CSCI301 Assignment 1 Report

Name: Ng Chin Chia ID: 7058901

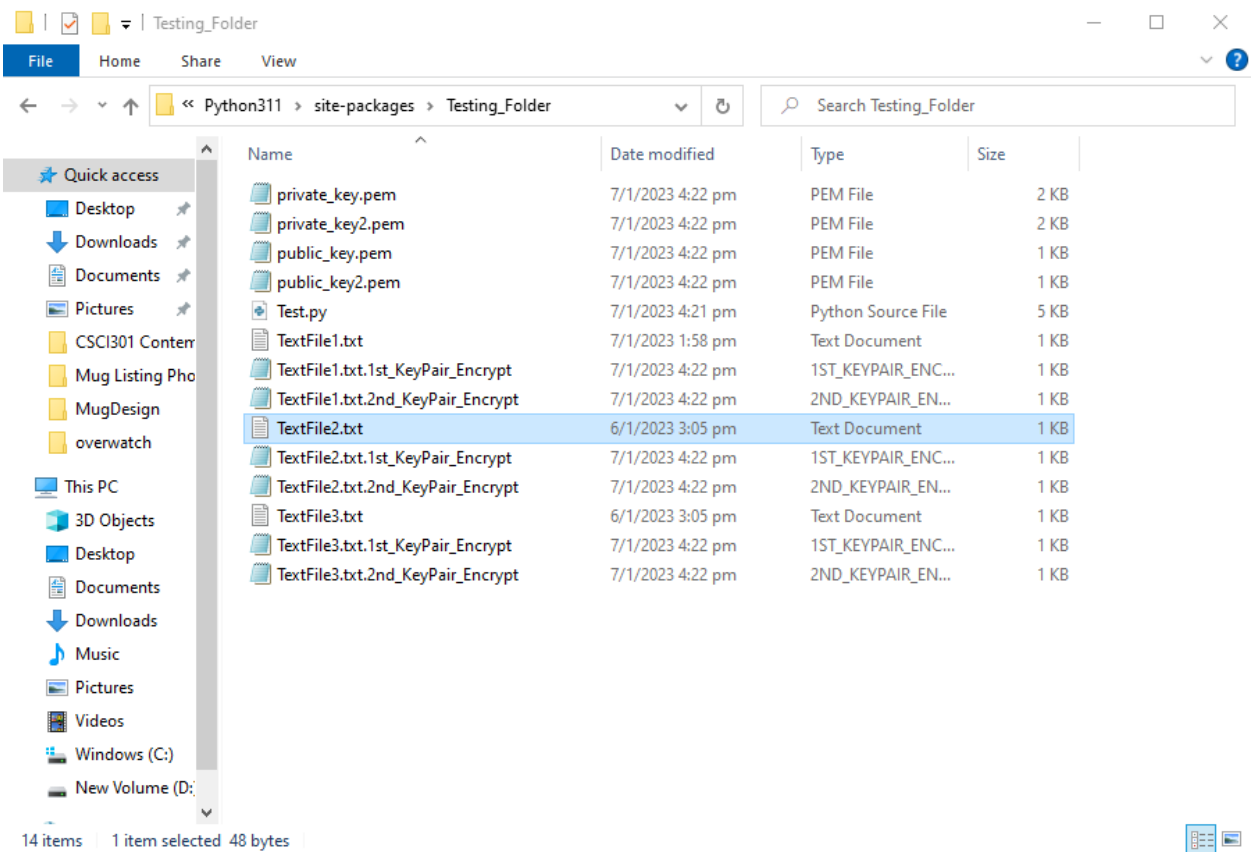
EXPLANATION FOR ENCRYPT FUNCTION

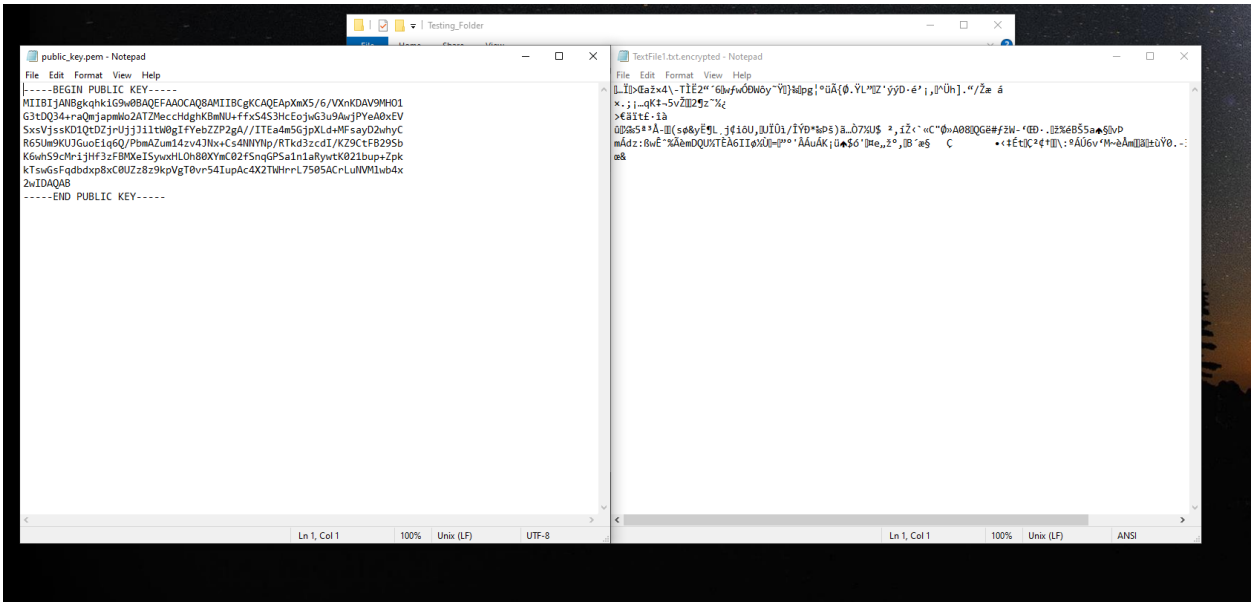
To ensure the security of the files, I'm using an advanced encryption system. First, I generate a random 128-bit key for AES encryption using os.urandom(). Then, I create a RSA 2048 public key using RSA.generate() and save it to a file in PEM format. The public key is then loaded and stored in a variable. The AES key is then encrypted using the RSA public key. After that, I get a list of all the .txt files in the current directory and encrypt them one by one. For each file, I generate a random initial vector (IV) for the AES encryption using os.urandom(). Then, I create a new AES cipher using the key and IV, and encrypt the contents of the file using the AES cipher's .encrypt() method. The encrypted data, IV, and encrypted key are then written to a new file. Finally, the function is tested by calling encrypt_files().

In order to have a clearer explanation, the codes for the encrypt function have been commented step by step to clear any doubts.

After running the function, a following numbers of files are created :

- public_key.pem
- public_key2.pem
- private_key.pem
- private_key2.pem
- TextFile1.txt.1st_KeyPair_Encrypt
- TextFile1.txt.2nd_KeyPair_Encrypt
- TextFile2.txt.1st_KeyPair_Encrypt
- TextFile2.txt.2nd_KeyPair_Encrypt
- TextFile3.txt.1st_KeyPair_Encrypt
- TextFile3.txt.2nd_KeyPair_Encrypt





The following screenshot below shows the RSA 2048 public key written in pem format and the encrypted text file of an example text file.

