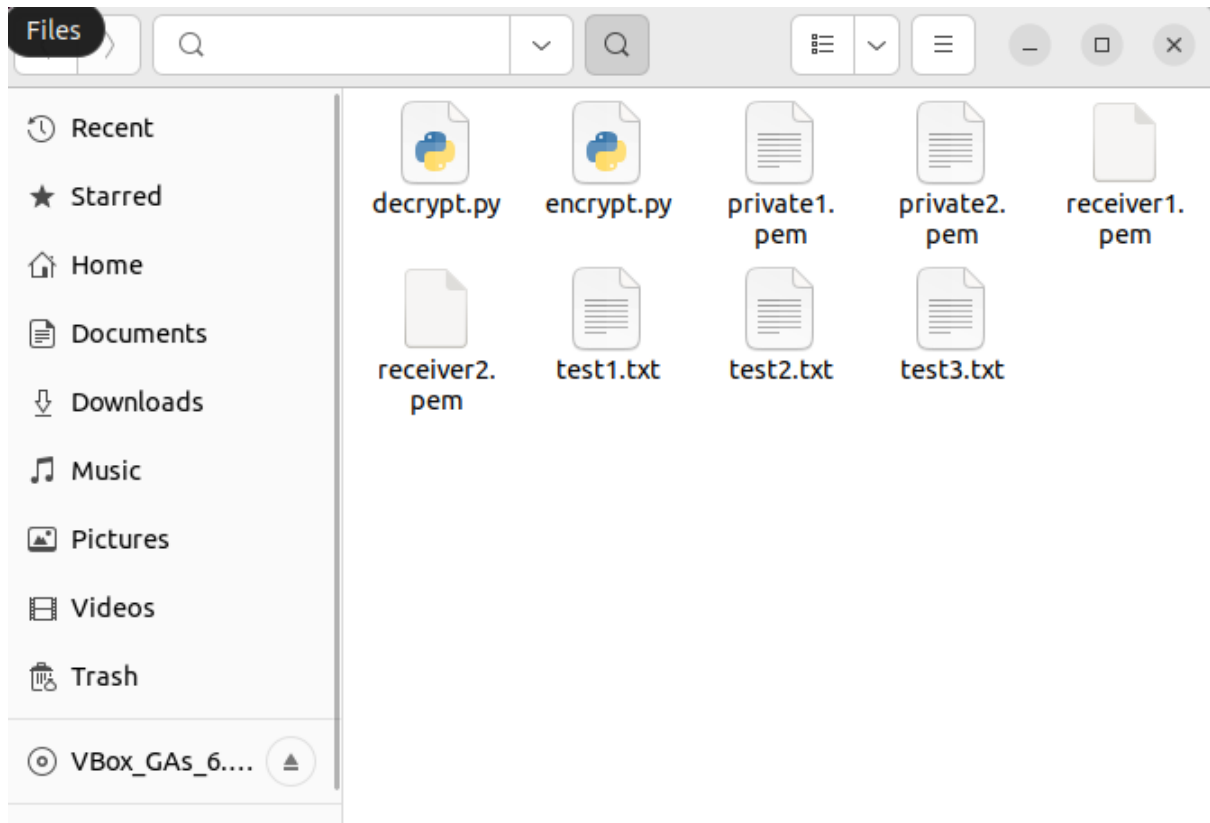


## Bronson Chan

### CSCI 301 – Assignment 1 Report

#### Information to run programs



There are only 9 files in the folder at the start.

