Report

Simulating Generating RSA, Encryption and Decryption

Generate RSA Key Pair

- First, a private RSA key is generated with a public exponent of 65537 and a key size of 2048.
- The private key is then saved to a file "private_key.pem" in PEM format with no encryption.
- A public key is derived from the private key, and the public key is saved to "public_key.pem" also in PEM format.

Encryption

- The code checks if the RSA key pair exists; if not, it generates a new pair.
- A plaintext file "longPlaintext.txt" is read.
- A symmetric key of 32 bytes and an initialization vector (IV) of 16 bytes are generated. The plaintext is encrypted using AES encryption in Galois/Counter Mode (GCM).
- The symmetric key is encrypted with the public key using RSA and saved along with the ciphertext to "longEncryptedData.bin."

Decryption

- The private key is loaded from "private_key.pem."
- The encrypted symmetric key and ciphertext are read from "longEncryptedData.bin."
- The symmetric key is decrypted using the private RSA key, and the IV is generated.
- The ciphertext is decrypted using the symmetric key and IV with AES GCM.
- The decrypted plaintext is saved to "longDecryptedText.txt."

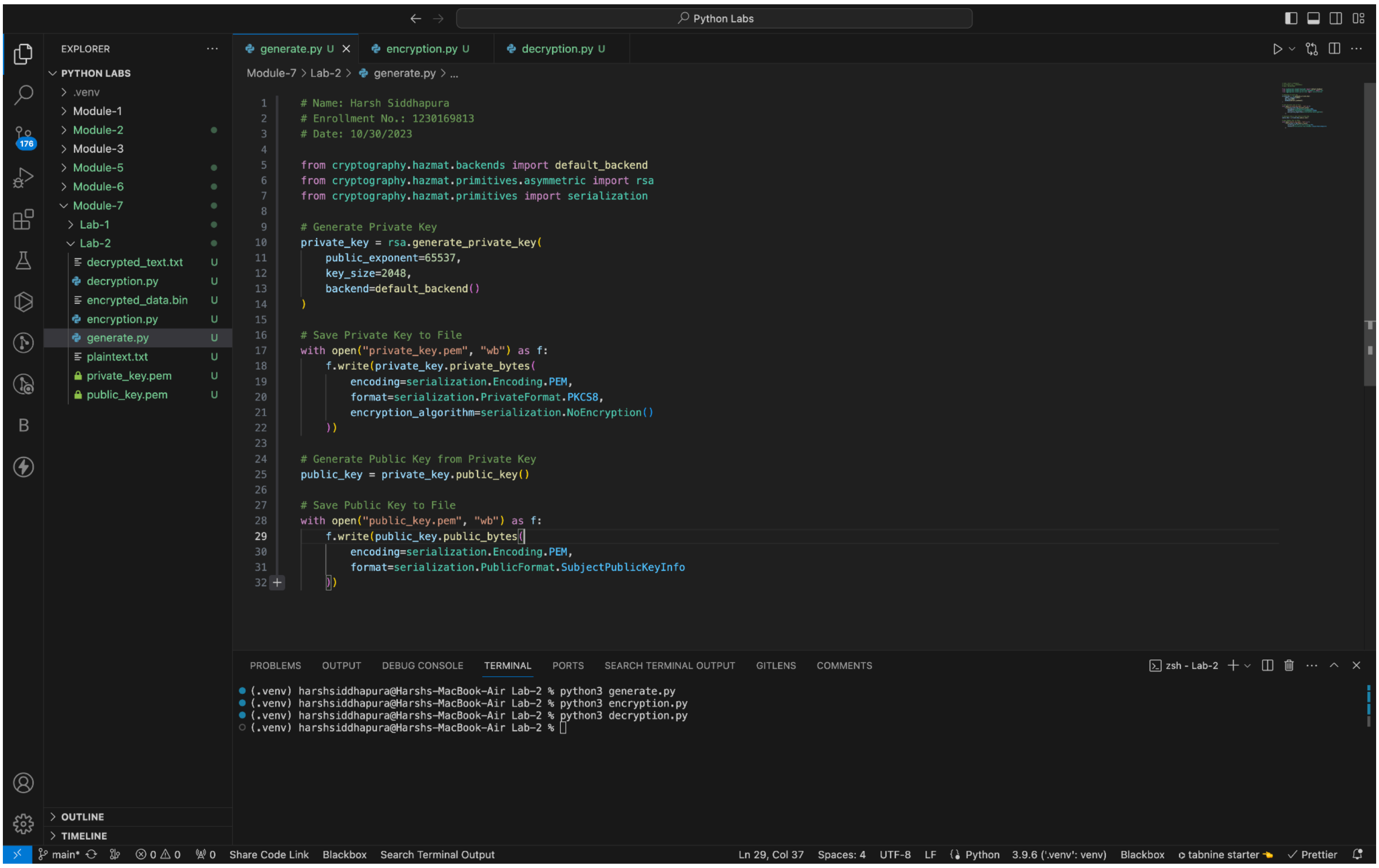
This code provides a basic example of RSA key generation, encryption, and decryption for large files. However, there are some improvements needed:

- The code doesn't handle exceptions, which may lead to crashes if files or keys are missing or invalid.
- For production use, error handling, key management, and secure storage of keys are crucial.
- The code assumes that the keys and IV are generated and handled securely, which is not evident from this example.
- In practice, it's essential to use secure random number generation and key storage mechanisms.
- Additionally, the code should ensure that the encryption key and IV are preserved, and the decryption code should use the same IV that was used for encryption.

