

**CSE3501 - Information Security Analysis and Audit**

**PROJECT FINAL REVIEW**

**Slot: D1+TD1**

**Faculty: Prof. CHANDRASEGAR.T**

**Team Members:**

**17MIS0025 – Ruchita Yenduru**

**17MIS0444 – D.Sumanth Reddy**

**Review-1**

Eqn = (X^3)\*y-r\*y(x^2)+x\*y=1

α = ((((e ^ 3) \* y) + (3 \* (x ^ 2) \* y) - ((e ^ 2) \* r \* y) - (2 \* e \* r \* x \* y) + (e \* y) + y) % sn)

key = (((e^2)\*y) + (3\*y\*(x^2))+(3\*x\*e\*y)-(r\*e\*y)-(2\*r\*x\*y)+y)

**CODE:**

def hcfnaive(a, b):

if (b == 0):

return a

else:

return hcfnaive(b, a % b)

r = int(input("Enter the r: "))

for x in range(5, 500):

for y in range(5, 500):

if ((x ^ 3)\*y - (((x ^ 2) \* y) \* r) + (y \* x) == 1):

print(x, y, r)

p = int(input("Enter the p: "))

q = int(input("Enter the q: "))

N = p \* q

sn = (p - 1) \* (q - 1)

h = []

for i in range(int(sn / 2), sn):

if (hcfnaive(i, sn) == 1):

h.append(i)

e = h[1]

alpha = ((((e ^ 3) \* y) + (3 \* (x ^ 2) \* y) - ((e ^ 2) \* r \* y) - (2 \* e \* r \* x \* y) + (e \* y) + y) % sn)

print(alpha)

k = (((e^2)\*y) + (3\*y\*(x^2))+(3\*x\*e\*y)-(r\*e\*y)-(2\*r\*x\*y)+y)

M = int(input("Enter the message: "))

ct1 = ((M \*\* alpha) % N)

z = k % sn

ct2 = ((M \*\* z) % N)

print("ct-1 is: ", ct1)

print("ct-2 is: ", ct2)

l = (-e) % sn

dt = dt=((ct1)\*(ct2\*\*e)\*(ct2\*\*l))%N

print("The Decrypted message is: ", dt)

if (M == dt):

print("The message and decrypted message are same")

else:

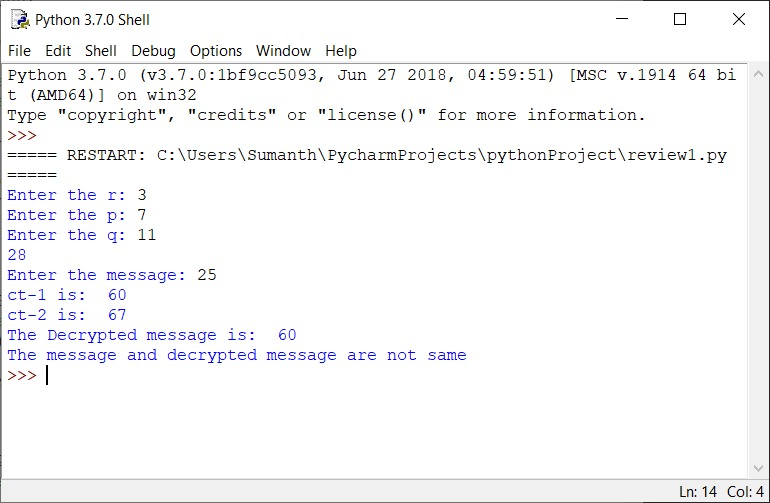
print("The message and decrypted message are not same")

