Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that ~~Mr.~~/Miss Niyati Savant

of Computer Department, Semester VI with

Roll No. 2103156 has completed a course of the necessary

experiments in the subject CSS under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In-Charge

Head of the Department

Date 08/04/24

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

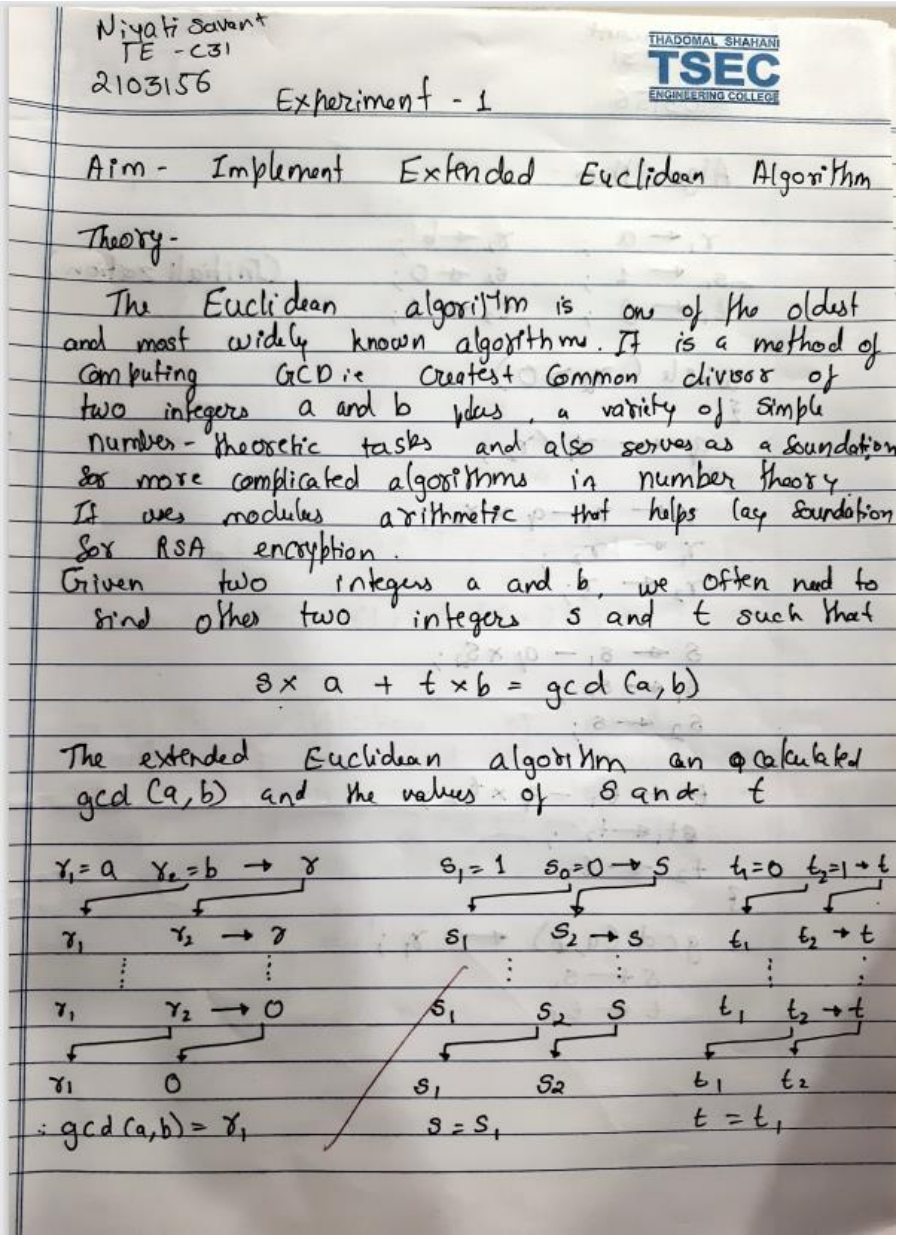
CONTENTS

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 1

Aim : Implementation of Extended Euclidean algorithm (LO1)

Description/Algorithm:



Niyati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment - 1

Aim - Implement Extended Euclidean Algorithm

Theory -

The Euclidean algorithm is one of the oldest and most widely known algorithms. It is a method of computing GCD i.e Greatest Common divisor of two integers a and b ydes, a variety of simple number - theoretic tasks and also serves as a foundation for more complicated algorithms in number theory It uses modules arithmetic that helps lay foundation for RSA encryption.

Given two integers a and b, we often need to find other two integers s and t such that

$$s \times a + t \times b = \gcd(a, b)$$

The extended Euclidean algorithm an calculated gcd (a, b) and the values of s and t

$r_1 = a \quad r_2 = b \rightarrow r$ $\quad s_1 = 1 \quad s_0 = 0 \rightarrow s$ $\quad t_1 = 0 \quad t_2 = 1 \rightarrow t$

$r_1 \quad r_2 \rightarrow r$ $\quad s_1 \quad s_2 \rightarrow s$ $\quad t_1 \quad t_2 \rightarrow t$

$r_1 \quad r_2 \rightarrow 0$ $\quad s_1 \quad s_2 \quad s$ $\quad t_1 \quad t_2 \rightarrow t$

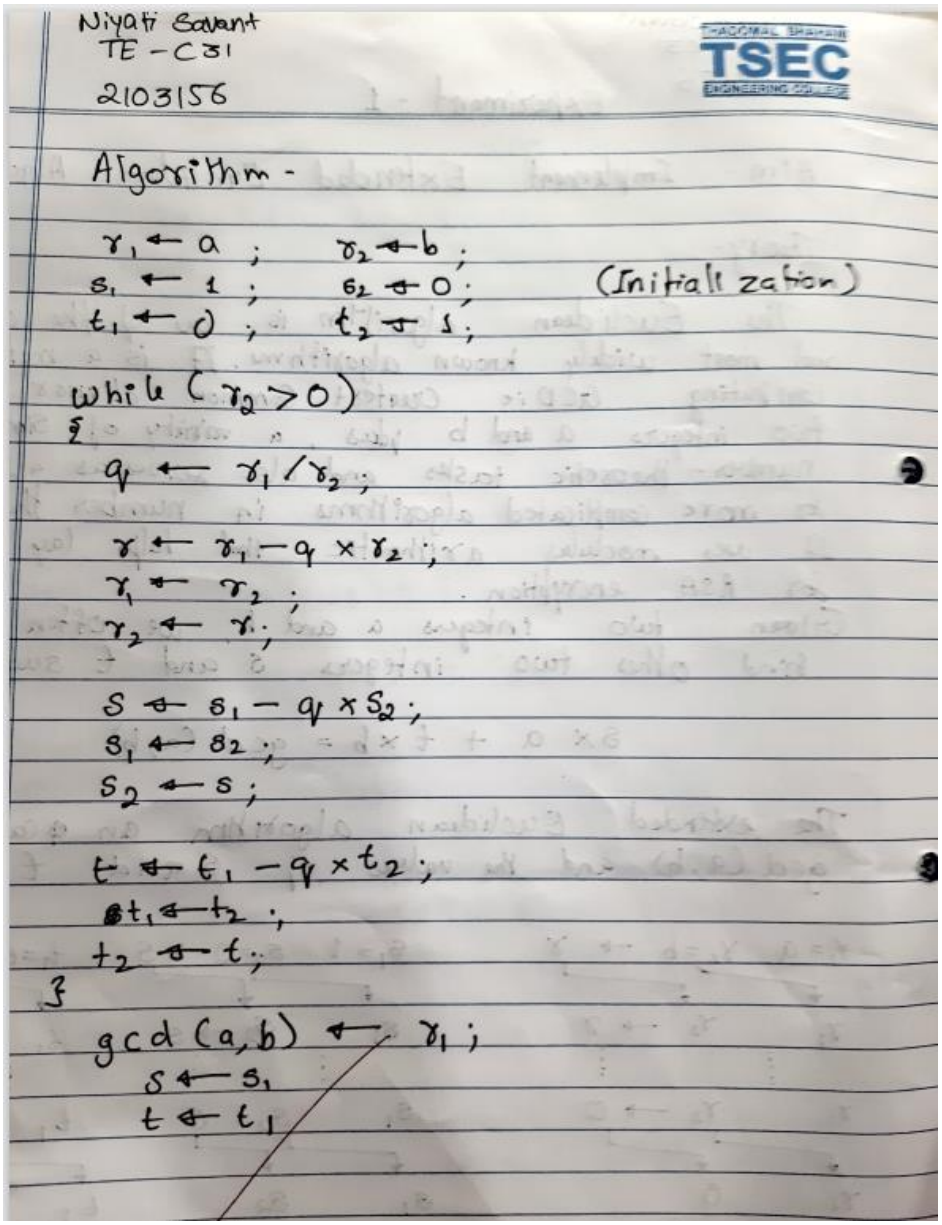
$r_1 \quad 0$ $\quad s_1 \quad s_2$ $\quad t_1 \quad t_2$

$\therefore \gcd(a, b) = r_1$ $\quad s = s_1$ $\quad t = t_1$

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Niyati Savant
TE - C31
2103156

Algorithm -

$r_1 \leftarrow a$; $r_2 \leftarrow b$;
$s_1 \leftarrow 1$; $s_2 \leftarrow 0$; (Initialization)
$t_1 \leftarrow 0$; $t_2 \leftarrow 1$;

while ($r_2 > 0$)
{
$q \leftarrow r_1 / r_2$;
$r \leftarrow r_1 - q \times r_2$;
$r_1 \leftarrow r_2$;
$r_2 \leftarrow r$;
$s \leftarrow s_1 - q \times s_2$;
$s_1 \leftarrow s_2$;
$s_2 \leftarrow s$;
$t \leftarrow t_1 - q \times t_2$;
$t_1 \leftarrow t_2$;
$t_2 \leftarrow t$;
}
gcd (a, b) $\leftarrow r_1$;
$s \leftarrow s_1$
$t \leftarrow t_1$

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

Niyati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Example

a = 161 , b = 28

Q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
	1								
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

$\therefore$ gcd (161, 28) = 7 , s = -1 , t = 6

and (-1 × 161) + (6 × 28) = 7

One of the applications of extended Euclidean algorithm is to find the multiplicative inverse of b in Z_n where n and b are given and gcd (n, b) = 1. It is the value of 't' after being mapped to Z_n

Example the multiplicative inverse of 11 in Z_{26}

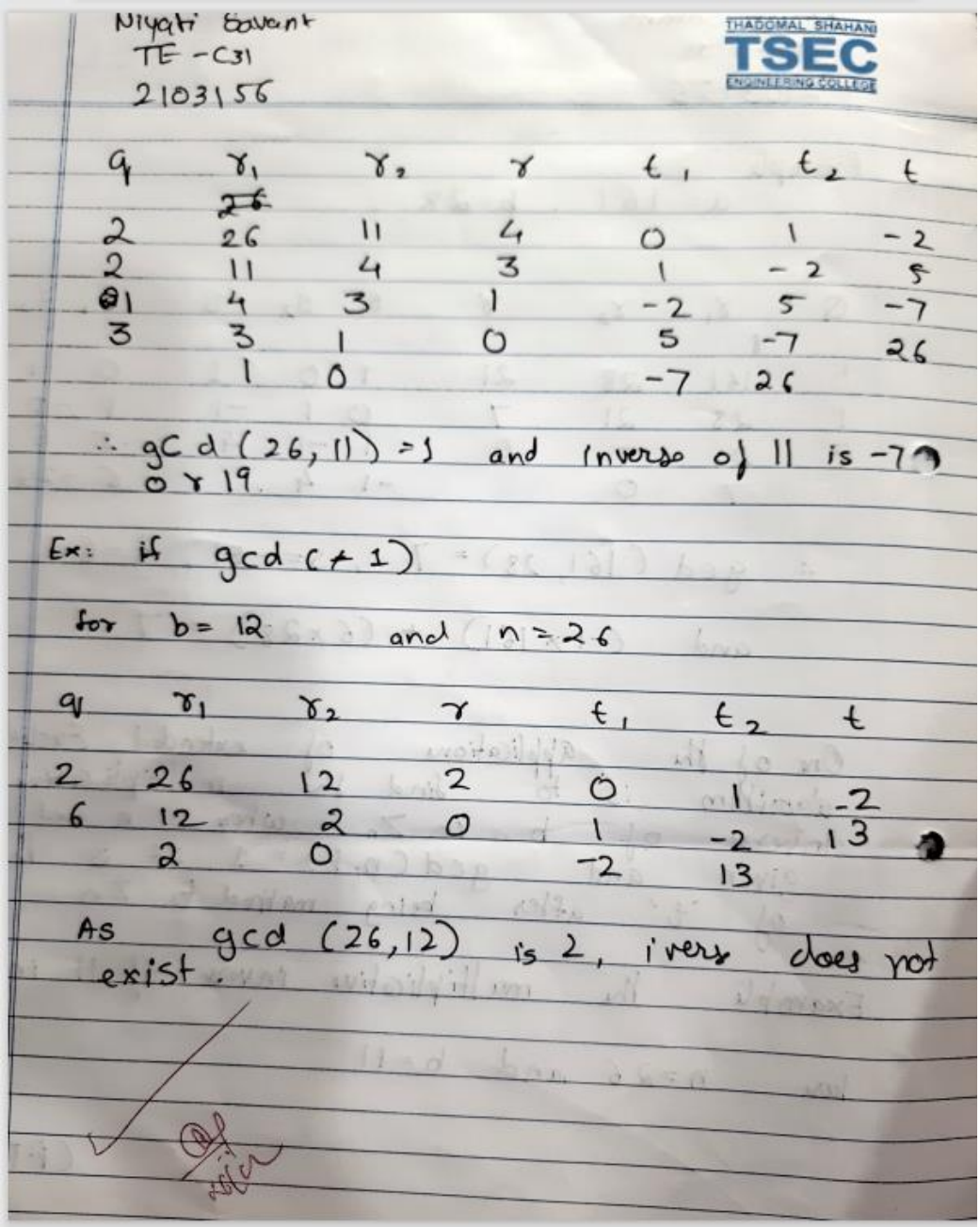
here n = 26 and b = 11

(PTO)

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Niyati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

q	r_1	r_2	r	t_1	t_2	t
	~~26~~					
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26
	1	0		-7	26	

∴ gcd (26, 11) = 1 and inverse of 11 is -7 or 19.

Ex: if gcd (≠ 1)

for b = 12 and n = 26

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	-2
6	12	2	0	1	-2	13
	2	0		-2	13	

As gcd (26,12) is 2, inverse does not exist.

