ADVANCED ACADEMIC CENTER

A CENTER FOR INTER-DISCIPLINARY RESEARCH

2020-2021

TITLE

"FILE ENCRYPTION AND DECRYPTION"

GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
AUTONOMOUS

Advanced Academic Center

(A Center For Inter-Disciplinary Research)

This is to certify that the project titled

“FILE ENCRYPTION AND DECRYPTRION”

is a bonafide work carried out by the following students in partial fulfilment of the requirements for Advanced Academic Center intern, submitted to the chair, AAC during the academic year 2020-2021.

NAME	**ROLL NO.**	**BRANCH**
MAHAJAN VAMSHI SAI	19241A12F8	IT-C
KANNAN VIJAYARAGHAVAN	19241A05V9	CSE-F

This work was not submitted or published earlier for any study.

Dr.B.R.K.Reddy
Program Coordinator

Dr.Ramamurthy Suri
Associate Dean, AAC

ACKNOWLEDGEMENTS

We express our deep sense of gratitude to our respected Director, Gokaraju Rangaraju Institute of Engineering and Technology, for the valuable guidance and for permitting us to carry out this project.

With immense pleasure, we extend our appreciation to our respected Principal, for permitting us to carry out this project.

We are thankful to the Associate Dean, Advanced Academic Centre, for providing us an appropriate environment required for the project completion. We are grateful to our project supervisor who spared valuable time to influence us with their novel insights.

We are indebted to all the above-mentioned people without whom we would not have concluded the project.

TABLE OF CONTENTS

ABSTRACT:

Cryptography assumes a vital job in security of information transmission. We also use Mutual Authentication process to satisfy all services in Cryptography i.e., Access Control, Confidentiality, Integrity, Authentication. In this way we can maintain the data more securely. Technically cryptography is the art of writing and solving codes. But the whole idea evolved when people started to use it in security. In this world where data is enormously generated, there must be an aspect of privacy when two individuals communicate or share data between them. In this situations where the need of security is required on the information passed down among people, these words of information security like cryptography, encryption, decryption and etc.., come into picture. The process of converting the plain text in to cipher text is called as encryption.

By this methods of cybersecurity one can fully give a pinch of confidentiality required on the data being shared. So by using many inbuilt libraries and algorithms which people developed keeping the safety of data in mind we can build different encrypting software and programs differing the features required by the one who prepares it. We can distinct our program with our personal key where the key will never leave our hand. The approach introduced in this project makes us build a program using python provided library to encrypt a file and also which can decrypt a previously encrypted file with the key provided. Such that any other person in the network cannot access the data present in the network. Only the sender and receiver can retrieve the message from the data who are actually in the possession of the unique key.

INTRODUCTION:

CRYPTOGRAPHY:

Cryptography is practice of using different techniques for secure communication. It is the process of constructing algorithms and protocols to prevent third parties or public from reading private messages or messages with confidential information.

Modern cryptography concerns with:

Confidentiality - Information cannot be understood by anyone.

Integrity - Information cannot be altered.

Non-repudiation - Sender cannot deny his/her intentions in the transmission of the information at a later stage.

Authentication - Sender and receiver can confirm each.

The terms like encryption, decryption and key are the basic blocks of cryptography. When put in simple terms cryptography is a method to change a plain text(human readable form) into cipher text(coded form). For achieving this we use different encryption algorithms like AES, SHA-256, Base64 etc.., which covert the data to be shared into cipher form.
A complete scheme has five ingredients namely :

- **Plain text:** This is the original message or data that is fed into algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plain text.
- **Secret Key:** The secret key is also input to the algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Cipher text:** This is the scrambled message produced as the output. It depends on the plain text and the secret key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This is essentially encryption algorithm run in reverse. It takes the ciphertext and the same secret key and produces the original plaintext.

Three types of cryptographic techniques used in general.

