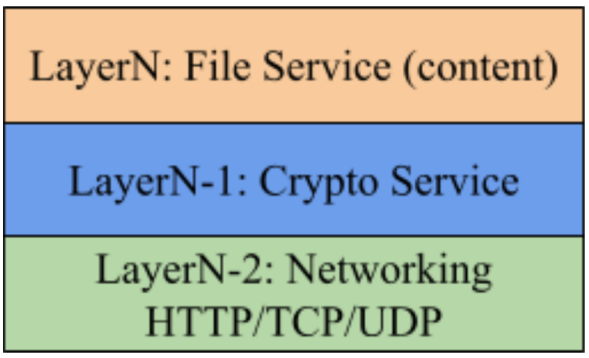
The code is well commented for all the lines and sections and is self explanatory in its terms. We have also made a text file on how to compile and execute our code (HowToCompileAndExecute.txt).

We create the client and server socket using the TCP protocol. Since it is more reliable for the correct delivery of data.

Mode of layering:



We have implemented a 3-layered architecture (File content, Crypt layer, Networking layer) while writing the code of client and server itself. In a layered architecture, the services of nth layer depend on the n-1th layer. This is the case as our code runs in a line-by-line manner.

Take the example of the upd command. The code section for this on client side looks like:

```
97          #Upload the specified file on client to the remote server in CWD
98          elif(command[:3] == "upd"):
99              filename = command[4:]
100             print(filename)
101             file = open(filename, "r")
102             data1 = file.read() #read the data from file
103             # print(data1)
104             # print("I am encrypting the data")
105             data_encrypted = encrypt(data1, offset) #encrypt the data
106             # print(data_encrypted)
107             s.sendall(bytes(data_encrypted, 'utf-8')) #send the encrypted data
```

Here you can see that we first invoke the LayerN by opening the file and reading contents from it.
After which, the LayerN-1 comes into play when we encrypt the data.
Finally, we exert the LayerN-2 when we send the data.

In a similar manner, while working on the sever side, the layers unfold in a bottom up manner which can be interpreted from the code below:

```
114             #Upload the specified file on client to the remote server in CWD
115             elif(data[:3]=="upd"):
116                 file_data = conn.recv(1024).decode()
117                 if not file_data:
118                     print("No data recieved. Status: NOK")
119                     break
120                 filename = data[4:]
121                 # print("the name of the file recieved by the server is: ", filename)
122                 file = open(filename, 'w')
123                 data_decrypted = encrypt(file_data, -2)
124                 file.write(data_decrypted)
125                 file.close()
```

Herein, we first receive the data which is a functionality of the Networking layer. We then decrypt the received data using the crypt layer and finally the data is written into the file using the services of the File service layer.

Some caveats associated with the ‘offset’ variable used in the crypt layer.

- For Plain text and Transpose mode you can pass the same value of the offset on both the client and the server side, i.e., 0 and “rev” respectively. But, for the Caesar cipher mode if you encrypt on one side using the offset = 2 then on the other side you need to decrypt it using the offset = -2.

For the execution, we will paste screenshots of the successful results for the execution of five commands.

```
(base) someshpratapsingh@SOMESHs-MacBook-Air SourceCode % python3 client.py
cwd
/Users/someshpratapsingh/Desktop/19110206_CS433_A1/Q1/SourceCode
ls
['server.py', '.DS_Store', 'client.py', 'checkupd.txt']
cd /Users/someshpratapsingh/Desktop/
Directory successfuly changed. Status: OK
dwd checkdwd.txt
File successfuly downloaded. Status: OK
cd /Users/someshpratapsingh/Desktop/19110206_CS433_A1/Q1/SourceCode
Directory successfuly changed. Status: OK
ls
['server.py', '.DS_Store', 'client.py', 'checkdwd.txt', 'checkupd.txt']
cd /Users/someshpratapsingh/Desktop/
Directory successfuly changed. Status: OK
ls
['Paper', 'firstcap.pcapng', 'assignment-1-osathire', 'Screenshots', '.DS_Store', '.localized', 'checkdwd.txt', 'hii-1', 'CNA1', '
Miami', '.git', '19110206_CS433_A1']
upd checkupd.txt
Go to server's terminal window
stop
Program successfuly stopped
(base) someshpratapsingh@SOMESHs-MacBook-Air SourceCode %
```

```
(base) someshpratapsingh@SOMESHs-MacBook-Air SourceCode % python3 server.py
Hello, I am a server and I am waiting for a client to connect with me.
Client came and is now connected by  {('127.0.0.1', 60020)}
File successfuly uploaded. Status: OK
Program successfuly stopped
(base) someshpratapsingh@SOMESHs-MacBook-Air SourceCode %
```

This is done for offset = “rev”. Similarly, the command can be run with other crypt modes with different offset values, as prescribed above.

We have also saved the terminal output files (for both client and the server) in the same folder as this design doc which can referred to see what happened sequentially on the terminal window as the commands were executed.