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Implementation /Code :

```
def Extend_Euclidean(a,b):
    r1 = a
    r2 =b
    s1=t2=1
    s2=t1=0
    print("Q \t r1 \t r2 \t r \t s1 \t s2 \t s \t t1 \t t2 \t t")

    while(r2>0):
        quotient = int(r1/r2)

        remainder = r1%r2
        print(f"{quotient} \t {r1} \t {r2} \t {remainder} ", end=" ")
        r1 = r2
        r2 = remainder

        s = s1 -(quotient*s2)
        print(f"\t {s1} \t {s2} \t {s} ", end=" ")
        s1=s2
        s2=s

        t= t1-(quotient*t2)
        print(f"\t {t1} \t {t2} \t {t} \t ")
        t1=t2
        t2=t
    print(f"GCD = {r1} , s = {s1} and t = {t1} and (s*a)+(t*b)= GCD(a,b)")
    print(f"i.e ({s1}*{a})+({t1}*{b})={(s1*a)+(t1*b)}")

print("Niyati's Code for Extended Euclidean Algorithm")
print("Enter Two numbers whose GCD is to be found")
n1 = int(input("Enter the 1st Number: "))
n2 = int(input("Enter the 2nd Number: "))

if n1>=n2:
    Extend_Euclidean(n1, n2)
else :
    Extend_Euclidean(n2,n1)
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Result/Screenshots:

Niyati's Code for Extended Euclidean Algorithm
Enter Two numbers whose GCD is to be found
Enter the 1st Number: 119
Enter the 2nd Number: 68

Q	r1	r2	r	s1	s2	s	t1	t2	t
1	119	68	51	1	0	1	0	1	-1
1	68	51	17	0	1	-1	1	-1	2
3	51	17	0	1	-1	4	-1	2	-7

GCD = 17 , s = -1 and t = 2 and (s*a)+(t*b)= GCD(a,b)
i.e (-1*119)+(2*68)=17

Niyati's Code for Extended Euclidean Algorithm
Enter Two numbers whose GCD is to be found
Enter the 1st Number: 28
Enter the 2nd Number: 161

Q	r1	r2	r	s1	s2	s	t1	t2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23

GCD = 7 s = -1 and t = 6 and (s*a)+(t*b)= GCD(a,b)
i.e (-1*161)+(6*28)=7

Roll No : 2103156

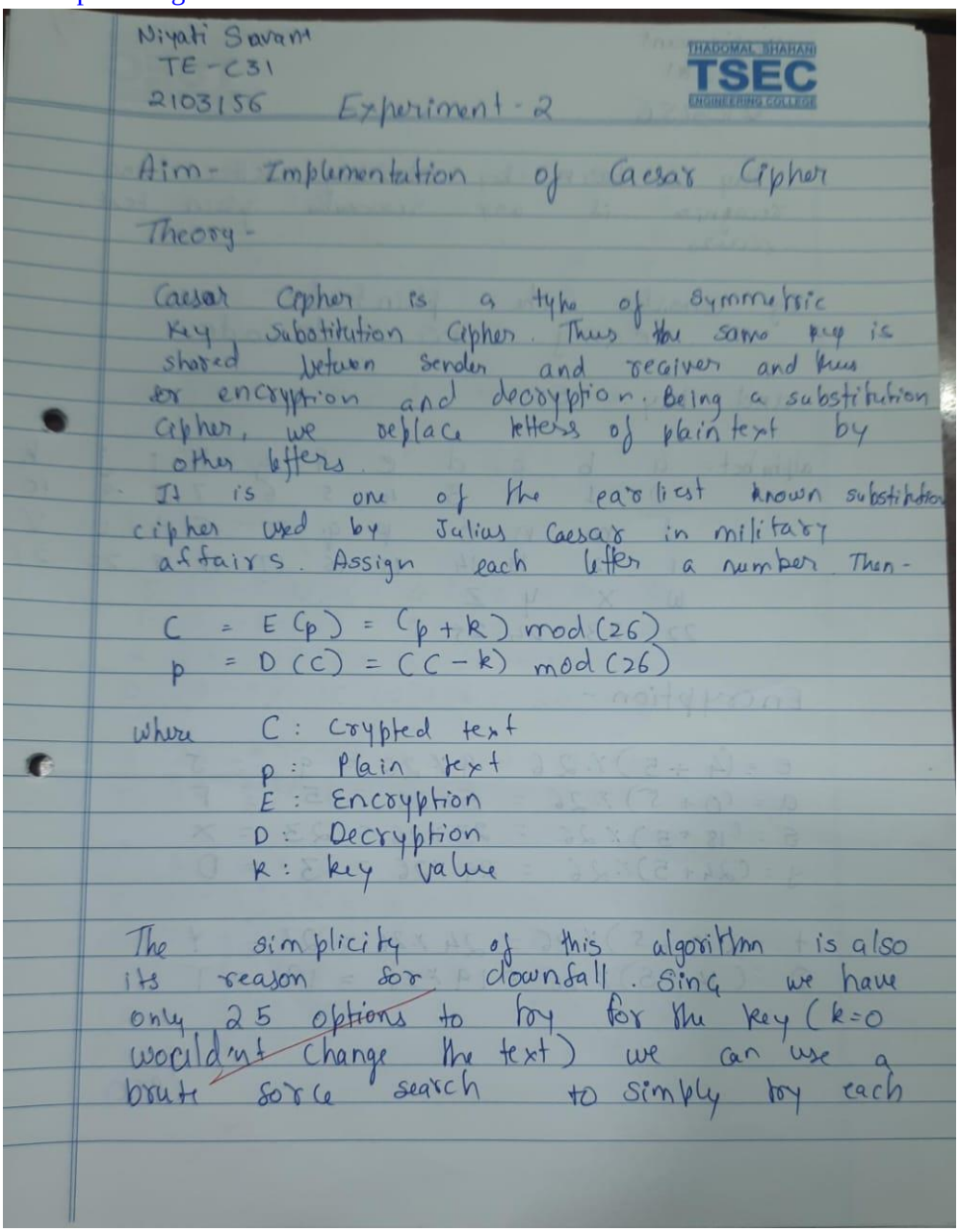
Batch : C31

Name : NIYATI SAVANT

Experiment 2

Aim : Implement Caeser Cipher (LO1)

Description/Algorithm:



Niyati Savant
TE-C31
2103156 Experiment-2

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Aim- Implementation of Caesar Cipher

Theory-

Caesar Cipher is a type of Symmetric Key substitution Cipher. Thus the same key is shared between sender and receiver and thus for encryption and decryption. Being a substitution Cipher, we replace letters of plain text by other letters.

It is one of the earliest known substitution cipher used by Julius Caesar in military affairs. Assign each letter a number. Then-

$C = E(p) = (p + k) \bmod (26)$

$p = D(C) = (C - k) \bmod (26)$

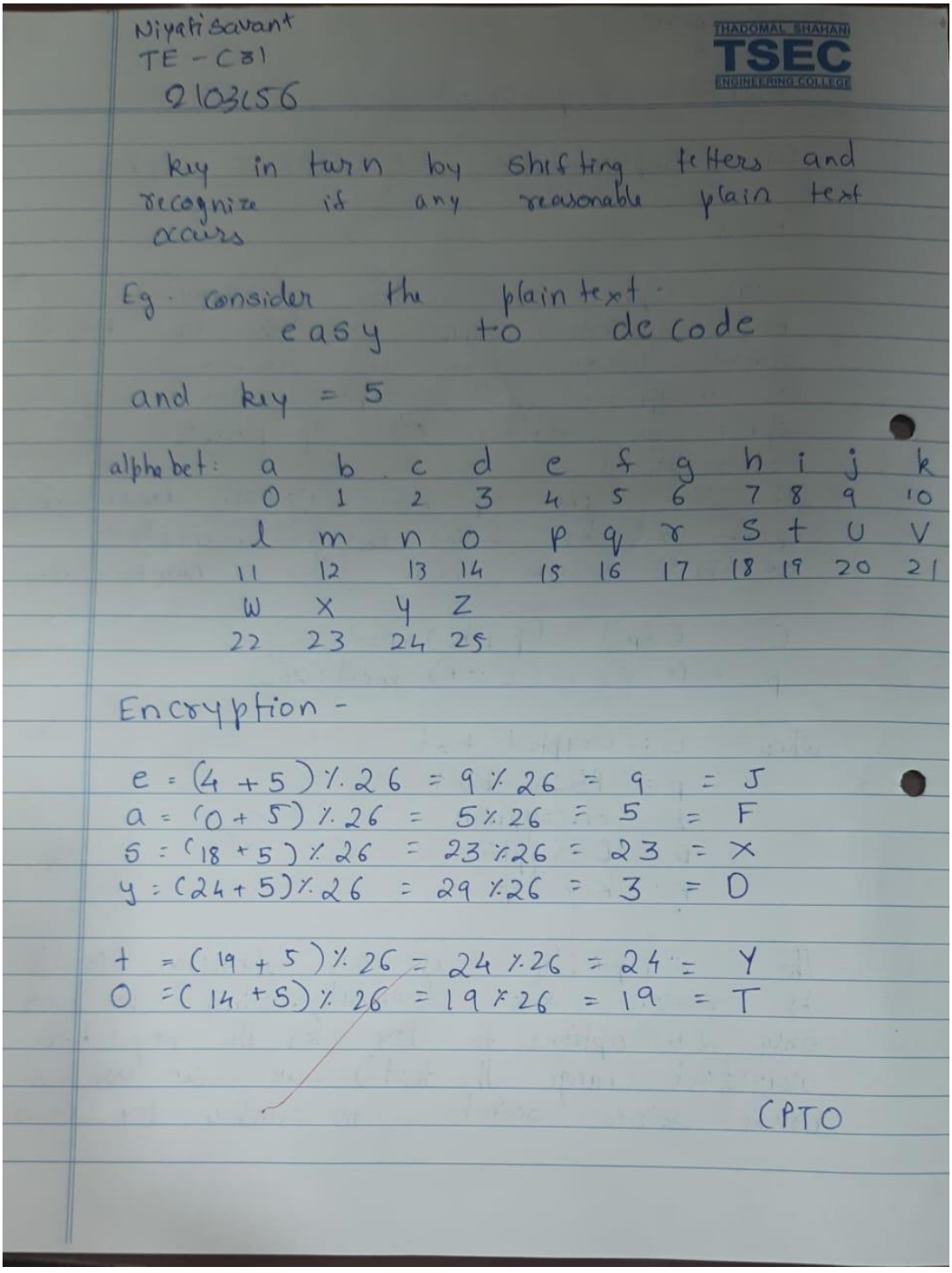
where C : Crypted text
p : Plain text
E : Encryption
D : Decryption
K : key value

The simplicity of this algorithm is also its reason for downfall. Sinc we have only 25 options to try for the key (k=0 wouldn't change the text) we can use a brute force search to simply try each

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Niyati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

key in turn by shifting letters and recognize if any reasonable plain text occurs

Eg. consider the plain text
easy to decode

and key = 5

alphabet:

a	b	c	d	e	f	g	h	i	j	k
0	1	2	3	4	5	6	7	8	9	10
l	m	n	o	p	q	r	s	t	u	v
11	12	13	14	15	16	17	18	19	20	21
w	x	y	z							
22	23	24	25							

Encryption -

e = (4 + 5) % 26 = 9 % 26 = 9 = J
a = (0 + 5) % 26 = 5 % 26 = 5 = F
s = (18 + 5) % 26 = 23 % 26 = 23 = X
y = (24 + 5) % 26 = 29 % 26 = 3 = D

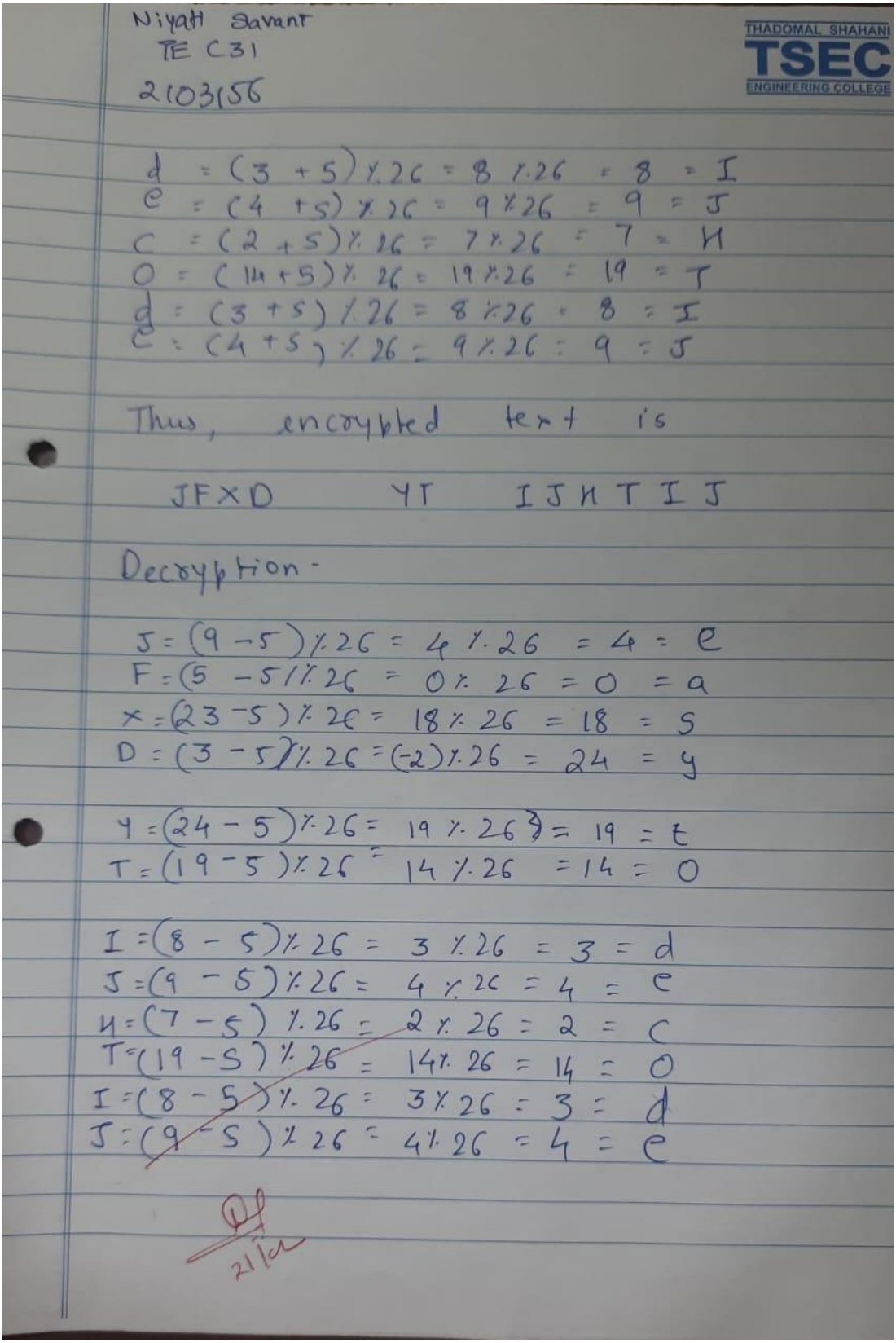
t = (19 + 5) % 26 = 24 % 26 = 24 = Y
o = (14 + 5) % 26 = 19 % 26 = 19 = T

CPTO

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Niyati Savant
TE C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

d = (3 + 5) % 26 = 8 % 26 = 8 = I
e = (4 + 5) % 26 = 9 % 26 = 9 = J
c = (2 + 5) % 26 = 7 % 26 = 7 = H
o = (14 + 5) % 26 = 19 % 26 = 19 = T
d = (3 + 5) % 26 = 8 % 26 = 8 = I
e = (4 + 5) % 26 = 9 % 26 = 9 = J

Thus, encrypted text is

JFXD YT IJHTIJ

Decryption -

J = (9 - 5) % 26 = 4 % 26 = 4 = e
F = (5 - 5) % 26 = 0 % 26 = 0 = a
x = (23 - 5) % 26 = 18 % 26 = 18 = s
D = (3 - 5) % 26 = (-2) % 26 = 24 = y

Y = (24 - 5) % 26 = 19 % 26 = 19 = t
T = (19 - 5) % 26 = 14 % 26 = 14 = o

I = (8 - 5) % 26 = 3 % 26 = 3 = d
J = (9 - 5) % 26 = 4 % 26 = 4 = e
H = (7 - 5) % 26 = 2 % 26 = 2 = c
T = (19 - 5) % 26 = 14 % 26 = 14 = o
I = (8 - 5) % 26 = 3 % 26 = 3 = d
J = (9 - 5) % 26 = 4 % 26 = 4 = e

21/2

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Implementation /Code :

```
lst_alphabet = ['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']
def Caeser_Cipher_encrypt(plaintxt, key):
    print("Encryption Process")
    key = key % 26
    cipher_txt = ""
    for letter in plaintxt:
        if letter in lst_alphabet:
            p = lst_alphabet.index(letter)


            val = (p + key) % 26
            E_p = (lst_alphabet[val]).upper()
            print(f"{letter} = ({p}+{key})%26 = {val} = {E_p}")
            cipher_txt += E_p
        else:
            print(" ")
            cipher_txt += " "
    print(f"The Encrypted text is: {cipher_txt}")



def Caeser_Cipher_decrypt(ciphertxt, key):
    print("Decryption Process")
    key = key % 26
    plain_txt = ""
    ciphertxt = ciphertxt.lower()
    for letter in ciphertxt:
        if letter in lst_alphabet:
            c = lst_alphabet.index(letter)
```

```
            val = (c - key) % 26
            D_c = (lst_alphabet[val])
            print(f"{letter.upper()} = ({c}-{key})%26 = {val} = {D_c}")
            plain_txt += D_c
        else:
            print(" ")
            plain_txt += " "
    print(f"The Decrypted text is: {plain_txt}")


print("Niyati's code for Caeser Cipher - ")
print("Enter 1 for encryption,2 for decryption")
user_choice = input("Enter choice: ")

txt = input("Enter Plain text: ")
k = int(input("Enter key Value: "))

if user_choice== '1':
    txt = txt.lower()
    Caeser_Cipher_encrypt(txt,k)

elif user_choice=='2':
    txt = txt.upper()
    Caeser_Cipher_decrypt(txt,k)
```

Result/Screenshots:

```
Niyati's code for Caeser Cipher -
Enter 1 for encryption,2 for decryption
Enter choice: 1
Enter text: easy to decode
Enter key Value: 5
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Encryption Process
e = (4+5)%26 = 9 = J
a = (0+5)%26 = 5 = F
s = (18+5)%26 = 23 = X
y = (24+5)%26 = 3 = D

t = (19+5)%26 = 24 = Y
o = (14+5)%26 = 19 = T

d = (3+5)%26 = 8 = I
e = (4+5)%26 = 9 = J
c = (2+5)%26 = 7 = H
o = (14+5)%26 = 19 = T
d = (3+5)%26 = 8 = I
e = (4+5)%26 = 9 = J
The Encrypted text is: JFXD YT IJHTIJ

Niyati's code for Caeser Cipher -
Enter 1 for encryption,2 for decryption
Enter choice: 2
Enter text: FYYFHP YTIFD
Enter key Value: 5
Decryption Process
F = (5-5)%26 = 0 = a
Y = (24-5)%26 = 19 = t
Y = (24-5)%26 = 19 = t
F = (5-5)%26 = 0 = a
H = (7-5)%26 = 2 = c
P = (15-5)%26 = 10 = k

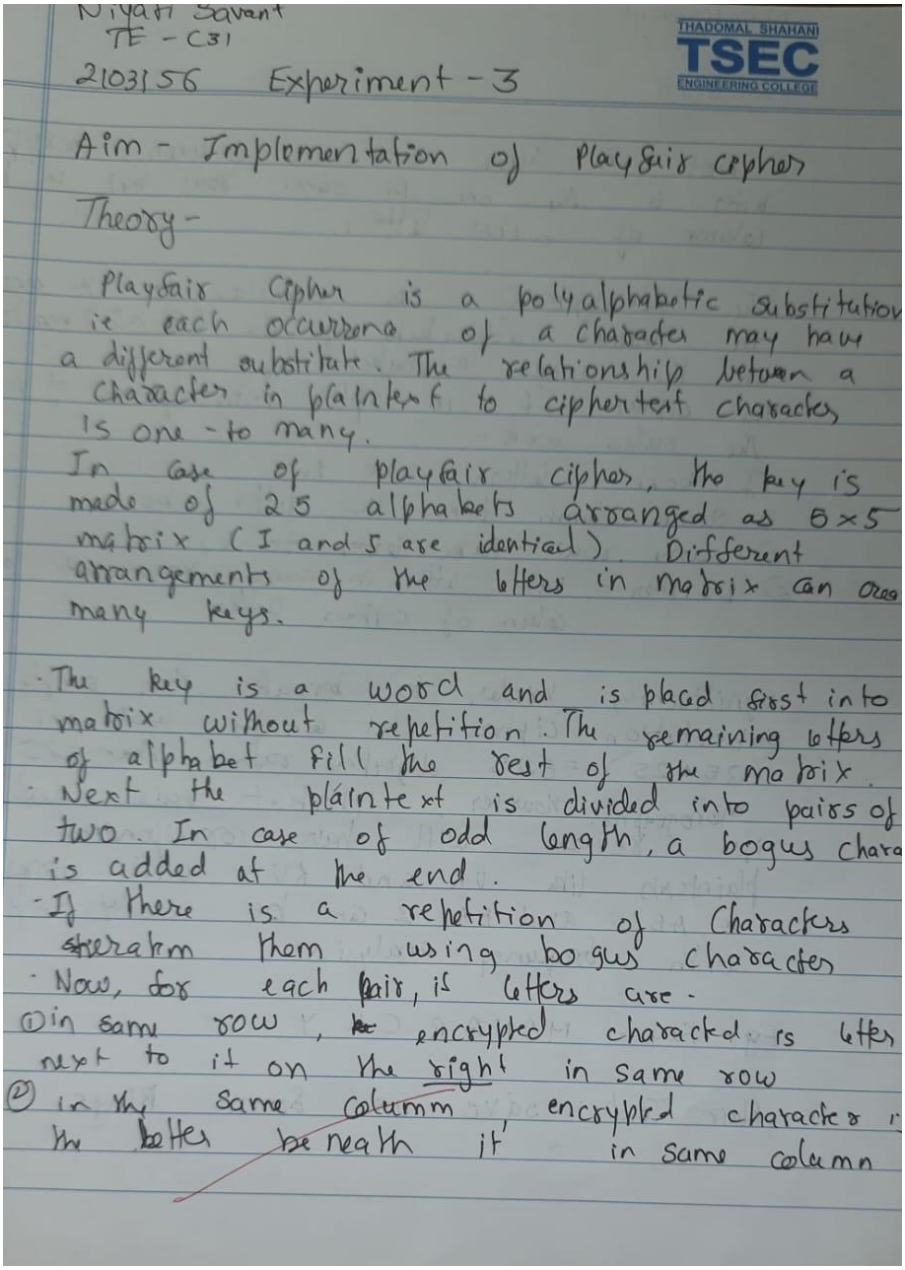
Y = (24-5)%26 = 19 = t
T = (19-5)%26 = 14 = o
I = (8-5)%26 = 3 = d
F = (5-5)%26 = 0 = a
D = (3-5)%26 = 24 = y
The Decrypted text is: attack today

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 3

Aim : Implementation of Playfair Cipher

Description/Algorithm:



Niyati Savant
TE - C31
2103156 Experiment - 3

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Aim - Implementation of Playfair cipher

Theory -

Playfair Cipher is a polyalphabetic substitution ie each occurrence of a character may have a different substitute. The relationship between a character in plaintext to ciphertext characters is one - to many.
In case of playfair cipher, the key is made of 25 alphabets arranged as 5×5 matrix (I and J are identical). Different arrangements of the letters in matrix can create many keys.

