National College of Ireland

Configuration Manual

MSc Research Project

MSc in Cloud Computing

Suyash Tripathi

Student ID:x21121443

School of Computing

National College of Ireland

Supervisor: Mr.Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Suyash Tripathi
Student ID:	X21121443
Programme:	MSc in Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Mr. Vikas Sahni
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	670
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Suyash Tripathi
Date:	14th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Suyash Tripathi
X21121443

1 Introduction

The paper provides instructions on how to correctly recreate the project. The project is composed of Python programming.

2 System Requirements

RAM – 4 GB
OS – Window 7 to 10
Python version – 3.8 or above
AWS account – for storing the file
Mail trap account – for sending key over mail

3 Libraries Required

Following libraries were required to run the code. Some libraries are inbuilt with python and some need to be installed.

Boto - Programmers can use Amazon Web Services (AWS Boto3's) Python Software Development Kit (SDK) to connect their Python-based applications to services like Amazon S3.

To install boto3 module following command will be used:

pip install boto3

Cryptography - is the research of widely used cryptographic techniques, such as message digests, key derivation functions, and symmetric cyphers. It includes both low-level interfaces for various approaches and high-level recipes. To install cryptography module following command will be used:

pip install cryptography

4 EncryptionScript

```
import os
import base64
import smtplib
from zipfile import ZipFile
# import the corresponding modules
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import cryptography
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding, rsa, utils
from cryptography.hazmat.backends import default_backend
import boto3
import time

# get the start time
st = time.process_time()
port = 2525
smtp_server = "smtp.mailtrap.io"
login = "a2329c8a81a8c1" # paste your login generated by Mailtrap
password = "4f33af52abebb7" # paste your password generated by Mailtrap

subject = "Encrypted Key Received"
sender_email = "suyashtripathi.in@gmail.com"
receiver_email = "suyashtripathi.in@gmail.com"

message = MIMEMultipart()
message["From"] = sender_email
message["To"] = receiver_email
message["Subject"] = subject

# Add body to email
body = "New Encryption key received"
message.attach(MIMEText(body, "plain"))
```

```
# Add body to email
body = "New Encryption key received"
message.attach(MIMEText(body, "plain"))
# Generate a private/public key pair
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048
)

public_key = private_key.public_key()

# Save the RSA private key to a PEM file
pem = private_key.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.PKCS8,
    encryption_algorithm=serialization.NoEncryption()
)
with open("private_key.pem", "wb") as file:
    file.write(pem)

# Get the file name from the user
file_name = input('Enter the file name: ')

# Read the contents of the file
with open(file_name, 'rb') as f:
    file_data = f.read()

# Encrypt the file using AES
key = Fernet.generate_key()
fernet = Fernet(key)
encrypted_data = fernet.encrypt(file_data)

# Encrypt the AES key using RSA
encrypted_key = public_key.encrypt(
    key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
```

```
filename = "Keysfile.zip"
# We assume that the file is in the directory where you run your Python script from
with open(filename, "rb") as attachment:
    # The content type "application/octet-stream" means that a MIME attachment is a binary file
    part = MIMEBase("application", "octet-stream")
    part.set_payload(attachment.read())

# Encode to base64
encoders.encode_base64(part)

# Add header
part.add_header(
    "Content-Disposition",
    f"attachment; filename= {filename}",
)

# Add attachment to your message and convert it to string
message.attach(part)
text = message.as_string()

# send your email
with smtplib.SMTP("smtp.mailtrap.io", 2525) as server:
    server.login(login, password)
    server.sendmail(
        sender_email, receiver_email, text
    )
print('Encrypted key sent on mail')

# Set up the S3 client
s3 = boto3.client('s3')

# Upload the encrypted file to S3
s3.upload_file(
    'encrypted_' + file_name,
    'suyashbucket',
    'encrypted_' + file_name
)

print('File successfully uploaded to S3')
```

```
# Encrypt the AES key using RSA
encrypted_key = public_key.encrypt(
    key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# Save the encrypted key to a file
with open('encrypted_key.txt', 'wb') as f:
    f.write(encrypted_key)




# Save the encrypted file to a new file
with open('encrypted_' + file_name, 'wb') as f:
    f.write(encrypted_data)
# get the end time

time.sleep(3)
print('File successfully encrypted and key saved to encrypted_key.txt')

et = time.process_time()
# get execution time
res = et - st
print('CPU Execution time:', res, 'seconds')

with ZipFile('Keysfile.zip', 'w') as zip_object:
   # Adding files that need to be zipped
   zip_object.write('encrypted_key.txt')
   zip_object.write('private_key.pem')

filename = "Keysfile.zip"
```

