# Ceaser cipher:

**Client.py**

import socket

s = socket.socket() s.connect(("127.0.0.1",12345))

print("\*\*\*\* CEASER CIPHER \*\*\*\*")

print("\nEnter\n\t1) To Encryption\n\t2) To Decryption\n\t3) To implement Brute Force Attack\n\t4) To implement Frequency Analysis Attack")

n = int(input("Enter your choice('0' to EXIT) :")) while(n != 0):

if(n == 1):

pt = input("Enter the plain text: ")

key = int(input("Enter the key value: ")) s.send(bytes(str(n)+":"+pt+":"+str(key),encoding='utf8'))

elif(n == 2):

ct = input("Enter the cipher text: ")

key = int(input("Enter the key value: ")) s.send(bytes(str(n)+":"+ct+":"+str(key),encoding='utf8'))

elif(n == 3):

ct = input("Enter the cipher text: ") s.send(bytes(str(n)+":"+ct,encoding='utf8'))

elif(n == 4):

ct = input("Enter the cipher text: ") s.send(bytes(str(n)+":"+ct,encoding="utf8"))

n = int(input("Enter your choice('0' to EXIT) :"))

# Server.py

import socket

s = socket.socket() s.bind(("",12345))

s.listen(10)

c,a = s.accept()

y = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"]

w = ["e","t","a","o","i","n","s","h","r","d","l","c","u","m","w","f","g","y","p","b","v","k","j","x","q","z"]

def encryption(pt,key): s = ""

for i in range(len(pt)):

s += y[(y.index(pt[i])+key)%26] print(pt," => ",s)

def decryption(ct,key): s = ""

for i in range(len(ct)):

s += y[((y.index(ct[i])-key)+26)%26] print(ct," => ",s)

def bruteForceAttack(ct): for i in range(26):

s = ""

for j in range(len(ct)):

s += y[((y.index(ct[j])-i)+26)%26] print(i," => ",s)

def frequencAnalysisAttack(ct): l1,l2,d = [],[],{}

for i in ct:

if(i not in d.keys()): d[i] = 1 else: d[i] = d[i]+1

l2 = sorted(d.items(), key = lambda kv:(kv[1], kv[0])) for i in w:

for j in range(len(l2)-1,-1,-1):

k = (y.index(l2[j][0])-y.index(i))%26 s = ""

for x in range(len(ct)):

s += y[((y.index(ct[x])-k)+26)%26] print(ct," => ",s)

n = int(input("Enter '0' to continue '1' to stop: ")) if(n == 1):break

if(n == 1):break while(True):

x = (c.recv(1024).decode("utf8")).split(":") choice = int(x[0])

if(choice == 1):

pt,key = x[1],int(x[2]) encryption(pt,key)

elif(choice == 2): ct,key = x[1],int(x[2]) decryption(ct,key)

elif(choice == 3): ct = x[1]

bruteForceAttack(ct) elif(choice == 4):

ct = x[1] frequencAnalysisAttack(ct)

# Sha:

import math def binary(l):

s=[]

for k in range(2,10): s.append(bin(int(l[k],16))) d=[]

for i in s:

i=i[2: ]

if(len(i)<4):

le = 4-len(i)

for j in range(0,le): y="0"

x=y+i i=x

d.append(x) else: d.append(i) e=""

for i in range(len(d)): di= e+d[i]

e=di return di

def shift(s,st):

l1 = s[0:st] l2 = s[st: ] lcs= l2+l1 return lcs

def hexdec(s): m=[]

m1=""

for i in range(len(s)): m1=m1+s[i] if((len(m1)%4)!=0): continue

else: m.append(m1) m1=""

v=hexad(m) return v

def hexad(m): g=""

for i in range(len(m)):

h = hex(int(m[i],2)) h = h[2: ]

g=g+h g=hex(int(g,16)) return g

def carry(l):

if(len(l)==11):

l1 = l[0:2 ] l2 = l[3:11]

l=l1+l2 return l

a = "0x67452301"

b = "0xefcdab89" c = "0x98badcfe" d = "0x10325476"

e = "0xc3d2e1f0" #input msg from user

msg = input("\nenter the msg : ") #converting character to hexadecimal m=[]

for char in msg: m.append(hex(ord(char)))

#print(m) g=""

for i in range(len(m)): k = m[i]

k = k[2: ]

g=g+k g=hex(int(g,16))

print("STEP 1 OF ROUND 1 \n")

print("A = ",a)

print("B = ",b)

print("C = ",c)

print("D = ",d)

print("E = ",e)

for z in range(2,21):