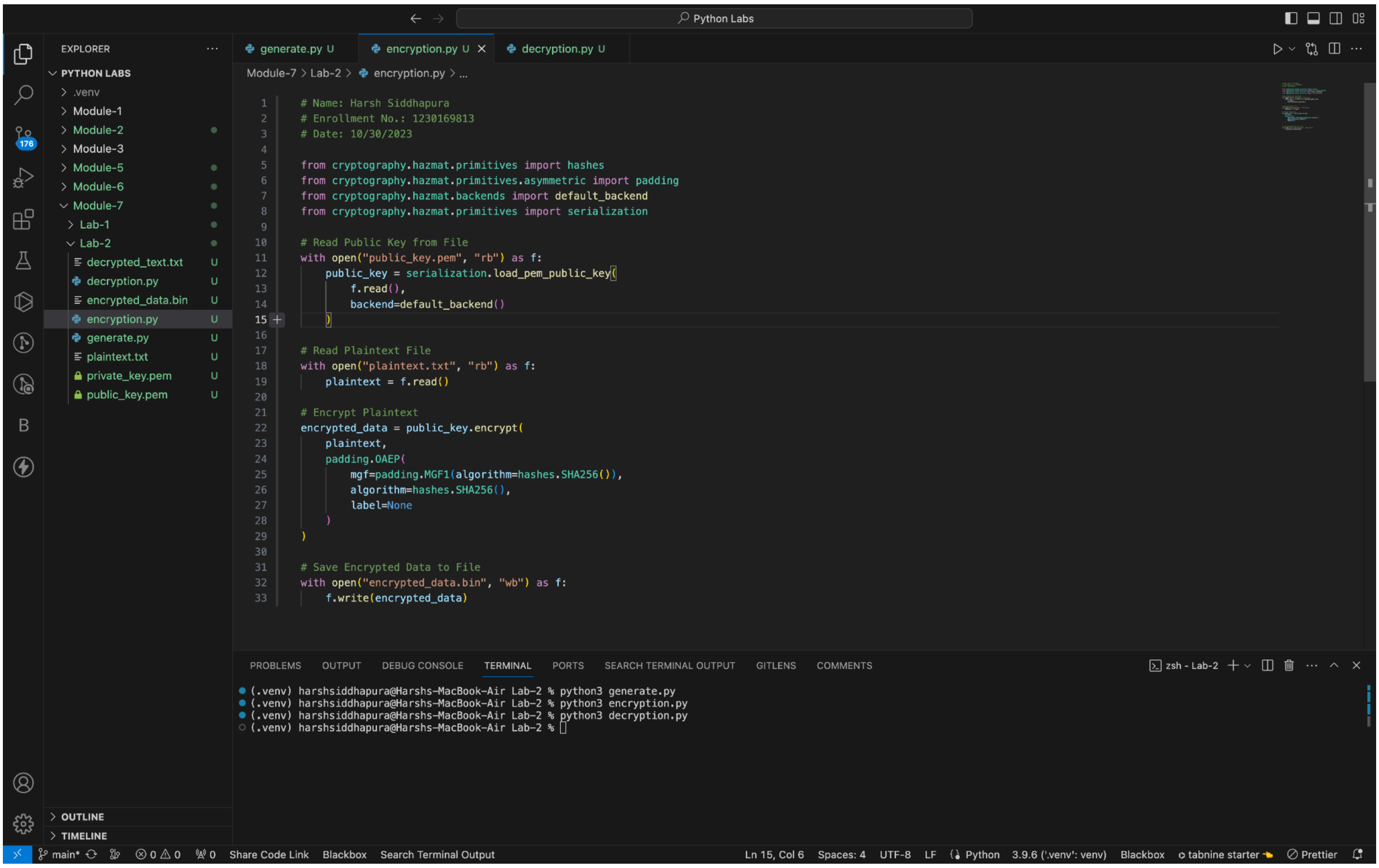
In the presented code, a process for generating RSA key pairs, encrypting a plaintext file using AES encryption, and decrypting the data back has been demonstrated. The code showcases the generation of a private RSA key, with the corresponding public key, and the storage of these keys in separate PEM files. It further outlines the encryption of a large plaintext file using a symmetric key generated with secure random bytes and AES encryption in Galois/Counter Mode (GCM), with the encrypted symmetric key also saved alongside the ciphertext in a binary file. The decryption process, on the other hand, reads the private key and decrypts the symmetric key, subsequently decrypting the ciphertext using the symmetric key and IV. The decrypted plaintext is saved to an output file. This code example offers a fundamental understanding of key generation, encryption, and decryption processes, yet it lacks comprehensive error handling and security measures required for production use.

In conclusion, the code provides a foundational illustration of cryptographic operations involving RSA key pairs and symmetric key encryption, showcasing essential concepts in cryptography. While it serves as a useful starting point for educational purposes, several critical security considerations and error handling mechanisms should be incorporated for real-world applications. In practice, secure key management, cryptographic libraries, and secure random number generation are essential for ensuring data confidentiality and integrity. The presented code should be viewed as a starting point for more robust and secure implementations, which must address the complexities of key storage, distribution, and management in a secure environment.