5 Decryption Script

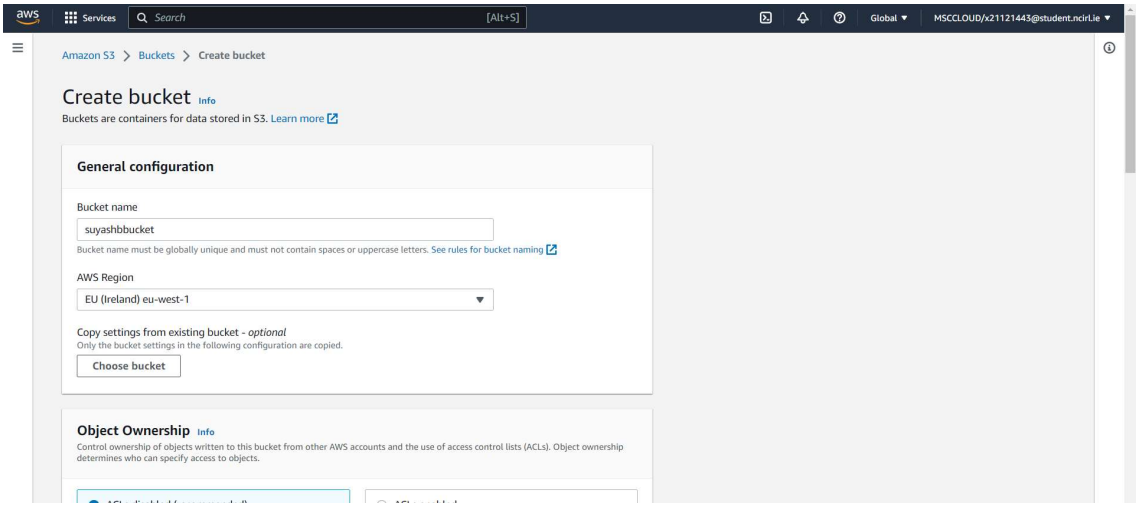
```
import os
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import padding, rsa, utils
from cryptography.hazmat.backends import default_backend
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives.asymmetric import rsa




# Read the RSA private key from the PEM file
with open("private_key.pem", "rb") as file:
    pem = file.read()
    private_key = serialization.load_pem_private_key(
        pem,
        password=None,
        backend=default_backend()
    )

# Read the encrypted key from the file
with open('encrypted_key.txt', 'rb') as f:
    encrypted_key = f.read()

# Decrypt the key using the RSA private key
key = private_key.decrypt(
    encrypted_key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
```

```
# Decrypt the key using the RSA private key
key = private_key.decrypt(
    encrypted_key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# Decrypt the file using the decrypted AES key
fernet = Fernet(key)
encrypted_data =open("encrypted_plain.txt", "rb").read()
decrypted_data = fernet.decrypt(encrypted_data)

# Save the decrypted file to a new file
with open('decrypted_data.txt', 'wb') as f:
    f.write(decrypted_data)

print('File successfully decrypted')
```

