**CMSC 626 Principles of Computer Security**

**Project**

**Exercise 3**

Faisal Rasheed Khan Shrenik PolySetty

VB02734 AZ61492

[vb02734@umbc.edu](mailto:vb02734@umbc.edu)

1.

1. Team Information:
2. Name: **Faisal Rasheed Khan**

University Id: **VB02734**

1. Name: **Shrenik PolySetty**

University Id: **AZ61492**

1. Secret Key: “Principles of Computer Security”
2. nc -l 12345

nc 130.85.220.34 12345

python3 rc4s.py -k 'Principles of Computer Security' -m 'Hello, it is a nice sunny day and we should enjoy the weather'

python3 rc4s.py <vm1pipe | nc 130.85.220.34 12345 >vm1pipe

nc -l 12345 <vm2pipe | python rc4r.py >vm2pipe

mkfifo vm1pipe

python3 rc4s.py -k 'Principles of Computer Security' -m 'Hello, it is a nice sunny day and we should enjoy the weather' | nc 130.85.220.34 12345

nc -l 12345 | python3 rc4r.py

python3 rc4s.py

python3 diffehellman.py

cat tcpdumpcapture.cap

nano rc4s.py

ls

ifconfig -a

tcpdump -D

tcpdump -n -i ens160 -w tcpdumpcapture.cap host 130.85.121.106 and port 12345

tcpdump -n -i ens160 -w tcpdumpcapture.cap host 130.85.220.34 and port 12345

tcpdump -n -i ens160 -w tcpdumpcapture.cap host 130.85.121.106 and port 12345

tcpdump -r capture.pcap

tcpdump -n -i ens160 -w vm1capture2.pcap host 130.85.220.34

1. The challenges faced were:

* While encrypting and decrypting the pain text and cipher text, faced issue of unable to convert bytes to text

Resolved this issue with the help of encoding and decoding with ‘utf-8’

* Faced issue while connecting to the other Virtual Machine to send the encrypted plain text using netcat command

Resolved the issue by passing the command of netcat to subprocess.Popen()

* While encrypting and decrypting the pain text and cipher text, faced issue of unable to decode with ‘utf-8’ as it was in other format

Resolved by removing unnecessary encoding.

* Faced challenges while using wireshark, tshark as access to install that was not there

Resolved using tcpdump command

* The tcpdump command was capturing every data incoming

Resolve by applying proper filters

* For tcpdump command, which interface to use to capture the data was a problem

Resolved by using this command, tcpdump -D

1. Successfully implemented the RC4 Algorithm to encrypt and decrypt the text over the communicating channels between two virtual machines. Learnt different ways of sending the data via command line arguments or incorporating everything in the python file. Successfully implemented the tcpdump/wireshark capture to capture the real time packets sent/received.
2. References:

[CH02-CompSec4e\_accessible\_L03 (blackboardcdn.com)](https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/5954eb74c7df4/20853920?X-Blackboard-S3-Bucket=blackboard.learn.xythos.prod&X-Blackboard-Expiration=1676851200000&X-Blackboard-Signature=j2dqL5m2BmuM9DMIXtTGCuDKFGLFvL%2FDBsSyiND3QKM%3D&X-Blackboard-Client-Id=100208&X-Blackboard-S3-Region=us-east-1&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27CH02-CompSec4e_accessible_L03.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEP3%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIEQQIAfwL%2B4qxIutoRXaW4vkzvlzQfeM2oh5H21bbDDsAiAC8GDHugloJsm3pz48TRF86QWRfmtCg8OLv9Zqkofw4SrVBAiV%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDU1NjkwMzg2MTM2MSIMYejZSnmP5%2BhufHY5KqkETSykdtU%2FtBOXBybTfaE3R2Cj4EjyFPTYHydpIy0jJaiSXHsQvi9XMPEgHr8%2BQgORzgBpWCxGGsd7juTQ8k7ZdPja7treVuK1evvd89jwba%2FXQNKVa1dvCyBUZ6OdYps6vZRVWxbw55G%2B%2BokjnBC2wXiOs3UGgm%2BwiTGtUqAiuQeDIDKLP8Z0zmLXuODBLDMGz91Z6Irv%2B3ABnAsTiEFeTNxHVvfqrBVPycGedW1tV8temBnLKm5nnGsu45NW9WSrSyXq9l7NZQ%2FXL%2Br8XnvZJie0zDAJx4KXe%2FRRvAlSEuUvQwIUYdpfFcdPDb2gK4NuDoAk6kGcAGn8T6XmaJ31zYuFqVX7FfrpOlEmBZHpW88L2wbp1I2SX8xVaMBfktFb2j3dBdUNWHPgqZPH0FY3f%2FyVUg2v05hjXnbfDZRwnKsBfuOeaUI69SCRubJvZWDXFIp%2BmrwOPiCaKD3GgJ3rJ45Pw5dEsft3c%2BDO8IPfq%2Bwwt0MwvdxdE1DFbZex1DlGIJHEd5gMqKIYwV8nzMeNBt4AlAys8NGt1iT%2FS49Ux9fhupE2JvVkH2Edp%2F0pP9XAJo11PZOlneXVcVIMl5FxsGE4rwjFIGAEzghXXNgbu%2BwB2bFz%2BMFB29cvKLlwWnMjyVh17%2Bjd91oC4bx%2Btv4%2FUMSKM9i3ak8Trvw8SifAY83qqBmgSubZsLXKnanvoYYXDCI1QKpkey3E7RRD1xt3NhWPg2HYByy%2B7DDTgcqfBjqqAQE%2B6HLFsyC%2FDsiXA8w8iyoxWQB7GTs8L6BiyY5bTj6GXvlXPSgZmU00zRuLnJBpipWh23oJo51K4G84%2Fua4Sj8hQ7VwNr97hO0uzc%2F5BOFRgkZcWdXaPVfQ0vaVleoL529rwNdxhegvMT4caLqR2G4RhkOL9Kkmbu4xBdcUi08vfSCqIZQrqbBNrCzWOsSBUObXu2qlptJV27dHjax%2FKttseYwEU9UbgANb&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230219T180000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRYTKDDXUVP%2F20230219%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=85d228a59c4b86a35927594bc0046b89cce45cc8be)