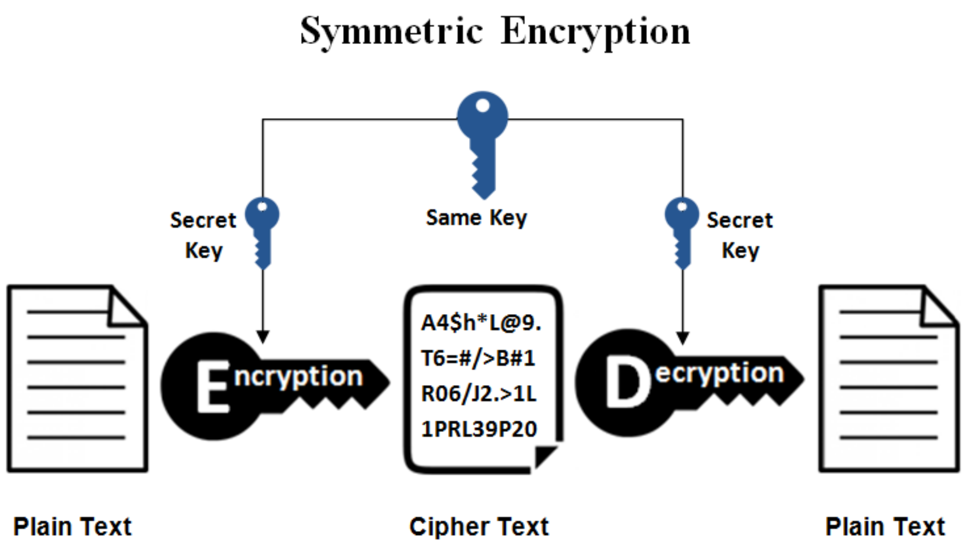
1. Symmetric-key cryptography

2. Hash functions.

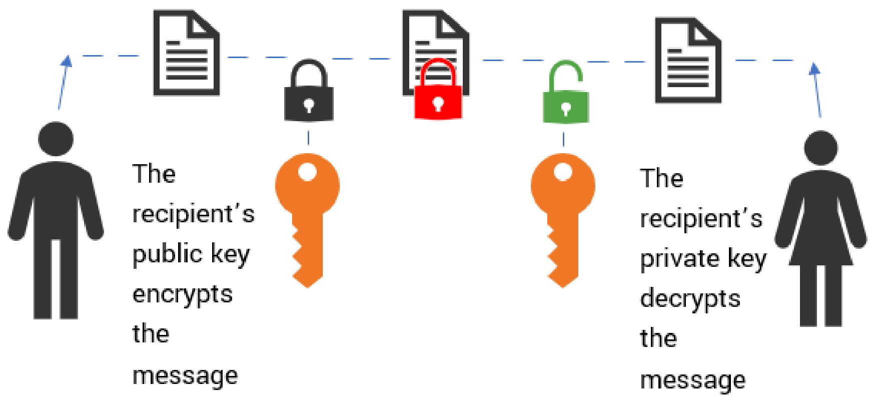
3. Public-key cryptography

Symmetric-key Cryptography: Both the sender and receiver share a single key. The sender uses this key to encrypt plaintext and send the cipher text to the receiver. On the other side the receiver applies the same key to decrypt the message and recover the plain text.

Examples include AES(Advanced Encryption Standard), DES(Data Encryption Standard), RC4(Rivest Cipher 4)…

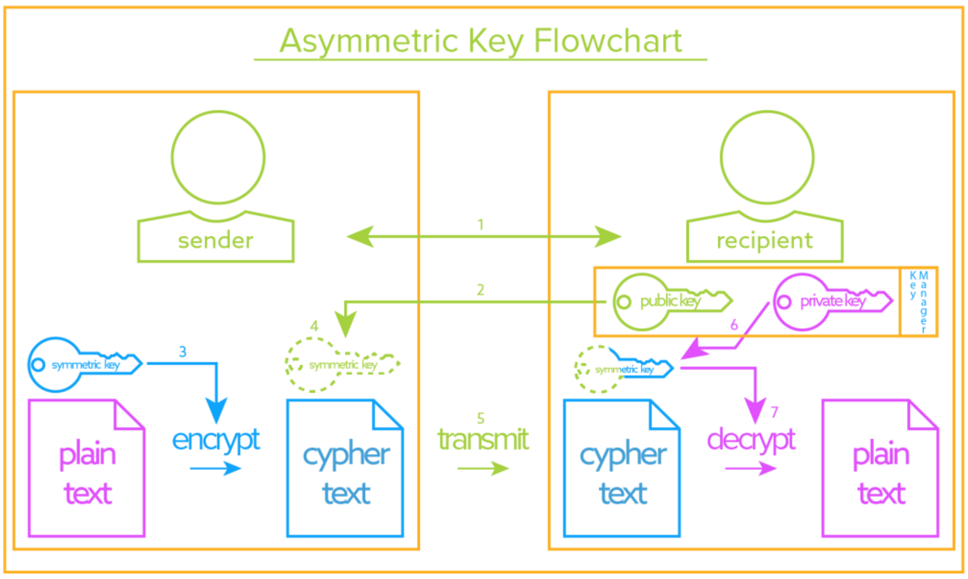


Public-Key Cryptography: This is the most revolutionary concept in the last 300-400 years. In Public-Key Cryptography two related keys (public and private key) are used. Public key may be freely distributed, while its paired private key, remains a secret. The public key is used for encryption and for decryption private key is used.


The recipient's public key encrypts the message
The recipient's private key decrypts the message

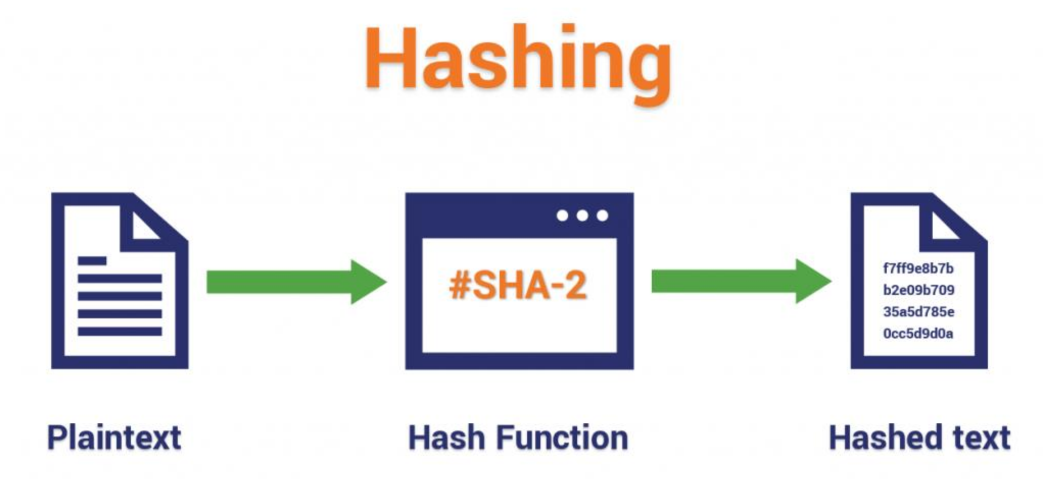
This method can also be called a asymmetric encryption format as there is no symmetry in the key used by the both sides of transmission for decrypting.

Examples are RSA(Rivest Shamir Adlemen), ECC(Elliptical Curve Cryptography), Diffie-Hellman…



Hash Functions: No key is used in this algorithm. A fixed-length hash value is computed as per the plain text that makes it impossible for the contents of the plain text to be recovered. Hash functions are also used by many operating systems to encrypt passwords.

Examples for different hashing algorithms are MD5, SHA-1, SHA-2, Base64 ….



PROJECT WORKFLOW

Software used :

This program can be executed in any software which can run python code. For this program specifically we don’t need high end software. We can limit ourselves to a Python IDE like PYCHARM (by Jet Brains).