6 Installation and Code Run Procedures

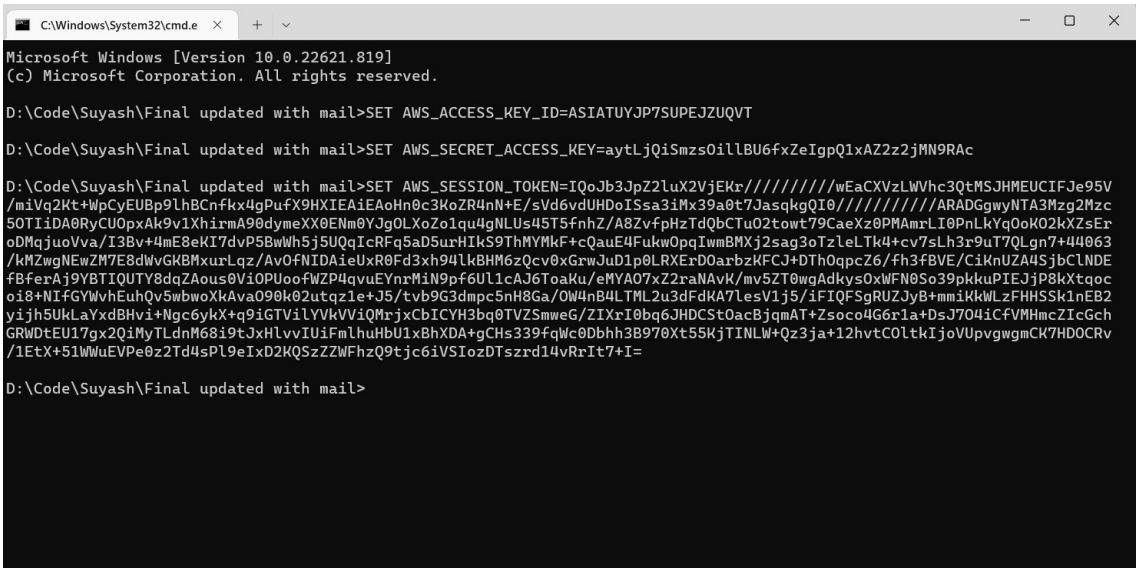
Following steps need to be followed for running the code.

Step 1 – Login to AWS account and create S3 bucket.



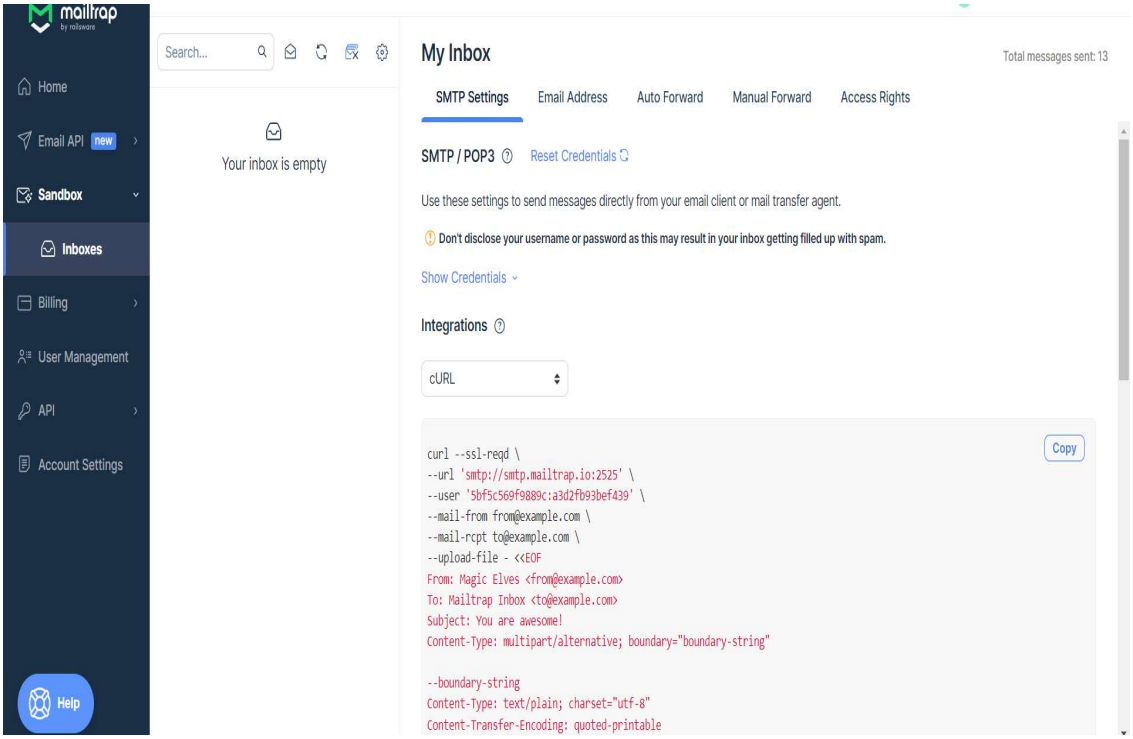
Step 2 – Open the AWS programmatic access