[How To Use Netcat to Establish and Test TCP and UDP Connections | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-netcat-to-establish-and-test-tcp-and-udp-connections)

[L07-CH21-CompSec4e\_accessible (blackboardcdn.com)](https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/5954eb74c7df4/21143920?X-Blackboard-S3-Bucket=blackboard.learn.xythos.prod&X-Blackboard-Expiration=1678093200000&X-Blackboard-Signature=DW6cy%2F8kGRywyR4Uz1aTD4LE0GlqRY%2BGXV%2ByhXk90PI%3D&X-Blackboard-Client-Id=100208&X-Blackboard-S3-Region=us-east-1&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27L07-CH21-CompSec4e_accessible.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEFUaCXVzLWVhc3QtMSJHMEUCICtOHmLIPqUPU6iwUCKXjzDvtSxgDiV2xUashAA2PfioAiEAnFjnrbRJxZhsBSMk2VU6amhzrfGGpa5hlYbyrbdY5joquwUI%2Fv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDBhfBAZQKdP0I%2FSlGSqPBZk9cCUrcoqcPv8e0Zzf8cEkOn4Jlye6gIPDBR2bR9KTt7dxCMwpS77vX5uZBu5GB8NUlpL8xppxCgBFVMt2feokJgqlqTBhtjaIEopVV167ZS1Ie8CV%2B2O4CuHyT%2FyLFxrGl0O%2Fqyb5ShsSquUScrWBGqGxbAXFvzg8wVx%2FS3o94fxlE%2FYcACn7IxfHaL2dF8Zd9K7dh%2BSG6G7iZ%2FVezsqEKSUpbP2LdxtGSlhItblHX58l7twvXLbvaH6mpZl1HtvknGo7u2ascB9BE7gCR5PWHpcuKurh2TkkOYIZsjEbRnM2hnq%2FlKjKS5C04cPHlEpd%2BenONZBuIftPLOHtv%2BdxnIsfymP%2BdAg674n1wpoMHXiyMm1M%2BpyTQjhZ78N0yY9x7GM7rCx8HFGFMiRm2hjyhxH7hNjpPNZN%2BuxSLeGQPf9HEv87iEVAzbHIrfeLRcxZs8KcHdx9nHhR54dbC3hVNExW9vG2f9kiAOdTx6EJnV2y1wslTSRX9OO2CaA6Zlp0lZ73QIFjpT0LGQL5Q4xtrNZqFB4ZfP4lLQfUw66Y30FknImiqAyi45nZzyjtjqZYWdDg6MveBNfDPVTeumpnFdgvsEJNBZ3rhZJLlpBFyK2qakNnRVEDL5vuKmq3IhnJvP7bO7u56%2Bdy8%2FDFK50umZGJMA8fSG%2BI14eS3u1JKzA6W87nUudtJxQ5sMt4N2AsucfzElAK5nz86pkqZxdaj3i4vMkOOvUw%2FWEf%2F7atvbimtRrCYu2lCRwlHhW%2BV%2FhRqblooDSKIAF7aEOCr%2BuXeX%2F3lMeXKx%2BVae3FIP29KalC2LuhiGGxiPgJnfwyV8efg1mhU3zZYuici2jKcRjlHy4vAk%2BpEh67rlIpLQkwmt%2BVoAY6sQH3enpFBbb6hQmSi%2BiBZNhaGMmamKs2eKfa8RYwGlGwUpC3OZCDesZkvtG0a%2BsazJvn9nRf%2BaWFNOvF%2Fs9oXDYOCb%2FTefMSydEd7q%2FGU0Bb96LvcglO3A5Qs812%2Fksy7VHliGFSjvKaOy0x3l3%2BFzC52637EGmayEJoW%2FpaR2gXOPwaOSXvG183vsOG%2BloR%2FArMxL4OAc0agtb74J48Hk1xIWGcGH%2F4lNvEnuluAQQFZh8%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230306T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=216)

[Primitive Root - Algorithms for Competitive Programming (cp-algorithms.com)](https://cp-algorithms.com/algebra/primitive-root.html)

[tcp\_client.py](file:///C:\Users\juver\Downloads\tcp_client.py)

[tcp\_server.py](file:///C:\Users\juver\Downloads\tcp_server.py)

[RC4-KeyGeneration(1).py](file:///C:\Users\juver\Downloads\RC4-KeyGeneration(1).py)

[subprocess — Subprocess management — Python 3.11.2 documentation](https://docs.python.org/3/library/subprocess.html)

[Tcpdump Command in Linux | Linuxize](https://linuxize.com/post/tcpdump-command-in-linux/)

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

2.

**Code on VM1(Faisal Rasheed Khan)** 130.85.121.106**:**

**rc4.py:**

def KSA(key):

#Key Scheduling Algorithm (KSA) for RC4.

Keylength = len(key)

# Initialize permutation array

S = list(range(256))

# Use key to permute the array

j = 0

for i in range(256):

j = (j + S[i] + key[i % keylength]) % 256

S[i], S[j] = S[j], S[i] # Swap values

return S

def PRGA(S, length):

#Pseudo-Random Generation Algorithm (PRGA) for RC4.

