```
!pip install pycryptodome
```

```
Requirement already satisfied: pycryptodome in
/usr/local/lib/python3.10/dist-packages (3.19.0)
```

Installing library

DES for .txt, .exe and .rar files

```
from Crypto.Cipher import DES
from Crypto.Random import get_random_bytes
from google.colab import drive
drive.mount('/content/drive')
from Crypto.Cipher import DES
import os
import time
from Crypto.Cipher import DES
from Crypto.Util.Padding import unpad

def pad(text):
    while len(text) % 8 != 0:
        text += b' '
    return text

def unpad(text):
    return text.rstrip()

def encrypt_file(input_file, output_file, key):
    cipher = DES.new(key, DES.MODE_ECB)

    with open(input_file, 'rb') as f_in:
        plaintext = f_in.read()
        padded_plaintext = pad(plaintext)
        ciphertext = cipher.encrypt(padded_plaintext)

    with open(output_file, 'wb') as f_out:
        f_out.write(ciphertext)


def decrypt_file(input_file, output_file, key):
    cipher = DES.new(key, DES.MODE_ECB)

    with open(input_file, 'rb') as f_in:
        ciphertext = f_in.read()
        decrypted_data = cipher.decrypt(ciphertext)

    with open(output_file, 'wb') as f_out:
        f_out.write(decrypted_data)
```

For encrypting using DES

For decrypting using DES

```
def generate_key():
    return os.urandom(8)
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

DES for .txt files (1MB, 10MB and 50MB)

encrypting & decrypting text files

Time take by 1MB text file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/1MB.txt'
output_file = f'/content/drive/MyDrive/1MBEncryptedTextDES.txt'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedTextD
ES.txt'
key = generate_key()

encrypt_file(input_file, output_file, key)
decrypt_file(output_file,decrypt_file_path,key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedTextDES.txt.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/1MBDecryptedTextDES.txt.
Time spent: 0.4380947379995632 seconds
```

Time taken by 10MB text file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/10MB.txt'
output_file = f'/content/drive/MyDrive/10MBEncryptedTextDES.txt'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/10MBDecryptedText
DES.txt'


key = generate_key()

encrypt_file(input_file, output_file, key)
decrypt_file(output_file,decrypt_file_path,key)
```

```
print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedTextDES.txt.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/10MBDecryptedTextDES.txt.
Time spent: 5.13945903000058 seconds
```

Time taken by 50MB text file

```
start_time = time.perf_counter()

input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/50MB.txt'
output_file = f'/content/drive/MyDrive/50MBEncryptedTextDES.txt'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/50MBDecryptedText
DES.txt'

key = generate_key()


encrypt_file(input_file, output_file, key)
decrypt_file(output_file,decrypt_file_path,key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedTextDES.txt.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/50MBDecryptedTextDES.txt.
Time spent: 33.26952064800025 seconds
```

DES for .exe files (1MB, 10MB and 50MB)

encrypting & decrypting exe files

DES for 1MB Exe file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/50MB.exe'
output_file = f'/content/drive/MyDrive/50MBEncryptedExeDES.exe'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedExeDE
S.exe'

key = generate_key()


encrypt_file(input_file, output_file, key)
decrypt_file(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedExeDES.exe.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/1MBDecryptedExeDES.exe.
Time spent: 3.899444702999972 seconds
```

DES for 10MB Exe file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/10MB.exe'
output_file = f'/content/drive/MyDrive/10MBEncryptedExeDES.exe'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/10MBDecryptedExeD
ES.exe'

key = generate_key()

encrypt_file(input_file, output_file, key)
decrypt_file(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()
```

```
time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedExeDES.exe.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/10MBDecryptedExeDES.exe.
Time spent: 0.5558436009996512 seconds
```

DES for 50MB Exe File

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/50MB.exe'
output_file = f'/content/drive/MyDrive/50MBEncryptedExeDES.exe'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/50MBDecryptedExeD
ES.exe'

key = generate_key()

encrypt_file(input_file, output_file, key)
decrypt_file(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedExeDES.exe.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/50MBDecryptedExeDES.exe.
Time spent: 2.0471452649999264 seconds
```