Copy the environment variables and paste it on the command prompt it will allow to connect with AWS.

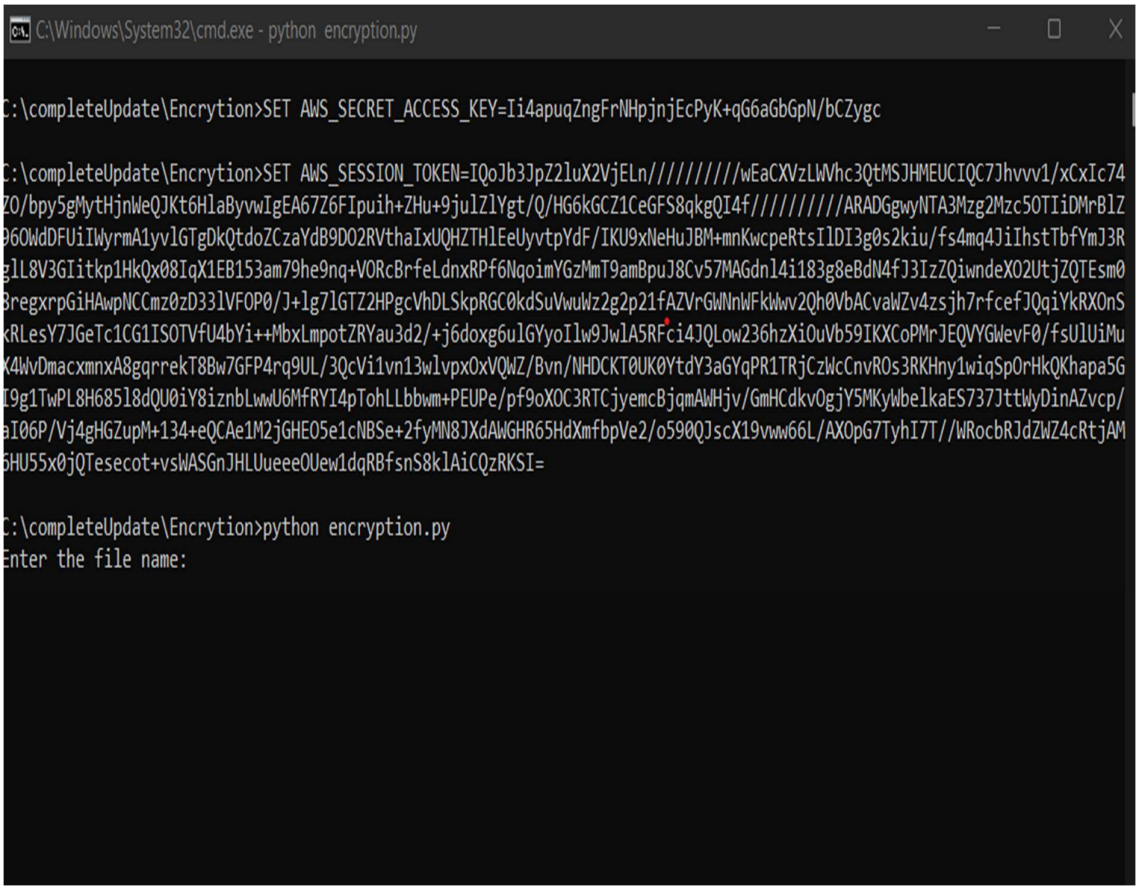


Step 3 – Setup the Mailtrap account