Modules used :

CRYPTOGRAPHY

`Cryptography` includes both high level recipes and low level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.
The command to install the module is

```
pip install cryptography
```

Keyword used :

FERNET

Fernet guarantees that a message encrypted using it cannot be manipulated or read without the key. Fernet is an implementation of symmetric (also known as “secret key”) authenticated cryptography. Fernet also has support for implementing key rotation via Multi Fernet.
A fernet key is used to encrypt and decrypt fernet tokens. Each key is actually composed of two smaller keys: a 128-bit AES encryption key and a 128-bit SHA256 HMAC signing key. The keys are held in a key repository that keystone passes to a library that handles the encryption and decryption of tokens.

Commands used :

Rb command

rb :
Opens the file as read-only in binary format and starts reading from the beginning of the file. While binary format can be used for different purposes, it is usually used when dealing with things like images, videos, etc. r+ : Opens a file for reading and writing, placing the pointer at the beginning of the file.

Wb command

The wb indicates that **the file is opened for writing in binary mode**. When writing in binary mode, Python makes no changes to data as it is written to the file.

STEPS TO FOLLOW :

- Installing modules
- Creating a key

- Loading a key
- Encrypting a file
- Decrypting a file

Installing modules

```python
from cryptography.fernet import Fernet
```

Creating a key

```python
key = Fernet.generate_key()

with open('mykey.key', 'wb') as mykey:
    mykey.write(key)
```

Now, let’s create the key and save it in the same folder as our data file:

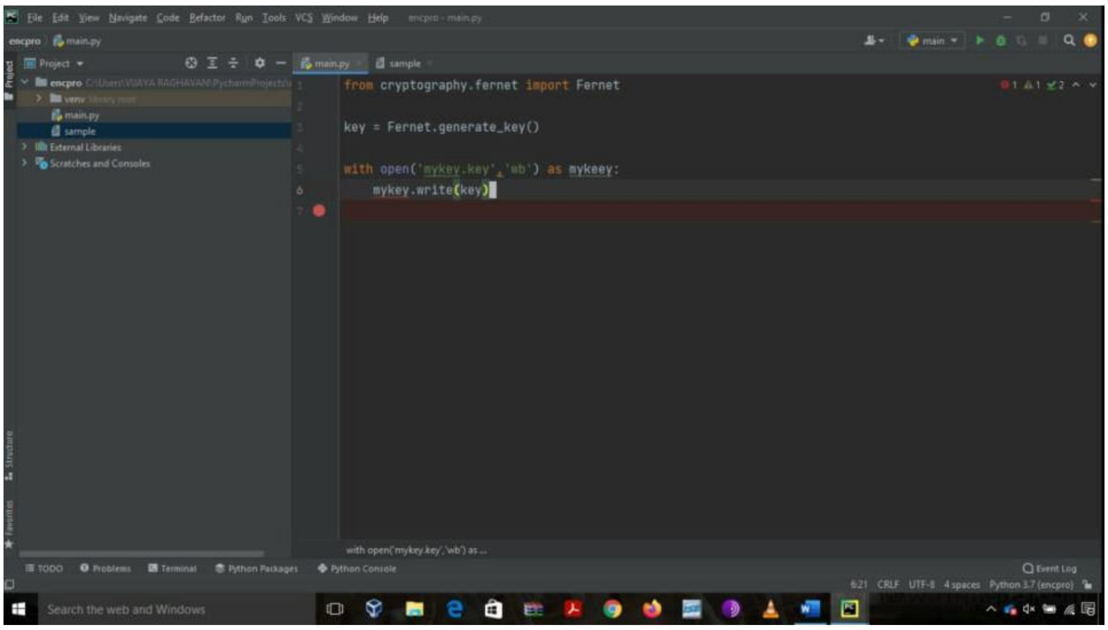
```python
with open('mykey.key', 'rb') as mykey:
    key = mykey.read()

print(key)
```

If you check the directory where you Python code is located, you should see the **mykey.key** file. You can open it with any text editor . The file should contain one line which is a string of some order of characters. In this program it is : #KEY

Loading a key

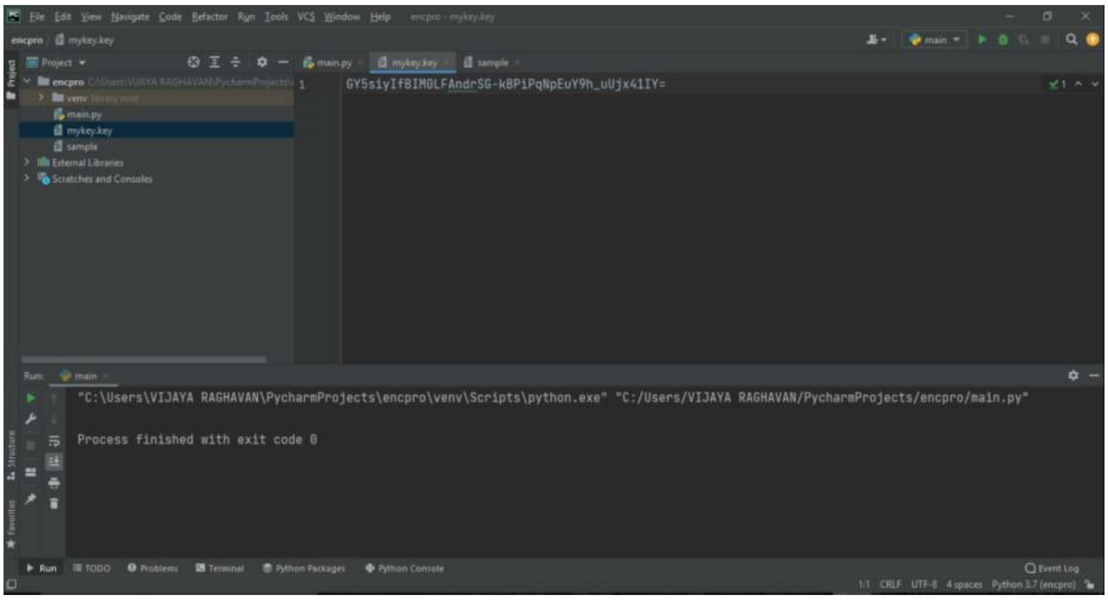
After we generated the encryption key, we would need to load it into our environment in order to encrypt/decrypt the files.