I = j = 0

keystream = []

for \_ in range(length):

i = (i + 1) % 256

j = (j + S[i]) % 256

S[i], S[j] = S[j], S[i] # Swap values

k = S[(S[i] + S[j]) % 256]

keystream.append(k)

return keystream

def encrypt(key, data):

#RC4 encryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return str(output)

def decrypt(key, data):

#RC4 decryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return bytes(output)

key = b”Principles of Computer Security”

**rc4s.py:**

import argparse

import subprocess

import signal

import rc4

import socket

import time

'''

import subprocess

# Create the named pipe

subprocess.call('mkfifo vm1pipe', shell=True)

# Open the pipe for writing

pipe = open('vm1pipe', 'w')

# Start the nc command to listen on port 12345 and write to the pipe

#nc\_proc

NC = subprocess.Popen(['nc', '130.85.220.34', '12345'], stdout=pipe, stderr=subprocess.STDOUT)

# Wait for 1 second to make sure nc is fully established

time.sleep(1)

# Open the pipe for reading and pass it to rc4s

with open('vm1pipe', 'r') as input\_pipe:

subprocess.call(['python3', 'rc4s.py'], stdin=input\_pipe)

# Close the pipe and terminate the nc process

pipe.close()

NC.terminate()

parser = argparse.ArgumentParser(description='Encrypt and send message using RC4')

parser.add\_argument('-k', '--key', type=str, required=True, help='key for RC4 encryption')

parser.add\_argument('-m', '--message', type=str, required=True, help='message to be encrypted')

#parser.add\_argument('ip', type=str, help='IP address of receiver')

#parser.add\_argument('port', type=int, help='port of receiver')

args = parser.parse\_args()

rc4.key=args.key

en\_messageargs.message

'''

# the netcat command to run and redirect its input/output to pipes

#pipelining the 2 VM inorder to encrypt and decrypt using shared key

os\_cmd = ['nc', '130.85.220.34', '12345']

#NetCat

NC = subprocess.Popen(os\_cmd, stdin=subprocess.PIPE, stdout=subprocess.PIPE)

print("Shared Key : " + rc4.key.decode("utf-8"))

tcpdump\_cmd1 = ['tcpdump', '-i', 'ens160', '-s','0','w','capture.pcap','&', 'host', '130.85.121.106', 'and', '130.85.220.34']

# to store the tcpdump in file

with open('tcpdump3.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd1, stdout=outfile, stderr=subprocess.STDOUT)

# tcpdump capture / wireshark capture, can use either .cap or .pcap

tcpdump\_cmd = ['tcpdump', '-n', '-i', 'ens160', '-w','tcpdumpcapture.cap', 'host', '130.85.121.106', 'and', 'port', '12345']

# to store the packets of wireshark capture in .txt file

with open('tcpdump1.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd, stdout=outfile, stderr=subprocess.STDOUT)

en\_message=bytes('Hello, it is a nice sunny day and we should enjoy the weather','utf-8')

#Encrypt the plain text to send to the receiver

encrypted\_message = rc4.encrypt(rc4.key,en\_message)

'''with open("vm2pipe", "w") as pipe:

pipe.write(encrypted\_message.encode().decode())

'''

NC.stdin.write(encrypted\_message.encode() + b'\n')

NC.stdin.flush()

'''

srvr\_addr = ('130.85.220.34', 9999)

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.connect(srvr\_addr)

sock.delay(5)

sent = sock.send(encrypted\_message.encode())

#sent = sock.send(msg.encode())

time.sleep(1)

#sock.close()

'''

#tcpdump\_proc.send\_signal(signal.SIGINT)

# receive the encrypted message from the receiver using netcat

#received\_message = subprocess.check\_output(nc\_cmd)

'''with open("vm2pipe", "r") as pipe:

cipher = pipe.read().strip()

'''

cipher = NC.stdout.readline().decode().strip()

'''

sock.bind(srvr\_addr)

sock.listen(5)

connsock, client = sock.accept()

cipher = connsock.recv(20)

'''

cipher=cipher[1:-1].split(", ")

cipher=[eval(i) for i in cipher]

cipher=bytes(cipher)

decrypted\_message = rc4.decrypt(rc4.key,cipher)

#print the decrypted message

print(decrypted\_message.decode("utf-8"))

#sock.close()

tcpdump\_proc.send\_signal(signal.SIGINT)

**Code on VM2(Shrenik PolySetty)** 130.85.220.34**:**

**rc4.py:**

def KSA(key):

#Key Scheduling Algorithm (KSA) for RC4.

Keylength = len(key)

# Initialize permutation array

S = list(range(256))

# Use key to permute the array

j = 0

for i in range(256):

j = (j + S[i] + key[i % keylength]) % 256

S[i], S[j] = S[j], S[i] # Swap values

return S

def PRGA(S, length):

#Pseudo-Random Generation Algorithm (PRGA) for RC4.

I = j = 0

keystream = []

for \_ in range(length):

i = (i + 1) % 256

j = (j + S[i]) % 256

S[i], S[j] = S[j], S[i] # Swap values

k = S[(S[i] + S[j]) % 256]

keystream.append(k)

return keystream

def encrypt(key, data):

#RC4 encryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return str(output)

def decrypt(key, data):

#RC4 decryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return bytes(output)

key = b”Principles of Computer Security”

**rc4r.py:**

import argparse

import subprocess

import signal

import rc4

#from rc4 import RC4

'''parser = argparse.ArgumentParser(description='Receive and decrypt message using RC4')

#parser.add\_argument('port', type=int, help='port to listen on')

parser.add\_argument('-k', '--key', type=str, required=True, help='key for RC4 encryption')

#parser.add\_argument('port', type=int, help='port to listen on')

args = parser.parse\_args()'''

os\_cmd = ['nc', '-l', '12345']

NC = subprocess.Popen(os\_cmd, stdin=subprocess.PIPE, stdout=subprocess.PIPE)

#with open("vm1pipe","r") as pipe:

# B=pipe.read().strip()

'''

rc4.key=NC.stdout.readline().decode("utf-8").strip()

print("Shared Key :")

print(rc4.key)

rc4.key=bytes(rc4.key,"utf-8")

'''

print("Shared key : "+rc4.key.decode("utf-8"))

tcpdump\_cmd1 = ['tcpdump', '-i', 'ens160', '-s','0','w','capture.pcap','&', 'host', '130.85.220.34', 'and', '130.85.121.106']

# to store the tcpdump in file

with open('tcpdump3.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd1, stdout=outfile, stderr=subprocess.STDOUT)

cipher=NC.stdout.readline().decode("utf-8").strip()

cipher=cipher[1:-1].split(", ")

cipher=[eval(i) for i in cipher]

cipher=bytes(cipher)

decrypted\_message = rc4.decrypt(rc4.key,cipher)

# print the decrypted message

print(decrypted\_message.decode("utf-8"))

# encrypt the response message using RC4

response\_message = 'Busy with assignment right now.'

