Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SUBJECT IN CHARGE: ER. SHRINIDHI GINDI**

**COURSE: SECURITY LAB**

**COURSE CODE: ITL502**

**ACADEMIC YEAR: 2021-2022**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SR NO.** | **EXPERIMENTS NAMES** | **DOP** | **DOS** | **MARKS** | **SIGN** |
| **1** | Design and implement a product cipher using Substitution ciphers | 28/7/21 | 04/08/21 |  |  |
| **2** | Encrypt long messages using various modes of operation using DES. | 04/08/21 | 11/08/21 |  |  |
| **3** | Cryptoanalysis or decoding Playfair, Vigenere Cipher | 11/08/21 | 25/08/21 |  |  |
| **4** | Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method. | 25/08/21 | 25/08/21 |  |  |
| **5** | Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA | 25/08/21 | 01/09/21 |  |  |
| **6** | Cryptographic Hash Functions and Applications | 01/09/21 | 08/09/21 |  |  |
| **7** | Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars. | 08/09/21 | 15/09/21 |  |  |
| **8** | Study of packet sniffer tools Wireshark:  a) Observer performance in promiscuous as well as non -promiscuous mode.  b) Show the packets can be traced based on different filters | 15/09/21 | 29/09/21 |  |  |
| **9** | Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc. | 29/09/21 | 06/10/21 |  |  |
| **10** | Study of malicious software using different tools:  a) Keylogger attack using a keylogger tool.  b) Simulate DOS attack using Hping or other tools  c) Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities. | 06/10/21 | 18/10/21 |  |  |
| **11** | Study of Network security by  a) Set up IPSec under Linux.  b) Set up Snort and study the logs.  c) Explore the GPG tool to implement email security | 18/10/21 | 22/10/21 |  |  |

**AVERAGE MARKS OF ALL EXPERIMENTS:**

**ASSIGNMENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SR NO.** | **ASSIGNMENT NAMES** | **DOP** | **DOS** | **MARKS** | **SIGN** |
| **1** | Cryptography: Key management, distribution and user authentication and malware | 03/08/21 | 10/08/21 |  |  |
| **2** | IP security, Transport level security, Email Security and NAC | 20/08/21 | 30/08/21 |  |  |

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **01** |
| Title: | **Design and implement a product cipher using Substitution ciphers.** |
| Date of Performance: | **28/07/21** |
| Date of Submission: | **04/08/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

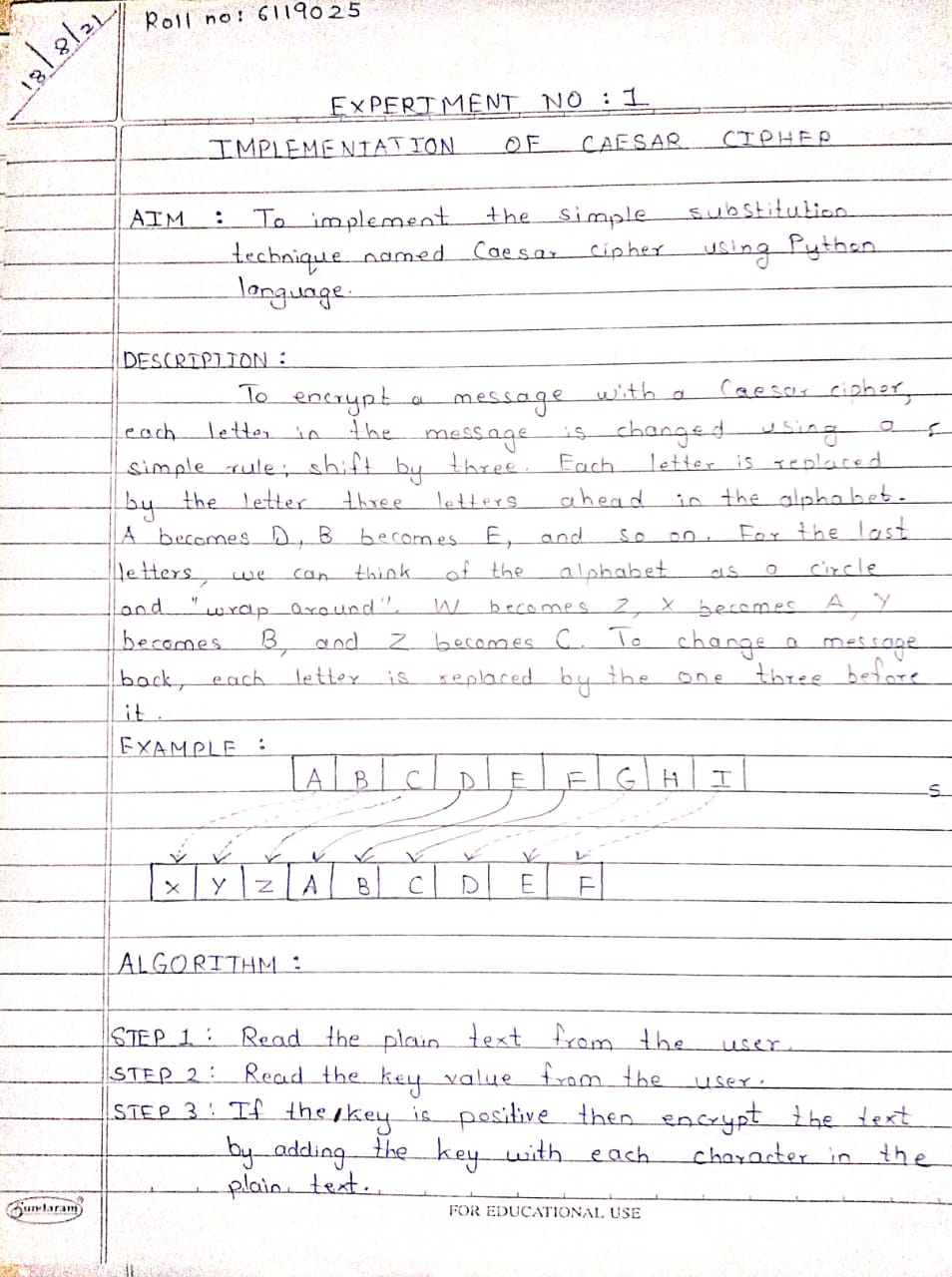
Date:

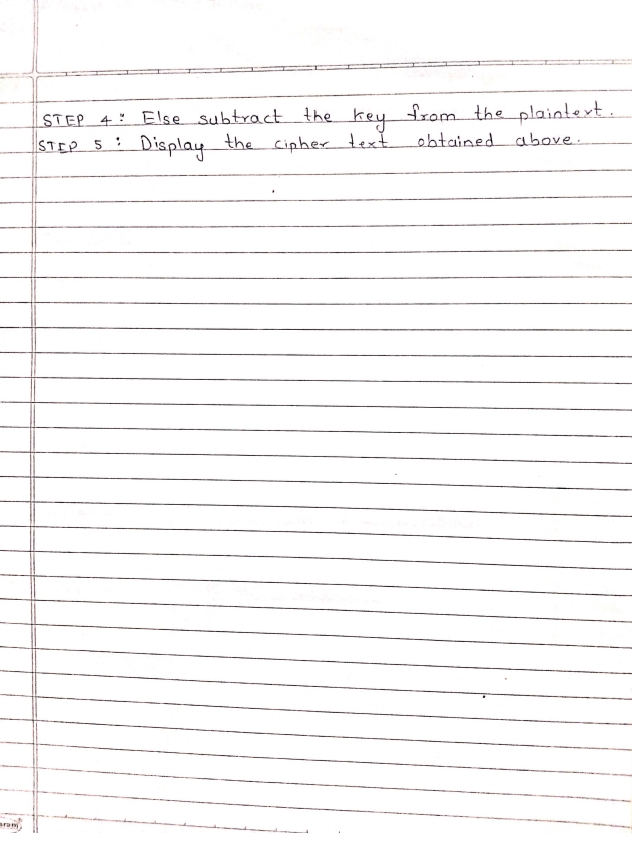
**EXPERIMENT NO: 01**

**AIM:** Design and implement a product cipher using Substitution ciphers.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Compiler.

**THEORY:**

****

****

**CODE:**

text=input("Enter text to be encrypted : ")

key=int(input("Enter key : "))

alpha ="abcdefghijklmnopqrstuvwxyz"

encrypt=""

decrypt=""

for c in text:

i = (alpha.index(c)+key)%26

encrypt+= alpha[i]

print("Encrypted text : ",encrypt)

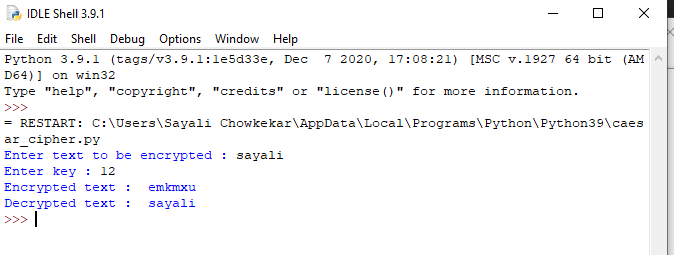
for c in encrypt:

i =(alpha.index(c)-key)%26

decrypt+=alpha[i]

print("Decrypted text : ",decrypt)

**OUTPUT:**

****

**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Product cipher using substitution implemented successfully.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **02** |
| Title: | **Encrypt long messages using various modes of operation using DES.** |
| Date of Performance: | **04/08/21** |
| Date of Submission: | **11/08/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

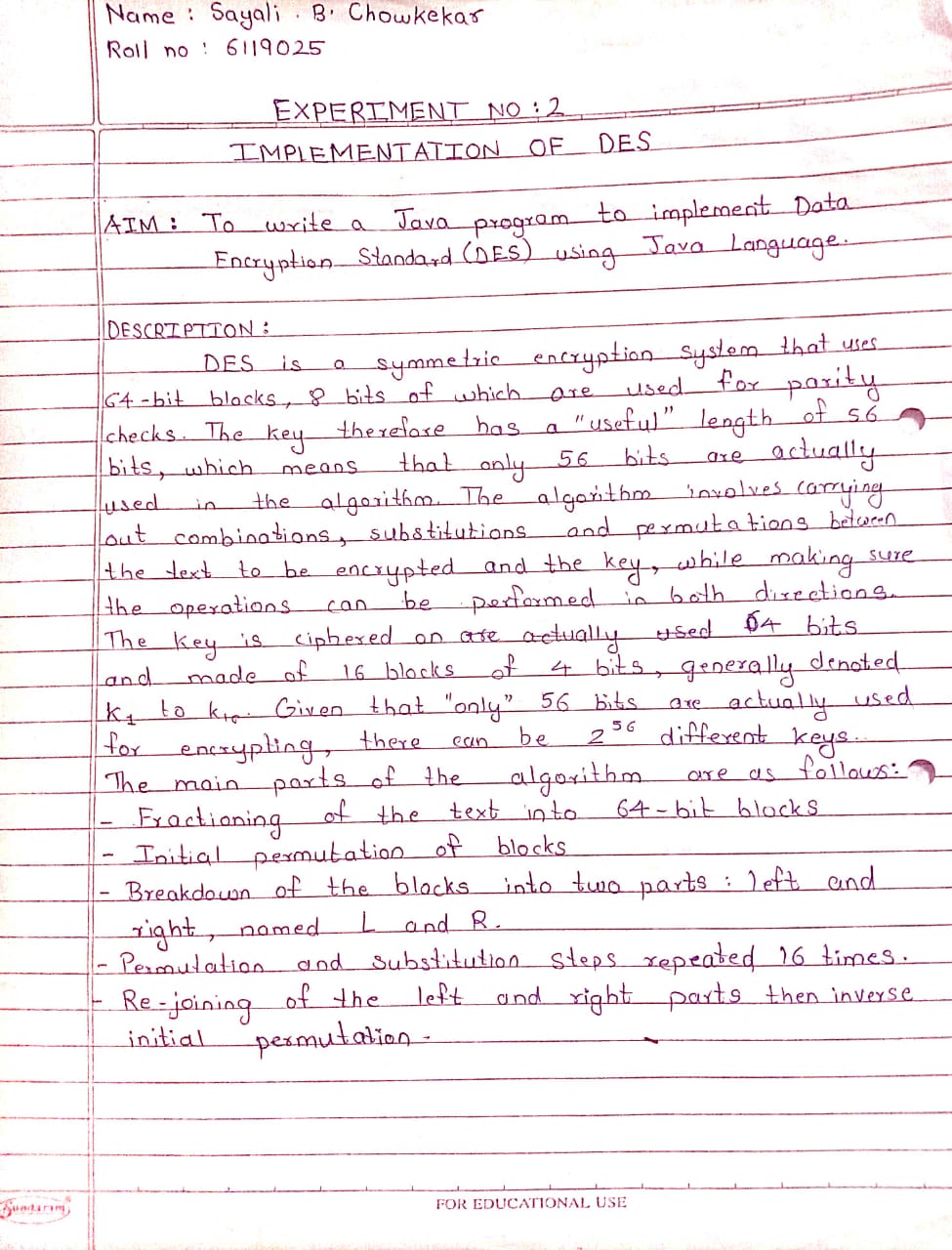
Date:

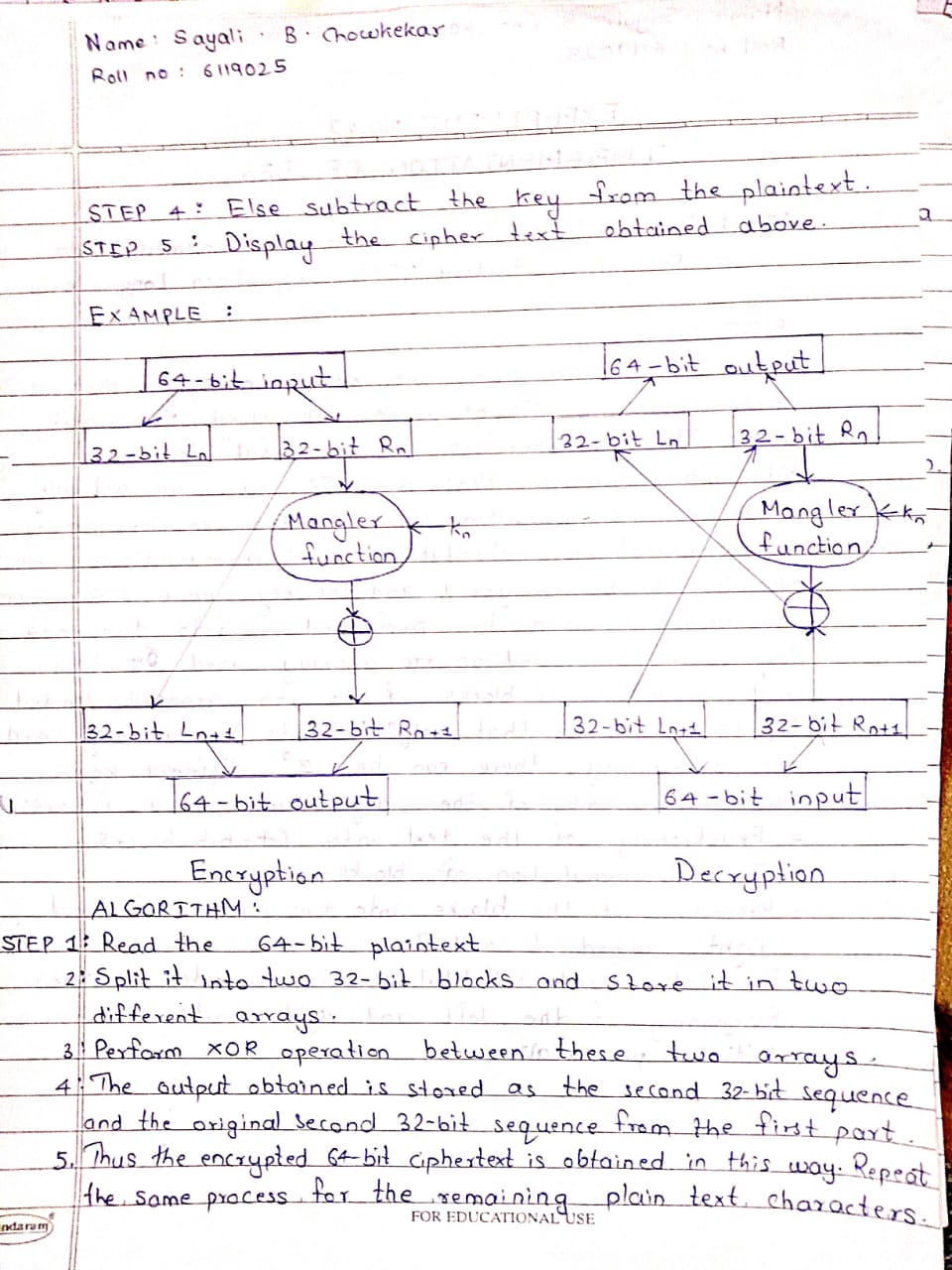
**EXPERIMENT NO: 02**

**AIM:** Encrypt long messages using various modes of operation using DES.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Compiler.

**THEORY:**

****

****

**CODE:**

import java.util.\*;

class Main {

    private static class DES {

        // CONSTANTS

        // Initial Permutation Table

        int[] IP = { 58, 50, 42, 34, 26, 18,

                    10, 2, 60, 52, 44, 36, 28, 20,

                    12, 4, 62, 54, 46, 38,

                    30, 22, 14, 6, 64, 56,

                    48, 40, 32, 24, 16, 8,

                    57, 49, 41, 33, 25, 17,

                    9, 1, 59, 51, 43, 35, 27,

                    19, 11, 3, 61, 53, 45,

                    37, 29, 21, 13, 5, 63, 55,

                    47, 39, 31, 23, 15, 7 };

        // Inverse Initial Permutation Table

        int[] IP1 = { 40, 8, 48, 16, 56, 24, 64,

                    32, 39, 7, 47, 15, 55,

                    23, 63, 31, 38, 6, 46,

                    14, 54, 22, 62, 30, 37,

                    5, 45, 13, 53, 21, 61,

                    29, 36, 4, 44, 12, 52,

                    20, 60, 28, 35, 3, 43,

                    11, 51, 19, 59, 27, 34,

                    2, 42, 10, 50, 18, 58,

                    26, 33, 1, 41, 9, 49,

                    17, 57, 25 };

        // first key-Permutation Table

        int[] PC1 = { 57, 49, 41, 33, 25,

                    17, 9, 1, 58, 50, 42, 34, 26,

                    18, 10, 2, 59, 51, 43, 35, 27,

                    19, 11, 3, 60, 52, 44, 36, 63,

                    55, 47, 39, 31, 23, 15, 7, 62,

                    54, 46, 38, 30, 22, 14, 6, 61,

                    53, 45, 37, 29, 21, 13, 5, 28,

                    20, 12, 4 };

        // second key-Permutation Table

        int[] PC2 = { 14, 17, 11, 24, 1, 5, 3,

                    28, 15, 6, 21, 10, 23, 19, 12,

                    4, 26, 8, 16, 7, 27, 20, 13, 2,

                    41, 52, 31, 37, 47, 55, 30, 40,

                    51, 45, 33, 48, 44, 49, 39, 56,

                    34, 53, 46, 42, 50, 36, 29, 32 };

        // Expansion D-box Table

        int[] EP = { 32, 1, 2, 3, 4, 5, 4,

                    5, 6, 7, 8, 9, 8, 9, 10,

                    11, 12, 13, 12, 13, 14, 15,

                    16, 17, 16, 17, 18, 19, 20,

                    21, 20, 21, 22, 23, 24, 25,

                    24, 25, 26, 27, 28, 29, 28,

                    29, 30, 31, 32, 1 };

        // Straight Permutation Table

        int[] P = { 16, 7, 20, 21, 29, 12, 28,

                    17, 1, 15, 23, 26, 5, 18,

                    31, 10, 2, 8, 24, 14, 32,

                    27, 3, 9, 19, 13, 30, 6,

                    22, 11, 4, 25 };