The following step is very simple, and requires to just open the mykey.key file and store it in local memory:



```python
from cryptography.fernet import Fernet

key = Fernet.generate_key()

with open('mykey.key','wb') as mykeey:
    mykey.write(key)
```

And just to verify, we will see the following output:



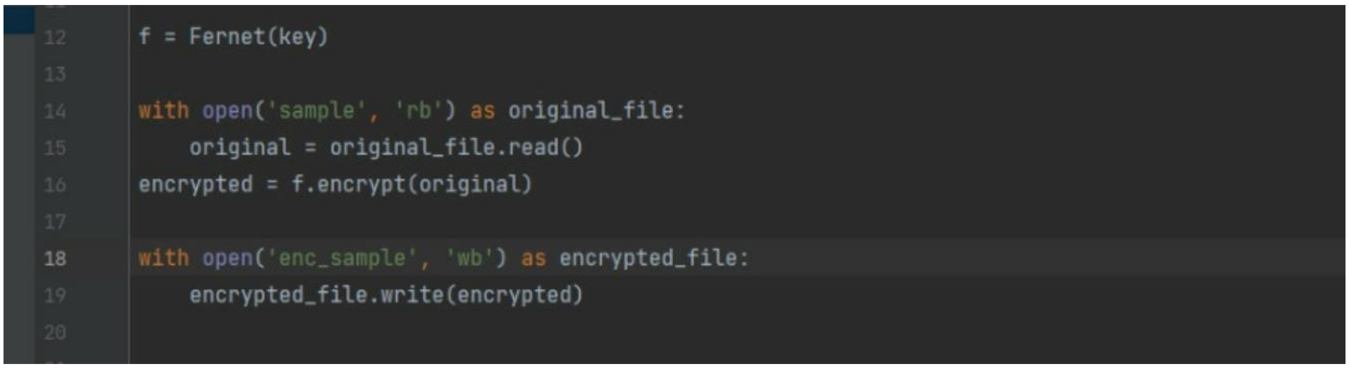
```
GY5siyIfBIM0LFAndrSG-kBPiPqNpEuY9h_uUjx41IY=
```

```
"C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro\venv\Scripts\python.exe" "C:/Users/VIJAYA RAGHAVAN/PycharmProjects/encpro/main.py"

Process finished with exit code 0
```

The encryption key is now stored locally as the **key** variable.

Encrypting a File

Now that we have the file to encrypt and the encryption key, we will now write a function to utilize these and return the encrypted file:

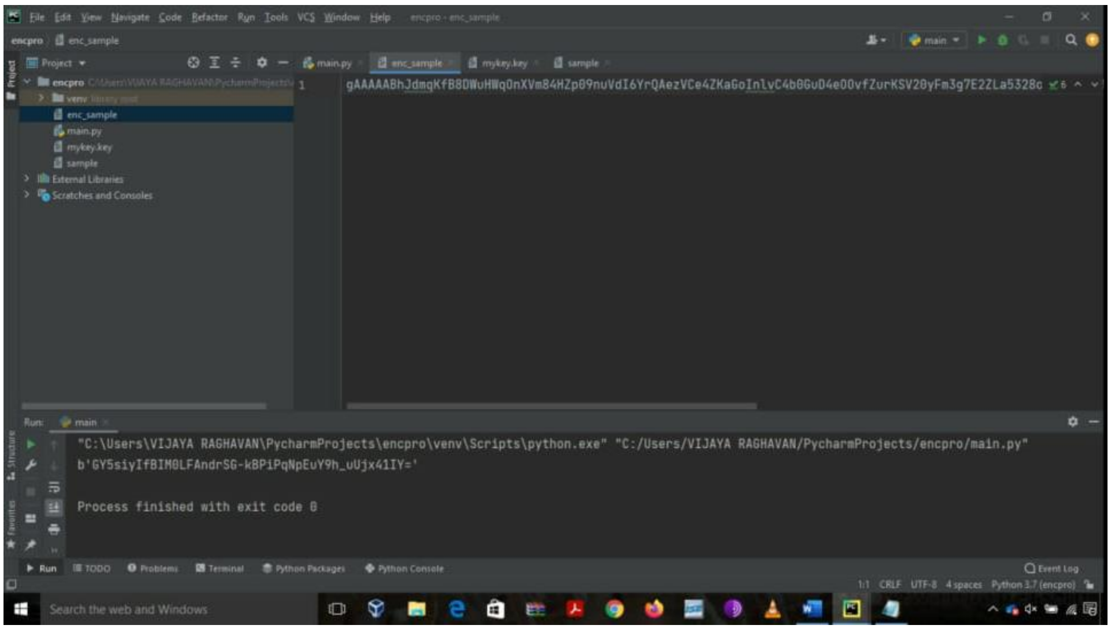


```
f = Fernet(key)

with open('sample', 'rb') as original_file:
    original = original_file.read()
encrypted = f.encrypt(original)

with open('enc_sample', 'wb') as encrypted_file:
    encrypted_file.write(encrypted)
```

Let’s discuss what we did here:

- We initialize the Fernet object as store is as a local variable **f**
- Next, we read our original data (sample file) into **original**
- Then we encrypt the data using the Fernet object and store it as **encrypted**
- And finally, we write it into a new .csv file called “enc_sample”

You can take a look at the encrypted file here:


File Edit View Navigate Code Refactor Run Tools VCS Window Help encpro - enc_sample
encpro enc_sample
Project
main.py enc_sample mykey.key sample
encpro
venv
enc_sample
main.py
mykey.key
sample
External Libraries
Scratches and Consoles
Run: main
Run TODO Problems Terminal Python Packages Python Console Event Log
1:1 CRLF UTF-8 4 spaces Python 3.7 (encpro)
Search the web and Windows