**OUTPUT:**

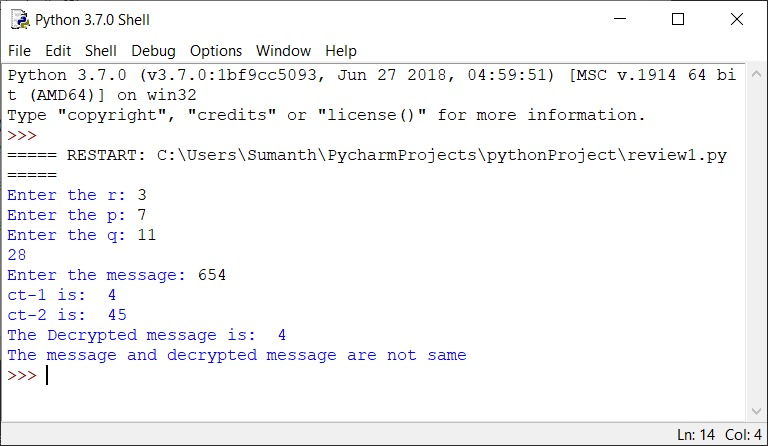
**Test Case-1:**



**Test Case 2:**



**Test Case 3:**



**Review2**

**CODE:**

import time

import sys

start=time.time()

def hctfi(a, b):

if (b == 0):

return a

else:

return hctfi(b, a % b)

r = int(input("Enter the r: "))#z=3

for x in range(1, 1000):

for y in range(1, 1000):

if ((x ^ 3)\*y - (((x ^ 2) \* y) \* r) + (y \* x) == 1):

print("Value of x is",x)

print("Value of y is",y)

print("Value of r is",r)

p = int(input("Enter the p: "))#p=5

q = int(input("Enter the q: "))#q=7

N = p \* q

sn = (p - 1) \* (q - 1)

d = []

for i in range(int(sn / 2), sn):

if (hctfi(i, sn) == 1):

d.append(i)

e = d[1]

alpha = ((((e ^ 3) \* y) + (3 \* (x ^ 2) \* y) - ((e ^ 2) \* r \* y) - (2 \* e \* r \* x \* y) + (e \* y) + y) % sn)

print("Alpha value is",alpha)

k = (((e^2)\*y) + (3\*y\*(x^2))+(3\*x\*e\*y)-(r\*e\*y)-(2\*r\*x\*y)+y)

print("k value is",k)

M = int(input("Enter the data value:"))

ct1 = ((M \*\* alpha) % N)

z = k % sn

ct2 = ((M \*\* z) % N)

print("Ciphertext-1 is:", ct1)

original\_stdout = sys.stdout

with open('ct1.txt', 'w') as f:

sys.stdout = f

print(ct1)

sys.stdout = original\_stdout

print("Ciphertext-2 is:", ct2)

original\_stdout = sys.stdout

with open('ct2.txt', 'w') as f1:

sys.stdout = f1

print(ct2)

sys.stdout = original\_stdout

l = (-e) % sn

dt = dt=((ct1)\*(ct2\*\*e)\*(ct2\*\*l))%N

print("The decrypted message is:", dt)

if (M == dt):

print(" Message and decrypted message are same and ciphered text values are stored")

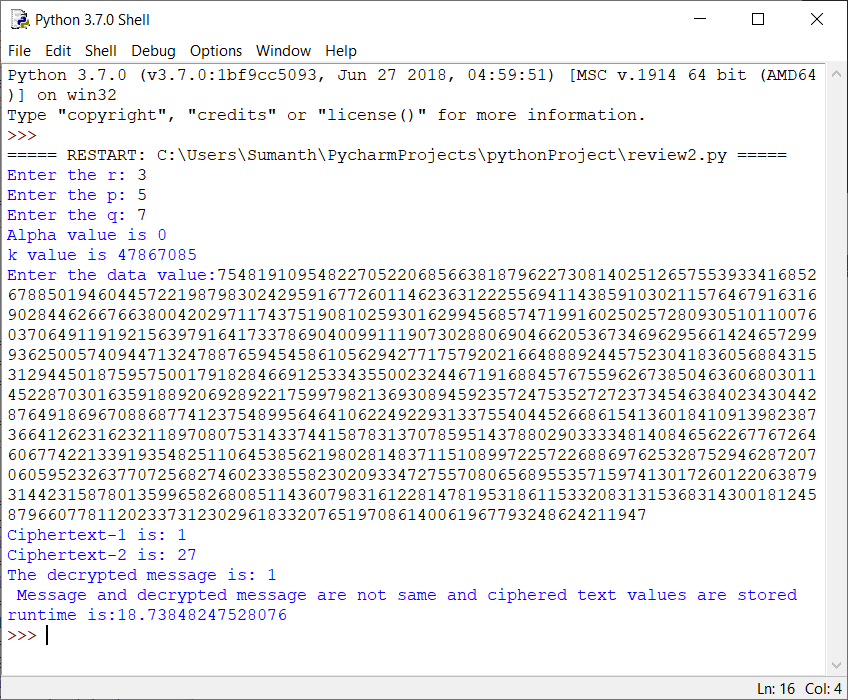
else:

print(" Message and decrypted message are not same and ciphered text values are stored")

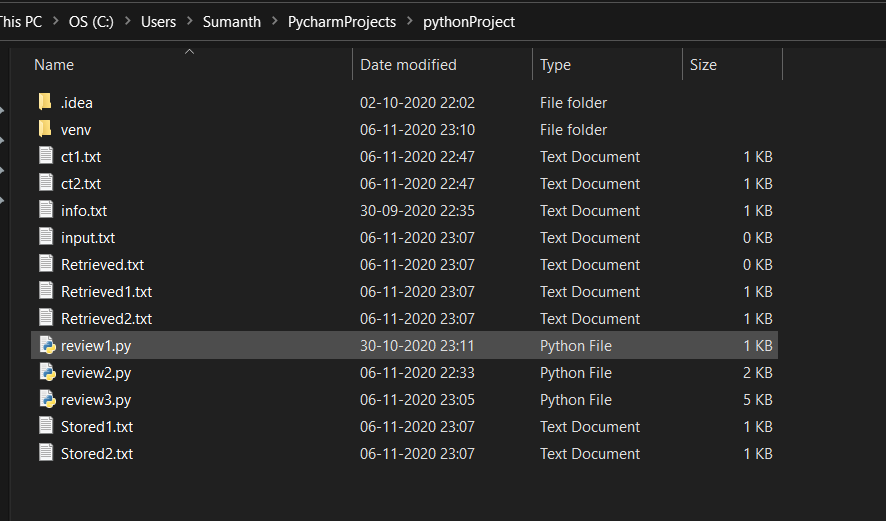
end=time.time()

print(f"runtime is:{end - start}")

**OUTPUT:**



And DATA is stored in ct1 and ct2 and those text files are generated for above 1KB data value.



**Review 3**

**CODE:**

import random

File\_object = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\input.txt","w+")

File\_object0 = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\Retrieved.txt","w")

File\_object1 = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\Stored1.txt","w")

File\_object2 = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\Retrieved1.txt","w")

File\_object3 = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\Stored2.txt","w")

File\_object4 = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\Retrieved2.txt","w")

#Intelligent Cipher Technique

S=input("Enter Info Data: ")

S= open("ct1.txt", 'r')

print(S.read())

print(S.readline())

#File\_object.write(S)

File\_object.close()

File\_object = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\info.txt","r")

D=File\_object.read()

File\_object1.writelines(["Info Data : ",D,"\r\n"])

y=int(input("Enter division number to divide the info :"))

#5650

c=bin(y)[2:].zfill(len(D))

r=bin(int(D,2)-int(c,2))[2:].zfill(len(D))

print("Information after division to two parts :",r," ",c)

key=bin(random.randrange(y,int(D,2)))[2:]

print("Random key Generated is : ",key,"\r\n")

File\_object1.writelines(["Random Key : ",key,"\r\n"])

cloudA=bin(int(c,2)^int(key,2))[2:].zfill(len(D))

cloudB=bin(int(r,2)^int(key,2))[2:].zfill(len(D))

#Data Storing in Cloud

print("Data stored in clouds are :")

print("Cloud A :",cloudA)

print("Cloud B :",cloudB)