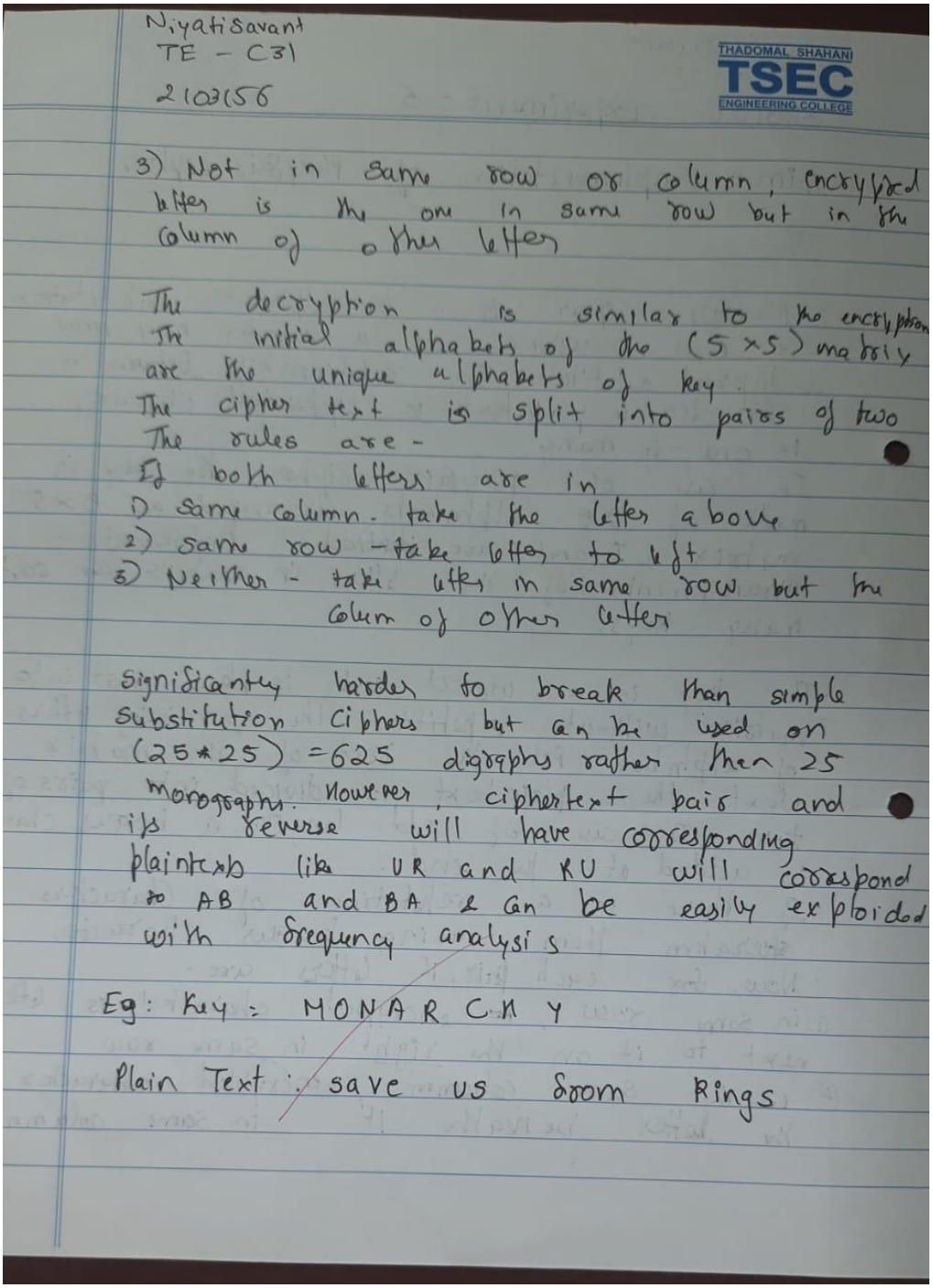
- The key is a word and is placed first into matrix without repetition. The remaining letters of alphabet fill the rest of the matrix.
- Next the plaintext is divided into pairs of two. In case of odd length, a bogus character is added at the end.
- If there is a repetition of characters then them using bogus character
- Now, for each pair, if letters are -

① in same row, the encrypted character is letter next to it on the right in same row
② in the same column encrypted character is the letter beneath it in same column

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Niyati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

3) Not in same row or column, encrypted letter is the one in same row but in the column of other letter

The decryption is similar to the encryption. The initial alphabets of the (5 x 5) matrix are the unique alphabets of key.
The cipher text is split into pairs of two
The rules are -
If both letters are in
1) same column. take the letter above
2) same row - take letter to left
3) Neither - take letter in same row but the colum of other letter

significantly harder to break than simple substitution ciphers but can be used on (25*25) = 625 digraphs rather than 25 monographs. However, ciphertext pair and its reverse will have corresponding plaintext like UR and RU will correspond to AB and BA & can be easily exploided with frequency analysis

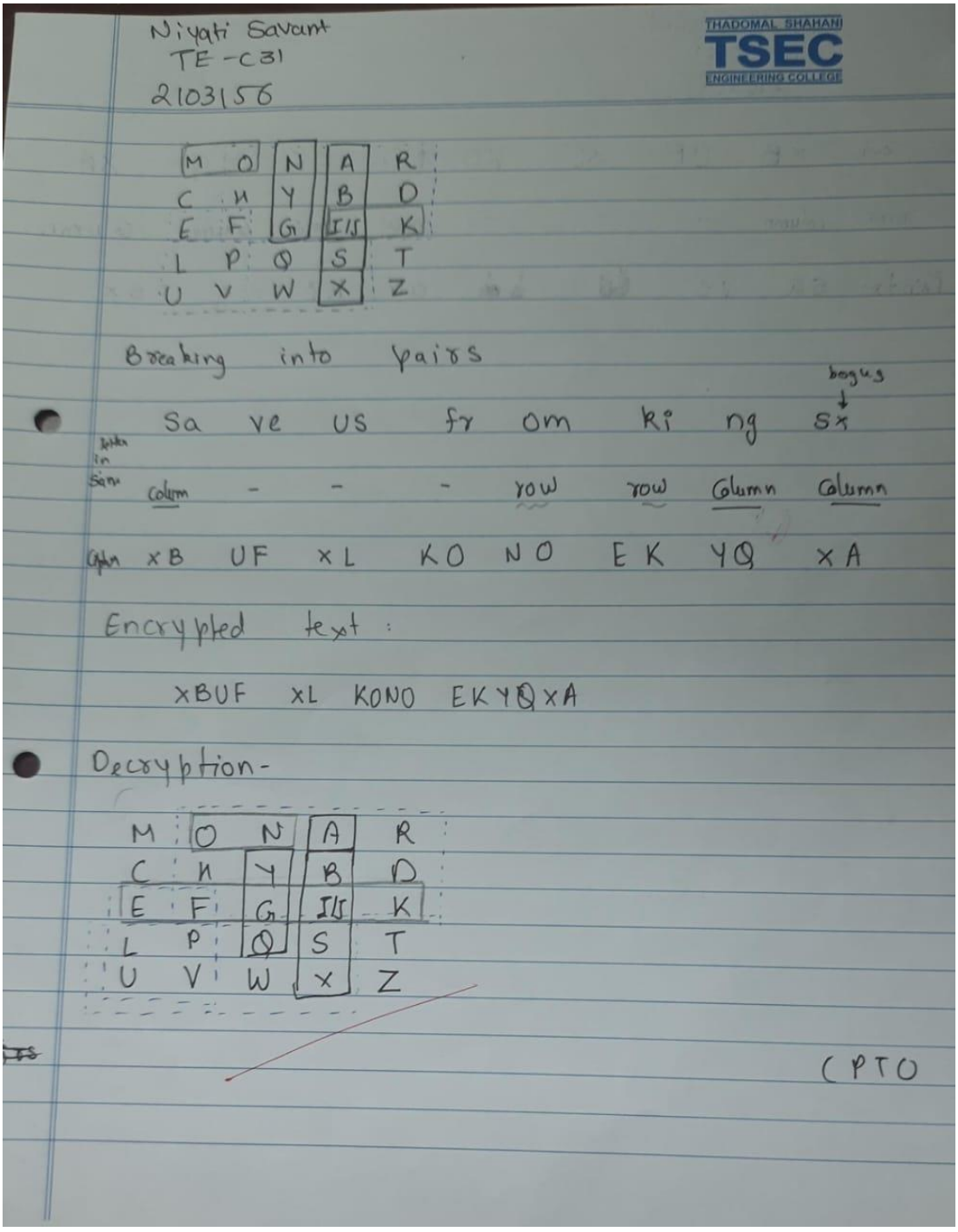
Eg: Key : MONARCHY

Plain Text : save us from Rings

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Niyati Savant
TE-C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Breaking into pairs

bogus

Sa ve us fr om ki ng sx

Column - - - row row Column Column

Cipher XB UF XL KO NO EK YQ XA

Encrypted text :

XBUF XL KONO EKYQXA

Decryption-

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

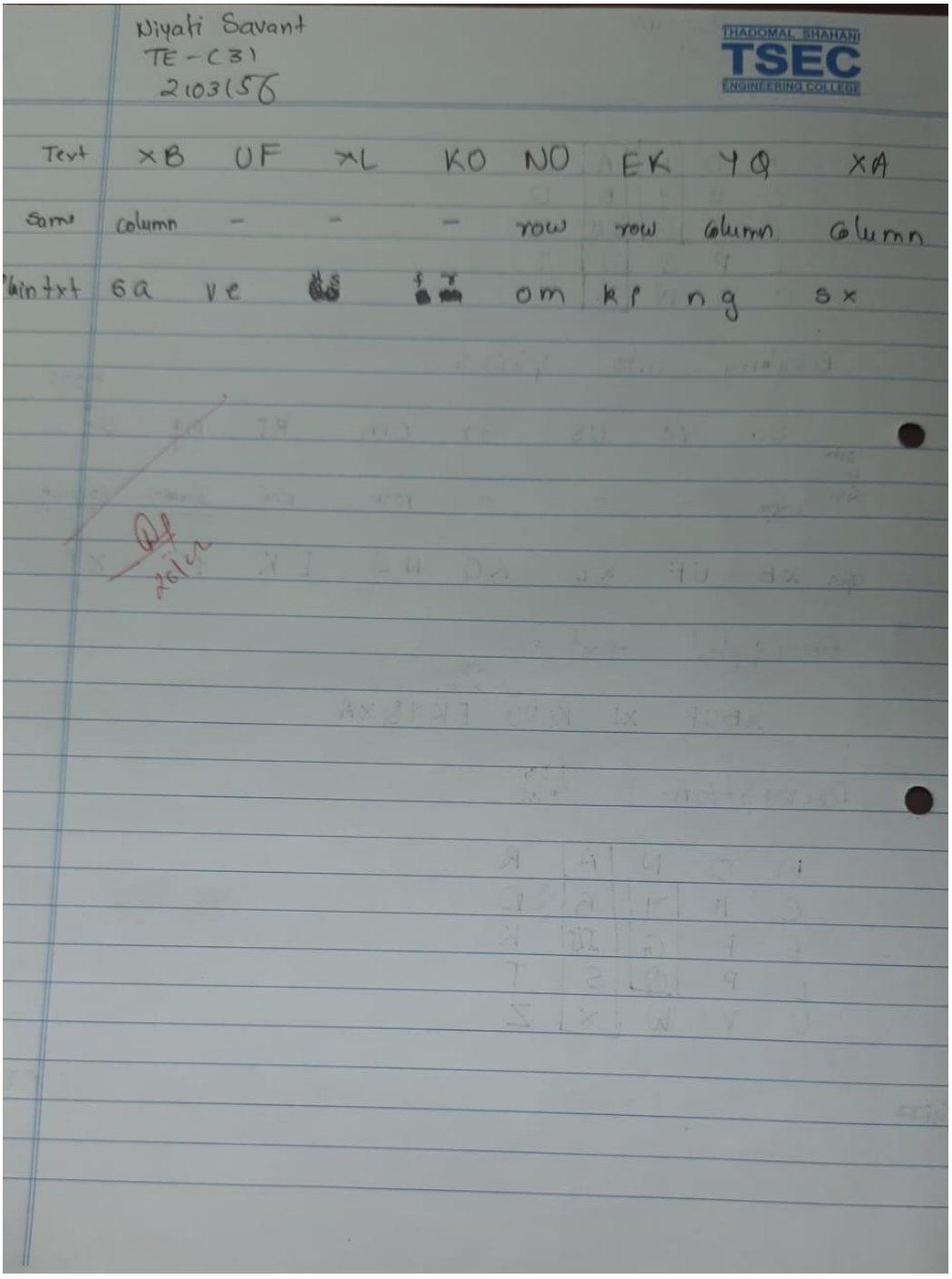
~~I/J~~

(PTO

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Implementation /Code :

```
print("Niyati's Code for Playfair Cipher")
key = input("Enter key: ")
key = key.replace(" ", "")
key = key.upper()


def matrix(x, y, initial):
    return [[initial for i in range(x)] for j in range(y)]


result = list()
for c in key:  # storing key
    if c not in result:
        if c == 'J':
            result.append('I')
        else:
            result.append(c)

flag = 0
for i in range(65, 91):  # storing other character
    if chr(i) not in result:
        if i == 73 and chr(74) not in result:
            result.append("I")
            flag = 1
        elif flag == 0 and i == 73 or i == 74:
            pass
        else:
            result.append(chr(i))

k = 0
my_matrix = matrix(5, 5, 0)  # initialize matrix
for i in range(0, 5):  # making matrix
    for j in range(0, 5):
        my_matrix[i][j] = result[k]
        k += 1
print("Matrix is: ")
for row in my_matrix:
    print(row)


def locindex(c):  # get location of each character
```

```
    loc = list()
    if c == 'J':
        c = 'I'
    for i, j in enumerate(my_matrix):
        for k, l in enumerate(j):
            if c == l:
                loc.append(i)
                loc.append(k)
                return loc


def encrypt():
    msg = str(input("Enter plain text: "))
    msg = msg.upper()
    msg = msg.replace(" ", "")
    # add bogus char if needed
    for s in range(0, len(msg) + 1, 2):
        if s < len(msg) - 1:
            if msg[s] == msg[s + 1]:
                msg = msg[:s + 1] + 'X' + msg[s + 1:]
    if len(msg) % 2 != 0:
        msg = msg[:] + 'X'

    print("CIPHER TEXT:", end=' ')
    i = 0
    while i < len(msg):
        loc = locindex(msg[i])
        loc1 = locindex(msg[i + 1])
        if loc[1] == loc1[1]:
            print("{}{}".format(my_matrix[(loc[0] + 1) % 5][loc[1]],
                                my_matrix[(loc1[0] + 1) % 5][loc1[1]]), end=' ')
        elif loc[0] == loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1] + 1) % 5],
                                my_matrix[loc1[0]][(loc1[1] + 1) % 5]), end=' ')
        else:
            print("{}{}".format(my_matrix[loc[0]][loc1[1]],
                                my_matrix[loc1[0]][loc[1]]), end=' ')
        i += 2


def decrypt():  # decryption
    msg = str(input("Enter Cipher text: "))
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

```
    msg = msg.upper()
    msg = msg.replace(" ", "")
    print("PLAIN TEXT:", end=' ')
    i = 0
    while i < len(msg):
        loc = locindex(msg[i])
        loc1 = locindex(msg[i + 1])
        if loc[1] == loc1[1]:
            print("{}{}".format(my_matrix[(loc[0] - 1) % 5][loc[1]],
                        my_matrix[(loc1[0] - 1) % 5][loc1[1]]), end=' ')
        elif loc[0] == loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1] - 1) % 5],
                        my_matrix[loc1[0]][(loc1[1] - 1) % 5]), end=' ')
        else:
            print("{}{}".format(my_matrix[loc[0]][loc1[1]],
                        my_matrix[loc1[0]][loc[1]]), end=' ')
        i += 2


choice = int(input("\nEnter  1 for Encryption and  2 for Decryption: "))
if choice == 1:
    encrypt()

elif choice == 2:
    decrypt()
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Result/Screenshots:

Niyati's Code for Playfair Cipher
Enter key: monarchy
Matrix is:
['M', 'O', 'N', 'A', 'R']
['C', 'H', 'Y', 'B', 'D']
['E', 'F', 'G', 'I', 'K']
['L', 'P', 'Q', 'S', 'T']
['U', 'V', 'W', 'X', 'Z']

Enter 1 for Encryption and 2 for Decryption: 1
Enter plain text: save me from evil kings
CIPHER TEXT: XB UF CL KO NO FU ES EK YQ XA

Niyati's Code for Playfair Cipher
Enter key: monarchy
Matrix is:
['M', 'O', 'N', 'A', 'R']
['C', 'H', 'Y', 'B', 'D']
['E', 'F', 'G', 'I', 'K']
['L', 'P', 'Q', 'S', 'T']
['U', 'V', 'W', 'X', 'Z']

Enter 1 for Encryption and 2 for Decryption: 2
Enter Cipher text: XB UF CL KO NO FU ES EK YQ XA
PLAIN TEXT: SA VE ME FR OM EV IL KI NG SX

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 4

Aim : Implementation of Euler's Totient Function

Description/Algorithm:

Euler's Totient function ? (n) for an input n is the count of numbers in {1, 2, 3, …, n-1} that are relatively prime to n, i.e., the numbers whose GCD (Greatest Common Divisor) with n is 1.
A simple solution is to iterate through all numbers from 1 to n-1 and count numbers with gcd with n as 1. Below is the implementation of the simple method to compute Euler's Totient function for an input integer n. The above code calls gcd function O(n) times. The time complexity of the gcd function is O(h) where "h" is the number of digits in a smaller number of given two numbers.

Some Interesting Properties of Euler's Totient Function

1) For a prime number p, phi(p) = p - 1
phi(5) = 5 - 1 = 4
phi(13) = 13 - 1 = 12
phi(29) = 29 - 1 = 28

2) For two prime numbers a and b phi(a * b) = phi(a) * phi(b) = (a - 1) * (b - 1)
phi(5 * 7) = phi(5) * phi(7) = (5 - 1) * (7 - 1) = 24
phi(3 * 5) = phi(3) * phi(5) = (3 - 1) * (5 - 1) = 8
phi(3 * 7) = phi(3) * phi(7) = (3 - 1) * (7 - 1) = 12

3) For a prime number p, phi(p ^ k) = p ^ k - p ^ {k - 1}
phi(2 ^ 5) = 2 ^ 5 - 2 ^ {5 - 1} = 32 - 16 = 16
phi(5 ^ 3) = 5 ^ 3 - 5 ^ {3 - 1} = 125 - 25 = 100
phi(3 ^ 5) = 3 ^ 5 - 3 ^ {5 - 1} = 243 - 81 = 162

4) For two number a and b phi(a * b) = phi(a) * phi(b) * frac {gcd(a, b)} {phi(gcd(a, b))}
Special Case : gcd(a, b) = 1
phi(a * b) = phi(a) * phi(b) * frac {1} {phi(1)} = phi(a) * phi(b)
phi(a * b) = phi(a) * phi(b)
phi(2 * 9) = phi(2) * phi(9) = 1 * 6 = 6p
hi(8 * 9) = phi(8) * phi(9) = 4 * 6 = 24
phi(5 * 6) = phi(5) * phi(6) = 4 * 2 = 8
Normal Case : gcd(a, b) is not 1
phi(a * b) = phi(a) * phi(b) * frac {gcd(a, b)} {phi(gcd(a, b))}
phi(4 * 6) = phi(4) * phi(6) * frac {gcd(4, 6)} {phi(gcd(4, 6))}
= 2 * 2 * frac{2}{1} = 2 * 2 * 2
= 8
phi(4 * 8) = phi(4) * phi(8) * frac {gcd(4, 8)} {phi(gcd(4, 8))}

= 2 * 4 * frac{4}{2} = 2 * 4 * 2
= 16

Implementation /Code :

```
def gcd(a,b):
  r1 = a
  r2 =b
  while(r2>0):
    quotient = int(r1/r2)
    remainder = r1%r2

    r1 = r2
    r2 = remainder
  return r1

print("Niyati's Code for Euler Totient function")
print("Enter number whose totient function is to be found")
n1 = int(input("Enter the 1st Number: "))

count=0
for i in range(1,n1):
  val = gcd(n1,i)
  print(f"GCD of {n1} and {i} is {val}")
  if (val == 1):
    count+=1
print(f"The phi({n1}) is {count}")
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Result/Screenshots:

```
Niyati's Code for Euler Totient function
Enter number whose totient function is to be found
Enter the 1st Number: 45
GCD of 45 and 1 is 1
GCD of 45 and 2 is 1
GCD of 45 and 3 is 3
GCD of 45 and 4 is 1
GCD of 45 and 5 is 5
GCD of 45 and 6 is 3
GCD of 45 and 7 is 1
GCD of 45 and 8 is 1
GCD of 45 and 9 is 9
GCD of 45 and 10 is 5
GCD of 45 and 11 is 1
GCD of 45 and 12 is 3
GCD of 45 and 13 is 1
GCD of 45 and 14 is 1
GCD of 45 and 15 is 15
GCD of 45 and 16 is 1
GCD of 45 and 17 is 1
GCD of 45 and 18 is 9
GCD of 45 and 19 is 1
GCD of 45 and 20 is 5
GCD of 45 and 21 is 3
GCD of 45 and 22 is 1
GCD of 45 and 23 is 1
GCD of 45 and 24 is 3
GCD of 45 and 25 is 5
GCD of 45 and 26 is 1
GCD of 45 and 27 is 9
GCD of 45 and 28 is 1
GCD of 45 and 29 is 1
GCD of 45 and 30 is 15
GCD of 45 and 31 is 1
GCD of 45 and 32 is 1
GCD of 45 and 33 is 3
GCD of 45 and 34 is 1
GCD of 45 and 35 is 5
GCD of 45 and 36 is 9
GCD of 45 and 37 is 1
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

GCD of 45 and 38 is 1
GCD of 45 and 39 is 3
GCD of 45 and 40 is 5
GCD of 45 and 41 is 1
GCD of 45 and 42 is 3
GCD of 45 and 43 is 1
GCD of 45 and 44 is 1
The phi(45) is 24

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 5

Aim : Function Implementation of RSA cryptosystem

Description/Algorithm:

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and the Private key is kept private.
An example of asymmetric cryptography:

- A client (for example browser) sends its public key to the server and requests some data.
- The server encrypts the data using the client's public key and sends the encrypted data.
- The client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.
The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future.
The **mechanism** behind the RSA algorithm :
Generating Public Key:
Select two prime no's. Suppose P = 53 and Q = 59.
Now First part of the Public key : n = P*Q = 3127.
We also need a small exponent say e :
But e Must be an integer, such that it is not a factor of phi(n) and 1 < e < phi(n)
Our Public Key is made of n and e
Generating Private Key:
We need to calculate phi(n) :
Such that phi(n) = (P-1)(Q-1)
so, phi(n) = 3016
Now calculate Private Key, d :
d = (k*phi(n) + 1) / e for some integer k
For k = 2, value of d is 2011.
Public Key (n = 3127 and e = 3) and Private Key(d = 2011)
If msg = 89
Encrypted Data c = (89 e)mod n
Thus our Encrypted Data comes out to be 1394
Now we will decrypt 1394 :

Decrypted Data = (cd)mod n
Thus our Encrypted Data comes out to be 89.