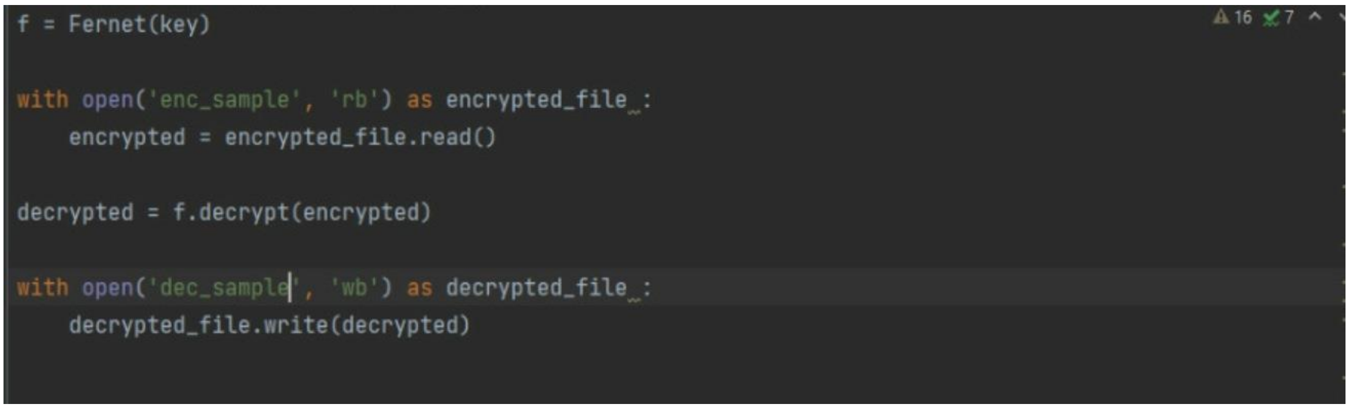
```
gAAAAABhJdmqKfB8DWuHWq0nXVm84HZp09nuVdI6YrQAezVCe4ZKaGoInlvC4b0GuD4e00vfZurKSV20yFm3g7E2ZLa5328c
```

```
"C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro\venv\Scripts\python.exe" "C:/Users/VIJAYA RAGHAVAN/PycharmProjects/encpro/main.py"
b'GY5siyIfBIM0LFAndrSG-kBPiPqNpEuY9h_uUjx41IY='

Process finished with exit code 0
```

Decrypting a File

After you encrypted the file and, for example, successfully transferred the file to another location, you will want to access it. Now, that data is in the encrypted format. The next step is to decrypt it back to the original content.

The process we will follow now is the reverse of the encryption in the previous part. Exactly the same process, but now we will go from encrypted file to decrypted file

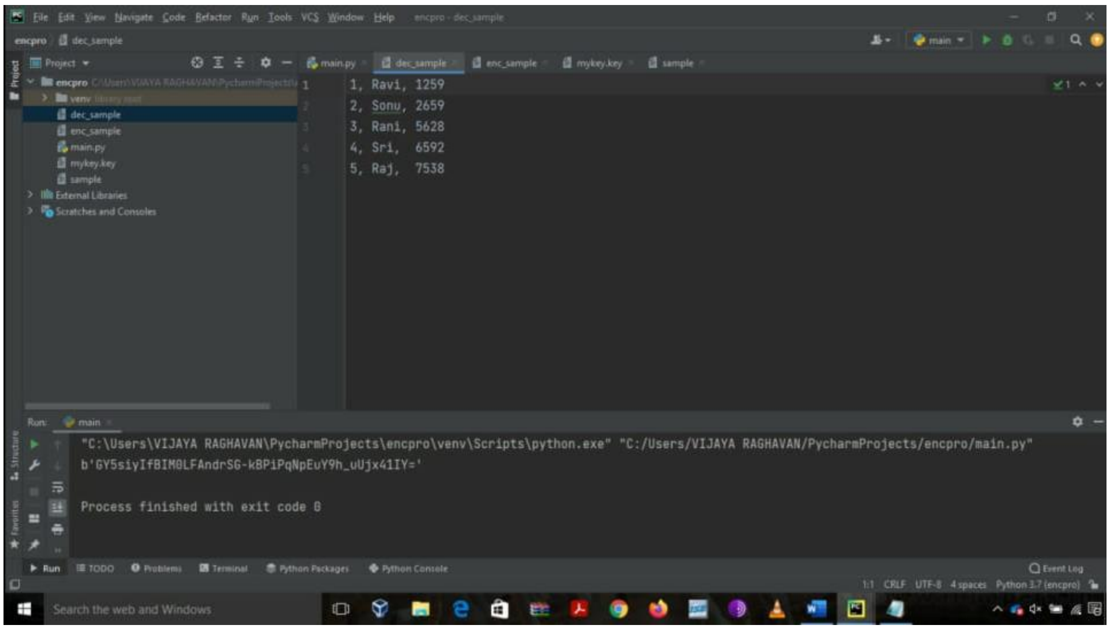


```
f = Fernet(key)

with open('enc_sample', 'rb') as encrypted_file:
    encrypted = encrypted_file.read()

decrypted = f.decrypt(encrypted)

with open('dec_sample', 'wb') as decrypted_file:
    decrypted_file.write(decrypted)
```

Let’s discuss what we did here:

- We initialize the Fernet object as store is as a local variable **f**
- Next, we read our encrypted data (enc_sample file) into **encrypted**
- Then we decrypt the data using the Fernet object and store it as **decrypted**
- And finally, we write it into a new .csv file called “dec_sample”

You can take a look at the decrypted file here:



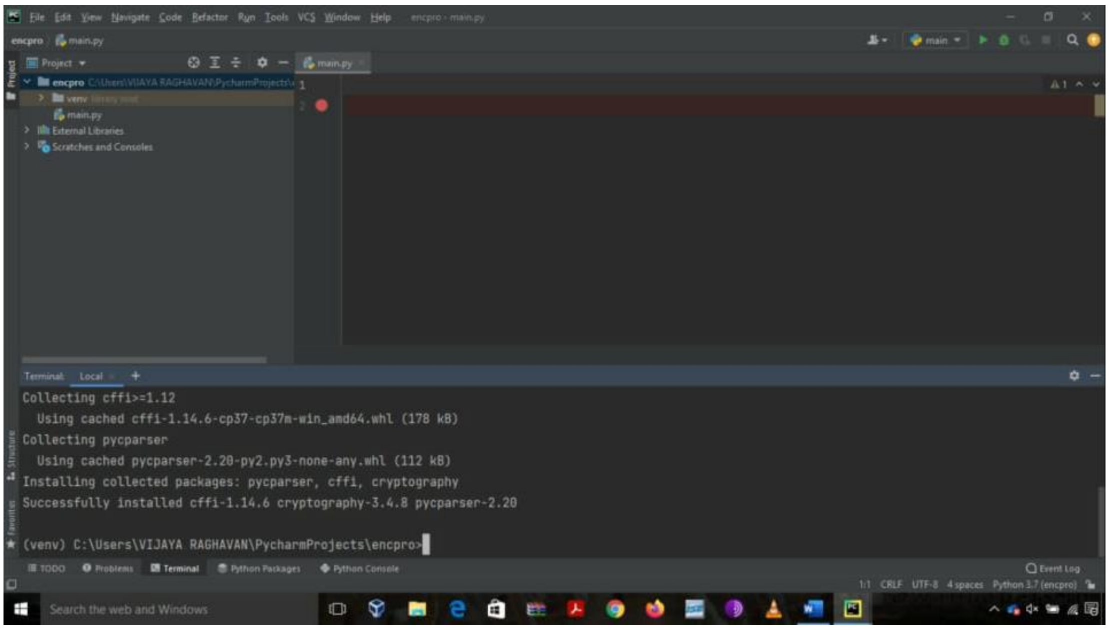
```
1, Ravi, 1259
2, Sonu, 2659
3, Rani, 5628
4, Sri,  6592
5, Raj,  7538
```

```
"C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro\venv\Scripts\python.exe" "C:/Users/VIJAYA RAGHAVAN/PycharmProjects/encpro/main.py"
b'GY5siyIfBIM0LFAndrSG-kBPiPqNpEuY9h_uUjx41IY='

Process finished with exit code 0
```

Comparing “dec_sample” with the original “sample”, you will see that in fact these two have identical contents. Our encryption/decryption process was successful.

IMPLEMENTATION

CODE:



```
Collecting cffi>=1.12
  Using cached cffi-1.14.6-cp37-cp37m-win_amd64.whl (178 kB)
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Installing collected packages: pycparser, cffi, cryptography
Successfully installed cffi-1.14.6 cryptography-3.4.8 pycparser-2.20

(venv) C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro>
```

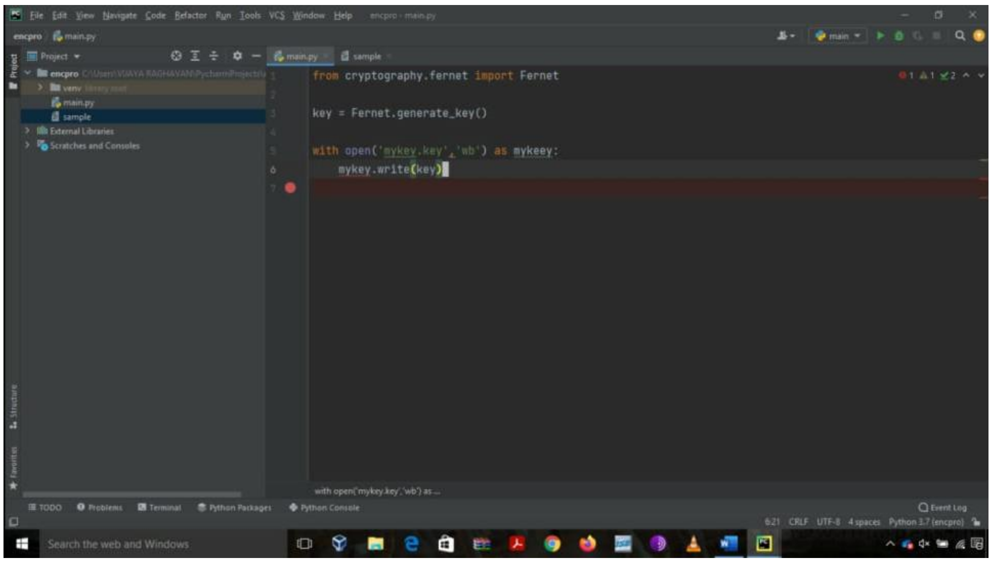
from cryptography.fernet import Fernet

Creating the key for the text file

key = Fernet.generate_key()

with open('mykey.key', 'wb') as mykey :

