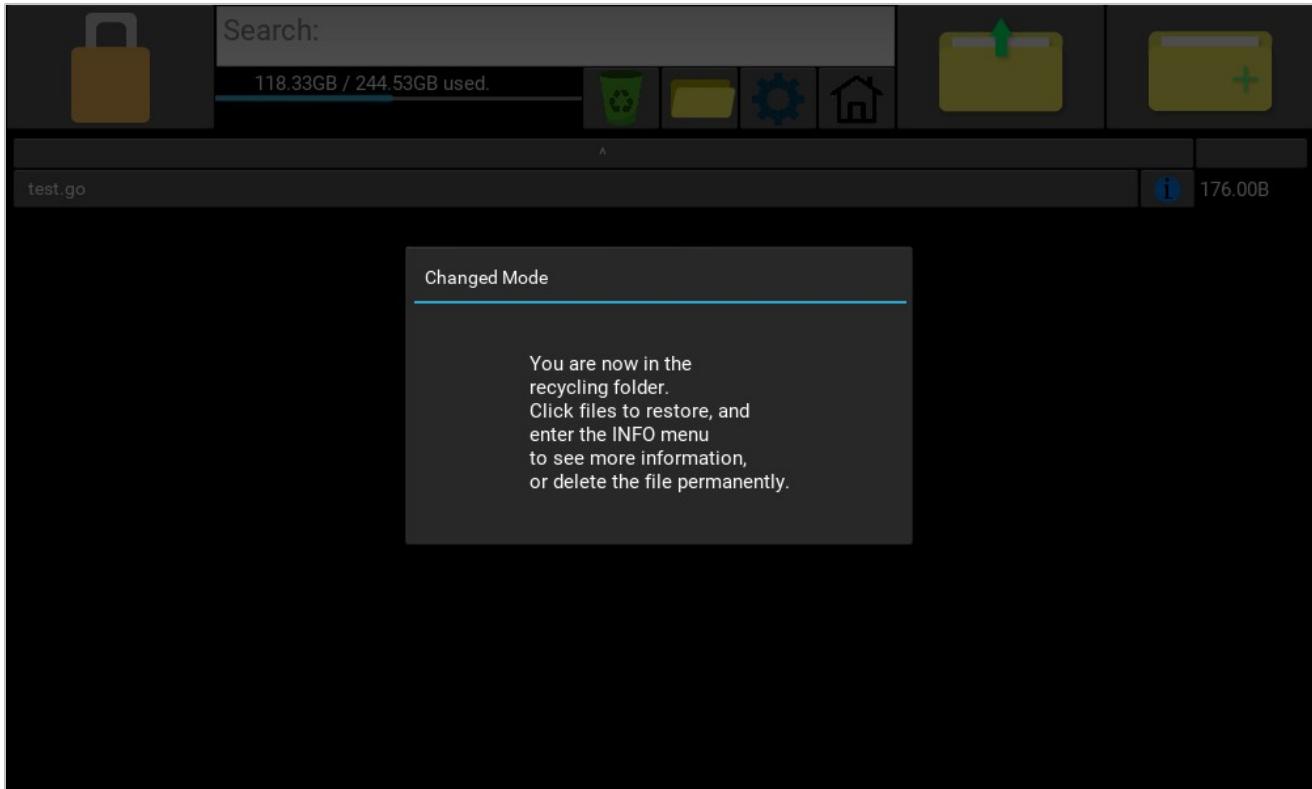
Delete test.go:



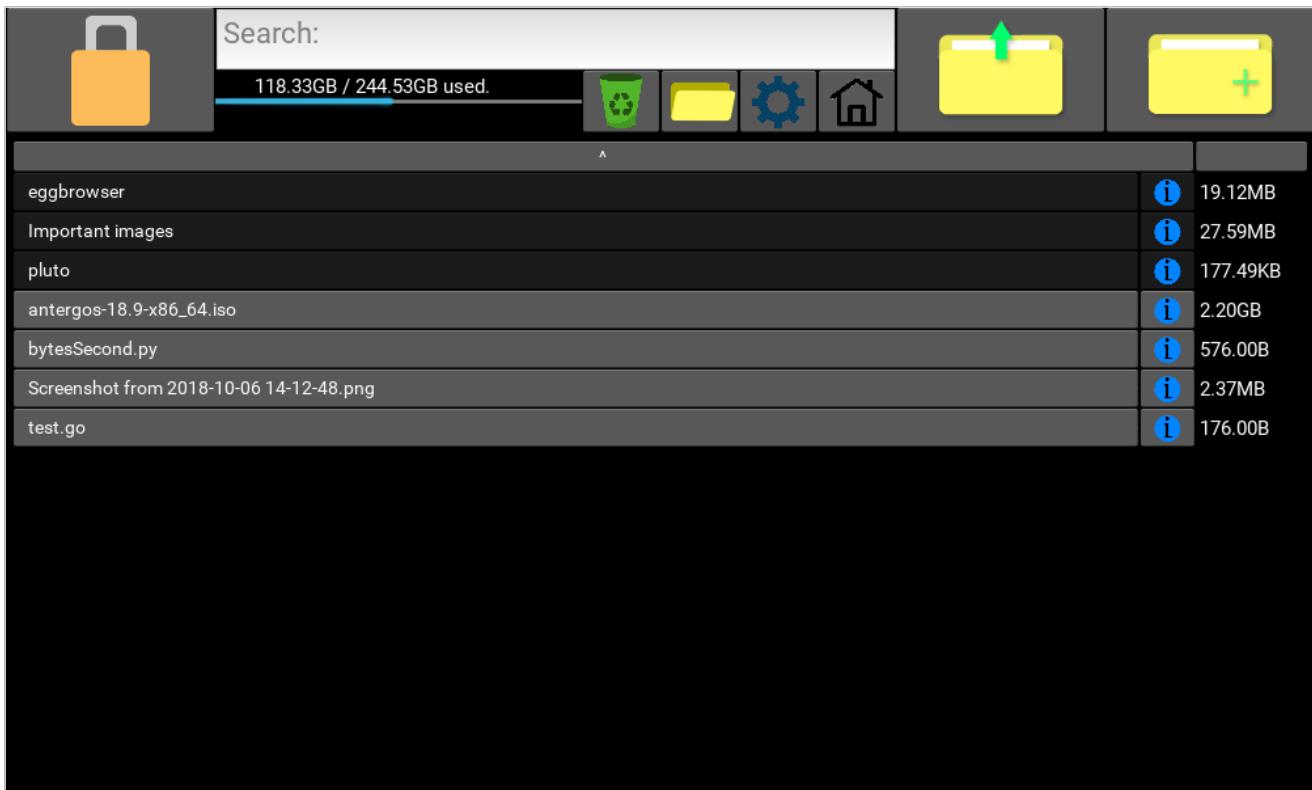
The screenshot shows the 'Vault' application interface with the 'eggbrowser' directory selected. The top bar includes a search field, a progress bar indicating '118.33GB / 244.53GB used.', and several icons for file operations like trash, folder, settings, and home.

File/Folder	Size
.git	13.14MB
main	4.06MB
main.go	3.25KB
README.md	32.00B
test	1.92MB

Go to recycling folder:



Recover test.go, and return to the root directory of the vault:



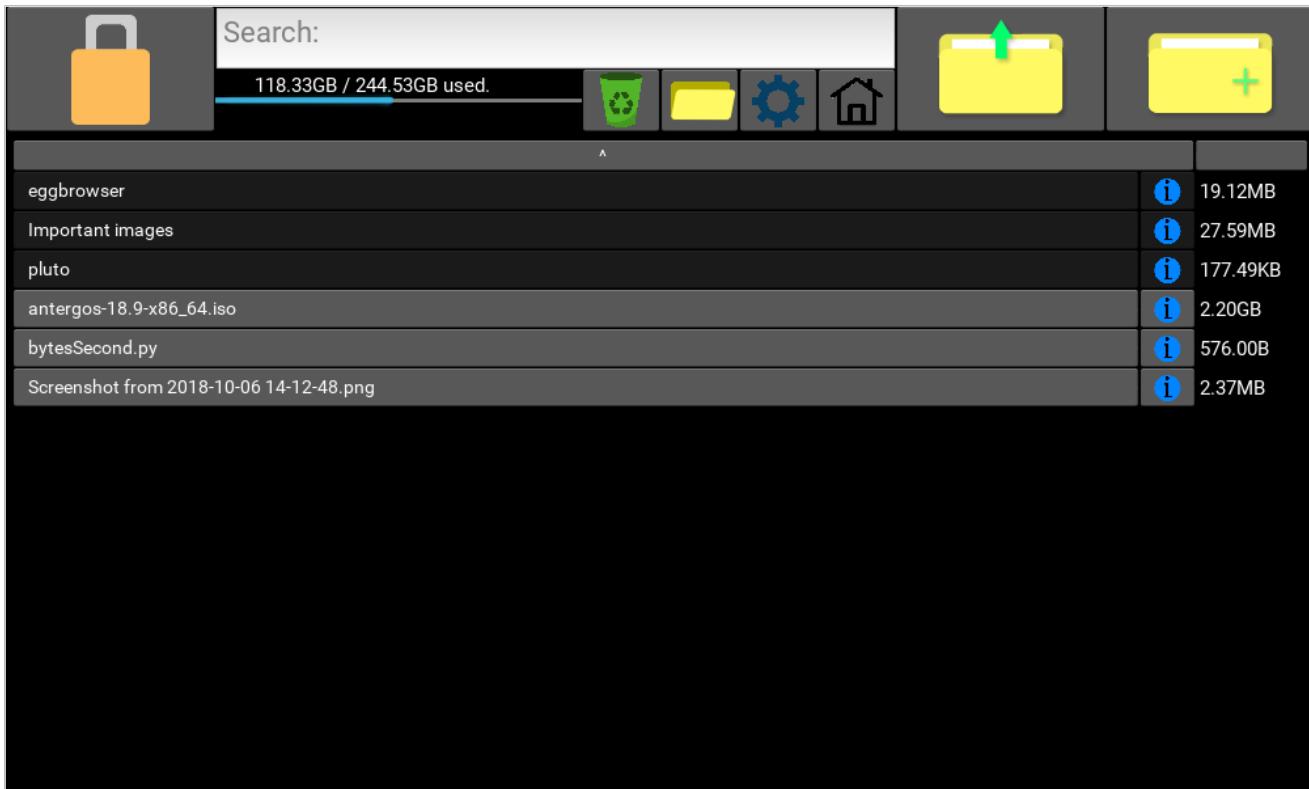
Test Number	8
Relevant Objective	1.m
Description	Click the home button to return to the root directory of the Vault..
Purpose	Check that the home button functions as required.
Test Data	T: [1]: Click the home button while in the root directory. [2]: Click the home button while in another folder.
Expected Outcome	[1]: Buttons in the file browser should remain the same. [2]: The contents of the home folder should be loaded into the file browser.
Actual Outcome	[All]: Pass

Evidence:

In a folder:

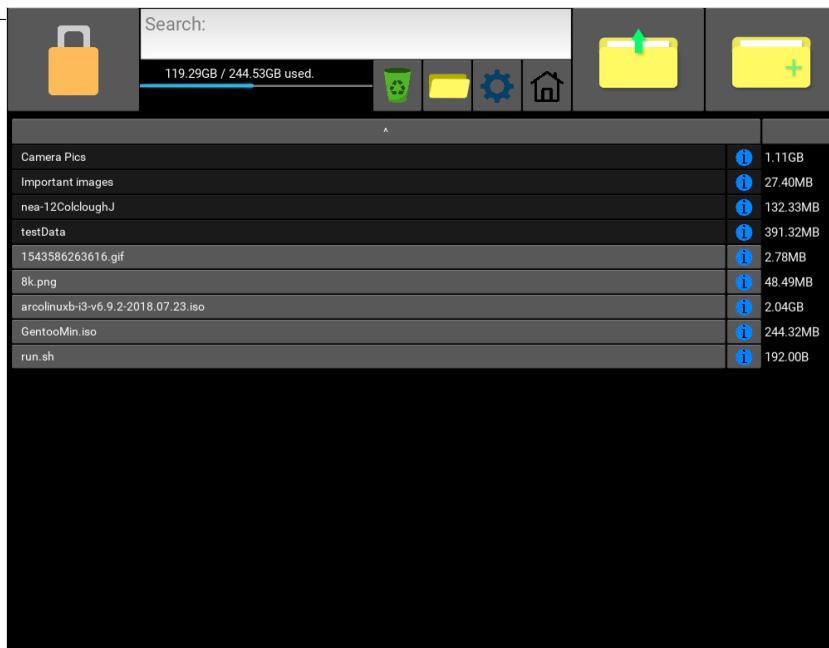


Pressed home button:



Test Number	9
Relevant Objective	1.k
Description	Re-size the window.
Purpose	Check that the GUI remains intact when you resize the window.
Test Data	T: Resize the window.
Expected Outcome	Widgets on the screen should resize to fit the new window size.
Actual Outcome	Pass

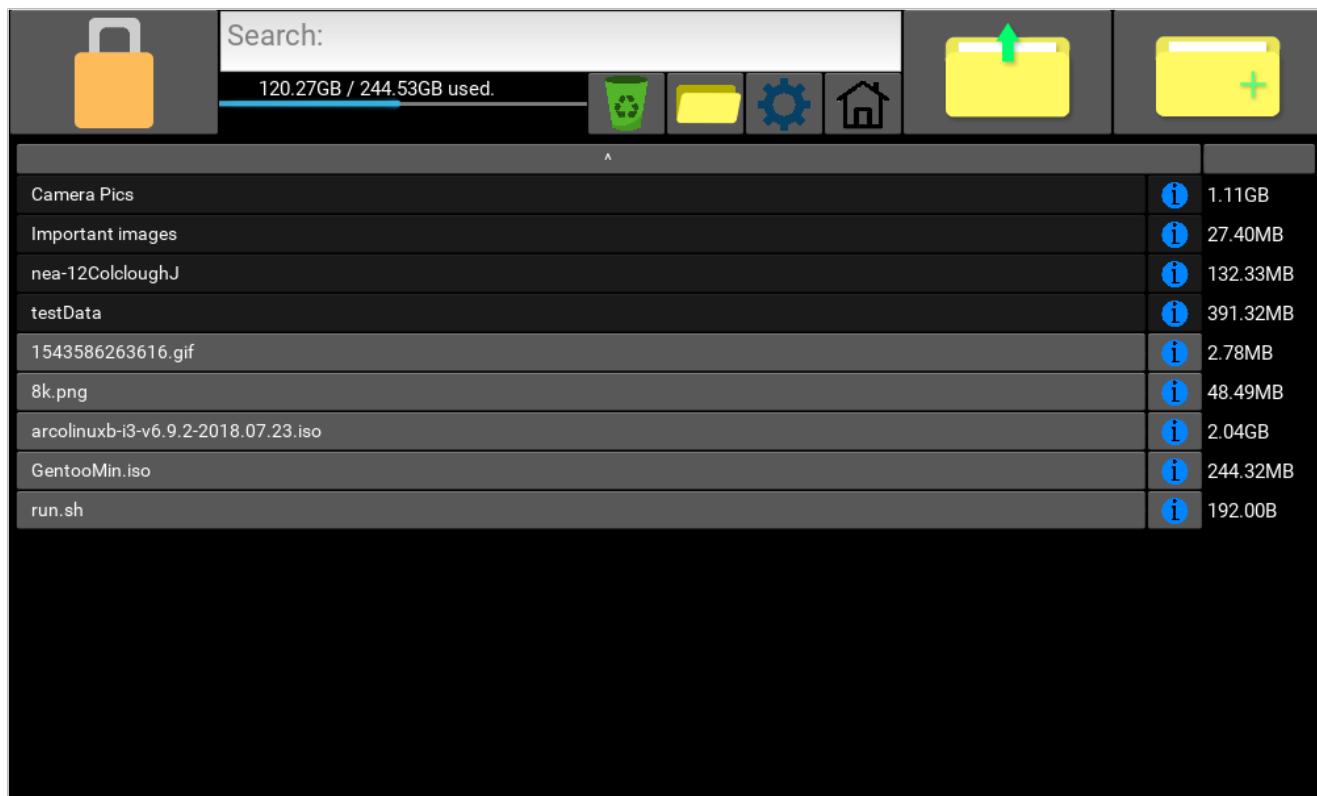
Evidence:



Test Number	10
Relevant Objective	1.0
Description	Change the direction of the sorts.
Purpose	To check that the sorts work.
Test Data	[1]: Sort by name ascending (folders separated by files). [2]: Sort by name descending (folders separated by files). [3]: Sort by size ascending. [4]: Sort by size descending.
Expected Outcome	[All]: Should sort correctly.
Actual Outcome	[All]: Pass

Evidence:

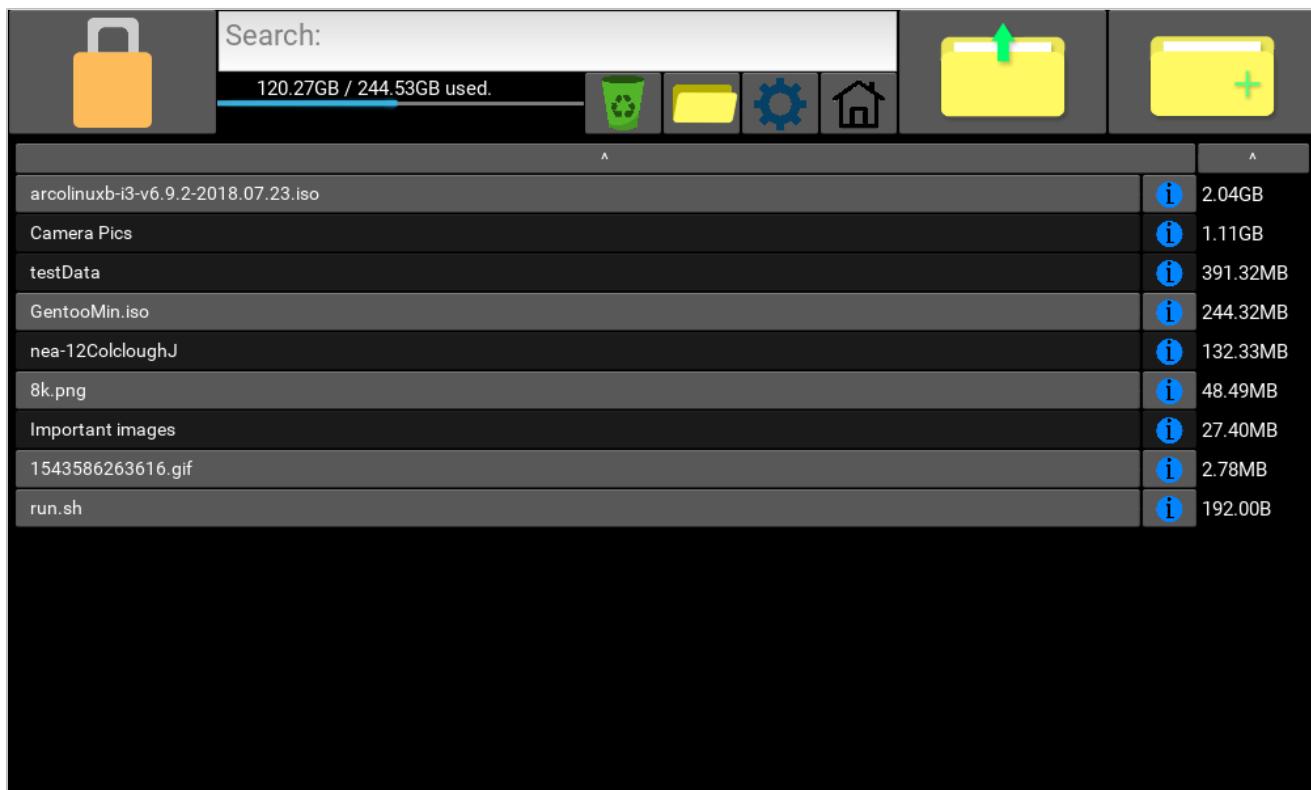
[1]:



[2]:



[3]:



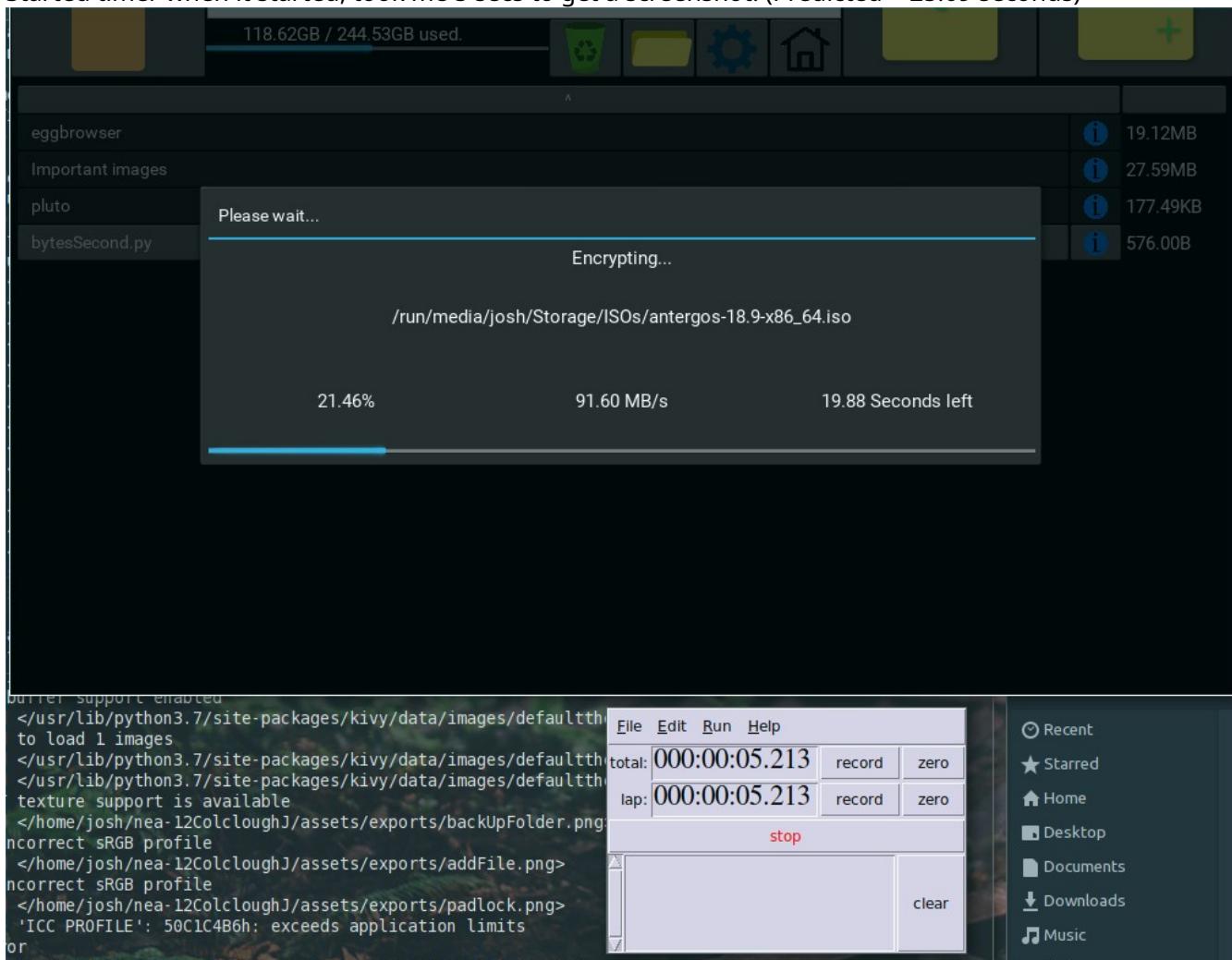
[4]:



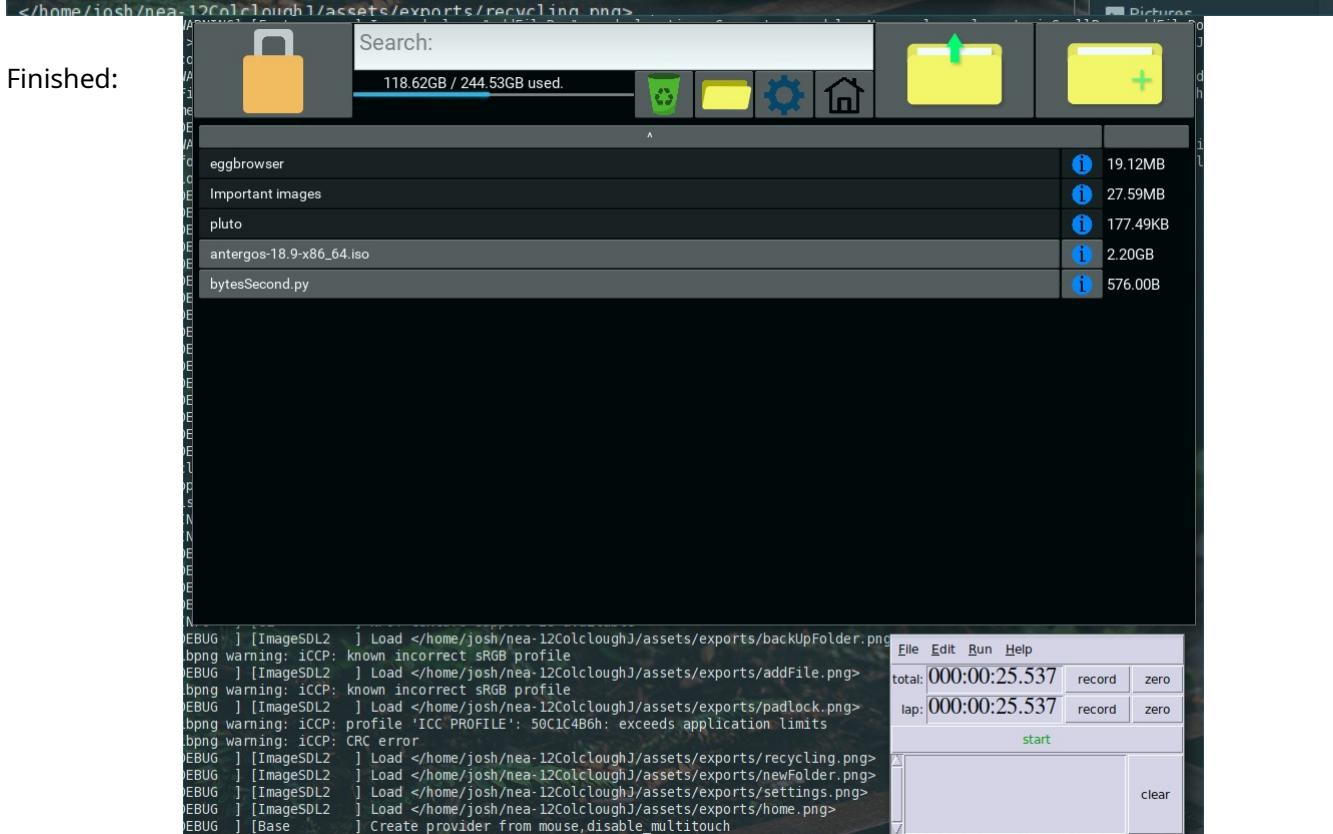
Test Number	11
Relevant Objective	1.n.iii
Description	Time how long it takes to decrypt a large file (2.04GB) compared to the prediction made by the popup that opens (encDecPop) displaying the predictions.
Purpose	Make sure the time prediction is accurate (otherwise it is pretty infuriating for the user).
Test Data	Decrypt a 2.2 GB file (2,203,779,072) and time it. Get the first available time prediction.
Expected Outcome	The difference between the time actually taken and the prediction should be +-2 seconds.
Actual Outcome	Pass: Predicted 25.09 seconds (19.88+5.21), took 25.537 seconds.

Evidence:

Started timer when it started, took me 5 secs to get a screenshot. (Predicted = 25.09 seconds)



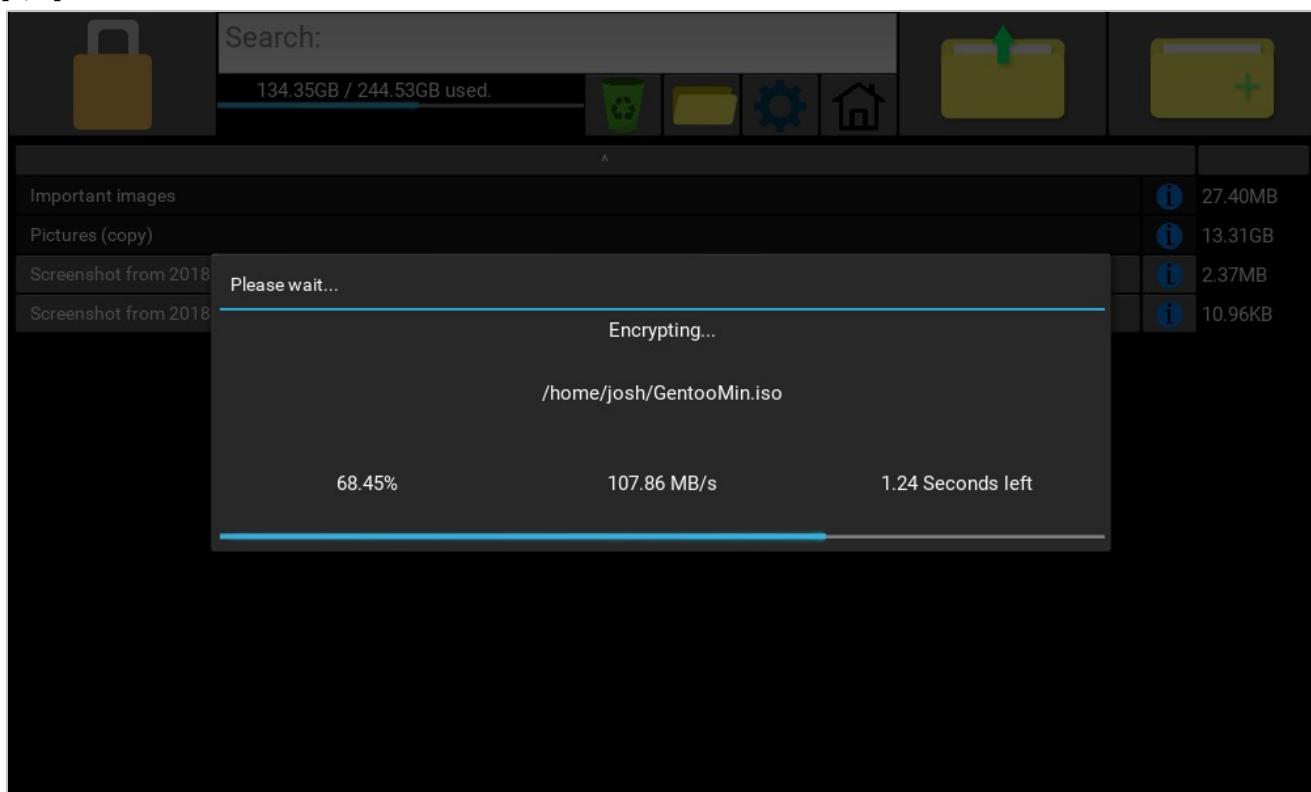
Finished:



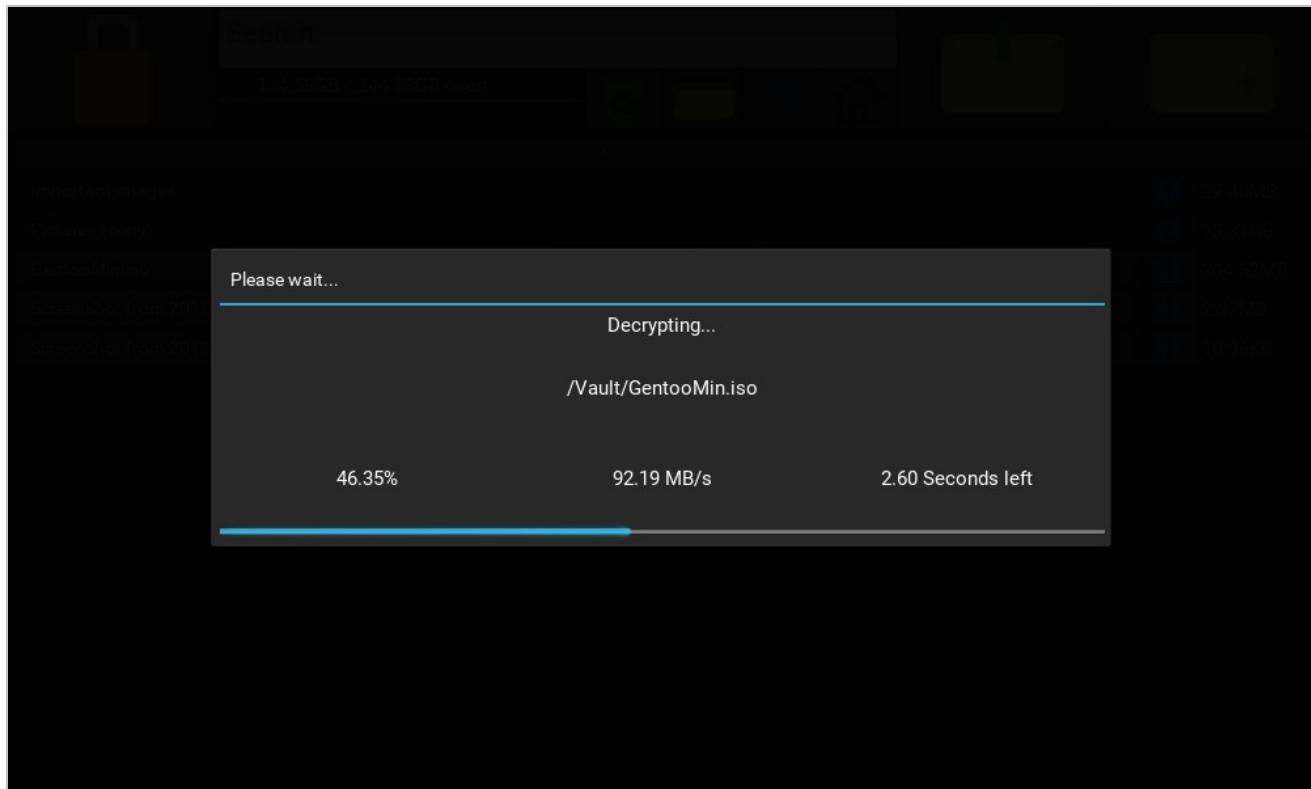
Test Number	12
Relevant Objective	1.n.ii
Description	Compare the speed of encryption/decryption with the Go benchmark function.
Purpose	To check that the speed is found to be accurate.
Test Data	[1]: Encrypt a 244.3 MB file using the PC program. [2]: Encrypt a 244.3 MB file using Go benchmark. [3]: Decrypt a 244.3 MB file using the PC program. [4]: Decrypt a 244.3 MB file using Go benchmark.
Expected Outcome	[All]: Speeds should be similar (within +-5 MB/s).
Actual Outcome	(Done on different computer to in unit tests) [1]: 107.86 MB/s 2269321613 ns/op [2]: 107.66 MB/s [3]: 95.56 MB/s 2556627030 ns/op [4]: 92.19 MB/s

Evidence:

[1, 2]:

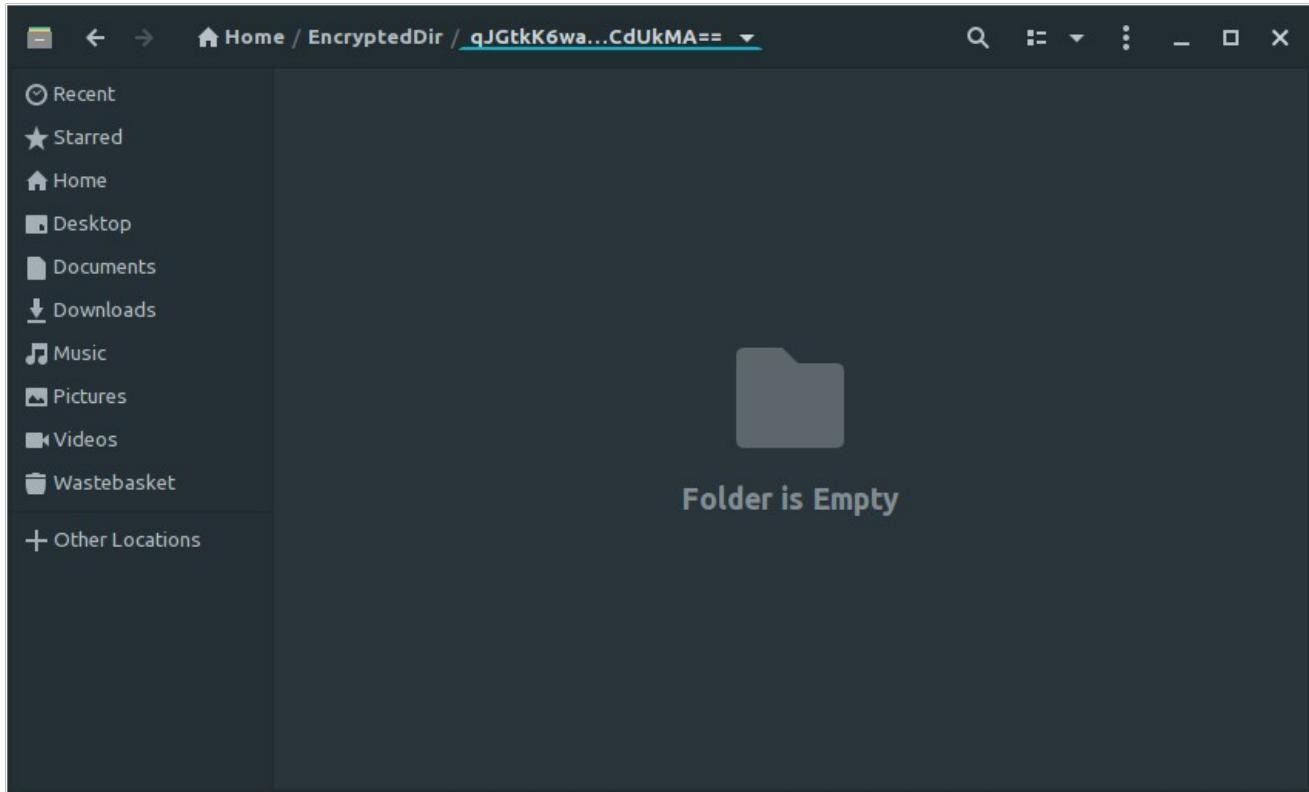


[3, 4] (darker background due to multiple popups):



Test Number	13
Relevant Objective	3.e
Description	Delete a file into the recycling bin, and then delete it permanently and check it has been deleted.
Purpose	Check that it works.
Test Data	Permanently delete the file called "TqAoUIWqlyvmQBgyAIS8PQ==" in encrypted form, "test12_3.png" in decrypted form.
Expected Outcome	The file should not be available anywhere.
Actual Outcome	Pass

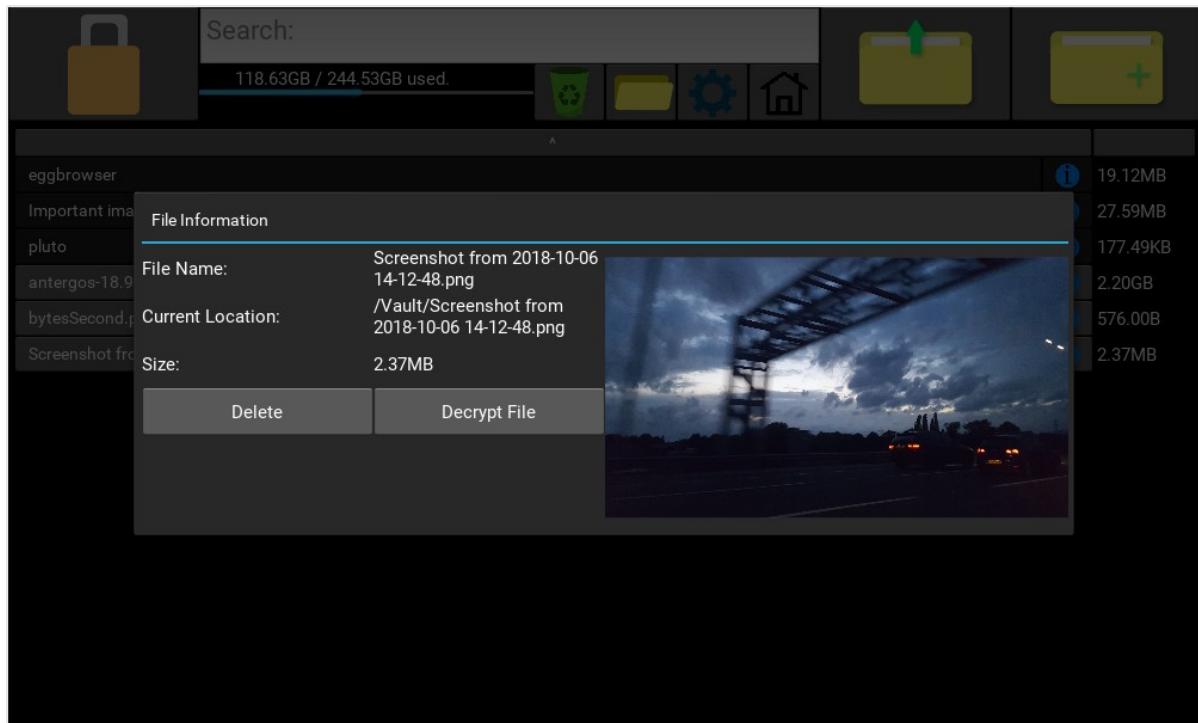
Evidence (checking recycling bin in vault):



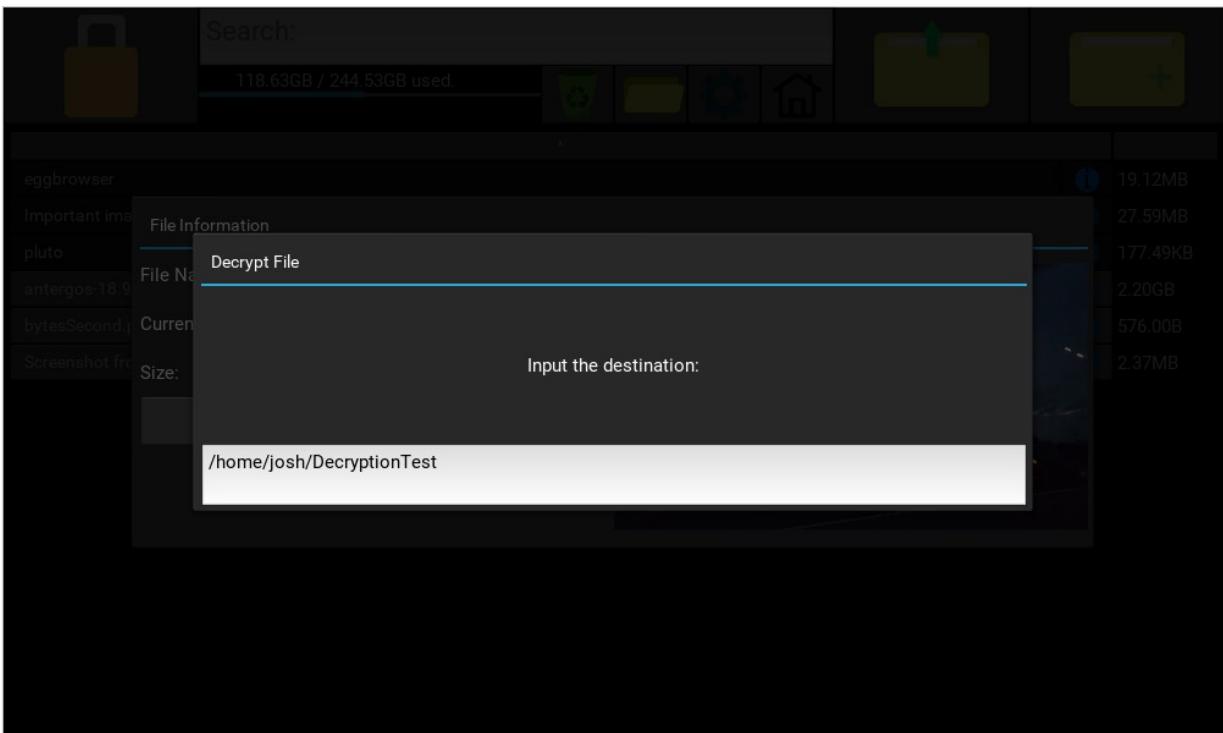
Test Number	14
Relevant Objective	3.i
Description	Decrypt a file to a location outside of the vault.
Purpose	To check it works normally.
Expected Outcome	File should be decrypted successfully.
Actual Outcome	Pass

Evidence:

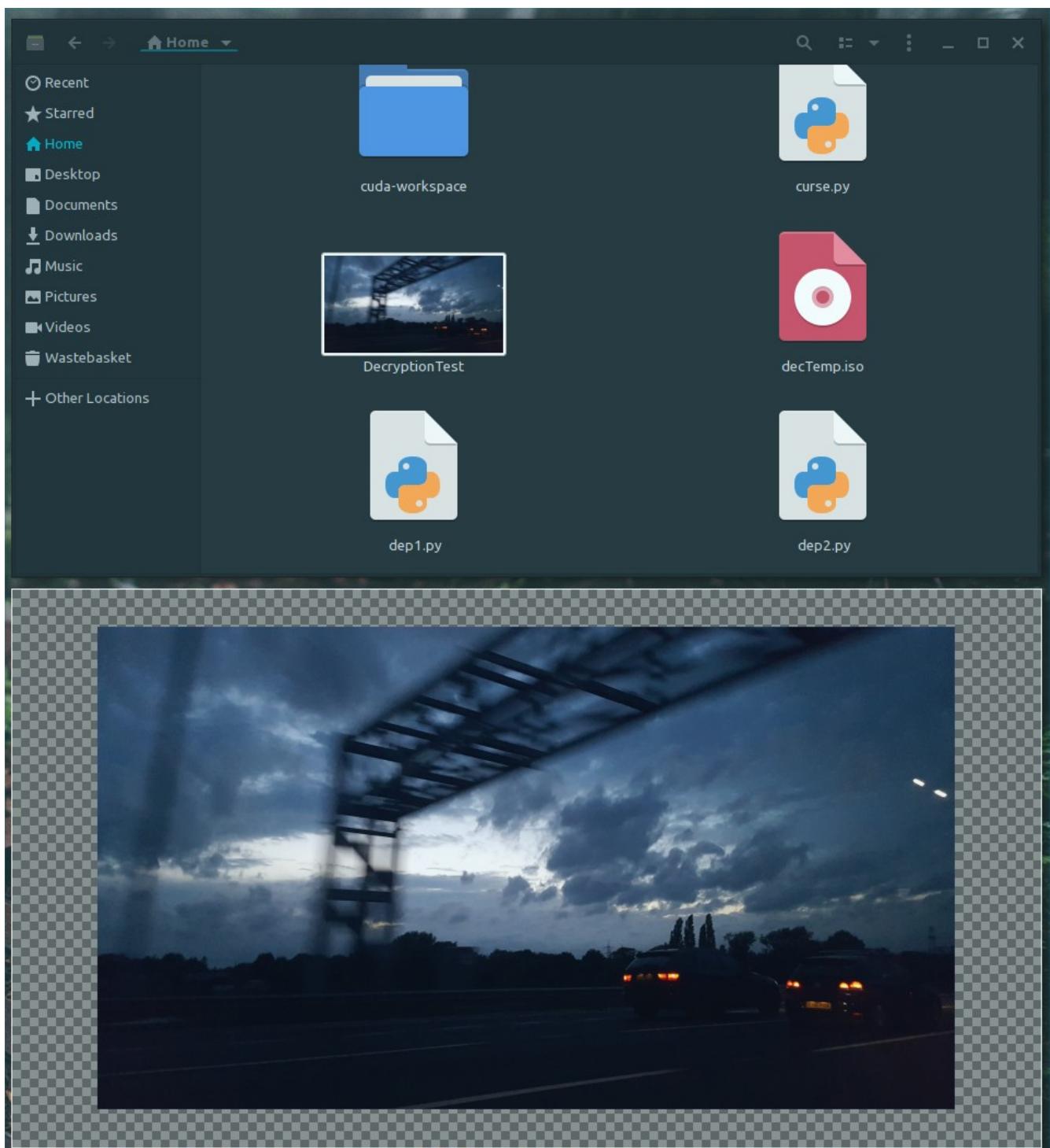
Before:



Input:



After:



Top tile is file manager in /home/josh/, bottom is the open decrypted image.

Test Number	15
Relevant Objective	3.i
Description	Decrypt a folder to a location outside of the vault.