**encrypt.py** (encryption program)

**decrypt.py** (decryption program)

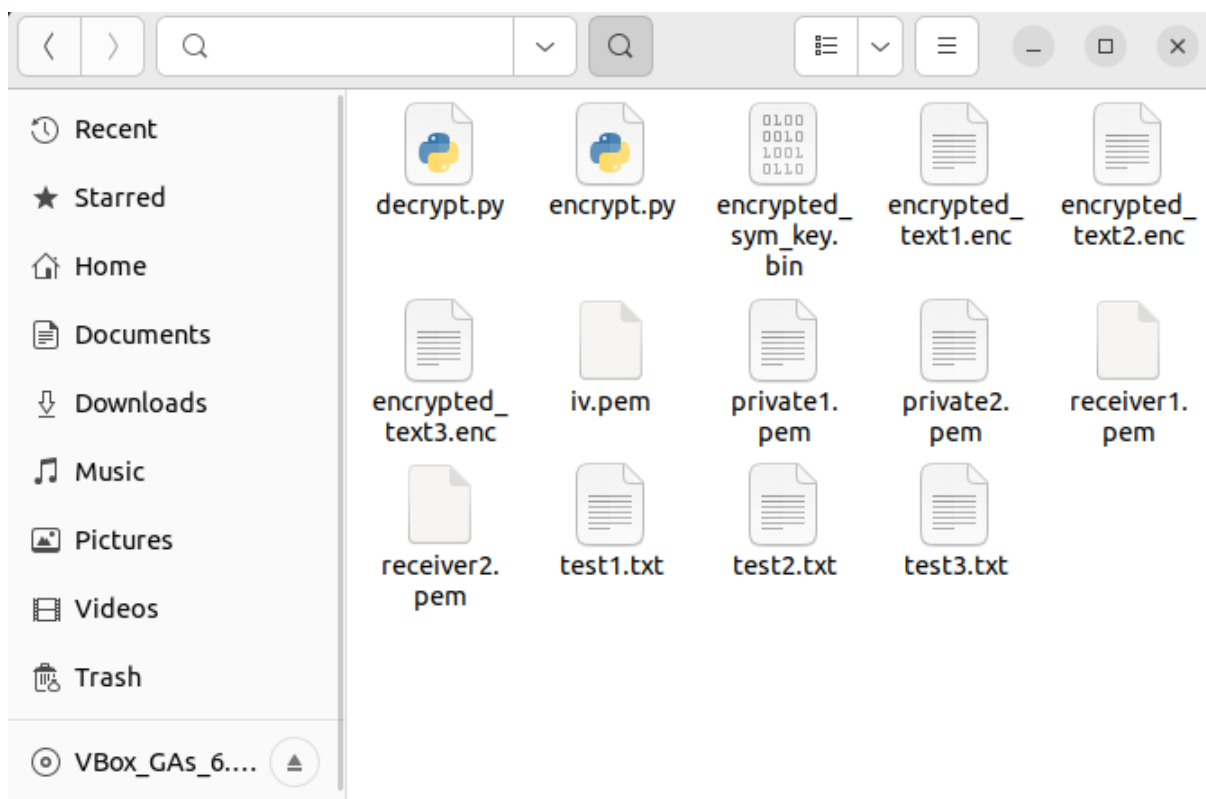
**receiver1.pem** & **private1.pem** (first key pair provided)

**receiver2.pem** & **private2.pem** (second key pair provided)

**test1-3.txt** (3 text files for testing)

1) Go to the directory that the program is in and run “python3 encrypt.py”

```
bronson@bronson-VirtualBox: ~/A1
bronson@bronson-VirtualBox:~$ cd A1
bronson@bronson-VirtualBox:~/A1$ python3 encrypt.py
bronson@bronson-VirtualBox:~/A1$
```



After running “python3 encrypt.py”, these files will appear in the folder.

iv.pem consist of the iv which is hardcoded (since iv must be the same for both)