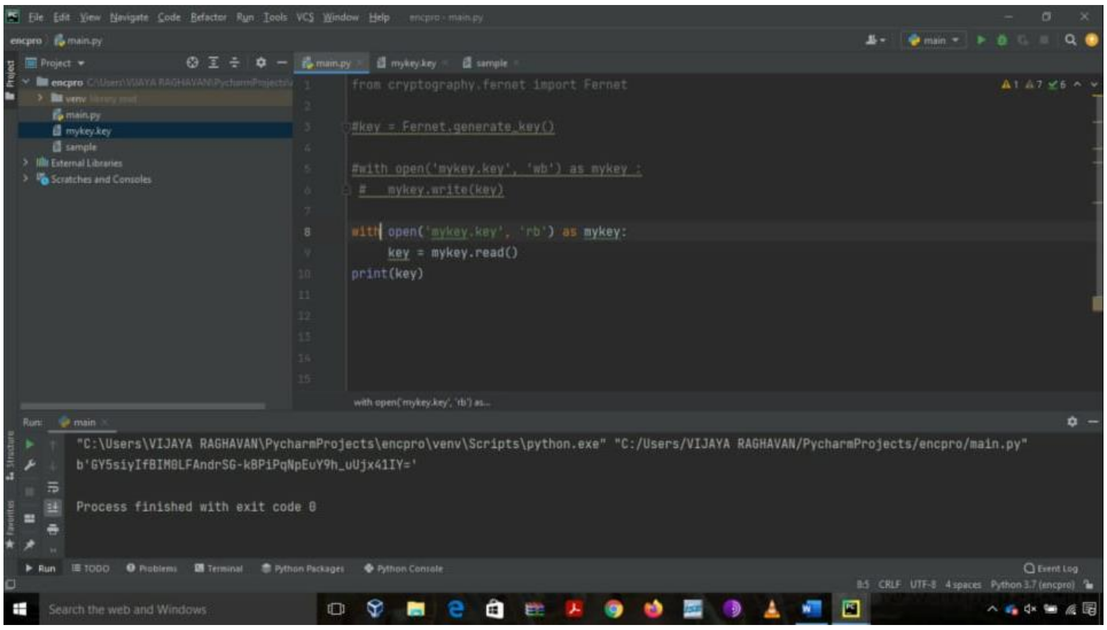
    mykey.write(key)



```
from cryptography.fernet import Fernet

key = Fernet.generate_key()

with open('mykey.key','wb') as mykeey:
    mykey.write(key)
```

Loading a key for the file

```
with open('mykey.key', 'rb') as mykey :

    key = mykey.read()



print(key)
```



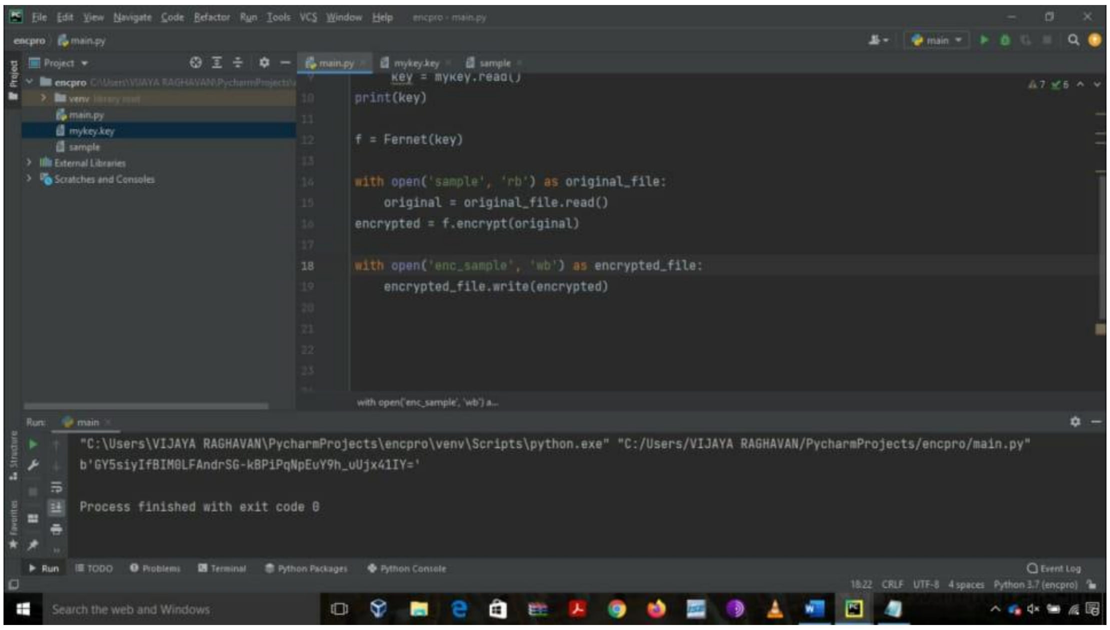
```
from cryptography.fernet import Fernet

#key = Fernet.generate_key()

#with open('mykey.key', 'wb') as mykey :
 #   mykey.write(key)

with open('mykey.key', 'rb') as mykey:
    key = mykey.read()
print(key)
```

```
"C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro\venv\Scripts\python.exe" "C:/Users/VIJAYA RAGHAVAN/PycharmProjects/encpro/main.py"
b'GY5siyIfBIM0LFAndrSG-kBPiPqNpEuY9h_uUjx41IY='

Process finished with exit code 0
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
encpro - main.py
encpro
main.py
Project
mykey.key
sample
venv
External Libraries
Scratches and Consoles
with open('mykey.key', 'rb') as...
Run: main
Run TODO Problems Terminal Python Packages Python Console
Event Log
8:5 CRLF UTF-8 4 spaces Python 3.7 (encpro)
Search the web and Windows

Encryption

```
 f = Fernet(key)



 with open('sample', 'rb') as original_file :

     original = original_file.read()

 encrypted = f.encrypt(original)



 with open('enc_sample', 'wb') as encrypted_file :

     encrypted_file.write(encrypted)
```



```
    key = mykey.read()
print(key)

f = Fernet(key)

with open('sample', 'rb') as original_file:
    original = original_file.read()
encrypted = f.encrypt(original)

with open('enc_sample', 'wb') as encrypted_file:
    encrypted_file.write(encrypted)
```

```
"C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro\venv\Scripts\python.exe" "C:/Users/VIJAYA RAGHAVAN/PycharmProjects/encpro/main.py"
b'GY5siyIfBIM0LFAndrSG-kBPiPqNpEuY9h_uUjx41IY='