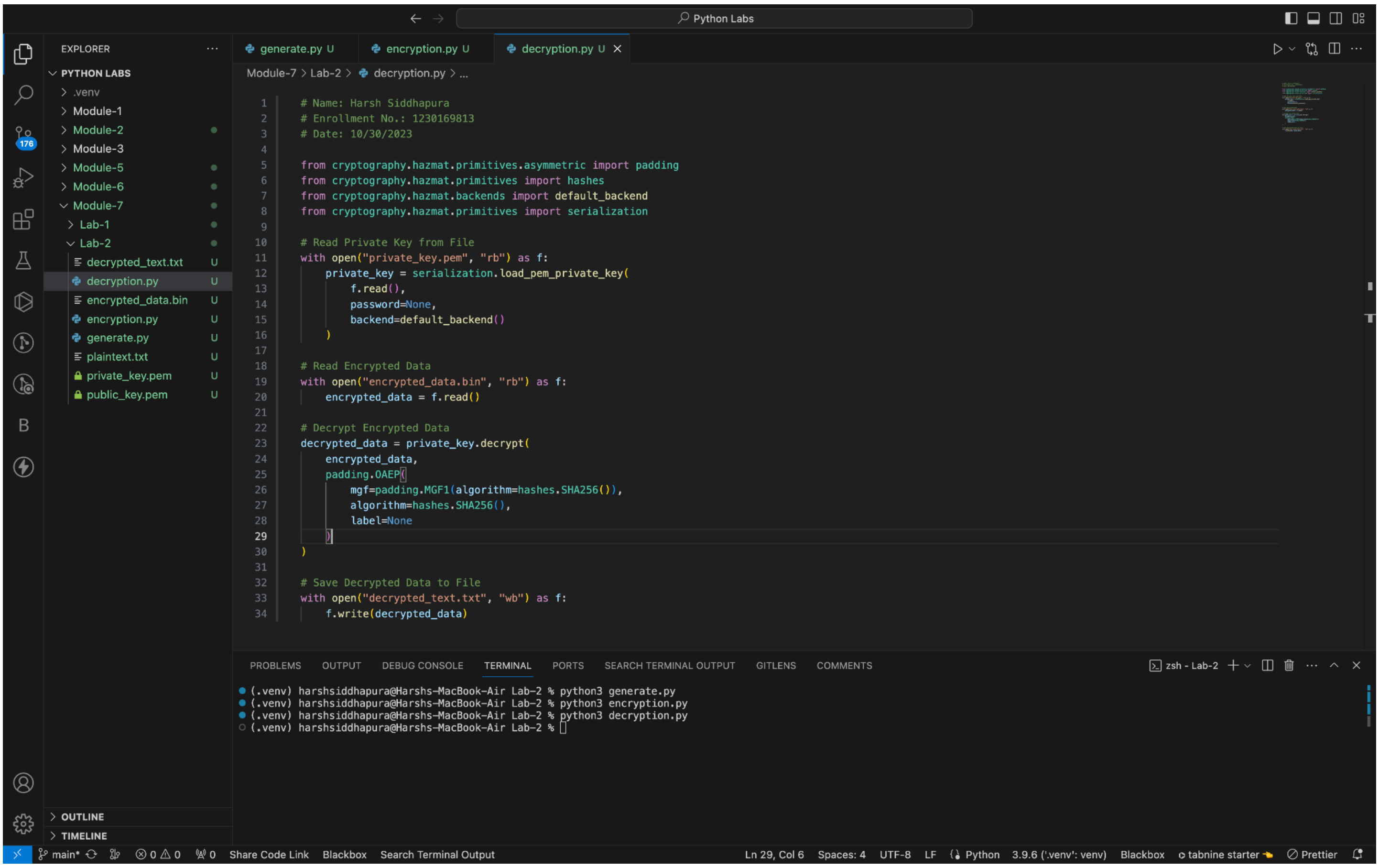
Screenshot

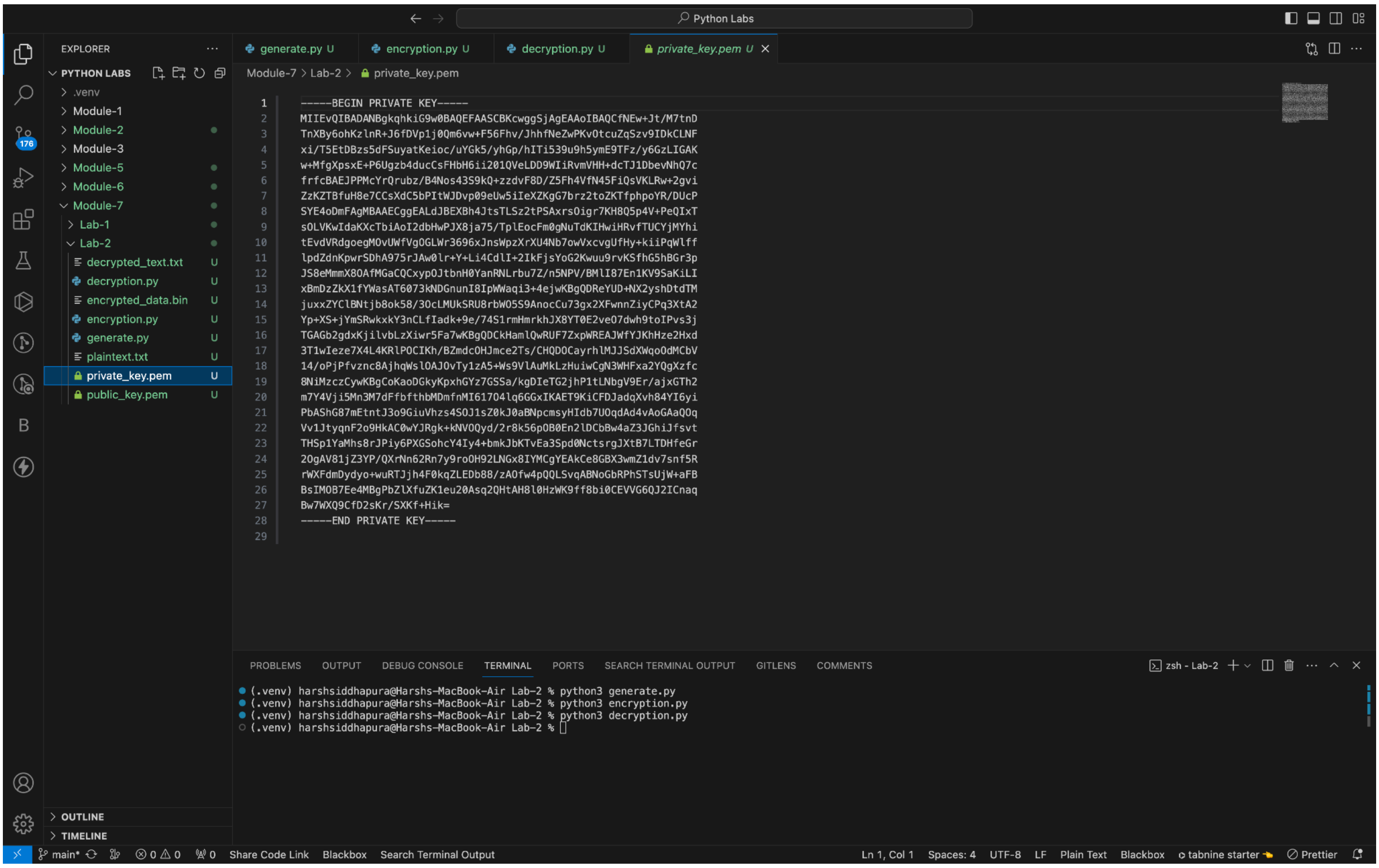


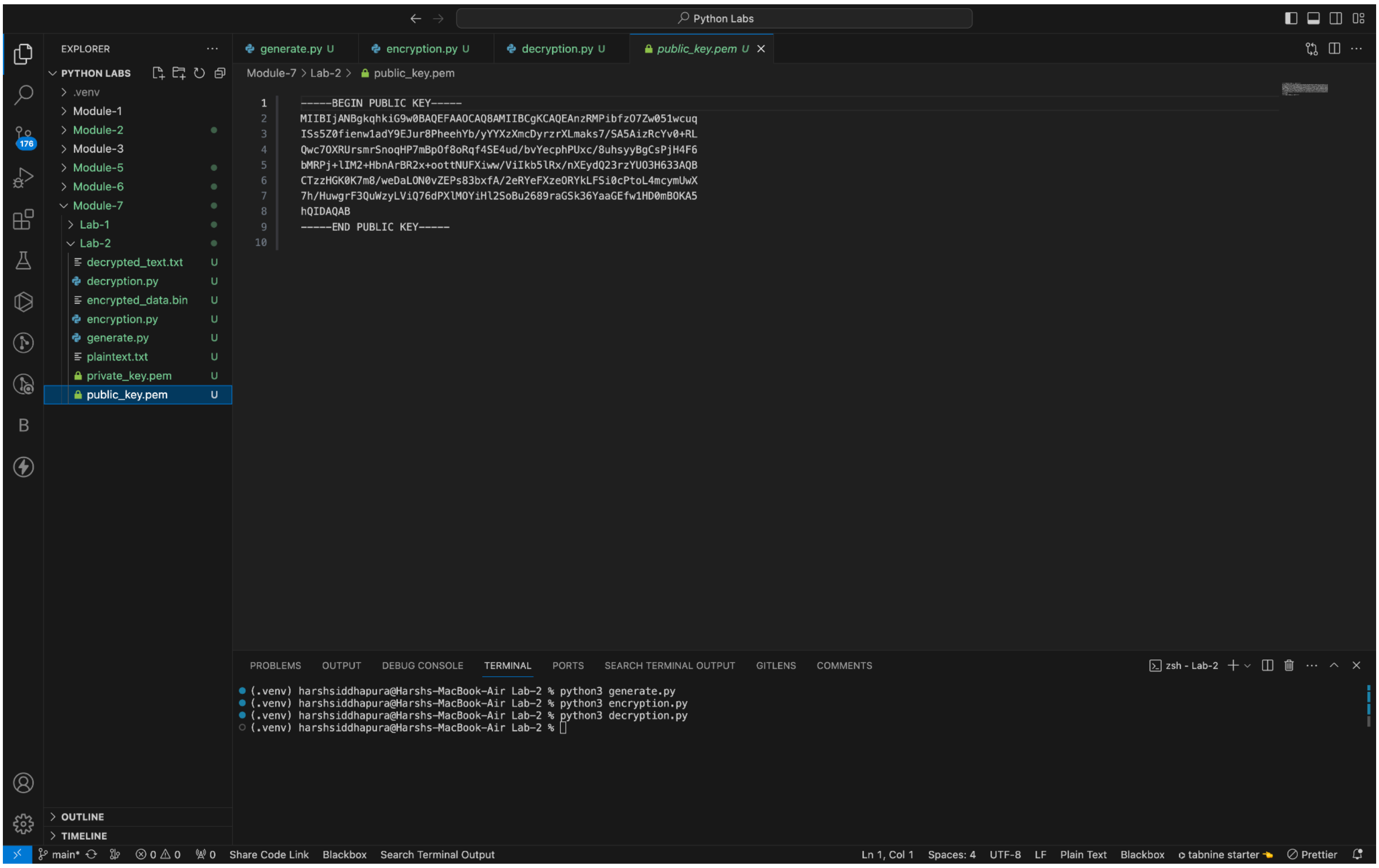
Python Labs
EXPLORER
PYTHON LABS
.venv
Module-1
Module-2
Module-3
Module-5
Module-6
Module-7
Lab-1
Lab-2
decrypted_text.txt
decryption.py
encrypted_data.bin
encryption.py
generate.py
plaintext.txt
private_key.pem
public_key.pem
generate.py U
encryption.py U
decryption.py U
Module-7 > Lab-2 > generate.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
# Generate Private Key
private_key = rsa.generate_private_key(
public_exponent=65537,
key_size=2048,
backend=default_backend()
)
Save Private Key to File
with open("private_key.pem", "wb") as f:
f.write(private_key.private_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PrivateFormat.PKCS8,
encryption_algorithm=serialization.NoEncryption()
))
# Generate Public Key from Private Key
public_key = private_key.public_key()
# Save Public Key to File
with open("public_key.pem", "wb") as f:
f.write(public_key.public_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PublicFormat.SubjectPublicKeyInfo
))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - Lab-2
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 decryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 %
OUTLINE
TIMELINE
main* Share Code Link Blackbox Search Terminal Output
Ln 29, Col 37 Spaces: 4 UTF-8 LF Python 3.9.6 ('.venv': venv) Blackbox tabnine starter Prettier