NECESSARY INFORMATION TO RUN THE PROGRAM

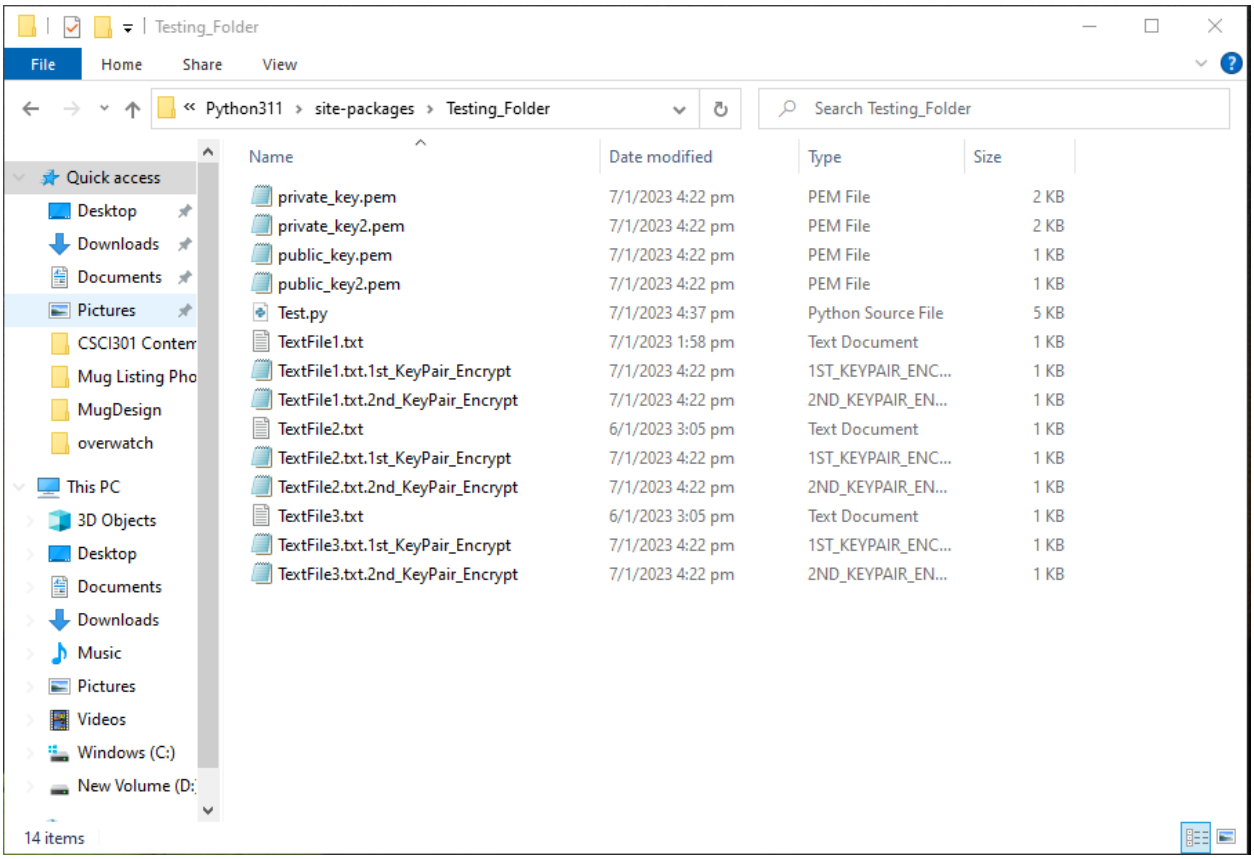
1. Install/Ensure Python 3.5 or above on the computer.
2. Install pycryptodome package which provides AES_CBC and RSA 2048 public-key encryption functionality in Python. To do so, open up terminal and type the following command. It should install pycryptodome in your computer automatically.

```
pip install pycryptodome
```