Process finished with exit code 0
```

Decryption

f = Fernet(key)

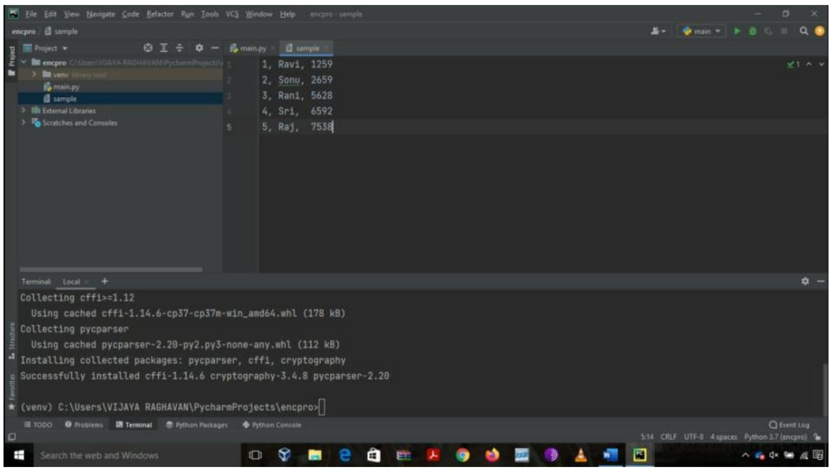
with open('enc_sample', 'rb') as encrypted_file :

    encrypted = encrypted_file.read()

decrypted = f.decrypt(encrypted)

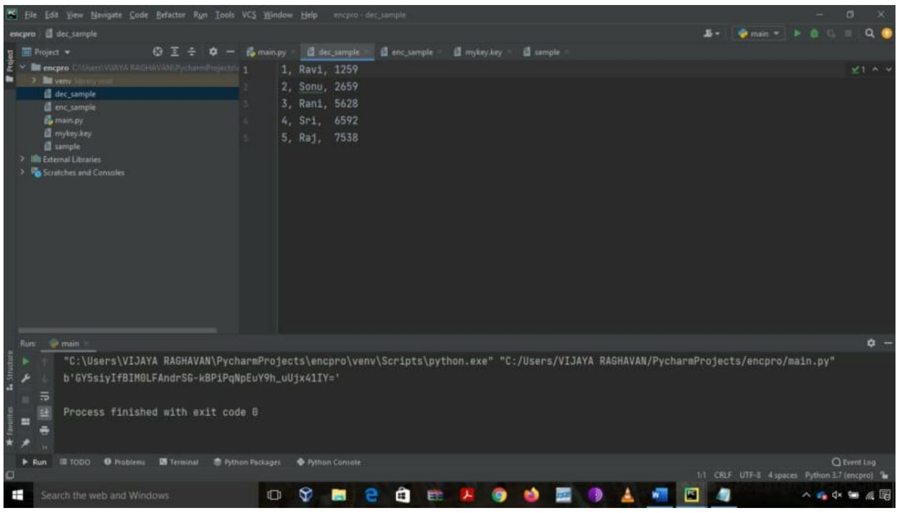
with open('dec_sample', 'wb') as decrypted_file :

    decrypted_file.write(decrypted)



```
1, Ravi, 1259
2, Sonu, 2659
3, Rani, 5628
4, Sri,  6592
5, Raj,  7538
```

```
Collecting cffi>=1.12
  Using cached cffi-1.14.6-cp37-cp37m-win_amd64.whl (178 kB)
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Installing collected packages: pycparser, cffi, cryptography
Successfully installed cffi-1.14.6 cryptography-3.4.8 pycparser-2.20

(venv) C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro>
```



```
1, Ravi, 1259
2, Sonu, 2659
3, Rani, 5628
4, Sri,  6592
5, Raj,  7538
```

```
"C:\Users\VIJAYA RAGHAVAN\PycharmProjects\encpro\venv\Scripts\python.exe" "C:/Users/VIJAYA RAGHAVAN/PycharmProjects/encpro/main.py"
b'GY5siyIfBIM0LFAndrSG-kBPiPqNpEuY9h_uUjx41IY='

Process finished with exit code 0
```

Object Oriented programming code

```
class Encryptor():


    def key_create(self):

        key = Fernet.generate_key()

        return key


    def key_write(self, key, key_name):

        with open(key_name, 'wb') as mykey:

            mykey.write(key)


    def key_load(self, key_name):

        with open(key_name, 'rb') as mykey:

            key = mykey.read()

        return key




    def file_encrypt(self, key, original_file, encrypted_file):
        f = Fernet(key)

        with open(original_file, 'rb') as file:
            original = file.read()

        encrypted = f.encrypt(original)

        with open (encrypted_file, 'wb') as file:
            file.write(encrypted)

    def file_decrypt(self, key, encrypted_file, decrypted_file):

        f = Fernet(key)

        with open(encrypted_file, 'rb') as file:
            encrypted = file.read()

        decrypted = f.decrypt(encrypted)

        with open(decrypted_file, 'wb') as file:
            file.write(decrypted)
encryptor=Encryptor()

mykey=encryptor.key_create()
```

```
encryptor.key_write(mykey, 'mykey.key')


loaded_key=encryptor.key_load('mykey.key')


encryptor.file_encrypt(loaded_key, 'sample', 'enc_sample)


encryptor.file_decrypt(loaded_key, 'enc_sample, 'dec_sample’)
```

FUTURE DEVELOPMENTS:

The concept of encryption and decryption is the base for any secure communication. This concept of encrypting a file can be used for higher levels of encryption (Ex: A Network).
Cryptography is an important concept and will always be used in Cybersecurity field for security and protection. My team will work on cryptography and try to give good algorithms that cannot be easily decoded, which will help and make internet a better place.

CONCLUSION:

In this project we have worked with the required libraries of python. We have taken a sample .csv file and passed through the program written. This has encrypted our file using a generated key.
We have also performed the decryption of the encrypted file using the same key. The file generated was compared with the original file and noted that they are both the same. In this way we were successful in illustrating a program which performs **Symmetric Encryption.**
Hence, we conclude that our project can perfectly perform encryption on given file in turn increasing its security.

REFERENCES:

Books:

- Network Security Essentials (William Stallings)
- Encryption and Decryption in digital Communications(Benard Sklar)
- Handbook of Applied Cryptography(Alfred J Menezes)
- Public-Key Cryptography: Theory and Practice(A Das and C E Veni Madhavan)