File\_object1.writelines(["Data stored in clouds are :","\r\n","cloudA :",cloudA,"\t","cloudB :",cloudB,"\r\n"])

# Retrieving Data From Cloud Phase

cloudAretrieval=bin(int(cloudA,2)^int(key,2))[2:].zfill(len(D))

cloudBretrieval=bin(int(cloudB,2)^int(key,2))[2:].zfill(len(D))

print("Data retrieved from clouds are :")

print("Cloud A :",cloudAretrieval)

print("Cloud B :",cloudBretrieval)

File\_object2.writelines(["Retrieving Data From Cloud Phase :","\r\n","cloud A retrieval Data:",cloudAretrieval,"\t","cloud B retrieval Data:",cloudBretrieval,"\r\n"])

sum=bin(int(cloudAretrieval,2)+int(cloudBretrieval,2))[2:].zfill(len(D))

sum1=bin(int(sum,2)^int(key,2))[2:].zfill(len(D))

#Obtaining the original data

original=bin(int(sum1,2)^int(key,2))[2:].zfill(len(D))

print("Original data obtained is: : ",original)

File\_object1.writelines(["Original data obtained is :",original])

if(original==D):

print("Success--The input data is equal to finally Obtained original data\n")

else:

print("failed--The input data is not equal to finally Obtained original data")

S=input("Enter Info Data: ")

S= open("ct2.txt", 'r')

print(S.read())

print(S.readline())

#File\_object.write(S)

File\_object0.close()

File\_object0 = open(r"C:\Users\Sumanth\PycharmProjects\pythonProject\info.txt","r")

D=File\_object0.read()

File\_object3.writelines(["Info Data : ",D,"\r\n"])

y=int(input("Enter division number to divide the info :"))

#5650

c=bin(y)[2:].zfill(len(D))

r=bin(int(D,2)-int(c,2))[2:].zfill(len(D))

print("Information after division to two parts :",r," ",c)

key=bin(random.randrange(y,int(D,2)))[2:]

print("Random key Generated is : ",key,"\r\n")

File\_object3.writelines(["Random Key : ",key,"\r\n"])

cloudA=bin(int(c,2)^int(key,2))[2:].zfill(len(D))

cloudB=bin(int(r,2)^int(key,2))[2:].zfill(len(D))

#Data Storing in Cloud

print("Data stored in clouds are :")

print("Cloud A :",cloudA)

print("Cloud B :",cloudB)

File\_object3.writelines(["Data stored in clouds are :","\r\n","cloudA :",cloudA,"\t","cloudB :",cloudB,"\r\n"])

# Retrieving Data From Cloud Phase

cloudAretrieval=bin(int(cloudA,2)^int(key,2))[2:].zfill(len(D))

cloudBretrieval=bin(int(cloudB,2)^int(key,2))[2:].zfill(len(D))

print("Data retrieved from clouds are :")

print("Cloud A :",cloudAretrieval)

print("Cloud B :",cloudBretrieval)

File\_object4.writelines(["Retrieving Data From Cloud Phase :","\r\n","cloud A retrieval Data:",cloudAretrieval,"\t","cloud B retrieval Data:",cloudBretrieval,"\r\n"])

sum=bin(int(cloudAretrieval,2)+int(cloudBretrieval,2))[2:].zfill(len(D))

sum1=bin(int(sum,2)^int(key,2))[2:].zfill(len(D))

#Obtaining the original data

original=bin(int(sum1,2)^int(key,2))[2:].zfill(len(D))

print("Original data obtained is: : ",original)

File\_object1.writelines(["Original data obtained is :",original])

if(original==D):

print("Success--The input data is equal to finally Obtained original data")

else:

print("failed--The input data is not equal to finally Obtained original data")

File\_object.close()

File\_object0.close()