        // S-box Table

        int[][][] sbox = {

            { { 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7 },

            { 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8 },

            { 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0 },

            { 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 } },

            { { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10 },

            { 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5 },

            { 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15 },

            { 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 } },

            { { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8 },

            { 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1 },

            { 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7 },

            { 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 } },

            { { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15 },

            { 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9 },

            { 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4 },

            { 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 } },

            { { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9 },

            { 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6 },

            { 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14 },

            { 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 } },

            { { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11 },

            { 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8 },

            { 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6 },

            { 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 } },

            { { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1 },

            { 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6 },

            { 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2 },

            { 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 } },

            { { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7 },

            { 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2 },

            { 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8 },

            { 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 } }

        };

        int[] shiftBits = { 1, 1, 2, 2, 2, 2, 2, 2,

                            1, 2, 2, 2, 2, 2, 2, 1 };

        // hexadecimal to binary conversion

        String hextoBin(String input)

        {

            int n = input.length() \* 4;

            input = Long.toBinaryString(

                Long.parseUnsignedLong(input, 16));

            while (input.length() < n)

                input = "0" + input;

            return input;

        }

        // binary to hexadecimal conversion

        String binToHex(String input)

        {

            int n = (int)input.length() / 4;

            input = Long.toHexString(

                Long.parseUnsignedLong(input, 2));

            while (input.length() < n)

                input = "0" + input;

            return input;

        }

        // per-mutate input hexadecimal

        // according to specified sequence

        String permutation(int[] sequence, String input)

        {

            String output = "";

            input = hextoBin(input);

            for (int i = 0; i < sequence.length; i++)

                output += input.charAt(sequence[i] - 1);

            output = binToHex(output);

            return output;

        }

        // xor 2 hexadecimal strings

        String xor(String a, String b)

        {

            // hexadecimal to decimal(base 10)

            long t\_a = Long.parseUnsignedLong(a, 16);

            // hexadecimal to decimal(base 10)

            long t\_b = Long.parseUnsignedLong(b, 16);

            // xor

            t\_a = t\_a ^ t\_b;

            // decimal to hexadecimal

            a = Long.toHexString(t\_a);

            // prepend 0's to maintain length

            while (a.length() < b.length())

                a = "0" + a;

            return a;

        }

        // left Circular Shifting bits

        String leftCircularShift(String input, int numBits)

        {

            int n = input.length() \* 4;

            int perm[] = new int[n];

            for (int i = 0; i < n - 1; i++)

                perm[i] = (i + 2);

            perm[n - 1] = 1;

            while (numBits-- > 0)

                input = permutation(perm, input);

            return input;

        }

        // preparing 16 keys for 16 rounds

        String[] getKeys(String key)

        {

            String keys[] = new String[16];

            // first key permutation

            key = permutation(PC1, key);

            for (int i = 0; i < 16; i++) {

                key = leftCircularShift(

                        key.substring(0, 7), shiftBits[i])

                    + leftCircularShift(key.substring(7, 14),

                                        shiftBits[i]);

                // second key permutation

                keys[i] = permutation(PC2, key);

            }

            return keys;

        }

        // s-box lookup

        String sBox(String input)

        {

            String output = "";

            input = hextoBin(input);

            for (int i = 0; i < 48; i += 6) {

                String temp = input.substring(i, i + 6);

                int num = i / 6;

                int row = Integer.parseInt(

                    temp.charAt(0) + "" + temp.charAt(5), 2);

                int col = Integer.parseInt(

                    temp.substring(1, 5), 2);

                output += Integer.toHexString(

                    sbox[num][row][col]);

            }

            return output;

        }

        String round(String input, String key, int num)

        {

            // fk

            String left = input.substring(0, 8);

            String temp = input.substring(8, 16);

            String right = temp;

            // Expansion permutation

            temp = permutation(EP, temp);

            // xor temp and round key

            temp = xor(temp, key);

            // lookup in s-box table

            temp = sBox(temp);

            // Straight D-box

            temp = permutation(P, temp);

            // xor

            left = xor(left, temp);

            System.out.println("Round "

                            + (num + 1) + " "

                            + right.toUpperCase()

                            + " " + left.toUpperCase() + " "

                            + key.toUpperCase());

            // swapper

            return right + left;

        }

        String encrypt(String plainText, String key)

        {

            int i;

            // get round keys

            String keys[] = getKeys(key);

            // initial permutation

            plainText = permutation(IP, plainText);

            System.out.println(

                "After initial permutation: "

                + plainText.toUpperCase());

            System.out.println(

                "After splitting: L0="

                + plainText.substring(0, 8).toUpperCase()

                + " R0="

                + plainText.substring(8, 16).toUpperCase() + "\n");

            // 16 rounds

            for (i = 0; i < 16; i++) {

                plainText = round(plainText, keys[i], i);

            }

            // 32-bit swap

            plainText = plainText.substring(8, 16)

                        + plainText.substring(0, 8);

            // final permutation

            plainText = permutation(IP1, plainText);

            return plainText;

        }

        String decrypt(String plainText, String key)

        {

            int i;

            // get round keys

            String keys[] = getKeys(key);

            // initial permutation

            plainText = permutation(IP, plainText);

            System.out.println(

                "After initial permutation: "

                + plainText.toUpperCase());

            System.out.println(

                "After splitting: L0="

                + plainText.substring(0, 8).toUpperCase()

                + " R0=" + plainText.substring(8, 16).toUpperCase()

                + "\n");

            // 16-rounds

            for (i = 15; i > -1; i--) {

                plainText = round(plainText, keys[i], 15 - i);

            }

            // 32-bit swap

            plainText = plainText.substring(8, 16)

                        + plainText.substring(0, 8);

            plainText = permutation(IP1, plainText);

            return plainText;

        }

    }

    public static void main(String args[])

    {

        String text = "123456ABCD132536";

        String key = "AABB09182736CCDD";

        DES cipher = new DES();

        System.out.println("Encryption:\n");

        text = cipher.encrypt(text, key);

        System.out.println(

            "\nCipher Text: " + text.toUpperCase() + "\n");

        System.out.println("Decryption\n");

        text = cipher.decrypt(text, key);

        System.out.println(

            "\nPlain Text: "

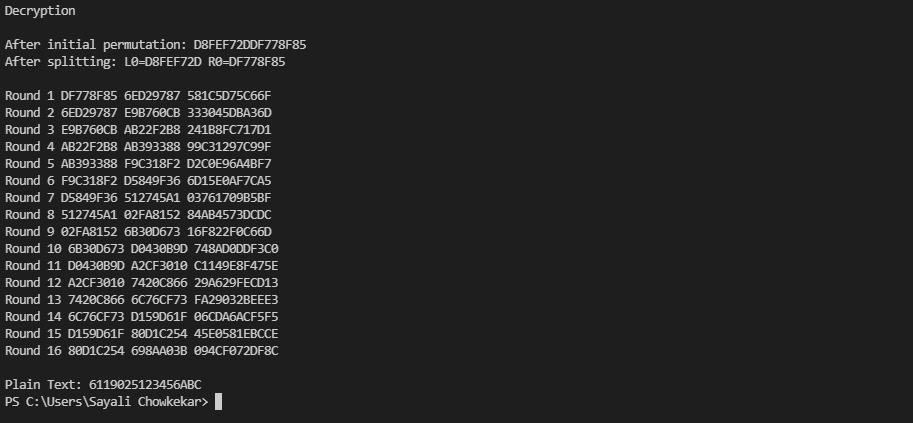
            + text.toUpperCase());

    }

}

**OUTPUT:**





**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** DES implemented successfully.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **03** |
| Title: | **Cryptoanalysis or decoding Playfair, Vigenere Cipher.** |
| Date of Performance: | **11/08/21** |
| Date of Submission: | **25/08/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

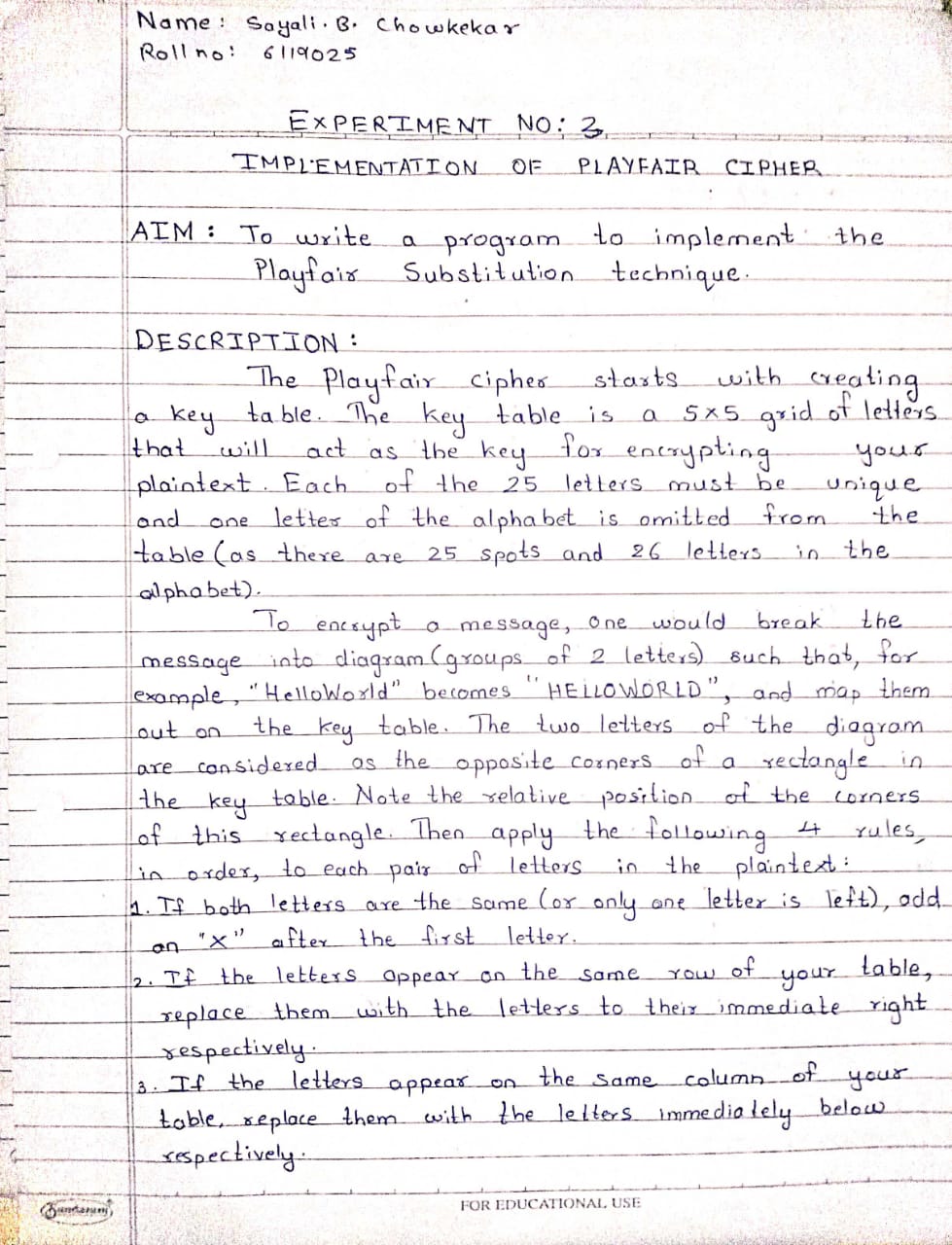
Date:

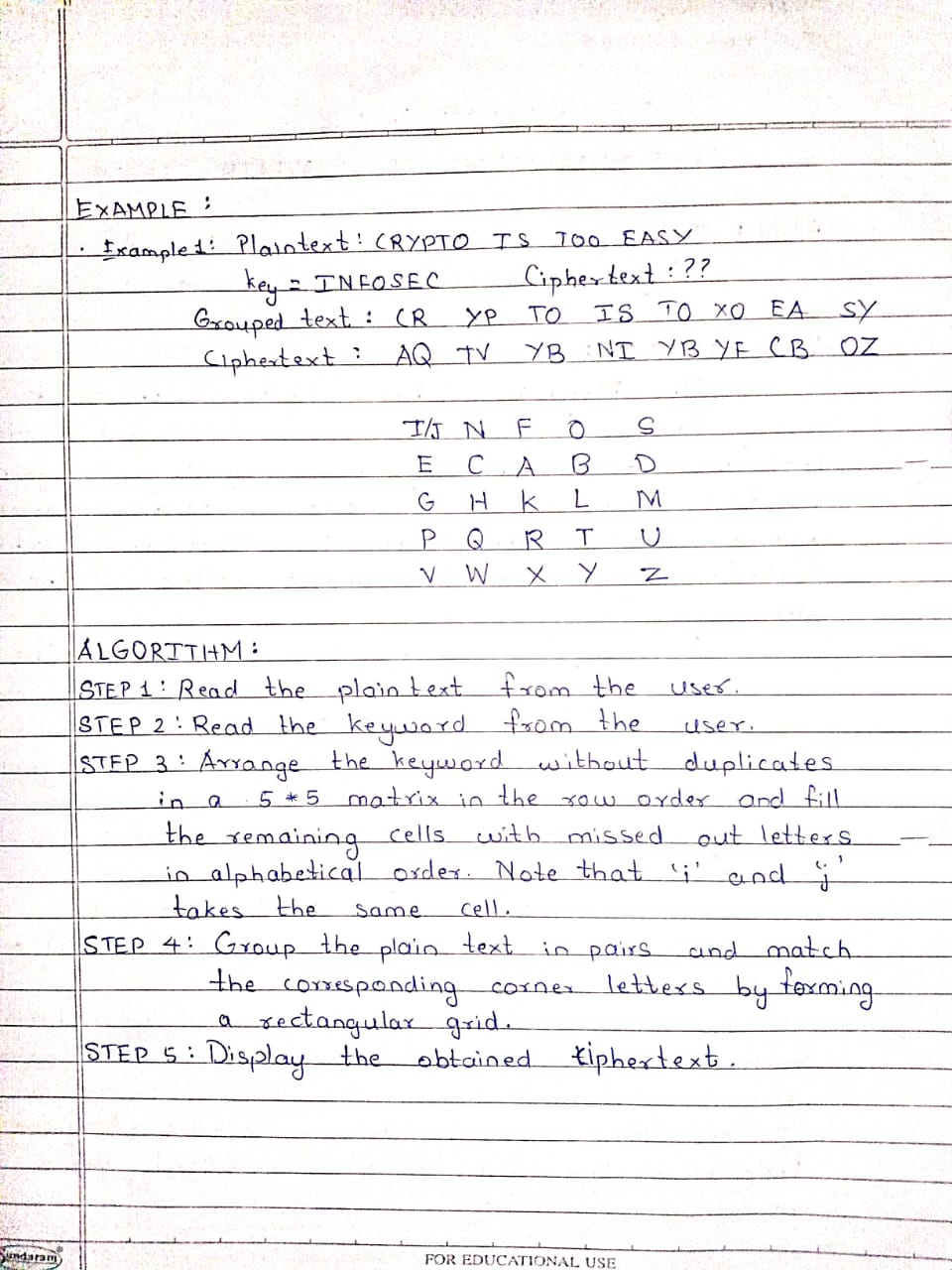
**EXPERIMENT NO: 03**

**AIM:** Cryptoanalysis or decoding Playfair, Vigenere Cipher.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Compiler.

**THEORY:**

****

****

**CODE:**

key=input("Enter key")

key=key.replace(" ", "")

print(key)

key=key.upper()

print(key)

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

for i in range(0,5):

    for j in range(0,5):

        print(my\_matrix[i][j], end=" ")

    print()

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():  #Encryption

    msg=str(input("ENTER MSG:"))

    msg=msg.upper()

    msg=msg.replace(" ", "")

    i=0

    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=' ')

    while i<len(msg):

        loc=list()

        loc=locindex(msg[i])

        loc1=list()

        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=i+2

def decrypt():  #decryption

    msg=str(input("ENTER CIPHER TEXT:"))

    msg=msg.upper()

    msg=msg.replace(" ", "")

    print("PLAIN TEXT:",end=' ')

    i=0

    while i<len(msg):

        loc=list()

        loc=locindex(msg[i])

        loc1=list()

        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=i+2

while(1):

    choice=int(input("\n 1.Encryption \n 2.Decryption: \n 3.EXIT"))

    if choice==1:

        encrypt()

    elif choice==2:

        decrypt()

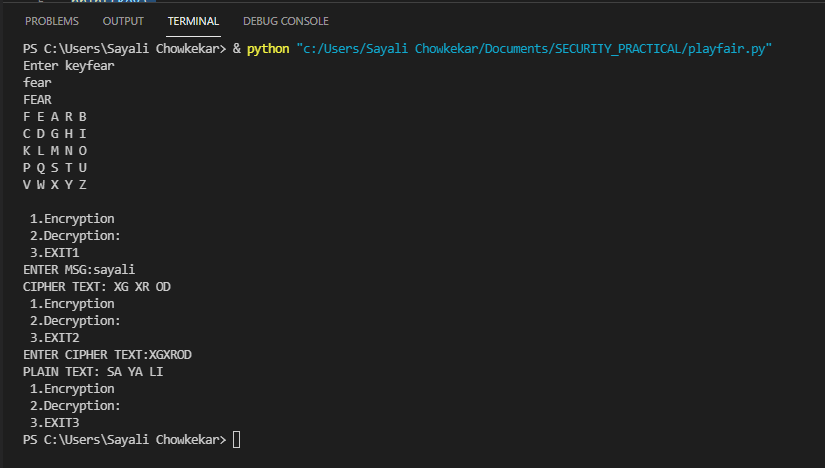
    elif choice==3:

        exit()

    else:

        print("Choose correct choice")

**OUTPUT:**



**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Successfully implemented the Playfair cipher’s encryption & decryption.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **04** |
| Title: | **Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.** |
| Date of Performance: | **25/08/21** |
| Date of Submission: | **25/08/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

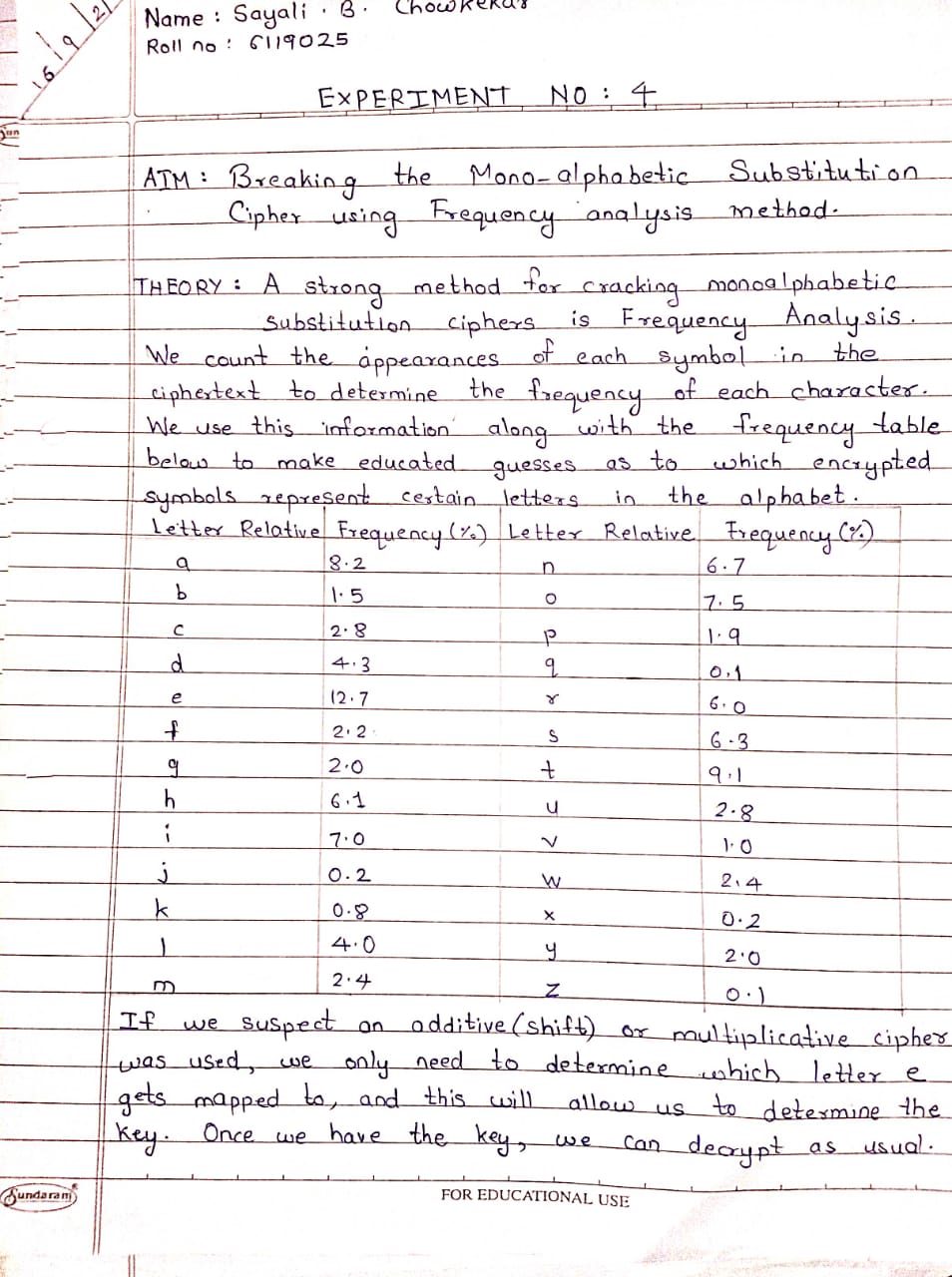
Date:

**EXPERIMENT NO: 04**

**AIM:** Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Compiler.

**THEORY:**

****

**CODE:**

import java.lang.\*;

import java.util.\*;

public class monoalpha {

public static void main(String args[]) {

String cipher = "UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZVUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSXEPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ";

String alph = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", key = "ETAOINSHRDLCUMWFGYPBVKJXQZ";

char a[] = new char[26], K[] = new char[26], temp;

int i, j, t, freq[] = new int[26];

float freqp[] = new float[26], tem;

Scanner sc = new Scanner(System.in);

System.out.println("Cipher Text: " + cipher);

for (i = 0; i < 26; i++) {

a[i] = alph.charAt(i);

K[i] = key.charAt(i);

}

for (i = 0; i < cipher.length(); i++) {

freq[((int) cipher.charAt(i)) - 65]++;

freqp[((int) cipher.charAt(i)) - 65] = freq[((int) cipher.charAt(i)) - 65] / (float) cipher.length() \* 100;

}

System.out.println("Alphabet\tNo of occurences\tFrequency");

for (i = 0; i < 26; i++)

System.out.println(a[i] + "\t\t\t" + freq[i] + "\t\t" + freqp[i]);

System.out.println("\nSorting list most to least frequency\n");

for (i = 0; i < 25; i++) {

for (j = i + 1; j < 26; j++) {

if (freqp[i] < freqp[j]) {

tem = freqp[i];

freqp[i] = freqp[j];

freqp[j] = tem;

temp = a[i];

a[i] = a[j];

a[j] = temp;

t = freq[i];

freq[i] = freq[j];

freqp[j] = t;

}

}

}

System.out.println("Alphabet\tNo of occurances\tFrequency");

for (i = 0; i < 26; i++)

System.out.println(a[i] + "\t\t\t" + freq[i] + "\t\t" + freqp[i]);

String decipher = "";

char c1, c2, ch = 'y';

System.out

.println("\nAssuming frequency of occurances matches with the frequency ofoccurances from the graph\n");

System.out.println("Alphabet\tChanged to");

for (i = 0; i < 26; i++) {

System.out.println(a[i] + "\t\t\t" + K[i]);

}

for (i = 0; i < cipher.length(); i++) {

for (j = 0; j < 26; j++) {

if (a[j] == cipher.charAt(i))

decipher += K[j];

}

}

System.out.println("Decrypted Text: " + decipher);

StringBuilder sb = new StringBuilder(decipher);

System.out.print("\nAre you satisfied with the decryption or would you like to continue?(y/n)");

ch = sc.next().charAt(0);

while (ch == 'y' || ch == 'Y') {

System.out.println("\nEnter the character to be replace and character it is to be replaced by");

c1 = sc.next().charAt(0);

c2 = sc.next().charAt(0);

for (i = 0; i < decipher.length(); i++) {

if (c1 == sb.charAt(i))

sb.setCharAt(i, c2);

else if (c2 == sb.charAt(i))

sb.setCharAt(i, c1);

}

System.out.println("Decrypted Text: " + sb);

System.out.print("\nAre you satisfied with the decryption or would you like to continue?(y/n)");

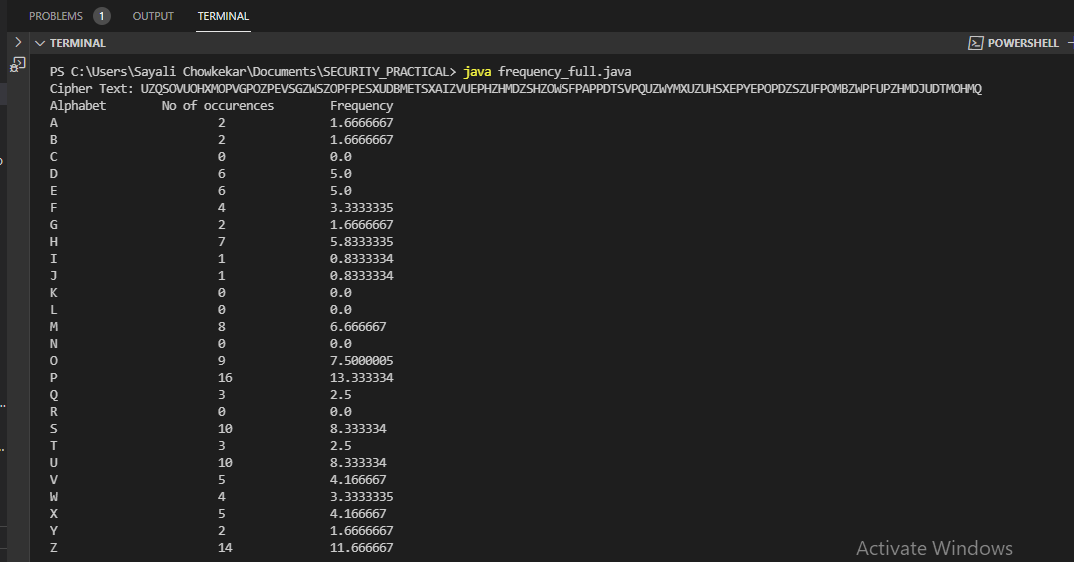
ch = sc.next().charAt(0);

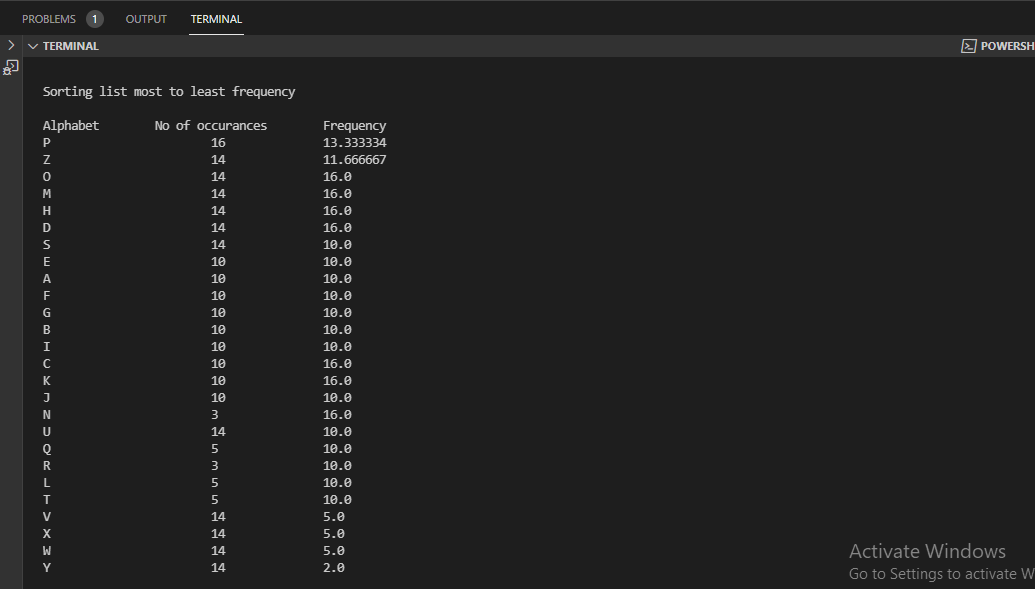
}

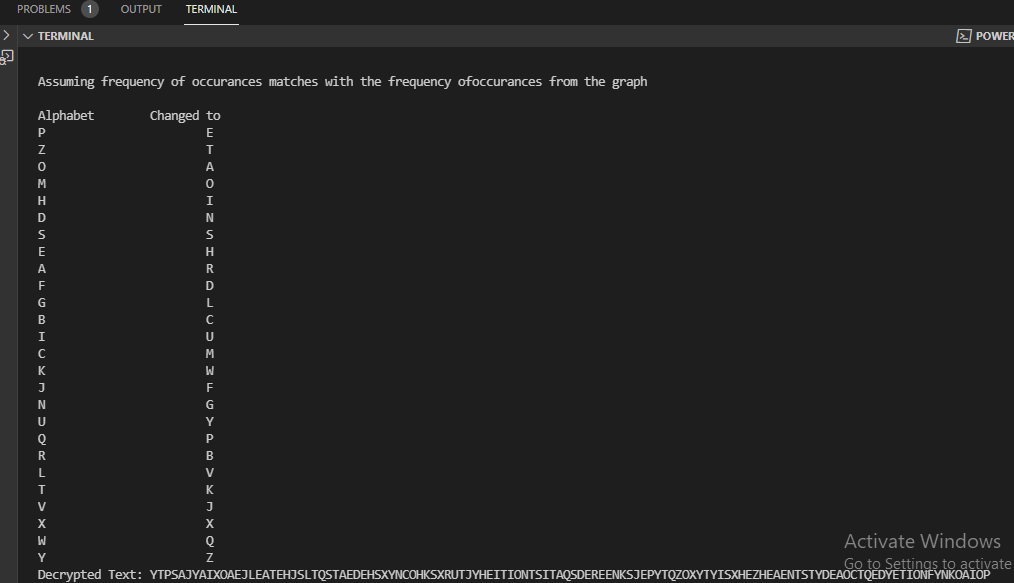
}

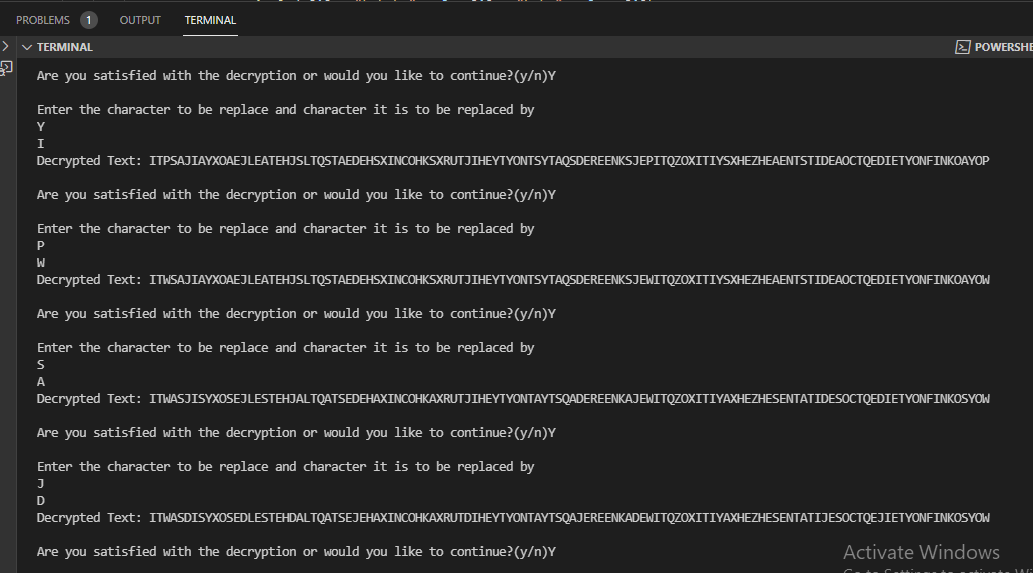
}

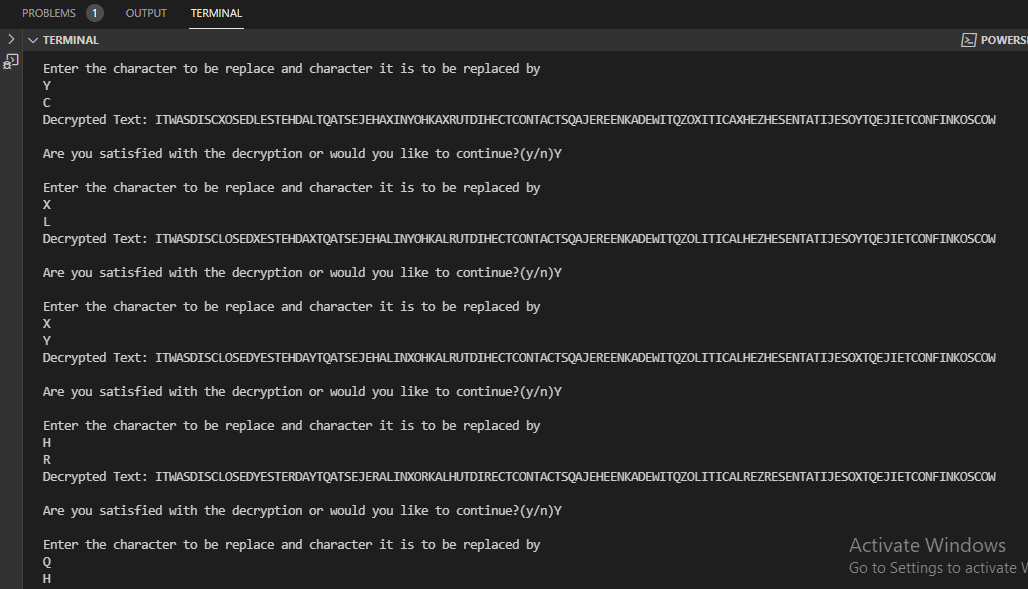
**OUTPUT:**

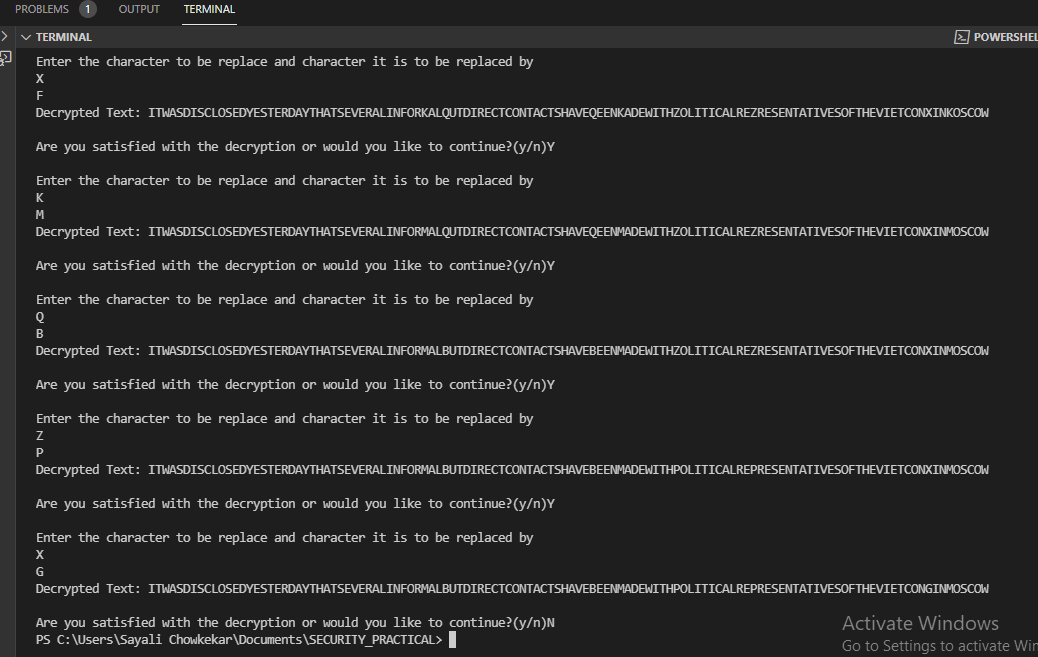
****

****

****

****

****

****

**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Frequency analysis method implemented successfully.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **05** |
| Title: | **Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.** |
| Date of Performance: | **25/08/21** |
| Date of Submission: | **01/09/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

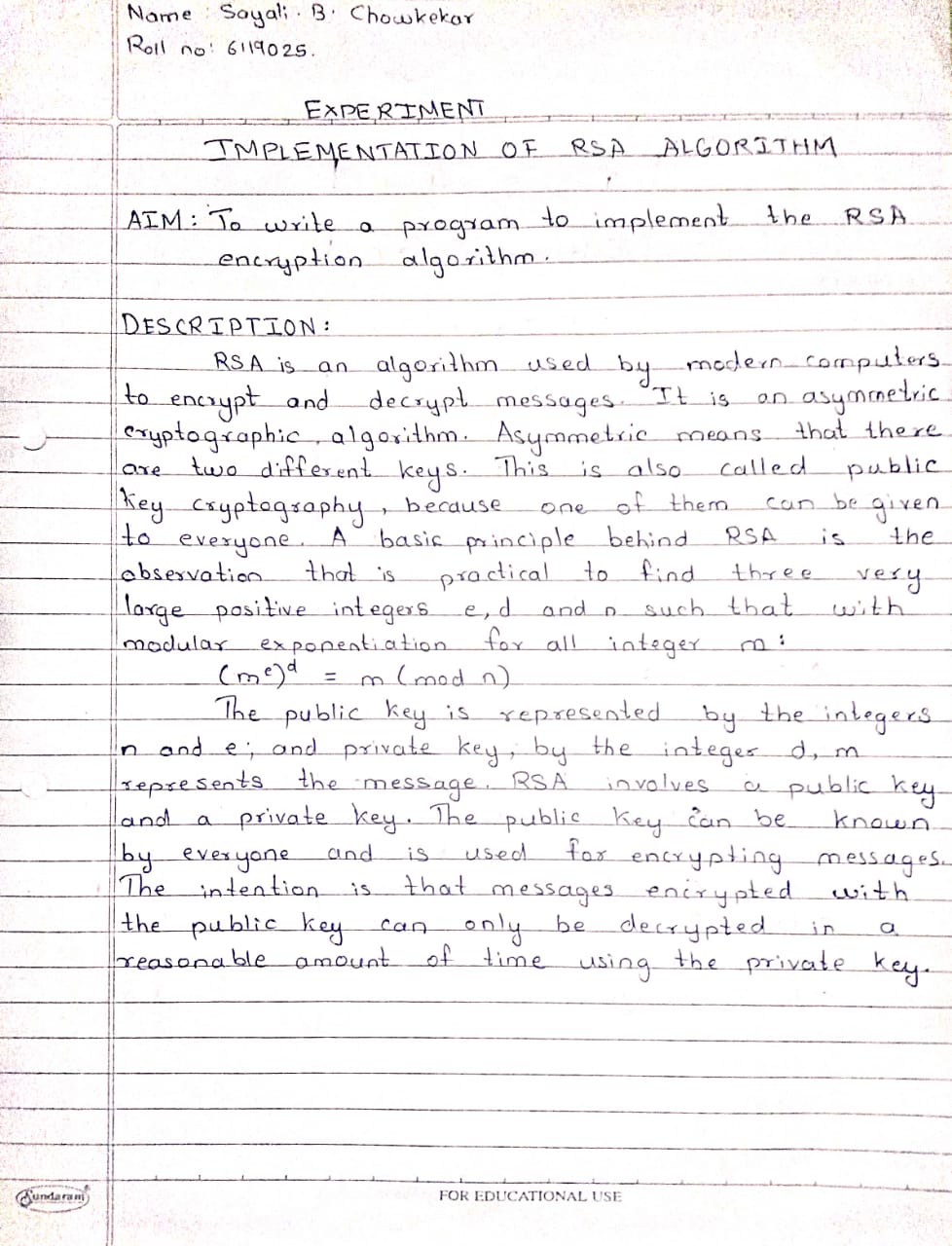
Date:

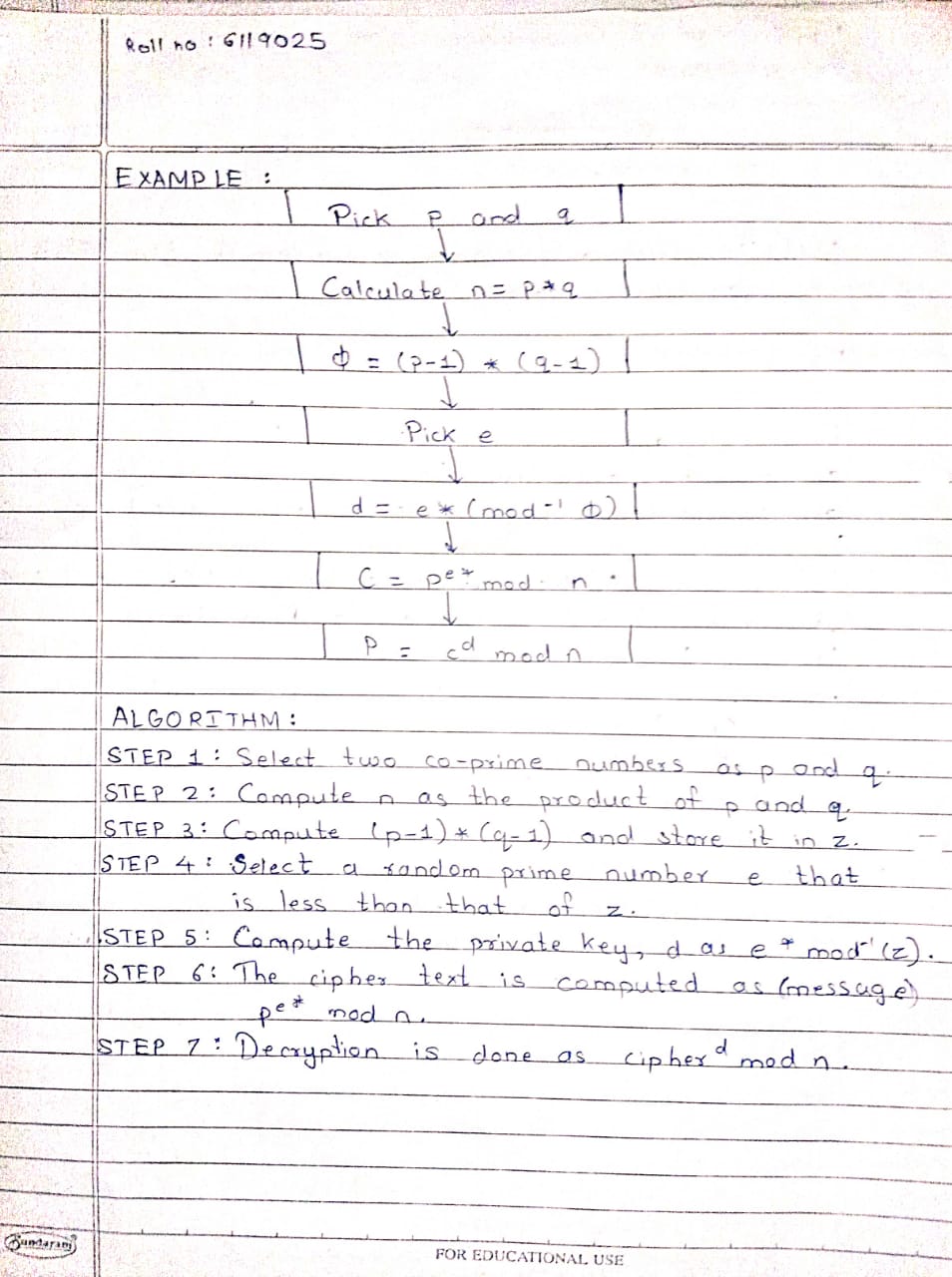
**EXPERIMENT NO: 05**

**AIM:** Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Compiler.

**THEORY:**





**CODE:**

import math

#Input Prime Numbers

p = int(input("Enter a prime number for p: "))

q = int(input("Enter a prime number for q: "))

#Check if Input's are Prime

'''THIS FUNCTION AND THE CODE IMMEDIATELY BELOW THE FUNCTION CHECKS WHETHER THE INPUTS ARE PRIME OR NOT.'''

def prime\_check(a):

if(a==2):

return True

elif((a<2) or ((a%2)==0)):

return False

elif(a>2):

for i in range(2,a):

if not(a%i):

return False

return True

check\_p = prime\_check(p)

check\_q = prime\_check(q)

while(((check\_p==False)or(check\_q==False))):

p = int(input("Enter a prime number for p: "))

q = int(input("Enter a prime number for q: "))

check\_p = prime\_check(p)

check\_q = prime\_check(q)

#RSA Modulus

'''CALCULATION OF RSA MODULUS 'n'.'''

n = p \* q

print("RSA Modulus(n) is:",n)

#Eulers Toitent

'''CALCULATION OF EULERS TOITENT 'r'.'''

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

print("Eulers Toitent(r) is:",r)

#GCD

'''CALCULATION OF GCD FOR 'e' CALCULATION.'''

def egcd(e,r):

while(r!=0):

e,r=r,e%r

return e

#Euclid's Algorithm

def eugcd(e,r):

for i in range(1,r):

while(e!=0):

a,b=r//e,r%e

if(b!=0):

print("%d = %d\*(%d) + %d"%(r,a,e,b))

r=e

e=b

#Extended Euclidean Algorithm

def eea(a,b):

if(a%b==0):

return(b,0,1)

else:

gcd,s,t = eea(b,a%b)

s = s-((a//b) \* t)

print("%d = %d\*(%d) + (%d)\*(%d)"%(gcd,a,t,s,b))

return(gcd,t,s)

#Multiplicative Inverse

def mult\_inv(e,r):

gcd,s,\_=eea(e,r)

if(gcd!=1):

return None

else:

if(s<0):

print("s=%d. Since %d is less than 0, s = s(modr), i.e., s=%d."%(s,s,s%r))

elif(s>0):

print("s=%d."%(s))

return s%r

#e Value Calculation

'''FINDS THE HIGHEST POSSIBLE VALUE OF 'e' BETWEEN 1 and 1000 THAT MAKES (e,r) COPRIME.'''

for i in range(1,1000):

if(egcd(i,r)==1):

e=i

print("The value of e is:",e)

#d, Private and Public Keys

'''CALCULATION OF 'd', PRIVATE KEY, AND PUBLIC KEY.'''

print("EUCLID'S ALGORITHM:")

eugcd(e,r)

print("END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.")

print("EUCLID'S EXTENDED ALGORITHM:")

d = mult\_inv(e,r)

print("END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.")

print("The value of d is:",d)

public = (e,n)

private = (d,n)

print("Private Key is:",private)

print("Public Key is:",public)

#Encryption

'''ENCRYPTION ALGORITHM.'''

def encrypt(pub\_key,n\_text):

e,n=pub\_key

x=[]

m=0

for i in n\_text:

if(i.isupper()):

m = ord(i)-65

c=(m\*\*e)%n

x.append(c)

elif(i.islower()):

m= ord(i)-97

c=(m\*\*e)%n

x.append(c)

elif(i.isspace()):

spc=400

x.append(400)

return x

#Decryption

'''DECRYPTION ALGORITHM'''

def decrypt(priv\_key,c\_text):

d,n=priv\_key

txt=c\_text.split(',')

x=''

m=0

for i in txt:

if(i=='400'):

x+=' '

else:

m=(int(i)\*\*d)%n

m+=65

c=chr(m)

x+=c

return x

#Message

while True:

choose = input("Type '1' for encryption and '2' for decrytion and '3' for exit.")

if(choose!='1' and choose!='2'):

break

message = input("Enter the message you want to decrypt or encrypt:")

print("Your message is:",message)

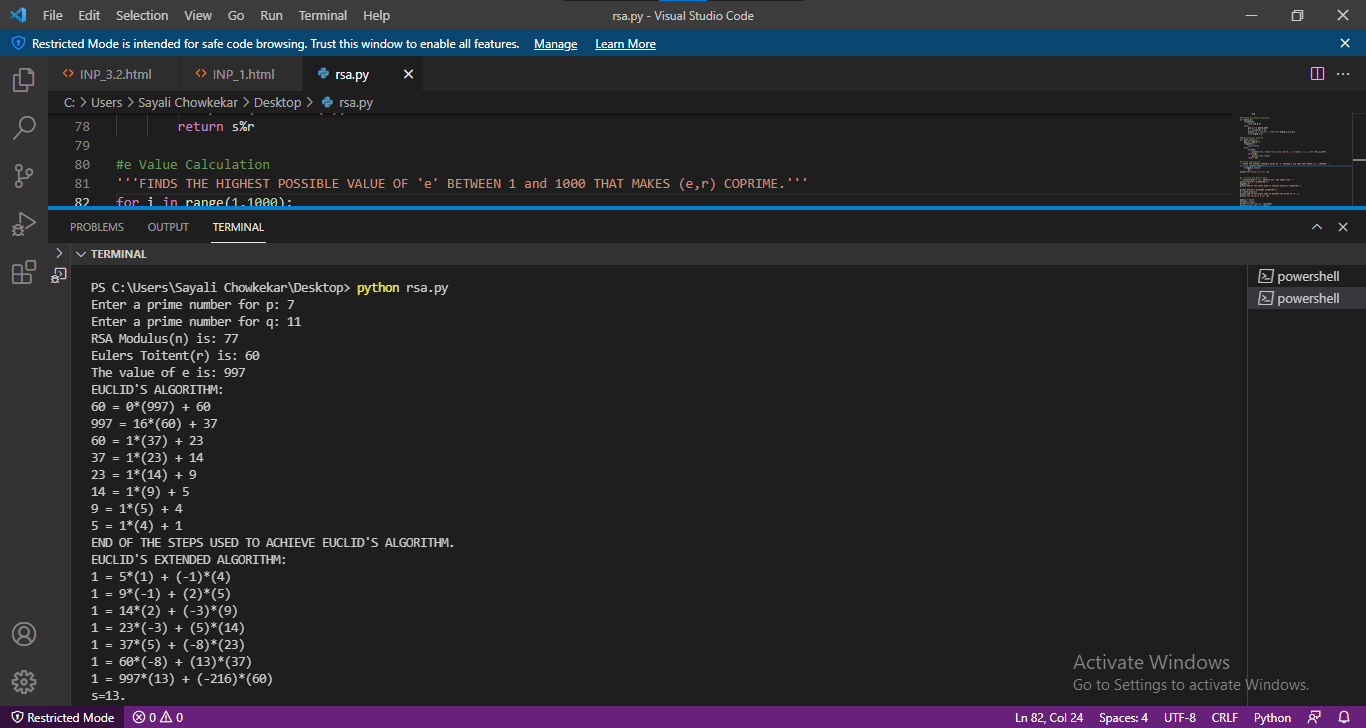
if(choose=='1'):

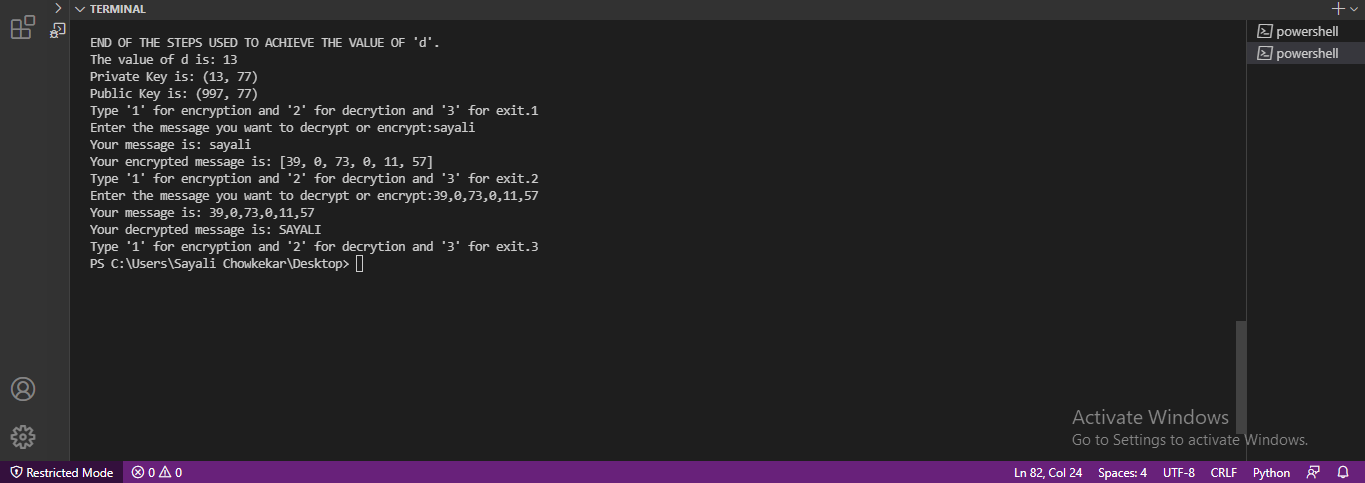
enc\_msg=print("Your encrypted message is:",encrypt(public,message))

elif(choose=='2'):

print("Your decrypted message is:",decrypt(private,message))

**OUTPUT:**





**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** RSA cryptosystem implemented successfully.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **06** |
| Title: | **Cryptographic Hash Functions and Applications.** |
| Date of Performance: | **01/09/21** |
| Date of Submission: | **08/09/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

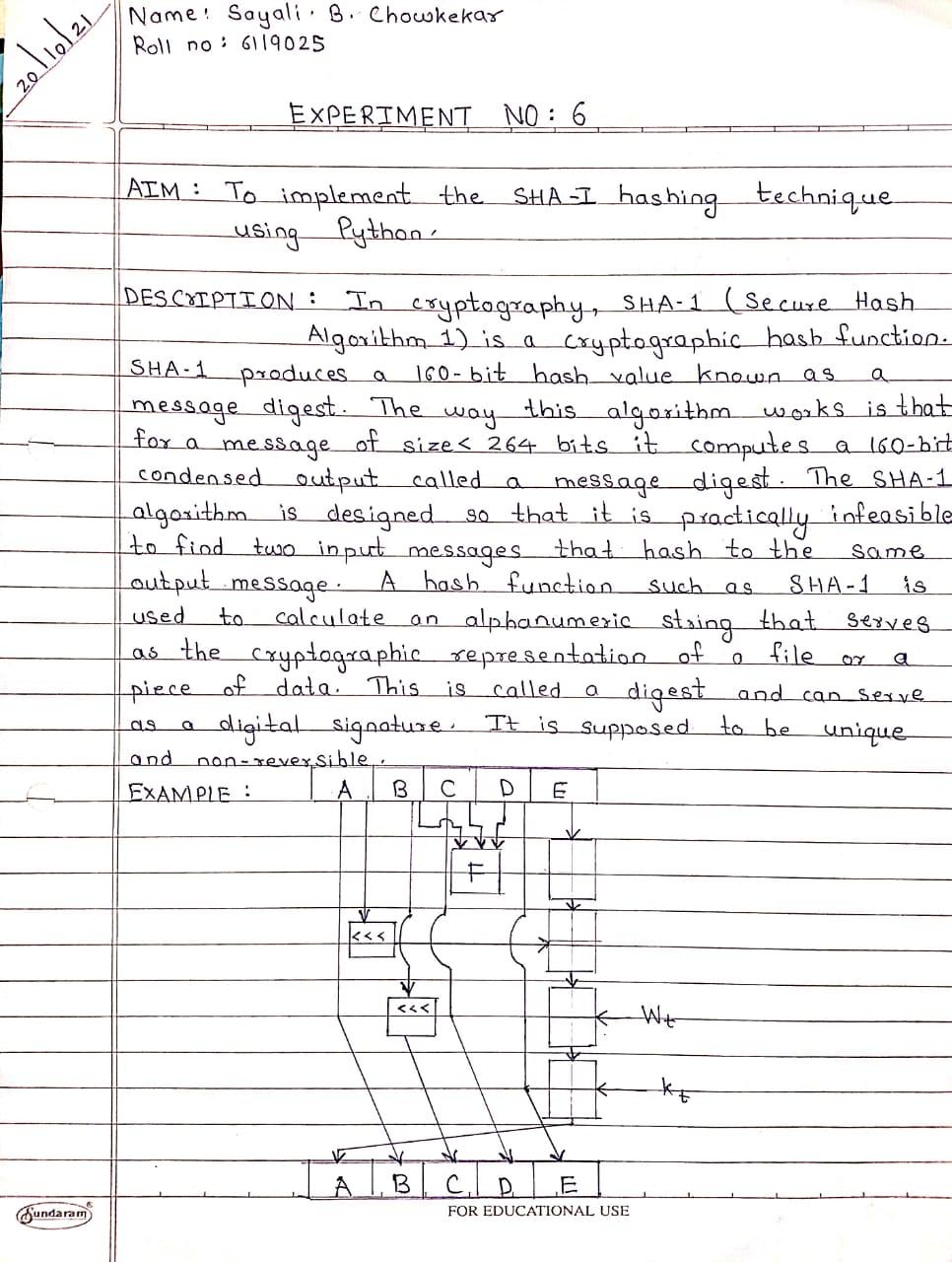
Date:

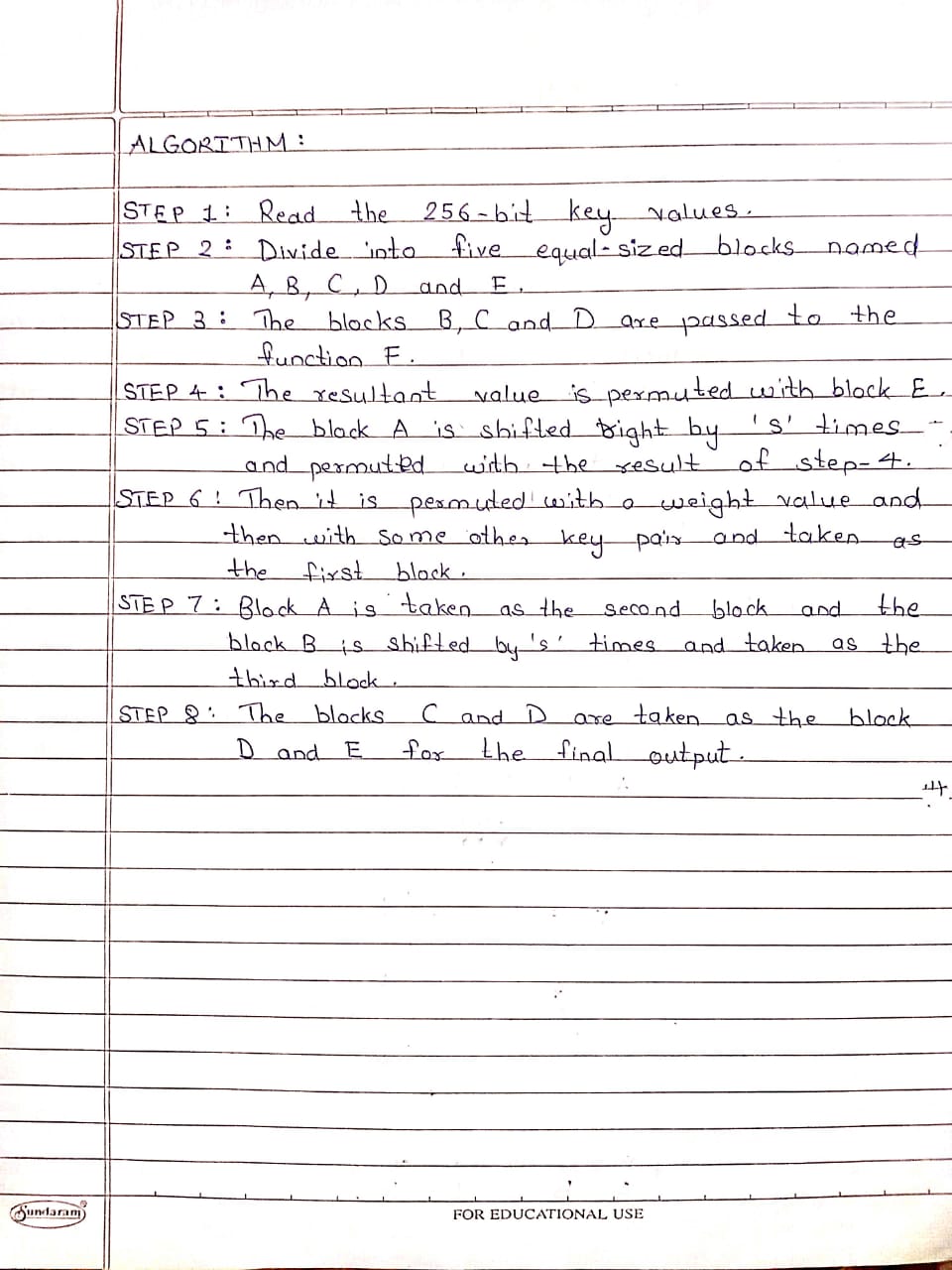
**EXPERIMENT NO: 06**

**AIM:** Cryptographic Hash Functions and Applications.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Compiler.

**THEORY:**

****

****

**CODE:**

**import java.security.\*;**

**public class SHA1 {**

**public static void main(String[] a) {**

**try{**

**MessageDigest md = MessageDigest.getInstance("SHA1");**

**System.out.println("Message digest object info: ");**

**System.out.println("Algorithm = "+md.getAlgorithm());**

**System.out.println("Provider = "+md.getProvider());**

**System.out.println("ToString = "+md.toString());**

**String input = "";**

**md.update(input.getBytes());**

**byte[] output = md.digest();**

**System.out.println("SHA1(\""+input+"\") = "+byteToHex(output));**

**input = "abc";**

**md.update(input.getBytes());**

**output = md.digest();**

**System.out.println();**

**System.out.println("SHA1(\""+input+"\") = "+byteToHex(output));**

**input = "abcdefghijklmnopqrstuvwxyz";**

**md.update(input.getBytes());**

**output = md.digest();**

**System.out.println();**

**System.out.println("SHA1(\""+input+"\") = "+byteToHex(output));**

**System.out.println("");**

**}**

**catch (Exception e) {**

**System.out.println("Exception: "+e);**

**}**

**}**

**public static String byteToHex(byte[] b)**

**{**

**char hexDigit[]= {'0', '1', '2', '3', '4','5','6','7','8','9','A','B','C','D','E','F'};**

**StringBuffer buf = new StringBuffer();**

**for(int j=0; j<b.length; j++) {**

**buf.append(hexDigit[(b[j] >>4) & 0x0f]);**

**buf.append(hexDigit[b[j] & 0x0f]);**

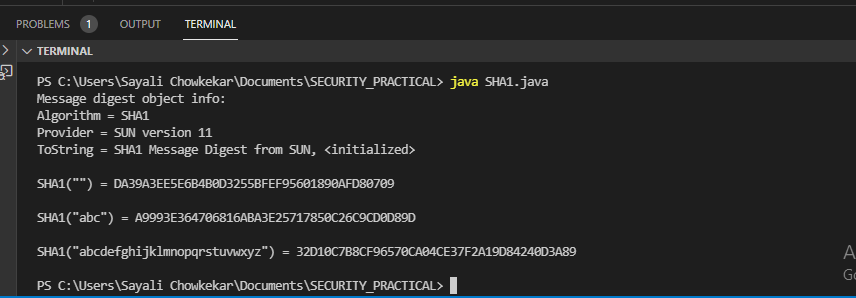
**}**

**return buf.toString();**

**}**

**}**

**OUTPUT:**

****

**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Cryptographic functions & its applications implemented successfully.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **07** |
| Title: | **Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.** |
| Date of Performance: | **08/09/21** |
| Date of Submission: | **15/09/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

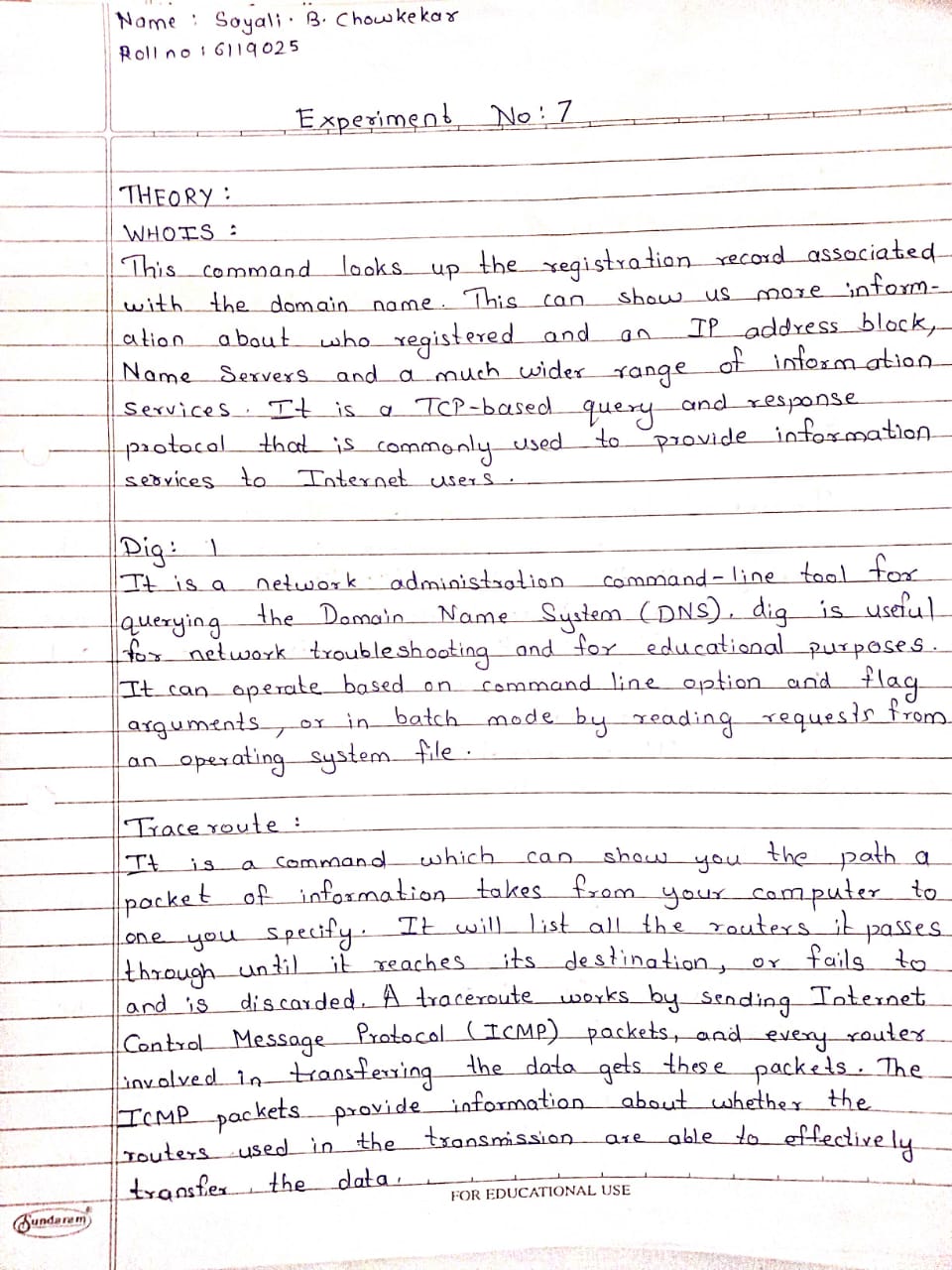
Date:

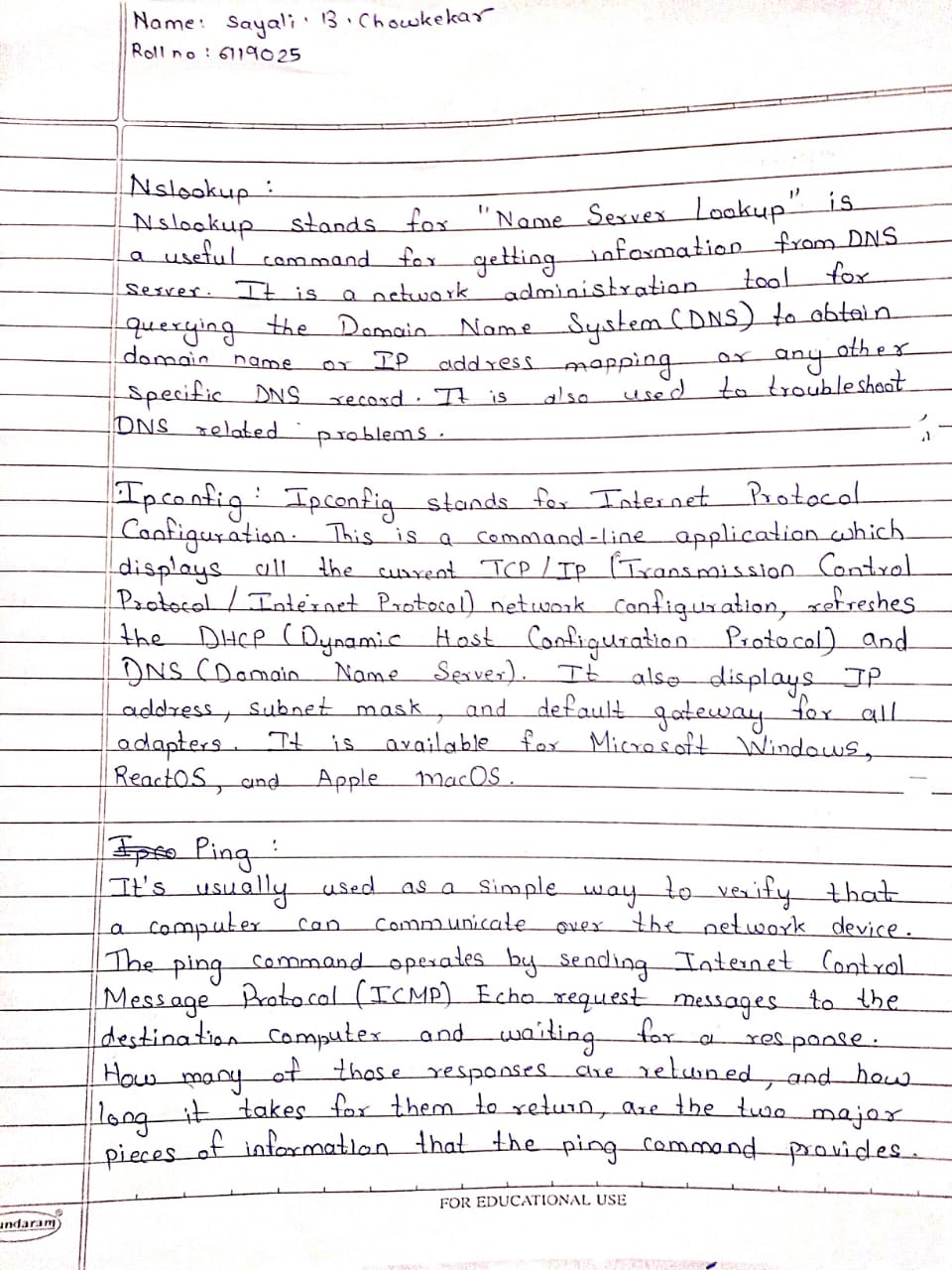
**EXPERIMENT NO: 07**

**AIM:** Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, CMD.

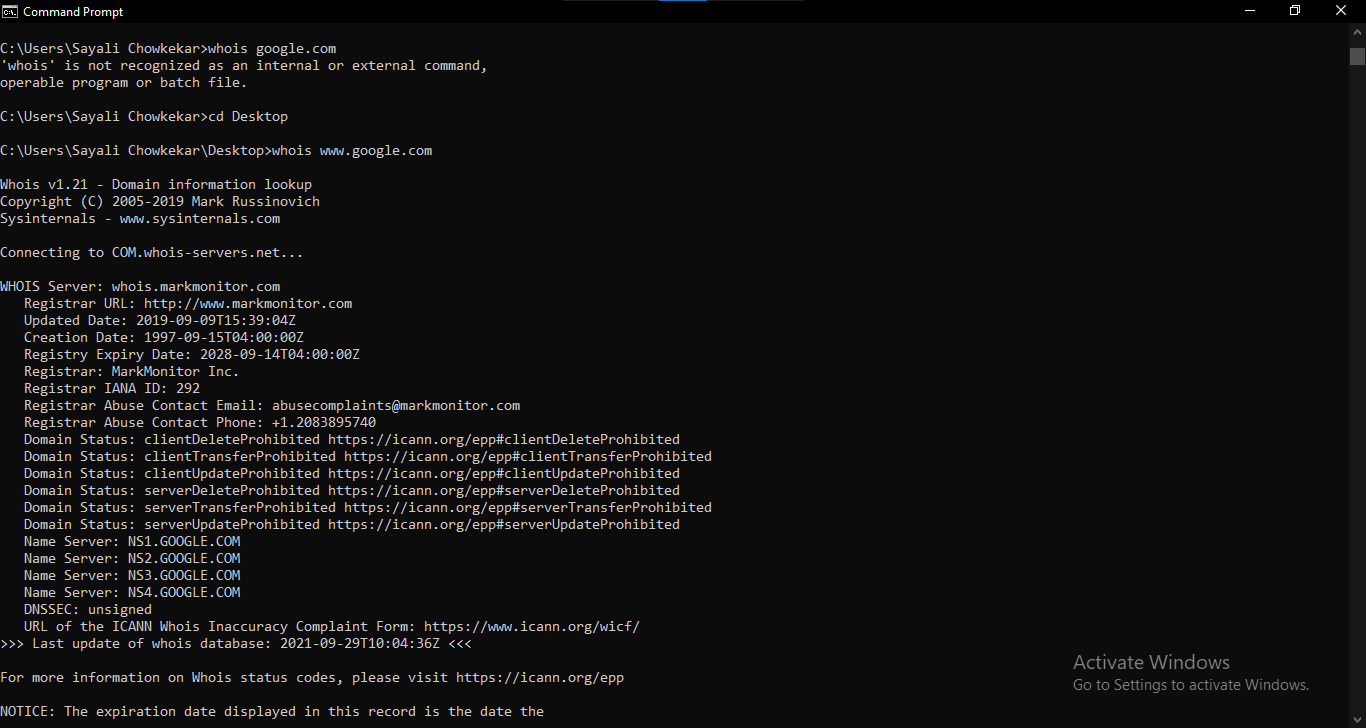
**THEORY:**

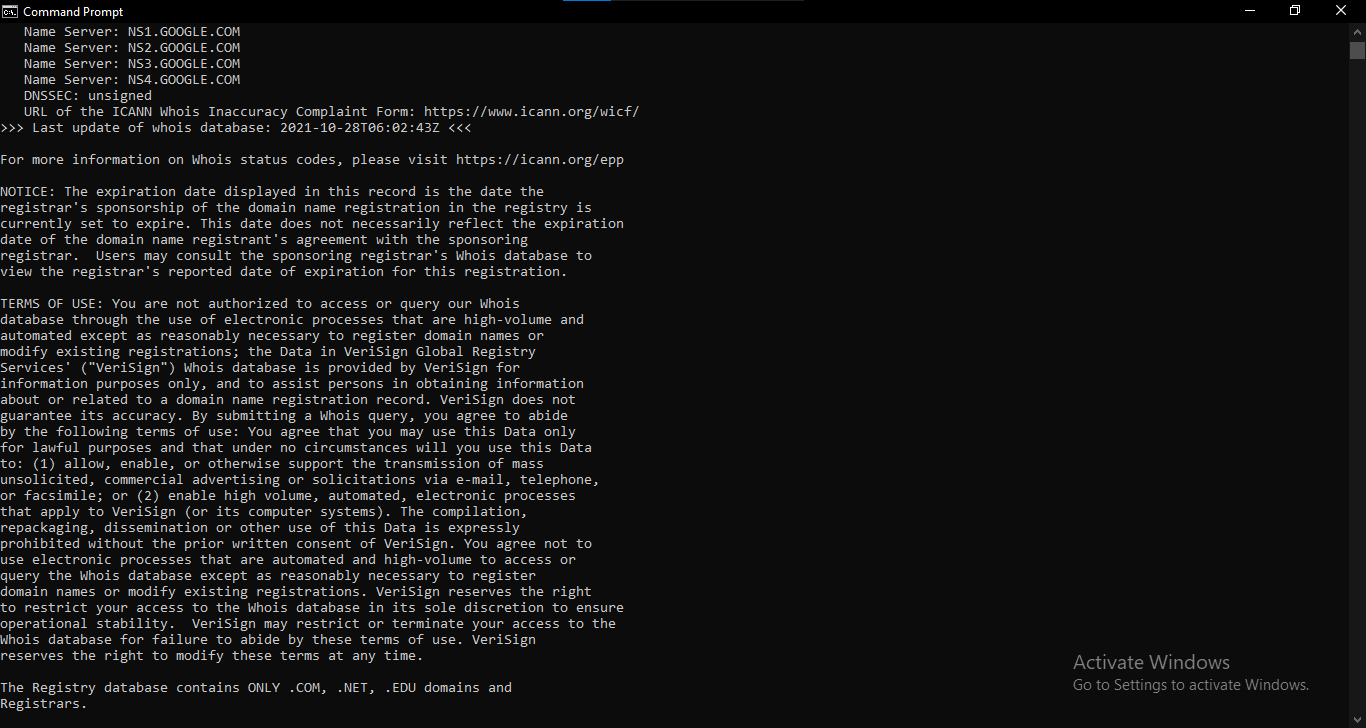
****

****

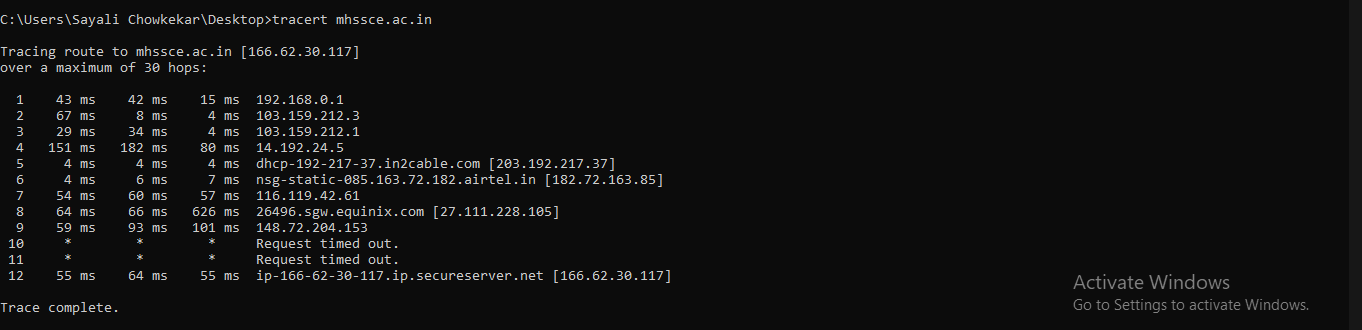
**COMMAND & OUTPUT:**

**Whois:**

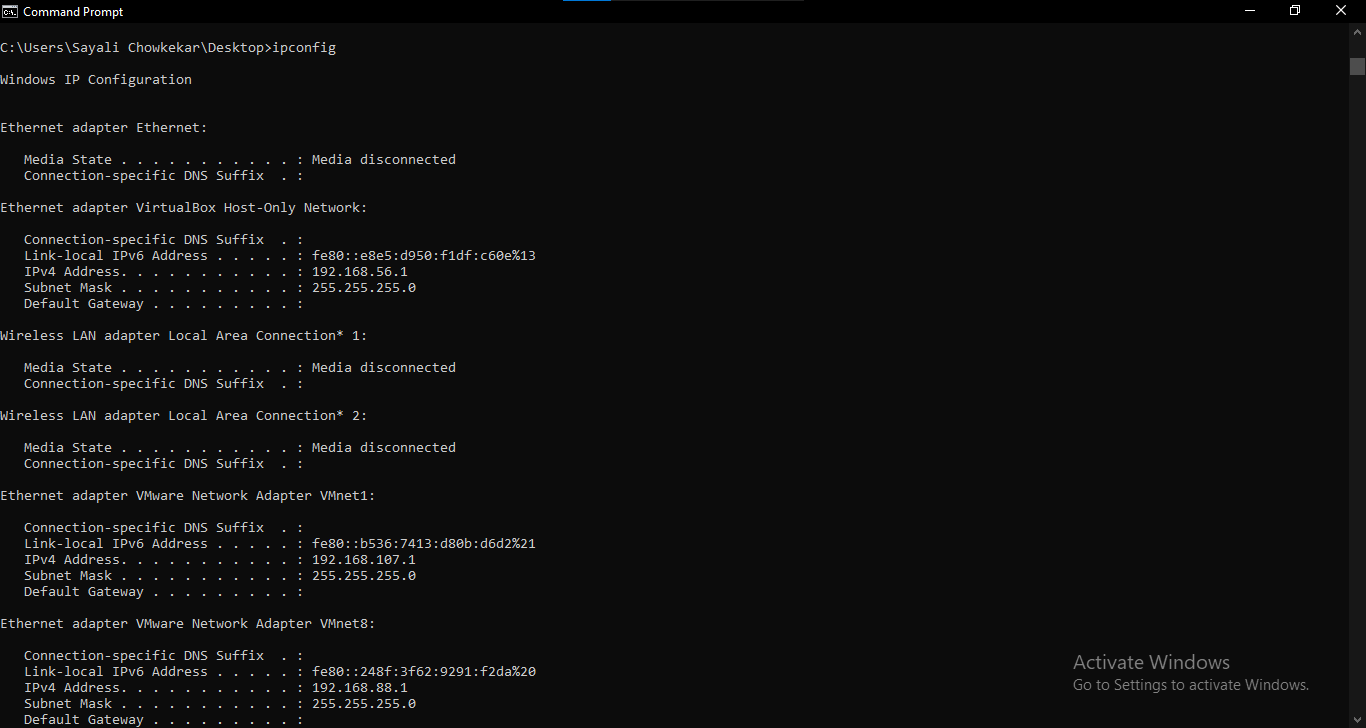
****

****

**Tracert:**

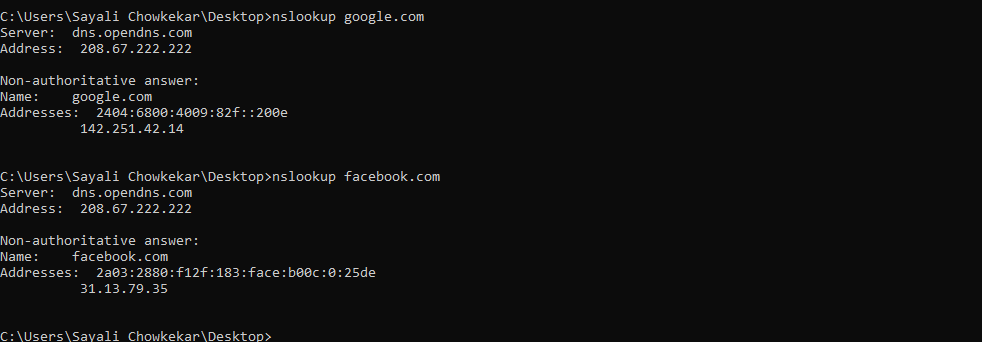
****

**Ipconfig:**

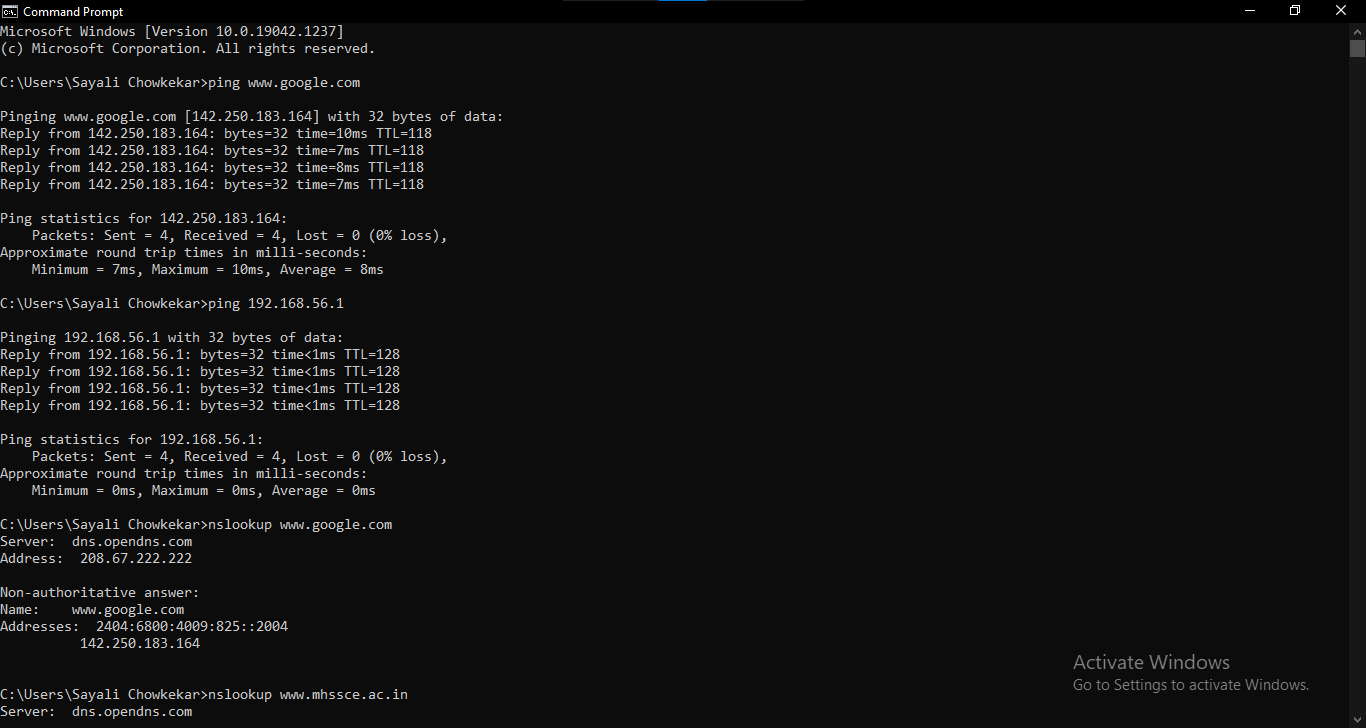
****

****

**Nslookup:**

****

**Ping:**

****

**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Successfully studied the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks & domain registrars.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **08** |
| Title: | **Study of packet sniffer tools Wireshark:**  **a) Observer performance in promiscuous as well as non -promiscuous mode.**  **b) Show the packets can be traced based on different filters** |
| Date of Performance: | **15/09/21** |
| Date of Submission: | **29/09/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

Date:

**EXPERIMENT NO: 08**

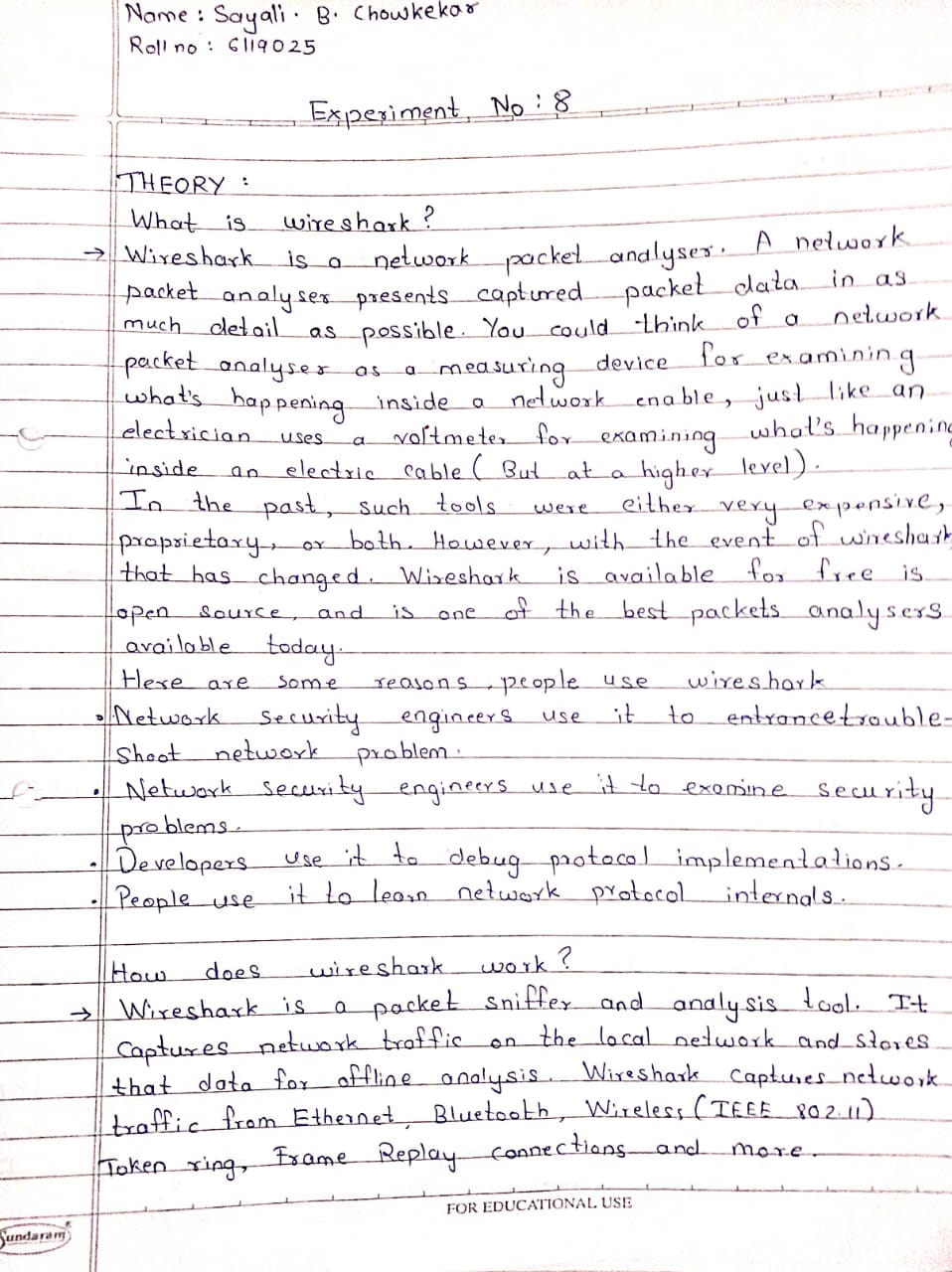
**AIM:** Study of packet sniffer tools Wireshark:

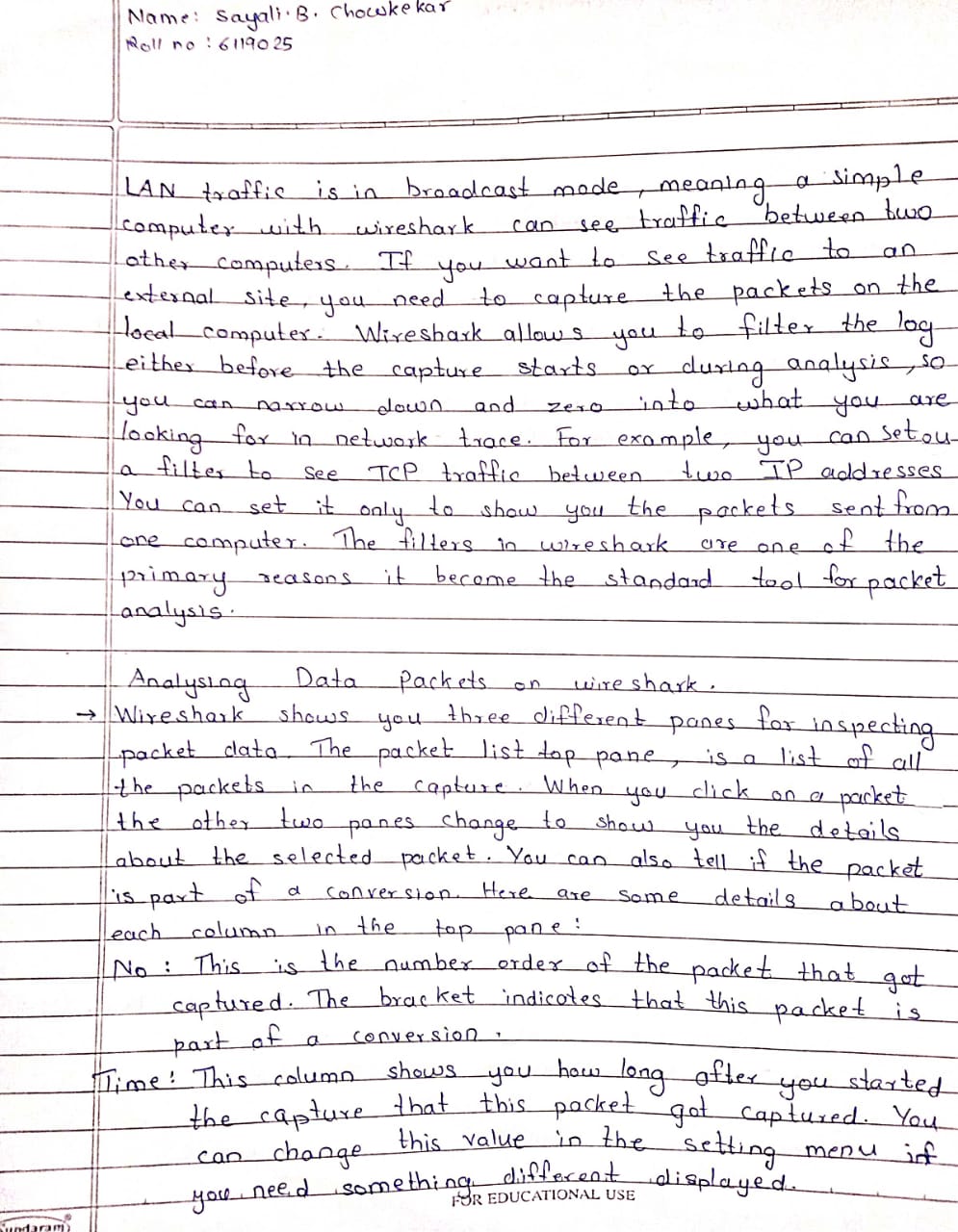
a) Observer performance in promiscuous as well as non -promiscuous mode.

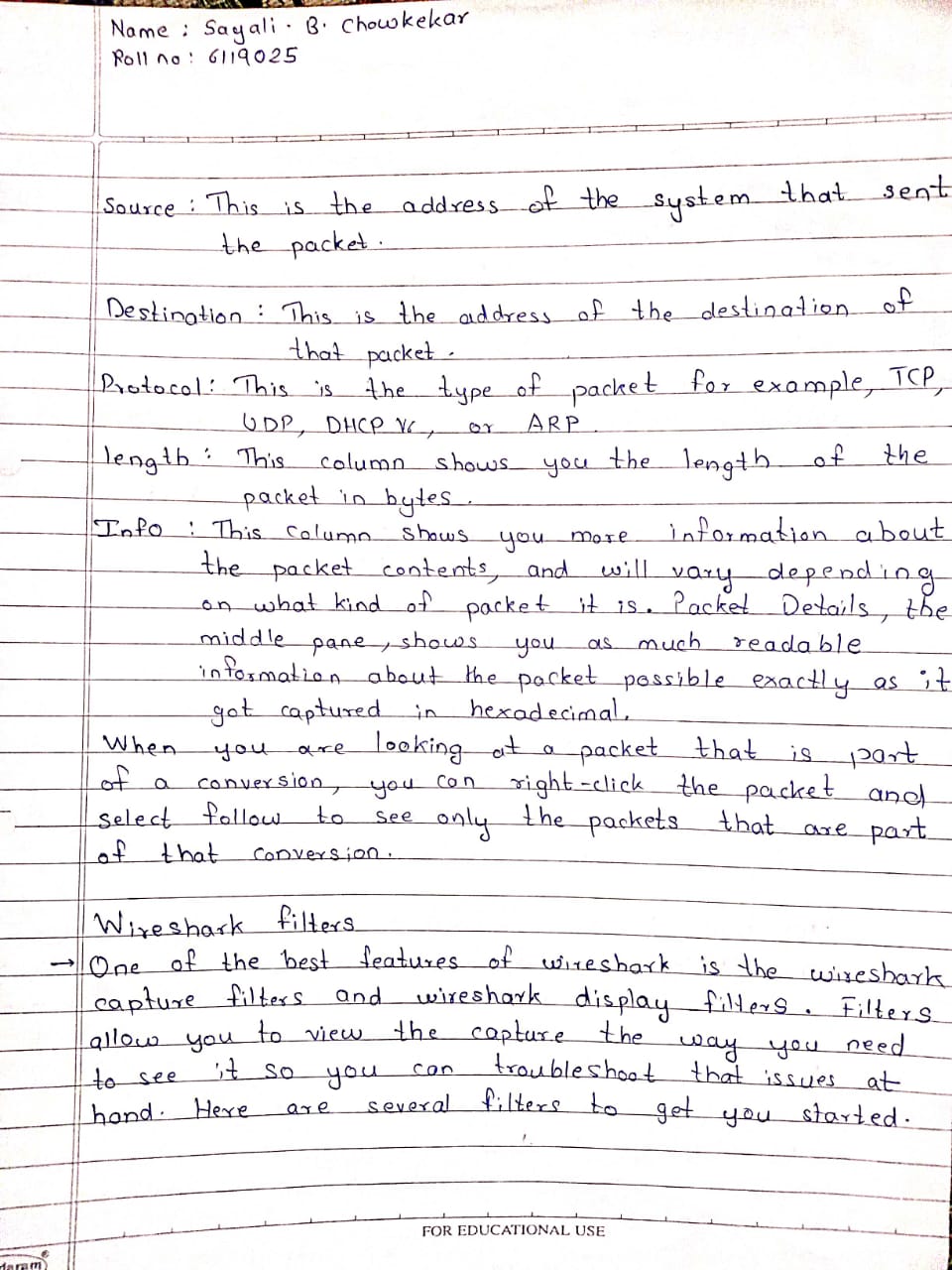
b) Show the packets can be traced based on different filters

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Wireshark installed.