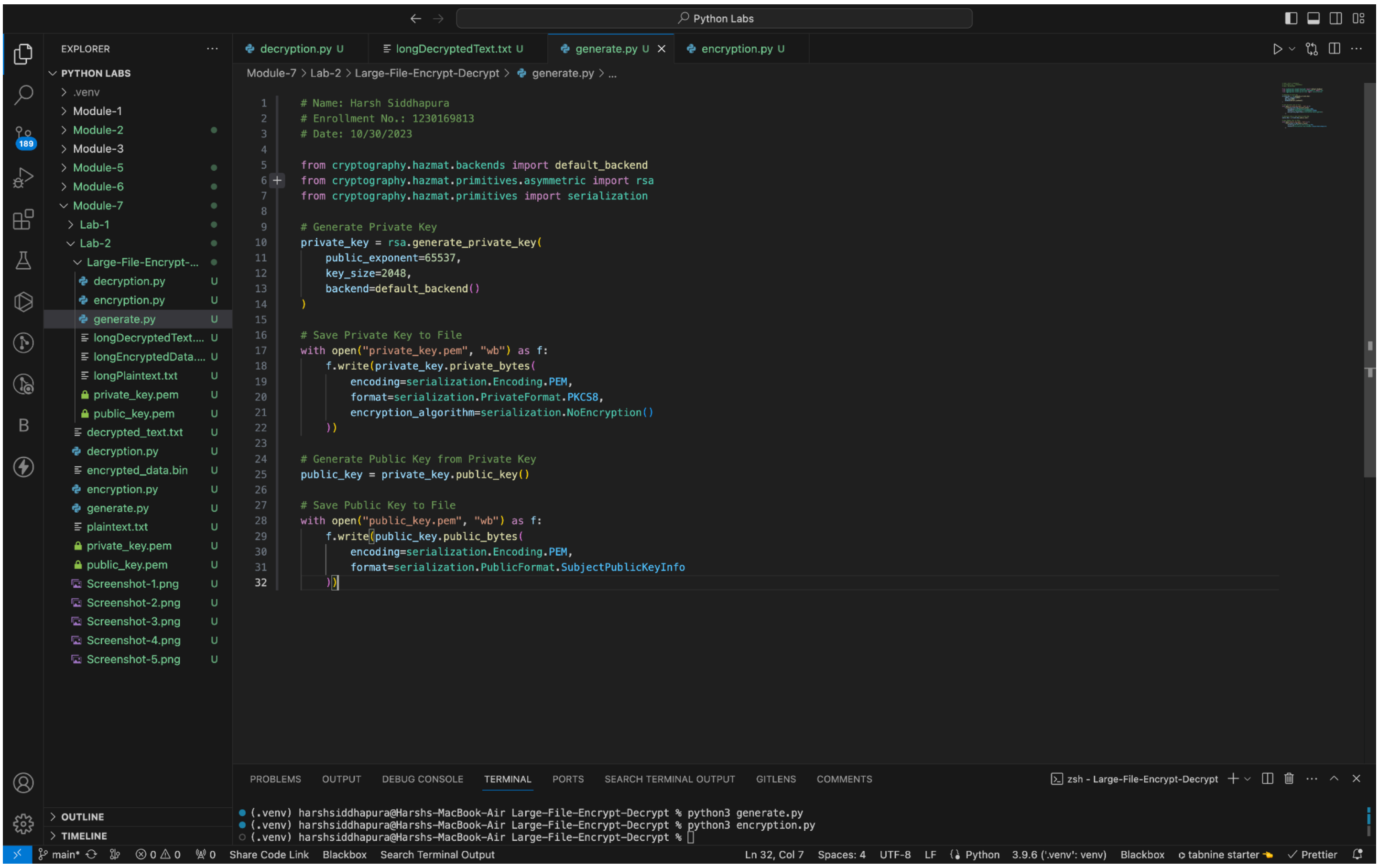


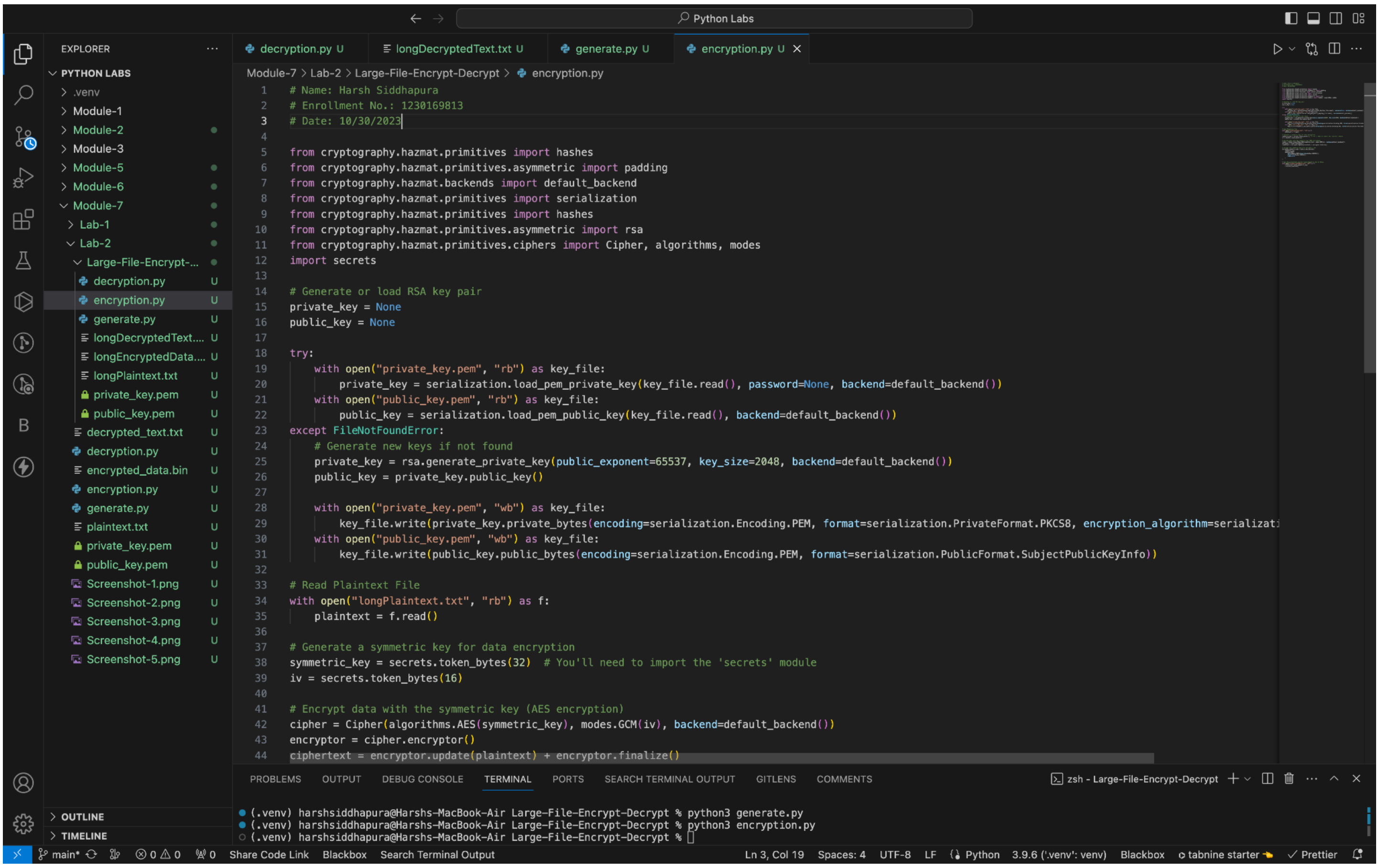
Python Labs
EXPLORER
PYTHON LABS
.venv
Module-1
Module-2
Module-3
Module-5
Module-6
Module-7
Lab-1
Lab-2
decrypted_text.txt
decryption.py
encrypted_data.bin
encryption.py
generate.py
plaintext.txt
private_key.pem
public_key.pem
generate.py U
encryption.py U
decryption.py U
Module-7 > Lab-2 > encryption.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
# Read Public Key from File
with open("public_key.pem", "rb") as f:
public_key = serialization.load_pem_public_key(
f.read(),
backend=default_backend()
)
Read Plaintext File
with open("plaintext.txt", "rb") as f:
plaintext = f.read()
# Encrypt Plaintext
encrypted_data = public_key.encrypt(
plaintext,
padding.OAEP(
mgf=padding.MGF1(algorithm=hashes.SHA256()),
algorithm=hashes.SHA256(),
label=None
)
)
Save Encrypted Data to File
with open("encrypted_data.bin", "wb") as f:
f.write(encrypted_data)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - Lab-2
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 decryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 %
OUTLINE
TIMELINE
main* Share Code Link Blackbox Search Terminal Output
Ln 15, Col 6 Spaces: 4 UTF-8 LF Python 3.9.6 ('.venv': venv) Blackbox tabnine starter Prettier