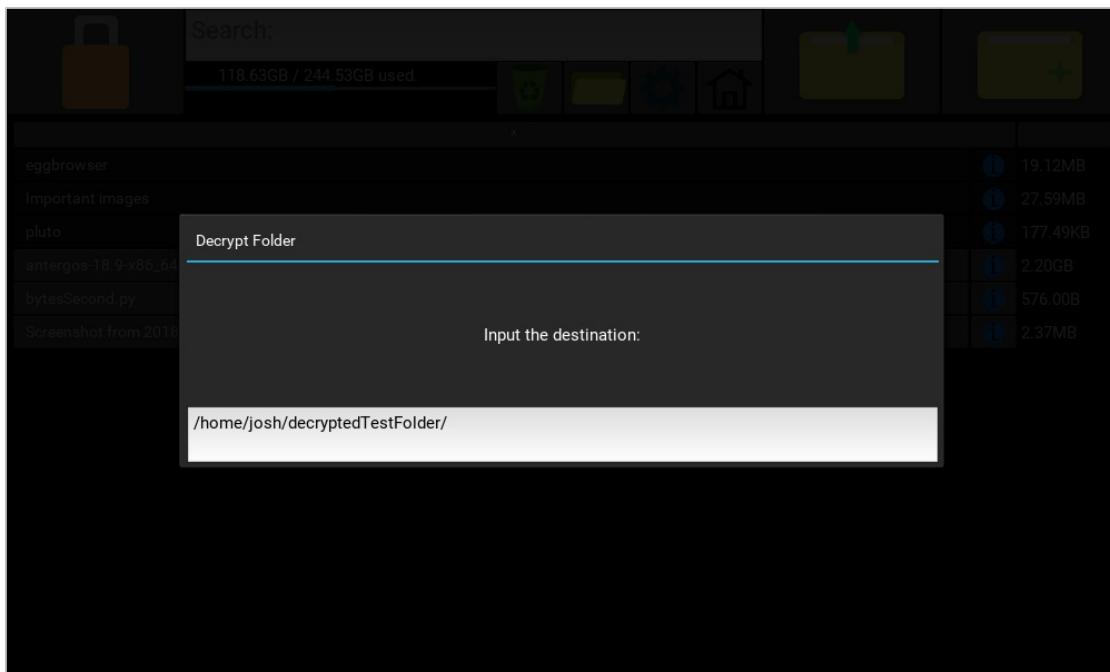
Purpose	To check it works normally.
Expected Outcome	Folder should be decrypted successfully, with all files and folders in the correct place.
Actual Outcome	Pass

Evidence:

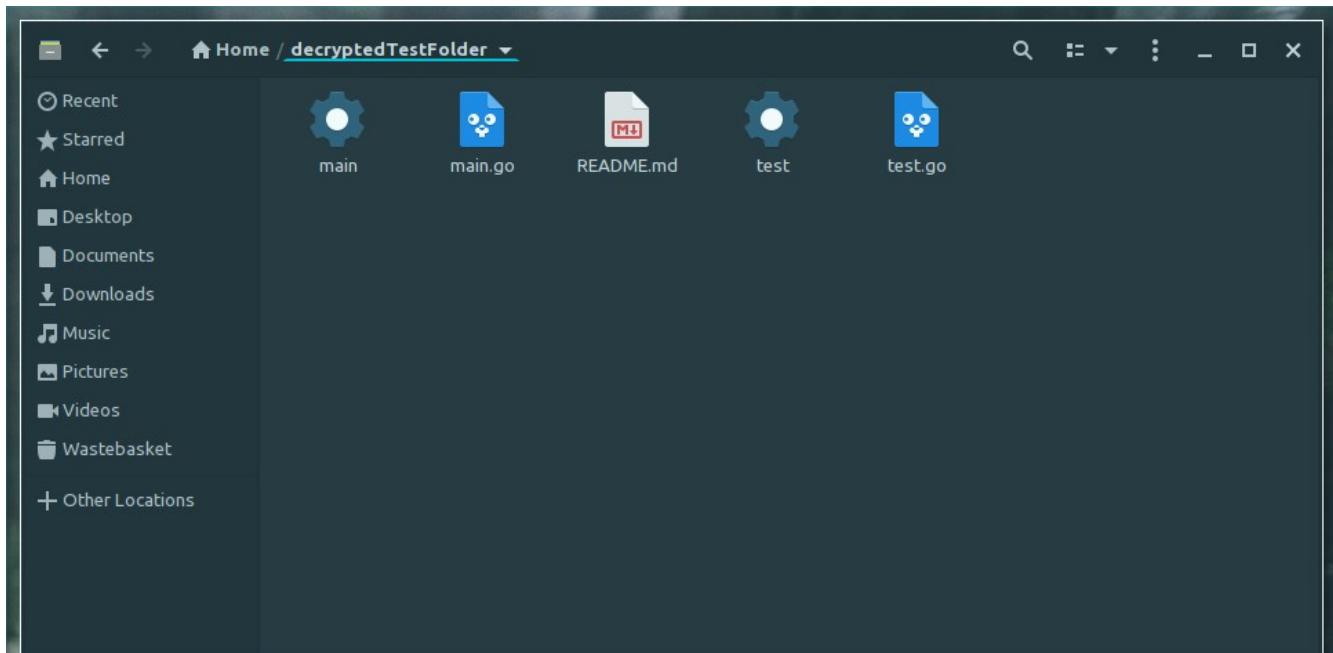
Before:



Input:



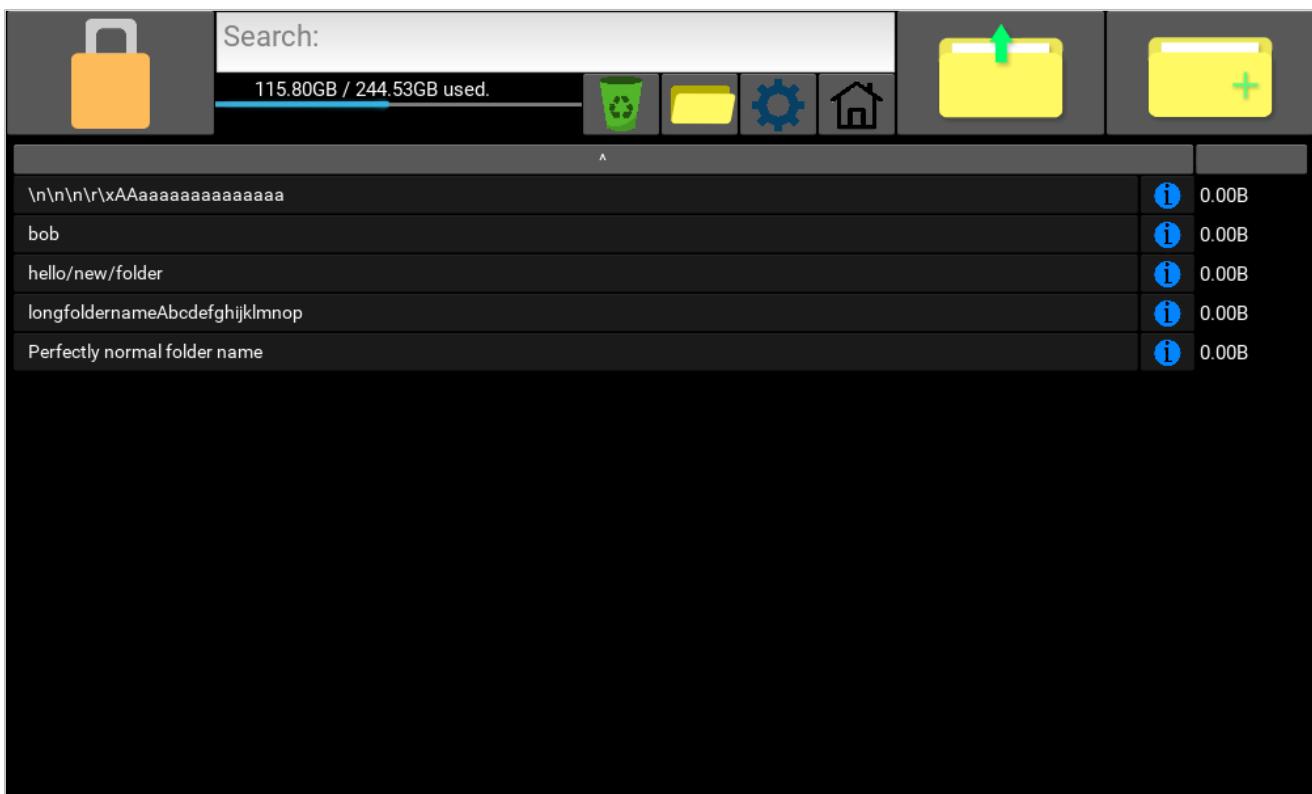
After:



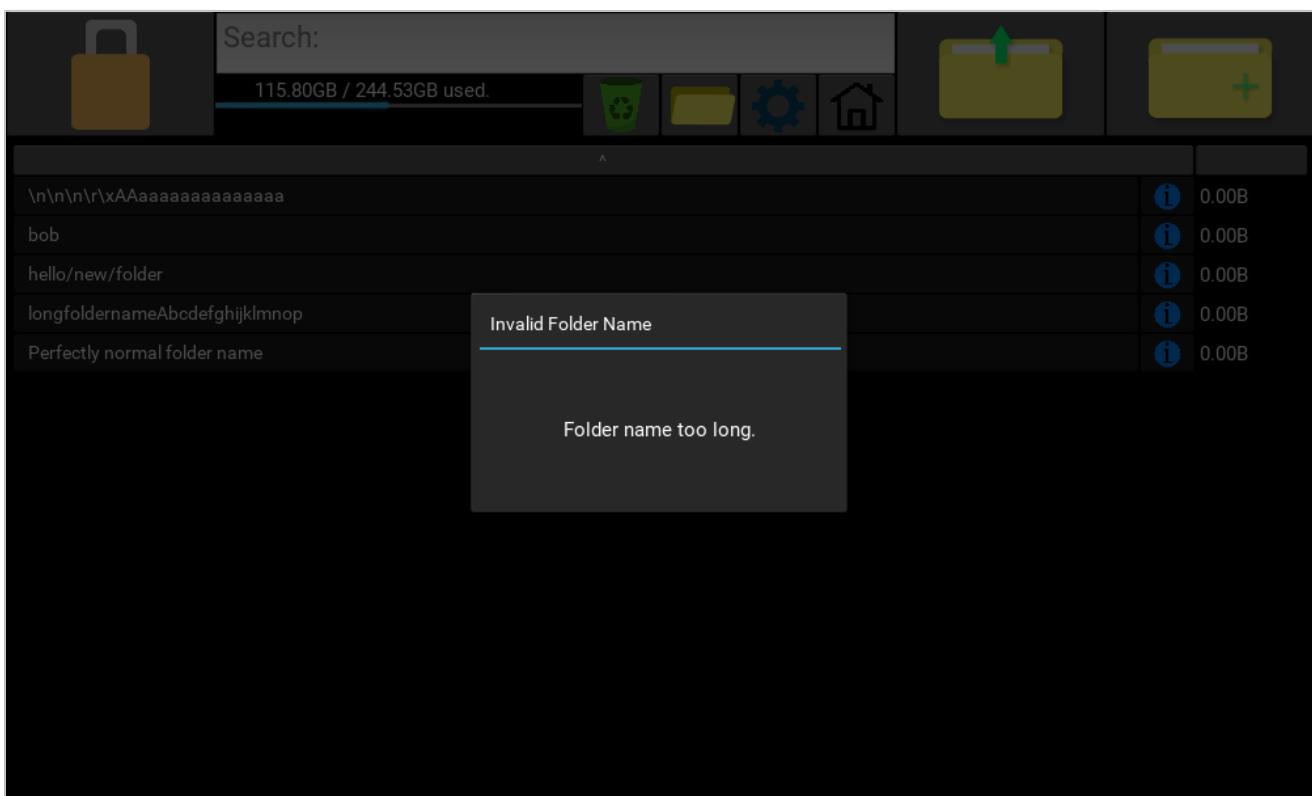
Test Number	16
Relevant Objective	1.f
Description	Create a new folder inside the vault.
Purpose	Make sure that it can be created.
Test Data	T: [1]: "bob" [2]: "longfoldernameAbcdefghijklmnop" [3]: "Perfectly normal folder name" (Possibly Erroneous) E: [4]: "\n\n\n\r\xAAaaaaaaaaaaaaaaaa" [5]: "FOLDER NAME THAT IS WAY TOO LONG aaa" [6]: "hello/new/folder"
Expected Outcome	[1-4, 6]: Folder should be created with no issues. [5]: A popup should show telling the user that the file name is too long.
Actual Outcome	[All]: Pass

Evidence:

File names 1-4 and 6:



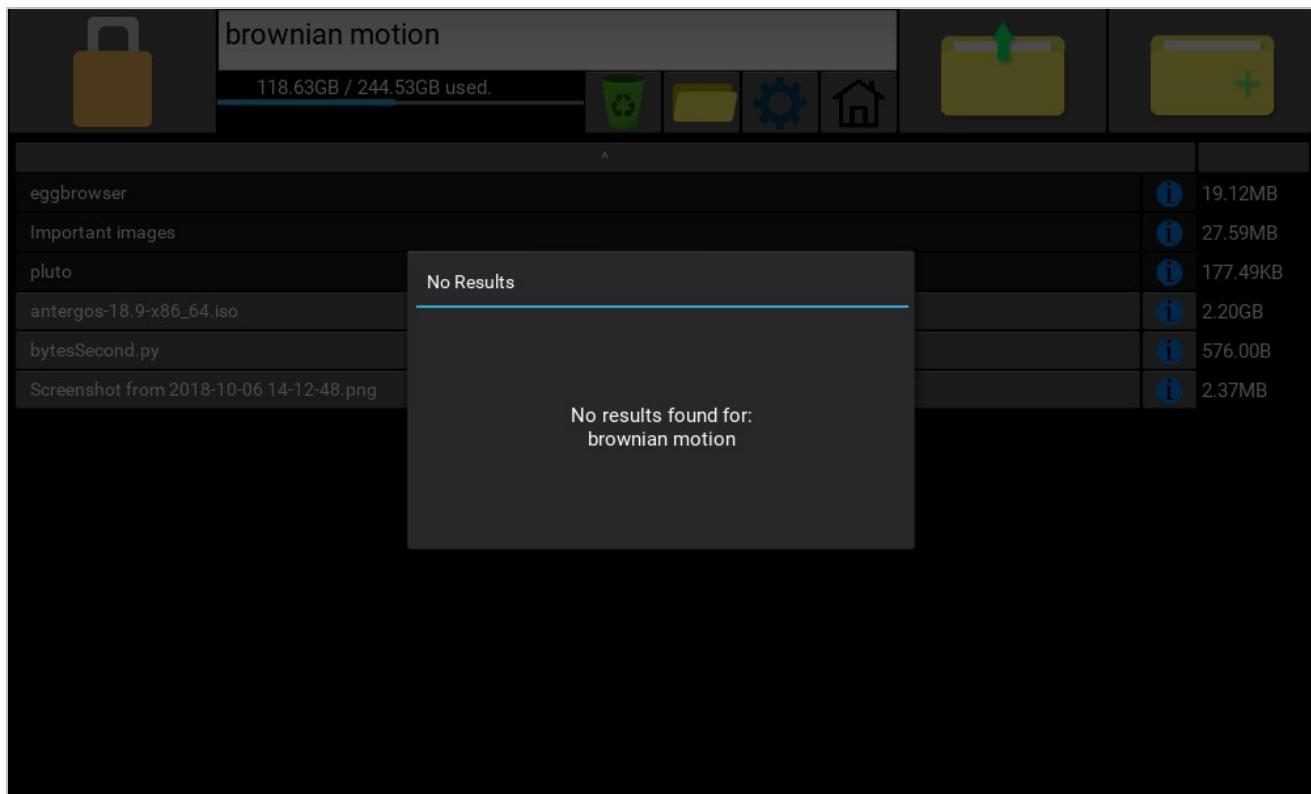
File name 5:



Test Number	17
Relevant Objective	1.r
Description	Attempt to go up one directory while in the root directory of the Vault.
Purpose	Check that the user is confined to the vault, so the program does not try to decrypt file names that are not encrypted.
Expected Outcome	The displayed files should remain the same when I click to go up.
Actual Outcome	Pass

Test Number	18
Relevant Objective	1.q
Description	Search for an item that is not in the vault.
Purpose	Check that the user is warned when the item is not found.
Test Data	"brownian motion" - search term (it is not in the vault).
Expected Outcome	A popup should open warning the user that the file could not be found.
Actual Outcome	Pass

Evidence:



Test Number	19
Relevant Objective	1.q
Description	Search for an item that is in the vault.
Purpose	To check that the item is found.
Test Data	A lot of files (534 files). Some of them contain the string "processing", so I will search for the folder "processing-3.4", which there is only 1 of.
Expected Outcome	Should only show the folder "processing-3.4"
Actual Outcome	Pass

Evidence:



Test Number	20
Relevant Objective	3.g
Description	Open a file then lock the program with the file open.
Purpose	Check that the file is removed regardless of if it is open.
Expected Outcome	File should be removed and should not be in /tmp/FileMate/file.
Actual Outcome	Pass

Evidence:

```

~ $ cd /tmp
/tmp $ # File has been opened
/tmp $ ls | grep "FileMate"
FileMate
/tmp $ ls FileMate
test12_3.png
/tmp $ # Vault has been locked
/tmp $ ls | grep "FileMate"
/tmp $ 

```

Changed directory to /tmp, which is the temporary folder on my PC. Then I opened the file in the program, then list the files in /tmp, filtering the output for "FileMate" (which is what I ended up calling the program).