DES for .zip files (1MB, 10MB and 50MB)

encrypting & decrypting zip files

DES for 1MB Zip File

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/1MB.zip'
output_file = f'/content/drive/MyDrive/1MBEncryptedZipDES.zip'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedZipDE
S.zip'

key = generate_key()
```

```
encrypt_file(input_file, output_file, key)
decrypt_file(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedZipDES.zip.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/1MBDecryptedZipDES.zip.
Time spent: 0.07019324999964738 seconds
```

DES for 10MB Zip File

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/10MB.zip'
output_file = f'/content/drive/MyDrive/10MBEncryptedZipDES.zip'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedZipDE
S.zip'

key = generate_key()

encrypt_file(input_file, output_file, key)
decrypt_file(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedZipDES.zip.
Decryption complete. Output saved to
/content/drive/Decrypted/1MBDecryptedZipDES.zip.
Time spent: 0.47955990199989174 seconds
```

DES for 50MB Zip File

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/50MB.zip'
```

```
output_file = f'/content/drive/MyDrive/50MBEncryptedZipDES.zip'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/50MBDecryptedZipD
ES.zip'


key = generate_key()

encrypt_file(input_file, output_file, key)
decrypt_file( output_file,decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedZipDES.zip.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/50MBDecryptedZipDES.zip.
Time spent: 2.885416791999887 seconds
```

AES for .txt, .exe and .rar files

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
import os
import time

def pad_AES(text):
    while len(text) % 16 != 0:
        text += b' '
    return text

def encrypt_file_AES(input_file, output_file, key):
    cipher = AES.new(key, AES.MODE_ECB)

    with open(input_file, 'rb') as f_in:
        plaintext = f_in.read()
        padded_plaintext = pad(plaintext)
        ciphertext = cipher.encrypt(padded_plaintext)

    with open(output_file, 'wb') as f_out:
        f_out.write(ciphertext)


def decrypt_file_AES(input_file, output_file, key):
```

} encrypting using AES

} decrypting using AES

```
    cipher = AES.new(key, AES.MODE_ECB)

    with open(input_file, 'rb') as f_in:
        ciphertext = f_in.read()
        decrypted_data = cipher.decrypt(ciphertext)

    decrypted_data = decrypted_data.rstrip(b' ')

    with open(output_file, 'wb') as f_out:
        f_out.write(decrypted_data)

def generate_key_AES():
    return get_random_bytes(16)  # For AES-128, use 16 bytes
```

AES for .txt files (1MB, 10MB and 50MB)

AES for text files encrypting & decrypting

Time taken by 1MB text file

```
start_time = time.perf_counter()

input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/1MB.txt'
output_file = f'/content/drive/MyDrive/1MBEncryptedTxtAES.txt'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedTxtAE
S.txt'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedTxtAES.txt.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/1MBDecryptedTxtAES.txt.
Time spent: 0.16342099899975437 seconds
```

Time taken by 10MB text file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/10MB.txt'
output_file = f'/content/drive/MyDrive/10MBEncryptedTxtAES.txt'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/10MBDecryptedTxtA
ES.txt'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedTxtAES.txt.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/10MBDecryptedTxtAES.txt.
Time spent: 2.093360515000313 seconds
```

Time taken by 50MB text file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/50MB.txt'
output_file = f'/content/drive/MyDrive/50MBEncryptedTxtAES.txt'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/50MBDecryptedTxtA
ES.txt'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedTxtAES.txt.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/50MBDecryptedTxtAES.txt.
Time spent: 15.429331012999683 seconds
```

AES encryption & decryption using zip files

AES for .zip files (1MB, 10MB and 50MB)

Time taken by 1MB zip file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/1MB.zip'
output_file = f'/content/drive/MyDrive/1MBEncryptedZipAES.zip'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedZipAE
S.zip'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)


print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedZipAES.zip.
Decryption complete. Output saved to
/content/drive/Decrypted/1MBDecryptedZipAES.zip.
Time spent: 0.04445512399979634 seconds
```

Time taken by 10MB zip file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/10MB.zip'
output_file = f'/content/drive/MyDrive/10MBEncryptedZipAES.zip'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/10MBDecryptedZipA
ES.zip'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)
```

```
print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedZipAES.zip.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/10MBDecryptedZipAES.zip.
Time spent: 0.1486570090000896 seconds
```