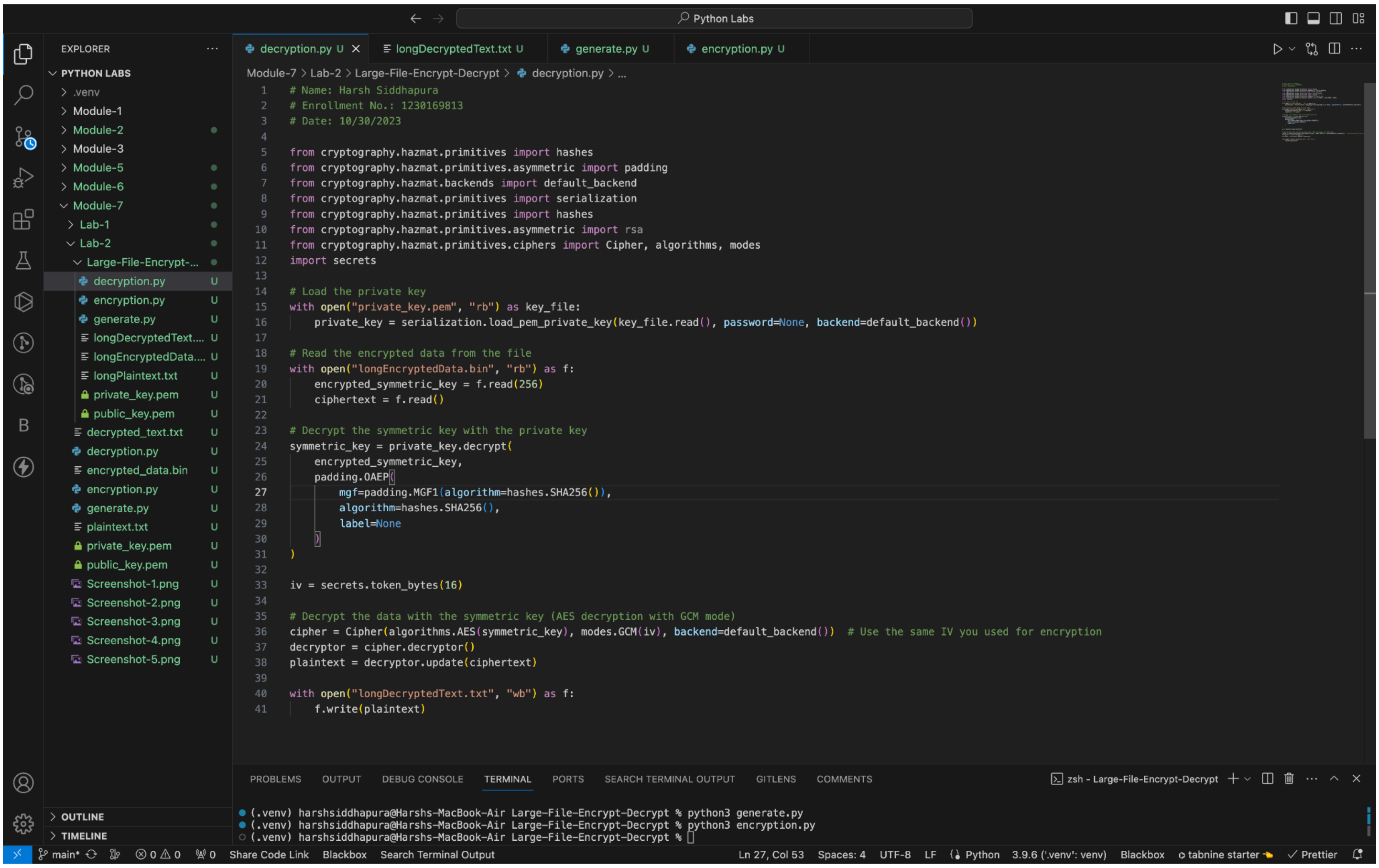

Python Labs
generate.py U
encryption.py U
decryption.py U
Module-7 > Lab-2 > decryption.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
# Read Private Key from File
with open("private_key.pem", "rb") as f:
    private_key = serialization.load_pem_private_key(
        f.read(),
        password=None,
        backend=default_backend()
)
Read Encrypted Data
with open("encrypted_data.bin", "rb") as f:
    encrypted_data = f.read()
# Decrypt Encrypted Data
decrypted_data = private_key.decrypt(
    encrypted_data,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
)
)
Save Decrypted Data to File
with open("decrypted_text.txt", "wb") as f:
    f.write(decrypted_data)
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 decryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 %
Ln 29, Col 6 Spaces: 4 UTF-8 LF Python 3.9.6 ('.venv': venv)