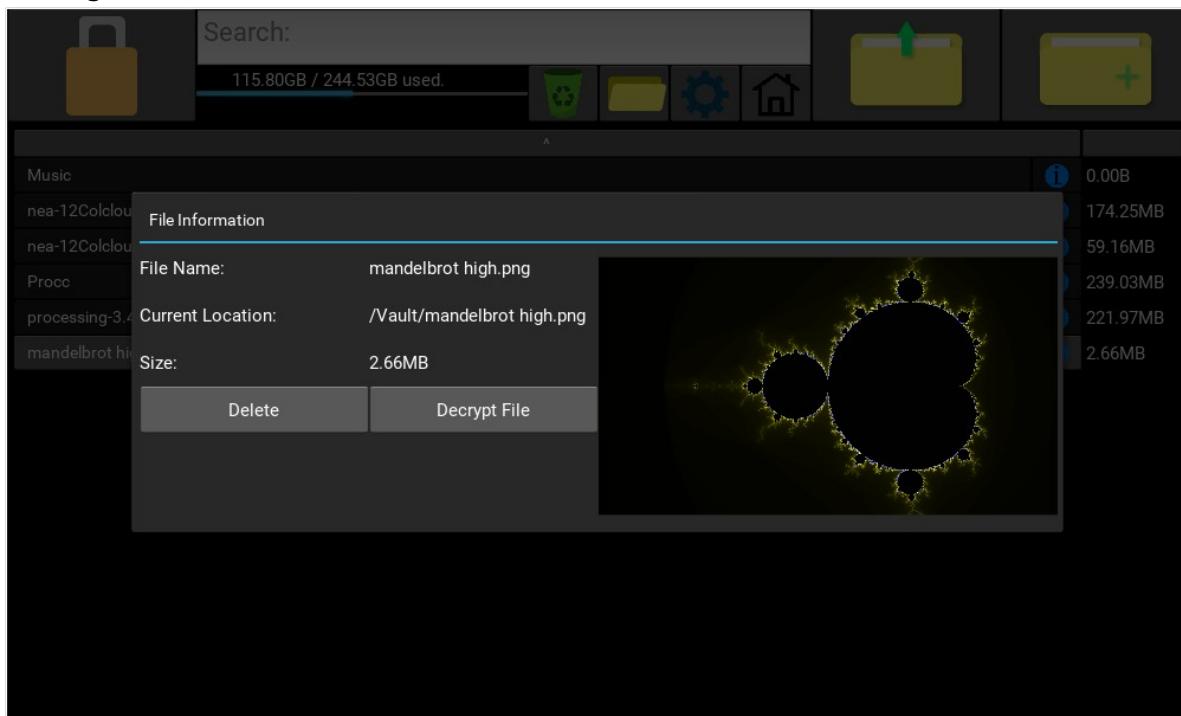
Then I listed the contents of /tmp/FileMate/, to view the temporary FileMate files.

Then I locked the app, and checked again for the FileMate folder, and it had been deleted.

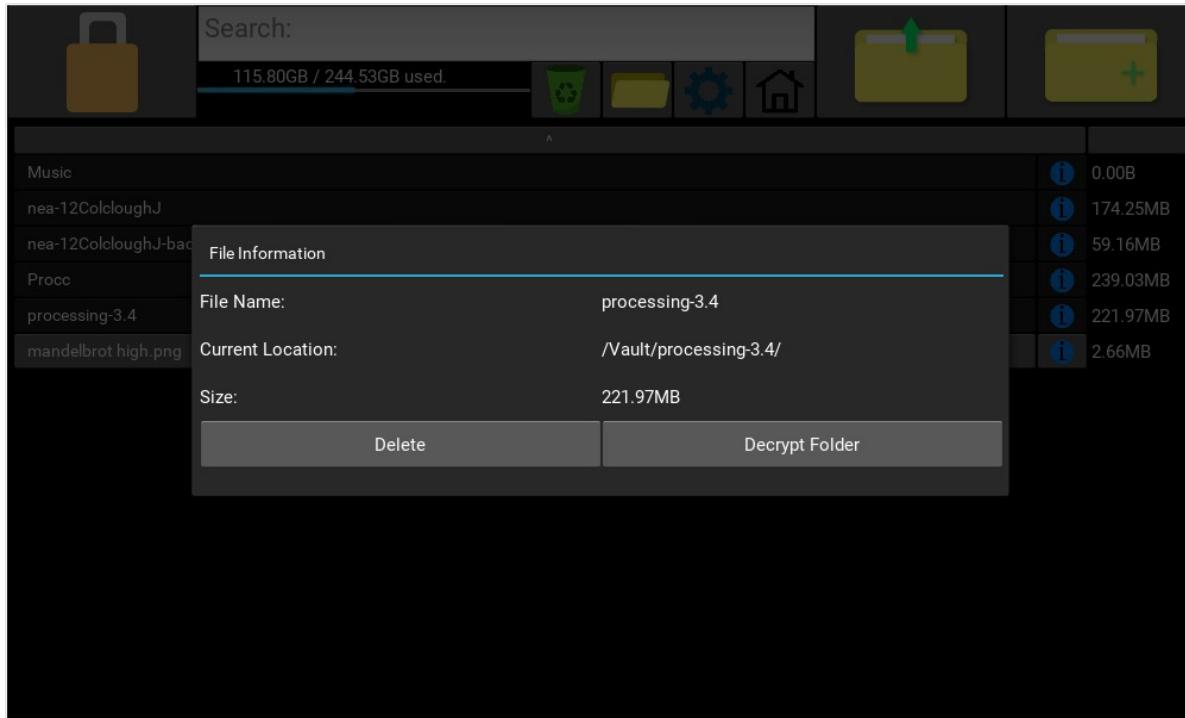
Test Number	21
Relevant Objective	1.j
Description	View more information on an image, then view more information on a file that is not an image.
Purpose	Check that a preview of the image is shown, and if the file is not an image then a preview is not shown.
Expected Outcome	The preview should be shown for the image, but not shown for the regular file.
Actual Outcome	Pass

Evidence:

With an image:



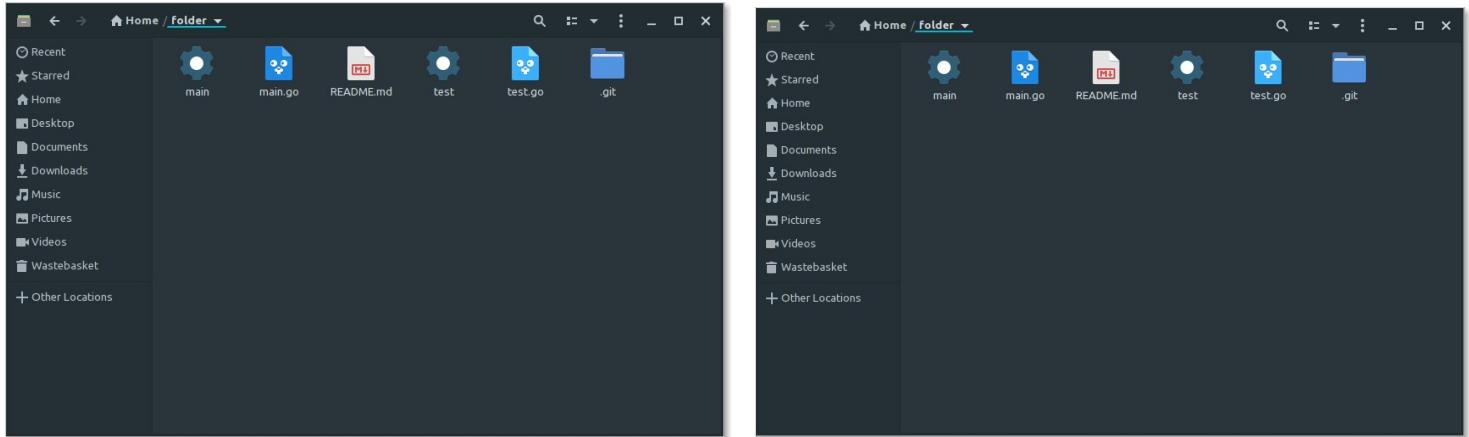
With a folder:



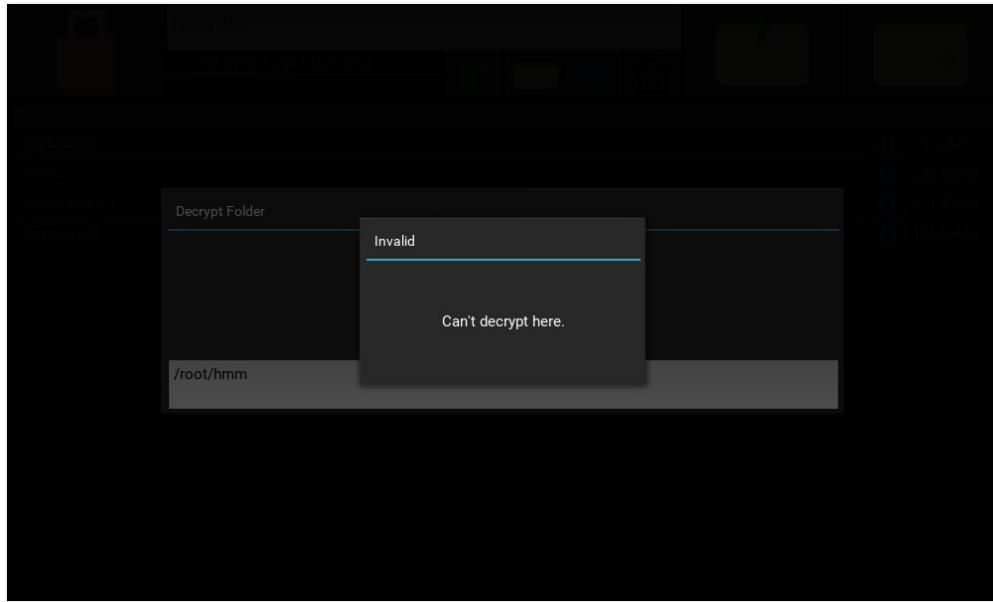
Test Number	22
Relevant Objective	1.d, 3.i
Description	Decrypt folder to different locations.
Purpose	Make sure that the user can't break the input.
Test Data	<pre>T : [1]: "/home/josh/folder/" [2]: "/home/josh/folder" [3]: "/home/joshaaaaaaaaaaaaaaaaaaaaaa/" (Potentially Erroneous) E: [4]: "/root/hmm" (Erroneous if user does not have permission) [5]: "home/josh/folder/" [6]: "/home/josh///////////folder/" B: [7]: "/home/josh/\n\n\n" [8]: "/home/josh/aaa/a/a/aa/"</pre>
Expected Outcome	<p>[1-3]: Should decrypt no problem. [4]: Popup should warn user that it cannot be decrypted here. [5]: Same as [4] [6]: Same as [4] and [5] [7]: May work fine. [8]: Should make the file path correctly then decrypt files to aaa/a/a/aa/.</p>
Actual Outcome	[All]: Pass

Evidence:

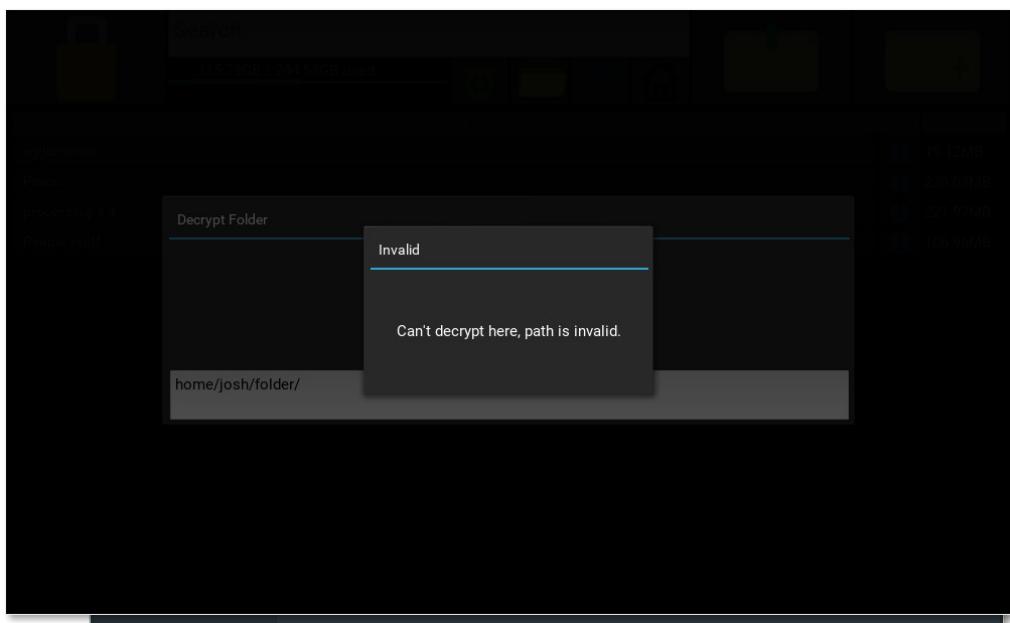
[1-3]:



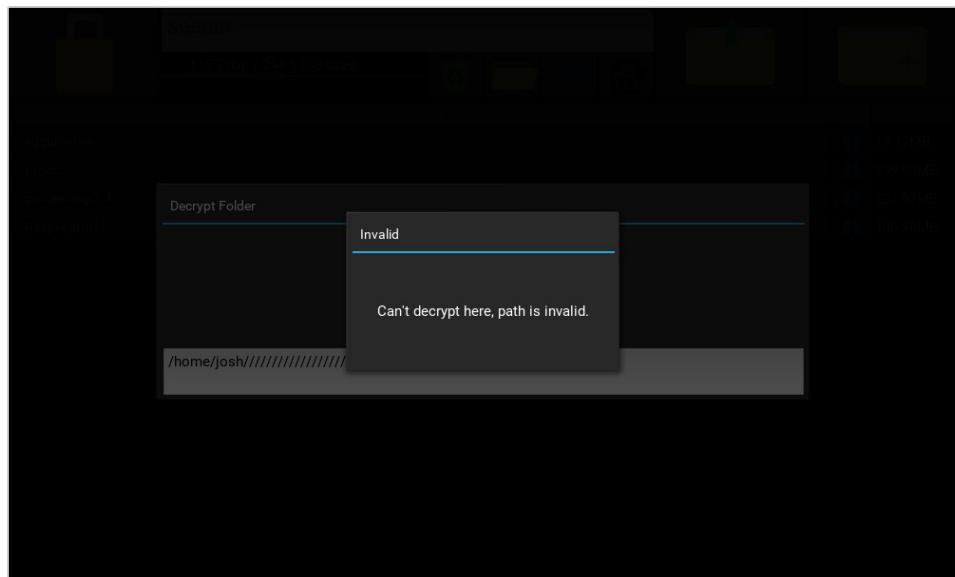
[4]:



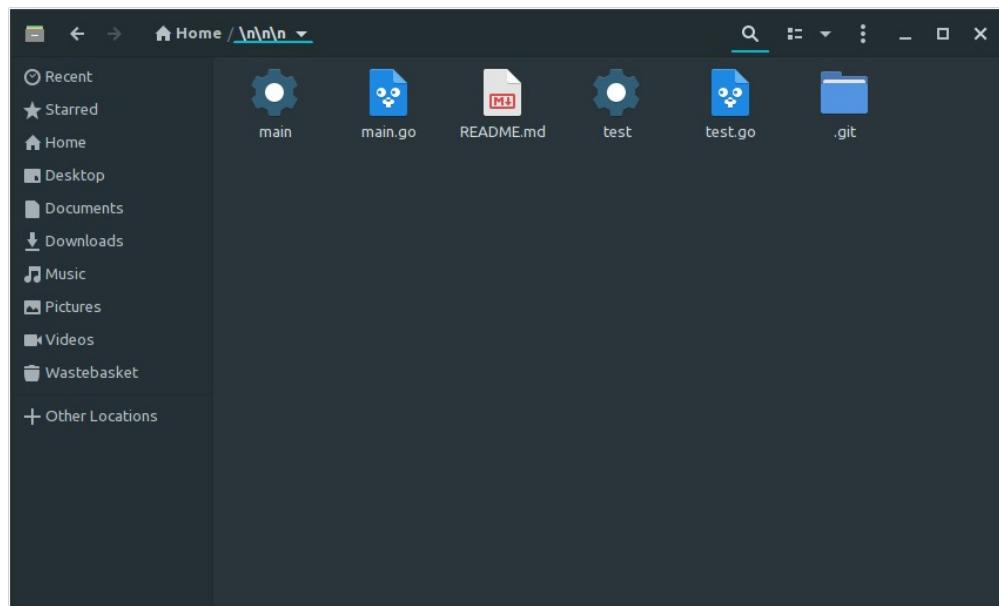
[5]:



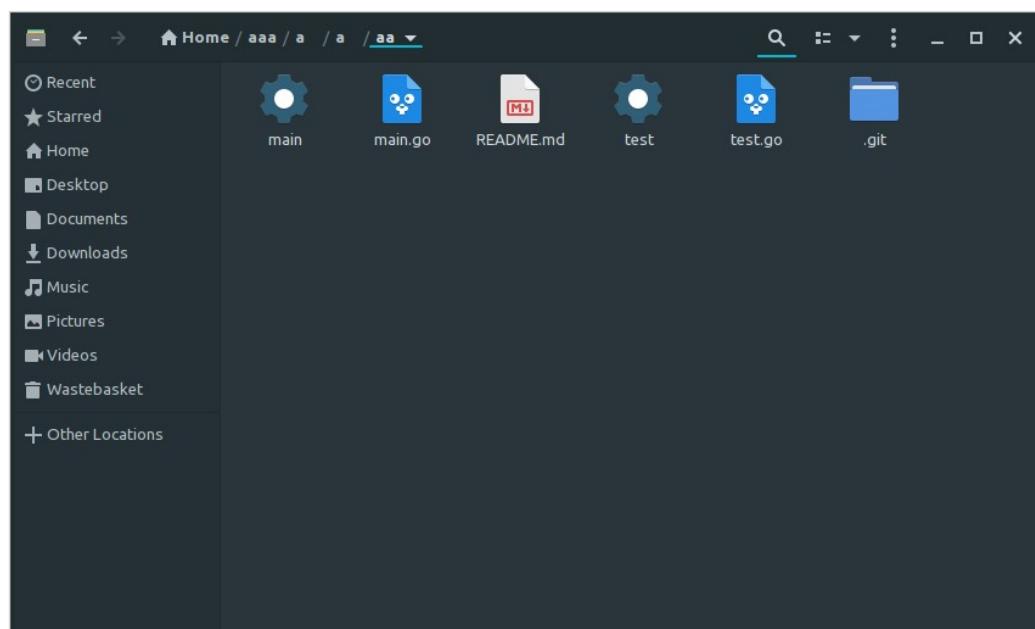
[6]:



[7]:

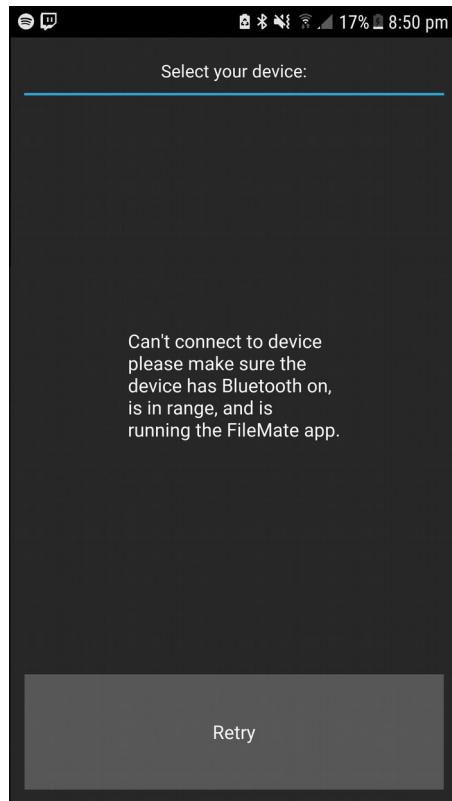


[8]:



Test Number	23
Relevant Objective	2.e
Description	On the mobile app, click to connect to a device that is not running the PC app.
Purpose	To check the user is warned that the program can't connect to the device.
Expected Outcome	A popup should be shown on the mobile telling the user that the app could not connect to the device.
Actual Outcome	Pass

Evidence:

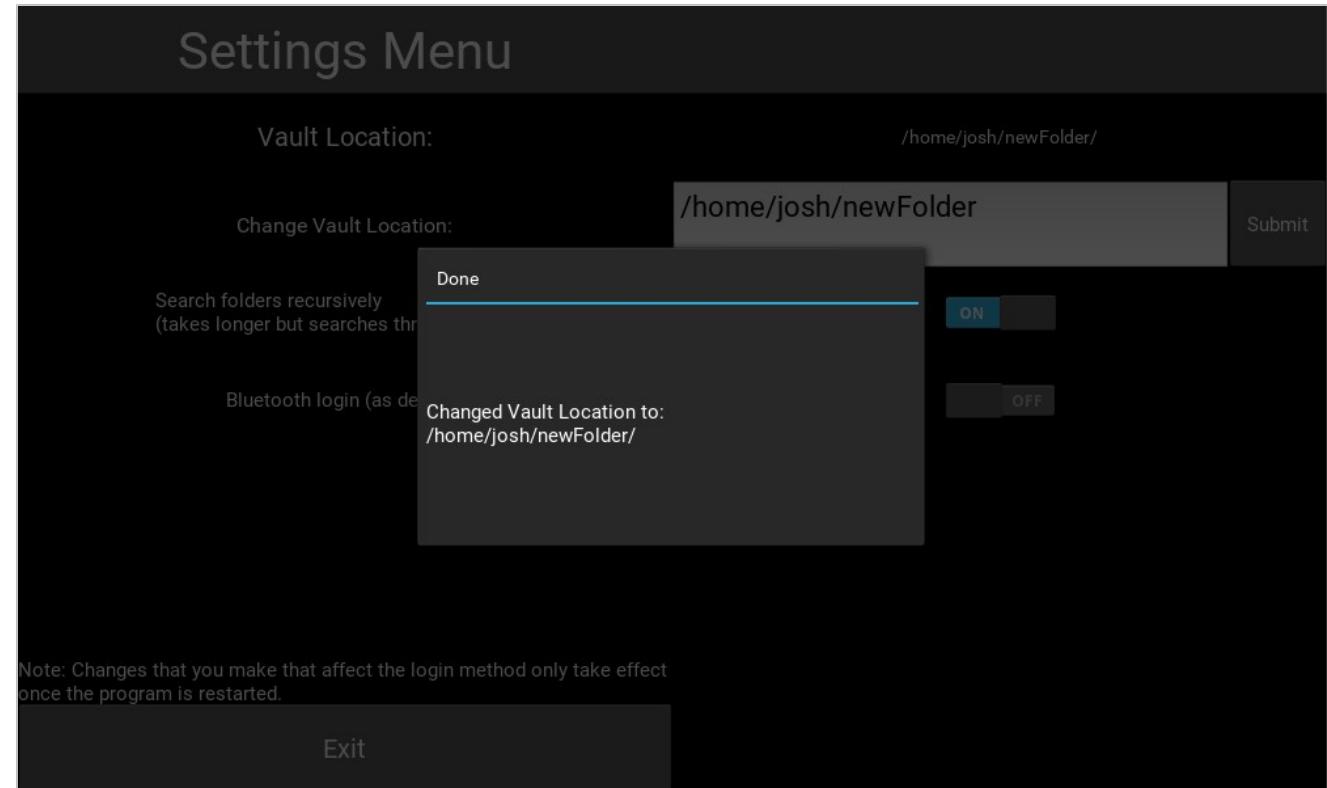
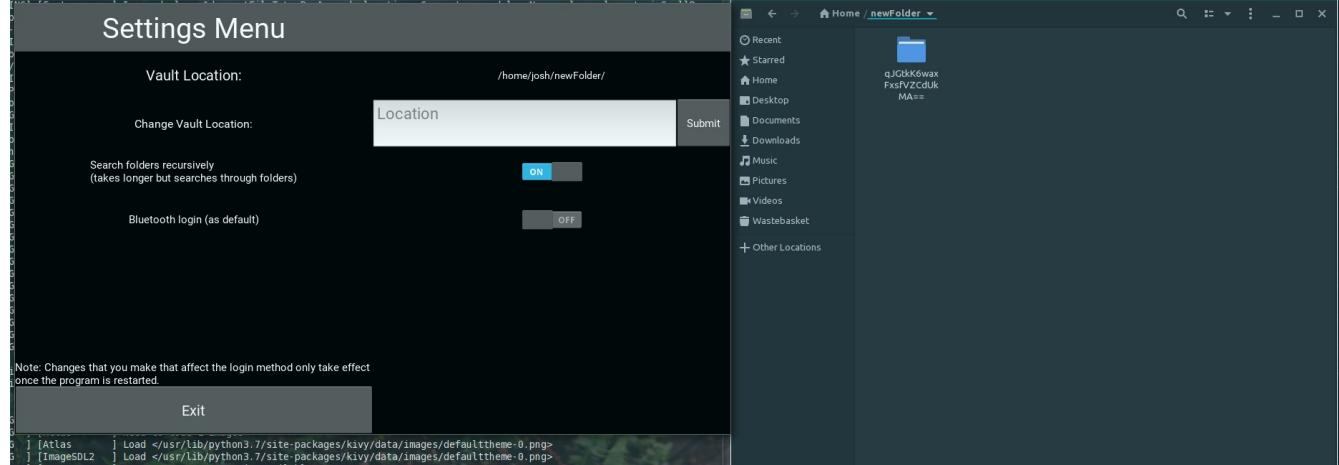


Test Number	24
Relevant Objective	3.a
Description	Change the vault location (using the Settings screen).
Purpose	Make sure the user can easily change vault location.
Test Data	<p>From "/home/josh/EncryptedDir/" to:</p> <p>T: [1]: "/home/josh/newFolder/" [2]: "/home/josh/newFolder" [3]: "newFolder" [4]: "newFolder/subFolder/" [5]: "newFolder/subFolder"</p> <p>B: [6]: "/home/josh/folderWithEncFiles/"</p>

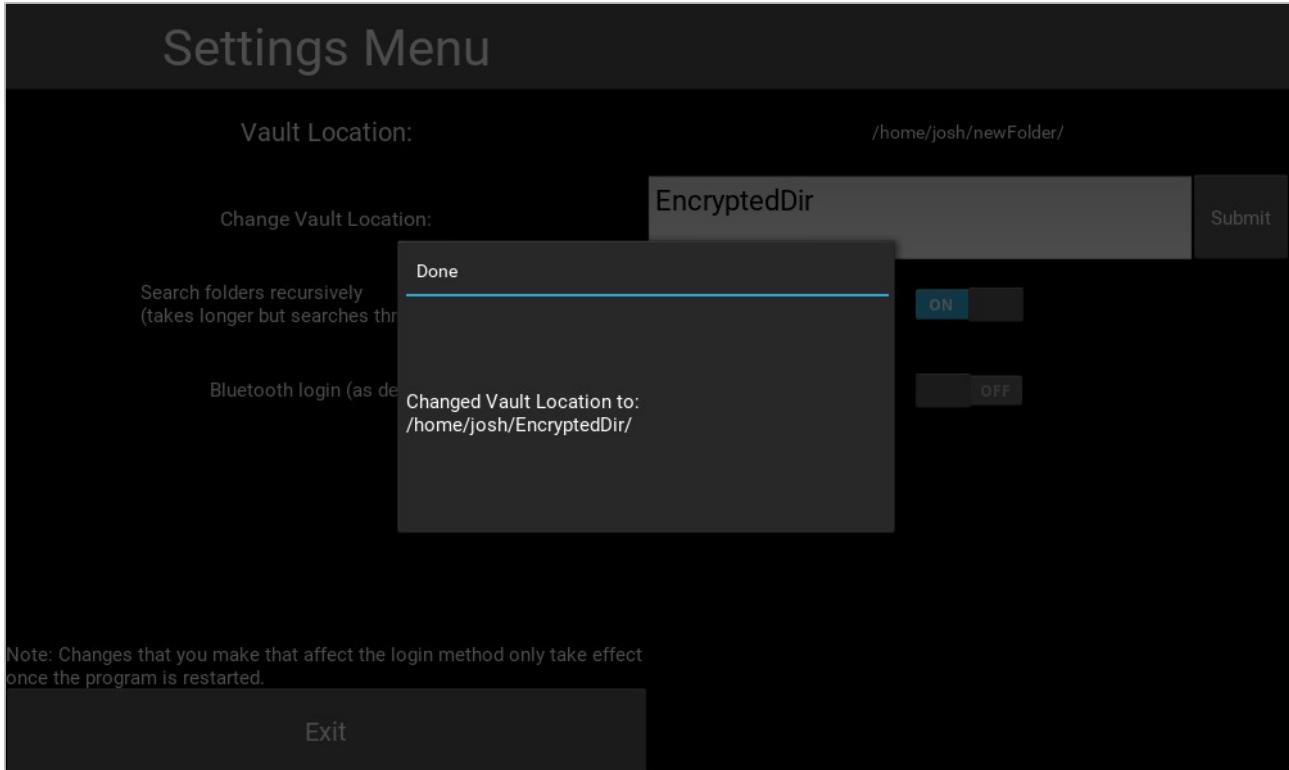
	E: [7]: “/home/josh/folderWithEncFilesDifferentKey/”, log in with the different key, then change vault to “/home/josh/EncryptedDir/”.
Expected Outcome	[1-2]: Should create folder. [3-5]: Should create folder relative to the input. [6]: Has the same key as the original location, so should work just fine. [7]: Hopefully handles it ok.
Actual Outcome	[1-6]: Pass [7]: Tried to decrypt file names, but showed gibberish.

Evidence:

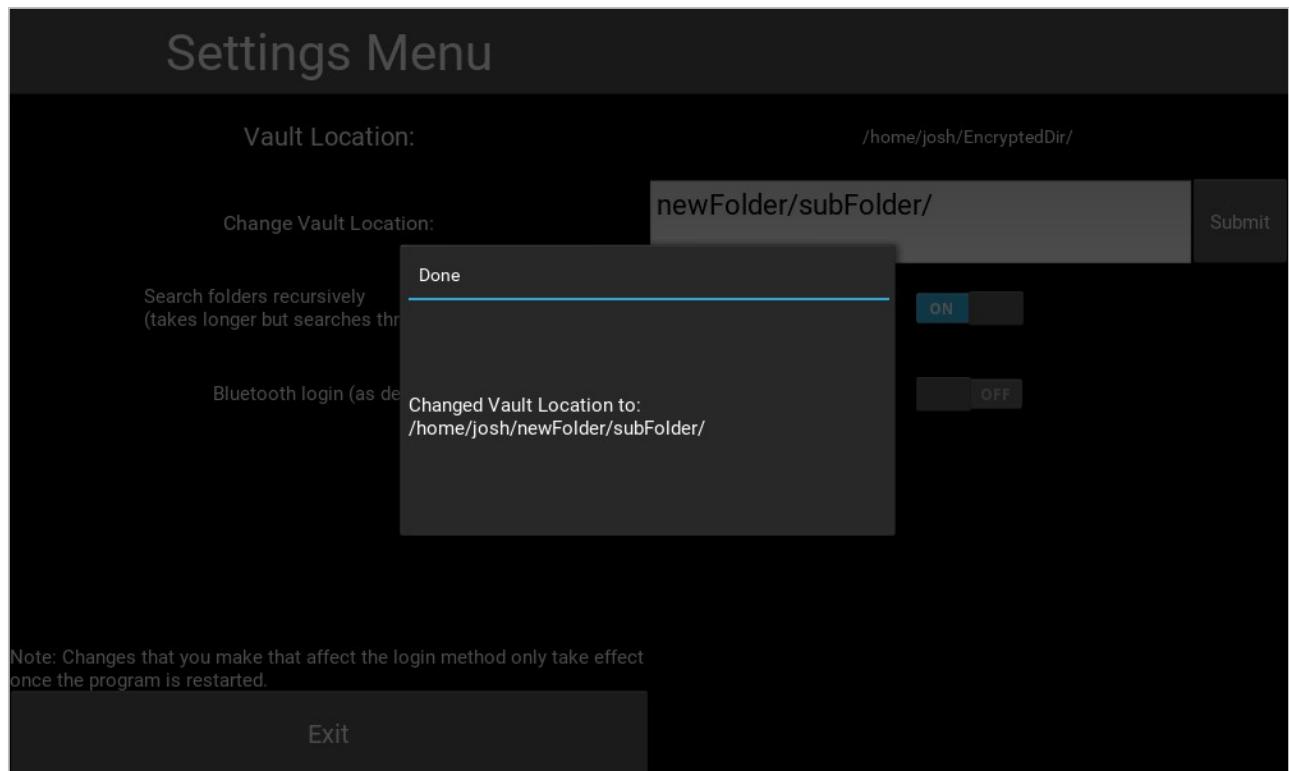
[1, 2]:



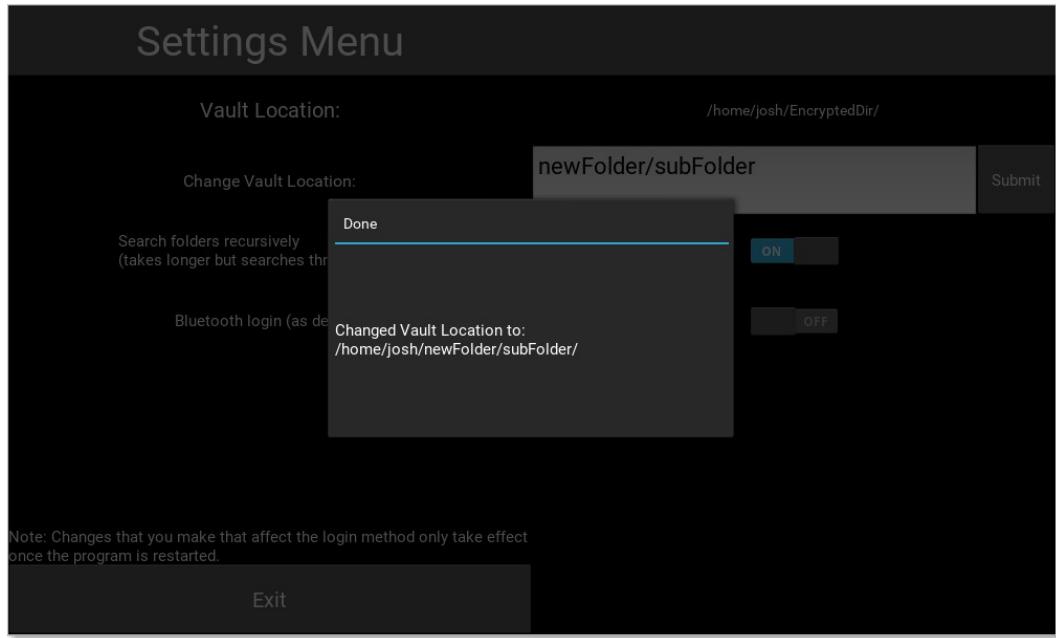
[3]:



[4]:

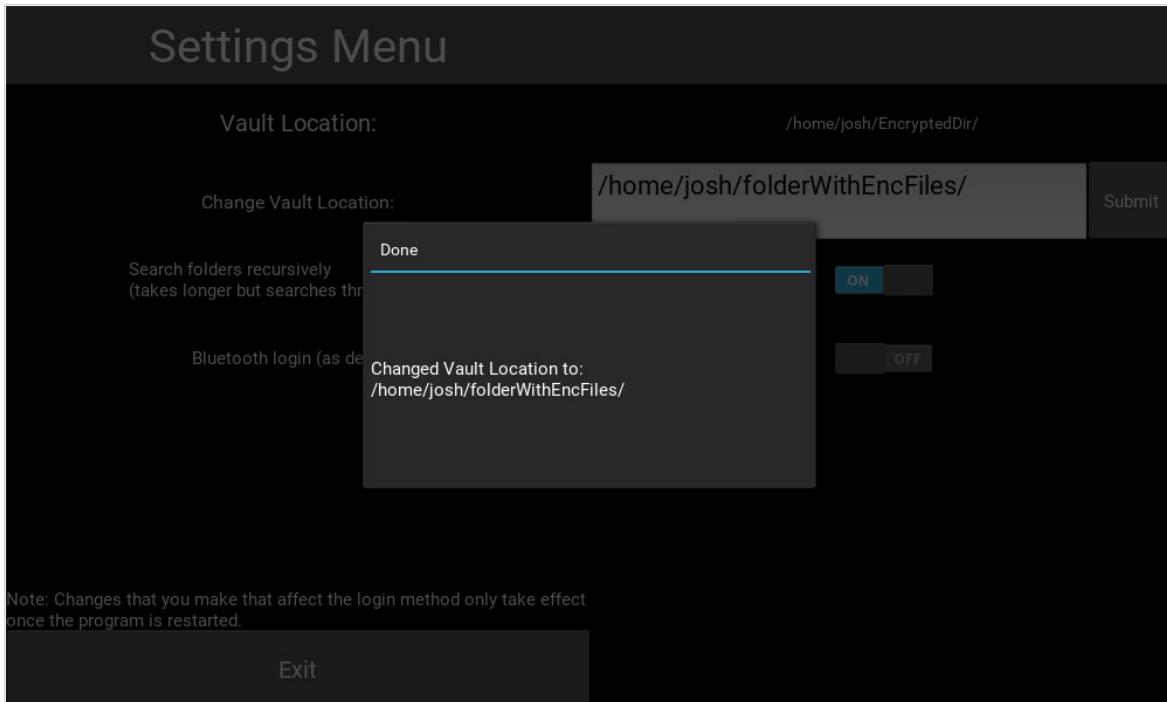


[5]:

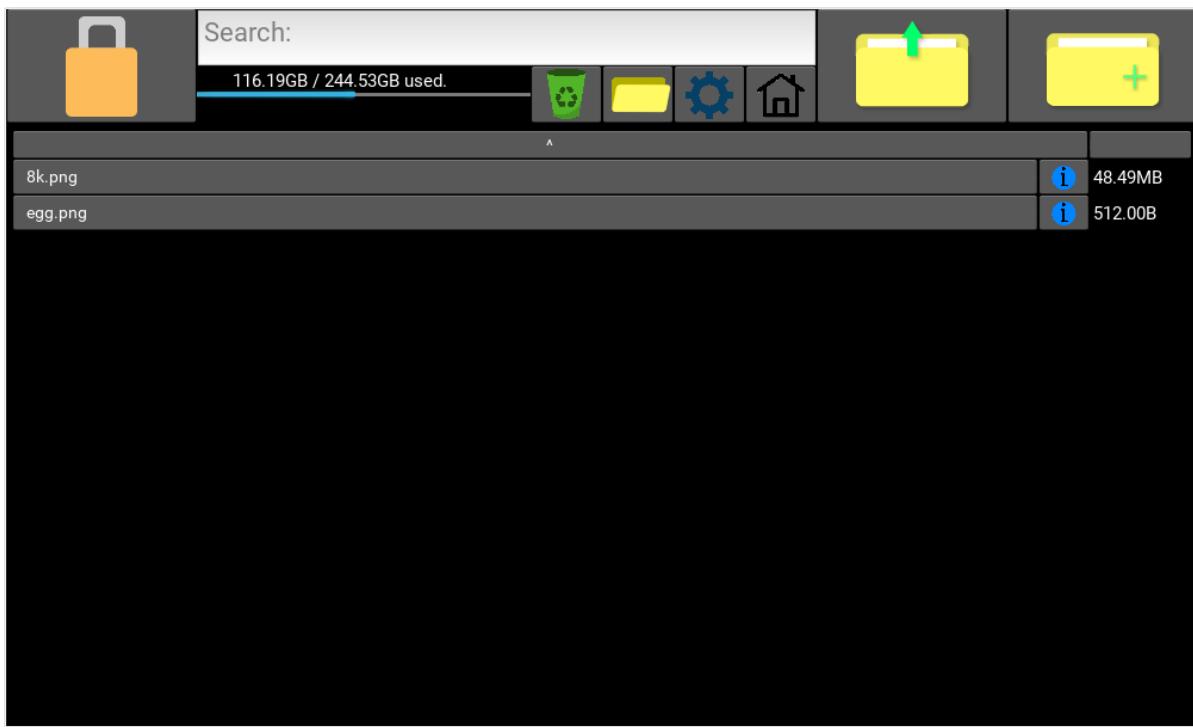


[6 (in 2 parts)]:

Change vault to /home/josh/folderWithEncFiles/



View vault:



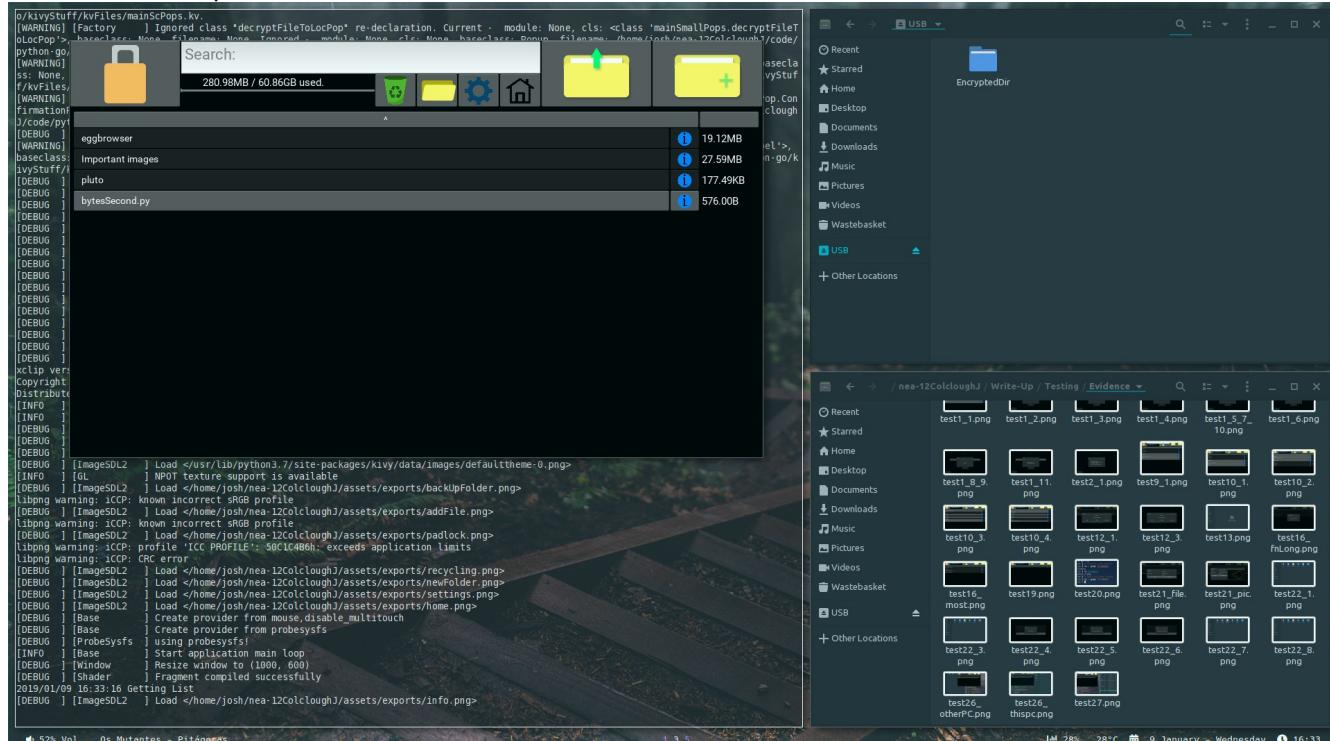
[7]: (The one that broke)



Test Number	25
Relevant Objective	3.a
Description	Move a vault onto a USB stick, and access it on another computer.
Purpose	To make sure that use case is possible.
Expected Outcome	The user should be able to access all files in the vault using my program, if they enter the key correctly and if the vault location is set correctly in the configuration files.
Actual Outcome	Pass

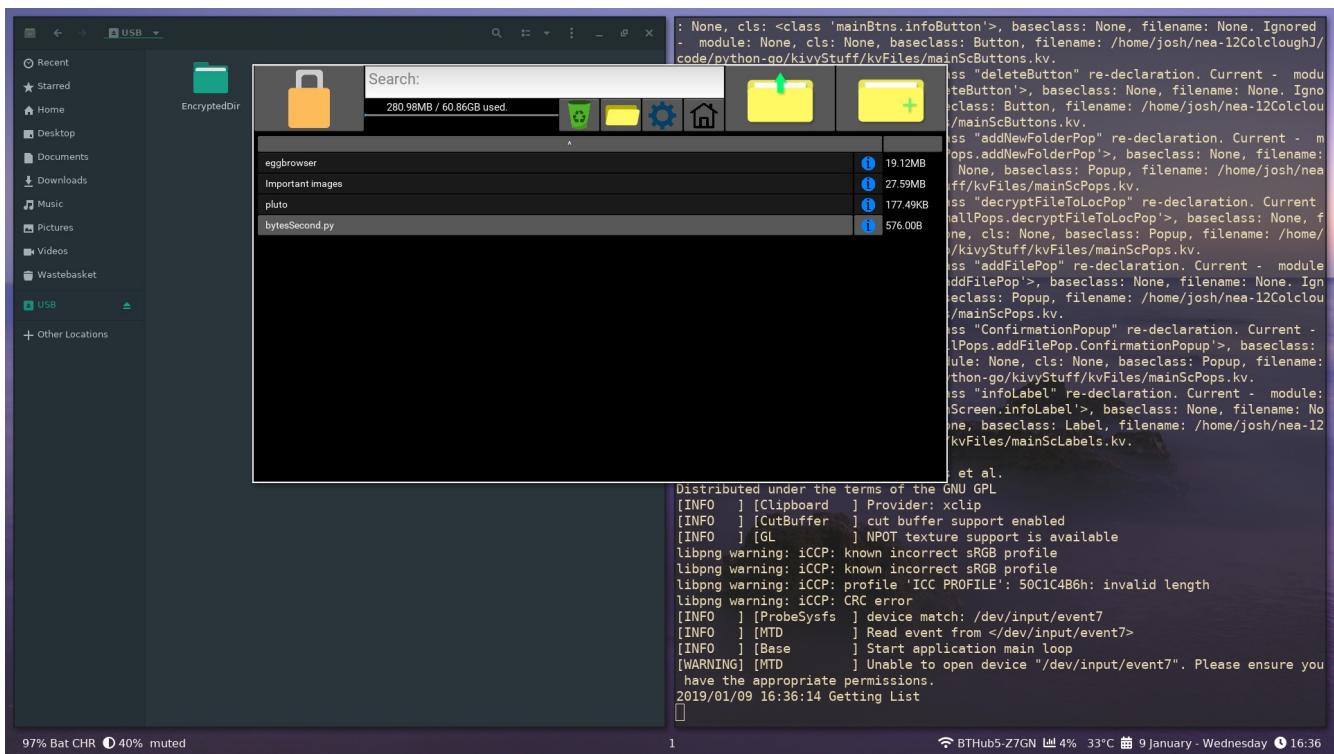
Evidence:

USB on this computer:



Left is the program, top right is the contents of the USB (= EncryptedDir).