Advantages:

1. Security: RSA algorithm is considered to be very secure and is widely used for secure data transmission.
2. Public-key cryptography: RSA algorithm is a public-key cryptography algorithm, which means that it uses two different keys for encryption and decryption. The public key is used to encrypt the data, while the private key is used to decrypt the data.
3. Key exchange: RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network.
4. Digital signatures: RSA algorithm can be used for digital signatures, which means that a sender can sign a message using their private key, and the receiver can verify the signature using the sender's public key.
5. Speed: The RSA technique is suited for usage in real-time applications since it is quite quick and effective.
6. Widely used: Online banking, e-commerce, and secure communications are just a few fields and applications where the RSA algorithm is extensively developed.

Disadvantages:

1. Slow processing speed: RSA algorithm is slower than other encryption algorithms, especially when dealing with large amounts of data.
2. Large key size: RSA algorithm requires large key sizes to be secure, which means that it requires more computational resources and storage space.
3. Vulnerability to side-channel attacks: RSA algorithm is vulnerable to side-channel attacks, which means an attacker can use information leaked through side channels such as power consumption, electromagnetic radiation, and timing analysis to extract the private key.
4. Limited use in some applications: RSA algorithm is not suitable for some applications, such as those that require constant encryption and decryption of large amounts of data, due to its slow processing speed.
5. Complexity: The RSA algorithm is a sophisticated mathematical technique that some individuals may find challenging to comprehend and use.
6. Key Management: The secure administration of the private key is necessary for the RSA algorithm, although in some cases this can be difficult.
7. Vulnerability to Quantum Computing: Quantum computers have the ability to attack the RSA algorithm, potentially decrypting the data.

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Implementation /Code :

```
import math
def gcd(a, b):
    r1 = a
    r2 = b
    while r2 > 0:
        quotient = int(r1 / r2)
        remainder = r1 % r2

        r1 = r2
        r2 = remainder
    return r1

print("Niyati's code for RSA algorithm")
p = int(input("Enter The value of p: "))
q = int(input("Enter The value of q: "))
n = p * q
print(f"The value of n is {n}")
phi = (p - 1) * (q - 1)
print(f"The value of phi({n}) is {phi}")
e = int(input(f"Enter encryption key e such that e and phi({n}) are co prime and e < phi({n}): "))
print(f"The public key is ({e},{n})")
for k in range(15):
    d = (1 + (k * phi)) / e
    if (int(d) == d):
        d = int(d)
        break

print(f"The private key is ({d},{n})")

msg = int(input("Enter Message: "))

print("Encryption is C = (msg ^ e) mod n")
c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)

print("Decryption is M = (C ^ d) mod n")
m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Result/Screenshots:

```
Niyati's code for RSA algorithm
Enter The value of p: 7
Enter The value of q: 17
The value of n is 119
The value of phi(119) is 96
Enter encryption key e such that e and phi(119) are co prime and e < phi(119): 7
The public key is (7,119)
The private key is (55,119)
Enter Message: 2
Encryption is C = (msg ^ e) mod n
Encrypted data =  9.0
Decryption is M = (C ^ d) mod n
Original Message Sent =  2.0
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 6

Aim: Function Implementation of RSA cryptosystem

Description/Algorithm:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.
For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.
Step-by-Step explanation is as follows:

Alice	**Bob**
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a Key generated = R1 = (G^a) mod P	Private Key Selected = b Key generated = R2 = (G^b) mod P
Exchange of generated keys takes place	
Key received = R2	key received = R1
Generated Secret Key = Ka = (R2)^a mod P	Generated Secret Key = Kb = (R1)^b mod P
Algebraically, it can be shown that Ka=Kb=K Where K = (G^ab) mod P	
Users now have a symmetric secret key to encrypt	

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice: R1 =(9^4 mod 23) = (6561 mod 23) = 6
Bob: R2 = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key R2 =16 and
Bob receives public key R1 = 6

Step 6: Alice and Bob compute symmetric keys
Alice: Ka = R1^a mod p = 65536 mod 23 = 9
Bob: Kb = R2^b mod p = 216 mod 23 = 9
i.e K = 9^(12) mod 23 = 9

Step 7: 9 is the shared secret.

Implementation /Code :

```
import random
print("Niyati's code for Diffie Hellman ")
p = int(input("Enter P:"))
g = int(input("Enter G:"))
a = random.randint(1, 100)
print(f"Private key of Sender {a} ")

b = random.randint(1, 100)
print(f"Private key of Receiver {b}")
R1 = ((pow(g, a)) % p)
R2 = ((pow(g, b)) % p)
print(f"Sender receives key {R2} and receiver gets key {R1} ")
print("The symmetric Keys")
Ka = ((pow(R2, a)) % p)
Kb = ((pow(R1, b)) % p)
print(f"Secret key at A = {Ka}")
print(f"Secret key at B = {Kb}")
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Result/Screenshots:

```
Niyati's code for Diffie Hellman
Enter P:71
Enter G:7
Private key of Sender 77
Private key of Receiver 10
Sender receives key 45 and receiver gets key 14
The symmetric Keys
Secret key at A = 1
Secret key at B = 1
-------------------------------------------------------------------------------------------------------------------------------
Niyati's code for Diffie Hellman
Enter P:7
Enter G:5
Private key of Sender 55
Private key of Receiver 86
Sender receives key 4  and receiver gets key 5
The symmetric Keys
Secret key at A = 4
Secret key at B = 4
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 7

Aim: Study the use of network reconnaissance tools and apply the following: WHOIS, dig, traceroute, nslookup.

Description/Algorithm:

Steps:

1. Open ubuntu terminal.
2. Get root access by typing “sudo su root”. Put the pc password.
3. Install the tool using the following command

#apt-get install whois
#apt-get install dig
#apt-get install traceroute
#apt-get install nslookup

Reconnaissance is the unauthorized discovery and mapping of systems, services, or vulnerabilities. Reconnaissance is also known as information gathering and, in most cases, precedes an actual access or DoS attack. First, the malicious intruder typically conducts a ping sweep of the target network to determine which IP addresses are alive. Then the intruder determines which services or ports are active on the live IP addresses. From this information, the intruder queries the ports to determine the type and version of the application and operating system running on the target host. Reconnaissance is somewhat analogous to a thief investigating a neighborhood for vulnerable homes, such as an unoccupied residence or a house with an easy to-open door or window. In many cases, intruders look for vulnerable services that they can exploit later when less likelihood that anyone is looking exists.

WHOIS

WHOIS is the Linux utility for searching an object in a WHOIS database. The WHOIS database of a domain is the publicly displayed information about a domain's ownership, billing, technical, administrative, and nameserver information. Running a WHOIS on your domain will look the domain up at the registrar for the domain information. All domains have WHOIS information. WHOIS database can be queried to obtain the following information via WHOIS:

- Administrative contact details, including names, email addresses, and telephone numbers
- Mailing addresses for office locations relating to the target organization
- Details of authoritative name servers for each given domain

Example: Querying tsec. ed

Dig (domain information groper) is a network administration command-line tool for querying Domain Name System (DNS) name servers. Dig is useful for network troubleshooting and for educational purposes.

When you pass a domain name to the dig command, by default it displays the A record

(the ip-address of the site that is queried) .
The dig command output has the following sections:

Header: This displays the dig command version number, the global options used by the dig command, and few additional header information.

QUESTION SECTION: This displays the question it asked the DNS. i.e. input. Since we said 'dig google.com', it indicates in this section that we asked for the record of the google.com website.

ANSWER SECTION: This displays the answer it receives from the DNS. i.e This is your output. This displays the record of google.com.

AUTHORITY SECTION: This displays the DNS name server that has the authority to re spond to this query. Basically this displays available name servers of google.com.

ADDITIONAL SECTION: This displays the ip address of the name servers listed in the AUTHORITY SECTION.
Stats section at the bottom displays few dig command statistics including how much time it took to execute this query

Query MX Records Using dig MX
To query MX records, pass MX as an argument to the dig command.
student@lab:~ #dig google.com MX +noall +answer
Query NS Records Using dig NS
To query the NS record use the type NS .
student@lab:~ #dig google.com NS +noall +answer
View ALL DNS Records Types Using dig -t ANY
To view all the record types (A, MX, NS, etc.), use ANY as the record type as shown be - low.
student@lab:~ #dig -t ANY google.com +noall +answer
View Short Output Using dig +short
To view just the ip-address of a web site (i.e the A record), use the short form option as shown below.
student@lab:~ #dig google.com +short
DNS Reverse Look-up Using dig –x
To perform a DNS reverse look up using the ip address using dig -x
student@lab:~ #dig -x 209.132.183.81

Traceroute
Traceroute prints the route that packets take to a network host. Traceroute utility uses the TTL field in the IP header to achieve its operation. TTL field describes how much hops

a particular packet will take while traveling on network. So, this effectively outlines the lifetime of the packet on network. This field is usually set to 32 or 64. Each time the packet is held on an intermediate router, it decreases the TTL value by 1. When a router finds the TTL value of 1 in a received packet then that packet is not forwarded but instead discarded. After discarding the packet, router sends an ICMP error message of ―Time exceeded back to the source from where ‖ packet generated. The ICMP packet that is sent back contains the IP address of the router. So now it can be easily understood that traceroute operates by sending packets with TTL value starting from 1 and then incrementing by one each time. Each time a router receives the packet, it checks the TTL field, if TTL field is 1 then it discards the packet and sends the ICMP error packet containing its IP address and this is what traceroute requires. So traceroute incrementally fetches the IP of all the routers between the source and the destination.

nslookup
The nslookup command is used to query internet name servers interactively for information. Nslookup, which stands for "name server lookup”. It is a useful tool for finding out information about a named domain. By default, nslookup will translate a domain name to an IP address (or vice versa). Nslookup has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.
1. Simple nslookup command
student@lab:~ #nslookup google.com

2. Query the MX Record using -query=mx
student@lab:~ #nslookup -query = mx google.com
MX (Mail Exchange) record maps a domain name to a list of mail exchange servers for that domain

3. Query the NS Record using -type=ns
student@lab: ~ #nslookup -type = ns google.com
NS (Name Server) record maps a domain name to a list of DNS servers authoritative for that domain.

4. Query the SOA Record using -type=soa
student@lab: ~ #nslookup -type = soa google.com
SOA record (start of authority) provides the authoritative information about the domain, the e-mail address of the domain admin, the domain serial number, etc

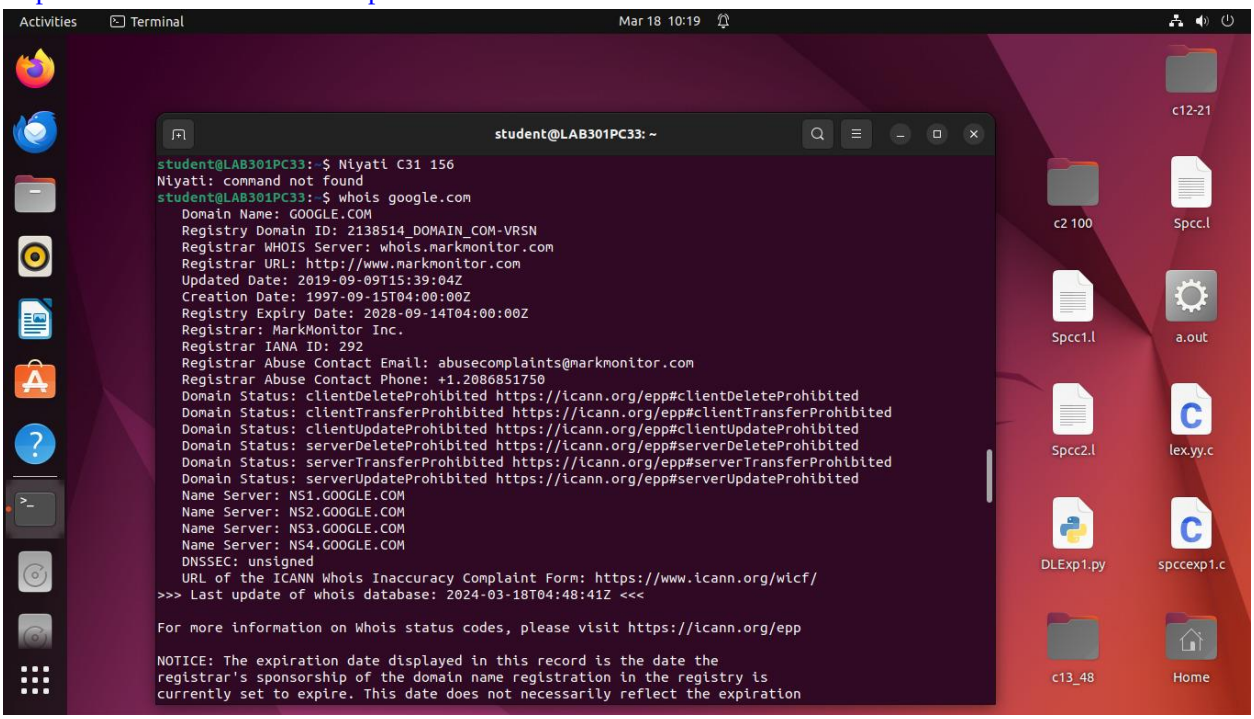
5. View available DNS records using -query=any
student@lab: ~ #nslookup -type = any google.co

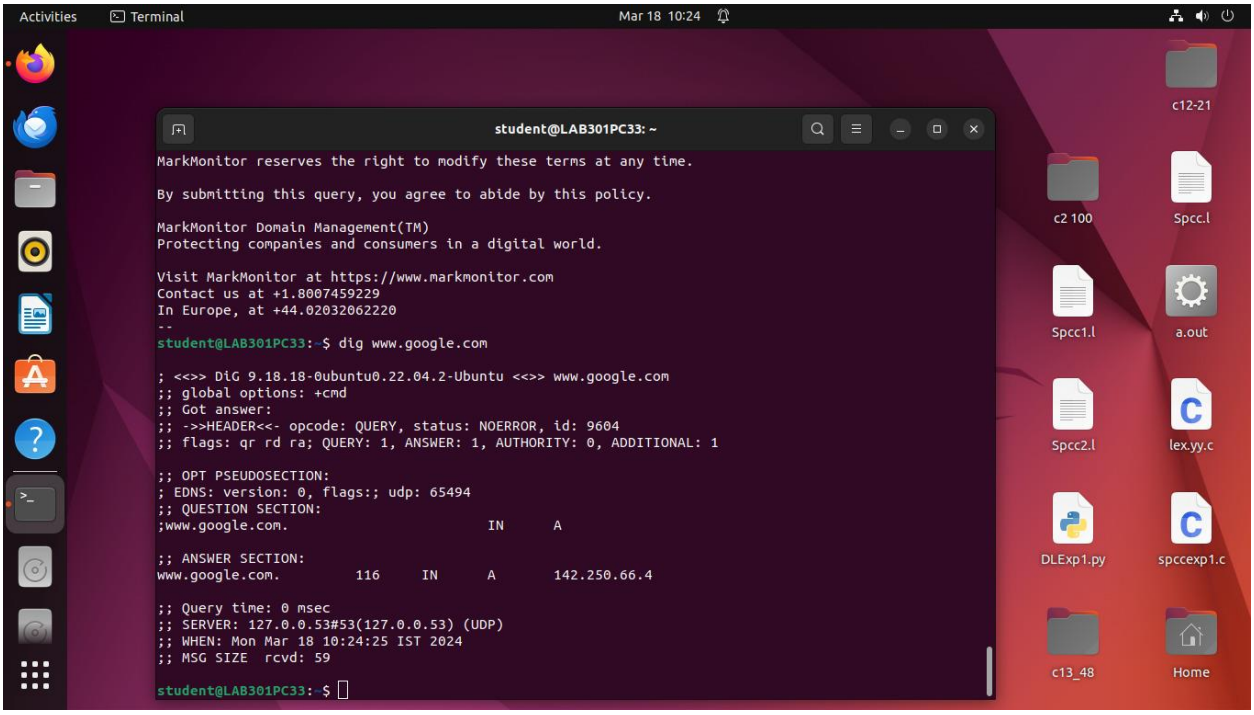
Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

Implementation /Code and output:

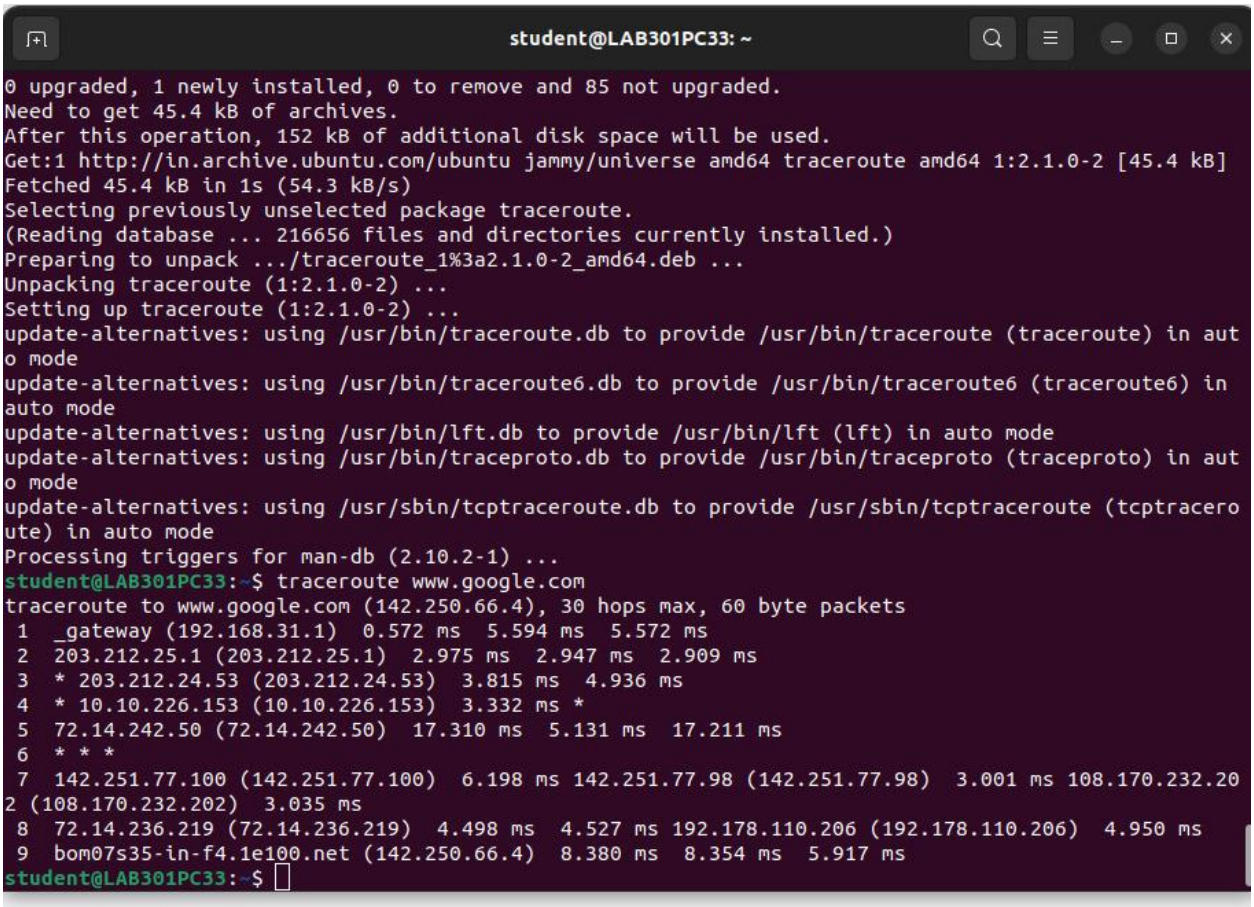


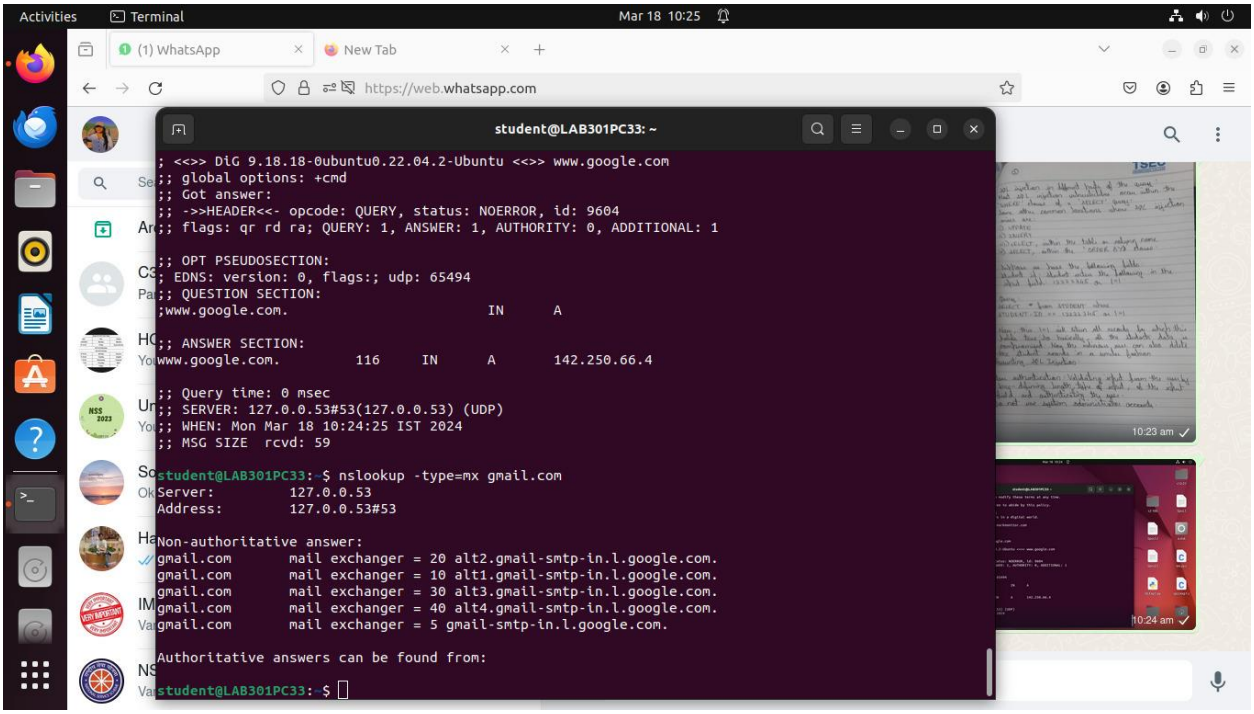


Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT





Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 8

Aim: Study and implementation of packet sniffer tool: Wireshark

Description/Algorithm:

Steps:

a) Download and install Wireshark and capture different packets like ICMP, TCP, and HTTP packets

b) Explore how the packets can be traced based on different filters

c) Capture packets of FTP and retrieve login ID and Password

Theory:-

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

Wireshark is used for:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and

display them in human-readable format. Wireshark includes filters, color-coding and other features that let you dig deep into network traffic and inspect individual packets.

Features of Wireshark :

• Available for UNIX and Windows.
• Capture live packet data from a network interface.
• Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a
• Import packets from text files containing hex dumps of packet data.
• Display packets with very detailed protocol information.
• Export some or all packets in a number of capture file formats.
• Filter packets on many criteria.
• Search for packets on many criteria.
• Colorize packet display based on filters.
Create various statistics.

Steps:

1. Open ubuntu terminal
2. Install wireshark
    # apt-get install wireshark
3. To know the name of your Ethernet interface: (Mostly it is “etht0”)

#ifconfig

4. Start wireshark

#sudo wireshark

5. Once wireshark window opens, select the interface and click on start

a) Capturing Packets

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface.

For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options.

As soon as you click the interface‘s name, you‘ll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

Click the stop capture button near the top left corner of the window when you want to stop capturing traffic

Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.

Wireshark can record the capturing information in the file with extension .pcap (packet capture). This file can be again reopened for analysis in offline mode.

There is no need to remember filtering commands. Filters can be applied by putting predefined strings in Wireshark.

Commands:-

1. Capturing packets of a particular host

ip.addr = = 192.168.42.3

Sets a filter for any packet with 192.168.42.3, as either the source or destination.

2. To capture a conversation between specified hosts

ip.addr == 10.0.5.119 && ip.addr == 91.189.94.25

Sets a conversation filter between the two defined IP addresses.

b) Filtering Packets

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type —dns and you‘ll see only DNS packets. When you start typing, Wireshark will help you auto complete your filter.

Commands:-

1. To filter packets for a specific protocol

http or dns

Sets a filter to display all http and dns requests.

2. To filter packets for specific port
tcp.port==4000
Sets a filter for any TCP packet with 4000 as a source or destination port.
3. Filter specific packets
tcp.flags.reset== 0
Displays all TCP resets.
4. Filter for http request packets
http.request
Displays all HTTP GET requests.
5. To filter traffic except given protocol packets
!(arp or icmp or dns)
Masks out arp, icmp, dns, or whatever other protocols may be background noise, allowing you to focus on the traffic of interest.
6. Capturing packets after applying multiple filters
not (tcp.port == 80) and not (tcp port == 25)
Get all packets which are not HTTP or UDP.

To stop capturing click on the “red square”

c) To capture packets of FTP server. (Login ID and Password)

What is FTP?
FTP stands for File Transfer Protocol. As the name suggest this network protocol allows you to transfer files or directories from one host to another over the network whether it is your LAN or Internet.

The package required to install FTP is known as VSFTPD (Very Secure File Transfer Protocol Daemon)

Steps:-
1. Get root access: $ sudo su root
2. Find your ip address: # ifconfig

Installation of FTP server in Ubuntu
Name of Packages required: VSFTPD, XINETD

1. # sudo apt-get install vsftpd
2. # sudo apt-get install xinetd
The above command will install and start the xinetd superserver on your system. The chances are that you already have xinetd installed on your system. In that case you can omit the above installation command.
In the next step we need to edit the FTP server's configuration file which is present in

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

/etc/vsftpd.conf
3. # cd /etc
4. # ls
5. # gedit vsftpd.conf
Change the following line:
Anonymous_enable=NO To Anonymous_enable=YES
This will instruct the FTP server to allow connecting with an anonymous client.
6. Save and close the gedit file

Now, that we are ready we can start the FTP server in the normal mode with:
7. # service xinetd restart
8. # service vsftpd restart OR # init.d/vsftpd restart

Start WIRESHARK. In the FILTER field put FTP. This will filter all FTP packets

Connectimg to a client present in other machine
$ ftp ip address of the FTP server
Name: anonymous
Please specify the password.
Password:
Login successful. (even if the login is not successful then also wireshark will capture the id and password)
ftp>
ftp> quit
Goodbye.

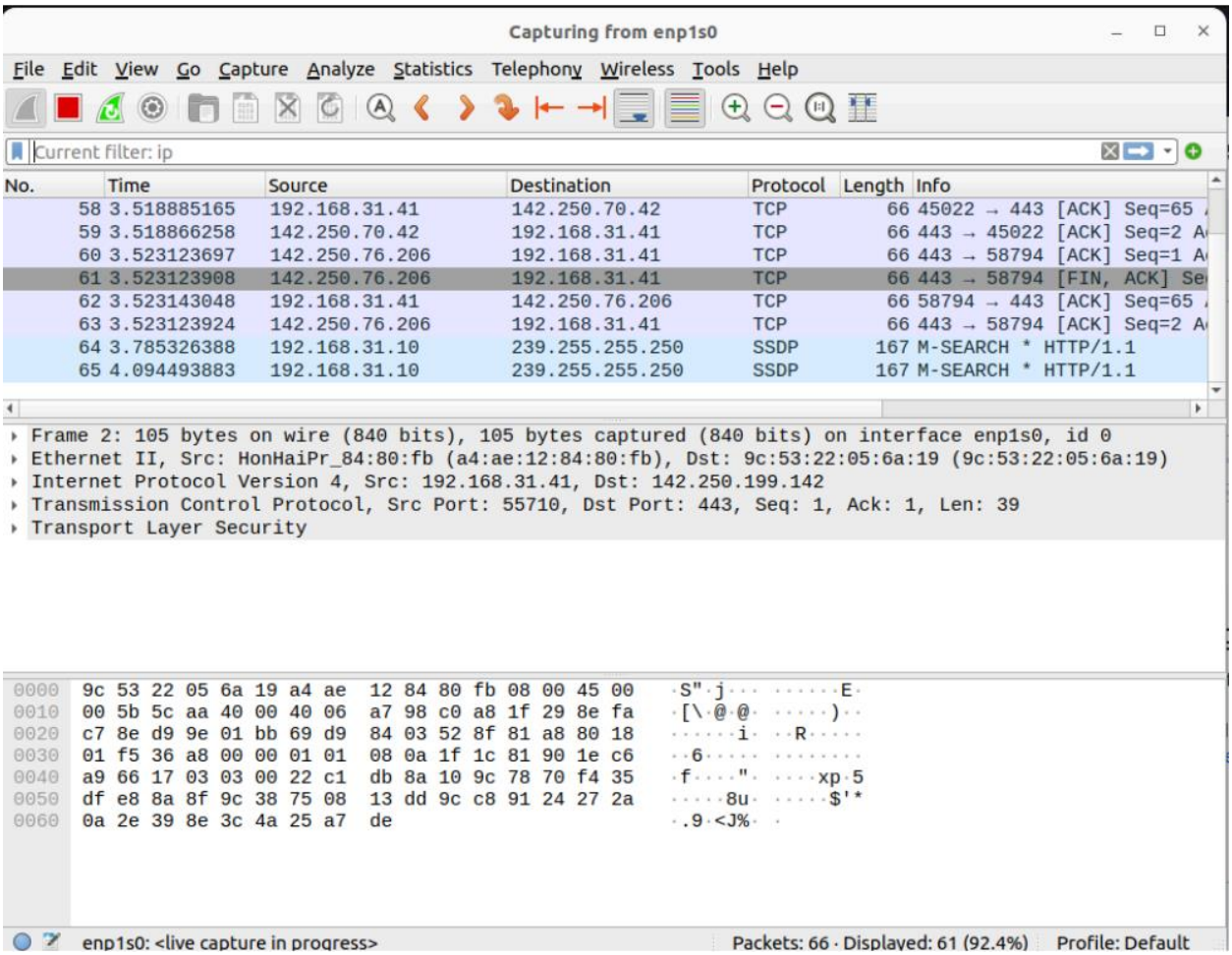
While the client is establishing a connection with the FTP server, the wireshark running in the background of the FTP server is able to capture all FTP packets. So, the Name and Password entered by the client is visible in plain text in Wireshark. Apart from that the source and destination address is also visible. If many clients are trying to connect with the server then source address, name and password are visible for all of them.

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

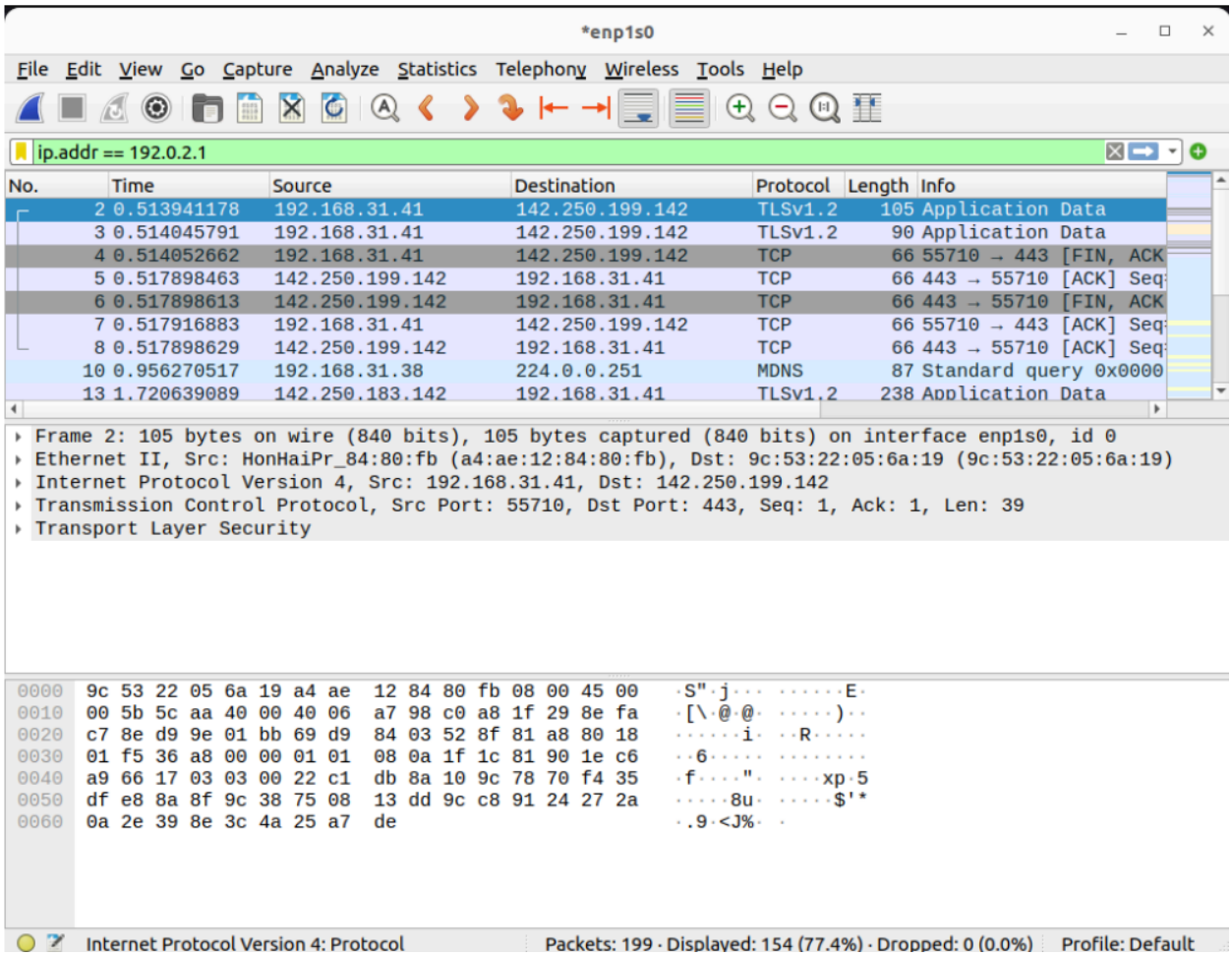
Implementation /Code and output:



Roll No : 2103156

Batch : C31

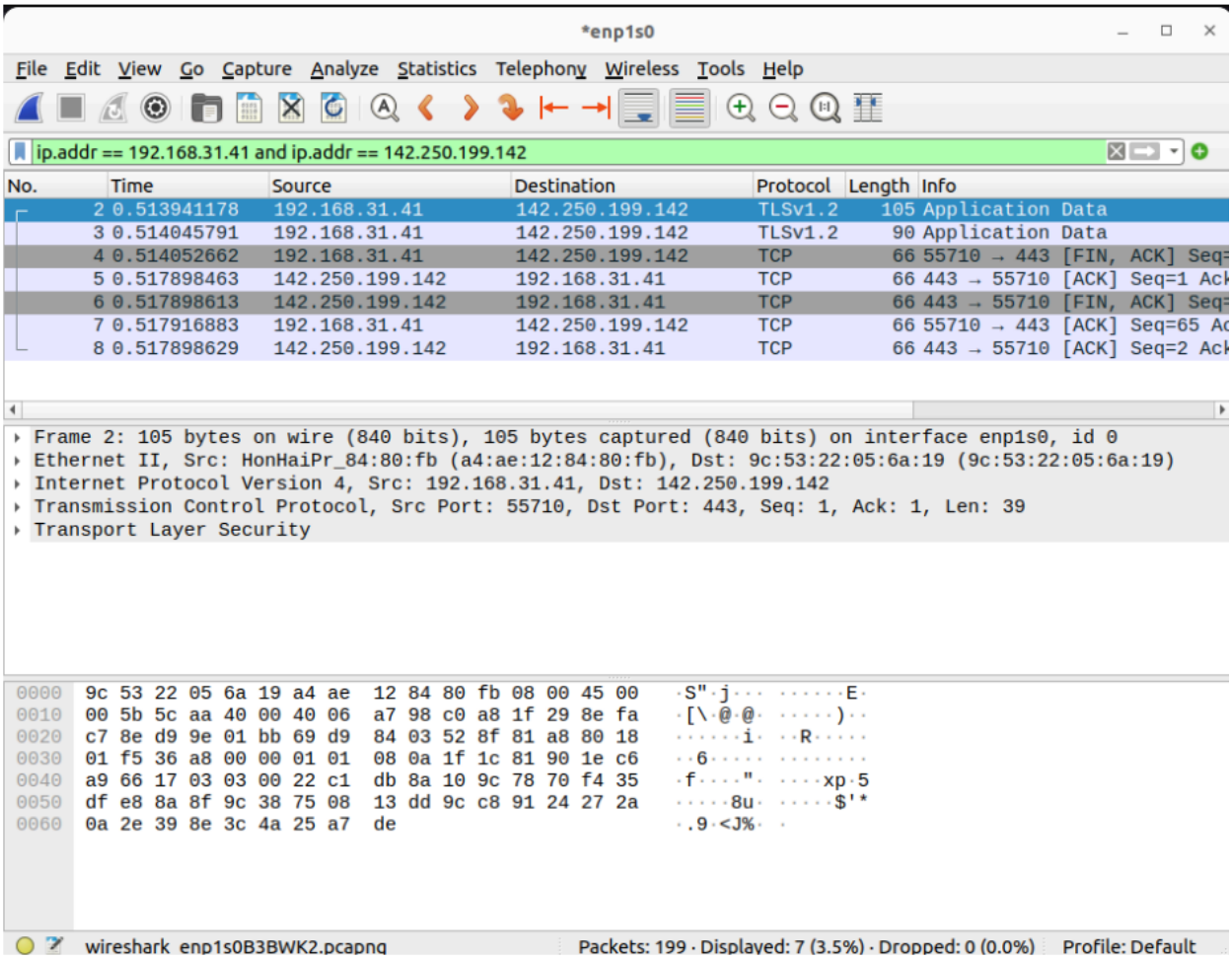
Name : NIYATI SAVANT



Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

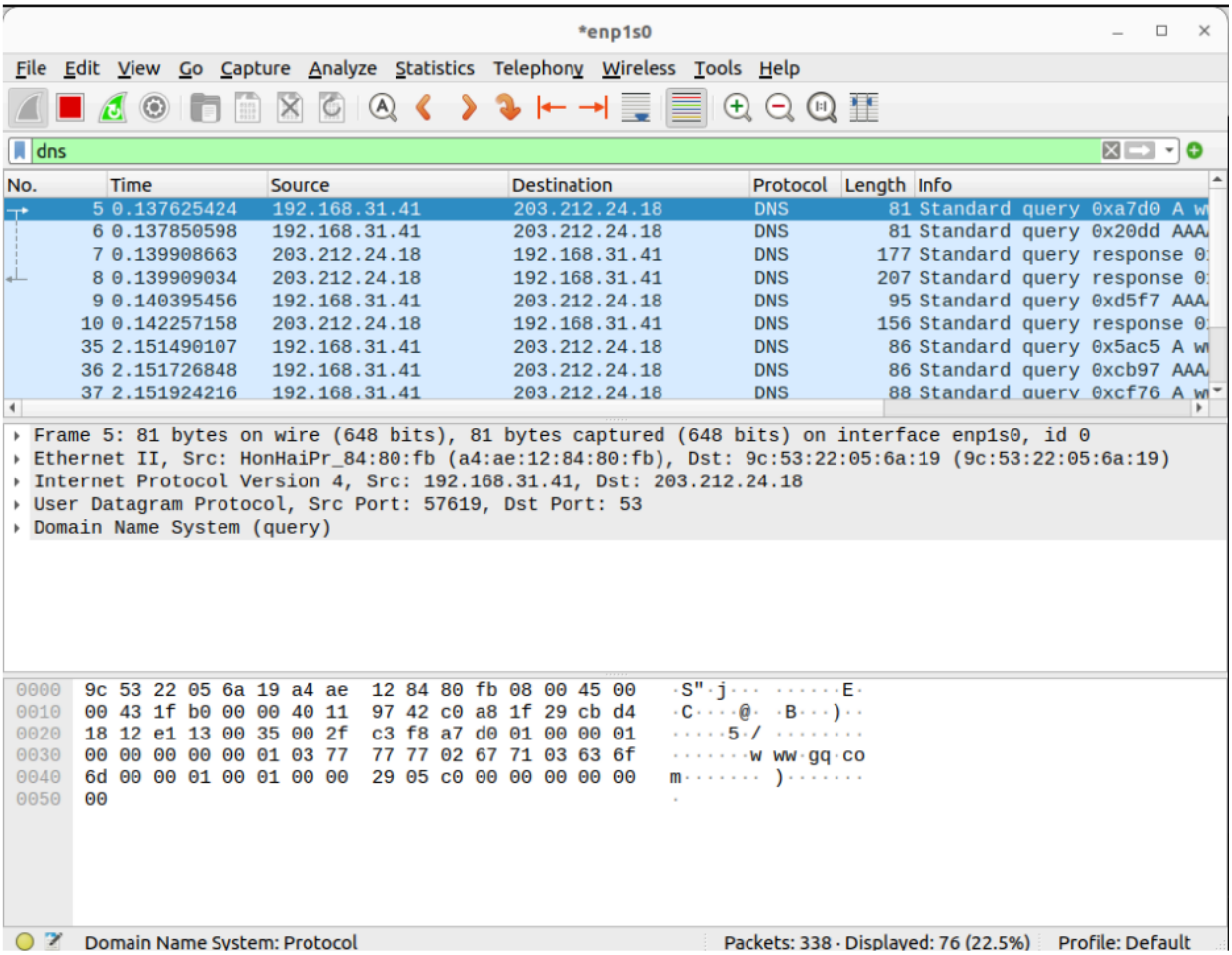


Cryptography and System Security Lab File (TE SEM VI Computer Engineering)

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

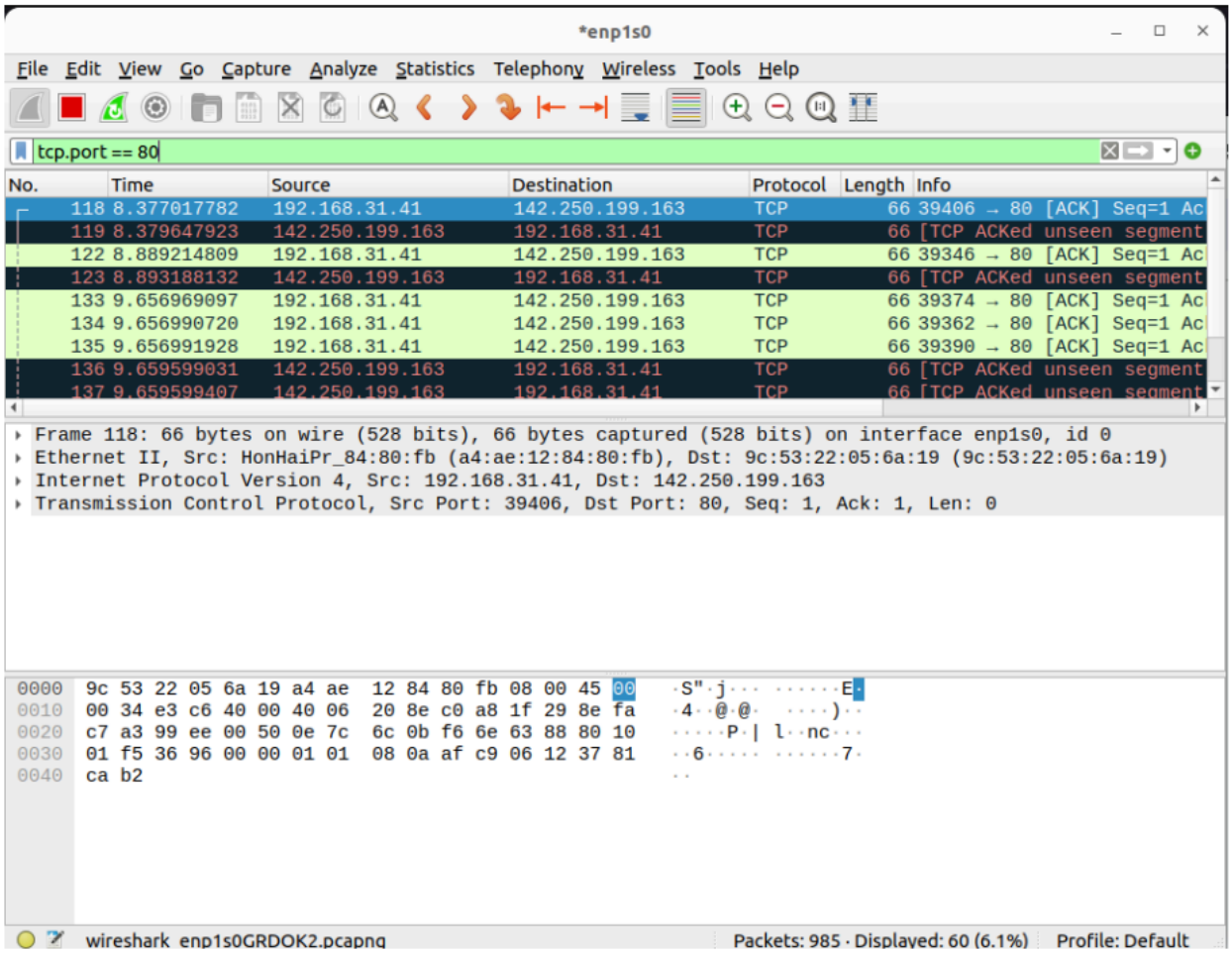


Cryptography and System Security Lab File (TE SEM VI Computer Engineering)

Roll No : 2103156

Batch : C31

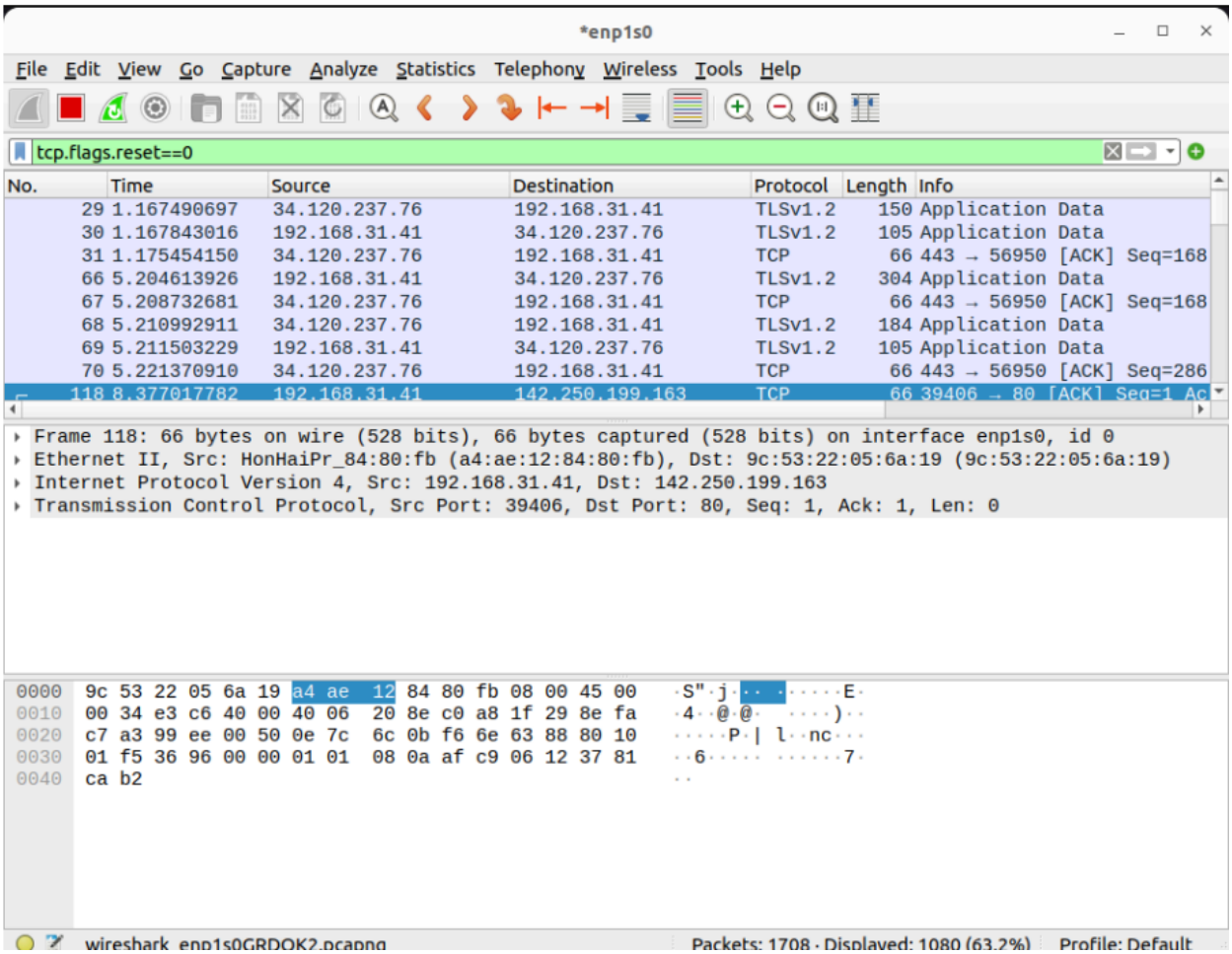
Name : NIYATI SAVANT



Roll No : 2103156

Batch : C31

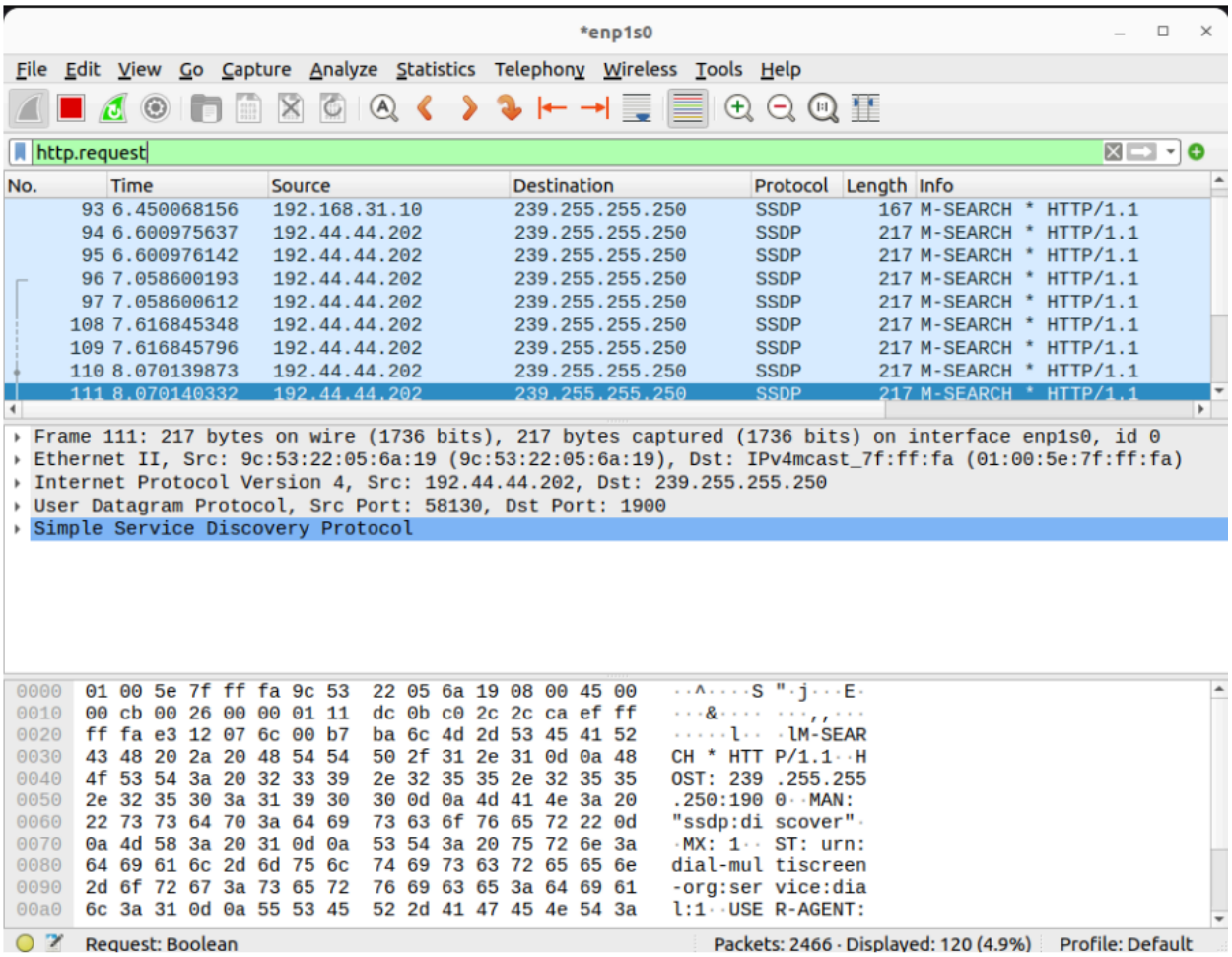
Name : NIYATI SAVANT



Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

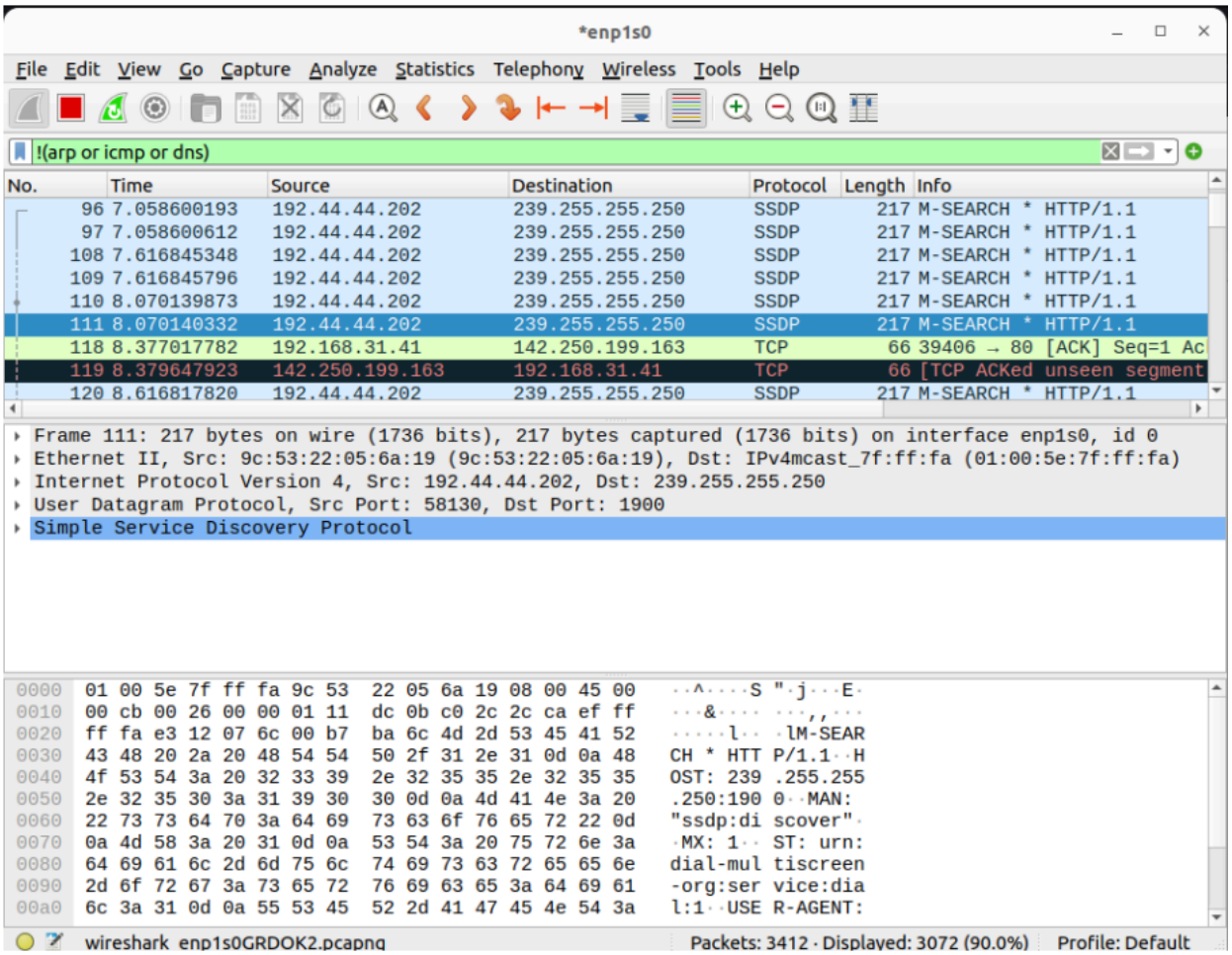


Cryptography and System Security Lab File (TE SEM VI Computer Engineering)