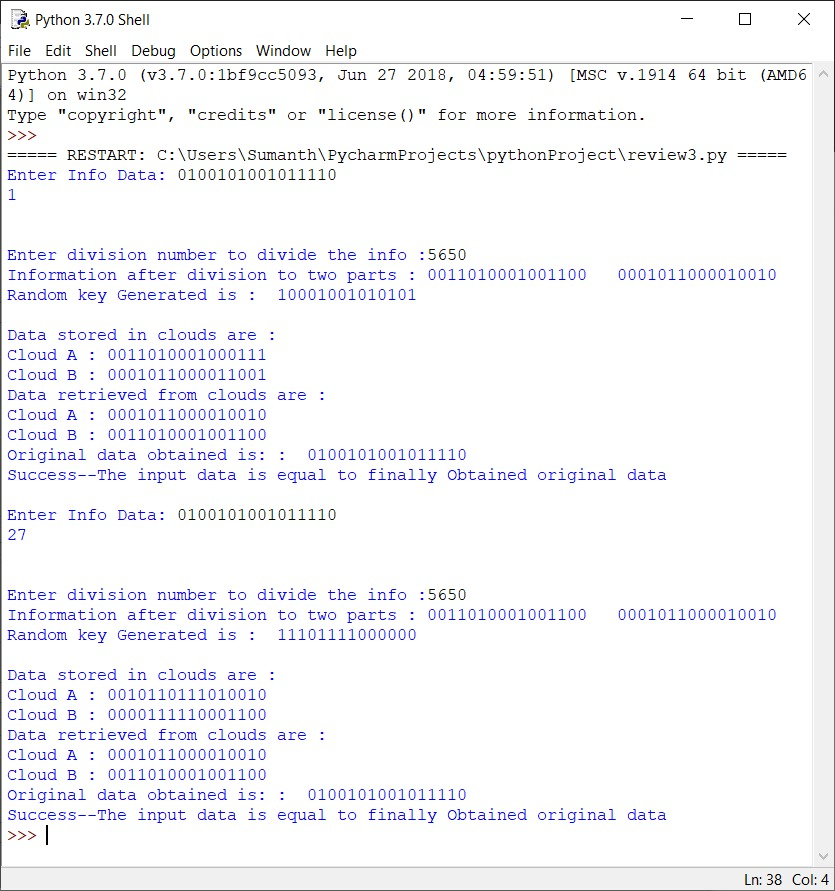
File\_object1.close()

File\_object2.close()

File\_object3.close()

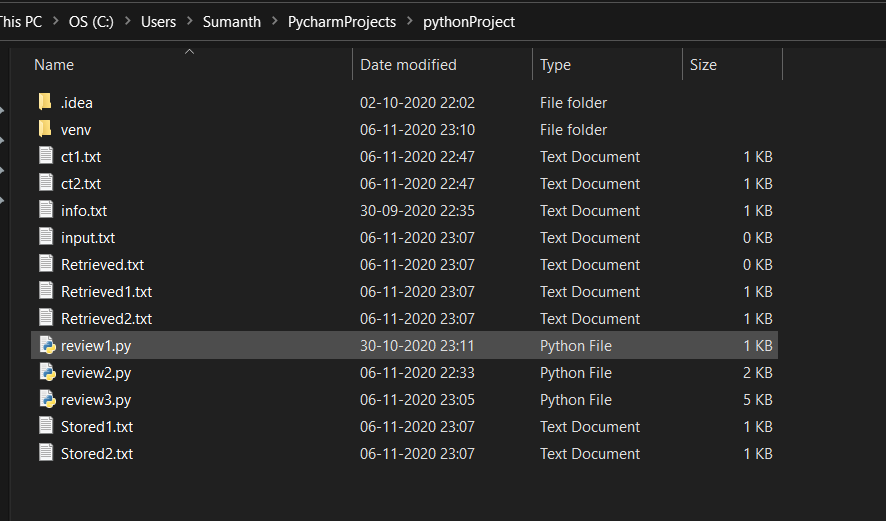
File\_object4.close()

**OUTPUT:**



Here Cloud Stored Values for ct1 and ct2 are stored as Stored1, Stored2

Here Cloud Retrieved Values for ct1 and ct2 are stored as Retrieved 1, Retrieved 2



And the values stored in them are:

