Vidyavardhini's
College of Engineering & Technology

Vasai Road (W)

Department of
Information Technology

Security Lab Manual

Semester	**V**	**Class**	**TE**
Course Code	**ITL502**	**Academic Year**	**2022-23**
Course Name	**Security Lab.**		
Name of Faculty	**Prof. Thaksen J. Parvat**		
Suppoting Staff	**Mr. Nitin Shingane**		

Vidyavardhini's College of Engineering & Technology

Vision

To be a premier institution of technical education, aiming at becoming a valuable resource for industry and society.

Mission

- To provide technologically inspiring environment for learning.
- To promote creativity, innovation and professional activities.
- To inculcate ethical and moral values.
- To cater personal, professional and societal needs through quality education.

Department Vision:

To foster and maintain excellence by orienting the captivating minds of the aspiring engineers towards IT- driven technological solutions for the benefits of the society.

Department Mission:

- To provide quality education, by employing best and diversified teaching practices and tools, and teaching beyond the confines of the university syllabus.
- To keep students abreast with latest technological advancements in the market.
- To prepare students to troubleshoot and solve IT system problems.

Program Education Objectives (PEOs):

- To produce skilled IT Professional to cater social/industrial needs.
- To inculcate an ability to implement modern practices with ethical and professional responsibilities.
- To establish graduate as Business Analyst, System Analyst, Data Scientist, Project Leader.

Program Specific Outcomes (PSOs):

The graduates will be able to

- Apply and implement IT solutions in allied fields of engineering to solve real word problems.
- Identify social and industrial problems, provide creative solutions and become quality asset for society and industry.
- Deploy secured solution using Information Technology practices and strategies.

Program Outcomes (POs):

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Syllabus (Optional)

Hardware & Software Requirements:

DETAILED SYLLABUS:

Sr. No.	Module	Detailed Content	Hours	LO Mapping
0	**Prerequisite**	Programming Language (C, C++, ,Python, Java), Windows and Unix/Linux operating system. editor commands (eg nano/vi editor etc)	**02**	-
I	**Fundamentals of Cryptography**	Classical Encryption techniques (mono-alphabetic and poly-alphabetic substitution techniques: Vigenere cipher, playfair cipher).	**04**	LO1
II	**Basics Cryptography**	1)Block cipher modes of operation using a)Data Encryption Standard b)Advanced Encryption Standard (AES). 2)Public key cryptography: RSA algorithm. 3)Hashing Techniques: HMAC using SHA 4)Digital Signature Schemes – RSA, DSS.	**04**	LO2
III	**Network Commands, different Protocols**	1) Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars. 2) 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.	**06**	LO3
IV	**Network Mapper and Commands**	1) Download and install nmap. 2) Use it with different options to scan open ports, perform OS fingerprinting, ping scan, tcp port scan, udp port scan, etc.	**04**	LO4
V	**Network Keylogers**	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.	**04**	LO5
VI	**Network Security Design**	• 1) Set up IPSec under Linux. 2) Set up Snort and study the logs. 3) Explore the GPG tool to implement email security	**04**	LO6

Text Books:

1 Build your own Security Lab, Michael Gregg, Wiley India.
2 CCNA Security, Study Guide, TIm Boyles, Sybex.
3 Hands-On Information Security Lab Manual, 4th edition, Andrew Green, Michael Whitman, Herbert Mattord.

4 The Network Security Test Lab: A Step-by-Step Guide Kindle Edition, Michael Gregg.

References:

1 Network Security Bible, Eric Cole, Wiley India.
2 Network Defense and Countermeasures, William (Chuck) Easttom.
3 Principles of Information Security + Hands-on Information Security Lab Manual, 4th Ed. , Michael E. Whitman , Herbert J. Mattord.
4 IITB virtual Lab: http://cse29-iiith.vlabs.ac.in/

https://www.dcode.fr/enTerm Work: Term Work shall consist of at least 10 to 12 practical's based on the above list. Also Term workJournal must include at least 2 assignments.

Term Work Marks: 25 Marks (Total marks) = 15 Marks (Experiment) + 5 Marks (Assignments) + 5 Marks(Attendance)

Practical & Oral Exam: An Oral & Practical exam will be held based on the above syllabus.

ITL (502) Security Lab.

Objectives:

Sr. No. Lab Objectives

The Lab experiments aims:

1 To apply the knowledge of symmetric cryptography to implement classical ciphers.
2 To analyze and implement public key encryption algorithms, hashing and digital signature algorithms..
3 To explore the different network reconnaissance tools to gather information about networks.
4 To explore the tools like sniffers, port scanners and other related tools for analyzing..
5 To Scan the network for vulnerabilities and simulate attacks.
6 To set up intrusion detection systems using open-source technologies and to explore email security.