#response\_message = b'Busy with assignment right now.'

encrypted\_response = rc4.encrypt(rc4.key,bytes(response\_message,'utf-8'))

tcpdump\_cmd = ['tcpdump', '-n', '-i', 'ens160', '-w','tcpdumpcapture.cap', 'host', '130.85.220.34', 'and', 'port', '12345']

# to store the tcpdump in file

with open('tcpdump1.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd, stdout=outfile, stderr=subprocess.STDOUT)

#with open("vm1pipe","w") as pipe:

NC.stdin.write(str(encrypted\_response).encode("utf-8")+ b'\n')

NC.stdin.flush()

tcpdump\_proc.send\_signal(signal.SIGINT)

'''import sys

import subprocess

# read the encrypted message from netcat

encrypted\_message = sys.stdin.buffer.read()

# decrypt the message using RC4

cipher = RC4(args.key.encode())

decrypted\_message = cipher.decrypt(encrypted\_message)

# display the decrypted message

print(decrypted\_message.decode())

# send the decrypted message back to sender using netcat

nc\_cmd = f'nc {args.ip} {args.port} >{sys.stdout.fileno()}'

subprocess.run(nc\_cmd, input=decrypted\_message, shell=True)

'''

**Exercise 4**

Faisal Rasheed Khan Shrenik PolySetty

VB02734 AZ61492

[vb02734@umbc.edu](mailto:vb02734@umbc.edu)

1.

1. Team Information:
2. Name: **Faisal Rasheed Khan**

University Id: **VB02734**

1. Name: **Shrenik PolySetty**

University Id: **AZ61492**

1. Prime Number, P = 1065601

Primitive root, α = 7

Xa = 139278

Xb = 111689

Key = 159571

1. nc -l 12345

nc 130.85.220.34 12345

python3 diffehellman.py | nc 130.85.220.34 12345

python3 diffehellman.py <vm1pipe | nc 130.85.220.34 12345 >vm1pipe

nc -l 12345 <vm2pipe | python diffehellman.py >vm2pipe

mkfifo vm1pipe

nc -l 12345 | python3 diffehellman.py

python3 diffehellman.py

cat tcpdumpcapture.cap

nano diffehellman.py

ls

ifconfig -a

tcpdump -D

tcpdump -n -i ens160 -w tcpdumpcapture.cap host 130.85.121.106 and port 12345

tcpdump -n -i ens160 -w tcpdumpcapture.cap host 130.85.220.34 and port 12345

tcpdump -n -i ens160 -w tcpdumpcapture.cap host 130.85.121.106 and port 12345

tcpdump -r capture.pcap

tcpdump -n -i ens160 -w vm1capture2.pcap host 130.85.220.34

1. The challenges faced were:

* While encrypting and decrypting the key, faced issue because key was of int type text

Resolved this issue with the help of converting the int to str ,encoding and decoding with ‘utf-8’

* Faced issue while connecting to the other Virtual Machine to send the encrypted plain text using netcat command

Resolved the issue by passing the command of netcat to subprocess.Popen()

* While encrypting and decrypting the pain text and cipher text, faced issue of unable to decode with ‘utf-8’ as it was in other format

Resolved by removing unnecessary encoding.

* Faced challenges while using wireshark, tshark as access to install that was not there

Resolved using tcpdump command

* The tcpdump command was capturing every data incoming

Resolve by applying proper filters

* For tcpdump command, which interface to use to capture the data was a problem

Resolved by using this command, tcpdump -D

1. Successfully implemented the Diffe-Hellman Key Exchange Algorithm to encrypt and decrypt the text over the communicating channels between two virtual machines while calculating the shared key with their respective private keys, where keys are generated using prime number, primitive root and private key. Learnt different ways of sending the data via command line arguments or incorporating everything in the python file. Successfully implemented the tcpdump/wireshark capture to capture the real time packets sent/received. Can store tcpdump/wireshark capture using .pcap or .cap extension.
2. References:

[CH02-CompSec4e\_accessible\_L03 (blackboardcdn.com)](https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/5954eb74c7df4/20853920?X-Blackboard-S3-Bucket=blackboard.learn.xythos.prod&X-Blackboard-Expiration=1676851200000&X-Blackboard-Signature=j2dqL5m2BmuM9DMIXtTGCuDKFGLFvL%2FDBsSyiND3QKM%3D&X-Blackboard-Client-Id=100208&X-Blackboard-S3-Region=us-east-1&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27CH02-CompSec4e_accessible_L03.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEP3%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIEQQIAfwL%2B4qxIutoRXaW4vkzvlzQfeM2oh5H21bbDDsAiAC8GDHugloJsm3pz48TRF86QWRfmtCg8OLv9Zqkofw4SrVBAiV%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDU1NjkwMzg2MTM2MSIMYejZSnmP5%2BhufHY5KqkETSykdtU%2FtBOXBybTfaE3R2Cj4EjyFPTYHydpIy0jJaiSXHsQvi9XMPEgHr8%2BQgORzgBpWCxGGsd7juTQ8k7ZdPja7treVuK1evvd89jwba%2FXQNKVa1dvCyBUZ6OdYps6vZRVWxbw55G%2B%2BokjnBC2wXiOs3UGgm%2BwiTGtUqAiuQeDIDKLP8Z0zmLXuODBLDMGz91Z6Irv%2B3ABnAsTiEFeTNxHVvfqrBVPycGedW1tV8temBnLKm5nnGsu45NW9WSrSyXq9l7NZQ%2FXL%2Br8XnvZJie0zDAJx4KXe%2FRRvAlSEuUvQwIUYdpfFcdPDb2gK4NuDoAk6kGcAGn8T6XmaJ31zYuFqVX7FfrpOlEmBZHpW88L2wbp1I2SX8xVaMBfktFb2j3dBdUNWHPgqZPH0FY3f%2FyVUg2v05hjXnbfDZRwnKsBfuOeaUI69SCRubJvZWDXFIp%2BmrwOPiCaKD3GgJ3rJ45Pw5dEsft3c%2BDO8IPfq%2Bwwt0MwvdxdE1DFbZex1DlGIJHEd5gMqKIYwV8nzMeNBt4AlAys8NGt1iT%2FS49Ux9fhupE2JvVkH2Edp%2F0pP9XAJo11PZOlneXVcVIMl5FxsGE4rwjFIGAEzghXXNgbu%2BwB2bFz%2BMFB29cvKLlwWnMjyVh17%2Bjd91oC4bx%2Btv4%2FUMSKM9i3ak8Trvw8SifAY83qqBmgSubZsLXKnanvoYYXDCI1QKpkey3E7RRD1xt3NhWPg2HYByy%2B7DDTgcqfBjqqAQE%2B6HLFsyC%2FDsiXA8w8iyoxWQB7GTs8L6BiyY5bTj6GXvlXPSgZmU00zRuLnJBpipWh23oJo51K4G84%2Fua4Sj8hQ7VwNr97hO0uzc%2F5BOFRgkZcWdXaPVfQ0vaVleoL529rwNdxhegvMT4caLqR2G4RhkOL9Kkmbu4xBdcUi08vfSCqIZQrqbBNrCzWOsSBUObXu2qlptJV27dHjax%2FKttseYwEU9UbgANb&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230219T180000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRYTKDDXUVP%2F20230219%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=85d228a59c4b86a35927594bc0046b89cce45cc8be)

[How To Use Netcat to Establish and Test TCP and UDP Connections | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-netcat-to-establish-and-test-tcp-and-udp-connections)

[L07-CH21-CompSec4e\_accessible (blackboardcdn.com)](https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/5954eb74c7df4/21143920?X-Blackboard-S3-Bucket=blackboard.learn.xythos.prod&X-Blackboard-Expiration=1678093200000&X-Blackboard-Signature=DW6cy%2F8kGRywyR4Uz1aTD4LE0GlqRY%2BGXV%2ByhXk90PI%3D&X-Blackboard-Client-Id=100208&X-Blackboard-S3-Region=us-east-1&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27L07-CH21-CompSec4e_accessible.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEFUaCXVzLWVhc3QtMSJHMEUCICtOHmLIPqUPU6iwUCKXjzDvtSxgDiV2xUashAA2PfioAiEAnFjnrbRJxZhsBSMk2VU6amhzrfGGpa5hlYbyrbdY5joquwUI%2Fv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDBhfBAZQKdP0I%2FSlGSqPBZk9cCUrcoqcPv8e0Zzf8cEkOn4Jlye6gIPDBR2bR9KTt7dxCMwpS77vX5uZBu5GB8NUlpL8xppxCgBFVMt2feokJgqlqTBhtjaIEopVV167ZS1Ie8CV%2B2O4CuHyT%2FyLFxrGl0O%2Fqyb5ShsSquUScrWBGqGxbAXFvzg8wVx%2FS3o94fxlE%2FYcACn7IxfHaL2dF8Zd9K7dh%2BSG6G7iZ%2FVezsqEKSUpbP2LdxtGSlhItblHX58l7twvXLbvaH6mpZl1HtvknGo7u2ascB9BE7gCR5PWHpcuKurh2TkkOYIZsjEbRnM2hnq%2FlKjKS5C04cPHlEpd%2BenONZBuIftPLOHtv%2BdxnIsfymP%2BdAg674n1wpoMHXiyMm1M%2BpyTQjhZ78N0yY9x7GM7rCx8HFGFMiRm2hjyhxH7hNjpPNZN%2BuxSLeGQPf9HEv87iEVAzbHIrfeLRcxZs8KcHdx9nHhR54dbC3hVNExW9vG2f9kiAOdTx6EJnV2y1wslTSRX9OO2CaA6Zlp0lZ73QIFjpT0LGQL5Q4xtrNZqFB4ZfP4lLQfUw66Y30FknImiqAyi45nZzyjtjqZYWdDg6MveBNfDPVTeumpnFdgvsEJNBZ3rhZJLlpBFyK2qakNnRVEDL5vuKmq3IhnJvP7bO7u56%2Bdy8%2FDFK50umZGJMA8fSG%2BI14eS3u1JKzA6W87nUudtJxQ5sMt4N2AsucfzElAK5nz86pkqZxdaj3i4vMkOOvUw%2FWEf%2F7atvbimtRrCYu2lCRwlHhW%2BV%2FhRqblooDSKIAF7aEOCr%2BuXeX%2F3lMeXKx%2BVae3FIP29KalC2LuhiGGxiPgJnfwyV8efg1mhU3zZYuici2jKcRjlHy4vAk%2BpEh67rlIpLQkwmt%2BVoAY6sQH3enpFBbb6hQmSi%2BiBZNhaGMmamKs2eKfa8RYwGlGwUpC3OZCDesZkvtG0a%2BsazJvn9nRf%2BaWFNOvF%2Fs9oXDYOCb%2FTefMSydEd7q%2FGU0Bb96LvcglO3A5Qs812%2Fksy7VHliGFSjvKaOy0x3l3%2BFzC52637EGmayEJoW%2FpaR2gXOPwaOSXvG183vsOG%2BloR%2FArMxL4OAc0agtb74J48Hk1xIWGcGH%2F4lNvEnuluAQQFZh8%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230306T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=216)

