

# **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 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://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.

---

## **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 security level (from 128 bit AES to 256 bit AES).
    - ii. Change the location of the vault.
    - iii. Set the default login method (Bluetooth or no Bluetooth).
    - iv. 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, 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.
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 viewable from outside of the app (encrypt the name).
-

# Design

---

## Bluetooth:

---

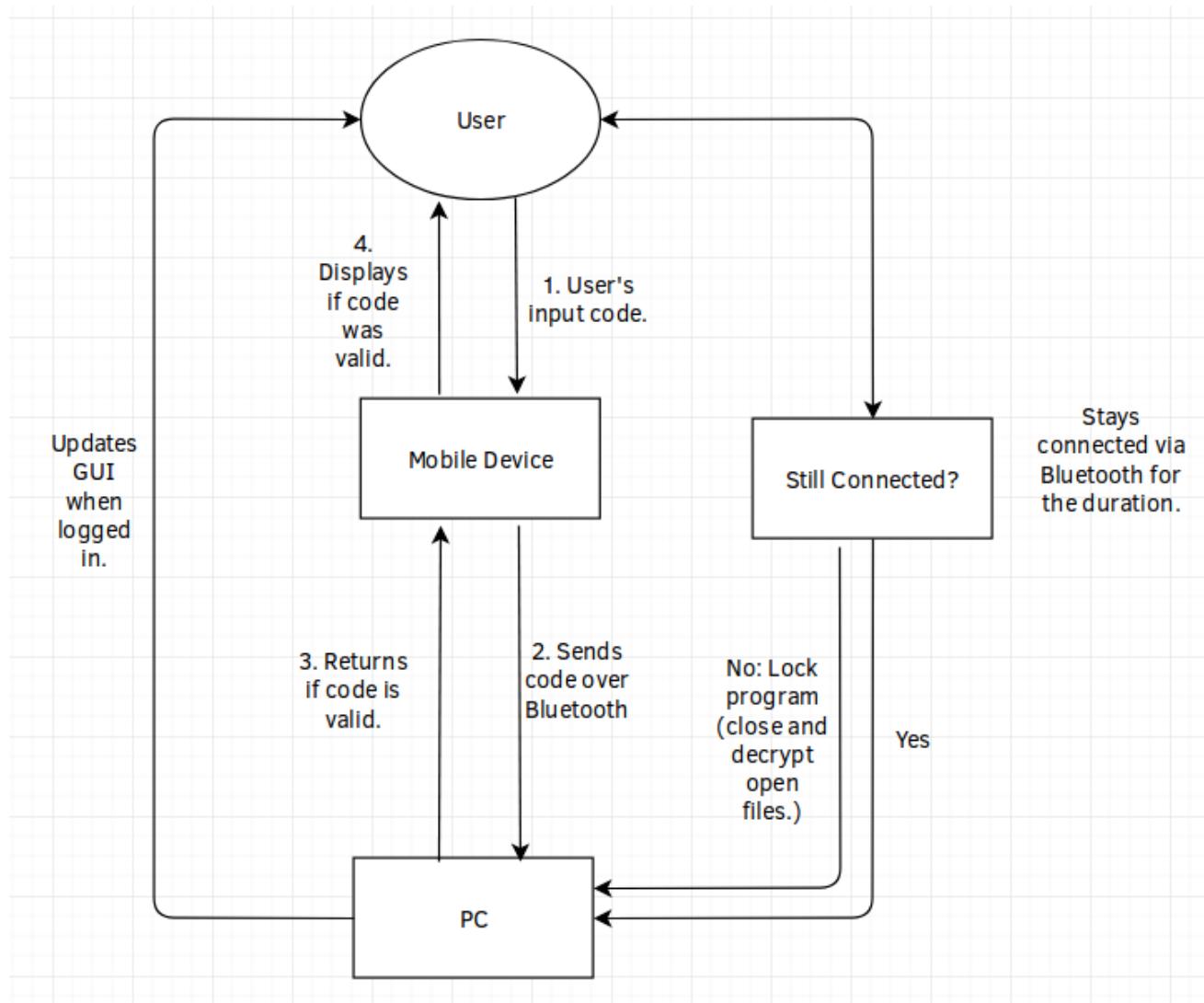
For the file store to be unlocked, I need to send the passcode to the computer via a Bluetooth connection.

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>~
```

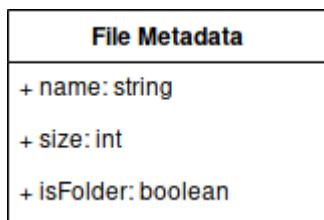
For items such as file metadata, I will use Python pickle to send an object (more of a structure) containing the metadata, rather than using separators, as then it is much easier for me to add information I want to send.

## 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:

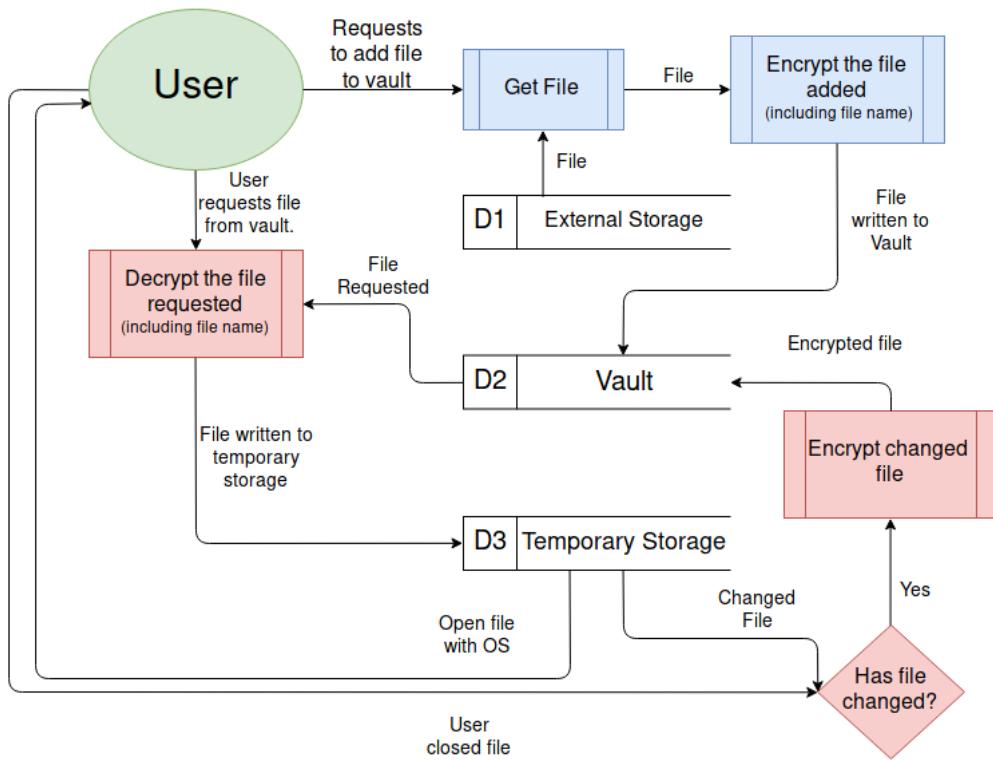
For storing the files, I will store the encrypted files in a directory set by the user. The directory will be managed using a tree structure, where the root folder contains folders for each file, with the name of every folder and file being encrypted, as otherwise anyone can see the name of your file.

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 might 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 key was incorrect. 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.

## 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 bytearray(data)
11
12 def test(times, size):
13     data = generate(times, size)   # Generate the data

```

```

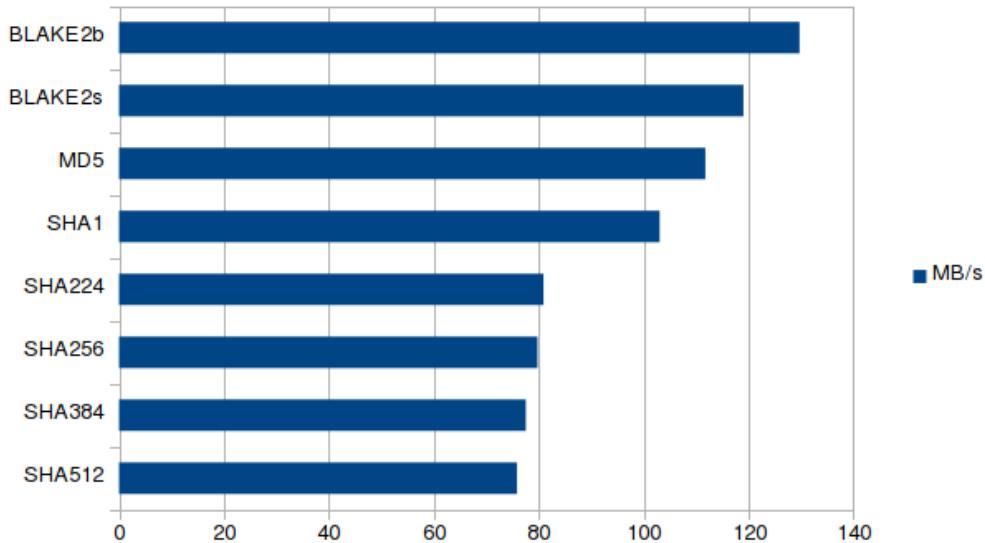
14     start = time()           # Get the start time
15     for i in range(times):
16         hashlib.sha256(data[i*size:(i+1)*size]).hexdigest() # Do the hash (in this case SHA256)
17
18     return (times*size)/(time()-start)      # Return the bytes per second.
19
20 print(test(1000, 128)) # Run the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



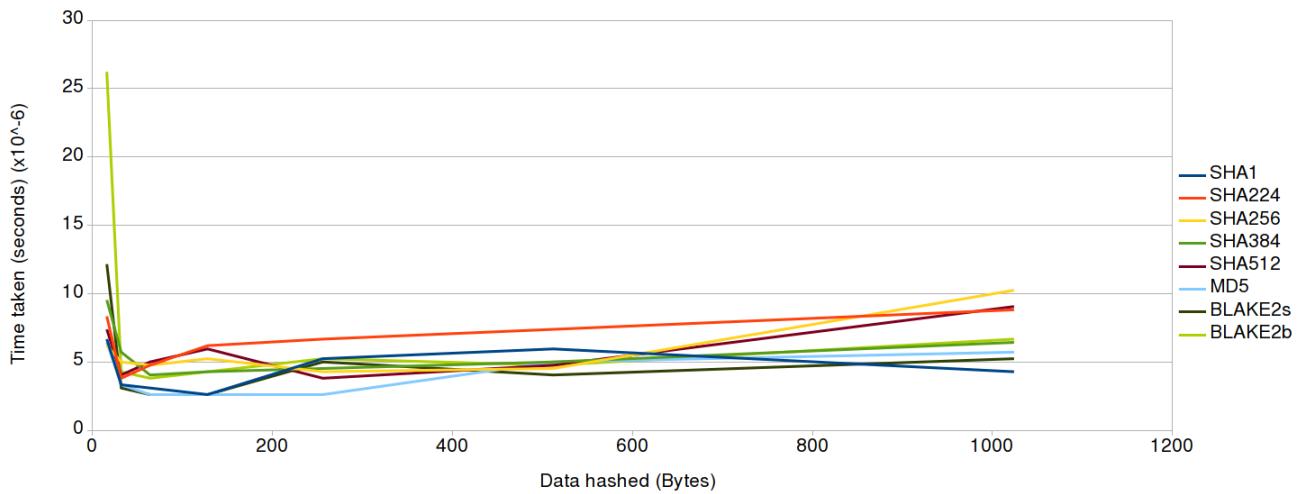
For my next tests, I will do data hashed against time. For this I will be using different sized files that I will make using this function:

```

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()

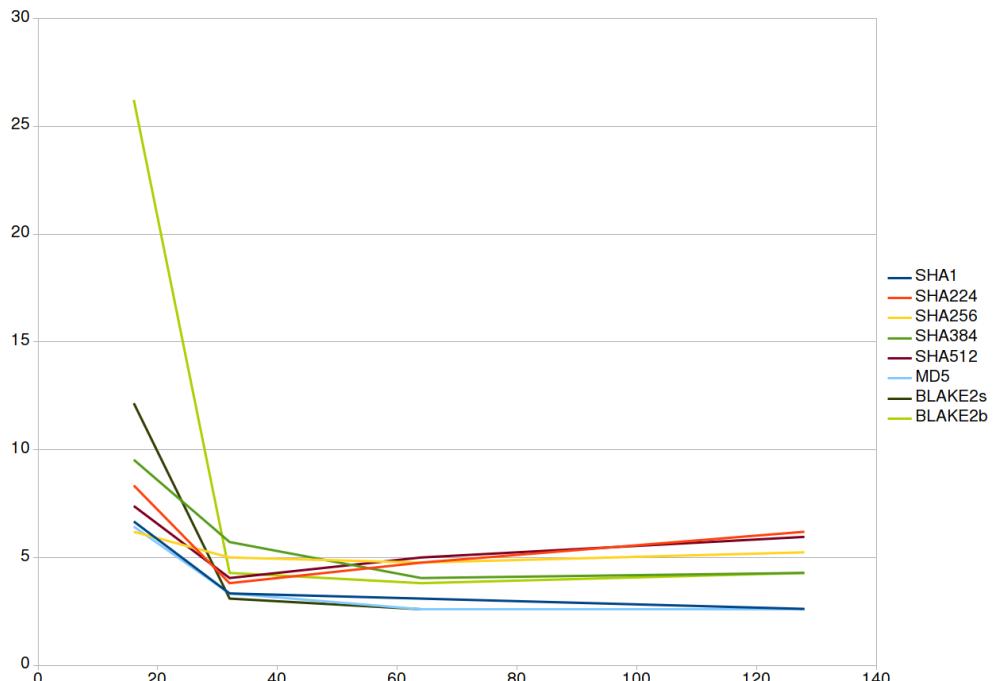
```

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



This image can be found larger in the **Large Images** section as **Figure 3**.

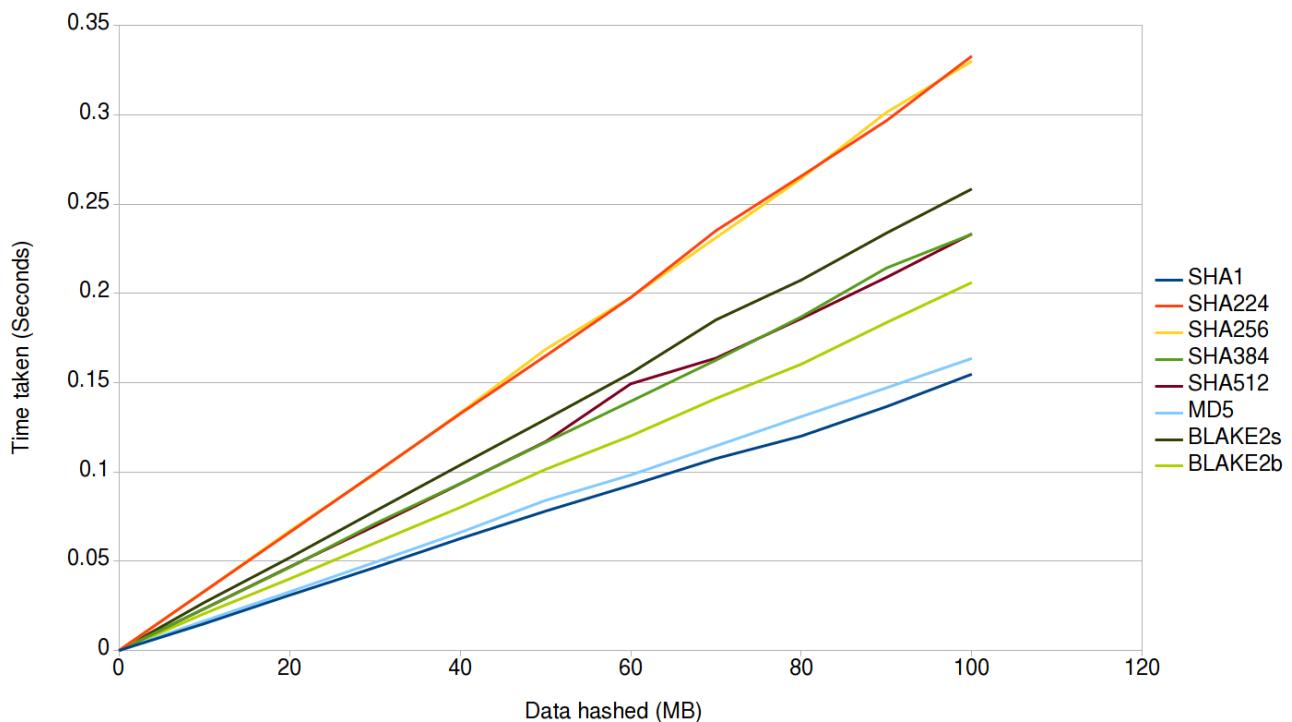
Here is the start of the graph, as that is the most interesting bit:



The axis on this graph are the same as the one before it.

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 increase) 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.

## 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.

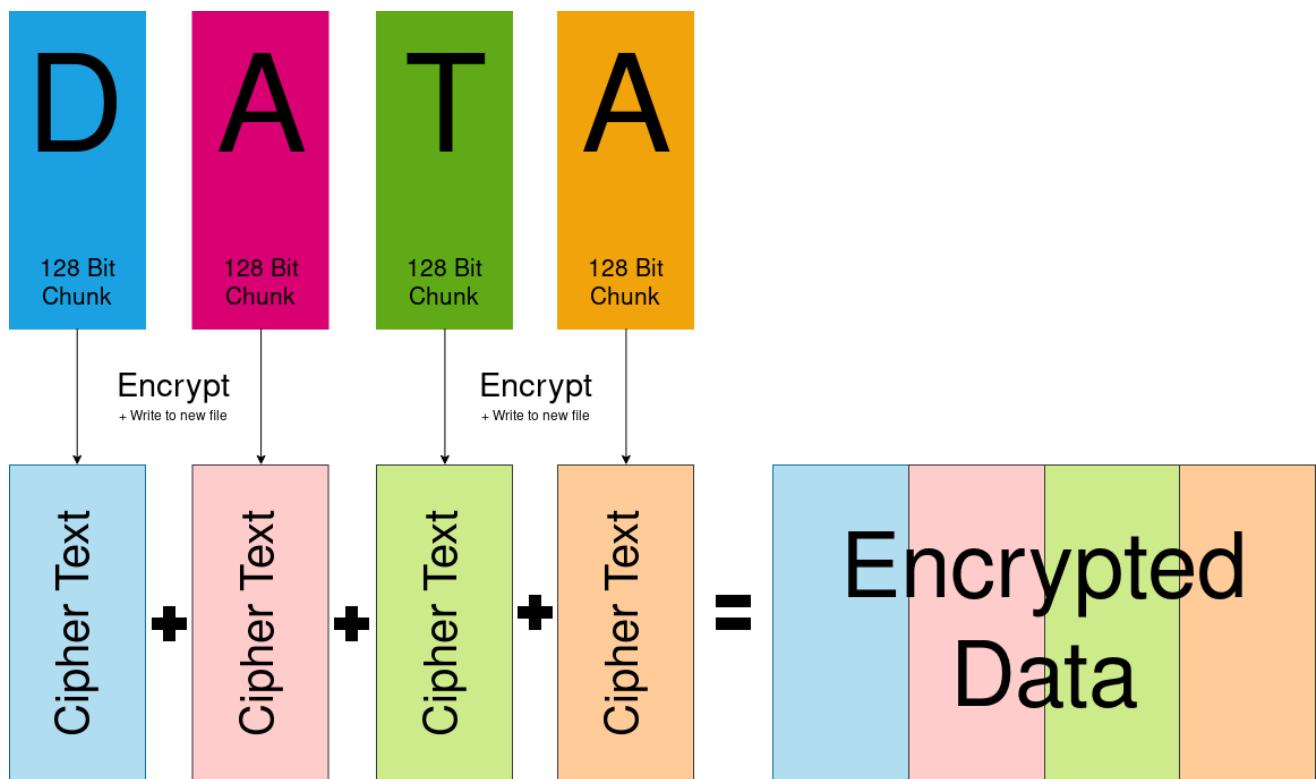
## **The Algorithm (128 bit AES):**

### **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]

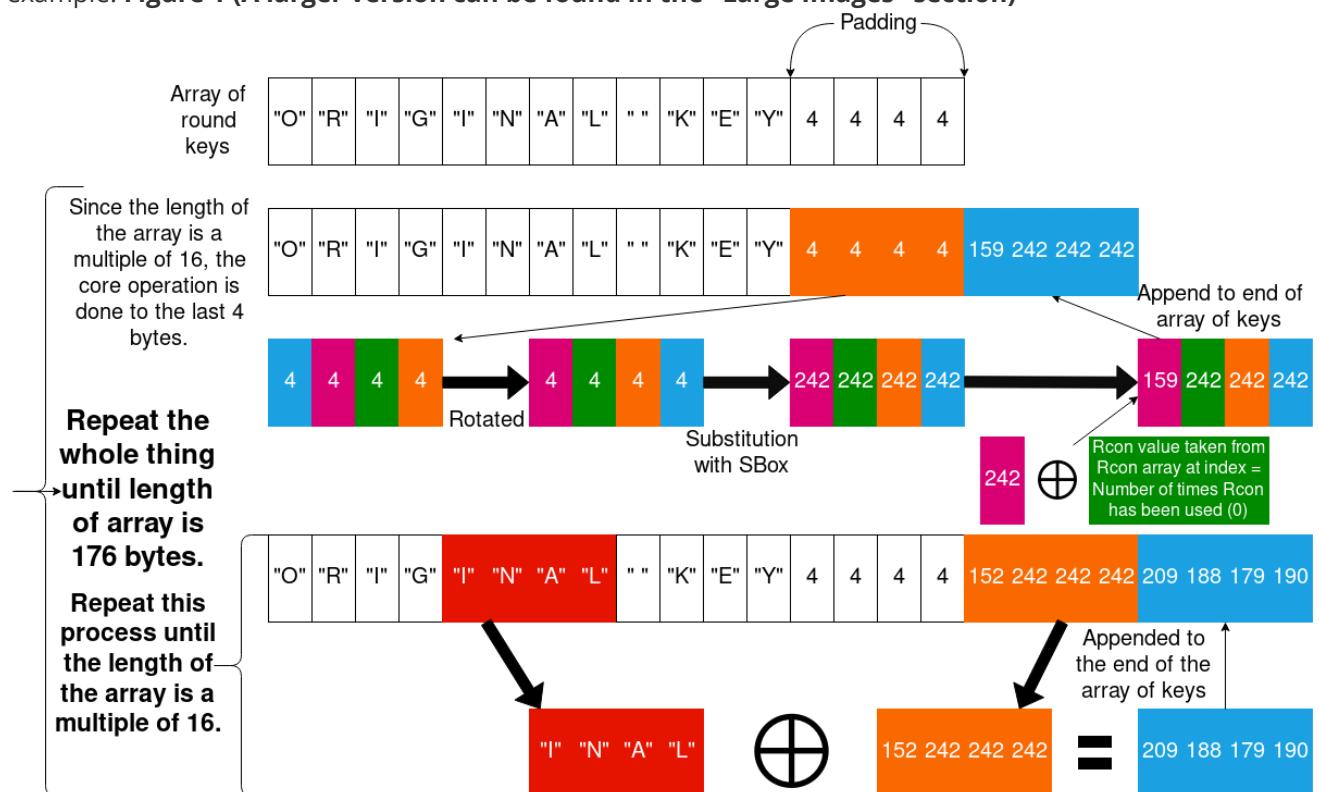
This block is not 16 bytes in length. To pad this block, we need to add 3 lots of the number 3 to the end (since  $16 - \text{length of the block} = 3$ ). The new block would look like this:

[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[i]
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](https://en.wikipedia.org/wiki/Rijndael_S-box)).
3. First byte of the 4 is XOR-ed with its 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.

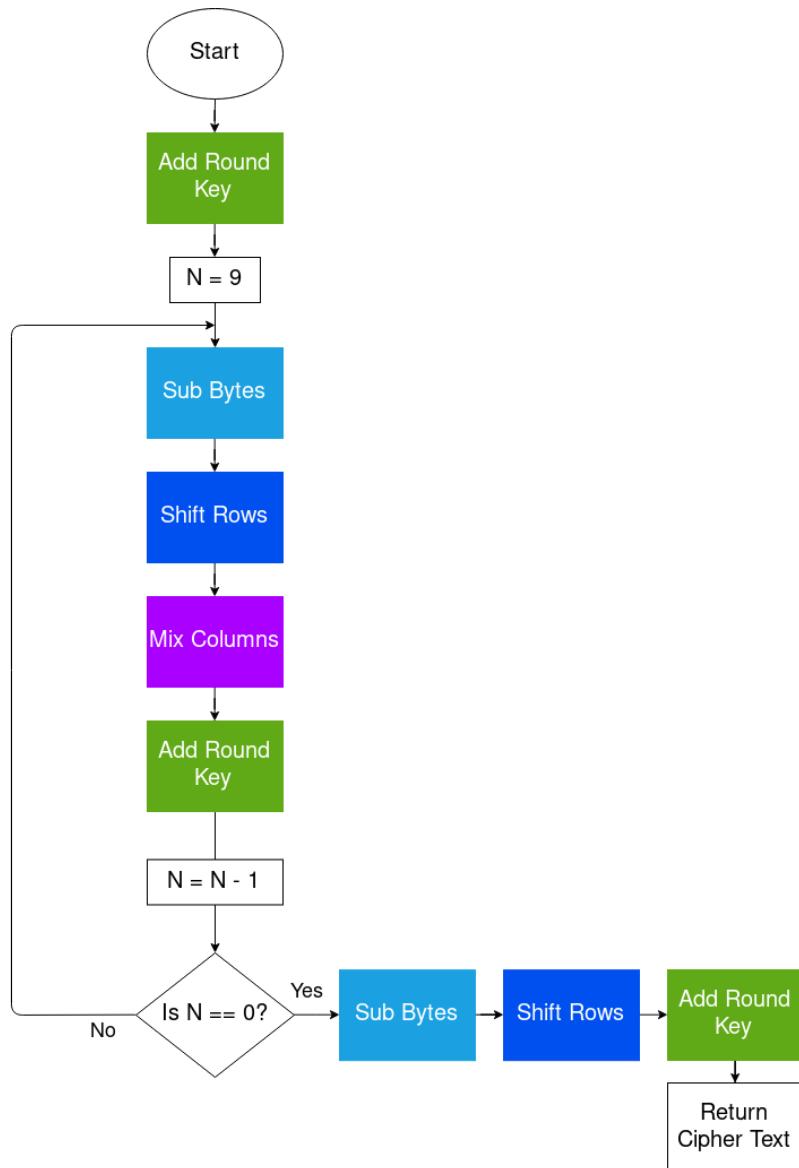
If the length of the round key array is not a multiple of 16, then the last 4 bytes in the array are XOR-ed with 4 bytes of the array that are 16 bytes before hand (shown in **Figure 1**).

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 `0x1A`, it would be substituted by the value: `0xA2`, as it is row "1", column "A".

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

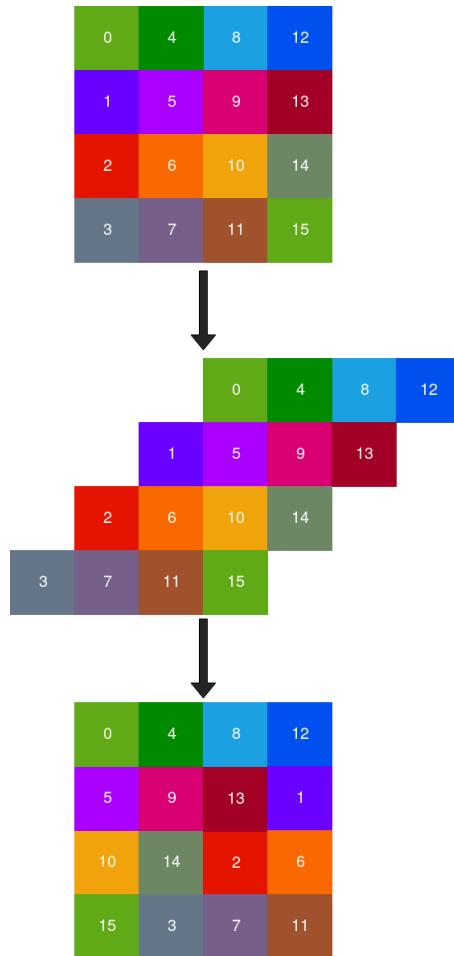
```
1 | function subBytes(state)
2 |   for i := 0 to 16
3 |     state[i] = sBox[state[i]]
4 |   return state
```

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](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:

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

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

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

And, thank god, that is the correct answer for the test vector on this page: [https://en.wikipedia.org/wiki/Rijndael\\_MixColumns](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

```

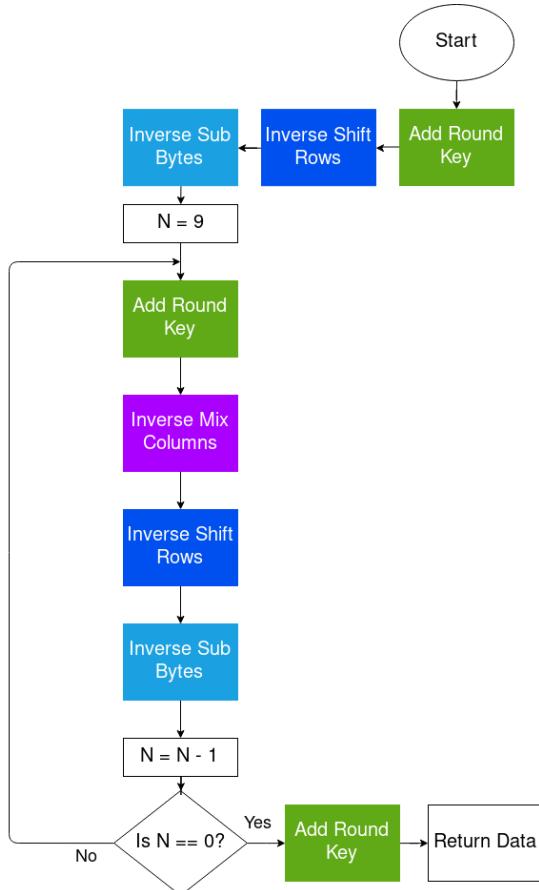
```

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 its own inverse, as XOR is the same forwards as it is backwards.

### **Inverse Sub Bytes:**

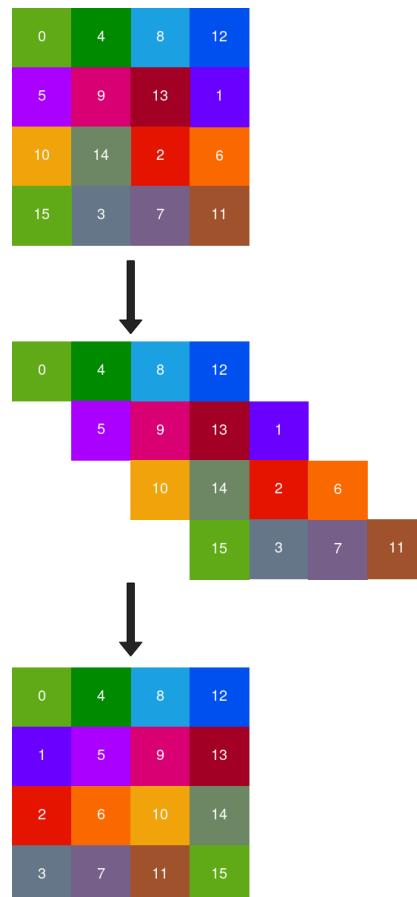
Inverse sub bytes is the same as sub bytes, it just has an inverse of the S-Box.

		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 | 4e4d20e9fc77e913bf56cc69a2b4685d761a9e44d833198612e80a72dc563f1
```

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).