Lab Outcomes:

Sr. No.	**Lab Outcomes**	**Cognitive levels of attainment as per Bloom's Taxonomy**
On successful completion, of course, learner/student will be able to:		
1	Illustrate symmetric cryptography by implementing classical ciphers.	L1, L2
2	Demonstrate Key management, distribution and user authentication.	L1, L2
3	Explore the different network reconnaissance tools to gather information about networks	L1, L2, L3
4	Use tools like sniffers, port scanners and other related tools for analyzing packets in a network.	L1, L2, L3
5	Use open-source tools to scan the network for vulnerabilities and simulate attacks.	L1, L2, L3
6	Demonstrate the network security system using open source tools.	L1, L2

List of experiments

Sr. No.	Name of the experiment
1.	Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.
2.	Cryptanalysis or decoding Playfair cipher.
3.	Cryptanalysis or decoding Vigenère cipher.
4.	Encrypt long messages using various modes of operation using AES.
5.	Encrypt long messages using various modes of operation using DES.
6.	Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions.
7.	Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.
8.	Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars
9.	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.
10	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.
11.	Use tools like sniffers, port scanners and other related tools for analyzing packets in a network.
12	Study of Network security by a) Set up IPSec under Linux. b) Set up Snort and study the logs.

Prof. Thaksen Parvat
Subject In charge

Prof. Thaksen Parvat
HOD IT

Course Outcomes:

At the end of the course student will be able to:		Bloom's Level
ITL502.1	Execute and evaluate different Cryptographic methods for short and long messages.	Apply
ITL502.2	Demonstrate the installation and configuration of network simulator and measure different network scenarios and their performance behaviour.	Apply
ITL502.3	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.	Analyse
ITL502.4	Demonstrate tools like sniffers, port scanners and other related tools for analyzing packets in a network.	Analyse

Mapping of Experiments with Course Outcomes

Experiment	Course Outcomes			
	ITL502.1	ITL502.2	ITL502.3	ITL502.4
Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.	3			
Cryptanalysis or decoding Playfaircipher.	3			
Cryptanalysis or decoding vigenere cipher.		3		
Encrypt long messages using various modes of operation using AES.		3		
Encrypt long messages using various modes of operation using DES.		3		
Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions.		3		
Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.			3	
Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars			3	
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.				3
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.				3
Use tools like sniffers, port scanners and other related tools for analyzing packets in a network.				3
Study of Network security by a) Set up IPSec under Linux. b) Set up Snort and study the logs.				3

CERTIFICATE

This is to certify that, Mr./MS. _____

_____Roll. No._____ of TE Information Technology work in the subject Security Laboratory (ITL 502), Satisfactory in the department IT as prescribed by Mumbai University in the academic year 2022-23 Semester V.

Staff In-charge **Head of the Department** **Principal**

Date: 28/20/2022
Place: VCET, Vasai Road

Index

Year/Sem: T.E/Sem V **Subject: Network Lab. (ITL 502)**

Sr. No.	Name of the experiment	Page No.	Marks Out of 10	Date	Sign
1.	Breaking the Mono-alphabetic Substitution Cipher using Frequency analysis method.				
2.	Cryptanalysis or decoding Playfair cipher.				
3.	Cryptanalysis or decoding vigenere cipher.				
4.	Encrypt long messages using various modes of operation using AES.				
5.	Encrypt long messages using various modes of operation using DES.				
6.	Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions.				
7.	Implementation and analysis of RSA cryptosystem and Digital signature scheme using RSA.				
8.	Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars				
9.	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.				
10	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.				

Prof. Thaksen Parvat
Subject In charge

Prof. Thaksen Parvat
HOD IT

Experiment No. 1

Aim :- To apply the knowledge of symmetric cryptography to implement classical ciphers.

Theory:- Cryptography is the technique which is used for doing secure communication between two parties in the public environment where unauthorized users and malicious attackers are present. In cryptography there are two processes i.e. encryption and decryption performed at sender and receiver end respectively. Encryption is the processes where a simple multimedia data is combined with some additional data (known as key) and converted into unreadable encoded format known as Cipher. Decryption is the reverse method as that of encryption where the same or different additional data (key) is used to decode the cipher and it is converted in to the real multimedia data.

Caesar Cipher in Cryptography

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text been moved down.
The encryption can be represented using modular arithmetic by first

transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25.
Encryption of a letter by a shift n can be described mathematically as.