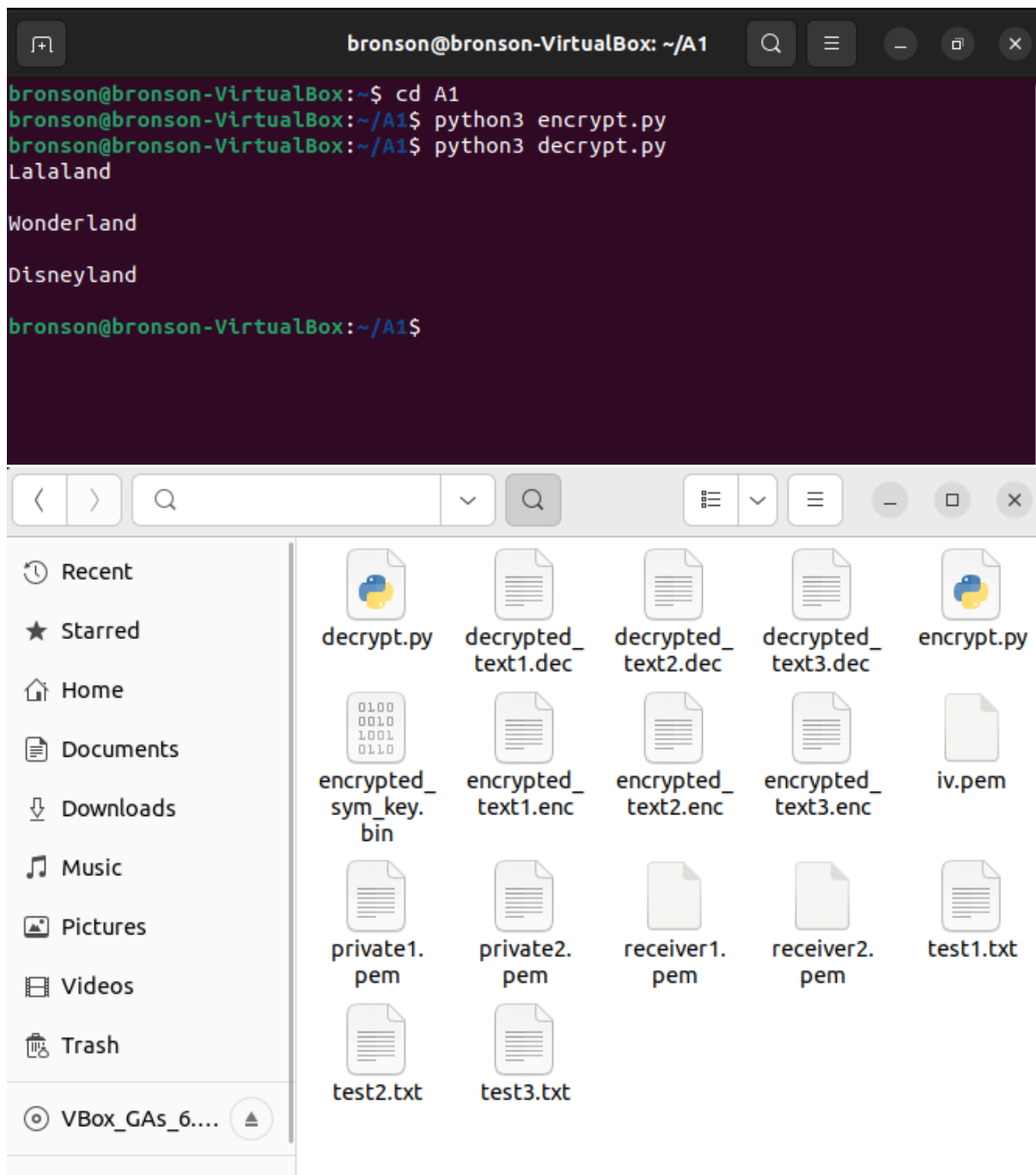
private.pem consist of private key (generated in encrypt program, not the same as the 2 pairs provided)

receiver.pem consist of public key (generated in encrypt program, not the same as the 2 pairs provided)

encrypted\_sym\_key.bin consist of encrypted symmetrical key

encrypted\_text 1-3.enc are all the encrypted version of the 3 text files.