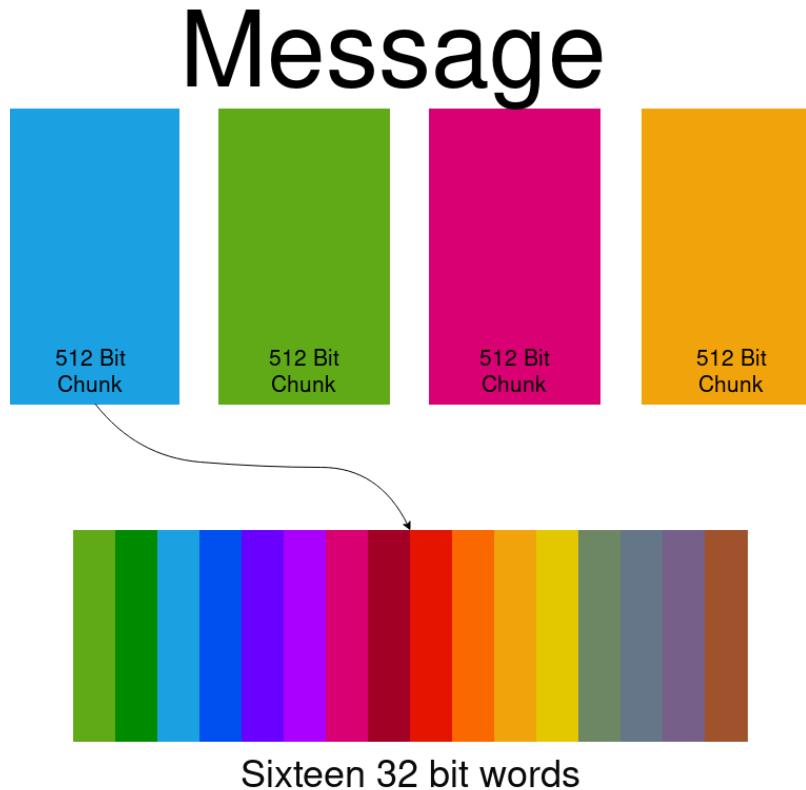
What I am doing instead, is taking the output of SHA256, splitting it in half, and XORing each half with each other to get a 128 bit output. This doesn't affect how secure it is, as you still have the extra step of XOR, making it still more secure than SHA-1.

## 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.



SHA 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:

$$l + 1 + k \equiv 448 \bmod 512$$

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

```

1 | k = 0
2 | while ((l+l+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:  $M = \text{"i don't know"}$ .

$$\begin{aligned} l &= 12 \times 8 = 96 \\ \text{Append a "1":} \\ M &= b\text{"i don't know"} + 1 \\ 448 - (96 + 1) &= 351 \text{ Zero Bits} \\ M &= b\text{"i don't know"} + 1 + 351(0s) \\ l &= 96 = 01100000 \\ \text{Final Padded Message:} \\ M &= b\text{"i don't know"} + 1 + 351(0s) + 56(0s) + 01100000 \end{aligned}$$

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

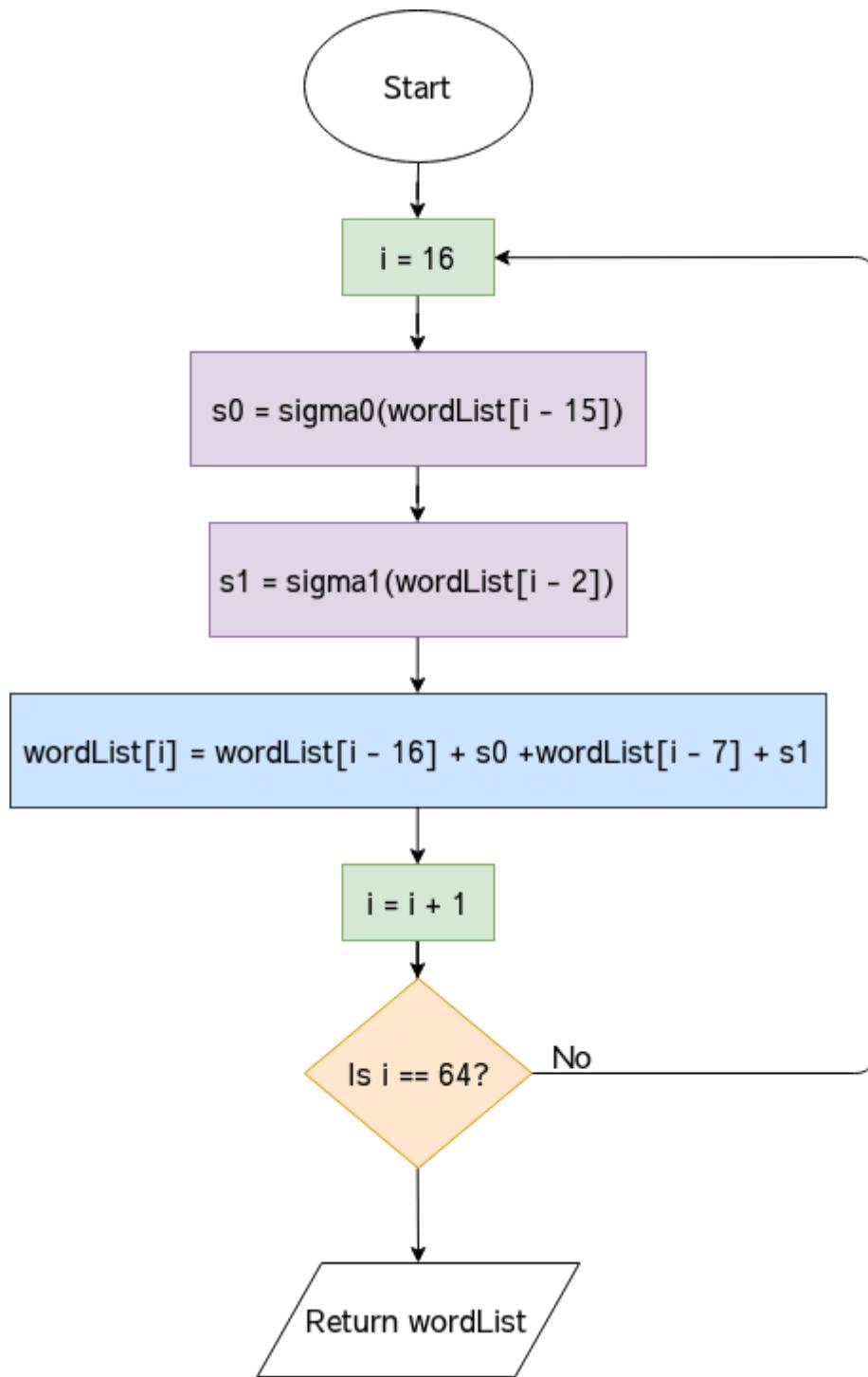
Then, we also need to **set the initial hash values** for each word in the current block. The initial hash values set by the creators of SHA:

"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. "

$$\begin{aligned} H_0 &= 6a09e667 \\ H_1 &= bb67ae85 \\ H_2 &= 3c6ef372 \\ H_3 &= a54ff53a \\ H_4 &= 510e527f \\ H_5 &= 9b05688c \\ H_6 &= 1f83d9ab \\ H_7 &= 5be0cd19 \end{aligned}$$

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  $\sigma_0$  and sigma 1  $\sigma_1$ .

#### **Sigma 0 (Expansion) ( $\sigma_0$ ):**

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.

$\oplus$  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:

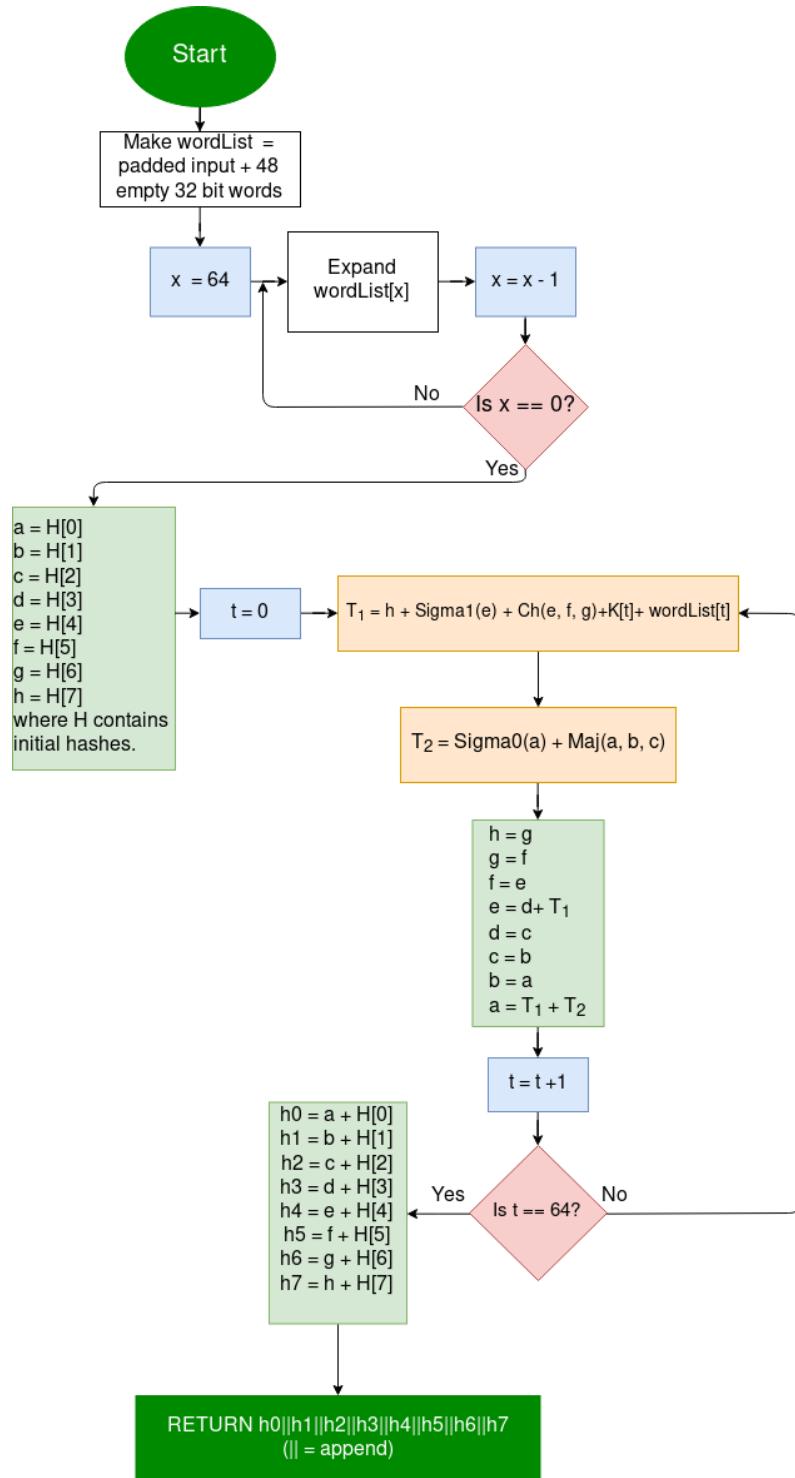
$$\sigma_0(x) = (x >>> 17) \oplus (x >>> 19) \oplus (x >> 10)$$

### 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  $h0||h1||h2||\dots$  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:

$$A = 1001011101101111000110111011101 \\ \Sigma_0 = (1001011101101111000110111011101 >>> 2) \oplus (1001011101101111000110111011101 >>> 13) \oplus \\ \dots (1001011101101111000110111011101 >>> 22) \\ (1001011101101111000110111011101 >>> 2) = 0110010111011011110001101110111 \\ \text{The two bits at the end have been moved to the front one by one.} \\ (1001011101101111000110111011101 >>> 13) = 11110001101110111011001011101101 \\ (1001011101101111000110111011101 >>> 22) = 01110111011001011101101111100011 \\ 0110010111011011110001101110111 \oplus 11110001101110111011001011101101 \oplus 01110111011001011101101111100011 \\ = 1110001100000101100010100111001$$

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:

$$\Sigma_0(x) = (x >>> 6) \oplus (x >>> 11) \oplus (x >>> 25)$$

## 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:

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

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

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

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

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

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

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

### 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.

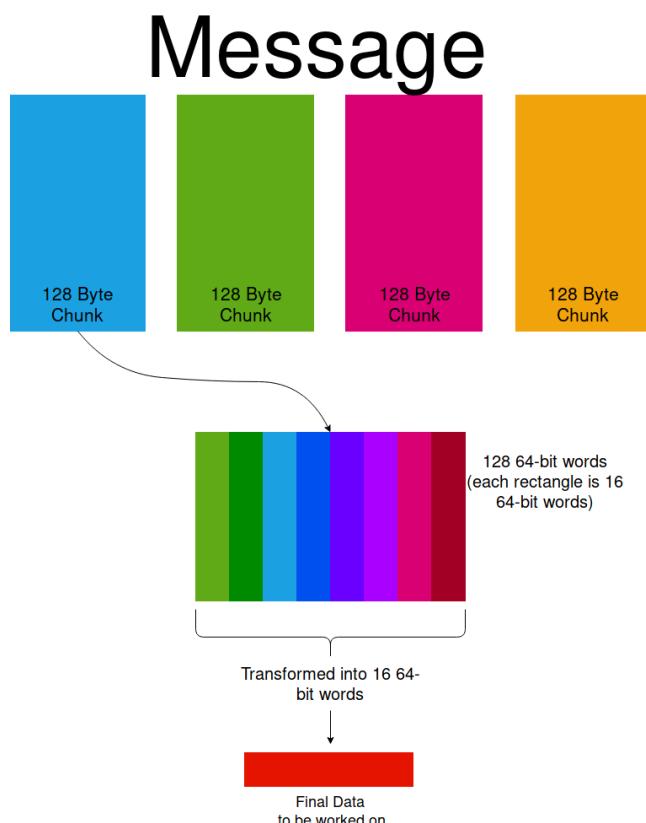
## The Algorithm:

### 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:

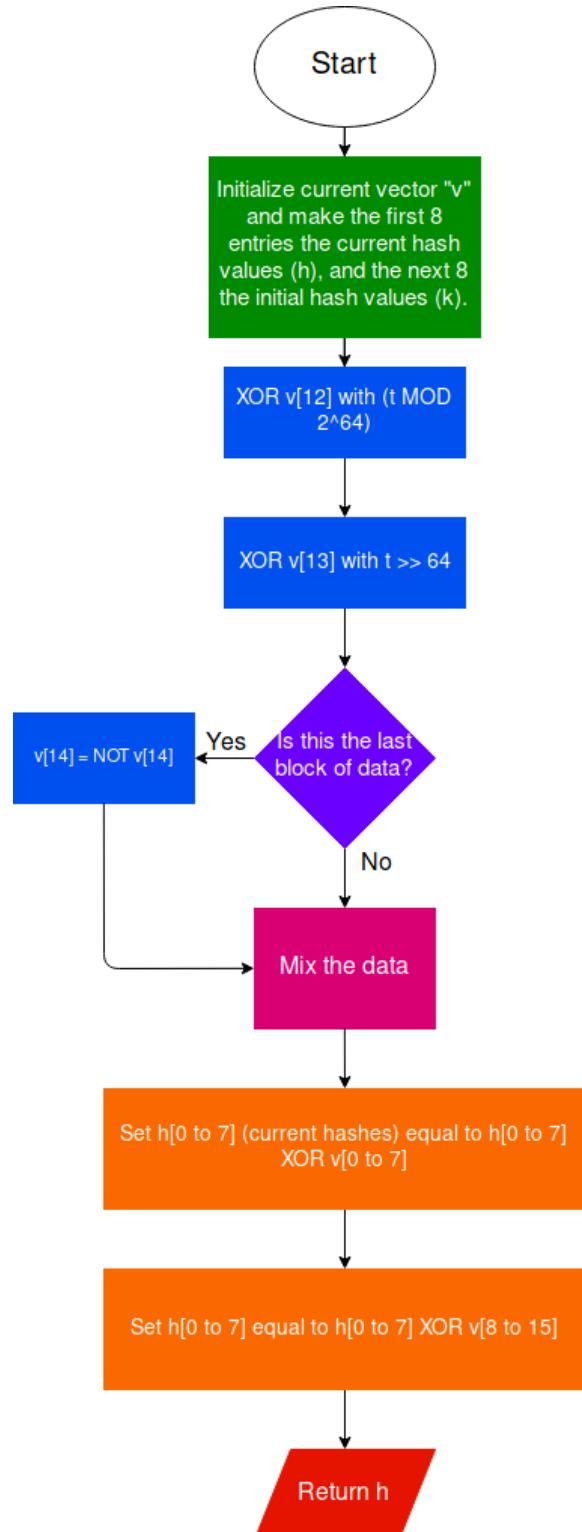


To transform a list of 16 64-bit words into 1 64-bit word, you do this algorithm (where  $a$  is the list of words):

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

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 ( $\sigma$ ) 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:

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

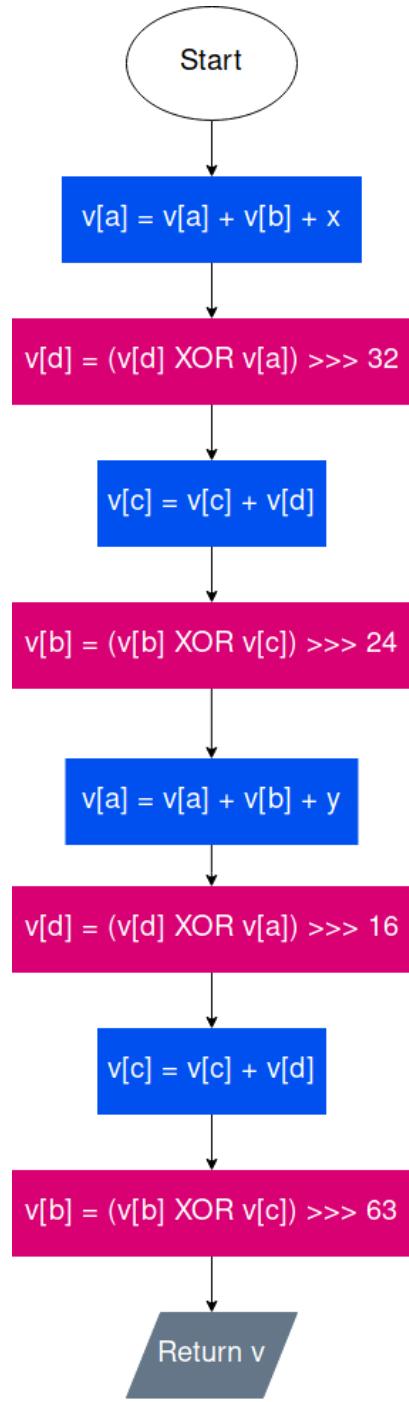
$\sigma$  is defined for BLAKE2s, and BLAKE2s only has 10 rounds, while BLAKE2b has 12, so  $\sigma_0$  and  $\sigma_1$  are repeated again to make the array 12 in length.

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.

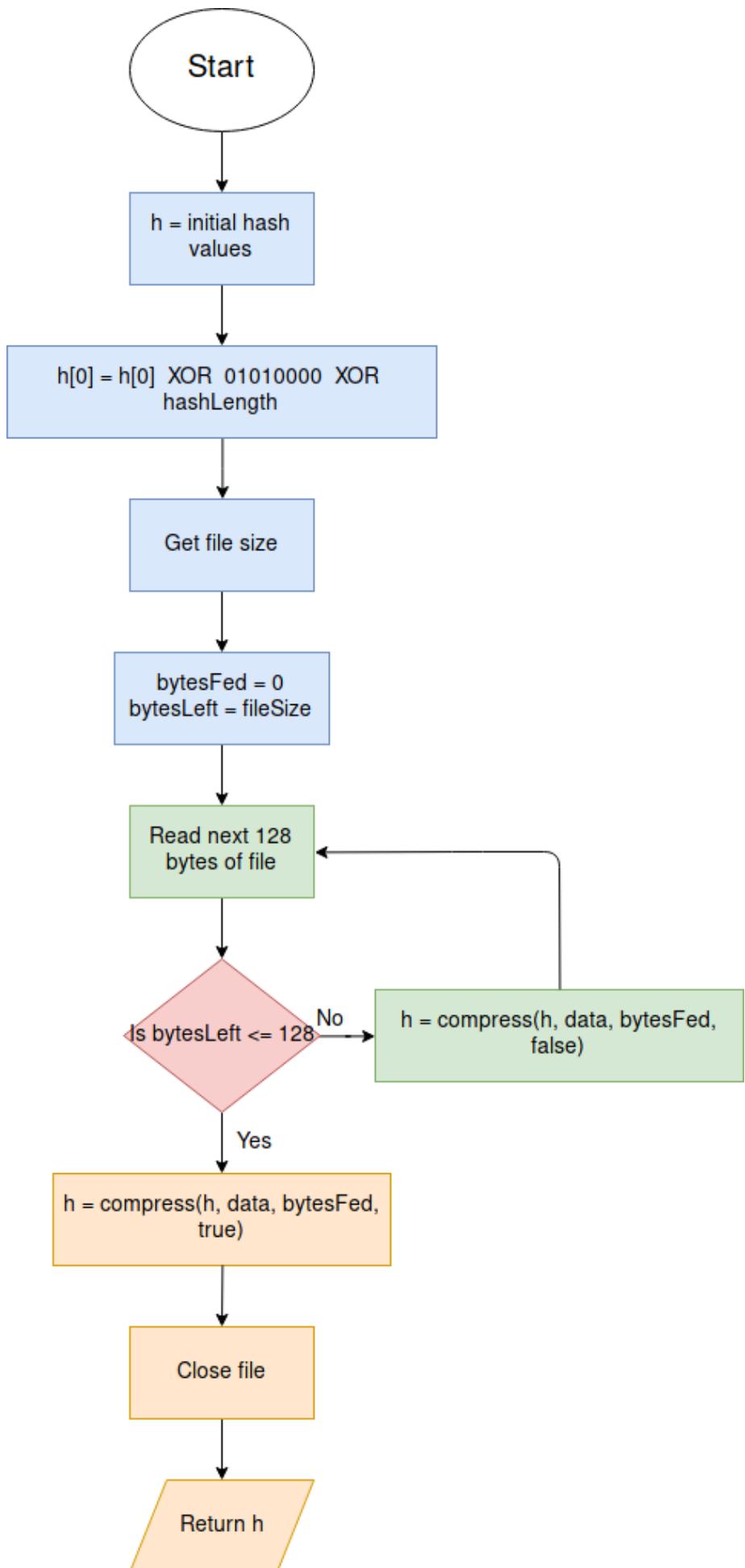
The main mixing function takes the inputs:

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

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  $nn$  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, usually resulting in putting more strain on the hardware, and also slows the overall process down because getting items from memory takes a fair amount of time. Here is a good video comparing merge sort and quick sort (along with a few other algorithms): <https://youtu.be/ZZuD6iUe3Pc>

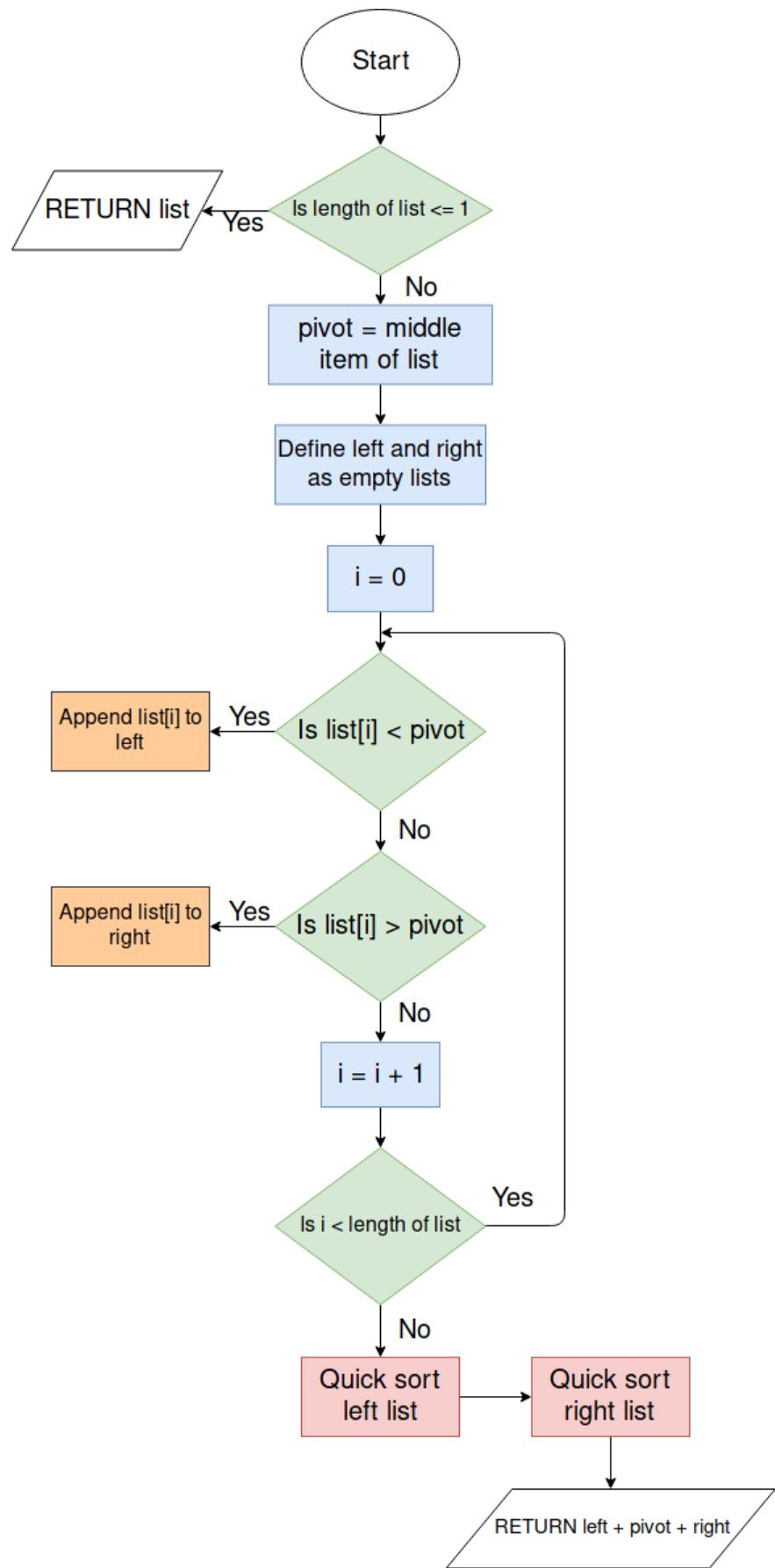
The algorithm goes like this (using a list of items to be sorted):

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

Here is a flow diagram to represent this:

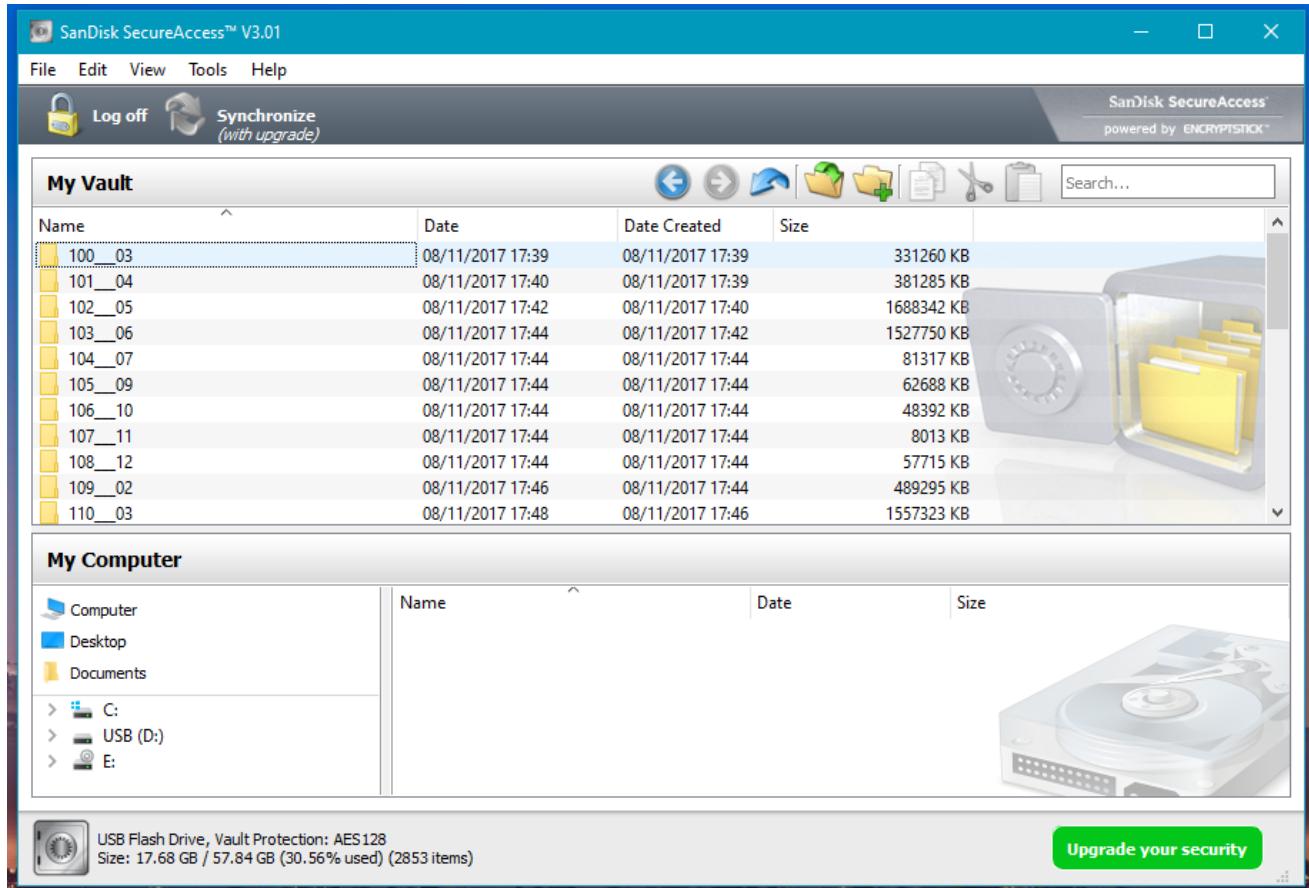


# UI Research:

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):

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.

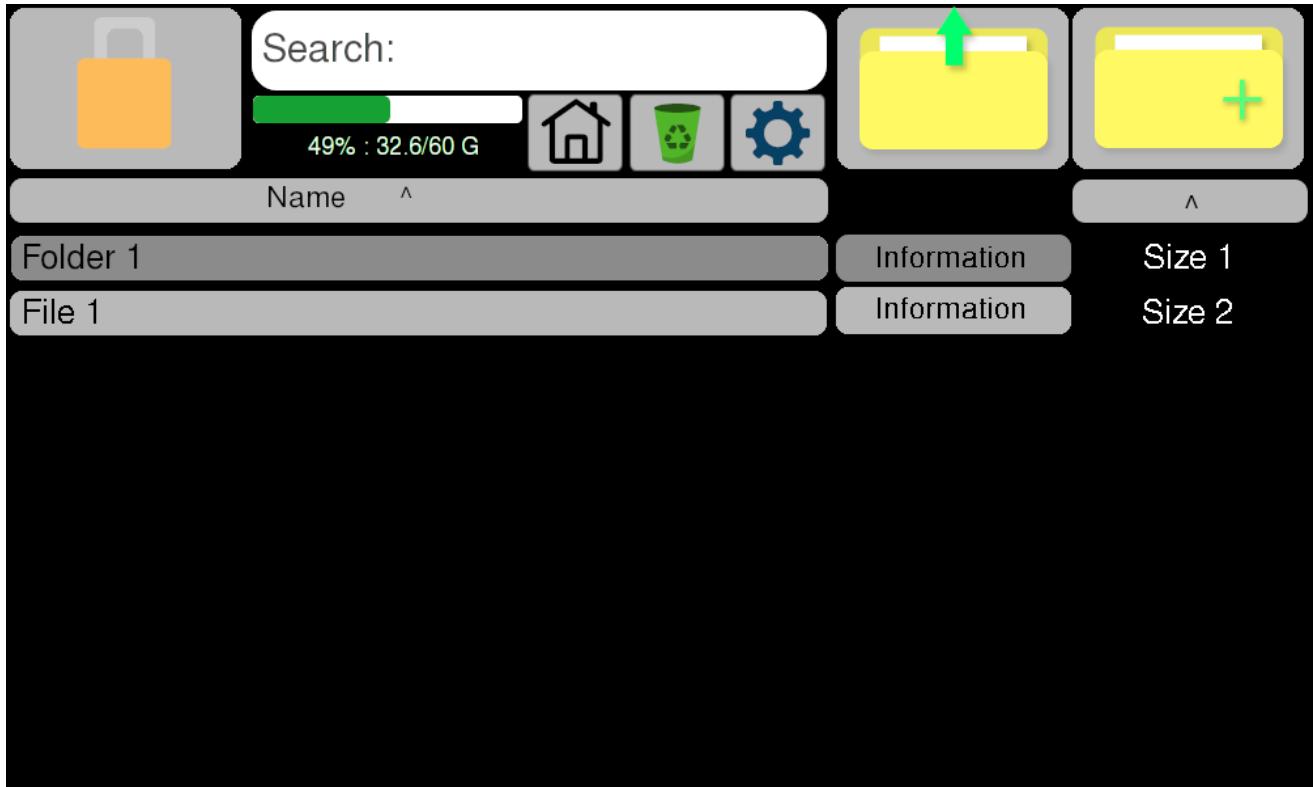
### SanDisk did these things right:

- 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 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.

Taking all of these points 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.

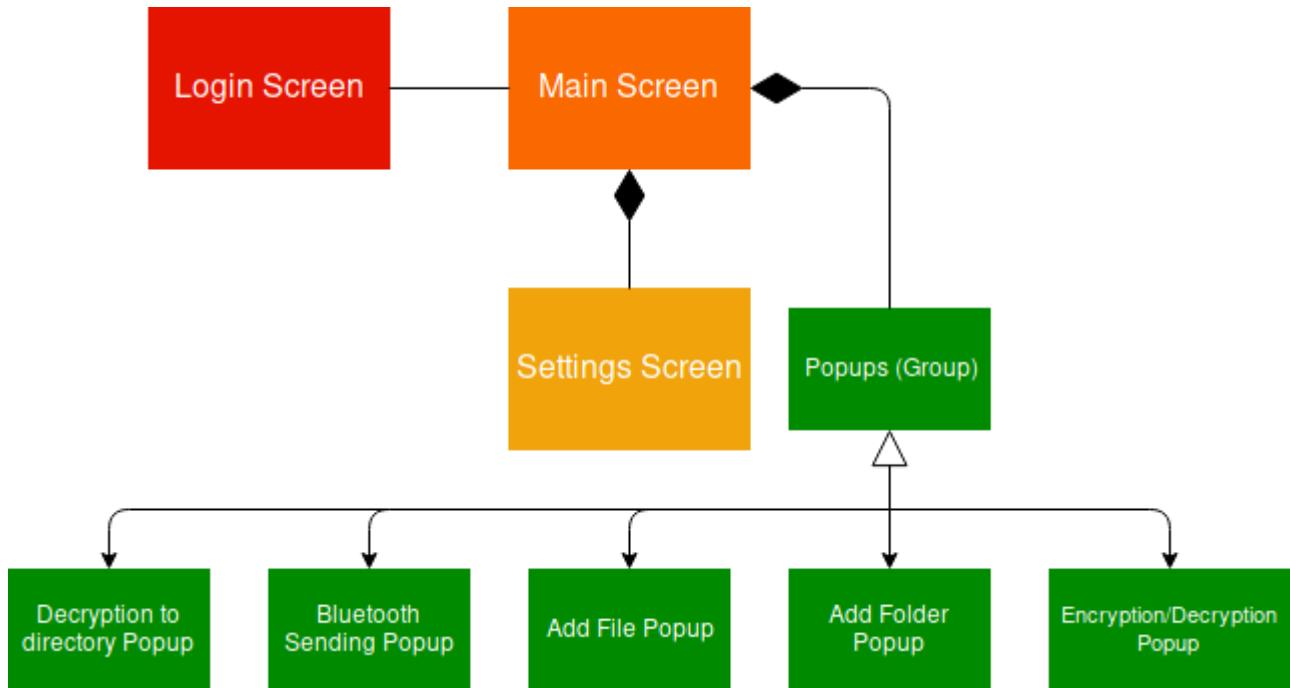
The login screen will have 2 modes:

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.

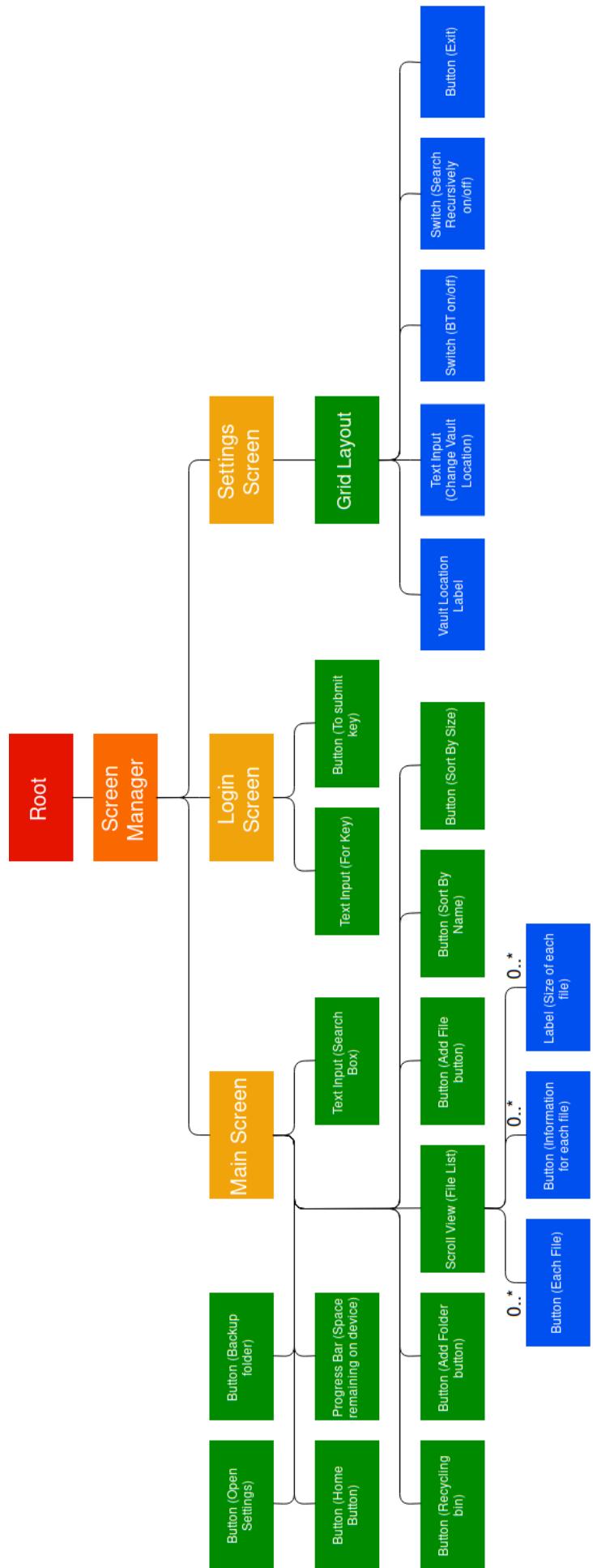
The Encryption/Decryption popup should be opened when the user encrypts/decrypts a file, and should display information including how fast the file is being enc/decrypted (in kb or mb per second), the percentage of the file that has been enc/decrypted so far, and how many items have been done out of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

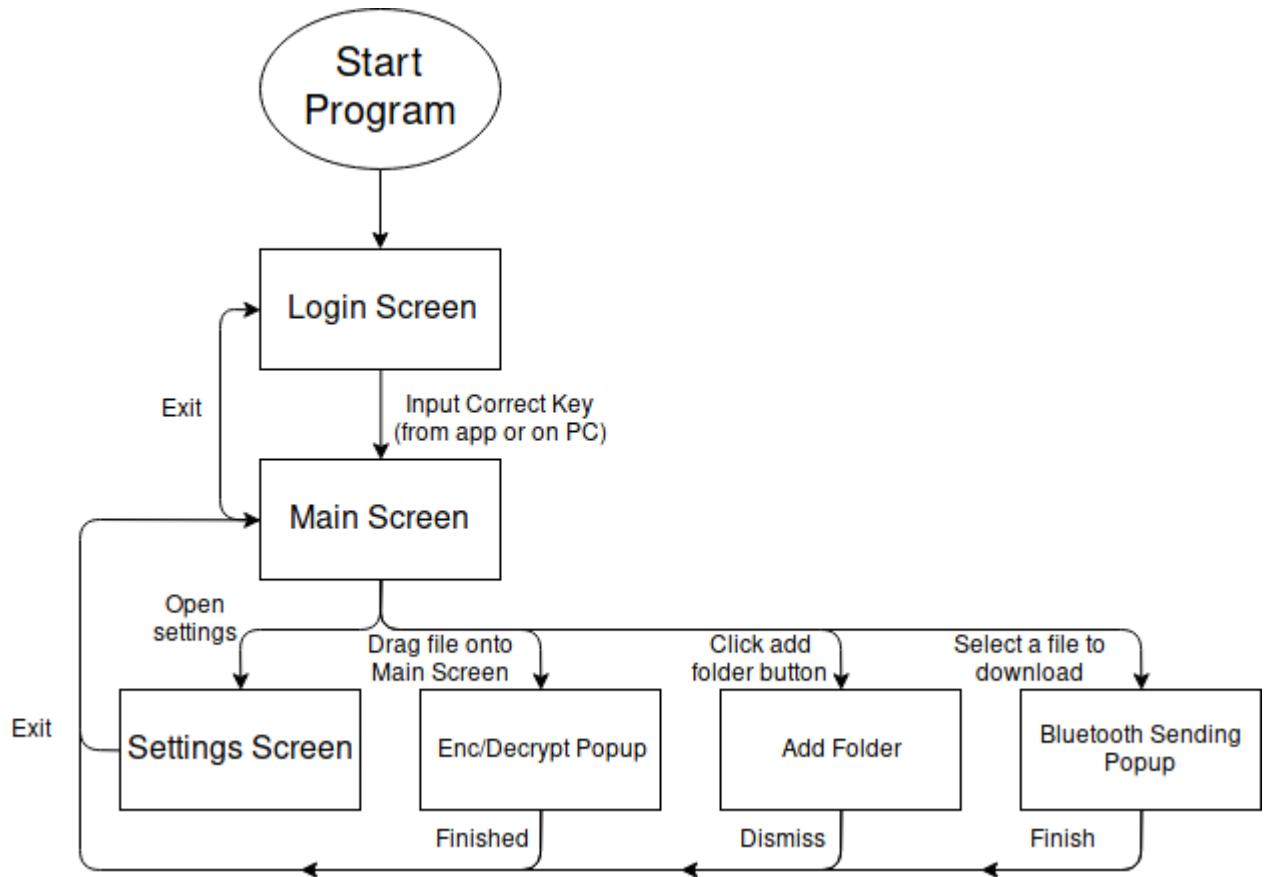
Popups are designed for one purpose only, and are usually used briefly before they are closed again. Screens will be used throughout the program, acting as the base of the GUI, where child widgets can be added to the screen, such as buttons, text inputs and views (such as scroll views). The screens will 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):



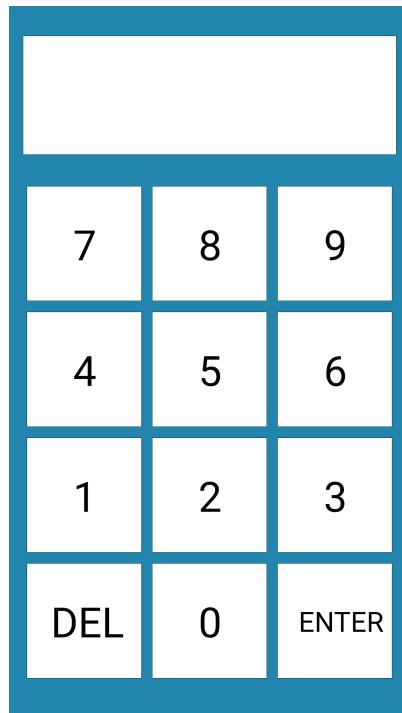
Each layer has its own colour, since I couldn't think of a better way of making this clear without making the image extremely wide. "0..\*" means 0 to many of this widget can exist at any time. This shows all of the widgets that will be on each screen at all times (unless obstructed by a popup) as default.

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):



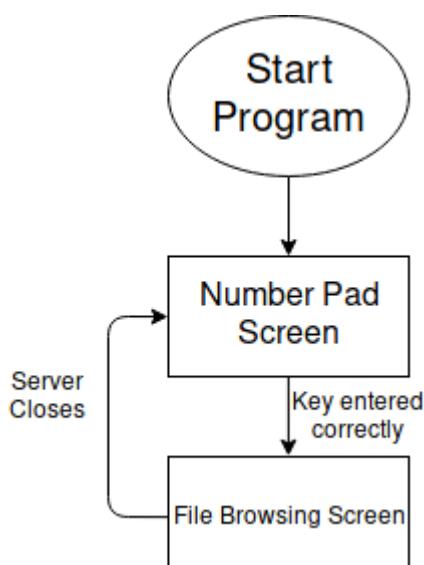
It is very minimal, as I decided to keep it as minimal as possible so that the user doesn't get confused, and to keep clutter at a minimum.

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:

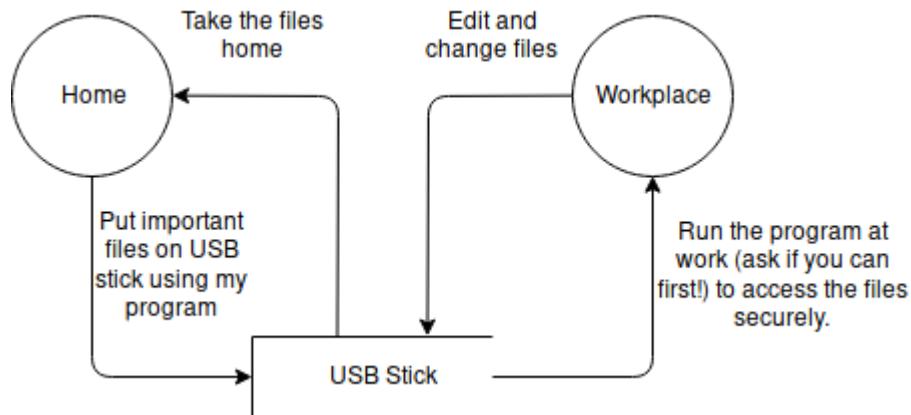
IPSO	Program Section	Item
Input	Login	Key (From user input). Vault directory path. Files in the Vault (for checking the key).
	File Browser (Main Screen)	Vault directory path. Recycling folder path (for when files are deleted). The key (for displaying encrypted file names).
	Search Bar (Main Screen)	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:

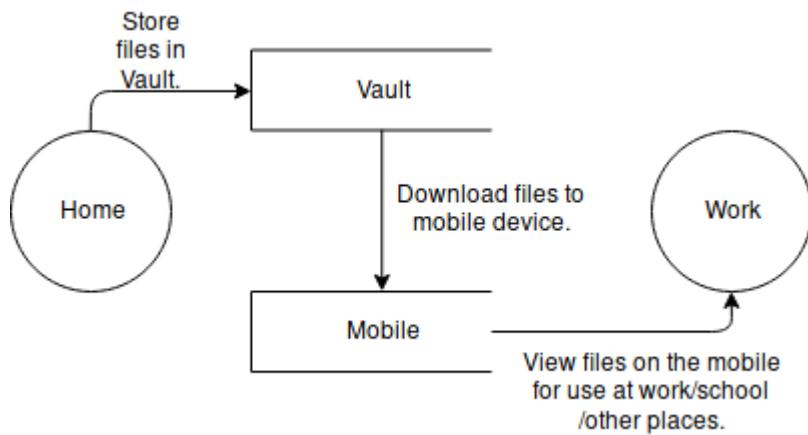


## Storing the files at home:

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.

## 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 |     |   └── mainBtns.py
59 |     |   └── mainScClass.py
60 |     |   └── mainSmallPops.py
61 |     |   └── settingsScreen.py
62 |     |   └── ui.py
63 |     └── SHA.py
64 |   └── sortsCy.c
65 |   └── sortsCy.cpython-37m-x86_64-linux-gnu.so
66 |   └── sortsCythonSource
67 |     |   └── build
68 |     |     |   └── temp.linux-x86_64-3.7
69 |     |     |       └── sortsCy.o
70 |     |   └── setup.py
71 |     |   └── sortsCy.pyx
72 |   └── 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.

`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):
30      if configLocation == None:
31          fSep = getFileSep()
32          configLocation = findConfigFile(getStartDir(fSep)[0], fSep)
```

```

34     configFile = open(configLocation, "r")
35     if lineNumberToRead == None:
36         for line in configFile:
37             lineSplit = line.split("--")
38             lineSplit[1] = lineSplit[1].replace("\n", "")
39             if lineSplit[0] == "vaultDir":
40                 path = lineSplit[1]
41             elif lineSplit[0] == "searchRecursively":
42                 if lineSplit[1] == "True":
43                     recurse = True
44                 elif lineSplit[1] == "False":
45                     recurse = False
46                 else:
47                     raise ValueError("Recursive search settings not set correctly in config file: Not True or False.")
48             elif lineSplit[0] == "bluetooth":
49                 if lineSplit[1] == "True":
50                     bt = True
51                 elif lineSplit[1] == "False":
52                     bt = False
53                 else:
54                     raise ValueError("Bluetooth not configured correctly in config file: Not True or False.")
55
56     configFile.close()
57
58     return path, recurse, bt
59
60 else:
61     lineSplit = configFile.readlines()[lineNumToRead].split("--")
62     lineSplit[1] = lineSplit[1].replace("\n", "")
63     return lineSplit[1]
64
65 def getFileSep():
66     if platform.startswith("win32"): # Find out what operating system is running.
67         return "\\"
68     else: #windows bad
69         return "/"
70
71 def getStartDir(fileSep=None):
72     if fileSep == None:
73         fileSep = getFileSep()
74     startDir = osPath.dirname(osPath.realpath(__file__))+fileSep
75     tempDir = startDir.split(fileSep)
76     for i in range(2):
77         del tempDir[-2]
78     return startDir, fileSep.join(tempDir)+fileSep+"assets"+fileSep+"exports"+fileSep
79
80
81 def editConfTerm(term, newContent, config): # Edits a given term in the config.cfg file.
82     with open(config, "r") as conf:
83         confContent = conf.readlines()
84
85     for i in range(len(confContent)):
86         a = confContent[i].split("--")
87         if term == a[0]:
88             a[1] = newContent+"\n"
89             confContent[i] = "--".join(a)
90
91     with open(config, "w") as confW:
92         confW.writelines(confContent)
93
94 def dirInputValid(inp, fileSep):
95     valid = bool((inp[0] == fileSep) and ("\\n" not in inp)) #If it starts with the file separator and
96     doesn't contain any new lines, then it is valid for now.
97     inp = inp.split(fileSep)
98     focusIsSlash = False
99     for item in inp: #Checks for multiple file separators next to each other, as that would be an
100        invalid folder name.
101            if item == "":

