**RC4 ALGORITHM IMPLEMENTATION**

**By-  
Aakriti Mehrotra**

**22BCE1954**

**CODE:**

**SERVER**

import socket

import codecs

MOD = 256

def KSA(key):

key\_length = len(key)

S = list(range(MOD))

j = 0

for i in range(MOD):

j = (j + S[i] + key[i % key\_length]) % MOD

S[i], S[j] = S[j], S[i]

return S

def PRGA(S):

i = 0

j = 0

while True:

i = (i + 1) % MOD

j = (j + S[i]) % MOD

S[i], S[j] = S[j], S[i]

K = S[(S[i] + S[j]) % MOD]

yield K

def get\_keystream(key):

S = KSA(key)

return PRGA(S)

def encrypt\_logic(key, text):

key = [ord(c) for c in key]

keystream = get\_keystream(key)

res = []

for c in text:

val = ("%02X" % (c ^ next(keystream)))

res.append(val)

return ''.join(res)

def encrypt(key, plaintext):

plaintext = [ord(c) for c in plaintext]

return encrypt\_logic(key, plaintext)

def decrypt(key, ciphertext):

ciphertext = codecs.decode(ciphertext, 'hex\_codec')

res = encrypt\_logic(key, ciphertext)

return codecs.decode(res, 'hex\_codec').decode('utf-8')

def start\_server(host='127.0.0.1', port=65432):

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.bind((host, port))

s.listen()

print(f"Server listening on {host}:{port}")

conn, addr = s.accept()

with conn:

print('Connected by', addr)

while True:

data = conn.recv(1024)

if not data:

break

encrypted\_message = data.decode('utf-8')

print(f"Received encrypted message: {encrypted\_message}")

key = 'not-so-random-key' # Same key as used by client

decrypted\_message = decrypt(key, encrypted\_message)

print(f"Decrypted message: {decrypted\_message}")

conn.sendall(b'Message received and decrypted')

if \_\_name\_\_ == "\_\_main\_\_":

start\_server()

**CLIENT**

import socket

import codecs

MOD = 256

def KSA(key):

key\_length = len(key)

S = list(range(MOD))

j = 0

for i in range(MOD):

j = (j + S[i] + key[i % key\_length]) % MOD

S[i], S[j] = S[j], S[i]

return S

def PRGA(S):

i = 0

j = 0

while True:

i = (i + 1) % MOD

j = (j + S[i]) % MOD

S[i], S[j] = S[j], S[i]

K = S[(S[i] + S[j]) % MOD]

yield K

def get\_keystream(key):

S = KSA(key)

return PRGA(S)

def encrypt\_logic(key, text):

key = [ord(c) for c in key]

keystream = get\_keystream(key)

res = []

for c in text:

val = ("%02X" % (c ^ next(keystream)))

res.append(val)

return ''.join(res)

def encrypt(key, plaintext):

plaintext = [ord(c) for c in plaintext]

return encrypt\_logic(key, plaintext)

def decrypt(key, ciphertext):

ciphertext = codecs.decode(ciphertext, 'hex\_codec')

res = encrypt\_logic(key, ciphertext)

return codecs.decode(res, 'hex\_codec').decode('utf-8')

def send\_encrypted\_message(message, host='127.0.0.1', port=65432):

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.connect((host, port))

key = 'not-so-random-key'

encrypted\_message = encrypt(key, message)

print(f"Sending encrypted message: {encrypted\_message}")

s.sendall(encrypted\_message.encode('utf-8'))

data = s.recv(1024)

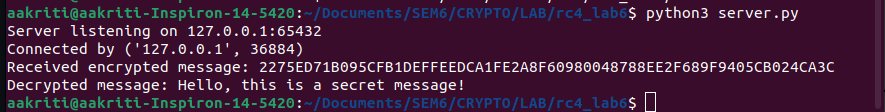
print(f"Received from server: {data.decode('utf-8')}")

if \_\_name\_\_ == "\_\_main\_\_":

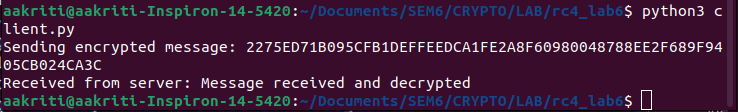
send\_encrypted\_message("Hello, this is a secret message!")

**OUTPUT:**

**SERVER SIDE**

****

**CLIENT SIDE**

****