[Primitive Root - Algorithms for Competitive Programming (cp-algorithms.com)](https://cp-algorithms.com/algebra/primitive-root.html)

[tcp\_client.py](file:///C:\Users\juver\Downloads\tcp_client.py)

[tcp\_server.py](file:///C:\Users\juver\Downloads\tcp_server.py)

[RC4-KeyGeneration(1).py](file:///C:\Users\juver\Downloads\RC4-KeyGeneration(1).py)

[subprocess — Subprocess management — Python 3.11.2 documentation](https://docs.python.org/3/library/subprocess.html)

[Tcpdump Command in Linux | Linuxize](https://linuxize.com/post/tcpdump-command-in-linux/)

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

2.

**Code on VM1(Faisal Rasheed Khan)** 130.85.121.106**:**

**rc4.py:**

def KSA(key):

#Key Scheduling Algorithm (KSA) for RC4.

Keylength = len(key)

# Initialize permutation array

S = list(range(256))

# Use key to permute the array

j = 0

for i in range(256):

j = (j + S[i] + key[i % keylength]) % 256

S[i], S[j] = S[j], S[i] # Swap values

return S

def PRGA(S, length):

#Pseudo-Random Generation Algorithm (PRGA) for RC4.

I = j = 0

keystream = []

for \_ in range(length):

i = (i + 1) % 256

j = (j + S[i]) % 256

S[i], S[j] = S[j], S[i] # Swap values

k = S[(S[i] + S[j]) % 256]

keystream.append(k)

return keystream

def encrypt(key, data):

#RC4 encryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return str(output)

def decrypt(key, data):

#RC4 decryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return bytes(output)

**diffehellman.py:**

import subprocess

import random

import signal

import rc4

# the prime and primitive root.

p = 1065601

# took the value by finding the value from the function defined below

g = 7

def is\_primitive\_root(g, p):

"""

Check if g is a primitive root of prime p

"""

s = set(pow(g, i, p) for i in range(1, p))

return len(s) == p-1

def find\_primitive\_root(p):

"""

Find a primitive root of prime p

"""

for g in range(2, p):

if is\_primitive\_root(g, p):

return g

return None

#gg = find\_primitive\_root(p)

#print(gg)

# Generate a random private key

private\_key = random.randint(1000, p-2)

print("Private Key YA :",private\_key)

# the netcat command to run and redirect its input/output to pipes

os\_cmd = ['nc', '130.85.220.34', '12345']

NC = subprocess.Popen(os\_cmd, stdin=subprocess.PIPE, stdout=subprocess.PIPE)

interface = 'ens160'

ip\_address = '130.85.220.34'

output\_file = 'sender.pcap'

#tcpdump\_cmd1 = f'tcpdump -i {interface} src {ip\_address} -w {output\_file}'

#tcpdump\_cmd2= f'tcpdump -i ens160 -s 0 -w capture.pcap &'

tcpdump\_cmd2= ['tcpdump', '-i', 'ens160', '-s', '0', '-w' ,'capture.pcap', '&','host', '130.85.220.34']

#subprocess.run(tcpdump\_cmd2, shell=True)

with open('tcpdump3.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd2, stdout=outfile, stderr=subprocess.STDOUT)

tcpdump\_cmd = ['tcpdump', '-n', '-i', 'ens160', '-w','vm1capture.pcap', 'host', '130.85.121.106', 'and', 'port', '12345']

# to store the tcpdump in file

with open('tcpdump2.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd, stdout=outfile, stderr=subprocess.STDOUT)

#Calculate YA = g^private\_key mod p

YA = pow(g, private\_key, p)

# Send YA to receiver

NC.stdin.write(str(YA).encode("utf-8") + b'\n')

NC.stdin.flush()

# Receive YA from sender

YB = int(NC.stdout.readline().decode("utf-8").strip())

# Calculate the shared secret key K = YA^private\_key mod p

K = pow(YB, private\_key, p)

print('Shared Secret Key:', K)

key = bytes(str(K), 'utf-8')

#send message to VM2

#send=input("Send message : ")

send="Hello, it is a nice sunny day and we should enjoy the weather"

plaintext = bytes(send, 'utf-8')

ciphertext\_encrypt = rc4.encrypt(key, plaintext)

#ciphertext\_encrypt = RC4\_encryption(key, plaintext)

NC.stdin.write(ciphertext\_encrypt.encode("utf-8") + b'\n')

NC.stdin.flush()

# received message from VM2

cipher\_decrypt = NC.stdout.readline().decode("utf-8").strip()

cipher\_decrypt=cipher\_decrypt[1:-1].split(", ")

cipher\_decrypt = [eval(i) for i in cipher\_decrypt]

cipher\_decrypt = bytes(cipher\_decrypt)

#,"utf-8")

plain\_decrypt = rc4.decrypt(key, cipher\_decrypt)

#decrypted = RC4\_decryption(key, response)

print("Received message :", plain\_decrypt.decode("utf-8"))

tcpdump\_proc.send\_signal(signal.SIGINT)

**Code on VM2(Shrenik PolySetty)** 130.85.220.34**:**

**rc4.py:**

def KSA(key):

#Key Scheduling Algorithm (KSA) for RC4.

Keylength = len(key)

# Initialize permutation array

S = list(range(256))

# Use key to permute the array

j = 0

for i in range(256):

j = (j + S[i] + key[i % keylength]) % 256

S[i], S[j] = S[j], S[i] # Swap values

return S

def PRGA(S, length):

#Pseudo-Random Generation Algorithm (PRGA) for RC4.