```

```

100         if focusIsSlash:
101             valid = False
102             focusIsSlash = True
103         else:
104             focusIsSlash = False
105     return valid
106
107 def changeVaultLoc(inp, fileSep, config):      #Sorts out the UI while the vault location is changed.
108     if inp != "":
109         if dirInputValid(inp, fileSep):
110             if osPath.exists(inp) and osPath.isdir(inp):
111                 editConfTerm("vaultDir", inp, config)
112             else:
113                 makedirs(inp)
114                 if inp[-1] != fileSep:
115                     inp += fileSep
116                 editConfTerm("vaultDir", inp, config)
117
118     return True
119
120     return False
121
122
123 def runConfigOperations():
124     fileSep = getFileSep()
125     osTemp = gettempdir() + fileSep #From tempfile module
126     # Get config settings.
127     startDir, sharedAssets = getStartDir(fileSep)
128
129     configLoc = findConfigFile(startDir, fileSep)
130     path, recurse, bt = readConfigFile(configLoc)
131     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.

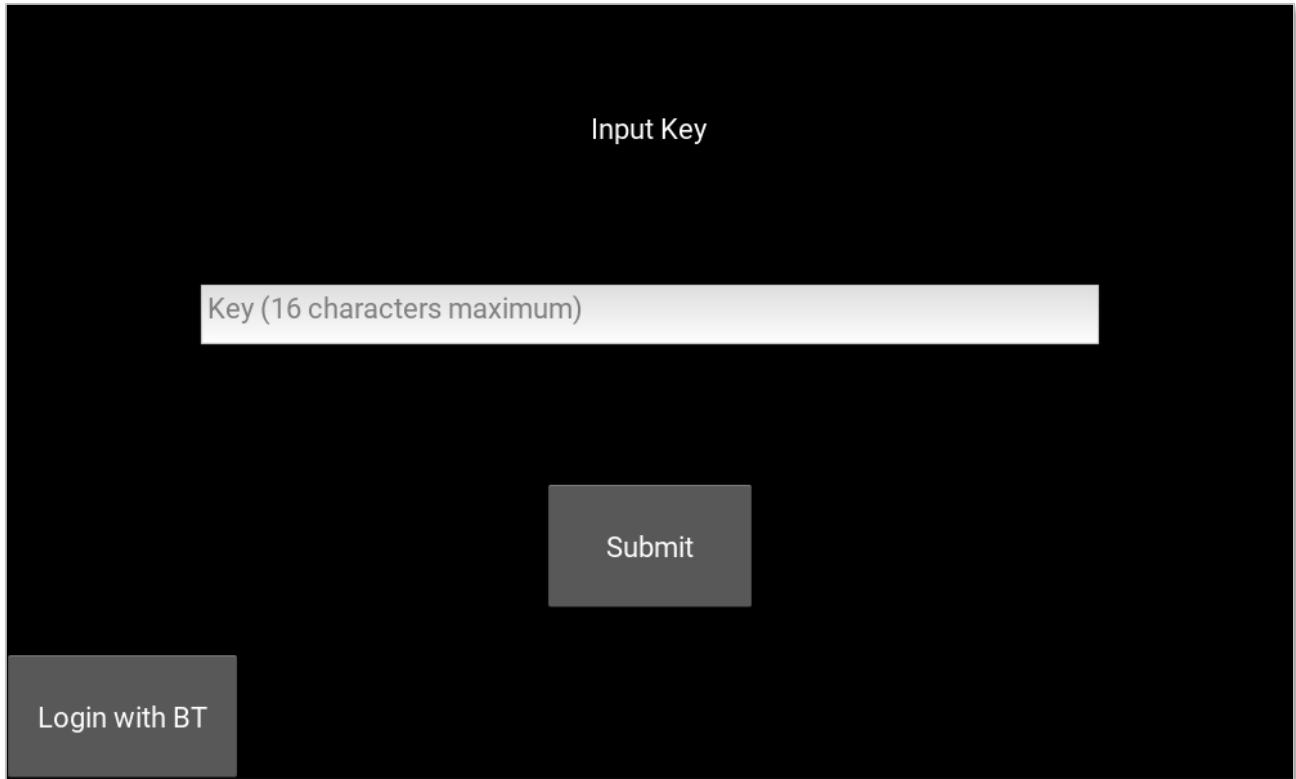
## The PC App GUI:

---

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

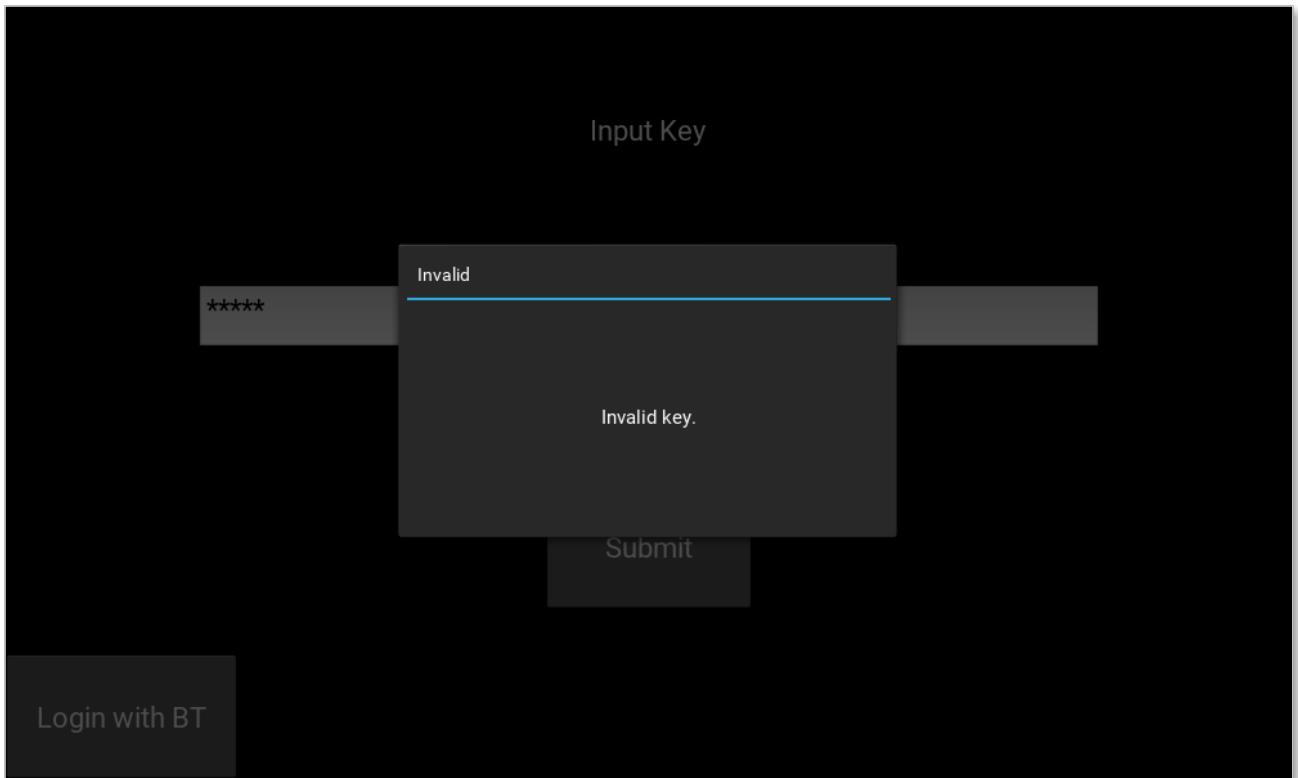
## Login Screen

Here is an image of the Login Screen:

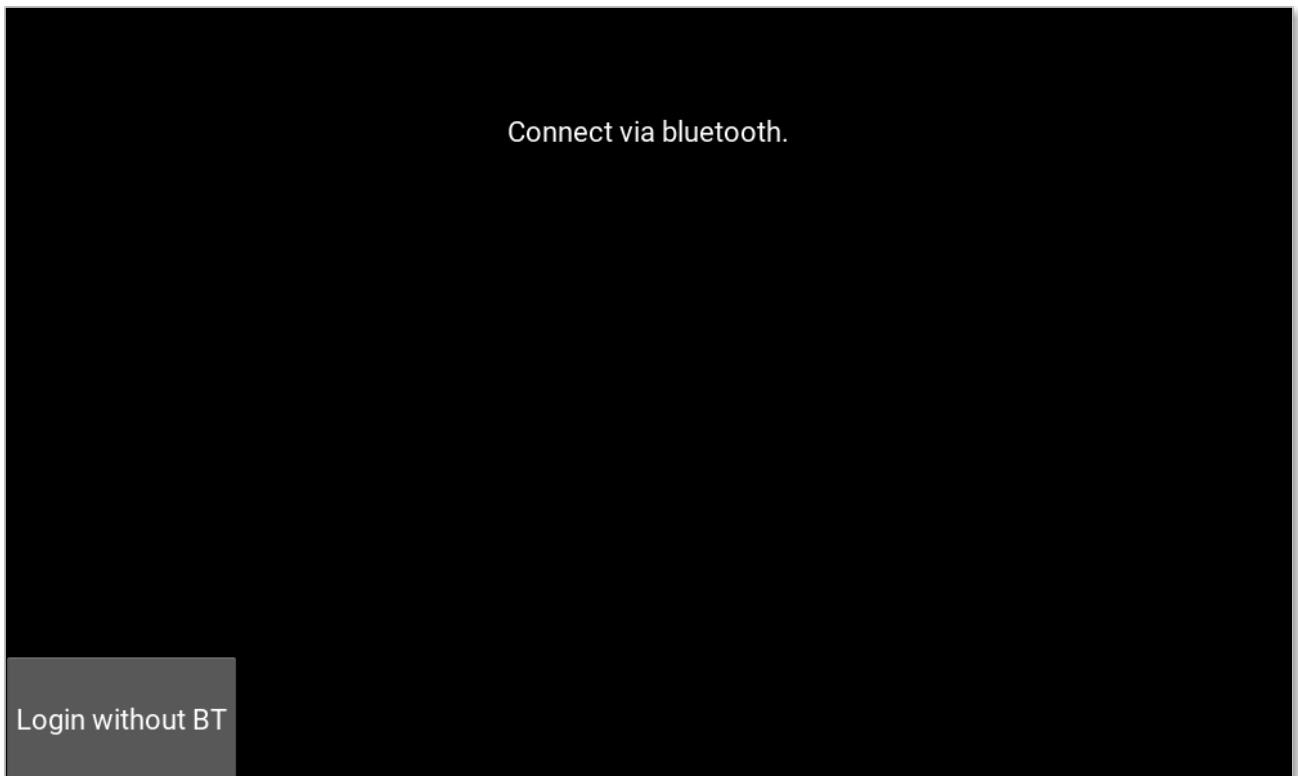


It consists of a key entry text input, a "Submit" button and a button to switch between logging in with Bluetooth and without Bluetooth.

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

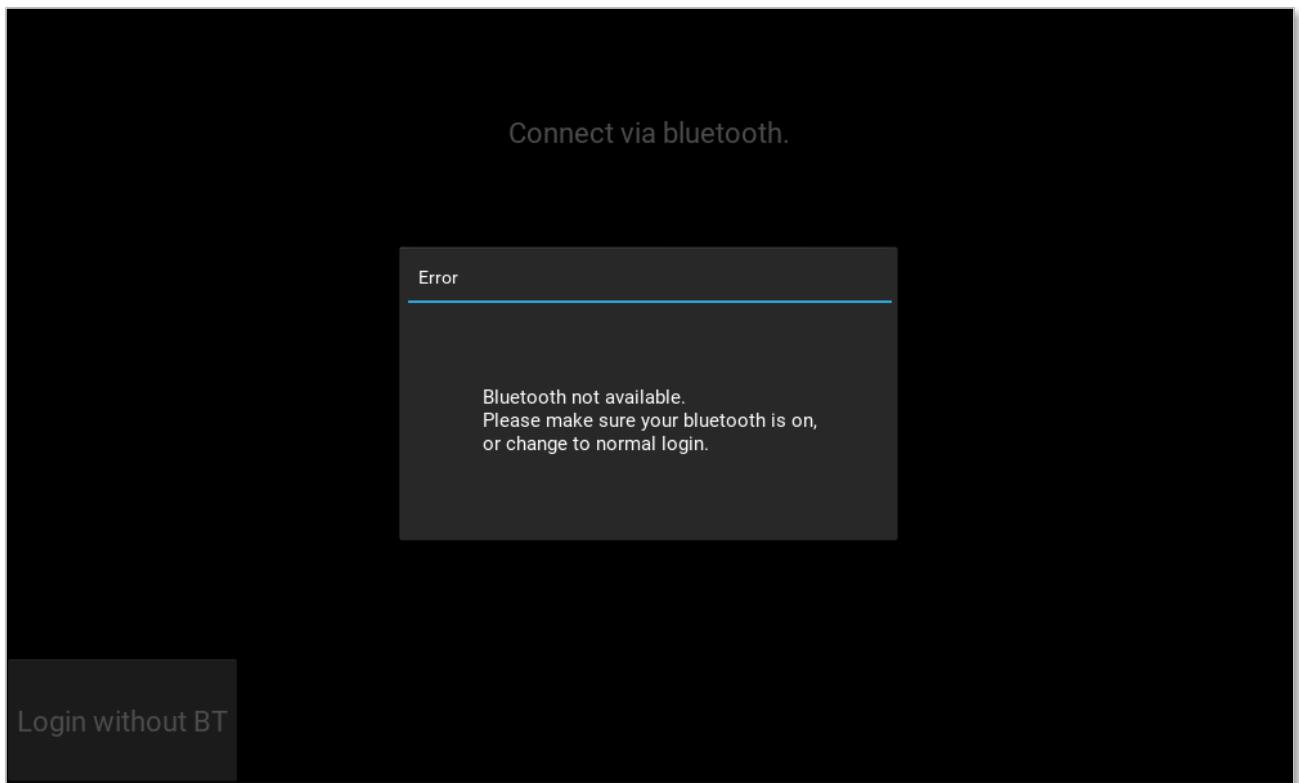


The Bluetooth login screen can be accessed by clicking the "Login with BT" button, changing the configuration file, or changing the settings once logged in. Here is an image of the Login Screen with Bluetooth:



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:

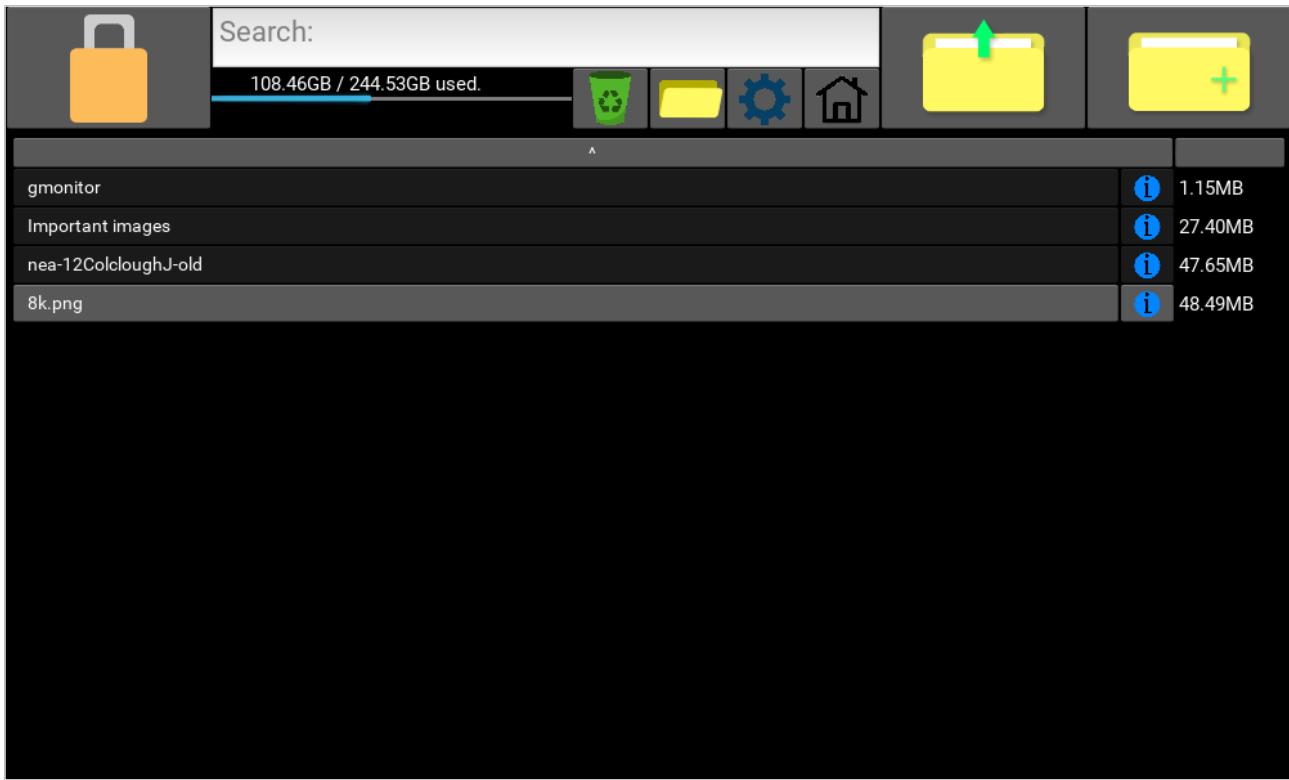


They can then click away from the popup to dismiss it, and do what they want from there.

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

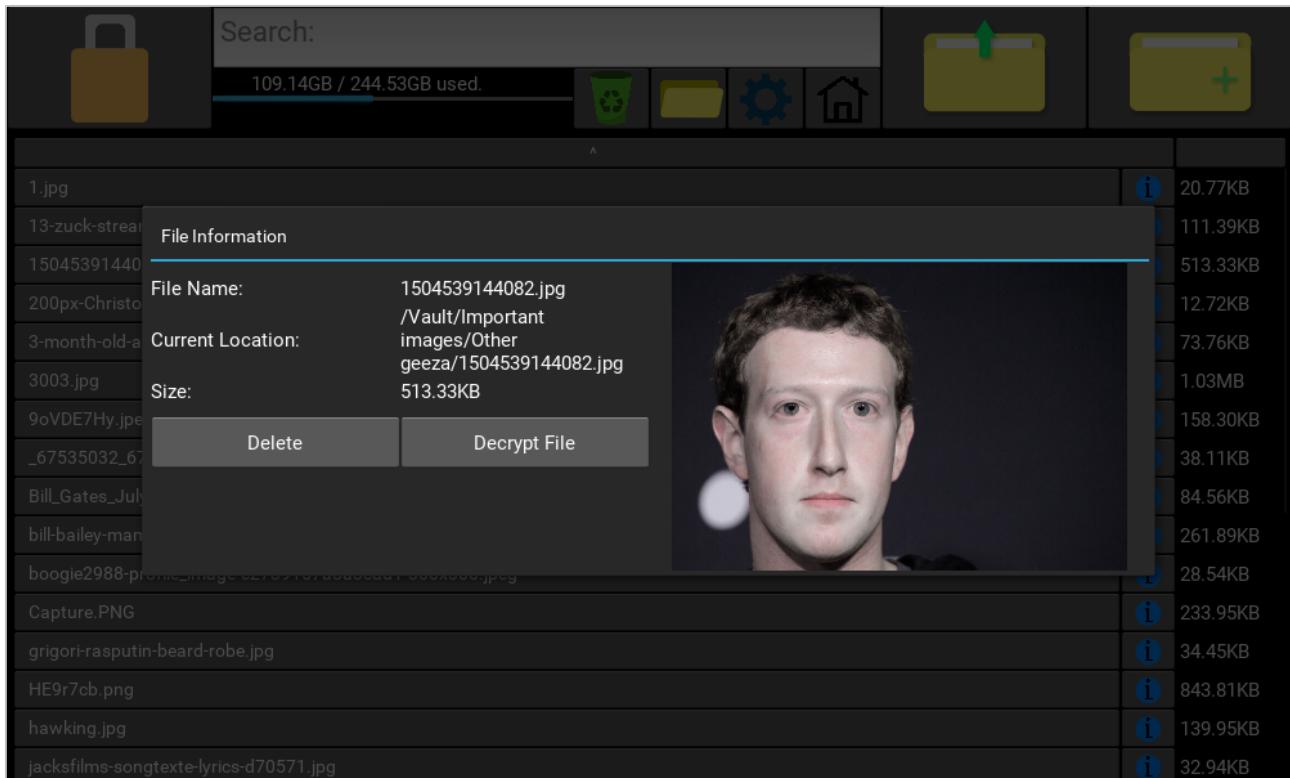
The information tab shows you information about the file:

- 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.

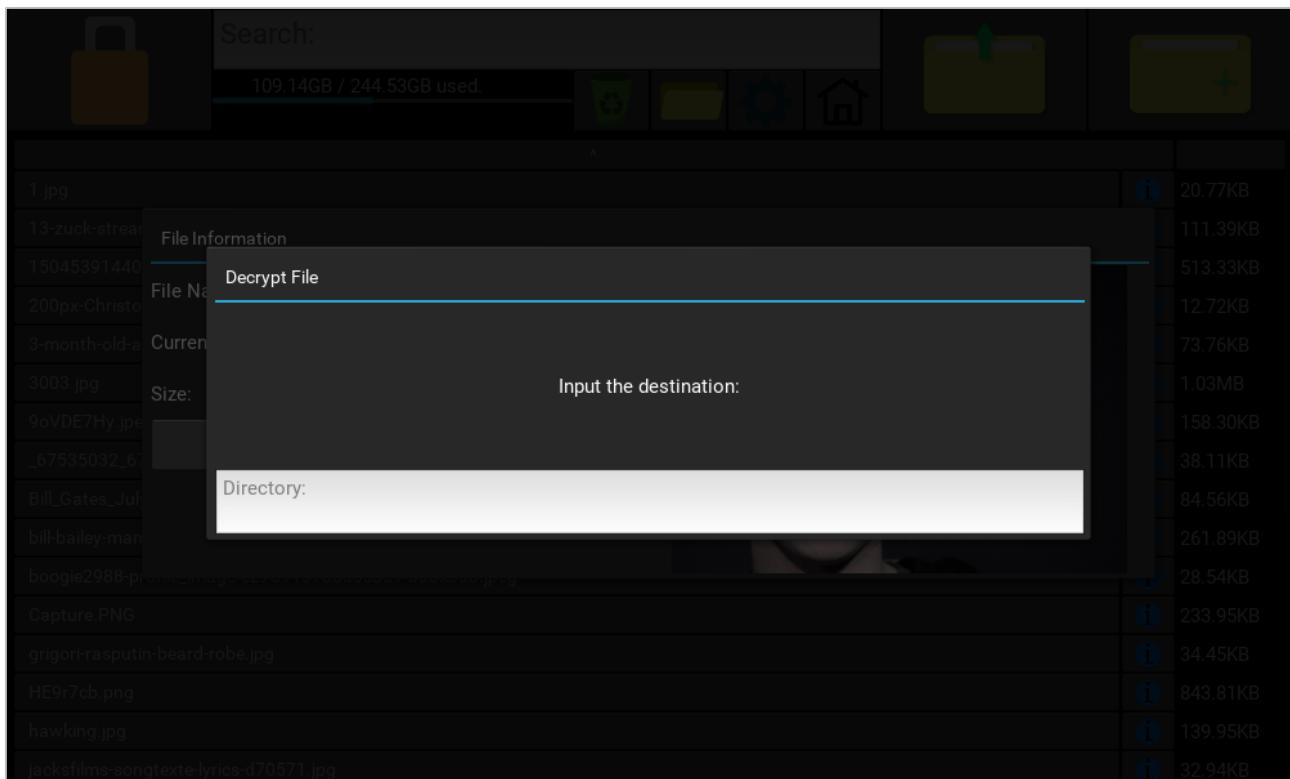
You also have a few options within the information tab to chose from:

- 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:



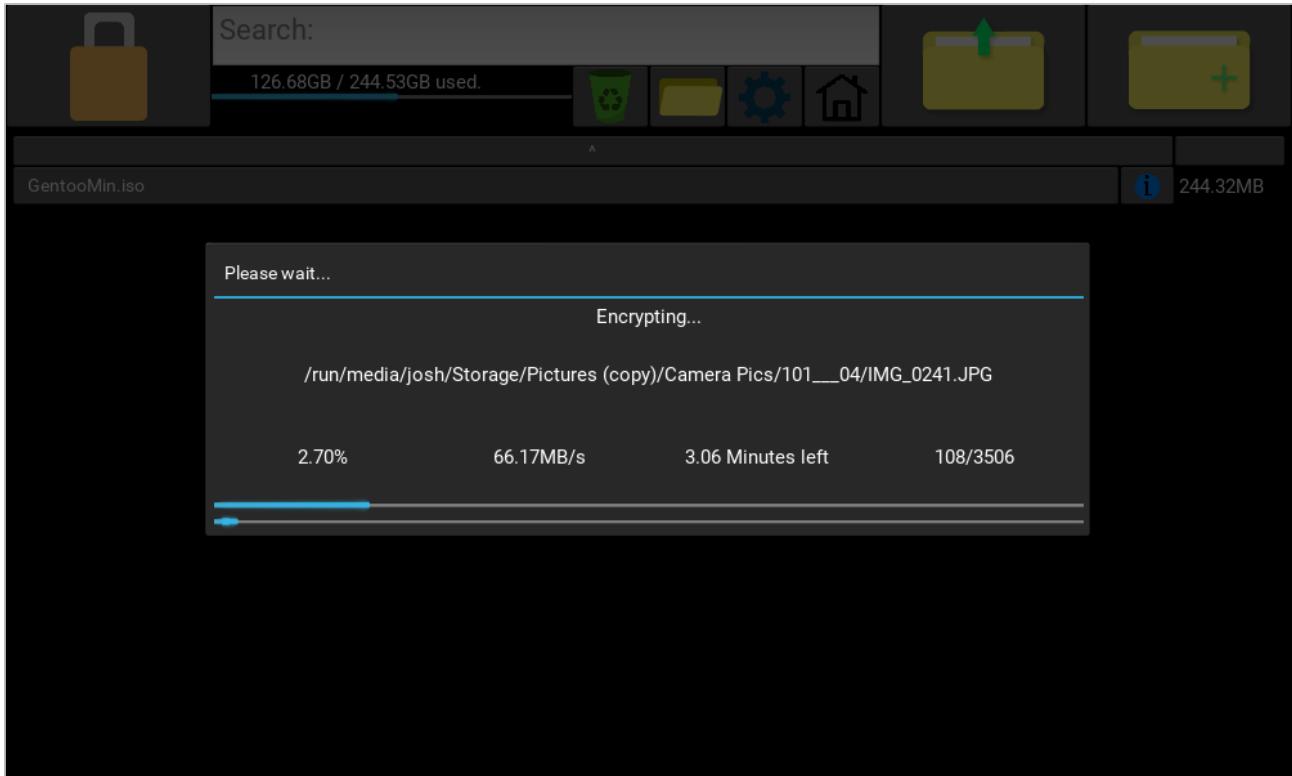
Once you input a correct directory name, it decrypts the file to that directory. If the path ends with the file separator (e.g. "/"), then it will be decrypted to that folder with its original name. If the path does not, then it is saved to that exact location, with that new name. For example, if I wanted 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.

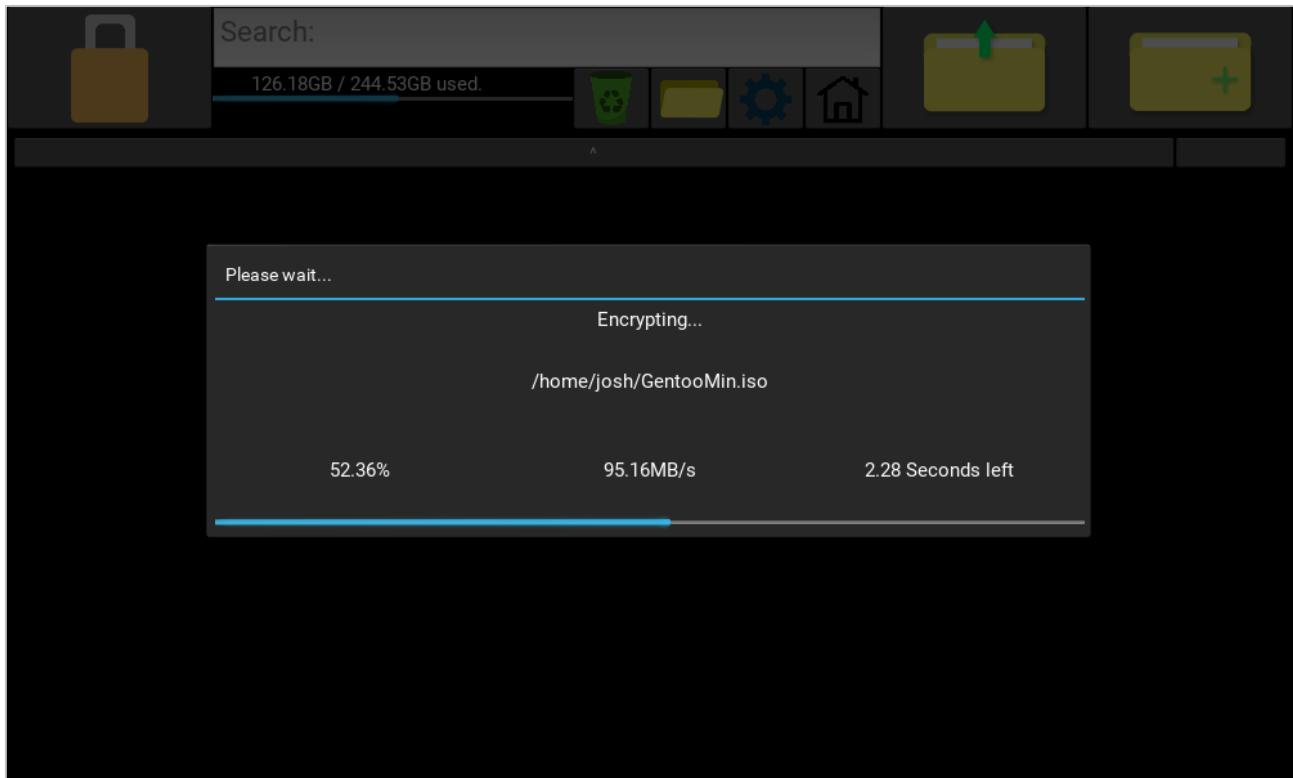
## Encryption and Decryption Status

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):



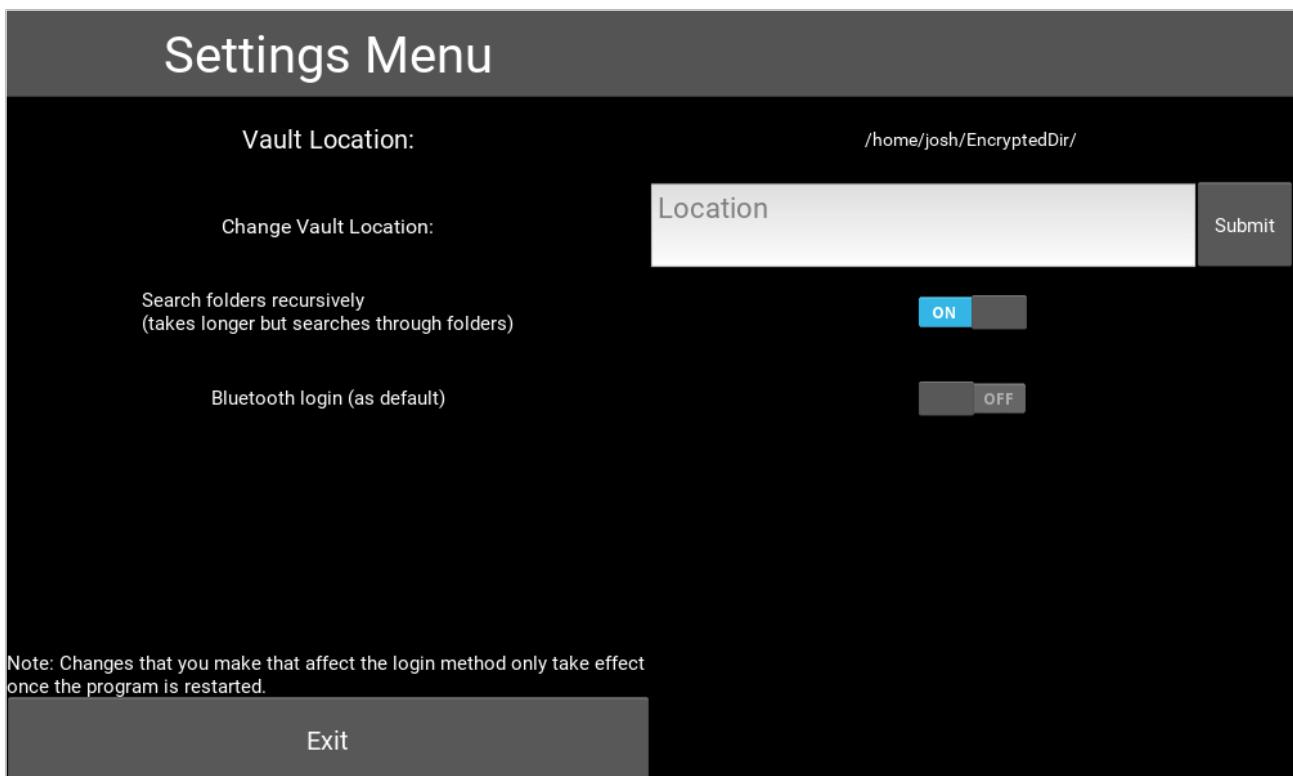
Here is an image of a file being encrypted (244 MB):



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:

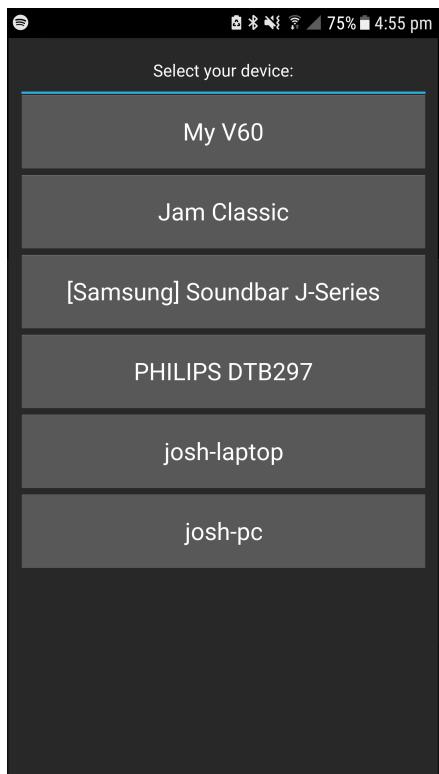


If no results are found, a popup opens saying "No results found for: search item".

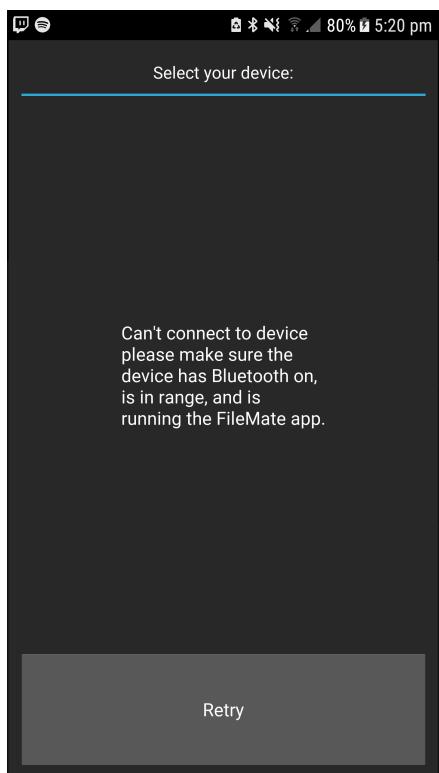
## The Mobile App GUI

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):

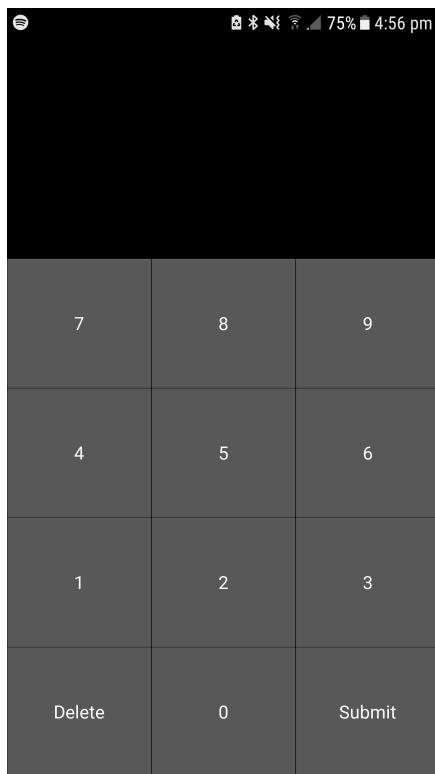


If you can't connect to a device, then you are greeted by this large popup:

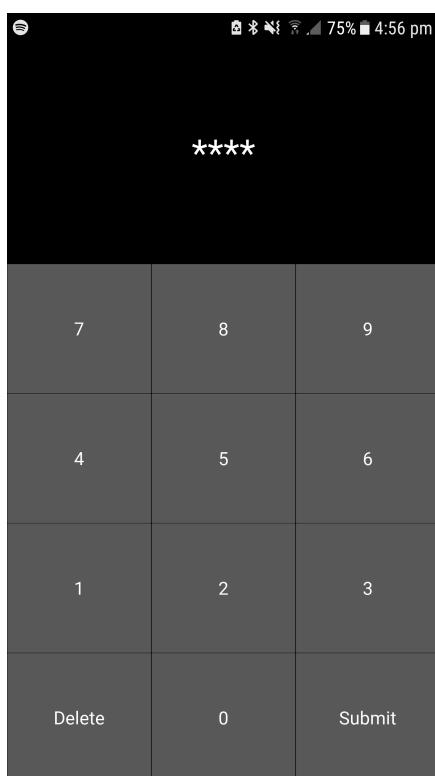


If you click retry, it takes you back to the start screen.

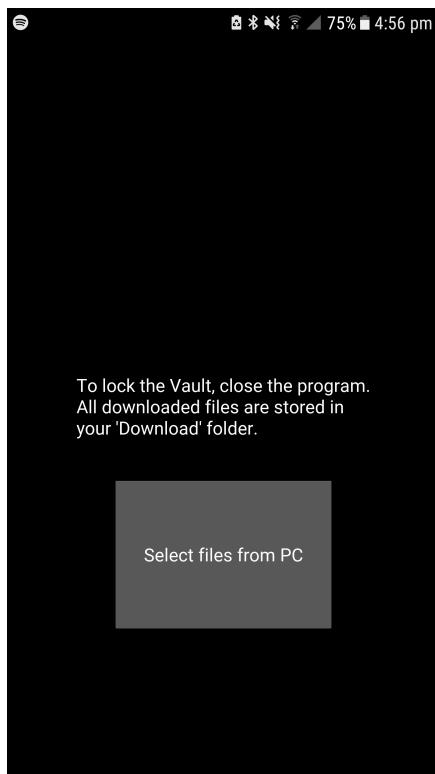
If you connect successfully, you are greeted by this screen:



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

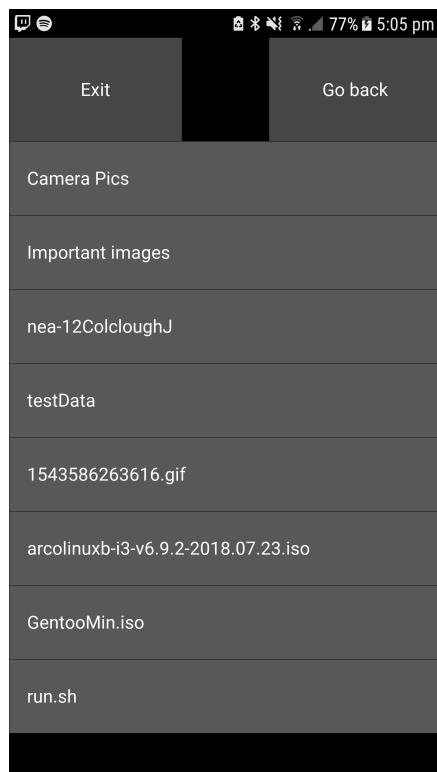


Once you press submit, a popup opens telling you if the key was correct or not, and then you are put into this screen:

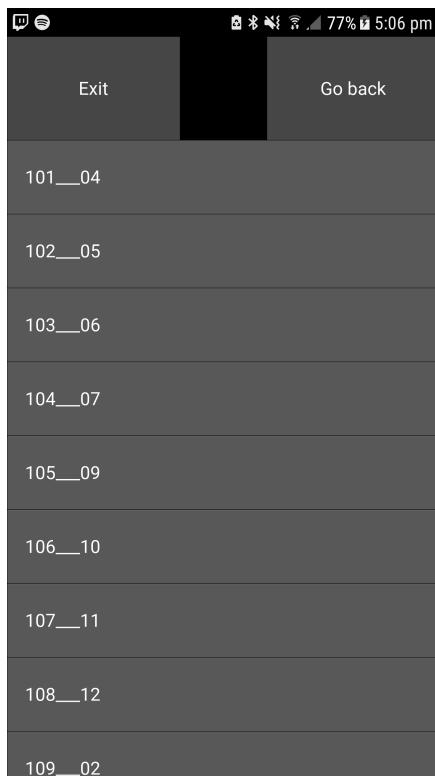


From here you can either close the app and leave it running in the background, or browse files. If you close the app completely, the app on the PC locks. **NOTE TO MR REGHIF: I will make a YouTube video here at some point showing the full process. Btw have a nice christmas.**

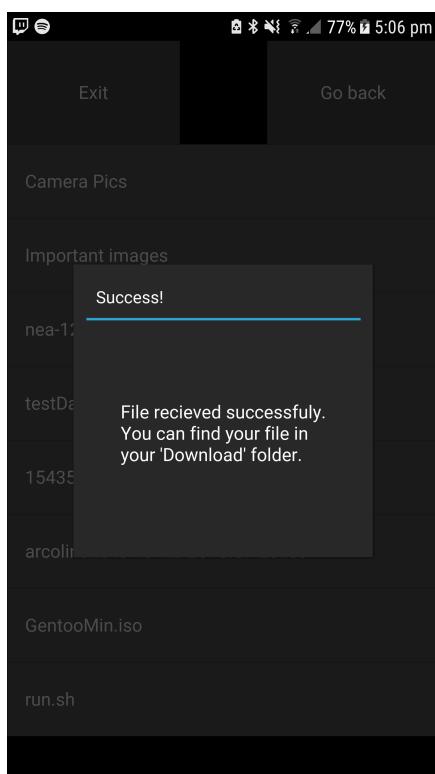
Here is what the app looks like while you are browsing files:



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.

# 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.

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, 0xEB, 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,0x0A,0x49,0x06,0x24,0x5C,0xC2,0xD3,0xAC,0x62,0x91,0x95,0xE4,0x79,
15     0xE7,0xC8,0x37,0x6D,0x8D,0xD5,0x4E,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,0x43,0xC4,0xDE,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,0x8B,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,0xE5,0x15,0x46,0x57,0xA7,0x8D,0x9D,0x84,
27     0x90,0xD8,0xAB,0x00,0x8C,0xBC,0xD3,0xA0,0xF7,0xE4,0x58,0x05,0xB8,0xB3,0x45,0x06,
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,0xB7,0x62,0x0E,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,0x61,0xc2,0x9f,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 {0x00,0x02,0x04,0x06,0x08,0x0a,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,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
61     0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xde,
62     0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,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,0x0a,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,0x74,0x77,0x72,0x71,
75     0x50,0x53,0x56,0x55,0x5c,0x5f,0x5a,0x59,0x49,0x48,0x4b,0x4e,0x4d,0x44,0x47,0x42,0x41,
76     0xc0,0xc3,0xc6,0xc5,0xcc,0xca,0xc9,0xd8,0xdb,0xde,0xdd,0xd4,0xd2,0xd1,0xd7,0xd4,0xd1}