Time taken by 50MB zip file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/50MB.zip'
output_file = f'/content/drive/MyDrive/50MBEncryptedZipAES.zip'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/50MBDecryptedZipA
ES.zip'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)


print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedZipAES.zip.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/50MBDecryptedZipAES.zip.
Time spent: 1.750436979999904 seconds
```

AES for .exe files (1MB, 10MB and 50MB)

AES encryption & decryption using exe files

Time taken by 1MB exe file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/1MB.exe'
output_file = f'/content/drive/MyDrive/1MBEncryptedExeAES.exe'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/1MBDecryptedExeAE
S.exe'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)


print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedExeAES.exe.
Decryption complete. Output saved to
/content/drive/Decrypted/1MBDecryptedExeAES.exe.
Time spent: 0.09105535199978476 seconds
```

Time taken by 10MB exe file

```
start_time = time.perf_counter()

input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/10MB.exe'
output_file = f'/content/drive/MyDrive/10MBEncryptedExeAES.exe'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/10MBDecryptedExeA
ES.exe'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)


print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()
```

```
time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedExeAES.exe.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/10MBDecryptedExeAES.exe.
Time spent: 0.395200935999128 seconds
```

Time taken by 50MB exe file

```
start_time = time.perf_counter()
input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/50MB.exe'
output_file = f'/content/drive/MyDrive/50MBEncryptedExeAES.exe'
decrypt_file_path=f'/content/drive/MyDrive/Decrypted/50MBDecryptedExeA
ES.exe'

key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
decrypt_file_AES(output_file, decrypt_file_path, key)


print(f'Encryption complete. Output saved to {output_file}.')
print(f'Decryption complete. Output saved to {decrypt_file_path}.')

end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedExeAES.exe.
Decryption complete. Output saved to
/content/drive/MyDrive/Decrypted/50MBDecryptedExeAES.exe.
Time spent: 1.5026926880000246 seconds
```

RSA for .txt, .exe and .rar files

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

def generate_key_pair():
    key = RSA.generate(2048)
    return key

def export_public_key(key, file_path):
```

} generating key pair

```
    public_key = key.publickey().export_key()
    with open(file_path, "wb") as file:
        file.write(public_key)

def import_public_key(file_path):
    with open(file_path, "rb") as file:
        public_key = RSA.import_key(file.read())
    return public_key

def encrypt_file_rsa(public_key, data, output_file):
    print("type of data get:", type(data))
    message = data

    cipher_rsa = PKCS1_OAEP.new(public_key)
    encrypted_message = cipher_rsa.encrypt(message)

    with open(output_file, "wb") as file:
        file.write(encrypted_message)

key_pair = generate_key_pair()

export_public_key(key_pair, "/content/drive/MyDrive/publicKey.pem")

public_key = import_public_key("/content/drive/MyDrive/publicKey.pem")
```

saving public key

encrypting file using RSA

decrypting file using RSA

#RSA for .txt files (1MB, 10MB and 50MB)

Time taken by 1MB txt file

RSA encryption & decryption using text files

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    # print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/1MB.txt'
output_file = f'/content/drive/MyDrive/1MBEncryptedTxtRSA.txt'
output_file_key = f'/content/drive/MyDrive/1MBEncryptedTxtRSAkey.txt'
key = generate_key_AES()

stringKey = key.decode('latin-1')

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()
```

```
time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedTxtRSA.txt.
Encryption complete. Output key saved to
/content/drive/MyDrive/1MBEncryptedTxtRSAkey.txt.
Time spent: 0.29497916900072596 seconds
```

Time taken by 10MB txt file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/10MB.txt'
output_file = f'/content/drive/MyDrive/10MBEncryptedTxtRSA.txt'
output_file_key = f'/content/drive/MyDrive/10MBEncryptedTxtRSAkey.txt'
key = generate_key_AES()

stringKey = key.decode('latin-1')

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Public RSA key at 0x78B87D955000
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedTxtRSA.txt.
Encryption complete. Output key saved to
/content/drive/MyDrive/10MBEncryptedTxtRSAkey.txt.
Time spent: 2.5753608810000514 seconds
```

Time taken by 50MB txt file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
```

```
    print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/TextFiles/50MB.txt'
output_file = f'/content/drive/MyDrive/50MBEncryptedTxtRSA.txt'
output_file_key = f'/content/drive/MyDrive/50MBEncryptedTxtRSAkey.txt'
key = generate_key_AES()

stringKey = key.decode('latin-1')

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Public RSA key at 0x78B86738F850
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedTxtRSA.txt.
Encryption complete. Output key saved to
/content/drive/MyDrive/50MBEncryptedTxtRSAkey.txt.
Time spent: 6.333579180000015 seconds
```