3. Open file location of where pycryptodome is installed in and create a folder called Testing_Folder or any folder name, making sure it is within the same directory as pycryptodome.
4. The folder will consist of the python program to be run and 3 other sample text files with some random information.
5. Open visual studio code and the python program and scroll all the way down to see instructions on how to test the code.

```
# Test the function. Instruction: 1st run encrypt_files() and comment it, followed by decrypt_file function.

#__________This part is to test for encrypt function____________#
#uncomment the function below and run it. It should generate 2 pairs of public and private key in pem files as well as the respective encrypted files.

encrypt_files()

#__________This part is to test for decrypt function____________#
#uncomment the following below and run it. It should generate the decrypted files for first key pair and second key pair encrypted files using different private keys.

file_names = glob.glob('*.1st_KeyPair_Encrypt') + glob.glob('*.2nd_KeyPair_Encrypt')

for file_name in file_names:
    if file_name.endswith('1st_KeyPair_Encrypt'):
        private_key_file = 'private_key.pem'
    else:
        private_key_file = 'private_key2.pem'
    decrypt_file(file_name, private_key_file)
```

6. Firstly, comment out the decrypt_file() function and only run the encrypt_files() function. This will generate 2 pairs of public and private key pem files, as well as the written encrypt files with their respective naming convention shown in the screenshot below.