2) Next, we will run “python3 decrypt.py” in the terminal

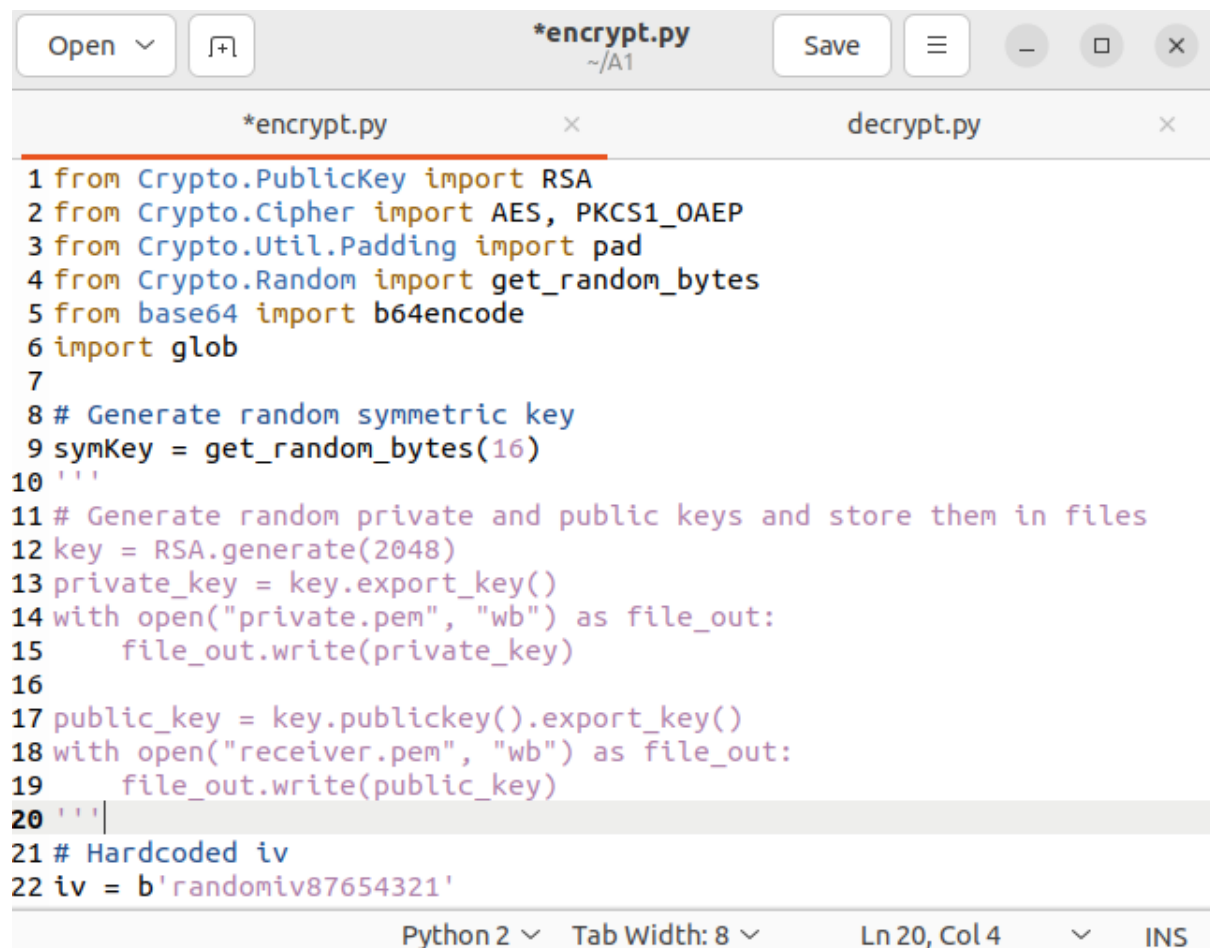


After running “python3 decrypt.py”, these files will appear in the folder.

**decrypted text 1-3** are all the decrypted version of encrypted text.

## Testing out with the 2 pairs of public and private key

Since the code in my encrypt program uses a new generated key (which means a pair different from the 2 required pairs that I have provided), we need to comment it away first.