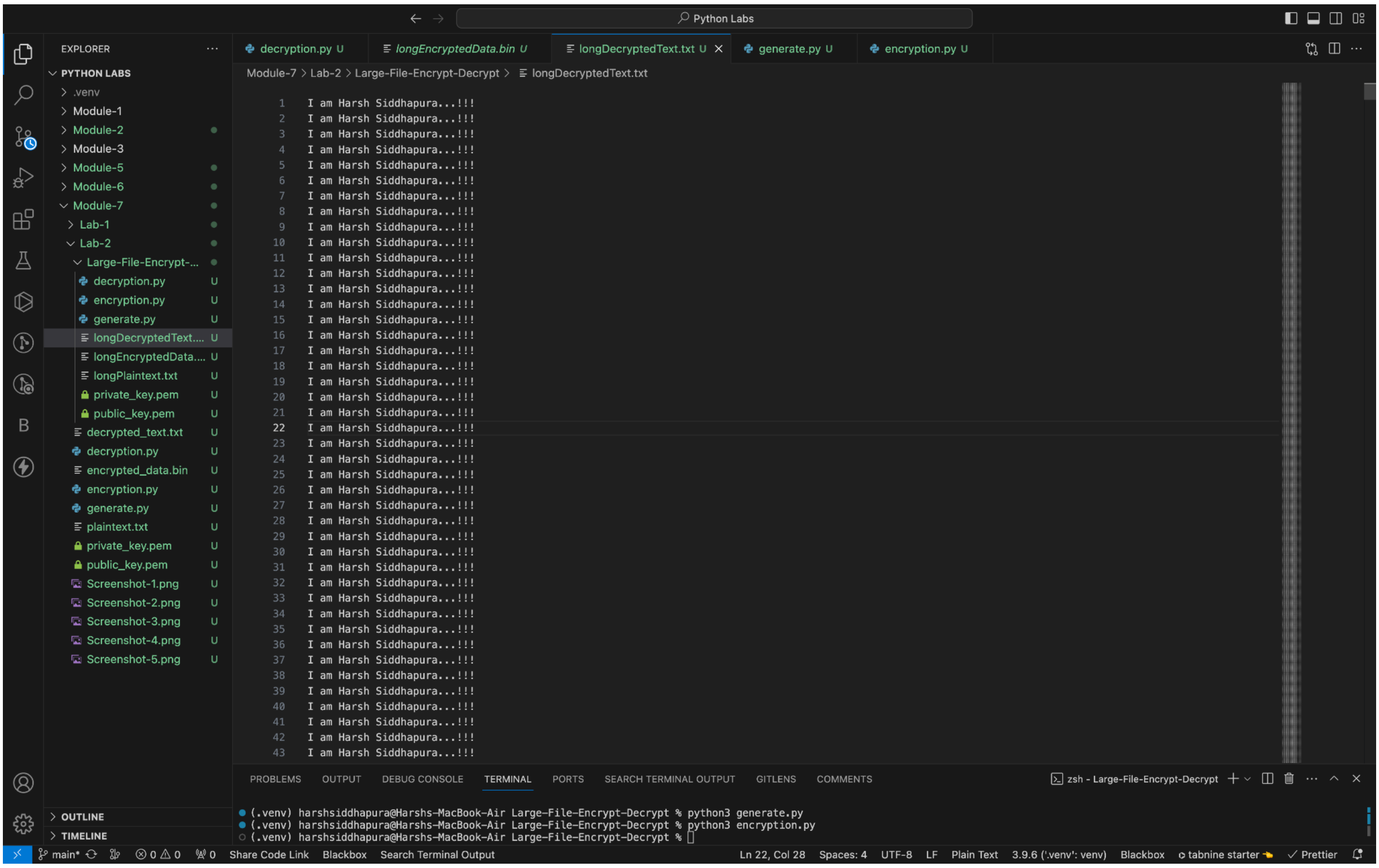

Python Labs
generate.py U
encryption.py U
decryption.py U
private_key.pem U
Module-7 > Lab-2 > private_key.pem
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCfNEw+Jt/M7tnD
TnXBy6ohKzlnR+J6fDVp1j0Qm6vw+F56Fhv/JhhfNeZwPKvOtcuZqSzv9IDkCLNF
xi/T5EtDBzs5dFSuyatKeioc/uYGk5/yhGp/hITi539u9h5ymE9TFz/y6GzLIGAK
w+MfgXpsxE+P6Ugzb4ducCsFHbH6ii201QVeLDD9WIiRvmVHH+dcTJ1DbevNhQ7c
frfcBAEJPPMcYrQrubz/B4Nos43S9kQ+zzdvF8D/Z5Fh4VfN45FiQsVKLRw+2gvi
ZzKZTBfuH8e7CCsXdC5bPItWJDvp09eUw5iIeXZKgG7brz2toZKTfphpoYR/DUcP
SYE4oDmFAgMBAAECggEALdJBEXBh4JtsTLSz2tPSAxrsOigr7KH8Q5p4V+PeQIxT
sOLVKwIdaKXcTbiAoI2dbHwPJX8ja75/TplEocFm0gNuTdKIHwiHRvfTUCYjMYhi
tEvdVRdgoegMOvUWfVgOGLWr3696xJnsWpzXrXU4Nb7owVxcvgUfHy+kiiPqWlff
lpdZdnKpwrSDhA975rJAw0lr+Y+Li4CdlI+2IkFjsYoG2Kwuu9rvKSfhG5hBGr3p
JS8eMmmX8OAfMGaCQCxypOJtbnH0YanRNLrbu7Z/n5NPV/BMlI87En1KV9SaKiLI
xBmDzZkX1fYWasAT6073kNDGnunI8IpWWaqi3+4ejwKBgQDReYUD+NX2yshDtdTM
juxxZYClBNtjb8ok58/3OcLMUkSRU8rbWO5S9AnocCu73gx2XFwnnZiyCPq3XtA2
Yp+XS+jYmSRwkxkY3nCLfIadk+9e/74S1rmHmrkhJX8YT0E2veO7dwh9toIPvs3j
TGAGb2gdxKjilvbLzXiwr5Fa7wKBgQDCkHamlQwRUF7ZxpWREAJWfYJKhHze2Hxd
3T1wIeze7X4L4KRlPOCIKh/BZmdcOHJmce2Ts/CHQDOCayrhlMJJSdXWqoOdMCbV
14/oPjPfvznc8AjhqWslOAJOvTy1zA5+Ws9VlAuMkLzHuiwCgN3WHFxa2YQgXzfc
8NiMzczCywKBgCoKaoDGkyKpxhGYz7GSSa/kgDIeTG2jhP1tLNbgV9Er/ajxGTh2
m7Y4Vji5Mn3M7dFfbfthbMDmfnMI617O4lq6GGxIKAET9KiCFDJadqXvh84YI6yi
PbAShG87mEtntJ3o9GiuVhzs4SOJ1sZ0kJ0aBNpcmsyHIdb7UOqdAd4vAoGAaQOq
Vv1JtyqnF2o9HkAC0wYJRgk+kNVOQyd/2r8k56pOB0En2lDCbBw4aZ3JGhiJfsvt
THSp1YaMhs8rJPiy6PXGSohcY4Iy4+bmkJbKTvEa3Spd0NctsrgJXtB7LTDHfeGr
2OgAV81jZ3YP/QXrNn62Rn7y9roOH92LNGx8IYMCgYEAkCe8GBX3wmZ1dv7snf5R
rWXFdmDydyo+wuRTJjh4F0kqZLEDb88/zAOfw4pQQLSvqABNoGbRPhSTsUjW+aFB
BsIMOB7Ee4MBgPbZlXfuZK1eu20Asq2QHtAH8l0HzWK9ff8bi0CEVVG6QJ2ICnaq
Bw7WXQ9CfD2sKr/SXKf+Hik=
-----END PRIVATE KEY-----
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 decryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 %
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text