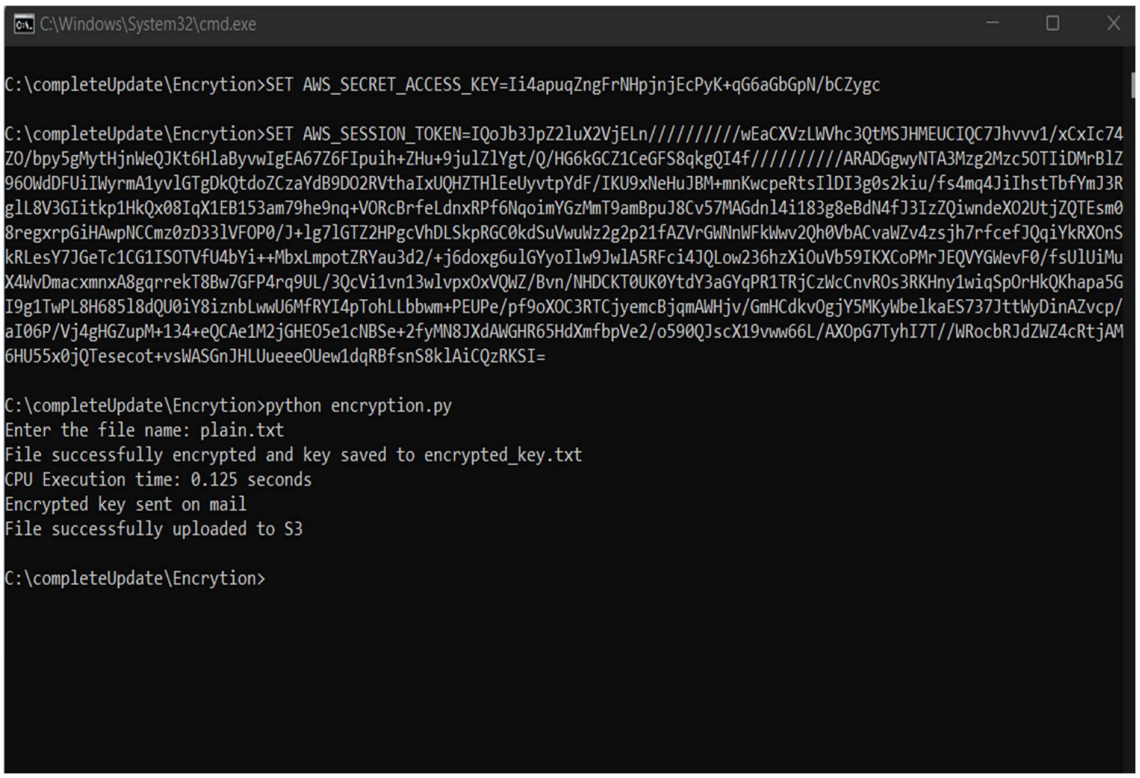
Signup for the mail trap account and paste the login and username in the script provided by the Mailtrap.