#Boolean function = (b^c^d)

ft = hex((int(b,16)^int(c,16)^int(d,16))) #to discard carry

dc=carry(ft) print("\nft(b^c^d) = ",dc)

#addition#01(e+Boolean function) a1 = hex(int(e,16)+int(ft,16))

#to discard carry ad1=carry(a1) print("a1(ft+e) = ",ad1)

#converting a to binary and shift by 5 bits abin= binary(a)

as5 = shift(abin,5)

#after shifting converting binary to hex s5 = hexdec(as5)

#print("as5 = ",as5) print("s5 = ",s5)

a2 = hex(int(a1,16)+int(s5,16)) a2 = carry(a2) print("a2(a1+s5) = ",a2)

#wt = input("enter the above list in hex format : ") print("wt = ",g)

a3 = hex(int(a2,16)+int(g,16)) a3 = carry(a3) print("a3(a2+wt) = ",a3)

#round constant for round 1

kt = hex(int(pow(2,30)\*math.sqrt(2))) print("\nkt = ",kt)

a4 = hex(int(a3,16)+int(kt,16)) a4 = carry(a4) print("a4(a3+kt) = ",a4)

#converting b to binary and shift by 30 bits bbin= binary(b)

bs30 = shift(bbin,30) s30 = hexdec(bs30) #print("bs30 = ",bs30) print("s30 = ",s30)

#printing register values for next step print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") print("\nSTEP ",z," OF ROUND 1 \n")

print("A = ",a4)

print("B = ",a)

print("C = ",s30)

print("D = ",c)

print("E = ",d) a=a4

b=a c=s30 d=c e=d

# AES:

import math

from collections import Counter

def rotword(a):

a=a[2:]

l1=a[0:2]

l2=a[2:8]

#print("rot = ",l1+l2) return l2+l1

def subytes(rw):

sbt = "8a84eb01"

s\_box = s\_box = [['63', '7c', '77', '7b', 'f2', '6b', '6f', 'c5', '30', '01', '67', '2b', 'fe', 'd7', 'ab',

'76'],['ca', '82', 'c9', '7d', 'fa', '59', '47', 'f0', 'ad', 'd4', 'a2', 'af', '9c', 'a4', '72', 'c0'],['b7', 'fd',

'93', '26', '36', '3f', 'f7', 'cc', '34', 'a5', 'e5', 'f1', '71', 'd8', '31', '15'],['04', 'c7', '23', 'c3',

'18', '96', '05', '9a', '07', '12', '80', 'e2', 'eb', '27', 'b2', '75'],['09', '83', '2c', '1a', '1b', '6e',

'5a', 'a0', '52', '3b', 'd6', 'b3', '29', 'e3', '2f', '84'],['53', 'd1', '00', 'ed', '20', 'fc', 'b1', '5b',

'6a', 'cb', 'be', '39', '4a', '4c', '58', 'cf'],['d0', 'ef', 'aa', 'fb', '43', '4d', '33', '85', '45', 'f9', '02',

'7f', '50', '3c', '9f', 'a8'],['51', 'a3', '40', '8f', '92', '9d', '38', 'f5', 'bc', 'b6', 'da', '21', '10', 'ff',

'f3', 'd2'],['cd', '0c', '13', 'ec', '5f', '97', '44', '17', 'c4', 'a7', '7e', '3d', '64', '5d', '19',

'73'],['60', '81', '4f', 'dc', '22', '2a', '90', '88', '46', 'ee', 'b8', '14', 'de', '5e', '0b', 'db'],['e0',

'32', '3a', '0a', '49', '06', '24', '5c', 'c2', 'd3', 'ac', '62', '91', '95', 'e4', '79'],['e7', 'c8', '37',

'6d', '8d', 'd5', '4e', 'a9', '6c', '56', 'f4', 'ea', '65', '7a', 'ae', '08'],['ba', '78', '25', '2e', '1c',

'a6', 'b4', 'c6', 'e8', 'dd', '74', '1f', '4b', 'bd', '8b', '8a'],['70', '3e', 'b5', '66', '48', '03', 'f6',

'0e', '61', '35', '57', 'b9', '86', 'c1', '1d', '9e'],['e1', 'f8', '98', '11', '69', 'd9', '8e', '94', '9b',