```

```

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,0xc6,0xf9,0xf0,0xeb,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,0x43,0x34,0x3d,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,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,0x81,0x88,0xe3,0fe,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,0x90,0xa1,0xaa,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,0x06,0xd,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,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     0xa0,0x07,0x10,0x1d,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,0xe8,0xe5,0xf2,0xff,0xb4,0xb9,0xae,0xa3,0x80,0x8d,0x9a,0x97}
139
140 var mul14 = [256]byte {0x00,0x0e,0x1c,0x12,0x38,0x36,0x24,0x2a,0x70,0x7e,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,0x45,0x57,0x59,0x73,0x7d,0x6f,0x61,
144     0xad,0xa3,0xb1,0xbff,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, 0xa2, 0xac, 0xbe, 0xb0, 0xea, 0xe4, 0xf6, 0xf8, 0xd2, 0xdc, 0xce, 0xc0,
151          0x7a, 0x74, 0x66, 0x68, 0x42, 0x4c, 0x5e, 0x50, 0xa0, 0x04, 0x16, 0x18, 0x32, 0x3c, 0x2e, 0x20,
152          0xec, 0xe2, 0xf0, 0xfe, 0xd4, 0xda, 0xc8, 0xc6, 0x9c, 0x92, 0x80, 0x8e, 0xa4, 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, 0xfd, 0xa7, 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).

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 }
32
33 func workerEnc(jobs <-chan work, results chan<- work, expandedKey *[176]byte) { // Encrypts a chunk when
34     given (a chunk of length bufferSize)
35     for job := range jobs {
36         for i := 0; i < len(job.buff); i += 16 {
```

```

35     AES.Encrypt(job.buf[i:i+16], expandedKey)
36   }
37   results<- work{buff: job.buf, offset: job.offset}
38 }
39 }
40
41 // Worker that encrypts chunk given
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.buf); i += 16 {
45       AES.Decrypt(job.buf[i:i+16], expandedKey)
46       if (fileSize - int(job.offset) - i) == 16 {      // If on the last block of whole file
47         var focus int = int(job.buf[i+15])
48         var focusCount int = 1
49         if focus < 16 {      // If the last number is less than 16 (the maximum amount of padding to add is 15)
50           for j := 14; (int(job.buf[i+j]) == focus) && (j >= 0); j-- {
51             if int(job.buf[i+j]) == focus { focusCount++ }
52           }
53           if focus == focusCount {
54             job.buf = append(job.buf[:(:i+16-focus)], job.buf[i+16:]...) // If the number of bytes at the
55             end is equal to the value of each byte, then remove them, as it is padding.
56           }
57         }
58       }
59       results<- work{buff: job.buf, offset: job.offset}
60     }
61   }
62
63 func EncryptFile(expandedKey *[176]byte, f, w string) {
64   a, err := os.Open(f)      // Open original file to get statistics and read data.
65   check(err)
66   aInfo, err := a.Stat()   // Get statistics
67   check(err)
68
69   fileSize := int(aInfo.Size()) // Get size of original file
70
71   if _, err := os.Stat(w); err == nil { // If file already exists, delete it
72     os.Remove(w)
73   }
74
75   var workingWorkers int = 0
76   var workerNum int = getNumOfCores()*2
77
78   jobs := make(chan work, workerNum)      // Make two channels for go routines to communicate over.
79   results := make(chan work, workerNum)    // Each has a buffer of length workerNum
80
81   for i := 0; i < workerNum; i++ {
82     go workerEnc(jobs, results, expandedKey)
83   }
84 /*
85 Each go routine will be given access to the job channel, where each worker then waits to complete the job.
86 Once the job is completed, the go routine pushes the result onto the result channel, where the result can be
87 received by the main routine. The results are read once all of the go routines are busy, or if the file
88 is completed, then the remaining workers still working are asked for their results.
89 */
90   var bufferSize int = DEFAULT_BUFFER_SIZE
91
92   if fileSize < bufferSize {      // If the buffer size is larger than the file size, just read the whole file.
93     bufferSize = fileSize
94   }
95
96   var buffCount int = 0    // Keeps track of how far through the file we are
97
98   e, err := os.OpenFile(w, os.O_CREATE|os.O_WRONLY, 0644) // Open file for writing.
99   check(err) // Check it opened correctly
100
101  // Append key so that when decrypting, the key can be checked before decrypting the whole file.
102  var originalKey = expandedKey[:16]

```

```

103     AES.Encrypt(originalKey, expandedKey)
104     e.Write(originalKey)
105     offset := 16
106
107     for buffCount < fileSize {    // Same as a while buffCount < fileSize: in python3
108         if bufferSize > (fileSize - buffCount) {
109             bufferSize = fileSize - buffCount    // If this is the last block, read the amount of data left in the
110             file.
111         }
112
113         buff := make([]byte, bufferSize) // Make a slice the size of the buffer
114         _, err := io.ReadFull(a, buff) // Read the contents of the original file, but only enough to fill the buff
115         array.
116
117         if len(buff) % 16 != 0 { // If the buffer is not divisible by 16 (usually the end of a file), then padding
118             needs to be added.
119             var extraNeeded int
120             var l int = len(buff)
121             for l % 16 != 0 {          // extraNeeded holds the value for how much padding the block needs.
122                 l++
123                 extraNeeded++
124             }
125
126             for i := 0; i < extraNeeded; i++ {           // Add the number of extra bytes needed to the end of
127                 the block, if the block is not long enough.
128             buff = append(buff, byte(extraNeeded)) // For example, the array [1, 1, 1, 1, 1, 1, 1, 1] would have
129             the number 8 appended to then end 8 times to make the array 16 in length.
130             } // This is so that when the block is decrypted, the pattern can be recognised, and the correct amount
131             of padding can be removed.
132         }
133
134         jobs <- work{buff: buff, offset: int64(offset)}
135         workingWorkers++
136
137         if workingWorkers == workerNum {
138             workingWorkers = 0
139             for i := 0; i < workerNum; i++ {
140                 wk := <-results
141                 e.WriteAt(wk.buffer, wk.offset)
142             }
143         }
144
145         offset += bufferSize
146         buffCount += bufferSize
147     }
148
149     if workingWorkers != 0 {
150         for i := 0; i < workingWorkers; i++ {
151             wk := <-results
152             e.WriteAt(wk.buffer, wk.offset)
153         }
154     }
155
156     close(jobs)
157     close(results)
158
159     a.Close() // Close the files used.
160     e.Close()
161 }
162
163 func DecryptFile(expandedKey *[176]byte, f, w string) {
164     a, err := os.Open(f)
165     check(err)
166     aInfo, err := a.Stat()
167     check(err)

```

```

165     fileSize := int(aInfo.Size())-16 // Take away length of added key for checksum
166
167     if _, err := os.Stat(w); err == nil { // If file exists, delete it
168         os.Remove(w)
169     }
170
171     var bufferSize int = DEFAULT_BUFFER_SIZE
172
173     var workingWorkers int = 0
174     var workerNum int = getNumOfCores()*2
175
176
177     jobs := make(chan work, workerNum)      // Make two channels for go routines to communicate over.
178     results := make(chan work, workerNum)   // Each has a buffer of length workerNum
179
180     for i := 0; i < workerNum; i++ {
181         go workerDec(jobs, results, expandedKey, fileSize)
182     }
183
184     if fileSize < bufferSize {
185         bufferSize = fileSize
186     }
187
188     var buffCount int = 0
189
190     e, err := os.OpenFile(w, os.O_CREATE|os.O_WRONLY, 0644) // Open file
191     check(err)
192
193     // Check first block is key
194     firstBlock := make([]byte, 16)
195     _, er := io.ReadFull(a, firstBlock)
196     check(er)
197     AES.Decrypt(firstBlock, expandedKey)
198
199     if AESCheckKey.CompareSlices(expandedKey[:16], firstBlock) { // If key is valid
200         offset := 0
201         a.Seek(16, 0) // Move past key
202         for buffCount < fileSize{ // While the data done is less than the fileSize
203             if bufferSize > (fileSize - buffCount) {
204                 bufferSize = fileSize - buffCount
205             }
206
207             buff := make([]byte, bufferSize)
208             _, err := io.ReadFull(a, buff) // Ignore the number of bytes read (_)
209             check(err)
210
211             jobs<- work{buff: buff, offset: int64(offset)}
212             workingWorkers++
213
214             if workingWorkers == workerNum {
215                 workingWorkers = 0
216                 for i := 0; i < workerNum; i++ {
217                     wk := <-results
218                     e.WriteAt(wk.buffer, wk.offset)
219                 }
220             }
221
222             offset += bufferSize
223             buffCount += bufferSize
224         }
225
226         if workingWorkers != 0 {
227             for i := 0; i < workingWorkers; i++ {
228                 wk := <-results
229                 e.WriteAt(wk.buffer, wk.offset)
230             }
231         }
232         close(jobs)
233         close(results)

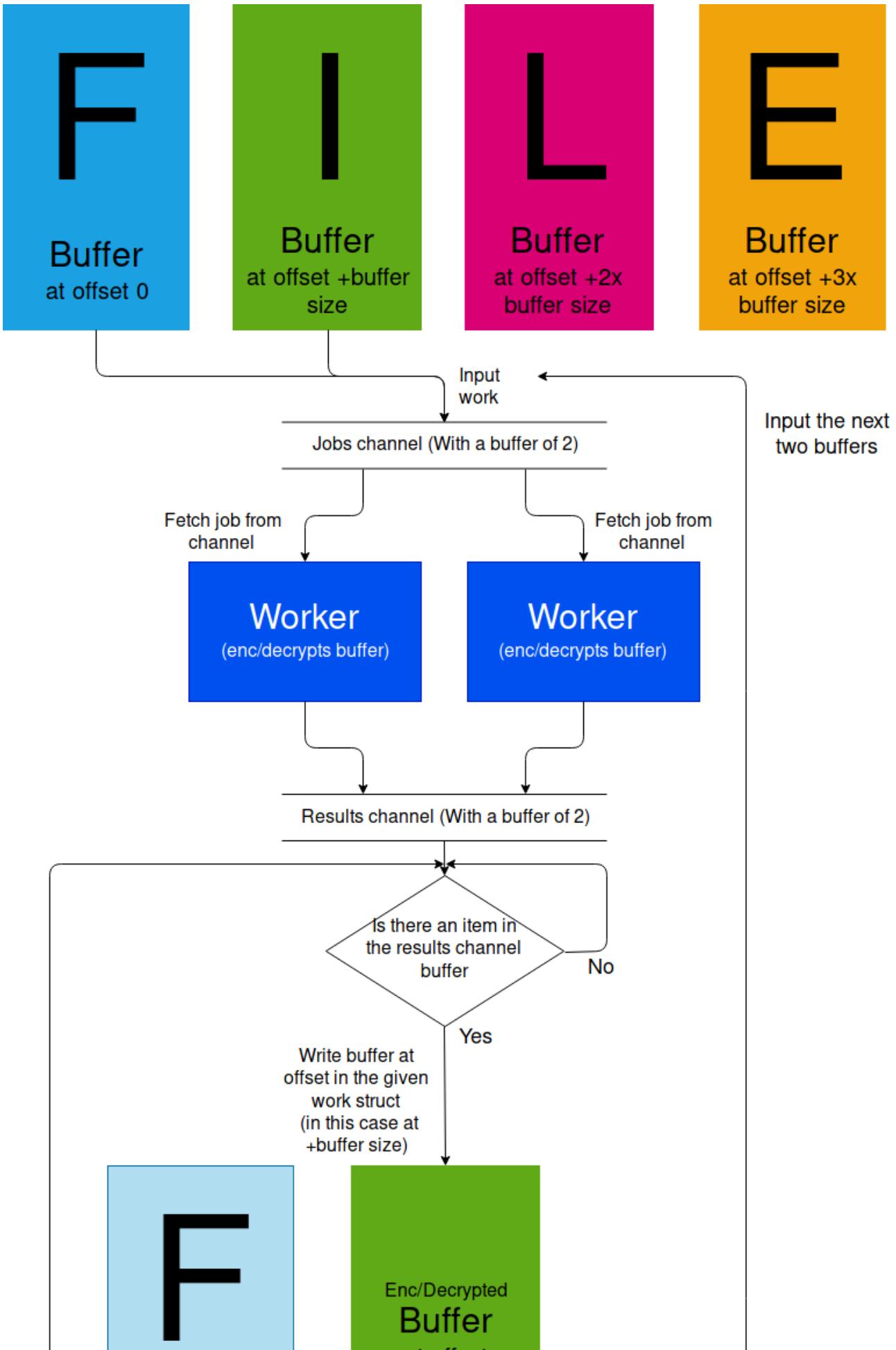
```

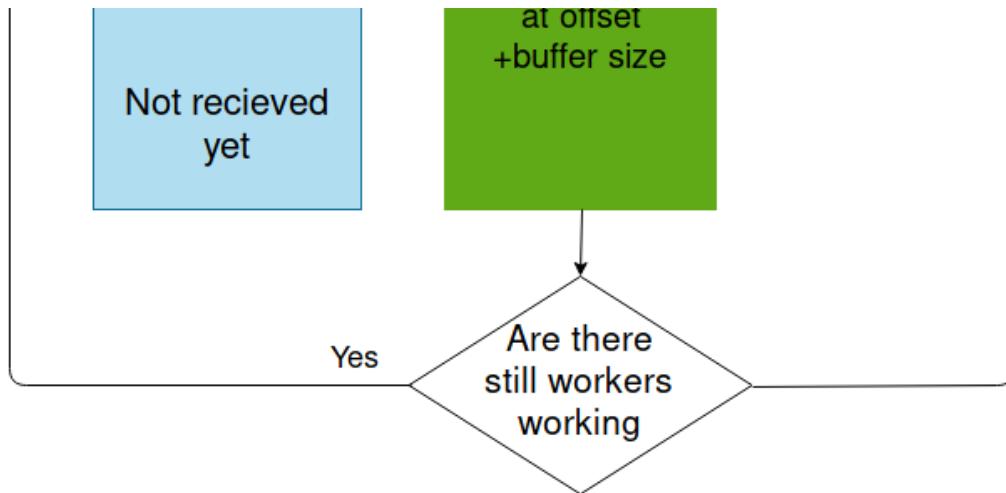
```

234     } else {
235         panic("Invalid Key") // If first block is not equal to the key, then do not bother trying to decrypt the
236         file.
237     }
238     a.Close()
239     e.Close()
240 }
241
242 // For dealing with directories
243 func EncryptList(expandedKey *[176]byte, fileList []string, targetList []string) { // Encrypts list of files
244     given to the corresponding targets.
245     if len(fileList) != len(targetList) { panic("fileList and targList are different in length") }
246     for i := range fileList {
247         EncryptFile(expandedKey, fileList[i], targetList[i])
248     }
249 }
250
251 func DecryptList(expandedKey *[176]byte, fileList []string, targetList []string) { // Decrypts list of files
252     given to the corresponding targets.
253     if len(fileList) != len(targetList) { panic("fileList and targList are different in length") }
254     for i := range fileList {
255         DecryptFile(expandedKey, fileList[i], targetList[i])
256     }
257 }
```

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 (routines 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 routine, 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 routine 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.

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, and block sizes of file systems are usually set to powers of two (the operating system reads data from the hard disk in blocks of a pre-determined amount).

`checkKey` 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. This is because when I encrypt files, I append the encrypted key to the beginning of the new file, so that it can be checked when decrypting the file.

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.

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.

I have created a benchmark function in `aes.go` to measure the true speed of the operation:

```
1 import (
2     ...      // Not actually a line of code, just skipping through the code
3     "testing"
4 )
5 ...
6 ...
7
8 // BENCHMARK
9 func BenchmarkEncryptFile(b *testing.B) {
10    f := "/home/josh/nea-12ColcloughJ/Write-Up/Write-up.pdf"    // This write up
11    w := "/home/josh/temp" // Temporary location
12    key := []byte{0x00, 0x0b, 0x16, 0x1d, 0x2c, 0x27, 0x3a, 0x31, 0x58, 0x53, 0x4e, 0x45, 0x74, 0x7f, 0x62, 0x69}
13    // Random key
14    for n := 0; n < b.N; n++ {
15        encryptFile(key, f, w)
16    }
16 }
```

Now if I run `go test aes_test.go -bench=.`, Go tests the function however many times it can in a certain period, and gives how long it took on average in nanoseconds. Here is an example output:

```
1 goos: linux
2 goarch: amd64
3 BenchmarkEncryptFile-4          10    187673212 ns/op
4 PASS
5 ok    command-line-arguments  2.085s
```

I made a small Python program to work out the speed, and for the sake of transparency here it is:

```
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)    # x10^-9 because it 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")
```

There is no error checking or anything since I am the only person using it (and because I'm lazy).

The result is that on an i7-6600k, the speed of AES was 18.92 MB/s (187673212 ns/op for 10 operations), while on a laptop i7-3537U it was 10.21 MB/s. The speed is quite good, as when working on small files (< 2 MB), opening and editing files should be almost instant, and even opening larger files shouldn't take too long. At 18.92 MB/s a 2 GB file would take 105.7 seconds, or 1:45 minutes, which isn't too bad. I timed a 2 GB file (2,036,826,112 bytes precisely) and it actually took 2:00 minutes (yes, on the dot), probably due to background processes.

## SHA256:

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

```
1  k = [0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,      #Round constants
2      0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
3      0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
4      0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
5      0xe49b69c1, 0xefbe4786, 0xfc19dc6, 0x240calcc,
6      0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
7      0x983e5152, 0xa831c66d, 0xb00327c8, 0xbff597fc7,
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
```



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:

Here is the code for BLAKE2b (`code/python-go/blake.go`):

```
1 package main
2
3 import (
4     "fmt"
5     "math"
6     "os"
7     "io"
8     "io/ioutil"
9 )
10
11
12 // Initial constants.
13 var k = [8]uint64 {0x6A09E667F3BCC908,
14                     0xBB67AE8584CAA73B,
15                     0x3C6EF372FE94F82B,
16                     0xA54FF53A5F1D36F1,
17                     0x510E527FADE682D1,
18                     0x9B05688C2B3E6C1F,
19                     0x1F83D9ABFB41BD6B,
20                     0x5BE0CD19137E2179}
21
22 var sigma = [12][16]uint64 {{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
23                             {14, 10, 4, 8, 9, 15, 13, 6, 1, 12, 0, 2, 11, 7, 5, 3},
24                             {11, 8, 12, 0, 5, 2, 15, 13, 10, 14, 3, 6, 7, 1, 9, 4},
25                             {7, 9, 3, 1, 13, 12, 11, 14, 2, 6, 5, 10, 4, 0, 15, 8},
26                             {9, 0, 5, 7, 2, 4, 10, 15, 14, 1, 11, 12, 6, 8, 3, 13},
27                             {2, 12, 6, 10, 0, 11, 8, 3, 4, 13, 7, 5, 15, 14, 1, 9},
28                             {12, 5, 1, 15, 14, 13, 4, 10, 0, 7, 6, 3, 9, 2, 8, 11},
29                             {13, 11, 7, 14, 12, 1, 3, 9, 5, 0, 15, 4, 8, 6, 2, 10},
30                             {6, 15, 14, 9, 11, 3, 0, 8, 12, 2, 13, 7, 1, 4, 10, 5},
31                             {10, 2, 8, 4, 7, 6, 1, 5, 15, 11, 9, 14, 3, 12, 13, 0},
32                             {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
33                             {14, 10, 4, 8, 9, 15, 13, 6, 1, 12, 0, 2, 11, 7, 5, 3}}
34
35 // Research: https://tools.ietf.org/pdf/rfc7693.pdf
36
37 func check(e error) {      //Used for checking errors when reading/writing to files.
38     if e != nil {
39         panic(e)
40     }
41 }
42
43 func rotR64(in uint64, n int) uint64 { // Rotates 64 bit words right by n amount.
44     return (in >> uint(n)) ^ (in << (64 - uint(n)))
45 }
46
47 func mix(v [16]uint64, a, b, c, d int, x, y uint64) [16]uint64 {
48     v[a] = v[a] + v[b] + x
49     v[d] = rotR64((v[d] ^ v[a]), 32)
50
51     v[c] = v[c] + v[d]
52     v[b] = rotR64((v[b] ^ v[c]), 24)
53
54     v[a] = v[a] + v[b] + y
55     v[d] = rotR64((v[d] ^ v[a]), 16)
56
57     v[c] = v[c] + v[d]
```

```

58     v[b] = rotR64((v[b] ^ v[c]), 63)
59
60     return v
61 }
62
63 func get64(in []uint64) uint64 { // Gets a full 64-bit word from a list of 8 64-bit bytes.
64     return uint64(in[0] ^ (in[1] << 8) ^ (in[2] << 16) ^ (in[3] << 24) ^ (in[4] << 32) ^ (in[5] << 40) ^ (in[6]
65     << 48) ^ (in[7] << 56))
66 }
67
68 func blakeCompress(h [8]uint64, block []uint64, t int, lastBlock bool) [8]uint64 { // Compressing function
69     var v = [16]uint64{} // Current vector
70     for i := 0; i < 8; i++ {
71         v[i] = h[i]
72         v[i+8] = k[i]
73     }
74     v[12] = v[12] ^ uint64(math.Mod(float64(t), 18446744073709552000)) // 2 ^ 64 = 18446744073709552000
75     v[13] = v[13] ^ (uint64(t) >> 64)
76
77     if lastBlock {
78         v[14] = ^v[14] // NOT v[14]
79     }
80
81     var m [16] uint64
82     for i := 0; i < 16; i++ {
83         m[i] = get64(block[i*8:(i*8)+8])
84     }
85     for i := 0; i < 12; i++ {
86         // Mix
87         v = mix(v, 0, 4, 8, 12, m[sigma[i][0]], m[sigma[i][1]])
88         v = mix(v, 1, 5, 9, 13, m[sigma[i][2]], m[sigma[i][3]])
89         v = mix(v, 2, 6, 10, 14, m[sigma[i][4]], m[sigma[i][5]])
90         v = mix(v, 3, 7, 11, 15, m[sigma[i][6]], m[sigma[i][7]])
91
92         v = mix(v, 0, 5, 10, 15, m[sigma[i][8]], m[sigma[i][9]]) // Rows have been shifted
93         v = mix(v, 1, 6, 11, 12, m[sigma[i][10]], m[sigma[i][11]])
94         v = mix(v, 2, 7, 8, 13, m[sigma[i][12]], m[sigma[i][13]])
95         v = mix(v, 3, 4, 9, 14, m[sigma[i][14]], m[sigma[i][15]])
96     }
97
98     for i := 0; i < 8; i++ {
99         h[i] ^= v[i]
100        h[i] ^= v[i+8]
101    }
102
103    return h
104 }
105
106 func getNiceOutput(h [8]uint64) [64]byte {
107     var out [64]byte
108     for i := 0; i < 8; i++ {
109         for j := 8; j != 0; j-- {
110             out[i*8+(j-1)] = byte(((h[i] << uint64(64 - uint64((j)*8))) & 0xFFFFFFFFFFFFFF) >> 56)
111         }
112     }
113     return out
114 }
115
116 func BLAKEchecksum(f string, hashL int) [64]byte {
117     h := k // Initialize h0-7 with initial values.
118     h[0] = h[0] ^ (0x01010000 ^ uint64(hashL)) // Not using a key
119
120     a, err := os.Open(f) // Open file
121     check(err)
122     aInfo, err := a.Stat() // Get statistics of file
123     check(err)
124
125     fileSize := int(aInfo.Size()) // Get size of original file

```

```

126
127     var bufferSize int = 65536
128
129     if fileSize < bufferSize {    // If the buffer size is larger than the file size, just read the whole file.
130         bufferSize = fileSize
131     }
132
133     var buffCount int = 0    // Keeps track of how far through the file we are
134     var bytesFed int = 0
135     var bytesLeft int = fileSize
136
137     for buffCount < fileSize {
138         if bufferSize > (fileSize - buffCount) {
139             bufferSize = fileSize - buffCount
140         }
141         buff := make([]uint64, bufferSize)
142         tempBuff := make([]byte, bufferSize) // Make a slice the size of the buffer
143         _, err := io.ReadFull(a, tempBuff) // Read the contents of the original file, but only enough to fill the
144         //buff array.
145         // The "_" tells go to ignore the value returned by io.ReadFull, which in
146         //this case is the number of bytes read.
147         check(err)
148         for i := range tempBuff {
149             buff[i] = uint64(tempBuff[i])
150         }
151         tempBuff = nil // Delete array
152
153         for len(buff) % 128 != 0 {
154             buff = append(buff, 0) // Append 0s when buffer is not long enough
155         }
156
157         for i := 0; i < bufferSize; i += 128 {
158             if bytesLeft <= 128 {
159                 h = blakeCompress(h, buff[i:i+128], bytesFed+bytesLeft, true)
160             } else {
161                 bytesFed += 128
162                 h = blakeCompress(h, buff[i:i+128], bytesFed, false)
163             }
164             bytesLeft -= 128
165         }
166
167         buffCount += bufferSize
168     }
169     a.Close()
170
171     return getNiceOutput(h)
172 }
173
174 func main() {
175     bytes, err := ioutil.ReadAll(os.Stdin) // Read file to hash from stdin
176     check(err)
177     f := string(bytes)
178
179     fmt.Printf("%x", BLAKEchecksum(f, 64)) // Returns the hex digest in hex form over stdout
180 }
```

The way that BLAKE2b goes through the file is very similar to AES, so I stole some of the code from my AES and adapted it slightly.

The `main()` function is much simpler than AES's `main()` function, as I am only receiving one input: the path of the file that needs to be hashed.

`getNiceOutput` 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 `BA80A53F981C4D0D`. The function `getNiceOutput` 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: `0000001`

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

The `get64` function turns 8 64-bit words into 1 64-bit word, or 8 bytes.

The executable itself ( `BLAKE` or `BLAKEWin` )

Now similar to `aes.go`, I wrote a benchmark function for `blake.go` that looks like this:

```
1 import (
2 ...
3 "testing"
4 )
5 ...
6 ...
7
8 func BenchmarkBLAKEchecksum(b *testing.B) {
9     f := "/home/josh/neal2ColcloughJ/Write-Up/Write-up.pdf"
10    for n := 0; n < b.N; n++ {
11        BLAKEchecksum(f, 64)
12    }
13 }
```

And the results on the i7-6600k were:

```
1 goos: linux
2 goarch: amd64
3 BenchmarkBLAKEchecksum-4      20      59748242 ns/op
4 PASS
5 ok   command-line-arguments  1.264s
```

which (using the Python program I used while testing `aes.go`) is 59.42 MB/s, which is very decent. The results of BLAKE on the i7-3537U was `104655494 ns/op`, which is 33.92 MB/s. You will only really get a slowdown when opening and editing large files (> around 300MB), however the user is probably not likely to do that very often. If the user does not need to open the file, but just encrypt or decrypt it, then checksums aren't needed so the slowdown doesn't apply.

## The Sorts:

Here is the code for the sorts ( [code/python-go/sortsCythonSource/sortsCy.pyx](#) ):

```
1 cpdef int compareStrings(fileObj, string2, fileObjects=True): # Returns 0 if str1 < str2, 1 if str1 > str2, and
2 if str1 == str2
3     cdef int count = 0
4
5     if fileObjects:
6         string1 = fileObj.name
7     else:
8         string1 = fileObj
9
10    while not (count >= len(string1) or count >= len(string2)):-#
11        if ord(string2[count].lower()) < ord(string1[count].lower()):
12            return 1
13        elif ord(string2[count].lower()) > ord(string1[count].lower()):
14            return 0
15        else:
16            if ord(string2[count]) < ord(string1[count]):      #if the same name but with capitals - e.g (Usb
Backup) and (usb backup)
17                return 1
18            elif ord(string2[count]) > ord(string1[count]):
19                return 0
20            else:
21                if string2 == string1:
22                    return 2
23                else:
24                    count += 1
25    if len(string1) > len(string2):
26        return 1
27    elif len(string1) < len(string2):
28        return 0
29    else:
30        raise ValueError("Two strings are the same in compareStrings.")
31
32
33 cpdef list quickSortAlph(list myList, fileObjects=True): #Quick sorts alphabetically
34     cdef list left = []
35     cdef list right = [] #Make seperate l+r lists, and add on at the end.
36     cdef list middle = []
37     if len(myList) > 1:
38         pivot = myList[int(len(myList)/2)]
39         for item in myList:
40             if fileObjects:
41                 leftSide = compareStrings(pivot, item.name)
42             else:
43                 leftSide = compareStrings(pivot, item, False)
44             if leftSide == 2:
45                 middle.append(item)
46             elif leftSide == 1:
47                 left.append(item)
48             elif leftSide == 0:
49                 right.append(item)
50
51     return quickSortAlph(left, fileObjects)+middle+quickSortAlph(right, fileObjects)
52 else:
53     return myList
54
55
56 cpdef list quickSortSize(list fileObjects):
57     cdef list left = []
58     cdef list right = [] #Make seperate l+r lists, and add on at the end.
59     cdef list middle = []
60     cdef int pivotSize
61     if len(fileObjects) > 1:
62         pivot = fileObjects[int(len(fileObjects)/2)]
63         if pivot.rawSize == " -":
64             pivotSize = 0
```

```

65     else:
66         pivotSize = pivot.rawSize
67
68     for i in fileObjects:
69         if i.rawSize == " -":
70             left.append(i)
71         elif i.rawSize < pivotSize:
72             left.append(i)
73         elif i.rawSize > pivotSize:
74             right.append(i)
75         else:
76             middle.append(i)
77     return quickSortSize(left)+middle+quickSortSize(right)
78 else:
79     return fileObjects
80
81 cpdef list quickSortTuples(list tuples): #Quick sorts tuples (for search results).
82     cdef list left = []
83     cdef list right = [] #Make seperate l+r lists, and add on at the end.
84     cdef list middle = []
85     cdef int pivot
86     if len(tuples) > 1:
87         pivot = tuples[int(len(tuples)/2)][0]
88         for i in tuples:
89             if i[0] < pivot:
90                 left.append(i)
91             elif i[0] > pivot:
92                 right.append(i)
93             else:
94                 middle.append(i)
95     return quickSortTuples(left)+middle+quickSortTuples(right)
96 else:
97     return tuples

```

The way Cython works, is that functions defined using `cpdef` are accessible by both Cython and Python, while variables can be defined using `cdef` internally, as they only need to be accessible via Cython.

Cython speeds Python code up significantly, depending on how many variables have a declared variable type. If variables / functions are used a lot, then it is a good idea to declare their type. Variables that are not used so often do not need to be defined with their data type, as they may only be used a couple of times during the program.

When you build the Cython program, you get a shared object file (.so), and a C file. You can import the name of the .c file in Python to use the module.

`quickSortTuples` 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"). `quickSortTuples` 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`. `quickSortAlph` 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:

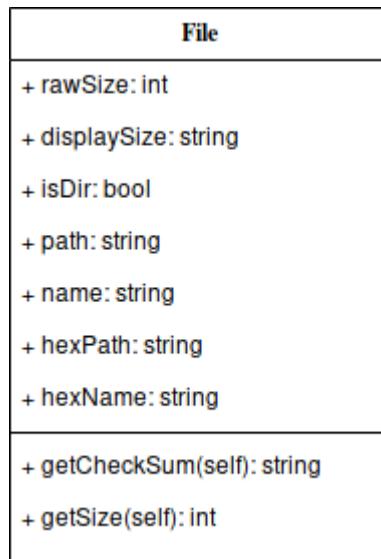
```
1  from os import path as osPath
2  from os import listdir
3  from subprocess import Popen, PIPE
4
5  import aesFName
6
7  class File:
8
9      def __init__(self, screen, hexPath, hexName, fileSep, extension=None, isDir=False, name=None, path=None):
10         self.outerScreen = screen
11         self._totalSize = 0
12         self.hexPath, self.hexName, self.isDir, self.fileSep, self.extension = hexPath, hexName, isDir, fileSep,
13         extension
14         self.thumbDir = ""
15         self.checkSum = None
16         self.rawSize = self.__getFileSize()
17         self.size = self.outerScreen.getGoodUnit(self.rawSize)
18         self.isDir = isDir
19         if path == None:
20             self.path = self.__getNormDir(self.hexPath)
21         else:
22             self.path = path
23         if name == None:
24             self.name = aesFName.decryptFileName(self.outerScreen.key, self.hexName)
25         else:
26             self.name = name
27
28         if extension == None:
29             extension = self.path.split(".")
30             self.extension = extension[-1].lower()
31         else:
32             extension = extension.lower()
33
34         if self.isDir:
35             self.hexPath += self.fileSep
36             self.path += self.fileSep
37
38             self.relPath = self.hexPath.replace(self.outerScreen.path, "") # Encrypted path relative to root
39             folder of Vault
40
41     def __getNormDir(self, hexDir):          # Private functions as they are usually only needed once and should
42             only be callable from within the class
43             hexDir = (hexDir.replace(self.outerScreen.path, "")).split(self.fileSep)
44             for i in range(len(hexDir)):
45                 hexDir[i] = aesFName.decryptFileName(self.outerScreen.key, hexDir[i])
46
47             return self.fileSep.join(hexDir)
48
49     def __getFileSize(self, recurse=True):
50         if self.isDir:
51             if recurse:
52                 self._totalSize = 0
53                 self.__recursiveSize(self.hexPath)
54                 size = self._totalSize
55                 return size
56             else:
57                 return " - "
58         else:
```

```

57     try:
58         size = osPath.getsize(self.hexPath) # Imported from os module
59         return size
60     except Exception as e:
61         print(e, "couldn't get size.")
62         return " -"
63
64     def __recursiveSize(self, f, encrypt=False): #Get size of folders.
65         fs =.listdir(f)
66         for item in fs:
67             if encrypt:
68                 item = aesFName.encryptFileName(self.key, item)
69             if osPath.isdir(f+self.fileSep+item):
70                 try:
71                     self.__recursiveSize(f+self.fileSep+item)
72                 except OSError:
73                     pass
74             else:
75                 try:
76                     self._totalSize += osPath.getsize(f+self.fileSep+item)
77                 except PermissionError: #Thrown when the file is owned by another user/administrator.
78                     pass
79
80     def decryptRelPath(self):      # Gets relative path from root of Vault in human form
81         splitPath = self.relPath.split(self.fileSep)
82         return self.fileSep.join([aesFName.decryptFileName(self.outerScreen.key, i) for i in splitPath])
83
84     def getCheckSum(self, new=True):
85         if self.checkSum == None or new:
86             if self.fileSep == "\\":
87                 goproc = Popen(self.outerScreen.startDir+"BLAKEWin.exe", stdin=PIPE, stdout=PIPE)
88             elif self.fileSep == "/":
89                 goproc = Popen(self.outerScreen.startDir+"BLAKE", stdin=PIPE, stdout=PIPE)
90
91             out, err = goproc.communicate((self.hexPath).encode())
92             if err != None:
93                 raise ValueError(err)
94
95             self.checkSum = out.decode()
96
97     return self.checkSum

```

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("input", "mouse", "mouse,disable_multitouch") # Disable multitouch features used on mobile apps.
7  Config.write()
8
9  from kivy.app import App
10 from kivy.uix.screenmanager import ScreenManager, Screen, FadeTransition
11 from kivy.lang import Builder
12
13 #####Import personal classes#####
14 from mainScClass import MainScreen
15 from loginClass import LoginScreen, LoginScreenBT
16 from settingsScreen import SettingsScreen
17
18 #####Import config functions#####
19 import configOperations
20
21
22 def runUI():
23     ui = uiApp(title="FileMate")
24     ui.run()
25
26     # When program closes:
27     print("Deleting temp files.")
28     try:
29         fSep = configOperations.getFileSep()
30         rmtree(gettempdir()+fSep+"FileMate"+fSep) # Remove all temporary files.
31     except FileNotFoundError:
32         print("No temp files.")
33     print("App closed.")
34
35 class uiApp(App):
```

```

36     def build(self):
37         sm = ScreenManager()
38
39         sm.transition = FadeTransition() # Set transition animation when changing screens.
40         fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc =
41         configOperations.runConfigOperations()
42         # Load kv files for each screen.
43         Builder.load_file(startDir+"kivyStuff/kvFiles/mainSc.kv")      # MainScreen styling.
44         Builder.load_file(startDir+"kivyStuff/kvFiles/settingsSc.kv") # SettingsScreen styling.
45
46         if useBT:
47             Builder.load_file(startDir+"kivyStuff/kvFiles/loginScBT.kv")
48             sm.add_widget(LoginScreenBT(fileSep, path, startDir, name="Login"))
49         else:
50             Builder.load_file(startDir+"kivyStuff/kvFiles/loginSc.kv")
51             sm.add_widget(LoginScreen(fileSep, path, startDir, name="Login"))
52
53         sm.add_widget(MainScreen(fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc,
54         name="Main")) # fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, **kwargs
55         sm.add_widget(SettingsScreen(sm.get_screen("Main"), configLoc, name="Settings"))
56         sm.current = "Login"
57
58
59
60     if __name__ == "__main__":
61         runUI()

```

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 class LoginScreen(Screen):
22
23     def __init__(self, fileSep, path, startDir, **kwargs):
24         self.fileSep, self.path, self.startDir = fileSep, path, startDir # Start dir is location of running
25         program, path is path of vault
26         super(Screen, self).__init__(**kwargs) # Run kivy's Screen.__init__ function with the key word
27         arguments (such as size or position)
28         self.key = ""
29
30     def cancel(self):
31         self.manager.get_screen("Main").useBT = True # Am now using BT
32         Builder.load_file(self.startDir+"kivyStuff/kvFiles/loginScBT.kv") # Load the styling file for BT login
33         screen
34         self.manager.add_widget(LoginScreenBT(self.fileSep, self.path, self.startDir, name="Login")) # Create
35         the new screen
36         self.name = "Dead" # To prevent clash with new login screen.
37         self.manager.current = "Login" # Change to Login
38         self.manager.remove_widget(self) # Remove self from the app
39         self = None # Kill self
40
41     def findFile(self, dir): # For finding a file to decrypt first block and compare it with key given.
42         fs =.listdir(dir)
43         for item in fs:
44             if osIsDir(dir+item+"/"):
45                 if self.count == 0:
46                     self.findFile(dir+item+"/")
47                 else:
48                     return
49             else:
50                 self.decryptTestFile = dir+item
51                 self.count += 1
52                 return
53
54     def passToTerm(self, key, d): # Makes a pipe to communicate with AES
55         if self.fileSep == "\\":
56             programe = "AESWin"
57         else:
58             programe = "AES"
59         goproc = Popen(self.startDir+programe, stdin=PIPE, stdout=PIPE)
60         out, err = goproc.communicate("test, "+d+", 0, ").encode()+key.encode()
61         return out
62
63     def getIfValidKey(self, inputKey): # Gets the output of the AES key checker.
64         if len(listdir(self.path)) > 1:
65             self.decryptTestFile = ""
66             self.count = 0
67             self.findFile(self.path)
68             diditwork = self.passToTerm(inputKey, self.decryptTestFile)
69             if diditwork == b"-Valid-\n": #The go program prints "-Valid-\n" or "-Invalid-\n" once it is done
70             checking the key.
71                 return True
72             else:
73                 return False
74         else:
75             return True
76
77     def checkKey(self, inputKey): # Handles the GUI while the key is checked, and passes key to functions to
78     check it.
79         if len(inputKey) < 1:
80             Popup(title="Invalid", content=Label(text="Invalid key, valid key\ncontains at least 1 digit."),
81             pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
82             return "Login"
83         try:
84             int(inputKey)
85         except:
86             Popup(title="Invalid", content=Label(text="Invalid key, valid key\ncontains no letters."),
87             pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
88             return "Login"
89         else:
90

```

```

82         if len(str(inputKey)) > 16:
83             Popup(title="Invalid", content=Label(text="Invalid key, longer than\n 16 characters."),
84             pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
85             return "Login"
86         else:
87             inputKeyTemp = []
88             for i in range(len(inputKey)):
89                 inputKeyTemp.append(int(inputKey[i]))
90             inputKey = inputKeyTemp
91             inputKey = SHA.getSHA128of16(inputKey)
92             key = " ".join(str(i) for i in inputKey)
93             valid = self.getIfValidKey(key)
94             if valid:
95                 self.ids.keyInput.text = "" #reset key input if valid
96                 self.key = key
97                 return "Main"
98             else:
99                 Popup(title="Invalid", content=Label(text="Invalid key."), pos_hint={"x_center": .5,
100 "y_center": .5}, size_hint=(.4, .4)).open()
101             return "Login"
102
103     def needToSetKey(self):      # Gets text to tell the user if they need to set a key.
104         if len(listdir(self.path)) == 0:    # If there are no files in the vault, then the key hasn't been set
105             yet.
106             return "Input New Key (Write this down if you have to)"
107         else:
108             return "Input Key"
109
110
111     class LoginScreenBT(LoginScreen, Screen):      #Has the same methods as LoginScreen, but some overwritten with
112         bluetooth.
113
114         def __init__(self, fileSep, path, startDir, **kwargs):
115             self.fileSep, self.path, self.startDir = fileSep, path, startDir
116             super(Screen, self).__init__(**kwargs)
117             self.key = ""
118
119         def on_enter(self):
120             self.serv = None
121             self.startServ = Clock.schedule_once(self.startSrv, 0.5) # Use the clock to allow the screen to be
122             rendered. (Waits 0.5 seconds for screen to be loaded.)
123
124         def checkKey(self, inputKey):
125             inputKey = inputKey.split(",")
126             inputKey = inputKey[:-1]
127             key = " ".join(str(i) for i in inputKey)      #Formatting for AES
128             valid = self.getIfValidKey(key)
129             if valid:
130                 self.key = key
131                 self.manager.get_screen("Main").key = key
132                 return True
133             else:
134                 return False
135
136         def cancel(self):
137             if self.serv != None:
138                 self.manager.get_screen("Main").serverSock.close() # Close the BT server
139                 self.serv.join() # Close the thread that runs the server (in LoginScreenBT)
140                 try:
141                     self.manager.get_screen("Main").clientSock.close()
142                 except AttributeError: # clientSock will not be initialized if there are no clients.
143                     pass
144             else:
145                 self.startServ.cancel() # Cancels scheduled task to start server, as we are switching screens
146             anyway.
147
148             print("Server closed.")
149             self.manager.get_screen("Main").useBT = False
150             Builder.load_file(self.startDir+"kivyStuff/kvFiles/loginSc.kv")

```

```

145     self.manager.add_widget(LoginScreen(self.filePath, self.path, self.startDir, name="Login"))
146     self.name = "Dead"      # To prevent name clash with other login screen.
147     self.manager.current = "Login"
148     self.manager.remove_widget(self)
149     self = None
150
151     def startSrv(self, dt=None):
152         self.serv = Thread(target=self.manager.get_screen("Main").startBT, daemon=True) # Runs the function in
153         MainScreen, which prevents segmentation, so I don't have to shutdown server when screen is switched
154         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 parsing in functions with multiple arguments to widgets/threads
5 from subprocess import Popen, PIPE
6
7 from kivy.uix.scrollview import ScrollView
8 from kivy.uix.gridlayout import GridLayout
9 from kivy.uix.boxlayout import BoxLayout
10 from kivy.uix.label import Label
11 from kivy.clock import Clock
12 from kivy.clock import mainthread
13 from kivy.core.window import Window
14 from kivy.uix.button import Button
15 from kivy.uix.progressbar import ProgressBar
16 from kivy.uix.popup import Popup
17 from kivy.uix.image import Image
18 from kivy.uix.screenmanager import Screen
19
20 from fileClass import File
21 import aesFName
22 import sortsCv
23 # Own kivy classes
24 import mainBtns
25 from settingsScreen import SettingsScreen
26 import mainSmallPops as mainSPops
27
28 try:
29     from bluetooth import *
30 except:
31     pass
32
33 class MainScreen(Screen):
34
35     class infoLabel(Label): # Not a popup so only suitable place.
36         pass
37
38     def __init__(self, fileSep, osTemp, startDir, assetsPath, path, recurseSearch, useBT, configLoc, **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         selfascending = True      # Sort order
43         self.key = ""
44         self.encPop = None
45         self.entered = False

```

```

45     self.validBTKey = False
46     self.useBTTemp = self.useBT
47     self.previousDir = None
48     self.lastPathSent = ""
49     self.recycleFolder = ""
50     self.recycleName = ""
51     self.thumbsName = ""

52
53     Window.bind(on_dropfile=self.onFileDrop)      #Binding the function to execute when a file is dropped
54     into the window.
55     self.currentDir = self.path

56
57     def on_enter(self): # When the screen is started.
58         self.key = self.manager.get_screen("Login").key # Fetch the key from the Login Screen.
59         if not self.entered:
60             self.setupSortButtons() #Put sort buttons in place.
61             self.recycleName = aesFName.encryptFileName(self.key, ".$recycling")      # Prepare recycling and
62             thumbnail folder names for use in the program.
63             self.thumbsName = aesFName.encryptFileName(self.key, ".$thumbs")
64             self.recycleFolder = self.path+self.recycleName+self.fileSep

65         if not os.path.exists(self.recycleFolder):
66             print("Recycling folder not found in directory, making one now.")
67             os.makedirs(self.recycleFolder)
68
69         self.entered = True

70
71         if self.recycleFolder in self.currentDir:
72             self.createButtons(self.List(self.path))      # Don't want to log into the recycling bin, as the
73             user might get confused.
74             else:
75                 self.createButtons(self.List(self.currentDir)) # Loads previous directory.

76         def on_leave(self):      # Kept separate from lock because i may want to add more screens that need the key,
77             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         def lock(self, fromRunServ=False): # Procedure for when the program is locked. If it has been called from
98             runServMain, then we might still be on login screen, so don't change screen to login, and restart the server.
99             self.clearUpTempFiles() # Delete all temporary files (decrypted files ready for use).
100            if self.useBT:
101                self.manager.get_screen("Login").ids.clientLabel.text = ""

102            if fromRunServ and self.validBTKey == False:
103                self.runServMain()
104            else:
105                self.validBTKey = False
106                return mainthread(self.changeToLogin())      #Change screen to the login screen. Ran on mainthread
in case it was called in