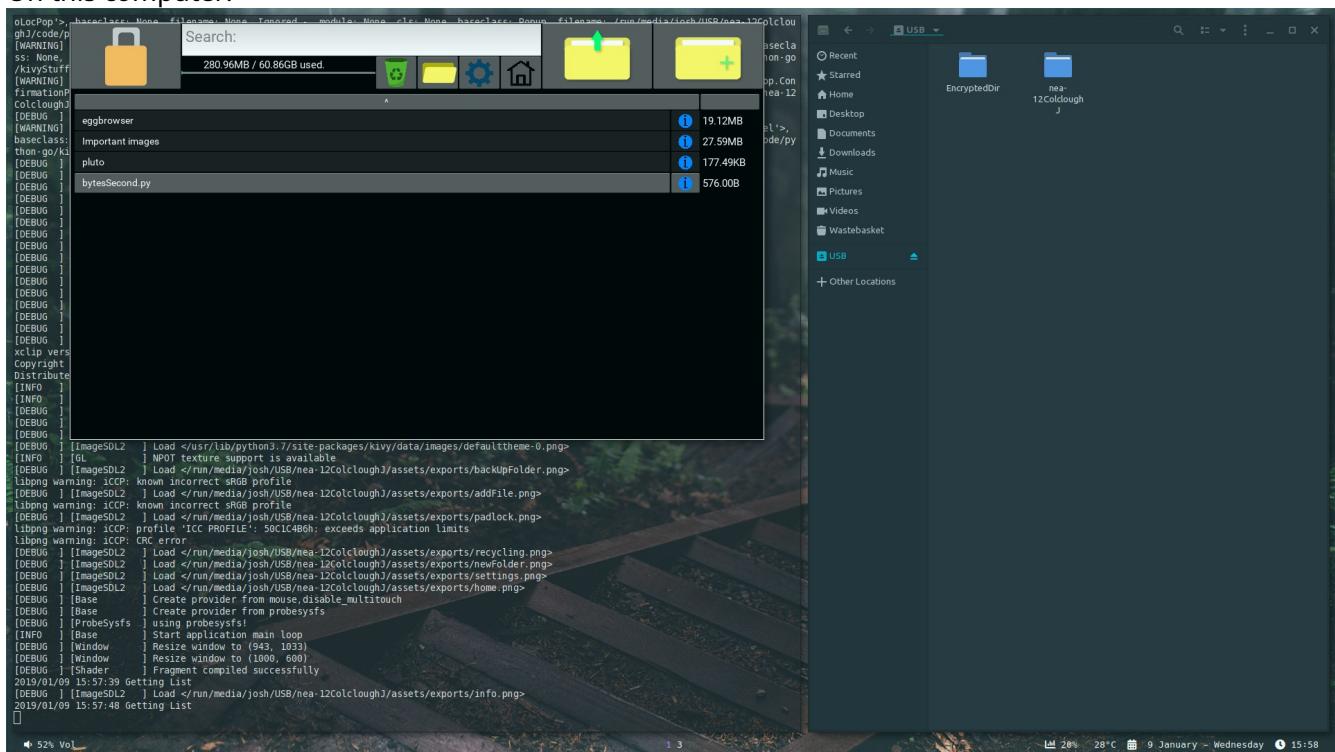
USB on another computer:



Test Number	26
Relevant Objective	3.a
Description	Move the vault AND the program onto a USB stick, and run the program from the USB stick on multiple computers.
Purpose	Check that the use case is possible, and working on different machines is smooth.
Expected Outcome	The user should be able to access all files in the vault using my program, if they enter the key correctly and if the vault location is set correctly in the configuration files, using a relative vault location.
Actual Outcome	Pass

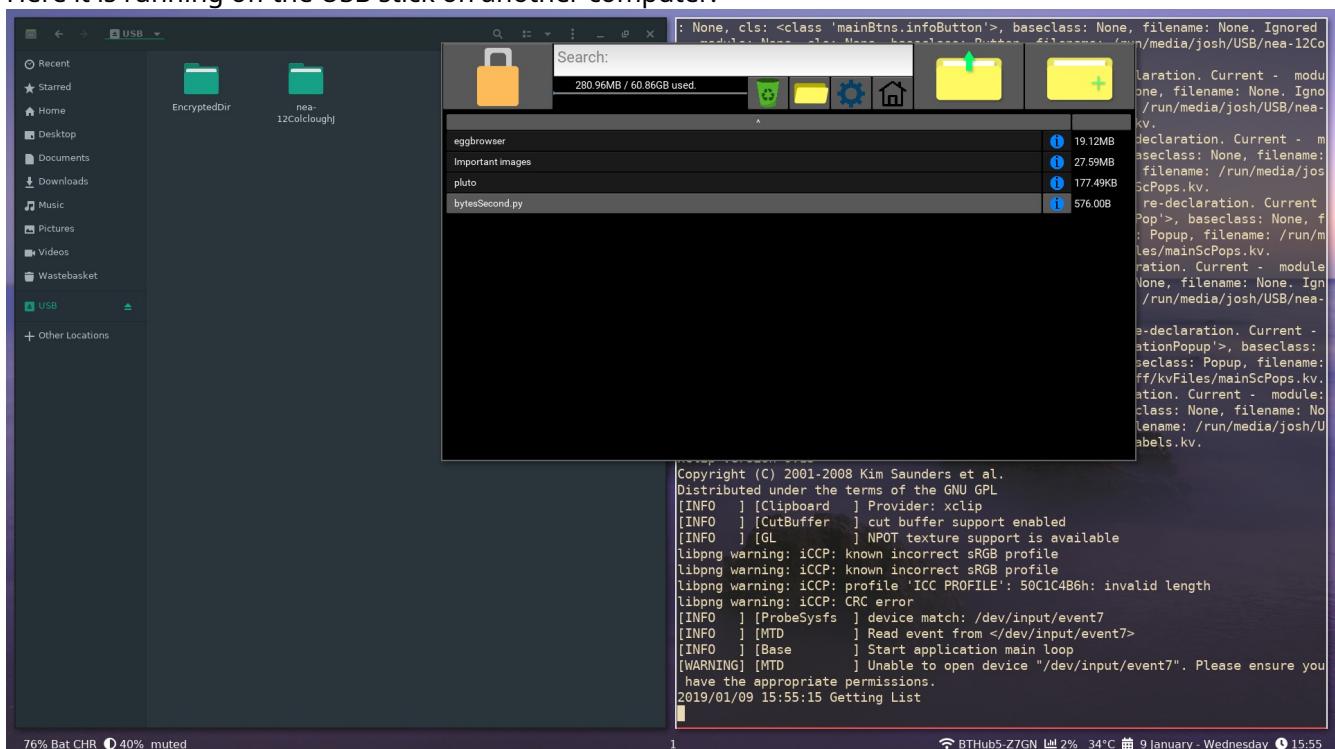
Evidence:

On this computer:



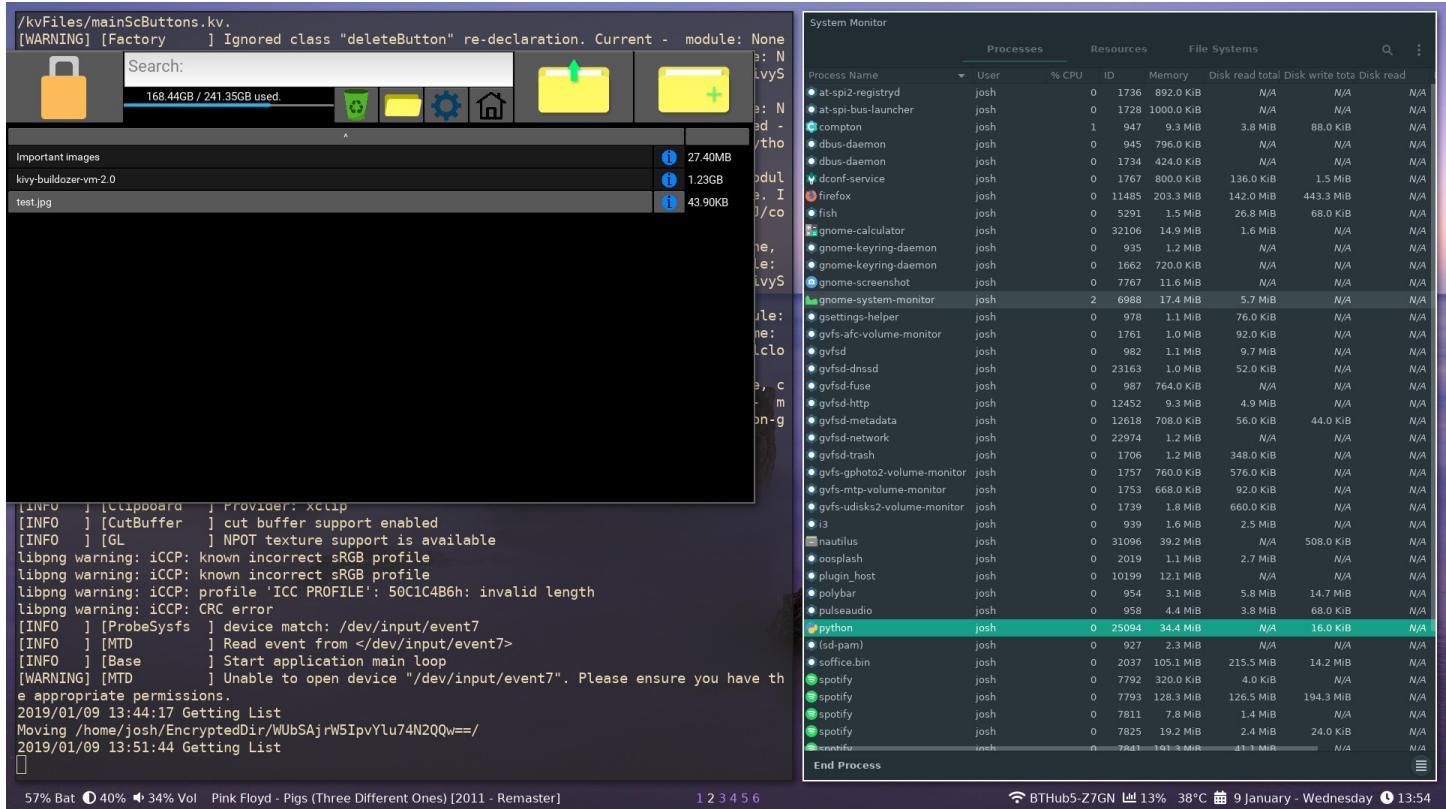
Right tile shows contents of USB stick, left is the program running showing the contents of USB/EncryptedDir.

Here it is running off the USB stick on another computer:



Test Number	27
Relevant Objective	1.h.i
Description	Check my program is not using too much CPU time when idle (<5%), and not too much memory (<100 MiB).
Purpose	To make sure that my program is not using up CPU time when it is really doesn't need to be doing anything, and that it is not too memory hungry.
Expected Outcome	Should be below 5% CPU usage and 100 MiB of memory, as it isn't a huge program.
Actual Outcome	Pass- 0% CPU usage and 34.4 MiB memory usage. The memory usage is still a bit high (comparable to my file manager at 35 Mib), and I believe this is due to the loading of images for the info button on files, when it could probably be loaded once instead of once for every button.

Evidence:



Here I have my program sitting idle, with a task manager open (the `python` task is highlighted).

Evaluation

Objectives

In this section I will go through each of my objectives and comment on their completeness.

1. GUI should:

- a. Be easy to use:
 - i. Logically laid out.
 - ii. Have simple options.

I feel that my program is quite easy to use due to the layout, the important buttons are the largest, and the file browser fills most of the screen, which is the most important part of the main screen. The settings screen is also quite easy to use, as the switches for the True/False configuration values are very straight forward to change, and it is clear what each setting changes.

b. Display the files currently stored in the vault, along with the file extension and the size of the file.

Works fine. The file extension is at the end of the file name anyway. The size is read from the file in the vault.

c. Display the storage space remaining on the storage device the program is running on.

Works fine.

d. The user should be able to easily encrypt and decrypt files:

- i. Using easy to access buttons in the UI.
- ii. Using drag and drop from outside of the program.
- iii. Decrypt to a directory specified.

All of the above work, as long as you have drag+drop on your machine.

e. Have an options menu, including the options to:

- i. Change from 128 bit security to 256 security. 128 bit is the bare minimum.

Didn't end up adding this, as it would require a lot more code and time for a feature that very few people would probably use.

ii. Change the location of the vault.

Can be changed relative to the program, or as an absolute path.

- iii. Set the default login method (Bluetooth or no Bluetooth).

Also done using a switch in the settings screen.

iv. Change if the search in the file browser is recursive or not.

Done using a switch in the settings screen.

f. Make it easy to manage the files in the vault (move to other folders in the vault, rename, delete, etc).

I ran out of time to implement this, as Kivy buttons by default do not have a right click, and it is quite hard to implement. If I were to implement this, I would show a bubble above where the user right clicked, with multiple options to carry out on that file/folder. Alternatively, I could use the information popup to hold each of these options.

- g. Have a secure login screen.
 - i. Ask the user to either input the key via their keyboard (no Bluetooth for that session), or connect via the app.
 - ii. Tell the user if the key is invalid or not, and smoothly transition into the main program.
 - iii. Validate all input.

All tested to be working.

- h. Look relatively good without being bloated.
- i. Allow the user to easily read file names, and easily tell folders and files apart.

Files are coloured in a darker shade than regular files, and the file names are in medium-sized font (relative to the window).

- j. Let the user preview images without opening them (using thumbnails or an information screen).

Works well, as AES decrypts the image quickly, then it is displayed in the information screen. I did not add thumbnails as I thought that it would probably drastically increase the loading time when opening folders containing images.

- k. Be resizeable, and all items on the screen should look ok.

Also looked at in testing. Each widget uses a relative layout, so that when the window is resized, the widgets also resize themselves based on the proportions they were given.

- l. Allow the user to switch between using Bluetooth and using regular login.

Works good.

- m. Make it easy for the user to return to the root folder of the Vault in case they get lost (a "panic" button).

A home button is included on the main screen, and takes the user back to the root folder.

- n. Give the user statistics during files being enc/decrypted, including:

Works well. I cover how this works in the Technical Solution, and it is tested to give accurate readings in Testing. Gives the user a nice readable output, displaying all useful information.

- o. Allow the user to sort the files by name or by size.

The user can do this using the two buttons above the list of files on the main screen.

- p. Allow the user to configure default settings using a configuration file that is easy to use.

The configuration is relatively simple, and easy to edit manually in case of an issue.

- q. Allow the user to search for file names in the vault.

Works well, however is quite slow because it is done in Python. I would have liked to do this in Go, however passing the file objects into Go would have been a pain, as they are not all in the same directory like the sorting buttons are (which is why I could do those in Go).

r. Not break when doing arbitrary tasks such as browsing the files.

See Testing.

2. App should:

a. Be easy to use.

The GUI could be improved on the mobile side, as the flow of the program is quite weird, going from the device screen, into the pad screen (this part is fine), and then into a blank screen with only one option. This last screen is a good idea if you have multiple options, however I only supply one, so it is a bit redundant. If I were to continue working on this project I would leave it in for future functionality. However, other than this each screen works well individually. The pad screen has large buttons with quite large font. The main screen is basic, but functional.

b. Connect via Bluetooth to the PC.

Connects seamlessly to the PC using pyjnius java classes.

c. Allow the user to input their pin code easily.

The Pad Screen is easy to use.

d. Tell the user if the pin code is invalid or not.

(b, c, d) Works fine.

e. Make it easy to recover from mistakes (e.g invalid pin code, or if they make a typo).

In the pad screen, the key input is reset to be empty if it is incorrect, and the user can easily delete the last digit while entering the key. If the user does get the key incorrect, a small popup shows up telling them, and then they can easily dismiss it.

f. Allow the user to see a list of files currently in the vault, and let the user download those files onto their mobile device.

The file browser is quite good, with a large scrollable list of files, however there is no way of distinguishing between files and folders at the moment. This is due to the protocol, as I decided against sending pickled objects between the PC and phone. I decided not to do this because I was using Python 2 for the mobile app, and Python 3 for the PC app. The way Python 2 handles bytes is quite different to Python 3, and so pickle had quite a few differences. If I was to continue working on this I would build the mobile app in Python 3, however at the time I was having issues building the app for Python 3. Other than this, the download feature is pretty good, as it is quite reliable (as far as I have tested).

3. File handling:

a. Store the encrypted contents in the location specified by the user.

Works fine.

b. Encrypt and decrypt relatively quickly, while still being secure.

In Testing, I benchmark AES, and the speed is adequate. I have also tested my implementation against test vectors to ensure it has been implemented correctly.

c. When the Bluetooth device goes out of range or disconnects (if using Bluetooth), delete all decrypted files and lock the program until the pin code is input correctly again.

Works fine.

e. Have a recycling bin so that the user can recover their files.

There is a recycling bin and it works fine, however it cannot restore files back to their original place in the vault. If I were to continue working on this project, I would save a pickled Python object in the recycling bin, that contains a class with attributes listing each file in the recycling bin with where they came from originally.

f. When a file is opened, check for changes once it is closed.

Works well, and BLAKE is of adequate speed, as tested in Testing. I also tested my implementation of BLAKE against official test vectors, so that my implementation is accurate to avoid hash collisions.

g. Files stored in the vault should not be accessible from outside of the app.

Could not open a file in the vault from my normal file manager (it worked).

h. Names of the files stored in the vault should also not be view-able from outside of the app (encrypt the name).

Each file and folder name is encrypted, and encoded to base64. When the files are listed within the app, they are decoded, decrypted and then encoded into string again, and displayed.

i. Allow the files/folders to be decrypted to an external location.

Works fine. Files can be decrypted to a new folder, with their original name, or they can be renamed when decrypting.

4. Overall system should work on Windows, MacOS and Linux.

Works great on Linux, so it should work on MacOS too, however I have no way of checking this as I do not own a computer that runs MacOS. The program works well on Windows as well, however I had issues installing PyBluez for the Bluetooth capability (it was asking for the Windows SDK, and even after I installed it, it still asked for it...). The rest of the program worked fine, as you can see in this video (type the link into the top bar on your web browser): <https://youtu.be/2dcJAWGE IE>

Things I would do to improve the project

- Parallelise BLAKE (use multiple CPU cores)
 - Would improve overall speed, which is essential for editing very large files.
 - Would make editing smaller files more seamless too, as the waiting time would be reduced by a small amount, but it will be slightly noticeable.
- Make it so you can right click on files to view more options, those being:
 - Move the file to another location in the vault.
 - Copy the file to another location in the vault.
 - Delete the file (move to recycling).
- Improve the recycling bin so that it recovers files to their original location using a pickled class (serialised object), that contains information about each of the files in the recycling bin.
- Change the Bluetooth protocol I made to use serialised objects, to reduce the likelihood that a pattern used in the protocol will appear in the file, and to also send more data more reliably, such as file metadata.
- Make it so that when you change vault location, you can enter a key you would like to set, or if there are files that already exist, logout and ask the user to log in again.
- Display the current status of file transfer in the mobile app while downloading.
- Be able to send files from your phone to the PC app to encrypt to the vault.