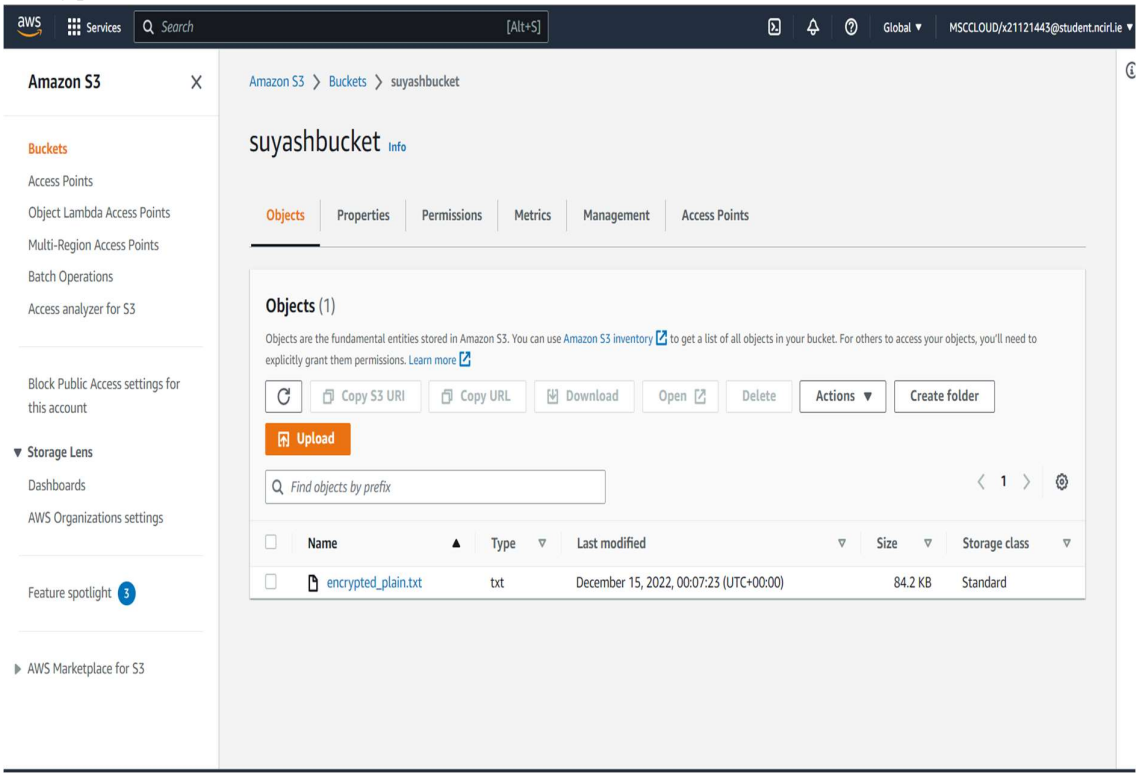
Step 4 – Run the script using command prompt



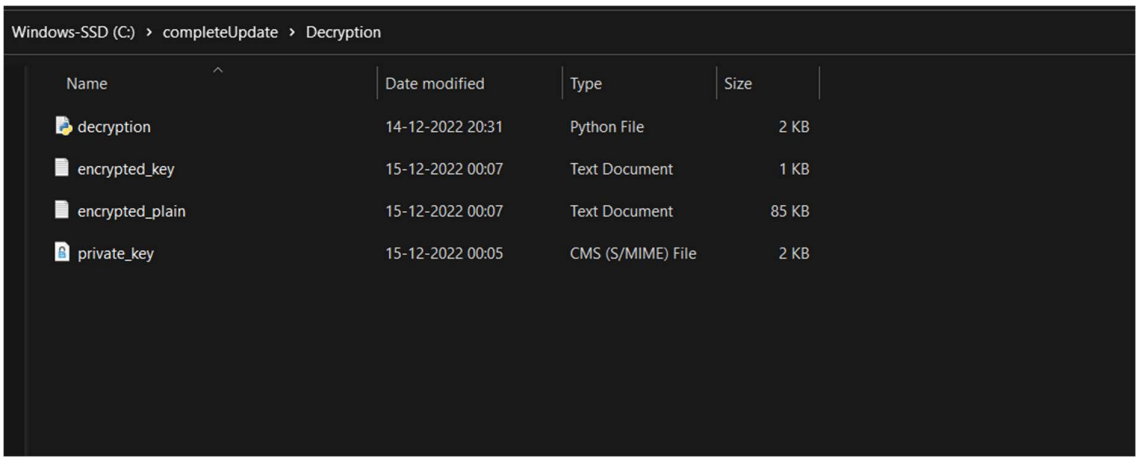
Step 5 – Enter the File name with its type, The file will get uploaded to the cloud and following message will be printed with the execution time.