Roll No : 2103156

Batch : C31

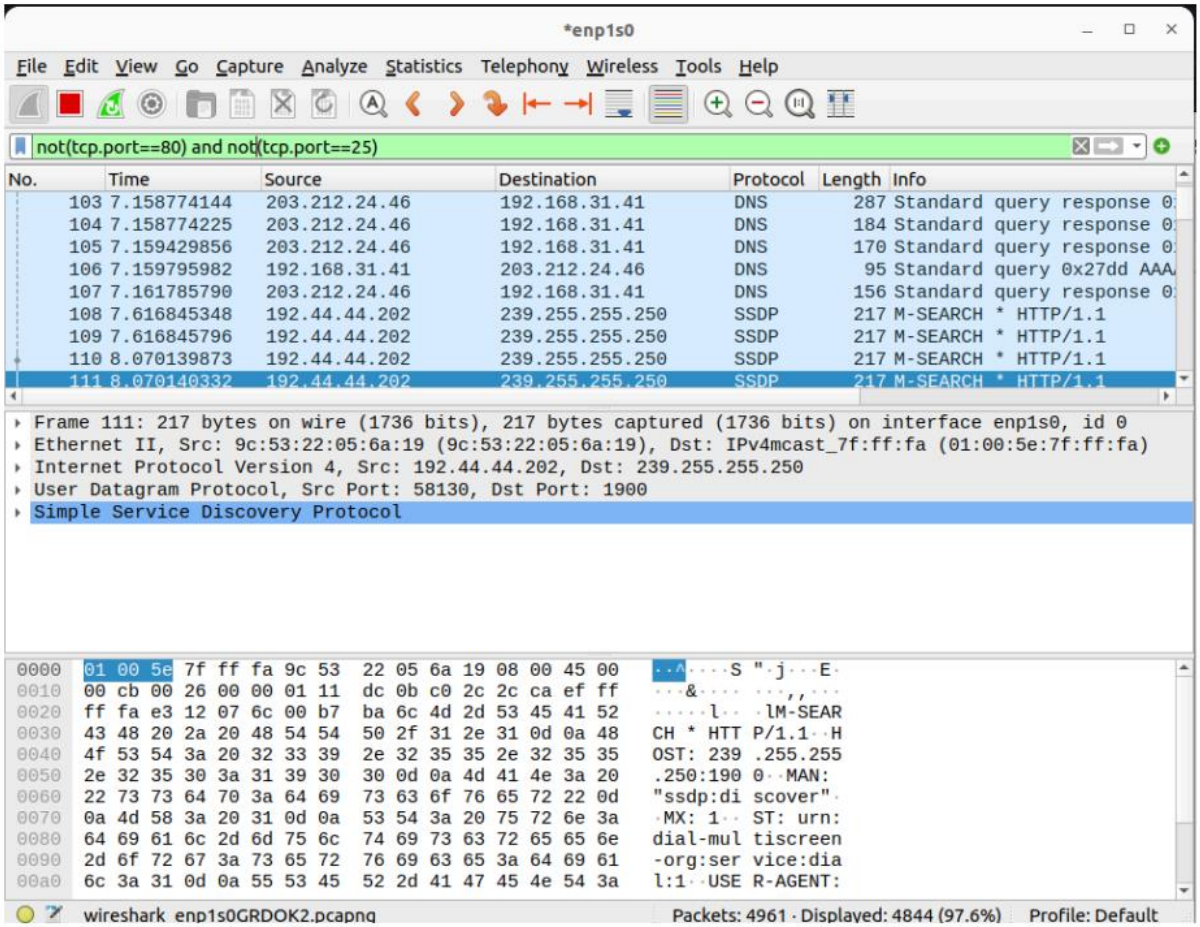
Name : NIYATI SAVANT



Roll No : 2103156

Batch : C31

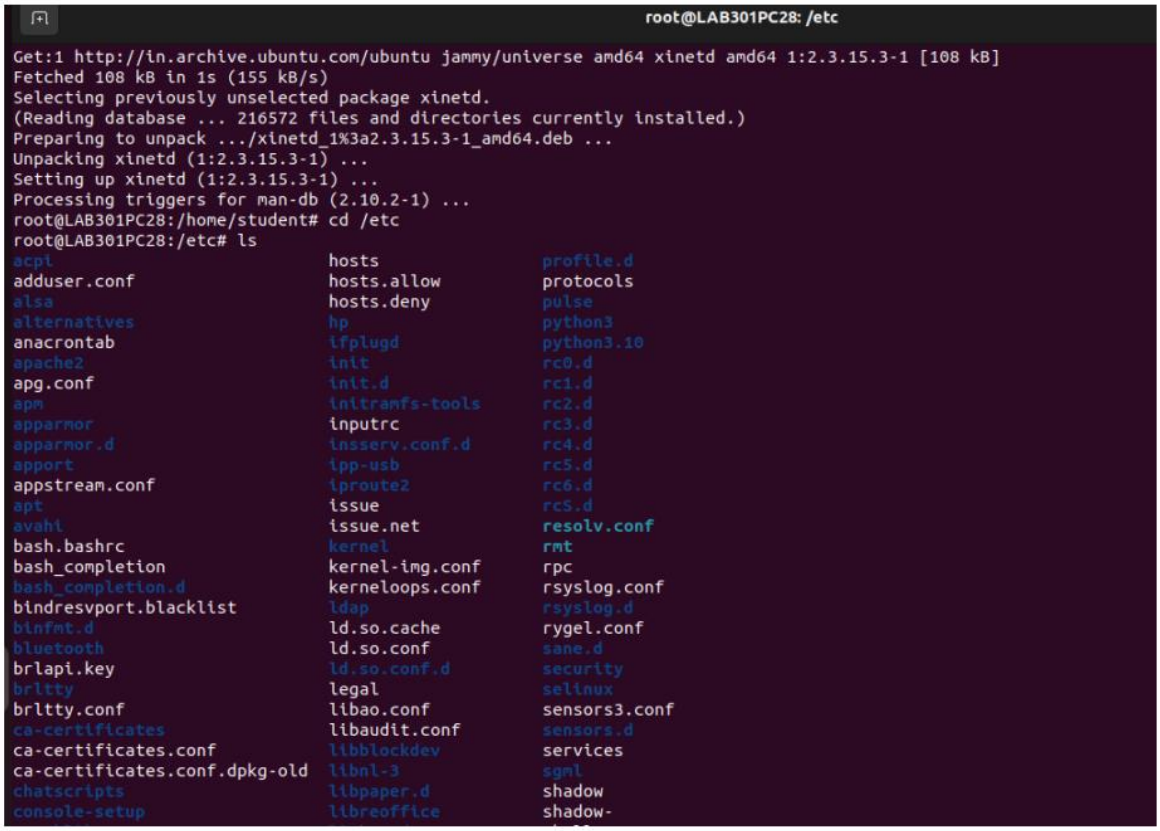
Name : NIYATI SAVANT

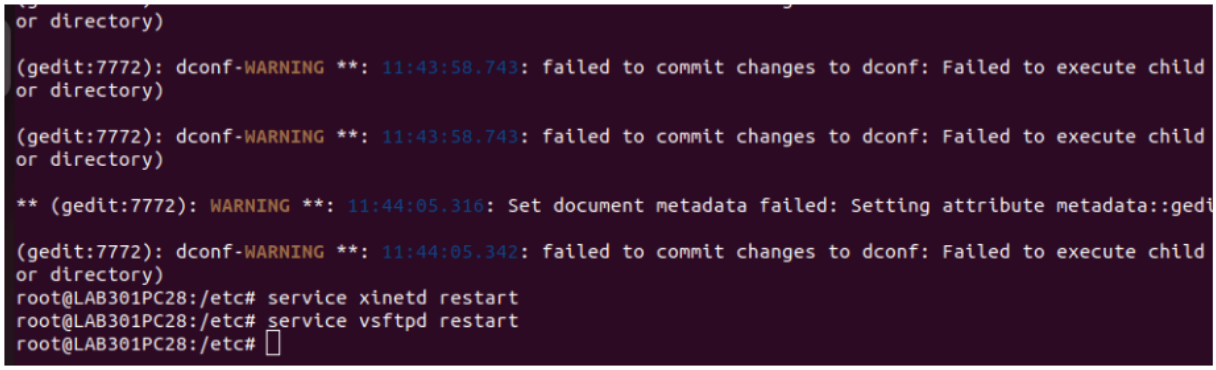


Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT



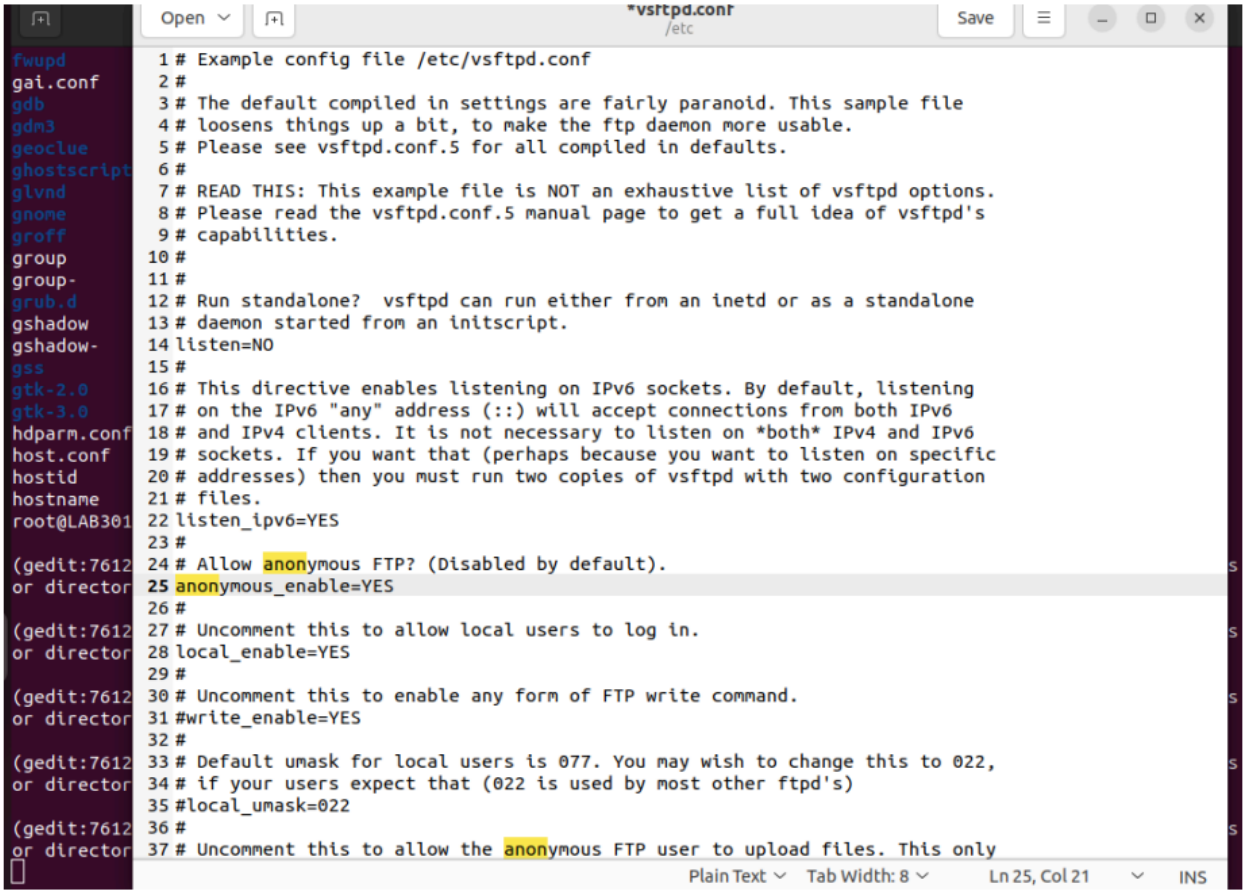


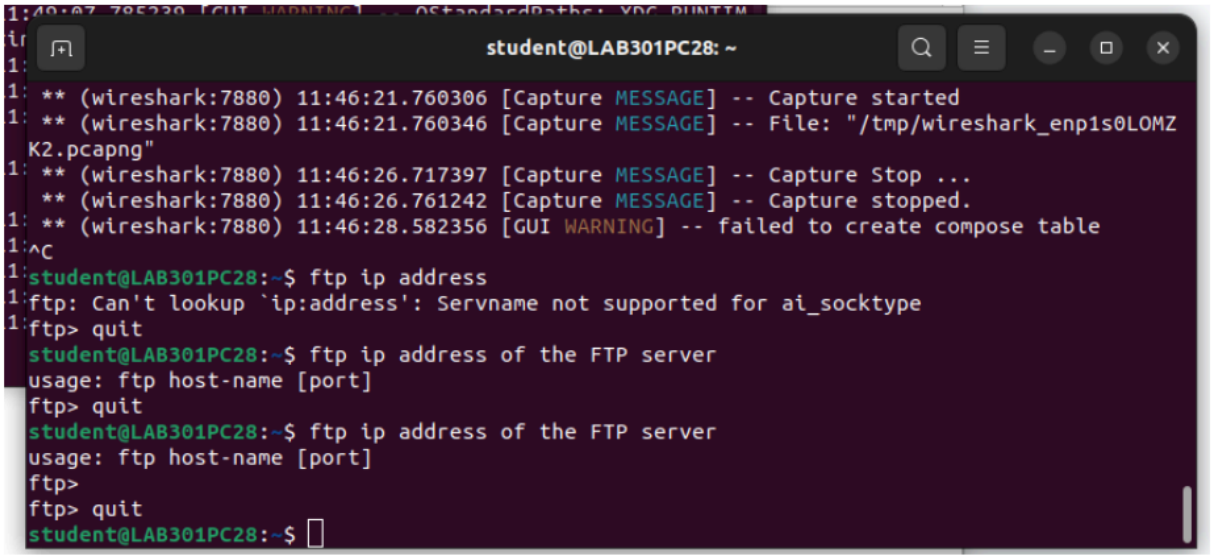
Cryptography and System Security Lab File (TE SEM VI Computer Engineering)

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT





Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 9

Aim: Design of personal Firewall using Iptables

Description/Algorithm:

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain.

1. Filter Table

Filter is default table for iptables.

Iptables's filter table has the following built-in chains.

- INPUT chain – Incoming to firewall. For packets coming to the local server.
- OUTPUT chain – Outgoing from firewall. For packets generated locally and going out of the local server.
- FORWARD chain – Packet for another NIC on the local server. For packets routed through the local server.

2. NAT Table

This table is consulted when a packet that creates a new connection is encountered.
Iptable's NAT table has the following built-in chains.

- PREROUTING chain – Alters packets before routing. i.e Packet translation happens immediately after the packet comes to the system (and before routing). This helps to translate the destination ip address of the packets to something that matches the routing on the local server. This is used for DNAT (destination NAT).
- POSTROUTING chain – Alters packets after routing. i.e Packet translation happens when the packets are leaving the system. This helps to translate the source ip address of the packets to something that might match the routing on the destination server. This is used for SNAT (source NAT).

- OUTPUT chain – NAT for locally generated packets on the firewall.

3. Mangle Table

Iptables's Mangle table is for specialized packet alteration. This alters QOS bits in the TCP header. Mangle table has the following built-in chains.

- PREROUTING chain
- OUTPUT chain
- FORWARD chain
- INPUT chain
- POSTROUTING chain

4. Raw Table

Iptable's Raw table is for configuration exemptions. Raw table has the following built-in chains.

- PREROUTING chain
- OUTPUT chain

5. Security Table

This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the SECMARK and CONNSECMARK targets. Mandatory Access Control is implemented by Linux Security Modules such as SELinux. The security table is called after the filter table, allowing any Discretionary Access Control (DAC) rules in the filter table to take effect before MAC rules. This table provides the following built-in chains: INPUT (for packets coming into the box itself), OUTPUT (for altering locally-generated packets before routing), and FORWARD (for altering packets being routed through the box).

Chains

Tables consist of *chains*, Rules are combined into different chains. The kernel uses chains to manage packets it receives and sends out. A chain is simply a checklist of rules which are lists of rules which are followed in order. The rules operate with an if-then -else structure.

Input – This chain is used to control the behaviour for incoming connections. For example, if a user attempts to SSH into your PC/server, iptables will attempt to match the IP address and port to a rule in the input chain.

Forward – This chain is used for incoming connections that aren't actually being delivered locally. Think of a router – data is always being sent to it but rarely actually destined for the router itself; the data is just forwarded to its target.

Output – This chain is used for outgoing connections. For example, if you try to ping howtogeek.com, iptables will check its output chain to see what the rules are regarding ping and howtogeek.com before making a decision to allow or deny the connection attempt.

Targets:

ACCEPT: Allow packet to pass through the firewall.

DROP: Deny access by the packet.

REJECT: Deny access and notify the server.

QUEUE: Send packets to user space.

RETURN: jump to the end of the chain and let the default target process it

iptables command Switch	Description
-L	Listing of rules present in the chain
-n	Numeric output of addresses and ports
-v	Displays the rules in verbose mode
-t <-table->	If you don't specify a table, then the filter table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
-j <target>	Jump to the specified target chain when the packet matches the current rule.
-A	Append rule to end of a chain
-F	Flush. Deletes all the rules in the selected table
-p <protocol-type>	Match protocol. Types include, icmp, tcp, udp, and all
-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters.
-o <interface-name>	Match "output" interface on which the packet exits

Steps:-

1. Get root access: $ sudo su root
2. # apt-get install iptables

Commands:-

1. To see the list of iptables rules

iptables -L

. Initially it is empty

2. To block outgoing traffic to a particular destination for a specific protocol from a machine

Syntax: iptables -I OUTPUT -s <your ip> -d <neighbour ip> -p <protocol> -j <action>

Open one terminal and Ping the neighbour. Let the ping run.
#ping 192.168.208.6

Open another terminal and run the iptables command
iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j DROP

2. To allow outgoing traffic to a particular destination for a specific protocol from a machine

iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j ACCEPT

3. To block outgoing traffic to a particular destination for a specific protocol from a machine for sometime

iptables -I OUTPUT -s 192.168.208.18 -d 192.168.208.6 -p icmp -j REJECT

Allow the traffic again by using ACCEPT instead of REJECT

4. To block incoming traffic from particular destination for a specific protocol to machine
Syntax: iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j <action>

Open one terminal and Ping the neighbour. Let the ping run.
#ping 192.168.208.6

Open another terminal and run the iptables command
iptables -I INPUT -s 192.168.208.6 -d 192.168.208.18 -p icmp -j DROP

5. To allow incoming traffic from particular destination for a specific protocol to machine
Syntax: iptables -I INPUT -s <neighbour ip> -d <firewall ip> -p <protocol> -j <action>

Open another terminal and run the iptables command
iptables -I INPUT -s 192.168.208.6 -d 192.168.208.18 -p icmp -j ACCEPT

Check the ping status on the other terminal

6. To clear the rules in iptables
iptables -F

7. To block specific URL from machine

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

 # iptables -t filter -I OUTPUT -m string --string facebook.com -j REJECT --algo kmp

It will block facebook.com by performing string matching. The algorithm used for string matching is KMP.
If we change target *from REJECT to ACCEPT*, the site can be visited again.

Observations:

1. In case of OUTPUT chain, for DROP and REJECT chain, at source machine we get two different messages.
   For DROP – 'Operation Not Permitted'. Here No acknowledgement is provided.
   For REJECT – 'Destination Port Unreachable'. Here acknowledgement is given.
2. In case of INPUT chain for DROP and REJECT chain at source machine we get two different responses as follows:
   For DROP – No message. Here No acknowledgement is provided.
   For REJECT – 'Destination Port Unreachable'. Here acknowledgement is given.

Implementation /Code and output:

```
root@admini-OptiPlex-3070:/home/admini# apt-get install iptables
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables is already the newest version (1.8.7-1ubuntu5.2).
The following packages were automatically installed and are no longer required:
  chromium-codecs-ffmpeg-extra gstreamer1.0-vaapi i965-va-driver
  intel-media-va-driver libaacs0 libaom3 libass9 libavcodec58 libavformat58
  libavutil56 libbdplus0 libblas3 libbluray2 libbs2b0 libchromaprint1
  libcodec2-1.0 libdav1d5 libflashrom1 libflite1 libftdi1-2 libgme0 libgsm1
  libgstreamer-plugins-bad1.0-0 libigdgmm12 liblilv-0-0 libllvm13 libmfx1
  libmysofa1 libnorm1 libopenmpt0 libpgm-5.3-0 libpostproc55 librabbitmq4
  librubberband2 libserd-0-0 libshine3 libsnappy1v5 libsord-0-0 libsratom-0-0
  libsrt1.4-gnutls libssh-gcrypt-4 libswresample3 libswscale5 libudfread0
  libva-drm2 libva-wayland2 libva-x11-2 libva2 libvdpau1 libvidstab1.1
  libx265-199 libxvidcore4 libzimg2 libzmq5 libzvbi-common libzvbi0
  mesa-va-drivers mesa-vdpau-drivers pocketsphinx-en-us va-driver-all
  vdpau-driver-all
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 164 not upgraded.
```

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

```
root@admini-OptiPlex-3070:/home/admini# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@admini-OptiPlex-3070:/home/admini# ping 192.168.208.6
PING 192.168.208.6 (192.168.208.6) 56(84) bytes of data.
```

```
root@admini-OptiPlex-3070:/home/admini# ping 192.168.1.185
PING 192.168.1.185 (192.168.1.185) 56(84) bytes of data.
64 bytes from 192.168.1.185: icmp_seq=1 ttl=64 time=0.527 ms
64 bytes from 192.168.1.185: icmp_seq=2 ttl=64 time=0.609 ms
64 bytes from 192.168.1.185: icmp_seq=3 ttl=64 time=0.541 ms
64 bytes from 192.168.1.185: icmp_seq=4 ttl=64 time=0.621 ms
64 bytes from 192.168.1.185: icmp_seq=5 ttl=64 time=0.475 ms
64 bytes from 192.168.1.185: icmp_seq=6 ttl=64 time=0.614 ms
64 bytes from 192.168.1.185: icmp_seq=7 ttl=64 time=0.614 ms
64 bytes from 192.168.1.185: icmp_seq=8 ttl=64 time=0.615 ms
64 bytes from 192.168.1.185: icmp_seq=9 ttl=64 time=0.507 ms
64 bytes from 192.168.1.185: icmp_seq=10 ttl=64 time=0.617 ms
64 bytes from 192.168.1.185: icmp_seq=11 ttl=64 time=0.272 ms
64 bytes from 192.168.1.185: icmp_seq=12 ttl=64 time=0.631 ms
64 bytes from 192.168.1.185: icmp_seq=13 ttl=64 time=0.604 ms
64 bytes from 192.168.1.185: icmp_seq=14 ttl=64 time=0.605 ms
64 bytes from 192.168.1.185: icmp_seq=15 ttl=64 time=0.353 ms
```

```
root@admini-OptiPlex-3070: /home/admini                root@admini-OptiPlex-3070: /home/admini
64 bytes from 192.168.1.185: icmp_seq=112 ttl=64 time=0.510 ms
64 bytes from 192.168.1.185: icmp_seq=113 ttl=64 time=0.613 ms
64 bytes from 192.168.1.185: icmp_seq=114 ttl=64 time=0.612 ms
64 bytes from 192.168.1.185: icmp_seq=115 ttl=64 time=0.615 ms
64 bytes from 192.168.1.185: icmp_seq=116 ttl=64 time=0.625 ms
64 bytes from 192.168.1.185: icmp_seq=117 ttl=64 time=0.620 ms
64 bytes from 192.168.1.185: icmp_seq=118 ttl=64 time=0.484 ms
^C
--- 192.168.1.185 ping statistics ---
126 packets transmitted, 118 received, 6.34921% packet loss, time 127970ms
rtt min/avg/max/mdev = 0.272/0.576/0.672/0.070 ms
```

```
root@admini-OptiPlex-3070:/home/admini# iptables -I OUTPUT -s 192.168.1.185 -d 192.168.1.185 -p icmp -j DROP
root@admini-OptiPlex-3070:/home/admini# iptables -I OUTPUT -s 192.168.1.140 -d 192.168.1.185 -p icmp -j DROP
root@admini-OptiPlex-3070:/home/admini#
```

```
root@admini-OptiPlex-3070:/home/admini# iptables -I OUTPUT -s 192.168.1.140 -d 192.168.1.185 -p icmp -j ACCEPT
root@admini-OptiPlex-3070:/home/admini# ping 192.168.1.185
PING 192.168.1.185 (192.168.1.185) 56(84) bytes of data.
64 bytes from 192.168.1.185: icmp_seq=1 ttl=64 time=0.490 ms
64 bytes from 192.168.1.185: icmp_seq=2 ttl=64 time=0.705 ms
64 bytes from 192.168.1.185: icmp_seq=3 ttl=64 time=0.703 ms
64 bytes from 192.168.1.185: icmp_seq=4 ttl=64 time=0.572 ms
^C
```

Roll No : 2103156

Batch : C31

Name : NIYATI SAVANT

```
root@admini-OptiPlex-3070:/home/admini# iptables -I OUTPUT -s 192.168.1.140 -d 192.168.1.185 -p icmp -j REJECT
root@admini-OptiPlex-3070:/home/admini# ping 192.168.1.185
PING 192.168.1.185 (192.168.1.185) 56(84) bytes of data.
From 192.168.1.140 icmp_seq=1 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.140 icmp_seq=2 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.140 icmp_seq=3 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.140 icmp_seq=4 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.140 icmp_seq=5 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 192.168.1.140 icmp_seq=6 Destination Port Unreachable
ping: sendmsg: Operation not permitted
^C
--- 192.168.1.185 ping statistics ---
6 packets transmitted, 0 received, +6 errors, 100% packet loss, time 5119ms
```

```
root@admini-OptiPlex-3070:/home/admini# iptables -t filter -A INPUT -i wlan0 -j DROP
root@admini-OptiPlex-3070:/home/admini# iptables --list --verbose
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  563 61429 DROP       all  --  any    any     anywhere             anywhere
    0     0 DROP       udp  --  any    any     anywhere             anywhere
    0     0 DROP       all  --  wlan0  any     anywhere             anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@admini-OptiPlex-3070:/home/admini#
```

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

Experiment 10

Aim: Simulation of Buffer Overflow Attack

Description/Algorithm:

A buffer is a temporary area for data storage. When more data (than was originally allocated to be stored) gets placed by a program or system process, the extra data overflows. It causes some of that data to leak out into other buffers, which can corrupt or overwrite whatever data they were holding.

In a buffer-overflow attack, the extra data sometimes holds specific instructions for actions intended by a hacker or malicious user; for example, the data could trigger a response that damages files, changes data or unveils private information.

Attacker would use a buffer-overflow exploit to take advantage of a program that is waiting on a user's input. There are two types of buffer overflows: stack-based and heap-based. Heap-based, which are difficult to execute and the least common of the two, attack an application by flooding the memory space reserved for a program. Stack-based buffer overflows, which are more common among attackers, exploit applications and programs by using what is known as a stack memory space used to store user input.

Modern compilers normally provide overflow checking option during the compile/link time but during the run time it is quite difficult to check this problem without any extra protection mechanism such as using exception handling.

Buffer overflow is a mistake that exist in some C implementations. These classes of bugs are dangerous as they write past the end of a buffer or array and hence corrupt the process stack They often change the return address of a process after a function call to a secret memory location where a malicious code is planted.

There are main two types

- Stack based attacks
- Heap based attacks

Heap-based attacks flood the memory space reserved for a program, but the difficulty involved performing such an attack makes them rare. Stack-based buffer overflows are by far the most common.

Splint is a tool for statically checking C programs for security vulnerabilities and programming mistakes

Splint does many of the traditional lint checks including unused declarations, type inconsistencies, use before definition, unreachable code, ignored return values, execution paths with no return, likely infinite loops, and fall through cases. More powerful checks are made possible by additional information given in source code annotations. Annotations are stylized comments that document assumptions about functions, variables, parameters and types. In addition to the checks specifically enabled by annotations, many of the traditional lint checks are improved by exploiting this additional information. Splint is designed to be flexible and allow programmers to select appropriate points on the effort benefit curve for particular projects. As different checks are turned on and more information is given in code annotations the number of bugs that can be detected increases dramatically.

Problems detected by Splint include:

- Dereferencing a possibly null pointer
- Using possibly undefined storage or returning storage that is not properly defined

Roll No : 2103156
Batch : C31
Name : NIYATI SAVANT

- Type mismatches, with greater precision and flexibility than provided by C compilers
- Violations of information hiding
- Memory management errors including uses of dangling references and memory leaks
- Dangerous aliasing
- Modifications and global variable uses that are inconsistent with specified interfaces
- Problematic control flow such as likely infinite loops, fall through cases or incomplete switches and suspicious statements
- Buffer overflow vulnerabilities
- Dangerous macro implementations or invocations
- Violations of customized naming conventions

Steps :
1. Installation
$ sudo apt-get install splint
2. Checking Vulnerability
$ splint program1.c
Program1.c is the program whose vulnerability is to be checked.

Implementation /Code and output:

```
#include <stdio.h>
int main(int argc, char *argv[])
{
int a = 100;
int b[8];
printf("Hello c\n");
b[8] = 100; // error
return 0;
}
```

```
admini@admini-OptiPlex-3070:~$ cd Desktop
admini@admini-OptiPlex-3070:~/Desktop$ splint prog1.c +bounds -paramuse -varuse
Splint 3.1.2 --- 21 Feb 2021

prog1.c: (in function main)
prog1.c:9:5: Likely out-of-bounds store: b[8]
    Unable to resolve constraint:
    requires 7 >= 8
     needed to satisfy precondition:
    requires maxSet(b @ prog1.c:9:5) >= 8
  A memory write may write to an address beyond the allocated buffer. (Use
  -likelyboundswrite to inhibit warning)

Finished checking --- 1 code warning
admini@admini-OptiPlex-3070:~/Desktop$
```

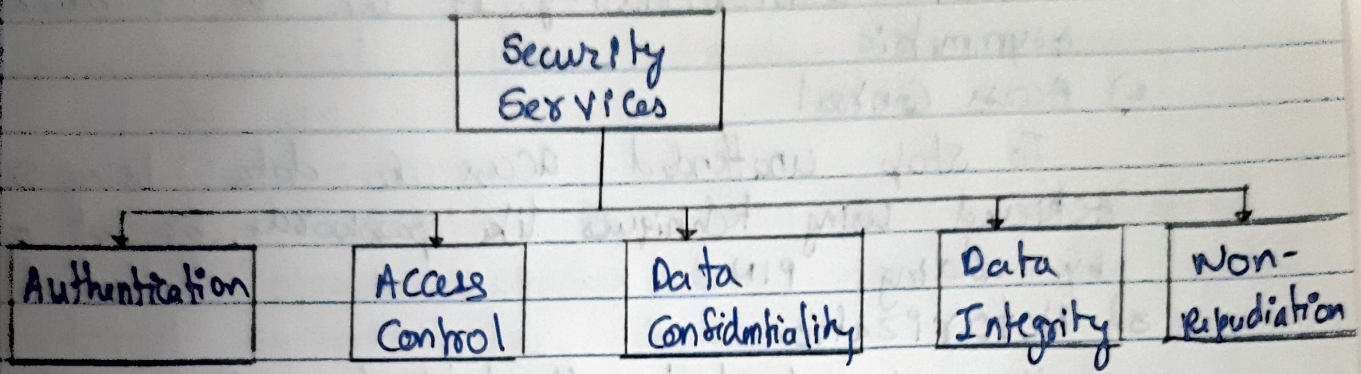
Aft Savant
:31
03156

Written Assignment -01

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Explain different services and security mechanisms

Security Services are safeguard controls recommended to be placed at various OSI layers



- Authentication assures recipient that message is from the source that it claims to be from like peer entity authentication, data origin authentication.
- Access control controls who has access to resource and under what condition.
- Confidentiality ensures that information is unavailable to unauthorized user and include connection confidentiality, connectionless, selective field and traffic flow confidentiality.
- Integrity ensures the message is unaltered
- Non-Repudiation - Protects against denial of sending or receiving in the communication. It could be non-repudiation with proof of origin and with proof of delivery.

Security mechanisms is a set of processes that deal with recovery from security attacks

1) Encipherment ie Encryption
   Deals with hiding and covering of data to maintain confidentiality. It can be symmetric or Asymmetric
2) Access Control
   To stop unattended access to data being send. Achieved using techniques like passwords, firewall or by adding PIN
3) Notarization
   Use of trusted third party that acts as mediator and reduces chances of conflict.
4) Integrity
   A value created by data itself is appended and is checked at both sides
5) Authentication
   Deals with identity to be known achieved at TCP/IP layer for two-way handshaking mechanism
6) Bit Stuffing
   To add extra bit to data being transmitted achieved using Even or Odd parity
7) Digital Signatural
   Form of electronic signature added by sender (not visible to eyes) which is checked by receiver electronically.

6] What are various types of attacks? Explain with example

- cyber attack or attempts by cybercriminals, hackers to attack a computer network or system, virtually to alter, steal, explore information. The common attacks are

a) Active attacks
Attacker attempts to alter, destroy or disrupt normal operation. Attacker takes direct action against target system. They includes

i) Masquerade - Attacker pretends to be someone else to gain access to systems or data. It includes username and password masquerade. IP address masquerade, website masquerade, email masquerade

ii) Modification - Attacks integrity by altering some part of a message or delay or reorder it to produce an unauthorized effect

iii) Repudiation - Attacker attempts to deny or repudiate actions they have taken like making transaction or sending message. It could happen with message transaction or data

iv) Replay - Involves passive capture of a message and its subsequent transmission to produce an authorized effect

v) Denial of service - Attack designed to make a system or network unavailable to its intended users by overwhelming with traffic or requests. It includes

flood attacks, amplification attacks

b) Passive attacks
It attempts to learn or use information system but not affect system resources. It to eavesdropping or monitoring transmission. It includes sniffing i.e. capture and analyze packets to steal sensitive information or eavesdrop when an attacker listens to network traffic. The types are

1) Release of message content
Telephonic conversation, electronic mail message or transferred file may contain sensitive information and we prevent an opponent from learning its contents.

2) Traffic analysis.
Encryption of SIP traffic is most useful protection against traffic analysis where attacker needs to have access to SIP proxy to determine who made the call.

c] List few latest viruses on net and explain

→ Virus is a malicious software or malware that spreads between computers and damages data and software.

1) Clop ransomware
A malware that encrypts your files until you pay a ransom to the hackers. It is a variant of well-known Cryptomix ransomware

which blocks every boot process and applications before encrypting data

2) Bidder Ransomware

It belongs to Win32 Crypt ransomware family that tricks users into installing it by displaying a fake Windows Update message with the .exe file in disguise. After installation, it encrypts data and demands money for decryption

3) Zeus Gameover

It is Sophisticated malware that steals banking and other credentials and steals all the money and is spread through spam e-mails.

4) RaaS

It is a Ransomware as a Service, a pay-for-use malware created by experts to be sold to customers who use it to hold people's data hostage

5) WLWS malware Attacks

Hackers utilize trending news by sending a link to readers mail and if contains a virus that steals your information

29/01/24

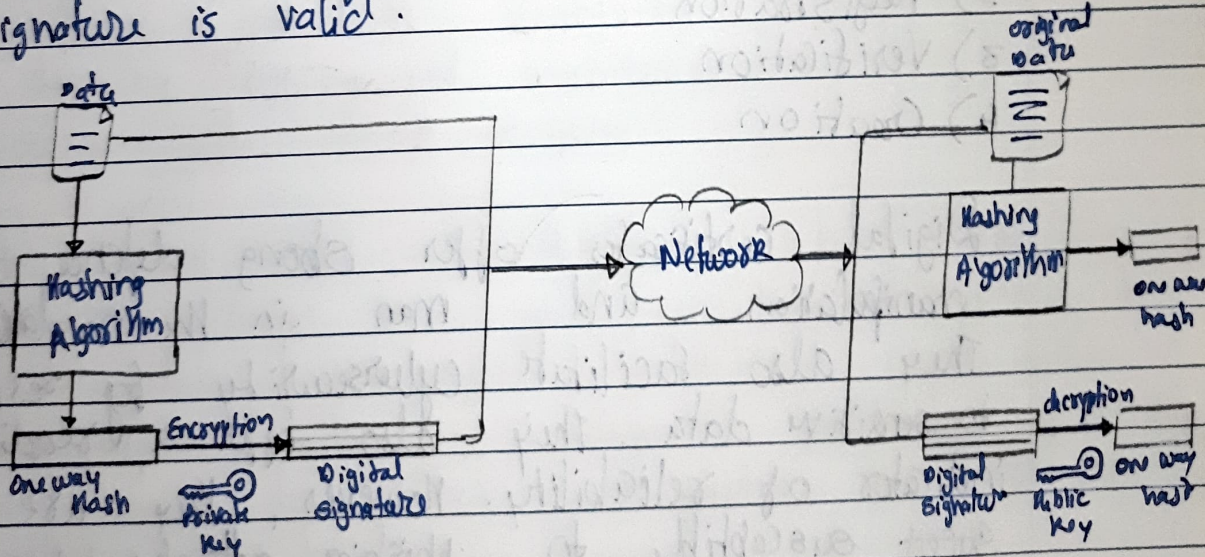
Riyati Savant
TE - C31
2103156

Written Assignment - 02

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Q Short note on digital signatures and digital certificates

→ Digital signature is a mathematical technique used to validate authenticity and integrity of a message, software, or digital document. These are electronic signatures, which assure that message was sent by the particular sender. Signing algorithms like email programs create a one-way hash of electronic data which is to be signed. It then encrypts hash value using private key. The hash along with other information like hashing algorithm is digital signature which is appended with data and send to verifier. Verifier receives signature along with data. It uses verification algorithm to process on signature and public key and generate value. It applies same hash function on received data to generate hash value. If both are equal, signature is valid.



They offer authenticity, integration, non-repudiation, notorization. However, they rely on technology

and setting up and using them is challenging and a not widely accepted.

A digital certificate is a file issued by a trusted third party that provides proof of sender's identity to receiver and vice versa. It is issued to CA (Certificate Authority) to verify identity of certificate holder. It is used to attach public key with a particular individual or entity. A digital certificate contains name of the certificate holder, serial number to identify a certificate, expiration date, copy of holder's public key (for decryption), digital signature of CA.

The certificate is also send with signature and message. It involves the following steps -

1) Key Generation
2) Registration
3) Verification
4) Creation

Digital certificates offer strong defence against manipulation and man in the middle assaults. They also facilitate cybersecurity by restricting access to sensitive data. They also offer a readily identifiable indicator of reliability. However, they are still susceptible to phishing attacks, weaker encryption, leading to intrusions and misconfigurations.

18/03/24

Niyati Savant
TE - C81
8103156

Written Assignment - 03

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q1] List software vulnerabilities. How are they exploited to launch an attack

→ Software vulnerabilities are weaknesses or flaws present in the code, which if left alone can impact the performance and security of software by allowing untrustworthy agents to exploit or gain access to your products and data

1) Broken Access control
2) Cryptographic Failures
3) Injection
4) Lack of Threat modeling, secure design patterns and principles
5) security misconfiguration due to insecure configurations, Open cloud storage, misconfured HTTP headers
6) Vulnerable & Outdate Components
7) Identification and Authentication Failures
8) Software and Data Integrity Failures
9) Security Logging and monitoring failures
10) Server-side request forgery

- Broken Access Control - The attackers exploit insufficient access restrictions to gain unauthorized access to sensitive data or perform privileged users actions
- Cryptographic Failures - Inadequate protection of sensitive data due to weak encryption or improper handling of keys.
- Injection - Attackers insert malicious code like SQL queries or commands, into input fields or data

streams, tricking the application into executing unintended actions or disclosing sensitive information from backend systems.

- Insecure Design - Attackers exploit design flaws, input validation or insecure default configurations to manipulate the system's behaviour
- Security Misconfiguration - Attackers leverage improperly configured settings or systems to gain unauthorized access, compromise sensitive data.
- Vulnerable and Outdated Components - Attackers exploit known vulnerabilities in outdated or vulnerable software components to gain unauthorized access, execute arbitrary code.
- Identification and Authentication Failures - Attackers exploit authentication mechanisms like weak passwords or lack of ~~mutual~~ multi-~~factor~~ factor authentication, to impersonate legitimate users.
- Software and data Integrity Failures - Attackers tampers with software updates or data transmission to inject malicious code, data or compromise integrity.
- Security logging and monitoring Failures - Attackers exploit insufficient logging and monitoring to ~~cover~~ covers their tracks, evade detection or launch attacks without being detected prolonging duration of their unauthorized access and activities.
- Server-Side Request Forgery (SSRF): Attackers abuse the vulnerabilities to make unauthorized requests from server to internal or external resources, enabling them to access sensitive data.

12] SQL injection

- A technique used to extract user data by injecting web page inputs as statements through SQL commands.
- The webserver communicate with database server anytime they need to retrieve or store user data. SQL statements are designed so that they can be executed while server is fetching content from application server
- Thus hacker can retrieve all data present in database like credit information, SSNs or delete user from table

Example of SQL injection

Consider a database for student records where student can view record using private student id i.e sid.

A student enters : 12 222 345 or 1=1

∴ The Query now becomes,

select * from student where,

Sid == 12222345 or 1=1

Now this 1=1 return all records for which this holds true. So basically, all students data is compromised. Now the malicious user can also delete the student record in a similar fashion.

Preventing SQL injection -

1. User Authentication
Validating user input from apps by pre-defining length, type of input field and user authentication

2. Restricting access privileges to users 'pre-defining how much data any outsider can access. They should not grant permission to access everything in the database

3. Do not use system administrator accounts.

4. Monitor SQL statements from database. Connected apps in real time. This will aid in detecting rogue SQL statements as vulnerabilities.

08/04/24