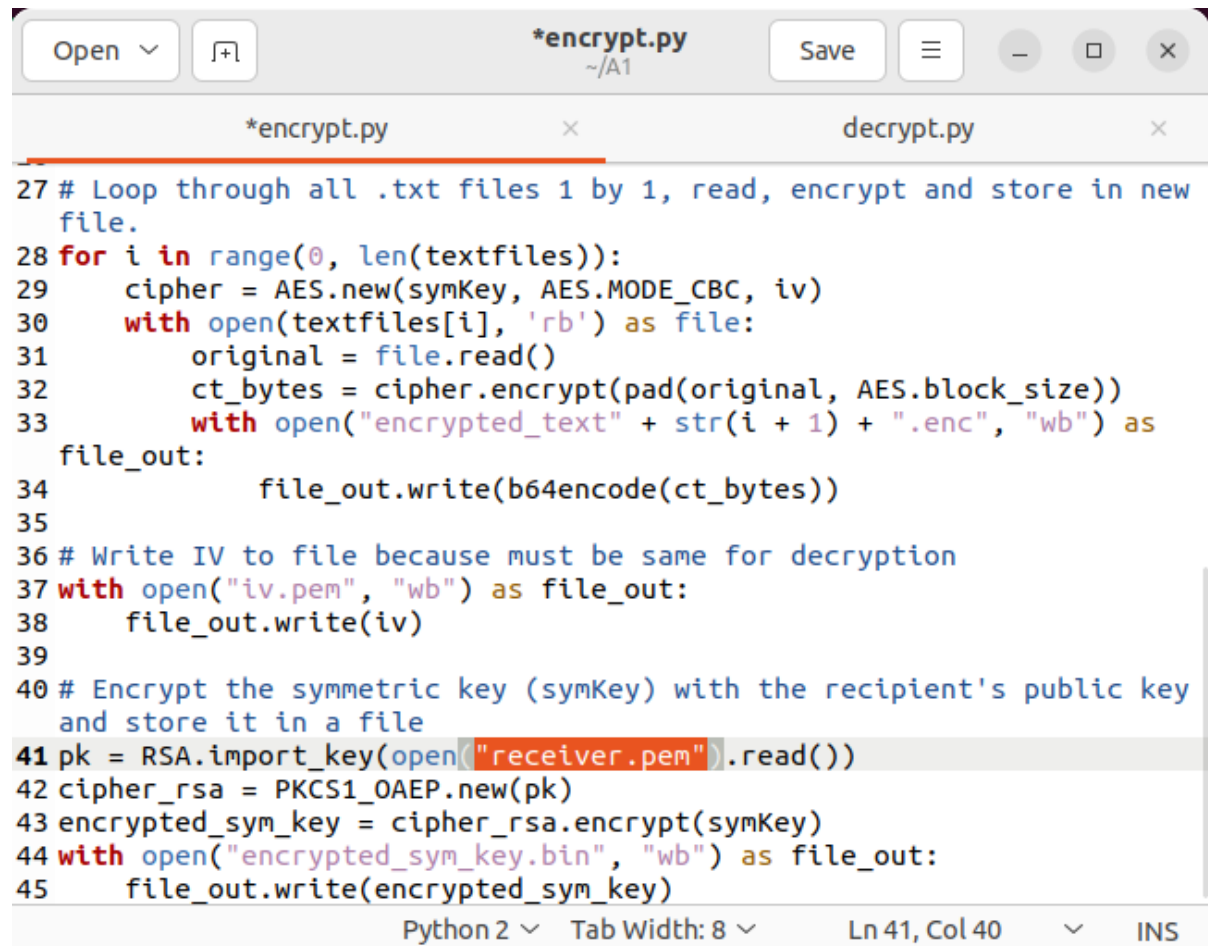


```
1 from Crypto.PublicKey import RSA
2 from Crypto.Cipher import AES, PKCS1_OAEP
3 from Crypto.Util.Padding import pad
4 from Crypto.Random import get_random_bytes
5 from base64 import b64encode
6 import glob
7
8 # Generate random symmetric key
9 symKey = get_random_bytes(16)
10 '''
11 # Generate random private and public keys and store them in files
12 key = RSA.generate(2048)
13 private_key = key.export_key()
14 with open("private.pem", "wb") as file_out:
15     file_out.write(private_key)
16
17 public_key = key.publickey().export_key()
18 with open("receiver.pem", "wb") as file_out:
19     file_out.write(public_key)
20 '''
21 # Hardcoded iv
22 iv = b'randomiv87654321'
```

We will need to comment away the block of code which generate a random pair of keys between line 10 and line 20.

Afterwards, instead of reading from the new generated pair of keys from private.pem and receiver.pem, we will read the keys from either (private1.pem & receiver1.pem) OR (private2.pem & receiver2.pem).

