Hrushti Naik
19110088

Computer Networks
Assignment 1: Part 1

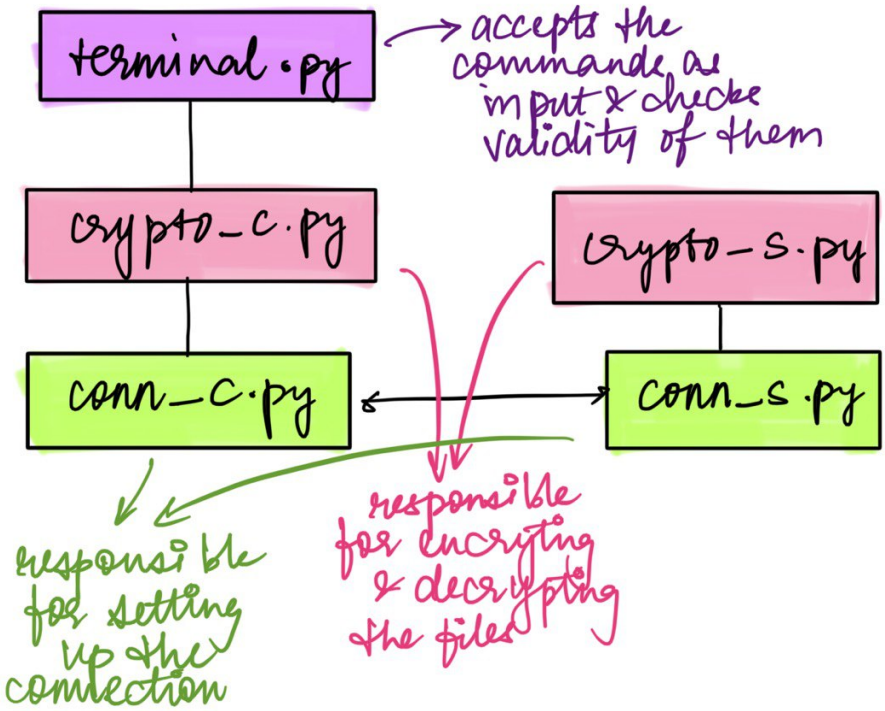
5. Network Application using Socket Programming

a. ***Source code (preferably on github) and means to compile and execute your code.***
The link to github repo is given here: https://github.com/Hrushti/cn_a1
The steps to compile and execute the code are written in the .readme file.

b. ***Design document detailing the mode of layering, protocol design for each of the layers, associated challenges and screenshots for the execution of the 5 commands.***

My understanding of the problem:
There will be two sides, client-side and server-side. Practically, these should be on different machines, but for the sake of this assignment, different machines are abstracted by different folders. The application will resemble a terminal, supporting the five commands mentioned. Using cwd, ls, and cd will control and give information about the *server-side* files, not the client-side files. By navigating to different folders in the server remotely, we can choose which files to download from where and which files to upload to what location on the server. However, the files downloaded will only be present in one particular location in the client's system, and only the files in this location can be uploaded to the server. When downloading/uploading files, the client can choose to encrypt them using the three options mentioned.

To model this, three layers are needed: file service/terminal (application layer), encryption layer, and networking (transport layer). Initially, there were 5 files present:



However, since there is no equivalent file of terminal.py on the server side, terminal.py and conn_c.py were merged to preserve the layer integrity.
Thus, only four files are present.

The in-depth function and methodology of each file are written as comments in the files.

To illustrate the working of the code, we will check all five commands:

1. **Make connection:**
    a. **Run conn_s.py:**

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\
assignments\cn_a1\p1\client>
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\
cn\assignments\cn_a1\p1\server> python c
onn_s.py
socket created
socket binded to 4000
socket is listening
```

b. Run conn_c.py:

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\client> python conn_c.py
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\server> python conn_s.py
socket created
socket binded to 4000
socket is listening
connection to: ('127.0.0.1', 56791)
```

2. cwd:

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\client> python conn_c.py
cwd
C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\server
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\server> python conn_s.py
socket created
socket binded to 4000
socket is listening
connection to: ('127.0.0.1', 56791)
```

3. cd and ls (navigating to the data folder and listing the files present):