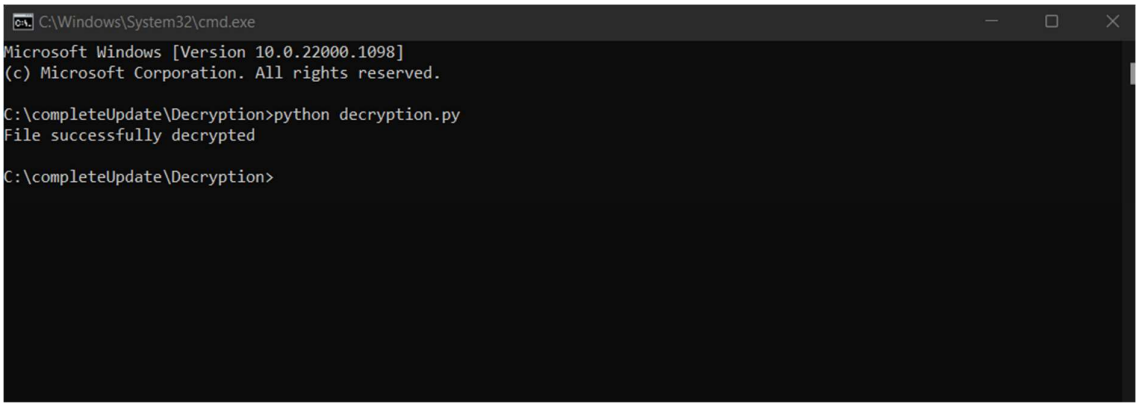
Step 6 – Check the mail trap account for the encryption key and AWS bucket for the encrypted file.



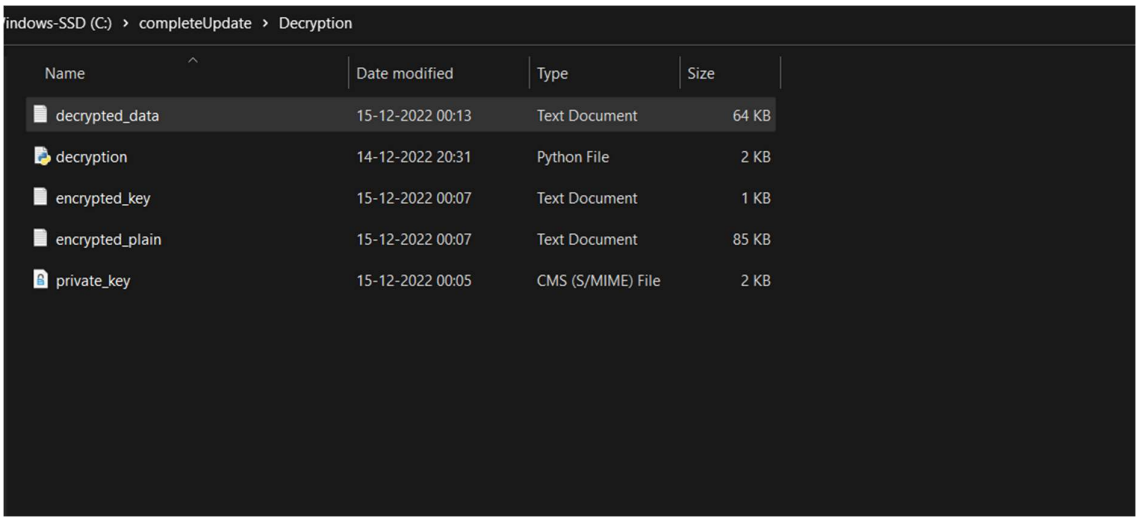
Step 7: Paste the encrypted key received on email and encrypted file from the cloud in the decryption folder for executing decryption.



Step 8: Open this folder in command prompt and run the decryption code for decrypting the file.



Step 9: A decrypted file will be created in the decryption folder.



References

Alla, S., & Adari, S. K. (2021). Deploying in AWS. In *Beginning MLOps with MLFlow* (pp. 229-252). Apress, Berkeley, CA.

Abood, O. G., & Guirguis, S. K. (2018). A survey on cryptography algorithms. *International Journal of Scientific and Research Publications*, *8*(7), 495-516.

mailtrap.io - Sending mail. (2022). MODX Documentation; MODX Docs. https://docs.modx.com/3.x/en/building-sites/sending-mail/mailtrap