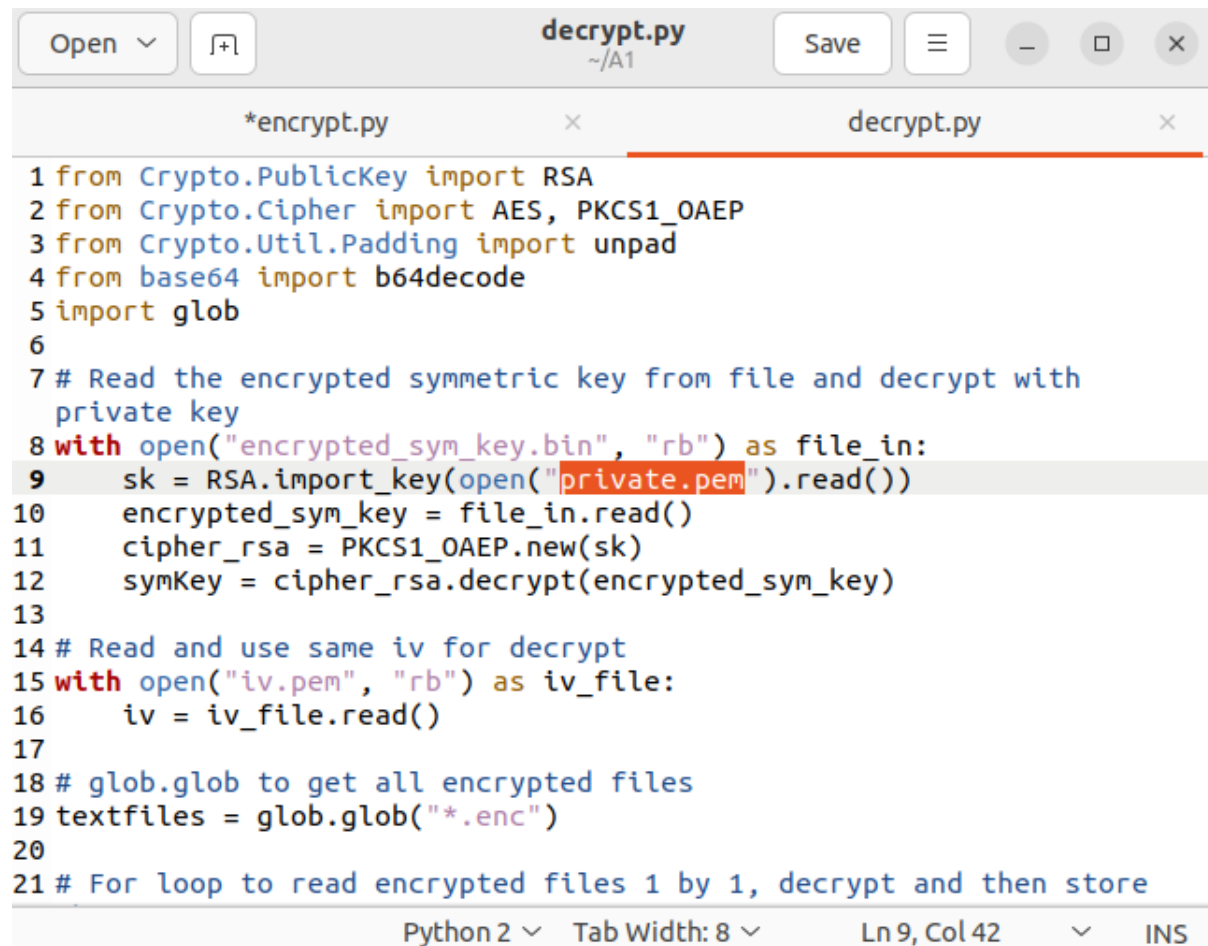
For encrypt.py program, simply change the receiver.pem file to either receiver1.pem or receiver2.pem



```
27 # Loop through all .txt files 1 by 1, read, encrypt and store in new
    file.
28 for i in range(0, len(textfiles)):
29     cipher = AES.new(symKey, AES.MODE_CBC, iv)
30     with open(textfiles[i], 'rb') as file:
31         original = file.read()
32         ct_bytes = cipher.encrypt(pad(original, AES.block_size))
33         with open("encrypted_text" + str(i + 1) + ".enc", "wb") as
            file_out:
34             file_out.write(b64encode(ct_bytes))
35
36 # Write IV to file because must be same for decryption
37 with open("iv.pem", "wb") as file_out:
38     file_out.write(iv)
39
40 # Encrypt the symmetric key (symKey) with the recipient's public key
    and store it in a file
41 pk = RSA.import_key(open("receiver.pem").read())
42 cipher_rsa = PKCS1_OAEP.new(pk)
43 encrypted_sym_key = cipher_rsa.encrypt(symKey)
44 with open("encrypted_sym_key.bin", "wb") as file_out:
45     file_out.write(encrypted_sym_key)
```

Python 2 ▾ Tab Width: 8 ▾ Ln 41, Col 40 ▾ INS

For decrypt.py program, simply change the private.pem file to the respective private1.pem or private2.pem



```
1 from Crypto.PublicKey import RSA
2 from Crypto.Cipher import AES, PKCS1_OAEP
3 from Crypto.Util.Padding import unpad
4 from base64 import b64decode
5 import glob
6
7 # Read the encrypted symmetric key from file and decrypt with
  private key
8 with open("encrypted_sym_key.bin", "rb") as file_in:
9     sk = RSA.import_key(open("private.pem").read())
10     encrypted_sym_key = file_in.read()
11     cipher_rsa = PKCS1_OAEP.new(sk)
12     symKey = cipher_rsa.decrypt(encrypted_sym_key)
13
14 # Read and use same iv for decrypt
15 with open("iv.pem", "rb") as iv_file:
16     iv = iv_file.read()
17
18 # glob.glob to get all encrypted files
19 textfiles = glob.glob("*.enc")
20
21 # For loop to read encrypted files 1 by 1, decrypt and then store
```

Python 2 ▾ Tab Width: 8 ▾ Ln 9, Col 42 ▾ INS

Before attempting to test using the 2 pairs of keys provided, please remove all additional files so that the folder looks exactly like how it was in Image number 1 right at the top.