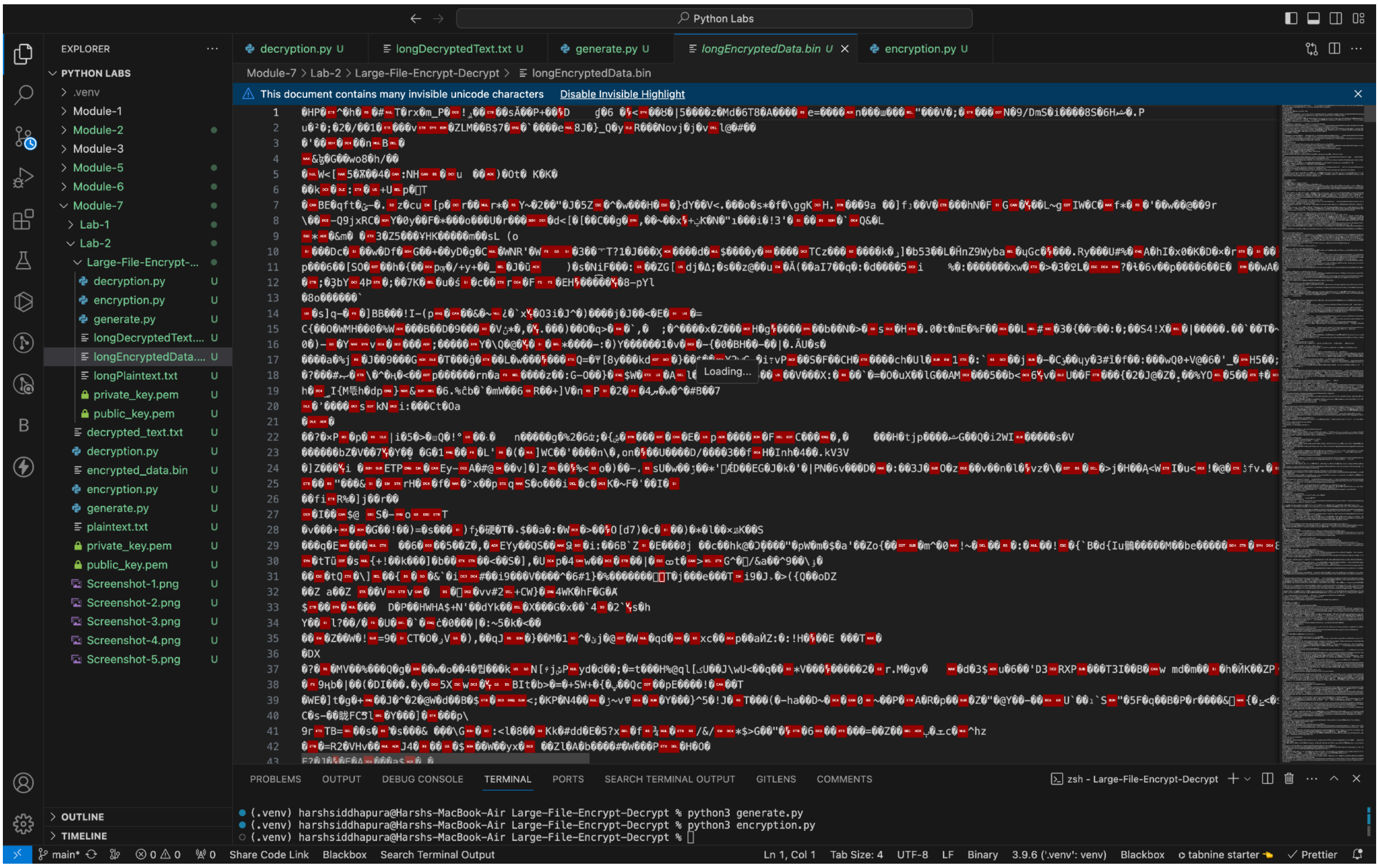

Python Labs
EXPLORER
PYTHON LABS
.venv
Module-1
Module-2
Module-3
Module-5
Module-6
Module-7
Lab-1
Lab-2
decrypted_text.txt
decryption.py
encrypted_data.bin
encryption.py
generate.py
plaintext.txt
private_key.pem
public_key.pem
generate.py U
encryption.py U
decryption.py U
public_key.pem U
Module-7 > Lab-2 > public_key.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnzRMPibfzO7Zw051wcuq
ISs5Z0fienw1adY9EJur8PheehYb/yYYXzXmcDyrzrXLmaks7/SA5AizRcYv0+RL
Qwc7OXRUrsmrSnoqHP7mBpOf8oRqf4SE4ud/bvYecphPUxc/8uhsyyBgCsPjH4F6
bMRPj+lIM2+HbnArBR2x+oottNUFXiww/ViIkb5lRx/nXEydQ23rzYUO3H633AQB
CTzzHGK0K7m8/weDaLON0vZEPs83bxfA/2eRYeFXzeORYkLFSi0cPtoL4mcymUwX
7h/HuwgrF3QuWzyLViQ76dPXlMOYiHl2SoBu2689raGSk36YaaGEfw1HD0mBOKA5
hQIDAQAB
-----END PUBLIC KEY-----
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - Lab-2
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 % python3 decryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Lab-2 %
OUTLINE
TIMELINE
main* Share Code Link Blackbox Search Terminal Output
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text Blackbox tabnine starter Prettier


Python Labs
EXPLORER
PYTHON LABS
.venv
Module-1
Module-2
Module-3
Module-5
Module-6
Module-7
Lab-1
Lab-2
Large-File-Encrypt-...
decryption.py
encryption.py
generate.py
longDecryptedText....
longEncryptedData....
longPlaintext.txt
private_key.pem
public_key.pem
decrypted_text.txt
decryption.py
encrypted_data.bin
encryption.py
generate.py
plaintext.txt
private_key.pem
public_key.pem
Screenshot-1.png
Screenshot-2.png
Screenshot-3.png
Screenshot-4.png
Screenshot-5.png
decryption.py U
longDecryptedText.txt U
generate.py U
encryption.py U
Module-7 > Lab-2 > Large-File-Encrypt-Decrypt > generate.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
# Generate Private Key
private_key = rsa.generate_private_key(
public_exponent=65537,
key_size=2048,
backend=default_backend()
)
Save Private Key to File
with open("private_key.pem", "wb") as f:
f.write(private_key.private_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PrivateFormat.PKCS8,
encryption_algorithm=serialization.NoEncryption()
))
# Generate Public Key from Private Key
public_key = private_key.public_key()
# Save Public Key to File
with open("public_key.pem", "wb") as f:
f.write(public_key.public_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PublicFormat.SubjectPublicKeyInfo
))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - Large-File-Encrypt-Decrypt
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt %
OUTLINE
TIMELINE
main* Share Code Link Blackbox Search Terminal Output
Ln 32, Col 7 Spaces: 4 UTF-8 LF Python 3.9.6 ('.venv': venv) Blackbox tabnine starter Prettier


Python Labs
Module-7 > Lab-2 > Large-File-Encrypt-Decrypt > encryption.py
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import secrets
# Generate or load RSA key pair
private_key = None
public_key = None
try:
with open("private_key.pem", "rb") as key_file:
private_key = serialization.load_pem_private_key(key_file.read(), password=None, backend=default_backend())
with open("public_key.pem", "rb") as key_file:
public_key = serialization.load_pem_public_key(key_file.read(), backend=default_backend())
except FileNotFoundError:
# Generate new keys if not found
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048, backend=default_backend())
public_key = private_key.public_key()
with open("private_key.pem", "wb") as key_file:
key_file.write(private_key.private_bytes(encoding=serialization.Encoding.PEM, format=serialization.PrivateFormat.PKCS8, encryption_algorithm=serializat
with open("public_key.pem", "wb") as key_file:
key_file.write(public_key.public_bytes(encoding=serialization.Encoding.PEM, format=serialization.PublicFormat.SubjectPublicKeyInfo))
# Read Plaintext File
with open("longPlaintext.txt", "rb") as f:
plaintext = f.read()
# Generate a symmetric key for data encryption
symmetric_key = secrets.token_bytes(32) # You'll need to import the 'secrets' module
iv = secrets.token_bytes(16)
# Encrypt data with the symmetric key (AES encryption)
cipher = Cipher(algorithms.AES(symmetric_key), modes.GCM(iv), backend=default_backend())
encryptor = cipher.encryptor()
ciphertext = encryptor.update(plaintext) + encryptor.finalize()
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt %


Python Labs
Module-7 > Lab-2 > Large-File-Encrypt-Decrypt > decryption.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import secrets
# Load the private key
with open("private_key.pem", "rb") as key_file:
private_key = serialization.load_pem_private_key(key_file.read(), password=None, backend=default_backend())
# Read the encrypted data from the file
with open("longEncryptedData.bin", "rb") as f:
encrypted_symmetric_key = f.read(256)
ciphertext = f.read()
# Decrypt the symmetric key with the private key
symmetric_key = private_key.decrypt(
encrypted_symmetric_key,
padding.OAEP(
mgf=padding.MGF1(algorithm=hashes.SHA256()),
algorithm=hashes.SHA256(),
label=None
)
)
iv = secrets.token_bytes(16)
# Decrypt the data with the symmetric key (AES decryption with GCM mode)
cipher = Cipher(algorithms.AES(symmetric_key), modes.GCM(iv), backend=default_backend()) # Use the same IV you used for encryption
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext)
with open("longDecryptedText.txt", "wb") as f:
f.write(plaintext)
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt %


Python Labs
decryption.py
longEncryptedData.bin
longDecryptedText.txt
generate.py
encryption.py
Module-7 > Lab-2 > Large-File-Encrypt-Decrypt > longDecryptedText.txt
I am Harsh Siddhapura...!!!
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt %
Ln 22, Col 28 Spaces: 4 UTF-8 LF Plain Text


Python Labs
decryption.py
longDecryptedText.txt
generate.py
longEncryptedData.bin
encryption.py
Module-7 > Lab-2 > Large-File-Encrypt-Decrypt > longEncryptedData.bin
This document contains many invisible unicode characters
Disable Invisible Highlight
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 generate.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt % python3 encryption.py
(.venv) harshsiddhapura@Harshs-MacBook-Air Large-File-Encrypt-Decrypt %
Ln 1, Col 1 Tab Size: 4 UTF-8 LF Binary