```
ASSIGNMENTS
cn_a1
p1
client
__pycache__
conn_c.py
crypto_c.py
data
alice_orig.txt
alice.txt
server
__pycache__
conn_s.py
crypto_s.py
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\client> python conn_c.py
cwd
C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\server

cd ..
OK

cd data
OK

ls
['alice.txt', 'alice_orig.txt']
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\assignments\cn_a1\p1\server> python conn_s.py
socket created
socket binded to 4000
socket is listening
connection to: ('127.0.0.1', 56791)
```

4. downloading alice.txt from server to client using transpose encoding:

While using upd and dwd, the encryption mode needs to be given in the following format:

-pt for plain text, -sb for substitute encoding, and -tp for transpose encoding.

It needs to be given after the file name, so the commands look like these:

[upd/dwd] [filename] -[pt/sb/tp]

If no encryption mode is given, it defaults to plain text encoding.

```
ASSIGN...
cn_a1
p1
client
__pycache__
conn_c.py
crypto_c.py
alice.txt U
data
alice_orig.txt
alice.txt
```

```
assignments\cn_a1\p1\client> python conn_c.
py
cd ..
OK

cd data
OK

ls
['alice.txt', 'alice_orig.txt']

dwd alice.txt -tp
OK
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\
cn\assignments\cn_a1\p1\server> python c
onn_s.py
socket created
socket binded to 4000
socket is listening
connection to: ('127.0.0.1', 56865)
```

5. **Navigating to the server folder and uploading alice.txt from the client folder using substitution encoding:**

```
ASSIGNMENTS
p1
client
__pycache__
conn_c.py
crypto_c.py
alice.txt U
data
alice_orig.txt
alice.txt
server
__pycache__
conn_s.py
crypto_s.py
alice.txt U
```

```
ls
['alice.txt', 'alice_orig.txt']

dwd alice.txt -tp
OK
cd ..
OK

cd server
OK

cwd
C:\Users\hrush\Documents\Acads\Sem_7\cn\ass
ignments\cn_a1\p1\server

upd alice.txt -sb
OK
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\
cn\assignments\cn_a1\p1\server> python c
onn_s.py
socket created
socket binded to 4000
socket is listening
connection to: ('127.0.0.1', 56865)
```

6. **In case of a wrong command, and exit:**

```
ignments\cn_a1\p1\server

upd alice.txt -sb
OK

jargon
invalid command, type exit to quit
exit
PS C:\Users\hrush\Documents\Acads\Sem_7\cn\
assignments\cn_a1\p1\client>
```

```
PS C:\Users\hrush\Documents\Acads\Sem_7\
cn\assignments\cn_a1\p1\server>
```

Challenges faced:

The main challenge I faced was with file paths and designing the code flow of the application. Making the connection via sockets and listing all five cases was simple enough. Even encrypting the files was simple once I got the idea; its job is to mainly change the file's contents depending on the cypher mentioned. crypto_c.py and crypto_s.py are essentially the same.

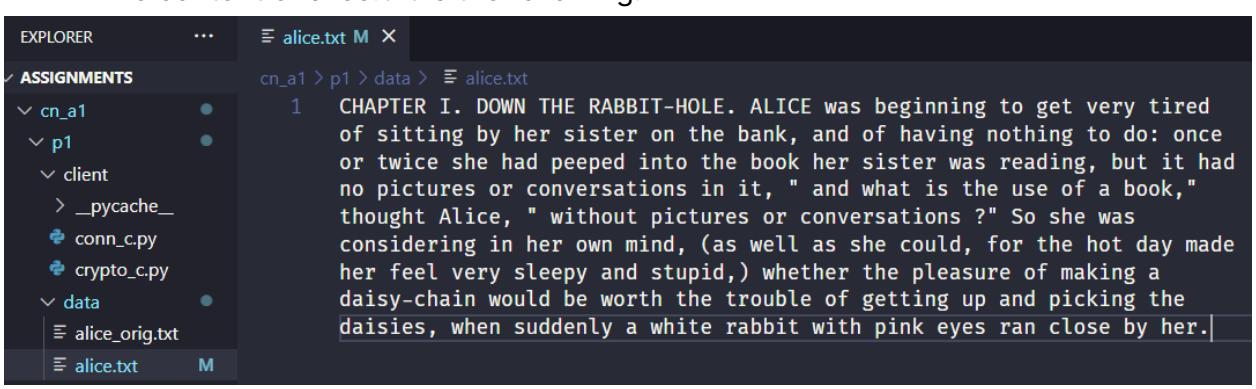
The first design only had support for any one of the encryption methods, and figuring out how to pass encryption modes was a challenge.
I also had difficulty figuring out the problem statement, but that cleared up over time.

c. *Wireshark dump and analysis indicating data was correctly encrypted.*

The commands were run on Kali, with Wireshark capturing the transfers.
When alice.txt was downloaded from the server using transpose encoding, this was the data that was transferred:

```
0010  45 00 02 ab 96 5e 40 00  40 06 a3 ec 7f 00 00 01   E····^@· @·******
0020  7f 00 00 01 d2 40 0f cb  59 08 96 b8 49 06 f2 fa   ·····@·· Y···I···
0030  80 18 02 00 00 a0 00 00  01 01 08 0a d0 49 0e b0   ******** *****I··
0040  d0 49 0e b0 52 45 54 50  41 48 43 20 2e 49 20 4e   ·I··RETP AHC .I N
0050  57 4f 44 20 45 48 54 20  2e 45 4c 4f 48 2d 54 49   WOD EHT  .ELOH-TI
0060  42 42 41 52 20 45 43 49  4c 41 20 73 61 77 20 67   BBAR ECI LA saw g
0070  6e 69 6e 6e 69 67 65 62  20 6f 74 20 74 65 67 20   ninnigeb  ot teg
0080  79 72 65 76 20 64 65 72  69 74 20 66 6f 20 67 6e   yrev der it fo gn
0090  69 74 74 69 73 20 79 62  20 72 65 68 20 72 65 74   ittis yb  reh ret
00a0  73 69 73 20 6e 6f 20 65  68 74 20 2c 6b 6e 61 62   sis no e ht ,knab
00b0  20 64 6e 61 20 66 6f 20  67 6e 69 76 61 68 20 67    dna fo  gnivah g
00c0  6e 69 68 74 6f 6e 20 6f  74 20 3a 6f 64 20 65 63   nihton o t :od ec
00d0  6e 6f 20 72 6f 20 65 63  69 77 74 20 65 68 73 20   no ro ec iwt ehs
00e0  64 61 68 20 64 65 70 65  65 70 20 6f 74 6e 69 20   dah depe ep otni
00f0  65 68 74 20 6b 6f 6f 62  20 72 65 68 20 72 65 74   eht koob  reh ret
0100  73 69 73 20 73 61 77 20  2c 67 6e 69 64 61 65 72   sis saw  ,gnidaer
0110  20 74 75 62 20 74 69 20  64 61 68 20 6f 6e 20 73    tub ti  dah on s
0120  65 72 75 74 63 69 70 20  72 6f 20 73 6e 6f 69 74   erutcip  ro snoit
0130  61 73 72 65 76 6e 6f 63  20 6e 69 20 2c 74 69 20   asrevnoc  ni ,ti
0140  22 20 64 6e 61 20 74 61  68 77 20 73 69 20 65 68   " dna ta hw si eh
0150  74 20 65 73 75 20 66 6f  20 61 20 22 2c 6b 6f 6f   t esu fo  a ",koo
0160  62 20 74 68 67 75 6f 68  74 20 2c 65 63 69 6c 41   b thguoh t ,ecilA
0170  20 22 20 74 75 6f 68 74  69 77 20 73 65 72 75 74    " tuoht iw serut
0180  63 69 70 20 72 6f 20 73  6e 6f 69 74 61 73 72 65   cip ro s noitasre
```

The content of alice.txt is the following:



In the right side of the first screenshot, we can see that the words are the exact inverses. The transfer is the same as what was expected, thus, the data was correctly encrypted.