#RSA for .zip files (1MB, 10MB and 50MB)

Time taken by 1MB zip file

RSA encryption & decryption using zip file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/1MB.zip'
output_file = f'/content/drive/MyDrive/1MBEncryptedZipRSA.txt'
output_file_key = f'/content/drive/MyDrive/1MBEncryptedZipRSAkey.txt'
key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
```

```
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
Public RSA key at 0x78B86738EA10
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedZipRSA.txt.
Encryption complete. Output key saved to
/content/drive/MyDrive/1MBEncryptedZipRSAkey.txt.
Time spent: 0.052257104000091203 seconds
```

Time taken by 10MB zip file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    # print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/10MB.zip'
output_file = f'/content/drive/MyDrive/10MBEncryptedZipRSA.txt'
output_file_key = f'/content/drive/MyDrive/10MBEncryptedZipRSAkey.txt'
key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedZipRSA.txt.
Encryption complete. Output key saved to
/content/drive/MyDrive/10MBEncryptedZipRSAkey.txt.
Time spent: 0.14051656799983903 seconds
```

Time taken by 50MB zip file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    # print(public_key)
```

```
input_file =
'/content/drive/MyDrive/FilesForEncryption/ZipFiles/50MB.zip'
output_file = f'/content/drive/MyDrive/50MBEncryptedZipRSA.txt'
output_file_key = f'/content/drive/MyDrive/50MBEncryptedZipRSAkey.txt'
key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedZipRSA.txt.
Encryption complete. Output key saved to
/content/drive/MyDrive/50MBEncryptedZipRSAkey.txt.
Time spent: 0.7896850639999684 seconds
```

#RSA for .exe files (1MB, 10MB and 50MB)

RSA encryption & decryption using Exe files

Time taken by 1MB exe file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    # print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/1MB.exe'
output_file = f'/content/drive/MyDrive/1MBEncryptedExeRSA.exe'
output_file_key = f'/content/drive/MyDrive/1MBEncryptedExeRSAkey.txt'
key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/1MBEncryptedExeRSA.exe.
Encryption complete. Output key saved to
/content/drive/MyDrive/1MBEncryptedExeRSAkey.txt.
Time spent: 0.06561561999933474 seconds
```

Time taken by 10MB exe file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    # print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/10MB.exe'
output_file = f'/content/drive/MyDrive/10MBEncryptedExeRSA.exe'
output_file_key = f'/content/drive/MyDrive/10MBEncryptedExeRSAkey.txt'
key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/10MBEncryptedExeRSA.exe.
Encryption complete. Output key saved to
/content/drive/MyDrive/10MBEncryptedExeRSAkey.txt.
Time spent: 0.18224922099943797 seconds
```

Time taken by 50MB exe file

```
start_time = time.perf_counter()
with open('/content/drive/MyDrive/publicKey.pem', 'r') as file:
    public_key = RSA.import_key(file.read())
    # print(public_key)


input_file =
'/content/drive/MyDrive/FilesForEncryption/ExeFiles/50MB.exe'
output_file = f'/content/drive/MyDrive/50MBEncryptedExeRSA.exe'
output_file_key = f'/content/drive/MyDrive/50MBEncryptedExeRSAkey.txt'
```

```
key = generate_key_AES()

encrypt_file_AES(input_file, output_file, key)
encrypt_file_rsa(public_key, key, output_file_key)

print(f'Encryption complete. Output saved to {output_file}.')
print(f'Encryption complete. Output key saved to {output_file_key}.')
end_time = time.perf_counter()

time_spent = end_time - start_time
print(f"Time spent: {time_spent} seconds")
```

```
type of data get: <class 'bytes'>
Encryption complete. Output saved to
/content/drive/MyDrive/50MBEncryptedExeRSA.exe.
Encryption complete. Output key saved to
/content/drive/MyDrive/50MBEncryptedExeRSAkey.txt.
Time spent: 0.6359828419999758 seconds
```