'1e', '87', 'e9', 'ce', '55', '28', 'df'],['8c', 'a1', '89', '0d', 'bf', 'e6', '42', '68', '41', '99', '2d',

'0f', 'b0', '54', 'bb', '16']]

return sbt

def exor(a,b):

op = hex(int(a,16)^int(b,16))

return op

def kg(j):

rw = rotword(w[i-1]) sb = subytes(rw)

xr = exor(sb,r[j]) return xr

#n= int(input("enter the round no : ")) # 8a84eb01

#print("Round key 01")

w = ["0x2b7e1516","0x28aed2a6","0xabf71588","0x09cf4f3c"] r =

["0x01000000","0x02000000","0x04000000","0x08000000","0x10000000","0x20000000","0x400 00000","0x80000000","0x1b000000","0x36000000"]

#rw = rotword(w[3]) #sb = subytes(rw)

#xr = exor(w[0]l,sb,r[0])

for i in range(4,44):

j=0

if(i%4 == 0):

g=kg(j) #print("g = ",g)

x = hex(int(w[i-4],16)^int(g,16)) w.append(x)

j=j+1 else:

x = hex(int(w[i-4],16)^int(w[i-1],16)) w.append(x)

for i in range(len(w)): print("w",i," = ",w[i])

# RSA:

import math #key generation

p= int(input("enter p : "))

q= int(input("enter q : ")) n=p\*q

print("n = ",n)

phi\_n = ((p-1)\*(q-1)) print("phi\_n = ",phi\_n)

#calculate e ==> gcd(e,phi\_n)=1 for i in range(2,phi\_n): if((math.gcd(i,phi\_n)==1)):

e = i break else:

continue print("e = ",i)

#calculate d ===> d\*e mod(phi\_n) =1 for i in range(1,phi\_n):

if(((((i%phi\_n)\*(e%phi\_n))%phi\_n)==1)):

d = i break else:

continue print("d = ",d)

def encryption(M,e,n):

#C = ((M\*\*e)%n)

C= pow(M,e,n) return C

def decryption(C,d,n):

#M = ((C\*\*d)%n)%26

M= pow(C,d,n) plain=txt[M] return plain

txt ="abcdefghijklmnopqrstuvwxyz"

choice = int(input("enter 0 - exit/n 1-encryption/n2 - decryption/n")) while(choice!=0):

if(choice==1):

msg=input(("enter the plain text(M) : ")) M= txt.index(msg) ciphertxt=encryption(M,e,n) print("Cipher txt of ",msg," is ",ciphertxt)

elif(choice==2):

C=int(input("enter the cipher text(C) : ")) plaintxt=decryption(C,d,n)

print("Plain text of ",C," is ",plaintxt) else:

print("wrong choice")

choice = int(input("enter 0 - exit/n 1-encryption/n2 - decryption/n"))

# Password cracking:

a ="abcdefghijklmnopqrstwvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ@₹\_&\*1234567890" d={}

for i in a:

for j in a:

for k in a:

s=i+j+k d[hash(s)]=s

choice= int(input("enter 1-hash conversation ,2-passwd cracking :")) while(choice!=0):

if(choice==1):

pwd=str(input("enter password : ")) print("hash value is : ",hash(pwd)) elif(choice==2):

hv=int(input("enter hash value : ")) print("passwd is ",d[hv])

else:

print("wrong choice")

choice= int(input("enter 1-hash conversation ,2-passwd cracking :"))

# DHKX

**Server.py**

import socket import random s=socket.socket() port=12345 s.bind(("",port)) s.listen(5)

c, addr=s.accept()

def DiscreteLog(y,a,q): for i in range(1,q):

if(pow(a,i,q)==y): x=i

break return x

choice=int.from\_bytes(c.recv(1024), "big") while(choice!=0):

if(choice==1):

print("\nKey exchange") print(" ")

q=int.from\_bytes(c.recv(1024), "big") alpha=int.from\_bytes(c.recv(1024), "big") print("alpha=",alpha) X\_A=int.from\_bytes(c.recv(1024), "big") X\_B = random.randint(1,alpha) while(X\_A==X\_B):

X\_B = random.randint(1,alpha)