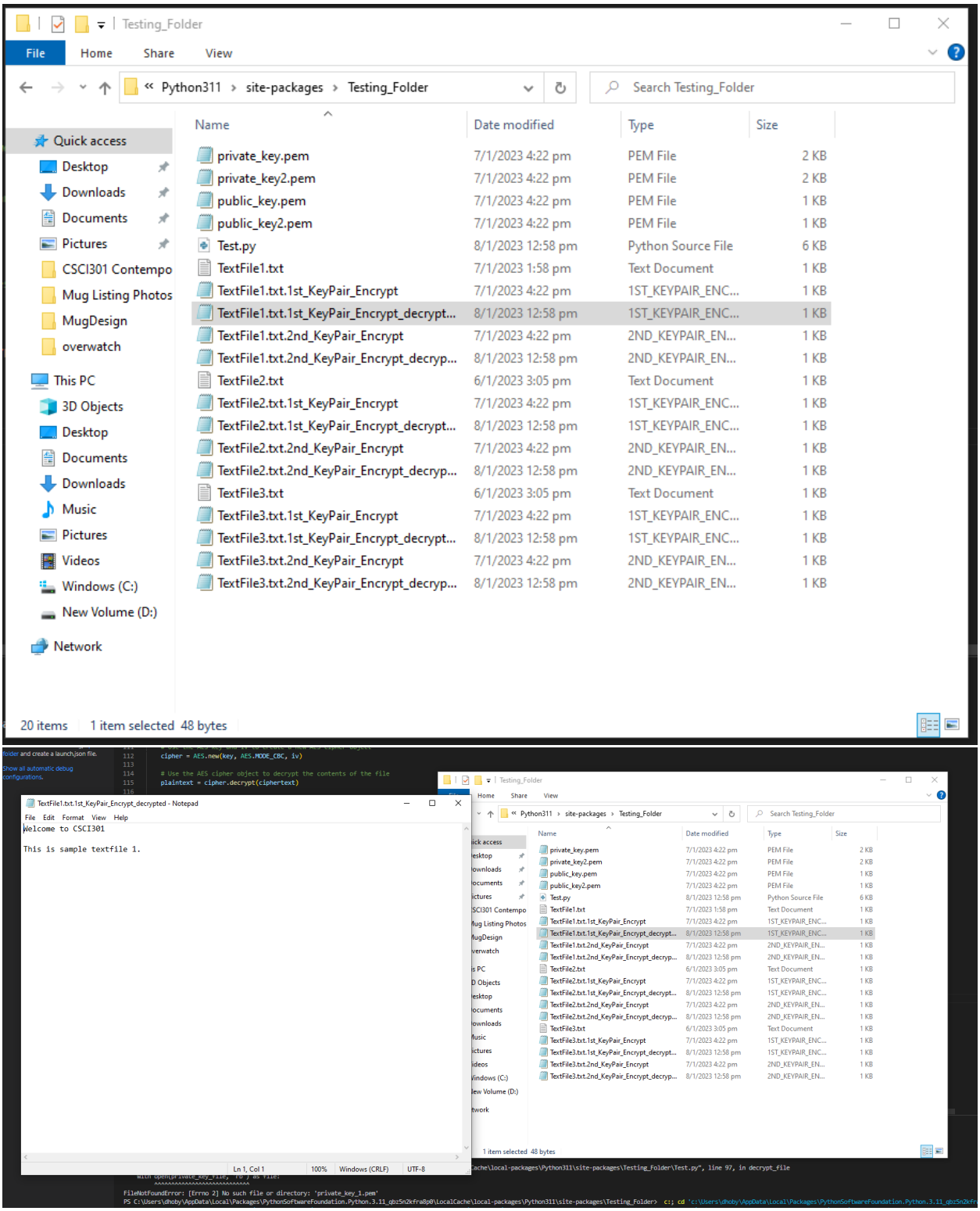


7. **NOTE:** To decrypt, uncomment and execute the following code in the screenshots below. The first private key should only be able to decrypt the encrypted files with 1st_KeyPair_Encrypt. The second private key should only be able to decrypt from the encrypted files with 2nd_KeyPair_Encrypt.

```
#_________This part is to test for decrypt function_____________#
#uncomment the following below and run it. It should generate the decrypted files for first key pair and second key pair encrypted files using different private keys.
#Remove the top ''' and the bottom ''' and execute the code.

'''
file_names = glob.glob('*.1st_KeyPair_Encrypt') + glob.glob('*.2nd_KeyPair_Encrypt')

for file_name in file_names:
    if file_name.endswith('1st_KeyPair_Encrypt'):
        private_key_file = 'private_key.pem'
    else:
        private_key_file = 'private_key2.pem'
    decrypt_file(file_name, private_key_file)
'''
```

```
#_________This part is to test for decrypt function_____________#
#uncomment the following below and run it. It should generate the decrypted files for first key pair and second key pair encrypted files using different private keys.
#Remove the top ''' and the bottom ''' and execute the code.


file_names = glob.glob('*.1st_KeyPair_Encrypt') + glob.glob('*.2nd_KeyPair_Encrypt')

for file_name in file_names:
    if file_name.endswith('1st_KeyPair_Encrypt'):
        private_key_file = 'private_key.pem'
    else:
        private_key_file = 'private_key2.pem'
    decrypt_file(file_name, private_key_file)
```

8. After running the decrypt_file() function, it should show the decrypted or original message of the encrypted file as shown in the following screenshots below.


Testing_Folder
File
Home
Share
View
« Python311 › site-packages › Testing_Folder
Search Testing_Folder
Quick access
Desktop
Downloads
Documents
Pictures
CSCI301 Contempo
Mug Listing Photos
MugDesign
overwatch
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
New Volume (D:)
Network
Name
Date modified
Type
Size
private_key.pem 7/1/2023 4:22 pm PEM File 2 KB
private_key2.pem 7/1/2023 4:22 pm PEM File 2 KB
public_key.pem 7/1/2023 4:22 pm PEM File 1 KB
public_key2.pem 7/1/2023 4:22 pm PEM File 1 KB
Test.py 8/1/2023 12:58 pm Python Source File 6 KB
TextFile1.txt 7/1/2023 1:58 pm Text Document 1 KB
TextFile1.txt.1st_KeyPair_Encrypt 7/1/2023 4:22 pm 1ST_KEYPAIR_ENC... 1 KB
TextFile1.txt.1st_KeyPair_Encrypt_decrypt... 8/1/2023 12:58 pm 1ST_KEYPAIR_ENC... 1 KB
TextFile1.txt.2nd_KeyPair_Encrypt 7/1/2023 4:22 pm 2ND_KEYPAIR_EN... 1 KB
TextFile1.txt.2nd_KeyPair_Encrypt_decryp... 8/1/2023 12:58 pm 2ND_KEYPAIR_EN... 1 KB
TextFile2.txt 6/1/2023 3:05 pm Text Document 1 KB
TextFile2.txt.1st_KeyPair_Encrypt 7/1/2023 4:22 pm 1ST_KEYPAIR_ENC... 1 KB
TextFile2.txt.1st_KeyPair_Encrypt_decrypt... 8/1/2023 12:58 pm 1ST_KEYPAIR_ENC... 1 KB
TextFile2.txt.2nd_KeyPair_Encrypt 7/1/2023 4:22 pm 2ND_KEYPAIR_EN... 1 KB
TextFile2.txt.2nd_KeyPair_Encrypt_decryp... 8/1/2023 12:58 pm 2ND_KEYPAIR_EN... 1 KB
TextFile3.txt 6/1/2023 3:05 pm Text Document 1 KB
TextFile3.txt.1st_KeyPair_Encrypt 7/1/2023 4:22 pm 1ST_KEYPAIR_ENC... 1 KB
TextFile3.txt.1st_KeyPair_Encrypt_decrypt... 8/1/2023 12:58 pm 1ST_KEYPAIR_ENC... 1 KB
TextFile3.txt.2nd_KeyPair_Encrypt 7/1/2023 4:22 pm 2ND_KEYPAIR_EN... 1 KB
TextFile3.txt.2nd_KeyPair_Encrypt_decryp... 8/1/2023 12:58 pm 2ND_KEYPAIR_EN... 1 KB
20 items
1 item selected 48 bytes
cipher = AES.new(key, AES.MODE_CBC, iv)
# Use the AES cipher object to decrypt the contents of the file
plaintext = cipher.decrypt(ciphertext)
TextFile1.txt.1st_KeyPair_Encrypt_decrypted - Notepad
File Edit Format View Help
Welcome to CSCI301
This is sample textfile 1.
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
FileNotFoundError: [Errno 2] No such file or directory: 'private_key_1.pem'
PS C:\Users\dhoby\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\Testing_Folder>

9. It should automatically write the decrypted messages into new files for all the encrypted files ending with .1st_KeyPair_Encrypt and 2nd_KeyPair_Encrypt based on the testing codes.

```
file_names = glob.glob('*.1st_KeyPair_Encrypt') + glob.glob('*.2nd_KeyPair_Encrypt')

for file_name in file_names:
    if file_name.endswith('1st_KeyPair_Encrypt'):
        private_key_file = 'private_key.pem'
    else:
        private_key_file = 'private_key2.pem'
    decrypt_file(file_name, private_key_file)
```

This code does the following:

1. It imports the **glob** module and uses the **glob** function to generate a list of file names that match either of two patterns: **'*.1st_KeyPair_Encrypt'** and **'*.2nd_KeyPair_Encrypt'**. The **glob** function searches the current directory for files that match the given pattern and returns a list of the names of the matching files. The **'*.1st_KeyPair_Encrypt'** pattern will match any file that has a name that ends with **.1st_KeyPair_Encrypt**, and the **'*.2nd_KeyPair_Encrypt'** pattern will match any file that has a name that ends with **.2nd_KeyPair_Encrypt**.
2. It combines the two lists of file names into a single list using the **+** operator.
3. It iterates through the list of file names using a **for** loop.
4. For each file in the list, it determines which private key file to use based on the file name. If the file name ends with **'1st_KeyPair_Encrypt'**, it sets the **private_key_file** variable to **'private_key.pem'**, otherwise it sets it to **'private_key2.pem'**.
5. It calls the **decrypt_file** function, passing in the file name and the private key file as arguments. This will decrypt the file using the specified private key.
   This code will therefore decrypt all of the files in the current directory that have either of the two specified file extensions, using the appropriate private key for each file. The decrypted files will be saved with a **_decrypted** suffix added to the original file name.