I = j = 0

keystream = []

for \_ in range(length):

i = (i + 1) % 256

j = (j + S[i]) % 256

S[i], S[j] = S[j], S[i] # Swap values

k = S[(S[i] + S[j]) % 256]

keystream.append(k)

return keystream

def encrypt(key, data):

#RC4 encryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return str(output)

def decrypt(key, data):

#RC4 decryption algorithm.

S = KSA(key)

keystream = PRGA(S, len(data))

output = [data[i] ^ keystream[i] for i in range(len(data))]

return bytes(output)

**diffehellman.py:**

import subprocess

import random

import signal

import rc4

# the prime and primitive root.

p = 1065601

g = 7

# Generate a random private key

private\_key = random.randint(1000, p-2)

print("private key YB:",private\_key)

# the netcat command to run and redirect its input/output to pipes

os\_cmd = ['nc', '-l', '12345']

NC = subprocess.Popen(os\_cmd, stdin=subprocess.PIPE, stdout=subprocess.PIPE)

tcpdump\_cmd1 = ['tcpdump', '-i', 'ens160', '-s','0','w','capture.pcap','&', 'host', '130.85.220.34', 'and', '130.85.121.106']

tcpdump\_cmd = ['tcpdump', '-n', '-i', 'ens160', '-w','vm1capture.pcap', 'host', '130.85.220.34', 'and', 'port', '12345']

# to store the tcpdump in file

with open('tcpdump3.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd1, stdout=outfile, stderr=subprocess.STDOUT)

# to store the tcpdump in file

with open('tcpdump2.txt', 'w') as outfile:

tcpdump\_proc = subprocess.Popen(tcpdump\_cmd, stdout=outfile, stderr=subprocess.STDOUT)

# Receive YA from computer A

YA = int(NC.stdout.readline().decode("utf-8").strip())

# Calculate YB = g^private\_key mod p

YB = pow(g, private\_key, p)

# Send YB to computer A

NC.stdin.write(str(YB).encode("utf-8") + b'\n')

NC.stdin.flush()

# Calculate the shared secret key K = YA^private\_key mod p

K = pow(YA, private\_key, p)

print('Shared Secret Key:', K)

key = bytes(str(K), 'utf-8')

#received message from VM2

cipher\_decrypt = NC.stdout.readline().decode("utf-8").strip()

cipher\_decrypt=cipher\_decrypt[1:-1].split(", ")

cipher\_decrypt = [eval(i) for i in cipher\_decrypt]

cipher\_decrypt = bytes(cipher\_decrypt)

#,"utf-8")

plain\_decrypt = rc4.decrypt(key, cipher\_decrypt)

print("Received message :", plain\_decrypt.decode("utf-8"))

#send message to VM2

#send=input("Send message : ")

send="Busy with assignment right now. "

plaintext = bytes(send, 'utf-8')

ciphertext\_encrypt = rc4.encrypt(key, plaintext)

#ciphertext\_encrypt = RC4\_encryption(key, plaintext)

NC.stdin.write(ciphertext\_encrypt.encode("utf-8") + b'\n')

NC.stdin.flush()

tcpdump\_proc.send\_signal(signal.SIGINT)

References:

[CH02-CompSec4e\_accessible\_L03 (blackboardcdn.com)](https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/5954eb74c7df4/20853920?X-Blackboard-S3-Bucket=blackboard.learn.xythos.prod&X-Blackboard-Expiration=1676851200000&X-Blackboard-Signature=j2dqL5m2BmuM9DMIXtTGCuDKFGLFvL%2FDBsSyiND3QKM%3D&X-Blackboard-Client-Id=100208&X-Blackboard-S3-Region=us-east-1&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27CH02-CompSec4e_accessible_L03.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEP3%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIEQQIAfwL%2B4qxIutoRXaW4vkzvlzQfeM2oh5H21bbDDsAiAC8GDHugloJsm3pz48TRF86QWRfmtCg8OLv9Zqkofw4SrVBAiV%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAAaDDU1NjkwMzg2MTM2MSIMYejZSnmP5%2BhufHY5KqkETSykdtU%2FtBOXBybTfaE3R2Cj4EjyFPTYHydpIy0jJaiSXHsQvi9XMPEgHr8%2BQgORzgBpWCxGGsd7juTQ8k7ZdPja7treVuK1evvd89jwba%2FXQNKVa1dvCyBUZ6OdYps6vZRVWxbw55G%2B%2BokjnBC2wXiOs3UGgm%2BwiTGtUqAiuQeDIDKLP8Z0zmLXuODBLDMGz91Z6Irv%2B3ABnAsTiEFeTNxHVvfqrBVPycGedW1tV8temBnLKm5nnGsu45NW9WSrSyXq9l7NZQ%2FXL%2Br8XnvZJie0zDAJx4KXe%2FRRvAlSEuUvQwIUYdpfFcdPDb2gK4NuDoAk6kGcAGn8T6XmaJ31zYuFqVX7FfrpOlEmBZHpW88L2wbp1I2SX8xVaMBfktFb2j3dBdUNWHPgqZPH0FY3f%2FyVUg2v05hjXnbfDZRwnKsBfuOeaUI69SCRubJvZWDXFIp%2BmrwOPiCaKD3GgJ3rJ45Pw5dEsft3c%2BDO8IPfq%2Bwwt0MwvdxdE1DFbZex1DlGIJHEd5gMqKIYwV8nzMeNBt4AlAys8NGt1iT%2FS49Ux9fhupE2JvVkH2Edp%2F0pP9XAJo11PZOlneXVcVIMl5FxsGE4rwjFIGAEzghXXNgbu%2BwB2bFz%2BMFB29cvKLlwWnMjyVh17%2Bjd91oC4bx%2Btv4%2FUMSKM9i3ak8Trvw8SifAY83qqBmgSubZsLXKnanvoYYXDCI1QKpkey3E7RRD1xt3NhWPg2HYByy%2B7DDTgcqfBjqqAQE%2B6HLFsyC%2FDsiXA8w8iyoxWQB7GTs8L6BiyY5bTj6GXvlXPSgZmU00zRuLnJBpipWh23oJo51K4G84%2Fua4Sj8hQ7VwNr97hO0uzc%2F5BOFRgkZcWdXaPVfQ0vaVleoL529rwNdxhegvMT4caLqR2G4RhkOL9Kkmbu4xBdcUi08vfSCqIZQrqbBNrCzWOsSBUObXu2qlptJV27dHjax%2FKttseYwEU9UbgANb&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230219T180000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=ASIAYDKQORRYTKDDXUVP%2F20230219%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=85d228a59c4b86a35927594bc0046b89cce45cc8be)