```

```

107     def runServMain(self):
108         self.serverSock = BluetoothSocket( RFCOMM )
109         self.serverSock.bind(("" ,PORT_ANY))
110         self.serverSock.listen(1)
111
112         uuid = "80677070-a2f5-11e8-b568-0800200c9a66"
113
114         try:
115             advertise_service(self.serverSock, "FileMateServer",
116                             service_id = uuid,
117                             service_classes = [ uuid, SERIAL_PORT_CLASS ],
118                             profiles = [ SERIAL_PORT_PROFILE ],)
119         except BluetoothError as e:
120             Popup(title="Error", content=Label(text="Bluetooth not available.\nPlease make sure your bluetooth is on,\nor change to normal login.\n\nReason: "+str(e)), size_hint=(.4, .4), auto_dismiss=True).open()
121             return
122
123         print("[BT]: Waiting for connection on RFCOMM channel", self.serverSock.getsockname()[1])
124
125         self.clientSock, self.clientInfo = self.serverSock.accept() # Wait for a connection
126         print("[BT]: Accepted connection from ", self.clientInfo)
127         self.manager.get_screen("Login").ids.clientLabel.text = "Connected to: "+str(self.clientInfo[0])
128
129         numbers = []
130         data = ""
131         buff = []
132         backCommand = [33, 66, 65, 67, 75, 33] # !BACK!
133         fileSelectCommand = [33, 70, 73, 76, 69, 83, 69, 76, 69, 67, 84, 33] # !FILESELECT!
134         endHeader = [126, 33, 69, 78, 68, 83, 69, 76, 69, 67, 84, 33] # ~!ENDSELECT!
135
136         try:
137             while len(data) > -1:
138                 data = self.clientSock.recv(1024) # Recieve 1kb of data
139                 print("[BT]: Received data.")
140                 if not self.validBTKey: # If the key is not valid yet, BT server has to wait
141                     for key
142                         numbers.append(str(data, "utf-8"))
143                         if b"~" in data: # End of key message
144                             append = False
145                             tempNums = "".join(numbers)
146                             tempNums = tempNums.replace("#", "")
147                             tempNums = tempNums.replace("~", "")
148                             if self.manager.get_screen("Login").checkKey(tempNums): # Check the key in login.
149                                 numbers = []
150                                 self.clientSock.send("1")
151                                 print("[BT]: Send true.")
152                                 self.validBTKey = True
153                                 self.recycleName = aesFName.encryptFileName(self.key, ".$recycling") # Set so that
154                                 file list can be sent
155                                 self.thumbsName = aesFName.encryptFileName(self.key, ".$thumbs")
156                                 self.sendFileList(self.getListForSend(self.path))
157                                 mainthread(self.changeToMain()) # Exit thread and change screen to main.
158                                 else:
159                                     numbers = []
160                                     self.clientSock.send("0")
161                                     print("[BT]: Send false.")
162                                     self.validBTKey = False
163
164                                     else:
165                                         for i in data:
166                                             buff.append(i)
167
168                                         if buff[:6] == backCommand: # Buffer is reset every time a header is found
169                                             pathBack = self.getPathBack(self.lastPathSent)
170                                             if (not pathBack) or (pathBack.replace(self.path, "") == pathBack): # If you can't
171 go further back (if pathBack has less than path, then remove returns the original string).
172                                             print("[BT]: Can't go further back.")
173                                             self.clientSock.send("!ENDOFTREE!")
174                                         else:

```

```

172             self.sendFileList(self.getListForSend(pathBack))
173             buff = []
174
175         elif buff[:12] == fileSelectCommand: # If the command is fileSelect
176             commandParams = buff[12:]           # Get parameters (buffer will not be reset)
177             if commandParams[-12:] == endHeader: # If end of the buffer is the endHeader, then
178                 proceed.
179
180             fileWantedList = commandParams[:-12]
181             fileWanted = ""
182             for letter in fileWantedList:
183                 fileWanted += chr(letter)
184
185             print("[BT]:", fileWanted, "fileWanted")
186             buff = []
187             filesInPath = self.List(self.lastPathSent) # Get list of files at directory
188             requested.
189
190             f = 0
191             fileObj = None
192             while (f < len(filesInPath)) and (fileObj == None): # Searches for the file in the
193                 path
194                 if filesInPath[f].name == fileWanted:
195                     fileObj = filesInPath[f]
196                     f += 1
197
198                 if fileObj != None:    # If the file was found, then send it
199                     if fileObj.isdir: # If it was a directory then send the list of files in that
200                         directory.
201                         self.sendFileList(self.getListForSend(fileObj.hexPath))
202                     else:
203                         self.makeSendFile(fileObj) # Otherwise send the file.
204
205                 else:
206                     print("[BT]: Couldn't find that file :/")
207                     self.clientSock.send("!NOTFOUND!")
208
209             elif len(buff) > 12: # Clear buffer and wait for next command.
210                 buff = []
211
212     except IOError as e:
213         print(e)      # Will be caused when app on mobile closes.
214
215         print("[BT]: Closed.")
216
217         self.clientSock.close()
218         self.serverSock.close()
219         self.lock(fromRunServ=True)
220
221     def sendFileList(self, fileList):
222         # File list sent like: !FILELIST!--filename1--filename2~!!ENDLIST!
223         self.clientSock.send("!FILELIST!")
224         print("[BT]: Sent !FILELIST!")
225
226         for i in fileList:
227             self.clientSock.send("--{}".format(i))
228
229         print("[BT]: Sent full list, now sent end.")
230         self.clientSock.send("~!!ENDLIST!")
231
232     def getListForSend(self, path):
233         if not path:
234             return False
235         else:
236             fs = os.listdir(path)
237             listOfFolders = []
238             listOfFiles = []

```

```

237     for item in fs:
238         if (item != self.thumbsName) and (item != self.recycleName):
239             if os.path.isdir(path+item):
240                 listOffFolders.append(aesFName.decryptFileName(self.key, item))
241             else:
242                 listOffFiles.append(aesFName.decryptFileName(self.key, item))
243
244     self.lastPathSent = path
245
246     return sortsCy.quickSortAlph(listOffFolders, fileObjects=False)+sortsCy.quickSortAlph(listOffFiles,
247 fileObjects=False) # Sort the list and return it
248
249
250 ##Functions for changing screen within threads (used to prevent segmentation faults)
251 @mainthread
252 def changeToMain(self):
253     self.manager.current = "Main"
254
255 @mainthread
256 def changeToLogin(self):    #Only used for checkServerStatus because you can only return a function or
variable, and if i execute this within the thread then it causes a segmentation fault.
257     self.manager.current = "Login"
258 #####
259
260     def startBT(self):
261         self.serverThread = Thread(target=self.runServMain, daemon=True)      #Start BT server as thread so the
screen still renders.
262         self.serverThread.start()
263
264     def setupSortButtons(self):
265         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.
266         self.nameSort = mainBtns.nameSortButton(self, text="^") # Default starts with Alphabetical sort
ascending.
267         self.sizeSort = mainBtns.sizeSortButton(self)
268         self.sortsGrid.add_widget(self.nameSort)
269         self.sortsGrid.add_widget(self.sizeSort)
270         self.add_widget(self.sortsGrid) #Add the sort buttons grid to the float layout of MainScreen.
271
272     def getGoodUnit(self, bytes):      #Get a good unit for displaying the sizes of files.
273         if bytes == " -":
274             return " -"
275         else:
276             divCount = 0
277             divisions = {0: "B", 1: "KB", 2: "MB", 3: "GB", 4: "TB", 5: "PB"}
278             while bytes > 1000:
279                 bytes = bytes/1000
280                 divCount += 1
281
282             return ("%.2f" % bytes) + divisions[divCount]
283
284     def getSortedFoldersAndFiles(self, fileObjects, inverse=False): # Get a sorted list of files for display.
Displays all folders before files.
285         folders = []
286         files = []
287         for i in range(len(fileObjects)):    #Separate into folders and files
288             if fileObjects[i].isDir:
289                 folders.append(fileObjects[i])
290             else:
291                 files.append(fileObjects[i])
292
293         foldersSort = sortsCy.quickSortAlph(folders)    #Quick sort the list of folders and the list of files.
294         filesSort = sortsCy.quickSortAlph(files)
295
296         if inverse: #If inverse
297             foldersSort = foldersSort[::-1] #Invert the array
298             filesSort = filesSort[::-1]
299

```

```

300     return foldersSort+filesSort
301
302     def openRecycling(self): # Open the recycling folder.
303         if not os.path.exists(self.recycleFolder):
304             print("Recycling folder doesn't exist, making one now.")
305             makedirs(self.recycleFolder)
306
307             Popup(title="Changed Mode",
308                 content=Label(text="You are now in the\nrecycling folder.\nClick files to restore, and \nenter the INFO menu\n\tto see more information,\nor delete the file permanently."),
309                 pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4)).open()
310             self.currentDir = self.recycleFolder
311             self.removeButtons()
312             print(self.currentDir, "current dir")
313             self.createButtons(self.List(self.currentDir))
314
315
316 #####Button Creation and button functions#####
317     def createButtonsCore(self, array): # Makes each file button with it's information and adds it to the scroll view.
318         self.currentList = array
319         for item in array:
320             if item.name != ".$recycling" and item.name != ".$thumbs": # If the folder is the recycling folder or thumbnail temporary folder, don't draw it.
321                 back = (1, 1, 1, 1)
322                 if item.isdir: # Colour folders darker than files
323                     back = (0.3, 0.3, 0.3, 1) # Works as a tint rather than a colour.
324
325                 btn = mainBtns.listButton(self, item, text="    "+item.name), background_color=back)
326                 info = mainBtns.infoButton(self, item, background_color=back)
327
328                 btn.bind(size=btn.setter("text_size")) # Set the text to wrap within the button
329                 info.bind(size=info.setter("text_size"))
330                 fileS = Label(text=" "+str(item.size), size_hint=(.1, 1), halign="left", valign="middle")
331                 fileS.bind(size=fileS.setter("text_size")) # Wrap text in label
332                 self.grid.add_widget(btn)
333                 self.grid.add_widget(info)
334                 self.grid.add_widget(fileS)
335
336     def createButtons(self, fileObjects, sort=True):
337         self.currentList = []
338         if sort:
339             fileObjects = self.getSortedFoldersAndFiles(fileObjects) #Sort the list of files.
340
341             self.grid = GridLayout(cols=3, size_hint_y=None)
342             self.grid.bind(minimum_height=self.grid.setter("height"))
343             self.scroll = ScrollView(size_hint=(.99, .79), pos_hint={"x": .005, "y": 0}) #Grid is added to the scroll view.
344             self.scroll.add_widget(self.grid)
345
346             self.createButtonsCore(fileObjects)
347             self.add_widget(self.scroll) #Scroll view is added to the float layout of MainScreen.
348
349
350     def traverseButton(self, fileObj): # Function when file is clicked.
351         if self.recycleFolder not in self.currentDir:
352             if fileObj.isdir: #If is a folder, then display files within that folder.
353                 self.previousDir = self.currentDir
354                 self.currentDir = fileObj.hexPath
355                 self.resetButtons()
356             else: # If is a file, decrypt the file and open it.
357                 self.decrypt(fileObj)
358         else:
359             print("Recovering this file to path:", fileObj.name)
360             move(fileObj.hexPath, self.path) # Imported from shutil
361             self.refreshFiles()
362
363     def openAddFilePop(self): # Needs to be assigned to self.smallPop because if the screen is closed with the popup open (only possible when using Bluetooth), all crucial popups need to be closed.

```

```

364     self.smallPop = mainSPops.addFilePop(self)
365     self.smallPop.open()
366
367     def openAddFolderPop(self):
368         self.smallPop = mainSPops.addNewFolderPop(self)
369         self.smallPop.open()
370
371     def onFileInfoClose(self, fileObj, _): # _ is me discarding the popup object.
372         if os.path.exists(fileObj.thumbDir): # Remove temporary thumbnail directory once done with thumbnail
373             os.remove(fileObj.thumbDir)
374
375     def getFileInfo(self, fileObj):      #Get information about a file/folder.
376         size = (.7, .4) # Size of popup
377         if fileObj.extension == "png" or fileObj.extension == "jpg":
378             thumb = self.getThumbnail(fileObj)
379             size = (.8, .5) # Increase size of popup to display image preview.
380
381         # Works as: internalLayout -> scrollView + (Image?)
382         # scrollView contains infoGrid with all of the file's information.
383         internalLayout = BoxLayout(orientation="horizontal", size_hint=(1, 1))
384         scrollView = ScrollView()
385         self.infoPopup = Popup(title="File Information", content=internalLayout, pos_hint={"center_x": .5,
386 "center_y": .5}, size_hint=size)
387         self.infoPopup.bind(on_dismiss=partial(self.onFileInfoClose, fileObj, ))
388
388     infoGrid = GridLayout(cols=2, size_hint_y=None, row_default_height=40)
389     scrollView.add_widget(infoGrid)
390     internalLayout.add_widget(scrollView)
391
392     if fileObj.extension == "png" or fileObj.extension == "jpg":
393         internalLayout.add_widget(thumb)
394
395     infoGrid.add_widget(self.infoLabel(text="File Name:", halign="left", valign="middle"))
396     infoGrid.add_widget(self.infoLabel(text=fileObj.name, halign="left", valign="middle"))
397
398     infoGrid.add_widget(self.infoLabel(text="Current Location:", halign="left", valign="middle"))
399     infoGrid.add_widget(self.infoLabel(text="/Vault/" + fileObj.decryptRelPath(), halign="left",
valign="middle"))
400
401     infoGrid.add_widget(self.infoLabel(text="Size:", halign="left", valign="middle"))
402     infoGrid.add_widget(self.infoLabel(text=str(fileObj.size), halign="left", valign="middle"))
403
404     delText = "Delete"
405     if self.recycleFolder in self.currentDir: # If in the recycling folder, then delete the item
permanently.
406         delText = "Delete Permanently"
407
408     infoGrid.add_widget(mainBtns.deleteButton(self, fileObj, text=delText))
409
410     decBtnText = "Decrypt File"
411     if fileObj.isDir:
412         decBtnText = "Decrypt Folder"
413
414     if fileObj.rawSize > 0:
415         decBtn = Button(text=decBtnText, halign="left", valign="middle")
416         decBtn.bind(on_release=partial(self.decryptFileToLoc, fileObj))
417         infoGrid.add_widget(decBtn)
418
419     self.infoPopup.open()
420
421     def makeSendFile(self, fileObj, buttonInstance=None):
422         self.sendFile = mainSPops.btTransferPop(self, fileObj)
423         self.sendFile.open()
424
425     def moveFileToRecycling(self, fileObj):
426         print("Moving", fileObj.hexPath)
427         if os.path.exists(fileObj.hexPath):
428             move(fileObj.hexPath, self.recycleFolder) # Imported from shutil
429         else:

```

```

430         raise FileNotFoundError(fileObj.hexPath, "Not a file, can't move to recycling.") # Doesn't exist,
431         so issue with code somewhere.
432
433     def deleteFile(self, fileObj):
434         if os.path.exists(fileObj.hexPath): #Checks file actually exists before trying to delete it.
435             if self.recycleFolder not in self.currentDir: # If outside of recycling bin.
436                 print("Moving", fileObj.hexPath)
437                 if os.path.exists(self.recycleFolder+fileObj.hexName):
438                     if os.path.isdir(self.recycleFolder+fileObj.hexName):
439                         rmtree(self.recycleFolder+fileObj.hexName)
440                     else:
441                         os.remove(self.recycleFolder+fileObj.hexName)
442                     move(fileObj.hexPath, self.recycleFolder) # Imported from shutil
443                 else:
444                     print("Deleting:", fileObj.hexPath, "and checking temp.")
445                     if os.path.exists(self.osTemp+"FileMate"+self.fileSep+fileObj.name): # If removing
446                         permanently, check that the file is not decrypted in <system_temp>.
447                         os.remove(self.osTemp+"FileMate"+self.fileSep+fileObj.name)
448                     if fileObj.isDirectory: # Delete the file/folder
449                         rmtree(fileObj.hexPath) # Imported from shutil
450                     else:
451                         os.remove(fileObj.hexPath)
452                     self.refreshFiles()
453                     self.infoPopup.dismiss()
454
455             else:
456                 raise FileNotFoundError(fileObj.hexPath, "Not a file, can't delete.")
457
458     def goBackFolder(self): #Go up a folder.
459         if self.currentDir != self.path: #Can't go further past the vault dir.
460             self.previousDir = self.currentDir
461             if self.recycleFolder in self.currentDir:
462                 self.goHome()
463             else:
464                 self.currentDir = self.getPathBack(self.currentDir)
465                 self.resetButtons()
466             else:
467                 print("Can't go further up.")
468                 return False
469
470     def getPathForButton(self, item): # Get the path to the asset for each button.
471         return self.assetsPath+item
472
473     def removeButtons(self): # Remove the list of files.
474         self.grid.clear_widgets()
475         self.scroll.clear_widgets()
476         self.remove_widget(self.scroll)
477
478     def resetButtons(self): # Goes back to self.currentDir, different to refresh.
479         self.removeButtons()
480         self.nameSort.text = "^^"
481         self.sizeSort.text = ""
482         self.createButtons(self.List(self.currentDir))
483
484     def refreshFiles(self): # Refreshes the files in the current directory
485         self.removeButtons()
486         self.createButtons(self.List(self.currentDir))
487
488     def refreshButtons(self): # Refreshes file list buttons currently displayed.
489         self.removeButtons()
490         self.createButtons(self.currentList, False)
491
492     def goHome(self): #Takes the user back to the vault dir.
493         self.currentDir = self.path
494         self.refreshFiles()
495
496     def List(self, dir): # Lists a directory, returning File objects.
497         fs = os.listdir(dir)

```

```

497     listOfFolders = []
498     listOfFiles = []
499     for item in fs:
500         if os.path.isdir(dir+item):
501             listOfFolders.append(File(self, dir+item, item, self.fileSep, isDir=True))
502         else:
503             if os.path.exists(self.currentDir+self.thumbsName+self.fileSep+item):
504                 listOfFiles.append(File(self, dir+item, item, self.fileSep,
505 dir+self.thumbsName+self.fileSep+item))
506             else:
507                 listOfFiles.append(File(self, dir+item, item, self.fileSep))
508
509     return listOfFolders+listOfFiles
510
511 def getPathBack(self, origPath): # Gets the path above the current folder.
512     tempDir = origPath.split(self.fileSep)
513     del tempDir[-2]
514     tempDir = self.fileSep.join(tempDir)
515     return tempDir
516
517 #####Searches#####
518 def findAndSortCore(self, dirName, item):
519     files = self.List(dirName)
520     for fileObj in files:
521         loc = fileObj.name.find(item) # Find where in the word the item is found, if it is a substring of
the word
522
523         if fileObj.name == item:
524             self.searchResults = [fileObj] + self.searchResults
525         elif loc != -1: # If the search term is a substring of the current word
526             self.unsorted.append((loc, fileObj)) #Adds loc found in word, so that it can be sorted by
where it is found
527
528         if (fileObj.isDir and self.searchRecursively) and (fileObj.hexPath != self.recycleFolder) and
(fileObj.hexName != self.thumbsName):
529             self.findAndSortCore(fileObj.hexPath, item) # Search folder if recursive and not recycle
folder or thumbnail folder.
530
531 def findAndSort(self, item): #Main search function.
532     self.unsorted = []
533     self.findAndSortCore(self.currentDir, item)
534
535     if len(self.unsorted) > 0:
536         sorted = sortsCy.quickSortTuples(self.unsorted)
537         for i in sorted:
538             self.searchResults.append(i[1])
539         mainthread(self.removeButtons())
540         return mainthread(self.createButtons(self.searchResults, False))
541
542     elif len(self.searchResults) == 0:
543         pop = Popup(title="No Results", content=Label(text="No results found for:\n"+item,
544 align="center"), pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.4, .4))
544         pop.open()
545
546
547 def searchForItem(self, item):
548     self.resetButtons()
549     self.searchResults = []
550     Thread(target=self.findAndSort, args=(item,), daemon=True).start()
551
552
553 #####Progress Bar Information#####
554 def values(self, st): #Information for space left on device.
555     values = disk_usage(self.path) # Imported from shutil
556     if st:
557         return self.getGoodUnit(int(values[1]))+" / "+self.getGoodUnit(int(values[0])) + " used."
558     else:
559         return [values[0], values[1]]

```

```

560
561
562 #####Encryption Stuff + opening decrypted files#####
563     def passToPipe(self, type, d, targetLoc, newName=None, endOfFolderList=False, op=True):      #Passes
564         parameters to AES written in go.
565         if self.fileSep == "\\":
566             programe = "AESWin.exe"
567         else:
568             programe = "AES"
569
570         goproc = Popen(self.startDir+programe, stdin=PIPE, stdout=PIPE)
571         out, err = goproc.communicate((type+, "+d+", "+targetLoc+", "+self.key).encode()) # Send parameters to
572         AES
573         if err != None: # AES throws error when key is invalid.
574             raise ValueError("Key not valid.")
575
576         if endOfFolderList:
577             if self.encPop != None:
578                 self.encPop.dismiss()
579                 self.encPop = None
580             if type == "y":
581                 self.refreshFiles()
582                 print("Refreshing files.")
583
584         if type == "n" and op and endOfFolderList:
585             mainthread(self.openFileTh(targetLoc, d))
586         return out
587
588     def getCheckSum(self, location): # Communicates to BLAKE to get checksum.
589         if self.fileSep == "\\": # If on windows
590             goproc = Popen(self.startDir+"BLAKEWin.exe", stdin=PIPE, stdout=PIPE)
591         elif self.fileSep == "/":
592             goproc = Popen(self.startDir+"BLAKE", stdin=PIPE, stdout=PIPE)
593
594         out, err = goproc.communicate((location).encode())
595         if err != None:
596             raise ValueError(err)
597
598         return out.decode()
599
600     def getFileExtension(self, fileName):
601         return fileName.split(".")[-1].lower()
602
603     def isImage(self, fileName): # Used to get a file extension from a given file name.
604         extension = self.getFileExtension(fileName).lower()
605         return bool(extension == "png" or extension == "jpg")
606
607     def getThumbnail(self, fileObj):
608         if self.thumbsName not in self.currentDir: # Only check this when not in the thumbnail folder
609             if self.thumbsName not in os.listdir(self.currentDir): # Checks that there is a thumbnail folder in
610                 this directory.
611                 os.makedirs(self.currentDir+self.thumbsName)
612                 print("Made thumbnail directory since it wasn't there")
613
614         fileObj.thumbDir = self.currentDir+self.thumbsName+self.fileSep+fileObj.hexName
615         self.passToPipe("n", fileObj.hexPath, fileObj.thumbDir) # Decrypts thumbnail temporarily. Is deleted
616         once program is finished displaying it.
617         thumb = Image(source=fileObj.thumbDir)
618         return thumb
619
620
621     # Handles GUI while encrypting a single file, and parses parameters to passToPipe
622     def encDecTerminal(self, type, d, targetLoc, isPartOfFolder=False, endOfFolderList=False, newName=None,
623     op=True): # Handels passToPipe and UI while encryption/decryption happens.
624         fileName = ""
625         if type == "y": #The file name also needs to be encrypted
626             tempDir = d.split(self.fileSep)
627             fileName = tempDir[-1]
628             targetLoc = targetLoc.split(self.fileSep)

```

```

624     #replace file name with new hex
625     targetLoc[-1] = aesFName.encryptFileName(self.key, fileName)
626     thumbTarget =
627     self.fileSep.join(targetLoc[:-1])+self.fileSep+self.thumbsName+self.fileSep+targetLoc[-1]
628
629     popText = "Encrypting..."
630     targetLoc = self.fileSep.join(targetLoc)
631     if os.path.exists(targetLoc):
632         if os.path.isdir(targetLoc):
633             rmtree(targetLoc) # Imported from shutil
634         else:
635             os.remove(targetLoc)
636
637     elif type == "n":    #Need to decrypt file name if decrypting
638         tempDir = d.split(self.fileSep)
639         fileName = tempDir[-1]
640         if newName == None:
641             targetLoc = targetLoc.split(self.fileSep)
642             newName = targetLoc[-1] #Stops you from doing it twice in decrypt()
643             targetLoc = self.fileSep.join(targetLoc)
644             fileName = newName
645         popText = "Decrypting..."
646
647     if not isPartOfFolder:    # If it is a single file, then open a popup. If it isn't, then a popup already
exists.
648         self.encPop = mainSPops.encDecPop(self, type, popText, [d], [targetLoc], op=op) #self, labText, d,
newLoc, **kwargs
649         mainthread(Clock.schedule_once(self.encPop.open, -1)) # Open the popup as soon as possible
650
651     if len(fileName) <= 112: #Any bigger than this and the file name is too long (os throws the error).
652         self.encryptProcess = Thread(target=self.passToPipe, args=(type, d, targetLoc, newName,
endOfFolderList, op,), daemon=True)
653         self.encryptProcess.start()
654     else:
655         print("File name too long: ", fileName)
656         print("Dismissed?")
657         lab = Label(text="File name too long, skipping:\n"+fileName)
658         lab.bind(size=info.setter("text_size")) # Wrap to label.
659         pop = Popup(title="Invalid file name", content=lab, size_hint=(.4, .4), pos_hint={"x_center": .5,
"y_center": .5})
660         pop.open()
661
662     def openFileTh(self, fileLoc, startLoc):    # Creates a thread to open a file (stops program locking up)
663         Thread(target=self.openFile, args=(fileLoc, startLoc,), daemon=True).start()
664
665     def openFile(self, location, startLoc):
666         locationFolder = location.split(self.fileSep)
667         nameOfOriginal = locationFolder[-1]
668         locationFolder = self.fileSep.join(locationFolder[:-1])
669         startList = os.listdir(locationFolder)
670         if self.fileSep == "\\":
671             location = location.split("\\")

672             location = "/".join(location) # Windows actually accepts forward slashes in terminal
673             command = "cmd /k start """ +'+' +location+'+' +"/D"
674         else:
675             command = "xdg-open """ +location+"""" # Quotation marks for if the dir has spaces in it
676
677         startCheckSum = self.getCheckSum(location) # Gets checksum of file before opening.
678         os.system(command)# Using the same for both instead of os.startfile because os.startfile doesn't wait
for file to close
679         # After this line, the file has been closed.
680         if os.path.exists(locationFolder):          # If the vault is locked while the file is being edited,
then the temporary files get deleted, so check it still exists.
681             endList = set(os.listdir(locationFolder)) # Get list of temp files afterwards, and encrypt any new
ones (like doing save-as)
682             endCheckSum = self.getCheckSum(location)
683             print(startCheckSum, "START CHECK SUM") # For debugging
684             print(endCheckSum, "END CHECK SUM")
685         else:

```

```

685         endList = []
686         endCheckSum = startCheckSum # Don't try and encrypt files that have been removed.
687
688         diffAdded = [d for d in endList if d not in startList] # Creates an array of differences between the
689         list of files currently in the temp folder, and the original contents of the temp folder.
690         tempLoc = startLoc.split(self.fileSep)
691         for i in diffAdded: # Encrypt any extra files in the temp folder that were not there before
692             print("Difference found:", i)
693             tempLoc = self.fileSep.join(tempLoc[:-1]) # Remove last file name
694             tempLoc += self.fileSep+i
695             self.encDecTerminal("y", locationFolder+self.fileSep+i, tempLoc) #Is encrypted when program
closes anyway
696
697         if nameOfOriginal in endList:
698             print("Original still here")
699             if endCheckSum != startCheckSum:
700                 print("Original file has changed.")
701                 self.encDecTerminal("y", location, startLoc)
702
703     def onFileDrop(self, window, filePath): # For draging + dropping files into the window.
704         self.checkCanEncrypt(filePath.decode())
705         return "Done"
706
707     def decrypt(self, fileObj, op=True):
708         if not os.path.isdir(self.osTemp+"FileMate"+self.fileSep):
709             os.makedirs(self.osTemp+"FileMate"+self.fileSep)
710         fileLoc = self.osTemp+"FileMate"+self.fileSep+fileObj.name #Place in temporary files where it is going
to be stored.
711         if os.path.exists(fileLoc) and op: #Checks file exists already in temp files, so it doesn't have
to decrypt again.
712             self.openFileTh(fileLoc, fileObj.hexPath)
713         else:
714             self.encDecTerminal("n", fileObj.hexPath, fileLoc, newName=fileObj.name, op=op)
715
716     def checkDirExists(self, dir): #Handles UI for checking directory exits when file added.
717         if os.path.exists(dir):
718             return True
719         else:
720             self.popup = Popup(title="Invalid", content=Label(text="{} - Not a valid directory.".format(dir)), pos_hint=
{"center_x": .5, "center_y": .5}, size_hint=(.4, .4))
721             self.popup.open()
722             return False
723
724     def encDecDir(self, encType, d, targetLoc, op=True): # Encrypt and decrypt folders.
725         if self.encPop != None:
726             self.encPop.dismiss()
727             self.encPop = None
728
729         self fileList = []
730         self locList = []
731         self.encDecDirCore(encType, d, targetLoc)
732
733         labText = "Encrypting..."
734         if encType == "n":
735             labText = "Decrypting..."
736
737         self.encPop = mainSPops.encDecPop(self, encType, labText, self fileList, self locList, op=op) #self,
labText, fileList, locList, **kwargs
738         mainthread(Clock.schedule_once(self.encPop.open, -1))
739
740     def decryptFileToLoc(self, fileObj, button): # Decrypt a file/folder to a location (just handles the
input)
741         mainSPops.decryptFileToLocPop(self, fileObj).open()
742
743     def encDecDirCore(self, encType, d, targetLoc): # Enc/decrypts whole directory.
744         fs = os.listdir(d)
745         targetLoc = targetLoc.split(self.fileSep)
746         if encType == "y": # Decrypt folder names
747             targetLoc[-1] = aesFName.encryptFileName(self.key, targetLoc[-1])

```

```

747     else:
748         targetLoc[-1] = aesFName.decryptFileName(self.key, targetLoc[-1])
749     targetLoc = self.fileSep.join(targetLoc)
750     for item in fs:
751         if os.path.isdir(d+item):
752             self.encDecDirCore(encType, d+item+self.fileSep, targetLoc+self.fileSep+item) #Recursive
753         else:
754             if encType == "n":
755                 name = aesFName.decryptFileName(self.key, item)
756             elif encType == "y":
757                 name = aesFName.encryptFileName(self.key, item)
758             else:
759                 name = item
760             try:
761                 self.createFolders(targetLoc+self.fileSep)
762             except PermissionError:
763                 pass
764             else:
765                 self fileList.append(d+item)
766                 self locList.append(targetLoc+self.fileSep+name)
767
768     def checkCanEncryptCore(self, inp): # Used for adding new files to the vault by the user.
769         if self.checkDirExists(inp):
770             if os.path.isdir(inp):
771                 if inp[-1] != self.fileSep:
772                     inp += self.fileSep
773                 inpSplit = inp.split(self.fileSep)
774                 self.encDecDir("y", inp, self.currentDir+inpSplit[-2])
775             else:
776                 inpSplit = inp.split(self.fileSep)
777                 self.encDecTerminal("y", inp, self.currentDir+inpSplit[-1])
778
779     def checkCanEncrypt(self, inp): # Used for adding new files to the vault by the user.
780         if "--" in inp: # Multiple files/folders input.
781             inp = inp.split("--")
782             for d in inp:
783                 self.checkCanEncryptCore(d) # Actually encrypt/decrypt it.
784             else:
785                 self.checkCanEncryptCore(inp)
786
787             self.resetButtons()
788
789     def createFolders(self, targetLoc): # Create a folder safely.
790         if not os.path.exists(targetLoc):
791             os.makedirs(targetLoc)
792             if self.thumbsName not in targetLoc: # If in the thumbnails folder, don't make a thumbnails folder.
793                 os.makedirs(targetLoc+self.thumbsName)
794
795     def clearUpTempFiles(self): # Deletes temp files when the program is locked.
796         print("Deleting temp files.")
797         try:
798             rmtree(self.osTemp+"FileMate"+self.fileSep) # Imported from shutil
799         except:
800             print("No temp files.")
801
802
803

```

It has a fair amount of annotation, but I will still go through a few of the more confusing functions.