**THEORY:**

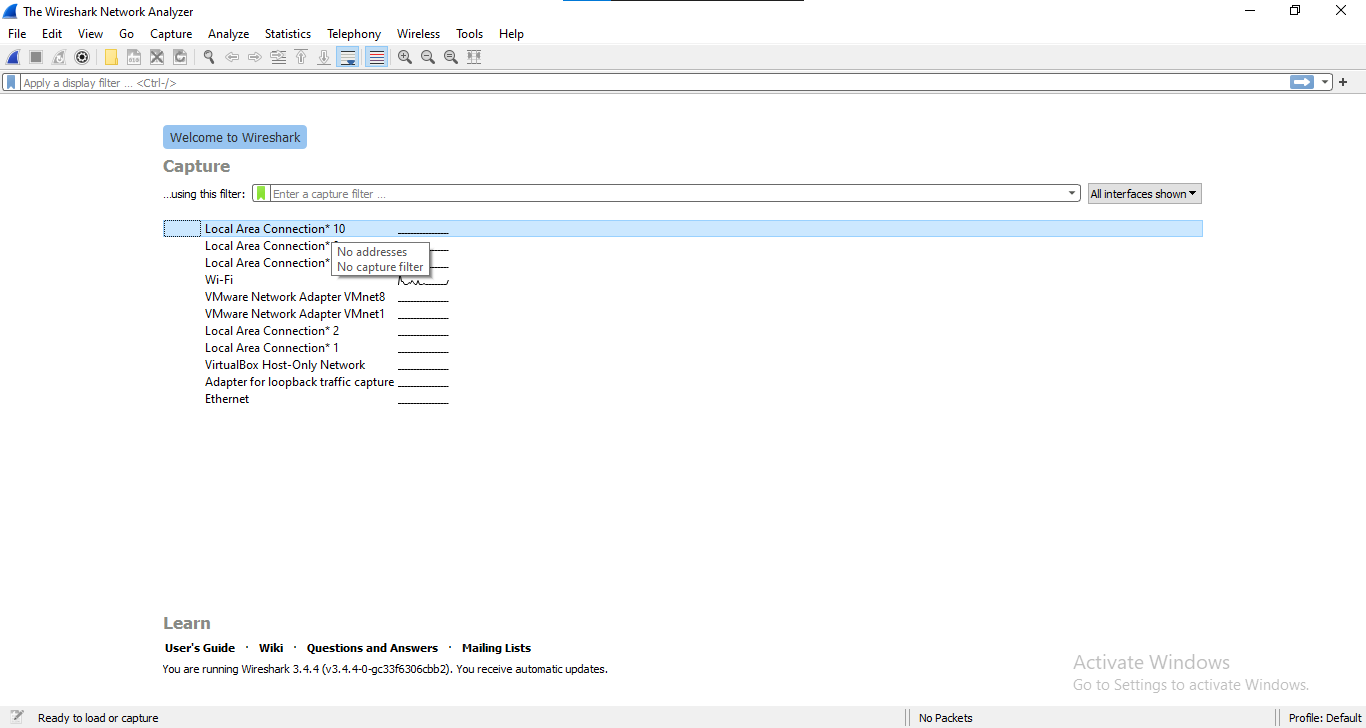
****

****

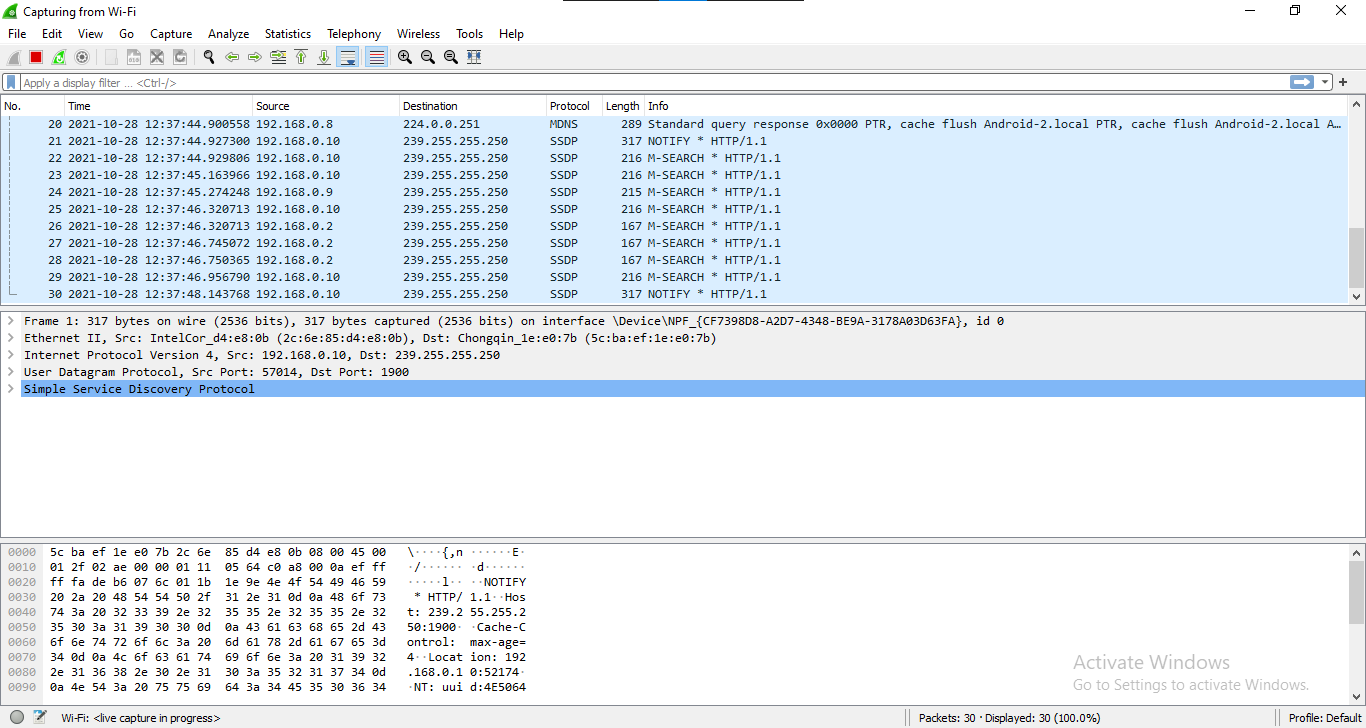
****

**COMMAND & OUTPUT:**

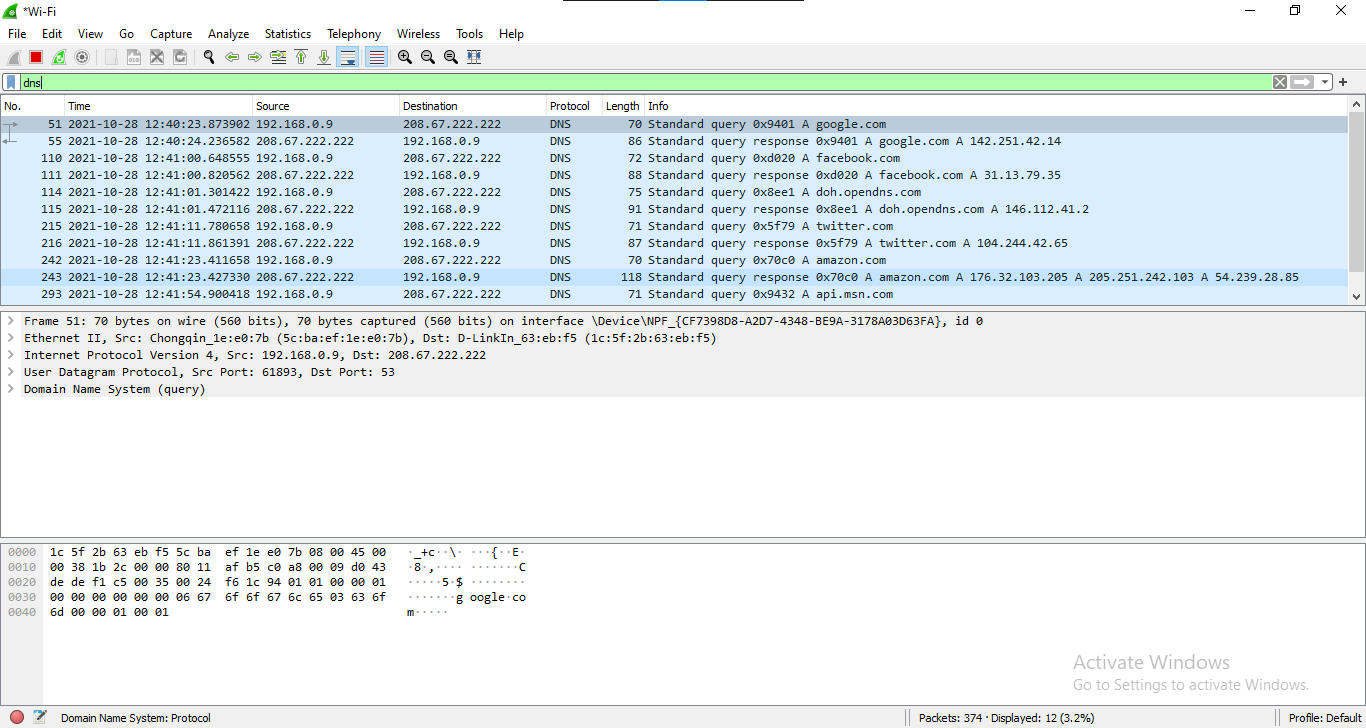
1. Selecting the Network



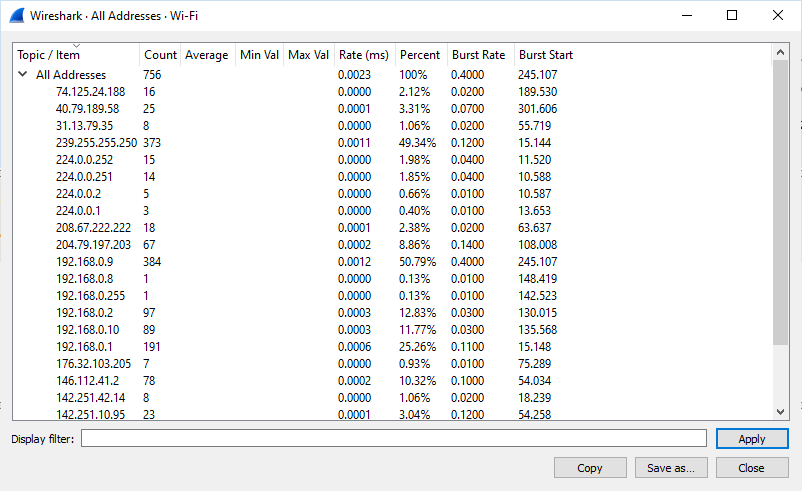
1. Capturing Packets

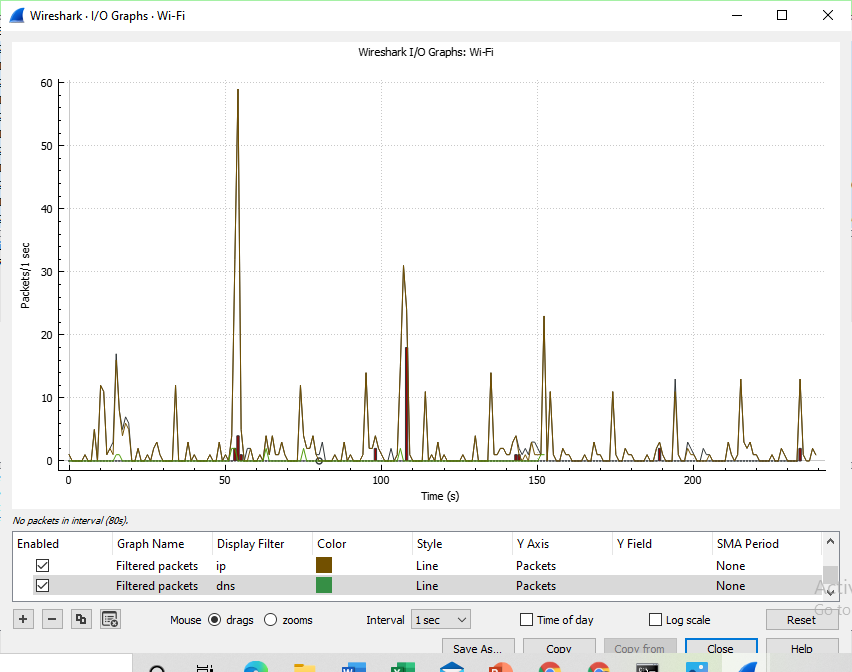


1. Applying Filters



1. Ipv4 tracer



1. I/O Graph

**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Study of packet sniffer tools Wireshark completed successfully.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **09** |
| Title: | **Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.** |
| Date of Performance: | **29/09/21** |
| Date of Submission: | **06/10/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

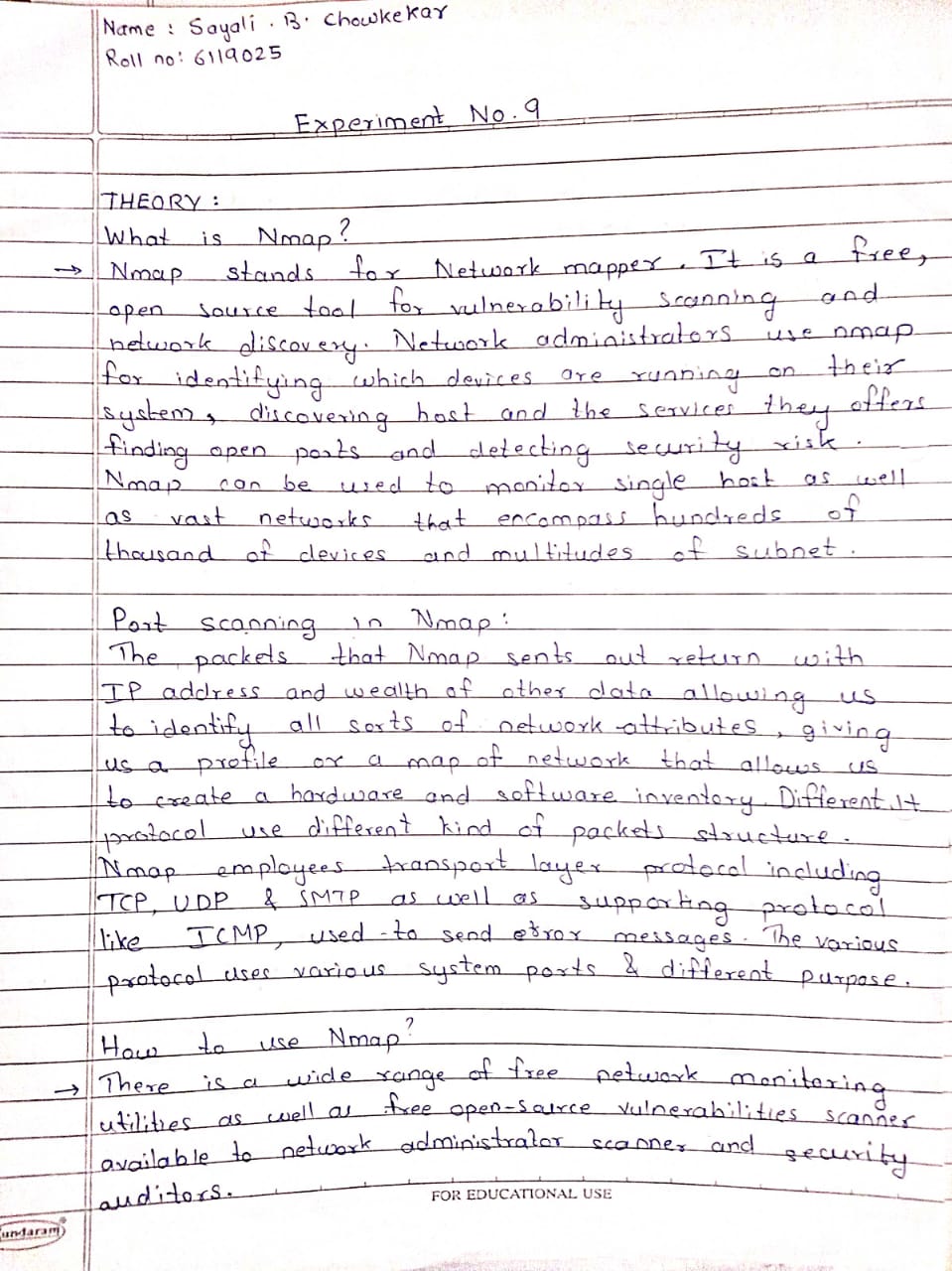
Date:

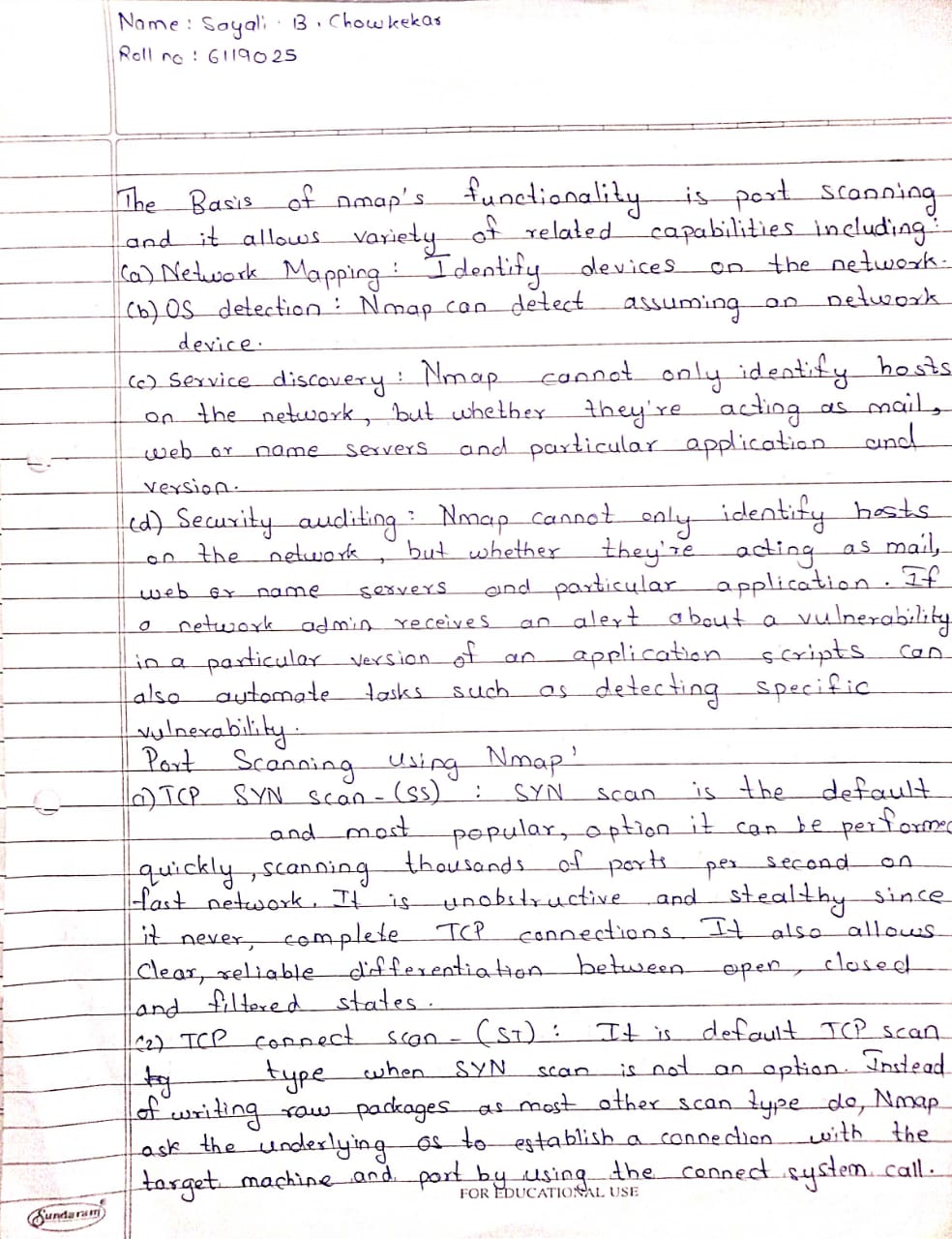
**EXPERIMENT NO: 09**

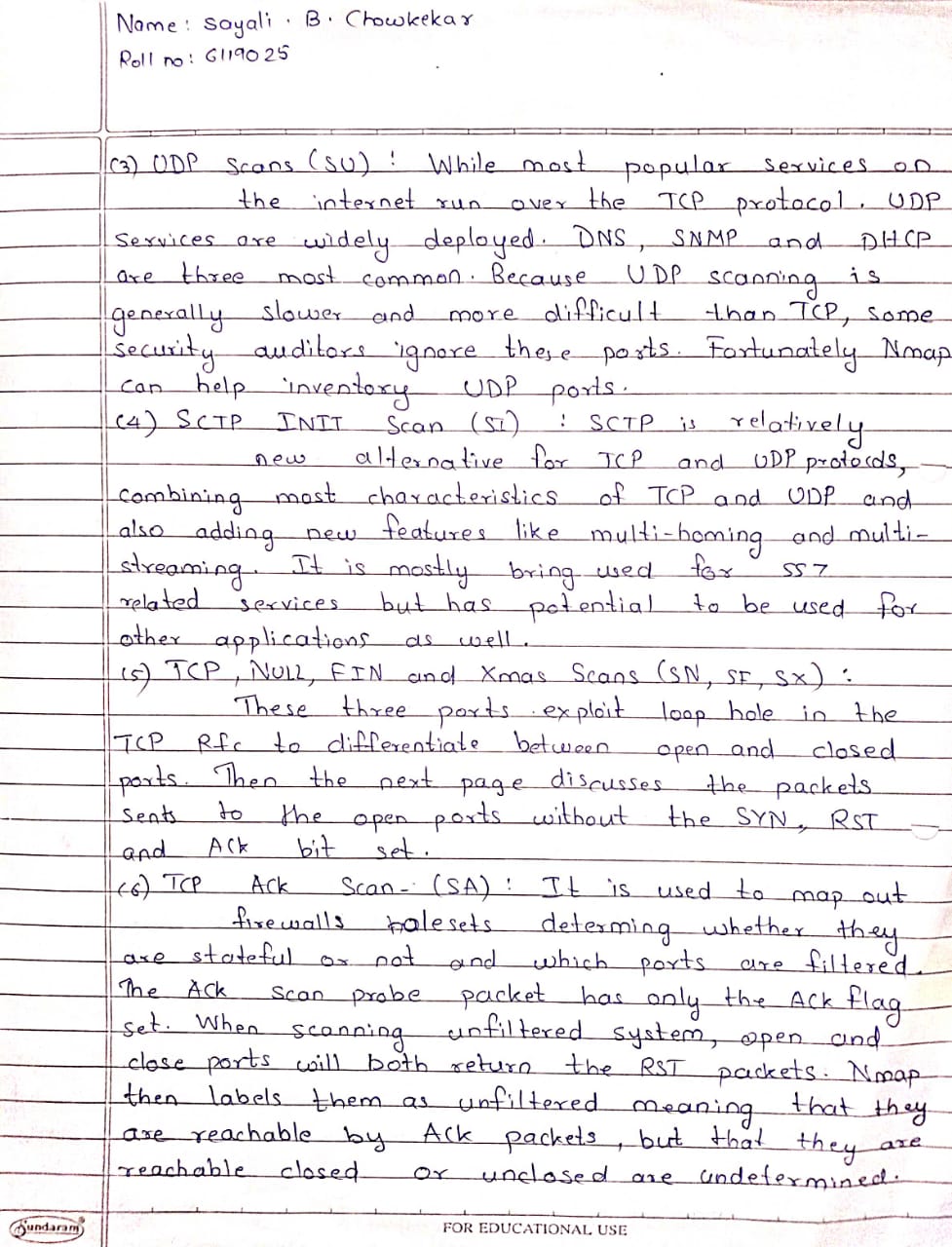
**AIM:** Download, install nmap and use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop, Nmap installed.

**THEORY:**

****

****

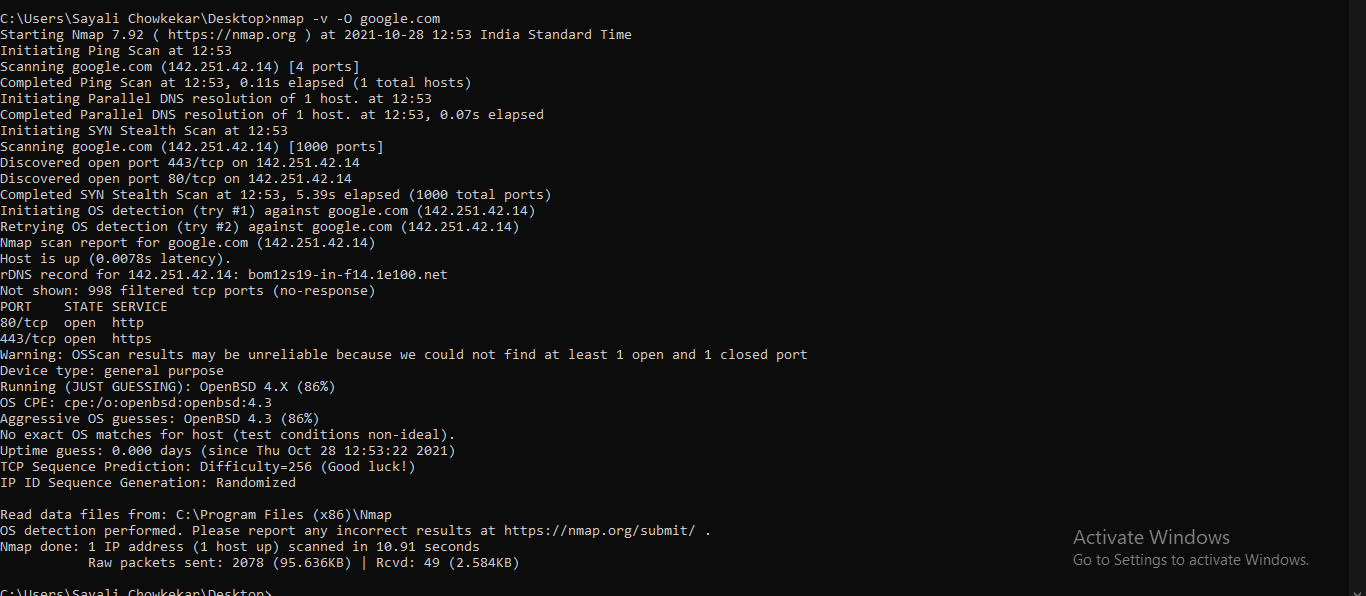
****

**PRACTICAL PERFORMING:**

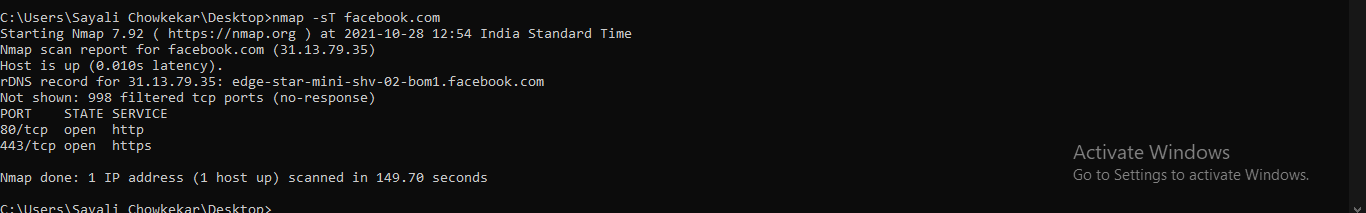
**Port scanning:**

****

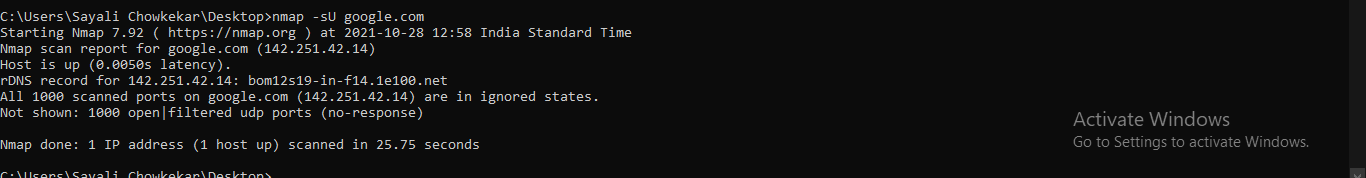
**OS Fingerprinting:**

****

**TCP Scan:**

****

**UDP Scan:**

****

**RESULT:** Program executed with zero errors.

**VERIFICATION & VALIDATION:** Output is verified & found correct.

**CONCLUSION:** Successfully completed various scanning techniques using Nmap.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **10** |
| Title: | **Study of malicious software using different tools:**  **a) Keylogger attack using a keylogger tool.**  **b) Simulate DOS attack using Hping or other tools**  **c) Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities.** |
| Date of Performance: | **06/10/21** |
| Date of Submission: | **18/10/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

Date:

**EXPERIMENT NO: 10**

**AIM:** Study of malicious software using different tools:

a) Keylogger attack using a keylogger tool.

b) Simulate DOS attack using Hping or other tools

c) Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities.

**TOOLS USED:** Internet connection, PC/Laptop/Desktop.

**THEORY:**

1. **Keylogger attacks using a keylogger tool.**

* Keyloggers or keystroke loggers are software programs or hardware devices that track the activities (keys pressed) of a keyboard. Keyloggers are a form of spyware where users are unaware of their actions are being tracked.
* Keyloggers can be used for a variety of purposes; hackers may use them to maliciously gain access to your private information, while employers might use them to monitor employee activities. Some keyloggers can also capture your screen at random intervals; these are known as screen recorders.
* Keylogger software typically stores your keystrokes in a small file, which is either accessed later or automatically emailed to the person monitoring your actions.

Keylogger Software:

Remote- access software keyloggers can allow access to locally recorded data from a remote location. This communication can happen by using one of the following methods:

* Uploading the data to a website, database or FTP server. ● Periodically emailing data to a predefined email address.
* Wirelessly transmitting data through an attached hardware system.
* Software enabling remote login to your local machine.

Additional features that some software keyloggers come with can capture additional information without requiring any keyboard key presses as input. They include:

* Clipboard logging – Anything that can be copied to the clipboard is captured.
* Screen logging – Randomly timed screenshots of your computer screen are logged.
* Control text capture – The Windows API allows for programs to request the text value of some controls, meaning that your password may be captured even if behind a password mask (the asterisks you see when you type your password into a form).
* Activity tracking – Recording of which folders, programs and windows are opened and possibly screenshots of each.
* Recording search engine queries, instant message conversations, FTP downloads along with any other internet activities.

1. Simulate DOS attack using Hping or other tools

* A denial of Service (DOS) attack is a very simple technique to deny accessibility to services (that’s why it is called “denial of service” attack). This attack consists of overloading the target with oversized packets, or a large quantity of them.
* A Distributed Denial of Service attack (DDOS) is like a DOS attack but carried out from different nodes (or different attackers) simultaneously. Commonly DDOS attacks are carried out by botnets.
* Botnets are automated scripts or programs which infect computers to carry out an automated task (in this case a DDOS attack). A hacker can create a botnet and infect many computers from which botnets will launch DOS attacks, the fact many botnets are shooting simultaneously turn the DOS attack into a DDOS attack (that’s why it is called “distributed”).
* The tool hping3 allows you to send manipulated packets. This tool allows you to control the size, quantity and fragmentation of packets in order to overload the target and bypass or attack firewalls. 75
* Hping3 can be useful for security or capability testing purposes, using it you can test firewalls efficiency and if a server can handle a big number of packets.
* A simple DOS (not DDOS) attack would be:

○ # Sudo hping3 -S --flood -V -p 80 170.155.9.185

* Where:
* Sudo: gives needed privileges to run hping3.
* hping3: calls hping3 program
* -S: specifies SYN packets.
* –flood: shoot at discretion, replies will be ignored (that’s why replies won’t be shown) and packets will be sent fast as possible.
* -V: Verbosity.
* -p 80: port 80, you can replace this number for the service you want to attack.
* 170.155.9.185: target IP.

1. Use the NESSUS/ISO Kali Linux tool to scan the network for vulnerabilities

* Nessus is commercial software made to scan for vulnerabilities, but the free home version offers plenty of tools to help explore and shore up your home network. It also points you to a variety of different tools to then penetration test a network.
* Nessus can scan for quite a few different problems, but most of us will be content using the Basic Network Scan because it offers a good overview. ○ Click the “New Scan.”
* Click “Basic Network Scan.”
* Name your scan and add a description.
* In the “Targets” field, you’ll want to enter IP scanning details about your home network. For example, if your router is at 192.168.0.1, you’d want to enter 192.168.0.1/24. This will make it so Nessus scans all the devices on your network (unless you have a ton of devices this is probably as high as you’d need to go). If you’re not sure about the local IP address for your router, here’s how to find it.
* Click “Save.”
* On the next screen, click the Play icon to launch the scan.
* Depending on what and how many devices you have on your network, the scan takes a while, so sit back and relax while Nessus does its work.

**RESULT:**  Case study completed.

**VERIFICATION & VALIDATION:** Case study verified & found appropriate.

**CONCLUSION:** Successfully completed the case study on malicious softwares using different tools.

Anjuman-I-Islam’s

**M. H. SABOO SIDDIK COLLEGE OF ENGINEERING**

8, Saboo Siddik Polytechnic Road, Byculla, Mumbai-400008

DEPARTMENT OF INFORMATION TECHNOLOGY

**Class: TE SEM V**

**Subject Name: Security Lab**

**Subject Code: ITL502**

|  |  |
| --- | --- |
| Practical No. | **11** |
| Title: | **Study of Network security by**  **a) Set up IPSec under Linux.**  **b) Set up Snort and study the logs.**  **c) Explore the GPG tool to implement email security** |
| Date of Performance: | **18/10/21** |
| Date of Submission: | **22/10/21** |
| Name of Student: | **Chowkekar Sayali Baban.** |
| Roll No: | **6119025.** |

**Evaluation:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rubric | Marks |
| 1 | On time Submission & Completion |  |
| 2 | Knowledge |  |
| 3 | Performance |  |
| 4 | Discipline |  |
| Total | |  |

Signature of the Teacher:

Date:

**EXPERIMENT NO: 11**

**AIM:** Study of Network security by:

a) Set up IPSec under Linux.

b) Set up Snort and study the logs.

c) Explore the GPG tool to implement email security

**TOOLS USED:** Internet connection, PC/Laptop/Desktop.

**THEORY:**

1. Set up IPsec under Linux.

There are so many benefits of using a VPN (Virtual Private Network), some of which include keeping you safe on the internet by encrypting your traffic and helping you to access blocked content/sites/web applications from anywhere. Not to mention, VPN also helps you to browse the internet anonymously. To set up the VPN server, we will use a wonderful collection of shell scripts created by Lin Song, that installs Libre swan as the IPsec server, and xl2tpd as the L2TP provider. The offering also includes scripts to add or delete VPN users, upgrade the VPN installation and much more. First, log into your VPS via SSH, then run the appropriate commands for your distribution to set up the VPN server. By default, the script will generate random VPN credentials (pre-shared key, VPN username, and password) for you and display them at the end of the installation. However, if you want to use your own credentials, first you need to generate a strong password and PSK Next, set these generated values as described in the following command all values MUST be placed inside ‘single quotes 'as shown.

* VPN\_IPSEC\_PSK – Your IPsec pre-shared key.
* VPN\_USER – Your VPN username.
* VPN\_PASSWORD – Your VPN password.

The main packages that will be installed are bind-utils, net-tools, bison, flex, gcc, libcapng

devel, libcurl-devel, libselinux-devel, nspr-devel, nss-devel, pamdevel, xl2tpd, iptables-services, systemd-devel, fipscheck-devel, libevent-devel, and fail2ban (to protect SSH), and their respective dependencies. Then it downloads, compiles, and installs Libreswan from source, enables and starts the necessary services. Once the installation is complete, the VPN details will be displayed.

1. Set up Snort and study the logs

Snort is a free lightweight network intrusion detection system for both UNIX and Windows.

1. Download the latest snort free version from the snort website. Extract the snort source code to the /usr/src directory as shown below.

# cd /usr/src

# wget -O snort-2.8.6.1.tar.gz <http://www.snort.org/downloads/116>

# tar xvzf snort-2.8.6.1.tar.gz

1. Install Snort

Before installing snort, make sure you have dev packages of libpcap and libpcre.

# apt-cache policy libpcap0.8-dev

libpcap0.8-dev:

Installed: 1.0.0-2ubuntu1

Candidate: 1.0.0-2ubuntu1

# apt-cache policy libpcre3-dev

libpcre3-dev:

Installed: 7.8-3

Candidate: 7.8-3

1. Verify the Snort Installation
2. Create the required files and directory.
3. Execute snort

Execute snort from command line, as mentioned below.

# snort -c /etc/snort/snort.conf -l /var/log/snort/

1. Try pinging some IP from your machine, to check our ping rule. Following is

the example of a snort alert for this ICMP rule.

# head /var/log/snort/alert

[\*\*] [1:477:3] ICMP Packet [\*\*]

[Priority: 0]

07/27-20:41:57.230345 > l/l len: 0 l/l type: 0x200 0:0:0:0:0:0

pkt type:0x4 proto: 0x800 len:0x64

209.85.231.102 -> 209.85.231.104 ICMP TTL:64 TOS:0x0 ID:0

IpLen:20 DgmLen:84

Type:8 Code:0 ID:24905 Seq:1 ECHO

c) Explore the GPG tool to implement email security

Email security, asymmetric encryption, keys…all confusing and complicated subjects to the uninitiated. However, there are some tools that help make the job of keeping your email secure much easier, especially on a Mac.

GPGTools is a free open-source software suite (based on GnuPG) that handles encrypting files and email messages. It has plugins for Apple Mail and the Services context-aware menu for encrypting text, files, or anything else you have selected.

GPGTools—and every solution based on GnuPG—uses private and public key pair technology. A key is basically a blob of random text that looks something like this, only much longer:

mRINBGH4Ws8BEACzr\QLVZ2XOqEHADX5dhqrl.

Keys are created in pairs: a public and a private key that fit together.

The public key is available for anybody to see and use. Messages sent to the key owner are encrypted by the sender using the public key. Once encrypted, they can only be decrypted by using the matching private key, since the encryption process is one-way. Public keys may be distributed on owners’ websites, in email signatures, on public keyservers (directories of public keys and their owners’ names/email addresses), or anywhere else the owner wishes to post it. The private key is for the owner’s use only; it allows the owner to decrypt anything that was encrypted with the matching public key. Private keys are the most important part of the entire key pair system—if the private key gets out, anybody can decrypt messages intended for the key owner. You should guard your private keys very carefully.

The public and private key pair can also be used to verify the sender of a message; if the sender signs it with his private key, anybody can use the public key to verify that the message in fact was signed with the matching private key and that it was not tampered with.

How to Use GPGTools

a. Creating a private-public key pair

After installing GPGTools, open GPG Keychain Access. You will see one public key in the keychain already from the GPGTools Team. Click on the New icon in the toolbar to set up your first key. Enter your name and email address and leave the other settings as they are unless you have a need to change them.

You will be prompted to enter a strong password for the key; once you’ve entered your password, you’ll want to move your mouse around and type random characters on the keyboard to help create extra random noise to generate a good key. After a few seconds, you’ll see your brand-new key show up in the GPGTools Keychain.

If you want other people to be able to send you encrypted email (and why wouldn’t you?), you’ll want to upload your public key to keyservers, collections of public keys available for anybody to search. Just right-click on the key and choose Send public key to keyserver. You could also post the key or a link to it on your website, in your email signature, etc., so people can find it easily.

b. Encrypting Email

When you compose a new email message, you’ll notice a new OpenPGP header in the compose window, as well as a lock icon and a seal icon. The seal means that this email will be digitally signed with your key. By default, this is turned on for all messages.

To encrypt a message, you need to have the public key for the recipient installed in your GPG Keychain (search the public keyservers or ask them for it). Once you have the public key installed, the lock icon becomes active and you can click on it to turn encryption on (if you don’t have their public key, the lock icon is disabled).

**RESULT:**  Case study completed.

**VERIFICATION & VALIDATION:** Case study verified & found appropriate.

**CONCLUSION:** Successfully completed the case study on Network security.