Y\_B=pow(alpha,X\_B,q) print("Y\_B=",Y\_B) Y\_A=int.from\_bytes(c.recv(1024), "big") c.send(Y\_B.to\_bytes(2,'big')) k=pow(Y\_A,X\_B,q)

print("k=",k) if(choice==2):

print("\n") print("DiscreteLog") print(" ")

x=DiscreteLog(Y\_A,alpha,q) print("X\_A=",x)

if(choice==3): print("\n")

print("Man in the middle attack") print(" ")

bob=int.from\_bytes(c.recv(1024), "big") alice=int.from\_bytes(c.recv(1024), "big") a1=int.from\_bytes(c.recv(1024), "big") q1=int.from\_bytes(c.recv(1024), "big") xB=DiscreteLog(bob,alpha,q)

print("X\_B after man in the middle attack : ",xB) #print("X\_A = ",DiscreteLog(alice,1,q1 k=pow(alice,xB,q)

print("K after man in the middle attack :",k) choice=int.from\_bytes(c.recv(1024), "big")

c.close()

# Client.py

import socket import random s=socket.socket() port=12345

s.connect(('localhost',port)) def isPrime(n):

flag=0

for i in range(2,int(n/2)): if(n%i==0):

flag=1 break

else :

continue if(flag==0):

return True else :

return False

def IsPrimitiveRoot(n,q): d=[]

flag=0

for i in range(1,q): p=pow(n,i,q)

if p not in d: d+=[p]

else :

flag=1 break if(flag==0):

return True else :

return False

def PrimitiveRootGenerate(n): l=[]

for i in range(2,n): if(IsPrimitiveRoot(i,n)==True):

l.append(i) return l

def DiscreteLog(y,a,q): for i in range(1,q):

if(pow(a,i,q)==y): x=i

break

return x

def ManInTheMiddle(): l=[]

for i in range(2,200): if(isPrime(i)):

l+=[i] q=random.choice(l) x=PrimitiveRootGenerate(q) a=random.choice(x) alice\_x=random.randint(2,q) bob\_x=random.randint(2,q) while(alice\_x==bob\_x):

bob\_x=random.randint(2,q) alice\_key=pow(a,alice\_x,q) bob\_key=pow(a,bob\_x,q) return alice\_key,bob\_key,a,q

choice = int(input("Enter 1-Key echange 2-Discrete log 3-Man in the middle attack 0-exit: ")) s.send(choice.to\_bytes(2,'big'))

while(choice!=0): if(choice==1):

q=int(input("Enter a prime number q : ")) s.send(q.to\_bytes(2,'big')) while(isPrime(q)==False):

print("q is not a prime!") q=int(input("Enter a prime number q : "))

primitive\_roots=PrimitiveRootGenerate(q) print("Primitive roots => ",primitive\_roots) alpha=int(input("Enter a primitive root : ")) s.send(alpha.to\_bytes(2,'big'))

X\_A = random.randint(1,alpha) s.send(X\_A.to\_bytes(2,'big')) Y\_A=pow(alpha,X\_A,q) print("Y\_A=",Y\_A) s.send(Y\_A.to\_bytes(2,'big')) Y\_B=int.from\_bytes(s.recv(1024), "big") k=pow(Y\_B,X\_A,q)

print("k=",k) if(choice==2):

print("\nDiscreteLog") print(" ")

x=DiscreteLog(Y\_B,alpha,q) print("X\_B=",x)

if(choice==3): alice,bob,a1,q1=ManInTheMiddle() print("public key=",alice) s.send(bob.to\_bytes(2,'big')) s.send(alice.to\_bytes(2,'big')) s.send(a1.to\_bytes(2,'big'))

s.send(q1.to\_bytes(2,'big')) xA=DiscreteLog(alice,alpha,q)

print("X\_A after man in the middle attack : ",xA) #print("X\_B = ",DiscreteLog(bob,a1,q1) k=pow(bob,xA,q)

print("K after man in the middle attack :",k)

choice = int(input("Enter 1-Key echange 2-Discrete log 3-Man in the middle attack 0-exit: ")) s.send(choice.to\_bytes(2,'big'))

s.close()

# Hill cipher Server.py

import socket import numpy as np from array import \* import math

import sympy

def decrypt(mat, text):

small = "abcdefghijklmnopqrstuvwxyz" cap="ABCDEFGHIJKLMNOPQRSTUVWXYZ"

mat=np.array(mat) det=np.linalg.det(mat) det=(round(det)%26)

adj = sympy.Matrix(mat).adjugate() adj = adj.tolist()

det\_inv=1/det

for i in range(len(adj)): for j in range(len(adj)):

if(adj[i][j]<0):

adj[i][j]=adj[i][j]%26 adj[i][j]=round(adj[i][j]\*det\_inv)

inverse = adj t1=""

t2=""

y=0

for i in text: if(y%2==0):

t1+=i else :

t2+=i y+=1

t=t1+t2 txt = []

ln=int(len(text)/len(mat[0])) count=1

m=[]

r=[]

for i in range(len(t)): if t[i] in small:

r+=[small.index(t[i])] if t[i] in cap:

r+=[cap.index(t[i])] if(count%ln==0):

m+=[list(r)] r=[]

count+=1

p = matrixMultiplication(inverse, m) for i in range(len(p)):

for j in range(len(p[0])): idx=p[i][j]%26 p[i][j]=idx

rez = [[p[j][i] for j in range(len(p))] for i in range(len(p[0]))] plain=""

for i in range(len(rez)):

for j in range(len(rez[0])): plain+=cap[rez[i][j]]

return plain

def encrypt(mat, text) :

small = "abcdefghijklmnopqrstuvwxyz" cap="ABCDEFGHIJKLMNOPQRSTUVWXYZ" t1=""

t2=""

y=0

for i in text: if(y%2==0):

t1+=i else :

t2+=i y+=1

t=t1+t2 txt = []

ln=int(len(text)/len(mat[0])) count=1

m=[]

r=[]

for i in range(len(t)): if t[i] in small:

r+=[small.index(t[i])] if t[i] in cap:

r+=[cap.index(t[i])] if(count%ln==0):

m+=[list(r)] r=[]

count+=1

p = matrixMultiplication(mat, m) for i in range(len(p)):

for j in range(len(p[0])): idx=p[i][j]%26 p[i][j]=idx

plain=""

rez = [[p[j][i] for j in range(len(p))] for i in range(len(p[0]))] for i in range(len(rez)):

for j in range(len(rez[0])): plain+=cap[rez[i][j]]

return plain

def matrixMultiplication(m1, m2): row = len(m2[0])

col = len(m1) result = []

for i in range(col): l=[]

for j in range(row): l+=[0]

result+=[l]

for i in range(col): for j in range(row):

for k in range(len(m2)): result[i][j]+=m1[i][k]\*m2[k][j]

return result

s=socket.socket() port=12345 s.bind(("",port)) s.listen(5)

c, addr=s.accept() choice=int.from\_bytes(c.recv(1024), "big") text = c.recv(1024)

text=text.decode() size=int.from\_bytes(c.recv(1024), "big") mat=[]

for i in range(size): row = c.recv(1024)

row = (row.decode()).split() mat+=[row]

for i in range(len(mat)): for j in range(len(mat)):

mat[i][j]=int(mat[i][j]) if(choice==1):

ans=encrypt(mat, text) c.send(bytes(ans, encoding='utf8'))

if(choice==2): ans=decrypt(mat, text)

c.send(bytes(ans, encoding='utf8')) c.close()

# Client.py

import socket s=socket.socket() port=12345 s.connect(('localhost',port))

choice = int(input("Enter 1-Encryption 2-Decryption : ")) s.send(choice.to\_bytes(2,'big'))

if(choice==1):

text=str(input("Enter the text for encryption : ")) if(choice==2):

text=str(input("Enter the text for decryption : ")) s.send(bytes(text, encoding='utf8'))

size = int(input("Enter the order of key matrix : ")) s.send(size.to\_bytes(2,'big'))

for i in range(size): row=str(input())

s.send(bytes(row, encoding='utf8')) ans=s.recv(1024)

print(ans.decode()) s.close()

# Openssl :

#aes encryption decryption

Openssl aes-128-cbc -e -in p.txt -out c1.bin -k “password” -nosalt Openssl aes-128-cbc -d -in c1.bin -out pt.txt -k “password” -nosalt

#avalanche effect

Openssl aes-128-cbc -e -in p.txt -out c2.bin -k “password” -nosalt

#rsa

Openssl genrsa -out pvtkey.pem

Openssl rsa -pubout -in pvtkey.pem -out pubkey.pem Openssl rsa -text -in pvtkey.pem

Openssl rsautl -encrypt -in plain.txt -pubin -inkey pubkey.pem -out c3.bin Openssl rsautl -decrypt -in c3.bin -inkey pvtkey.pem -out plain1.txt

#hash

Openssl md5 plain.txt Openssl SHA256 plain.txt

#digital signature

Openssl dgst -sha1 -sign pvtkey.pem -out s.bin plain.txt Openssl dgst -sha1 -verify pubkey.pem -signature s.bin plain.txt