[How To Use Netcat to Establish and Test TCP and UDP Connections | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-netcat-to-establish-and-test-tcp-and-udp-connections)

[L07-CH21-CompSec4e\_accessible (blackboardcdn.com)](https://learn-us-east-1-prod-fleet01-xythos.content.blackboardcdn.com/blackboard.learn.xythos.prod/5954eb74c7df4/21143920?X-Blackboard-S3-Bucket=blackboard.learn.xythos.prod&X-Blackboard-Expiration=1678093200000&X-Blackboard-Signature=DW6cy%2F8kGRywyR4Uz1aTD4LE0GlqRY%2BGXV%2ByhXk90PI%3D&X-Blackboard-Client-Id=100208&X-Blackboard-S3-Region=us-east-1&response-cache-control=private%2C%20max-age%3D21600&response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27L07-CH21-CompSec4e_accessible.pdf&response-content-type=application%2Fpdf&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEFUaCXVzLWVhc3QtMSJHMEUCICtOHmLIPqUPU6iwUCKXjzDvtSxgDiV2xUashAA2PfioAiEAnFjnrbRJxZhsBSMk2VU6amhzrfGGpa5hlYbyrbdY5joquwUI%2Fv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FARAAGgw1NTY5MDM4NjEzNjEiDBhfBAZQKdP0I%2FSlGSqPBZk9cCUrcoqcPv8e0Zzf8cEkOn4Jlye6gIPDBR2bR9KTt7dxCMwpS77vX5uZBu5GB8NUlpL8xppxCgBFVMt2feokJgqlqTBhtjaIEopVV167ZS1Ie8CV%2B2O4CuHyT%2FyLFxrGl0O%2Fqyb5ShsSquUScrWBGqGxbAXFvzg8wVx%2FS3o94fxlE%2FYcACn7IxfHaL2dF8Zd9K7dh%2BSG6G7iZ%2FVezsqEKSUpbP2LdxtGSlhItblHX58l7twvXLbvaH6mpZl1HtvknGo7u2ascB9BE7gCR5PWHpcuKurh2TkkOYIZsjEbRnM2hnq%2FlKjKS5C04cPHlEpd%2BenONZBuIftPLOHtv%2BdxnIsfymP%2BdAg674n1wpoMHXiyMm1M%2BpyTQjhZ78N0yY9x7GM7rCx8HFGFMiRm2hjyhxH7hNjpPNZN%2BuxSLeGQPf9HEv87iEVAzbHIrfeLRcxZs8KcHdx9nHhR54dbC3hVNExW9vG2f9kiAOdTx6EJnV2y1wslTSRX9OO2CaA6Zlp0lZ73QIFjpT0LGQL5Q4xtrNZqFB4ZfP4lLQfUw66Y30FknImiqAyi45nZzyjtjqZYWdDg6MveBNfDPVTeumpnFdgvsEJNBZ3rhZJLlpBFyK2qakNnRVEDL5vuKmq3IhnJvP7bO7u56%2Bdy8%2FDFK50umZGJMA8fSG%2BI14eS3u1JKzA6W87nUudtJxQ5sMt4N2AsucfzElAK5nz86pkqZxdaj3i4vMkOOvUw%2FWEf%2F7atvbimtRrCYu2lCRwlHhW%2BV%2FhRqblooDSKIAF7aEOCr%2BuXeX%2F3lMeXKx%2BVae3FIP29KalC2LuhiGGxiPgJnfwyV8efg1mhU3zZYuici2jKcRjlHy4vAk%2BpEh67rlIpLQkwmt%2BVoAY6sQH3enpFBbb6hQmSi%2BiBZNhaGMmamKs2eKfa8RYwGlGwUpC3OZCDesZkvtG0a%2BsazJvn9nRf%2BaWFNOvF%2Fs9oXDYOCb%2FTefMSydEd7q%2FGU0Bb96LvcglO3A5Qs812%2Fksy7VHliGFSjvKaOy0x3l3%2BFzC52637EGmayEJoW%2FpaR2gXOPwaOSXvG183vsOG%2BloR%2FArMxL4OAc0agtb74J48Hk1xIWGcGH%2F4lNvEnuluAQQFZh8%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230306T030000Z&X-Amz-SignedHeaders=host&X-Amz-Expires=216)

[Primitive Root - Algorithms for Competitive Programming (cp-algorithms.com)](https://cp-algorithms.com/algebra/primitive-root.html)

[tcp\_client.py](file:///C:\Users\juver\Downloads\tcp_client.py)

[tcp\_server.py](file:///C:\Users\juver\Downloads\tcp_server.py)

[RC4-KeyGeneration(1).py](file:///C:\Users\juver\Downloads\RC4-KeyGeneration(1).py)

[subprocess — Subprocess management — Python 3.11.2 documentation](https://docs.python.org/3/library/subprocess.html)

[Tcpdump Command in Linux | Linuxize](https://linuxize.com/post/tcpdump-command-in-linux/)