`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
124

```

The first three lines import the styling for the custom child widgets, which I will explain 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.outerScreen ascending = not self.outerScreen ascending
20         if self.outerScreen ascending:
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 self.ascending:
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 = " ^

```

```

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 `listButton`s 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
11 <nameSortButton@Button>:
12     font_size: 14
13     size_hint: 10.4, 1
14     on_release: root.changeSortOrder()
15
16 <sizeSortButton@Button>:
17     font_size: 14
18     on_release: root.changeSizeOrder()
19
20 <infoButton@Button>:
21     font_size: 14
22     size_hint: .05, 1
23     halign: "left"
24     valign: "middle"
25     on_release: root.outerScreen.getFileInfo(self.fileObj)
26     Image:
27         source: self.parent.outerScreen.assetsPath+"info.png"

```

```

28     center_x: self.parent.center_x
29     center_y: self.parent.center_y
30     size: 20, 20
31
32 <deleteButton@Button>:
33     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 | import aesFName
17 | from configOperations import dirInputValid
18 |
19 | class encDecPop(Popup): #For single files
20 |
21 |     def __init__(self, outerScreen, encType, labText, fileList, locList, op=True, **kwargs):
22 |         super(Popup, self).__init__(**kwargs)
23 |         self.outerScreen = outerScreen
24 |         self.fileList = fileList
25 |         self.locList = locList
26 |         self.done = False
27 |
28 |         # Kivy stuff
29 |         self.title = "Please wait..."
30 |         self.pos_hint = {"center_x": .5, "center_y": .5}
31 |         self.size_hint = (.7, .4)
32 |         self.auto_dismiss = False
33 |
34 |         self.grid = GridLayout(cols=1)
35 |         self.subGrid = GridLayout(cols=4)
36 |         self.currFile = Label(text="", halign="center", valign="center")
37 |         self.currFile.bind(size=self.currFile.setter("text_size")) # Wrap text inside label
38 |         self.per = Label(text="")
39 |         self.spd = Label(text="")
40 |         self.tim = Label(text="")
41 |         self.outOf = Label(text="")
42 |         self.pb = ProgressBar(value=0, max=os.path.getsize(self.fileList[0]), size_hint=(.9, .2))
43 |         self.wholePb = ProgressBar(value=0, max=self.__getTotalSize(), size_hint=(.9, .2))
44 |         self.grid.add_widget(Label(text=labText, size_hint=(1, .4)))
45 |         self.grid.add_widget(self.currFile)
46 |         self.subGrid.add_widget(self.per)
47 |         self.subGrid.add_widget(self.spd)
48 |         self.subGrid.add_widget(self.tim)
49 |         self.grid.add_widget(self.subGrid)
50 |         if len(self.fileList) > 1: # Don't bother showing 2 progress bars if the user is only doing 1 file.

```

```

51         self.grid.add_widget(self.pb)
52         self.subGrid.add_widget(self.outOf)
53     self.grid.add_widget(self.wholePb)
54     self.content = self.grid
55
56     self.checkThread = Thread(target=self.encDec, args=(encType, op,), daemon=True)
57     self.checkThread.start()
58
59     def __getTotalSize(self):
60         total = 0
61         for file in self.fileList:
62             total += os.path.getsize(file)
63         return total
64
65     def getGoodUnit(self, bps):
66         divCount = 0
67         divisions = {0: "B/s", 1: "KB/s", 2: "MB/s", 3: "GB/s", 4: "TB/s"}
68         while bps > 1000:
69             bps = bps/1000
70             divCount += 1
71
72         return ("%.2f" % bps) + divisions[divCount]
73
74     def __getGoodUnitTime(self, time):
75         divCount = 0
76         times = [(0.001, "Milliseconds"), (1, "Seconds"), (60, "Minutes"), (3600, "Hours"), (86400, "Days"),
77 (604800, "Weeks"), (2419200, "Months"), (31557600, "Years")] # 1 second, 1 minute, 1 hour, 1 day, 1 week, 1
month, 1 year in seconds
78         i = 0
79         while i < len(times): # Is broken when return is found
80             if time > times[i][0]:
81                 i += 1
82             else:
83                 return ("%.2f" % float(time/times[i-1][0])) + " " + times[i-1][1] + " left"
84
85         return "A lot of time left."
86
87     def __getRelPathDec(self, path): # Similar to decryptRelPath in fileClass
88         splitPath = (path.replace(self.outerScreen.path, "")).split(self.outerScreen.fileSep)
89         return "/Vault/" + self.outerScreen.fileSep.join([aesFName.decryptFileName(self.outerScreen.key, i) for i
in splitPath])
90
91     def encDec(self, encType, op):
92         total = 0
93         totalPer = 0
94         factor = 0.5
95         timeLast = 0
96         lastSize = 0
97         timeDelta = 0
98         perDelta = 0
99         per = 0
100        prevPer = 0
101        for i in range(len(self.fileList)):
102            done = False
103            self.pb.value = 0
104            self.pb.max = os.path.getsize(self.fileList[i])
105            if i == len(self.fileList)-1:
106                self.outerScreen.encDecTerminal(encType, self.fileList[i], self.locList[i], True, True, op=op)
107            else:
108                self.outerScreen.encDecTerminal(encType, self.fileList[i], self.locList[i], True, op=op)
109
110            self.outOf.text = str(i)+"/"+str(len(self.fileList))
111            if encType == "n":
112                self.currFile.text = self.__getRelPathDec(self.fileList[i])
113            else:
114                self.currFile.text = self.fileList[i]
115
116            while not done: # Padding can cause issues as original size is not known.
117                if os.path.exists(self.locList[i]):
```

```

117     self.pb.value = os.path.getsize(self.locList[i])
118     self.wholePb.value = total + self.pb.value
119     per = self.wholePb.value_normalized*100
120
121     a = time()    # Temporary variable to hold the time
122     timeDelta = a - timeLast      # Get time difference
123     if timeDelta >= 0.5: # Update every 0.5 seconds
124         perDelta = per - prevPer   # Change in percentage in that time.
125         timeLast = a
126         sizeDelta = self.wholePb.value - lastSize  # Get change in size of the file being
127         encrypted
128         speed = sizeDelta/timeDelta # Get speed of encryption in bytes/second
129
130         if speed != 0:
131             self.tim.text = self.__getGoodUnitTime((self.wholePb.max -
132             self.wholePb.value)/speed)
133             self.spd.text = self.getGoodUnit(speed)
134
135         lastSize = self.wholePb.value
136         prevPer = per
137
138         self.per.text = "{0:.2f}%".format(per)
139
140         if self.pb.value >= self.pb.max-32 or self.pb.value_normalized == 1: # -32 is due to padding.
141             done = True
142         else:
143             sleep(0.005) # Reduces the rate the file is checked, so python doesn't use too much CPU.
144             AES will still run the same regardless, the file just doesn't need to be checked as soon as possible.
145
146         self.pb.value = self.pb.max
147         totalPer += 100
148         total += self.pb.max
149
150     self.dismiss()
151
152 class btTransferPop(encDecPop):
153
154     def __init__(self, mainScreen, fileObjTmp, **kwargs):
155         super(Popup, self).__init__(**kwargs)
156         self.outerScreen = mainScreen
157         self.title = "Please wait..."
158         self.size_hint = (.7, .4)
159         self.pos_hint = {"center_x": .5, "center_y": .5}
160         self.auto_dismiss = False
161         self.grid = GridLayout(cols=1)
162         self.subGrid = GridLayout(cols=3)
163         self.currFile = Label(text=fileObjTmp.path)
164         self.per = Label(text="")
165         self.spd = Label(text="")
166         self.tim = Label(text="")
167         self.pb = ProgressBar(value=0, max=1, size_hint=(.9, .2))
168         self.grid.add_widget(Label(text="Sending..."))
169         self.grid.add_widget(self.currFile)
170         self.subGrid.add_widget(self.per)
171         self.subGrid.add_widget(self.spd)
172         self.subGrid.add_widget(self.tim)
173         self.grid.add_widget(self.subGrid)
174         self.grid.add_widget(self.pb)
175         self.content = self.grid
176
177         self.sendThread = Thread(target=self.sendFile, args=(fileObjTmp,), daemon=True) # can be cancelled mid
178         way through
179         self.sendThread.start()
180
181     def sendFile(self, fileObj):
182         # File name is sent with !NAME#!!!<name here>!~
183         # File data is sent right afterwards, ending with ~!!ENDF!
184         # Overall, it is sent as: !NAME#!!!<name here>!~<datahere>~!!ENDF!
```

```

182     self.outerScreen.clientSock.send("!NAME!{}~~!~~".format(fileObj.name))
183     #print("!NAME!{}~~!~~".format(fileObj.name), "Sent")
184
185     newLoc = self.outerScreen.osTemp+"FileMate"+self.outerScreen.fileSep+fileObj.name
186     if not os.path.isdir(self.outerScreen.osTemp+"FileMate"+self.outerScreen.fileSep):
187         os.makedirs(self.outerScreen.osTemp+"FileMate"+self.outerScreen.fileSep)
188
189     self.outerScreen.passToPipe("\n", fileObj.hexPath, newLoc, fileObj.name, op=False) #self, type, d,
targetLoc, newName=None, endOffolderList=False
190
191     bufferSize = 1024
192     buff = []
193     fr = open(newLoc, "rb")
194     buff = fr.read(bufferSize) #Read 1Kb of data
195     buffCount = 0
196     self.per.text = "{0:.2f}%".format(0)
197
198     start = time()
199     #Send data
200     while buff:
201         self.outerScreen.clientSock.send(buff)
202         buffCount += bufferSize
203         buff = fr.read(bufferSize)
204
205         self.pb.value = buffCount/fileObj.rawSize
206         self.per.text = "{0:.2f}%".format(self.pb.value*100)
207         self.spd.text = self.getGoodUnit(buffCount/(time() - start))
208
209     self.outerScreen.clientSock.send("~!ENDFILE!")
210     self.dismiss()
211
212
213 class decryptFileDialog(Popup): # Input box for location of where directory is to be saved.
214
215     def __init__(self, mainScreen, fileObj, **kwargs):
216         self.outerScreen = mainScreen
217         self.fileObj = fileObj
218         super(Popup, self).__init__(**kwargs)
219
220     def checkCanDec(self, inp):
221         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.
222         if self.fileObj.isDir:
223             if not os.path.exists(inp):
224                 os.makedirs(inp)
225             if inp[-1] != self.outerScreen.fileSep: inp += self.outerScreen.fileSep
226             self.outerScreen.encDecDir("\n", self.fileObj.hexPath, inp, op=False)
227         else:
228             if inp[-1] == self.outerScreen.fileSep: # If ends with "/", then decrypt with it's file name.
229                 if not os.path.exists(inp):
230                     os.makedirs(inp)
231                 inp += self.fileObj.name
232             self.outerScreen.encDecTerminal("\n", self.fileObj.hexPath, inp, op=False)
233
234     def getTitle(self):
235         if self.fileObj.isDir:
236             return "Decrypt Folder"
237         else:
238             return "Decrypt File"
239
240
241 class addNewFolderPop(Popup):
242
243     def __init__(self, mainScreen, **kwargs):
244         super(Popup, self).__init__(**kwargs)
245         self.outerScreen = mainScreen
246
247     def makeFolder(self, text):
248         if dirInputValid(self.outerScreen.currentDir+text, self.outerScreen.fileSep):

```

```

249     try:
250         os.makedirs(self.outerScreen.currentDir+aesFName.encryptFileName(self.outerScreen.key, text))
251     except OSError as e:
252         if "[Errno 36]" in str(e): #OSError doesn't store the error code for some reason.
253             pop = Popup(title="Invalid Folder Name", content=Label(text="Folder name too long."),
254                         halign="center"), size_hint=(.3, .3), pos_hint={"x_center": .5, "y_center": .5})
255             pop.open()
256
257         self.outerScreen.refreshFiles()
258         self.dismiss()
259
260     class addFilePop(Popup):      #The screen (it's actually a Popup) for adding folders/files to the vault.
261
262     def __init__(self, mainScreen, **kwargs):
263         super(Popup, self).__init__(**kwargs)
264         self.outerScreen = mainScreen
265
266     class ConfirmationPopup(Popup):      #Popup for confirming encryption.
267
268     def __init__(self, fileScreen, input, **kwargs):
269         super(Popup, self).__init__(**kwargs)
270         self.fileScreen = fileScreen
271         self.inputText = input
272
273     def checkIfSure(self, input):
274         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)), calling `self.outerScreen.encDecTerminal(<parameters>)`.
3. The maximum value of the status bar is set based on the size of the original file.
4. While the file is enc/decrypting, its size is monitored using `os.path.getsize(<file>)`, and the 1 or 2 progress bars' (depending on if multiple files are being operated on) values are updated with the new size.
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 total amount of data left by the speed (`(self.wholePb.max - self.wholePb.value)/speed`).
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 loop sleeps for 0.005 seconds to not max out the CPU, as it doesn't need to be updated that often (reduces usage to about 5%).

`btTransferPop` does works 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 <ConfirmationPopup@Popup>:
4     title: "Confirmation"
5     size_hint: .4, .4
6     pos_hint: {"center_x": .5, "center_y": .5}
7     auto_dismiss: False
8     FloatLayout:
9         Label:
10            text: "Are you sure?"
11            pos_hint: {"center_x": .5, "center_y": .7}
12         Button:
13             text: "No!!!"
14             pos_hint: {"center_x": .25, "center_y": .25}
15             size_hint: .4, .3
16             on_release: Clock.schedule_once(root.dismiss, -1)
17
18         Button:
19             text: "Yes"
20             pos_hint: {"center_x": .75, "center_y": .25}
21             size_hint: .4, .3
22             on_release: Clock.schedule_once(root.dismiss,
23 -1);root.fileScreen.outerScreen.checkCanEncrypt(root.inputText)
24
25 <addNewFolderPop@Popup>:
26     title: "Add New Folder"
27     size_hint: .7, .4
28     pos_hint: {"center_x": .5, "center_y": .5}
29     auto_dismiss: True
30     GridLayout:
31         cols: 1
32         Label:
33             text: "New Folder Name:"
34         TextInput:
35             id: folderNameInput
36             size_hint: .7, .4
37             multiline: False
38             hint_text: "Name"
39             on_text_validate: root.makeFolder(self.text)
40
41 <decryptFileToLocPop@Popup>:
42     title: root.getTitle()
43     size_hint: .7, .4
44     pos_hint: {"center_x": .5, "center_y": .5}
45     auto_dismiss: True
46     GridLayout:
47         cols: 1
48         Label:
49             text: "Input the destination:"
50         TextInput:
51             id: decDirInput
52             size_hint: .7, .4
53             multiline: False
54             hint_text: "Directory:"
55             on_text_validate: root.checkCanDec(self.text)
56
57 <addFilePop@Popup>:
58     id: addFile
59     submitDirs: submitDirs
60     size_hint: .7, .7
61     title: "Add File"
```

```

62     FloatLayout:
63         Label:
64             size_hint: .46, .08
65             font_size: 18
66             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."
67             pos_hint: {"center_x": 0.5, "y": 0.8}
68
69         Button:
70             size_hint: .16, .16
71             pos_hint: {"center_x": .5, "center_y": .2}
72             text: "Close"
73             on_release: root.dismiss()
74
75         TextInput:
76             size_hint: .9, .12
77             font_size: 22
78             hint_text: "Directories"
79             id: dirInp
80             pos_hint: {"center_x": 0.5, "center_y": 0.7}
81             multiline: False
82
83         Button:
84             id: submitDirs
85             size_hint: .16, .16
86             pos_hint: {"center_x": .5, "center_y": .4}
87             text: "Submit"
88             on_release: root.checkIfSure(root.ids.dirInp.text)
89

```

## 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  from configOperations import changeVaultLoc, editConfTerm
5
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[len(inp)-1] != self.outerScreen.fileSep:
27            inp += self.outerScreen.fileSep
28        try:
29            worked = changeVaultLoc(inp, self.outerScreen.fileSep, self.config)
30        except FileNotFoundError:
31            Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Directory not valid:\n"+inp),
size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()
32        except PermissionError as e:
33            Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Can't make a folder here:\n"+inp),
size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()
34        except Exception as e:
35            print(e)
36            Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Can't make a folder here:\n"+inp),
size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5}).open()
37        else:
38            if worked:
39                done = Popup(title="Done", content=self.outerScreen.infoLabel(text="Changed Vault Location
to:\n"+inp), size_hint=(.4, .4), pos_hint={"x_center": .5, "y_center": .5})
40                self.outerScreen.path = inp
41                self.outerScreen.currentDir = inp
42                self.outerScreen.recycleFolder =
self.outerScreen.path+self.outerScreen.recycleName+self.outerScreen.fileSep
43                done.open()
44            else:
45                Popup(title="Invalid", content=self.outerScreen.infoLabel(text="Directory not valid:\n"+inp),
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      # Approximately 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 is
restarted."
70         text_size: self.size
71     Label:
72
73     Button:
74         text: "Exit"
75         font_size: 20
76
77         on_release: root.manager.current = "Main"
78
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             Button:
66                 text: "Delete"
67                 on_release: root.backSpace()
68             Button:
69                 text: "0"
70                 on_release: root.addNum(self.text)
71             Button:
72                 text: "Submit"
73                 on_release: root.confirm()
74
75             Label:
76                 id: display
77                 text: root.numsString
78                 font_size: Window.height/20
79                 pos_hint: {"center_x": .5, "y": .35}
80
81 <MainScreen>:
82     FloatLayout:
83         Label:
84             text: "To lock the Vault, close the program.\nAll downloaded files are stored in\nyour 'Download' folder."

```

```

86     Button:
87         pos_hint: {"x": .25, "y": .2}
88         size_hint: .5, .2
89         text: "Select files from PC"
90         on_release: root.manager.current = "Select"
91
92 <listButton@Button>:
93     size_hint: 1, None
94     on_release: root.outerScreen.selectFile(self.fileName)
95
96 <FileSelectionScreen>:
97     FloatLayout:
98         Button:
99             text: "Exit"
100            size_hint: .4, .14
101            background_color: (0.8, 0.8, 0.8, 1)
102            pos_hint: {"top": 1, "left": 1}
103            on_release: root.exit()
104
105         Button:
106             text: "Go back"
107             size_hint: .4, .14
108             background_color: (0.8, 0.8, 0.8, 1)
109             pos_hint: {"top": 1, "right": 1}
110             on_release: root.goBackDir()
111

```

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

```

```

29         sStream = socket.getOutputStream() #Stream for sending data
30         self.devName = devName
31         found = True
32         break #Stop when device found
33     if found:
34         socket.connect()
35         return rStream, sStream
36     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\nnis on, you are in\nrange 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
77                 self.view = ScrollView(size_hint=(1, 1))
78                 self.view.add_widget(self.layout)
79                 self.content = self.view
80
81             def getDeviceList(self):
82                 result = []
83                 pairedDevs = BluetoothAdapter.getDefaultAdapter().getBondedDevices().toArray()
84                 for dev in pairedDevs:
85                     result.append(dev.getName()) # Get the names of each device.
86
87             return result
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    def addNum(self, num):
122        if len(self.nums) < 16:      # Maximum length of key is 16
123            self.nums.append(int(num))
124            self.numsString += "*"
125            self.updateDisplay()
126
127    def updateDisplay(self):      # Updates the Label at the top of PadScreen
128        self.ids.display.text = self.numsString  # self.numsString just contains asterisks "*"
129
130    def backSpace(self):        # For deleting the input
131        if len(self.nums) != 0:
132            del self.nums[-1]
133            self.numsString = self.numsString[:len(self.nums)]
134            self.updateDisplay()
135
136    def confirm(self):
137        pop = Popup(title="Please Wait...", content=Label(text="Waiting for confirmation."), size_hint=(1, 1),
138        pos_hint={"x_center": .5, "y_center": .5}, auto_dismiss=False)
139        if self.rStream != None and self.sStream != None: # if the connection is still up
140            self.sStream.write("{}".format("#"))    # Key sent as #<key>-
141            self.nums = SHA.getSHA128of16(self.nums)
142            for num in self.nums:
143                self.sStream.write("{},".format(num))
144            self.sStream.write("{}.".format("-"))
145            self.sStream.flush()
146            print u"Numbers sent."
147            pop.open()
148
149            data = self.rStream.read()
150            while len(str(data)) == 0:
151                try:
152                    data = self.rStream.read()
153                except Exception as e:
154                    print e, u"Couldn't receive data."
155
156            print u"Out of while loop"
157            print data, u"Response"
158            if data == 49:  # Response sent by the PC if the key is valid.
159                pop.dismiss()
160                print u"Valid"

```

```

161         corPop = Popup(title="Valid.", content=Label(text="Valid passcode!\nPlease leave the app open
162 in the background\notherwise the vault will lock."), size_hint=(.9, .5), pos_hint={"x_center": .5, "y_center": .5})
163         Clock.schedule_once(corPop.open, -1)
164
165         # Time to receive file names of current directory
166         listOffFiles = recieveFileList(self.rStream)
167
168         self.manager.get_screen("Main").sStream, self.manager.get_screen("Main").rStream =
169         self.sStream, self.rStream # Hand over the streams to the other screens
170
171         self.manager.get_screen("Select").sStream, self.manager.get_screen("Select").rStream =
172         self.sStream, self.rStream
173         self.manager.get_screen("Select").fileList = listOffFiles
174
175         self.manager.current = "Main"
176
177         elif data == 48: # Response sent by the PC if code is invalid.
178             print u"Invalid."
179             pop.dismiss()
180             invPop = Popup(title="Invalid.", content=Label(text="Invalid passcode, please try again."),
181             size_hint=(.9, .5), pos_hint={"x_center": .5, "y_center": .5})
182             self.nums = []
183             self.numsString = u""
184             self.updateDisplay()
185             invPop.open()
186
187         else:
188             print type(data), "data was not either 49 or 48..."
189
190         else:
191             print u"Can't connect to device."
192             Popup(self, title="Can't connect.",
193                 content=Label(text="Can't connect to device\nplease make sure the\ndevice has Bluetooth
194                 on,\nis in range, and is\nrunning the FileMate program."),
195                 title_align="center",
196                 size_hint=(.6, .6),
197                 pos_hint={"x_center": .5, "y_center": .5},
198                 auto_dismiss=True).open()
199
200         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 receive both the list of files and to download files.

Here is the code for receiving a list of files, and for receiving 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 actual list of files from the buffer
23
24     listOfFile = "".join([chr(i) for i in buff]) # Join input into a string
25     listOfFile = listOfFile.split("--")[1:] # First element will be "" due to first part of string being "--"
26
27     print "List of files given:", listOfFile
28     return listOfFile
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 receiving stream
52         except Exception as e:
```

```

53     print e, u"Failed receiving 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
name every time a new element is added to buffer, while the name has not been found.
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
been received.
71             nameFound = True # Name has been found
72             buff[i:z+5] = [] # Clear name + separator
73
74         for letter in name:
75             fileName += chr(letter)
76
77         fo = open(downloadsDir+"/"+fileName, "wb") # Open for writing
78
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
file in\nyour 'Download' folder."), pos_hint={"x_center": .5, "y_center": .5}, size_hint=(.7, .4))
88             pop.open()
89             return True
90
91     else:
92         fo.write(bytarray(buff[:bufferSize]))
93         buff[:bufferSize] = []
94         buffCount += bufferSize

```

`receiveFileList` 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`.

`receiveFile` 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):
13
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
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
51          # future (file size etc)
52          self.grid.bind(minimum_height=self.grid.setter("height"))
53          for item in array:
54              btn = self.listButton(self, item, text=( " "+str(item)), height=Window.height/10, halign="left",
55              valign="middle")
56              btn.bind(size=btn.setter("text_size"))
57              self.grid.add_widget(btn)
58
59          self.scroll = ScrollView(size_hint=(1, .86))
60          self.scroll.add_widget(self.grid)
61          self.add_widget(self.scroll)
62
63      def recreateButtons(self, array):
64          self.removeButtons()
65          self.createButtons(array)
66
67      def selectFile(self, fileName):
68          # File request looks like: !FILESELECT!<name here>~!ENDSELECT!
69          msg = [33, 70, 73, 76, 69, 83, 69, 76, 69, 67, 84, 33] # !FILESELECT!
70
71          for letter in fileName:
72              msg.append(ord(letter))

```

```

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 recieving 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 recieve 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    def getBackDir(self):
117        # Back dir request looks like: !BACK!
118        backCommand = [33, 66, 65, 67, 75, 33]
119        self.sStream.flush() # Clear the buffer for sending
120        for i in backCommand:
121            self.sStream.write(i)
122
123        data = ""
124        buff = []
125        responseFound = False
126        while not responseFound:
127            try:
128                data = self.rStream.read()
129            except Exception as e:
130                print e, "Failed recieving server response to BACK request."
131                return False
132            else:
133                buff.append(data)
134
135        if (buff[:11] == self.endOfTreeResponse) and (len(buff) >= 11): # If you cannot go further back
136            in the directory, !ENDOFTREE! is sent from the PC

```

```

137         print "END OF TREE"
138         buff = []
139         responseFound = True
140
141     elif (buff[:10] == self.startList) and (len(buff) >= 10):    # Otherwise, it should be a list of new
file names
142         responseFound = True
143         self.fileList = receiveFileList(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 # (list) Garden requirements
32 #garden_requirements =
33
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 white-box testing of important functions.

The key for the different types of data will be:

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