The formula of encryption is:

$$E_n(x) = (x + k) \bmod 26$$

The formula of decryption is:

$$D_n(x) = (xi - k) \bmod 26$$

Example :- encryption

Key : 3

Plaintext : vartak

Plaintext	**v**	**a**	**r**	**t**	**a**	**k**
Plaintext value	**21**	**0**	**17**	**19**	**0**	**10**
(x + k) mod 26	**24**	**3**	**20**	**22**	**3**	**13**
ciphertext	**Y**	**D**	**U**	**W**	**D**	**N**

Plaintext : vartak

Ciphertext : YDUWDN

Program & Codes : So we have implemented The Shift Encryption technique in C++ ,Java and Python the 3 generally used programming languages ;

#1 In C++

```
#include <iostream>

using namespace std;

// using Shift Cipher string
newmessage = "";
string encrypt(string m, int k)
{
  string encryp = "";
  for (int i = 0; i < m.length(); i++)
  {
    // char ex = letter.at(i);
    // cout<< int(ex) << "\n";

    if (isupper(m[i]))
      // For Upper case
      encryp += char(int(m[i] + k - 65) % 26 + 65);

    // For Lowercase letters
    else
      encryp += char(int(m[i] + k - 97) % 26 + 97);
  }

  newmessage = encryp;

  return encryp;
}

string decrypt(string newmessage, int k)
{
  string decrypt = "";
  for (int i = 0; i < newmessage.length(); i++)
  {
    if (isupper(newmessage[i]))
      // for upper case
      decrypt += char(int(newmessage[i] - k - 65) % 26 + 65);
```

```
    else
      decrypt += char(int(newmessage[i] - k - 97) % 26 + 97);

  }

  return decrypt;
}

int main()
{
  string m;
  int k;
  cout << "Enter the Key:\n";
  cin >> k;
  cout << "Enter a message:\n";
  cin >> m;
  cout << "Encrypted Message is:\n";
  cout << encrypt(m, k) << "\n";

  cout << "Decrypted Message is:\n";
  cout << decrypt(newmessage, k);
  return 0;
}
```

OUTPUT :

#2 In Java

```
import java.util.*;
import java.io.*;

// #24
public class encrypt {
   public static String temp = "";
   public static int k;
```

```
StringBuffer encrypto = new StringBuffer();

// Encrypts text using a shift cipher
public static StringBuffer encrypt(String m, int k) {

   StringBuffer encrypto = new StringBuffer();

   for (int i = 0; i < m.length(); i++) {
      if (Character.isUpperCase(m.charAt(i))) {
         char ch = (char) (((int) m.charAt(i) +
                  k - 65) % 26 + 65);
            encrypto.append(ch);
      } else {
         char ch = (char) (((int) m.charAt(i) +k
               - 97) % 26 + 97);
         encrypto.append(ch);
      }
   }

   temp = encrypto.toString();

   return encrypto;

}

// decryption
public static StringBuffer decrypt(String temp, int k) {

   StringBuffer decrypto = new StringBuffer();



   for (int i = 0; i < temp.length(); i++) {
      if (Character.isUpperCase(temp.charAt(i))) {
         char ch = (char) (((int) temp.charAt(i) -
                  k - 65) % 26 + 65);
            decrypto.append(ch);
      } else {
         char ch = (char) (((int) temp.charAt(i) -k
               - 97) % 26 + 97);
         decrypto.append(ch);
      }
   }
   return decrypto;
```

```
    }

    // main class & method
    public static void main(String[] args) { Scanner

        sc = new Scanner(System.in);

        System.out.println("Enter a key");
        int k = sc.nextInt();
        System.out.println("Enter the text\n");
        String m = sc.next();

        System.out.println("Text : " + m); System.out.println("Key :
        " + k); System.out.println("Cipher: " + encrypt(m, k));
        System.out.println("Decrypted " + decrypt(temp, k));
    }
}
```

OUTPUT :

#3 in Python

```
def encypt_func(txt,s):
    global result

# transverse the plain txt
    for i in range(len(txt)):
        char = txt[i]
        # encypt_func uppercase characters in plain txt

        if (char.isupper()):
            result += chr((ord(char) + s - 65) % 26 + 65)
        # encypt_func lowercase characters in plain txt
        else:
```

```
            result += chr((ord(char) + s - 97) % 26 + 97)

    return result

def decypt_func():
    result1 = ""


# transverse the plain txt for i
    in range(len(result)):
        char = result[i]
        # encypt_func uppercase characters in plain txt

        if (char.isupper()):
            result1 += chr((ord(char) - s - 67) % 26 + 65) #
        encypt_func lowercase characters in plain txt
        else:
            result1 += chr((ord(char) - s - 97) % 26 + 97)
    return result1

from glob import glob
s = int(input("Enter the  shift key \t"))

txt = input("Enter a message \t")

result = ""

print("Plain txt : " + txt)
print("Shift pattern : " + str(s))#
calling the above function
print("Cipher: " + encypt_func(txt, s))
print("Decrypted: " + decypt_func())
```

OUTPUT :

SELF TESTING QUESTIONS :

Q1 List down all the substitution type Cipher methods in Cryptology .

Q2 What is the difference between The Substitution Cipher and Shift Cipher .

Q3 Give an application of Shift Cipher in Cyber security.

Q4 State drawbacks of Shift Cipher .

Q5 Judging on the working and algorithm of all and only substitution Cipherswhich one do you feel is a better option for real time deployment ?

Conclusion : So therefore we learnt cryptology in Computer Networking , studied simple encryption techniques and implemented shift cipher methodology successfully in the big three languages – C++ , Java & Python

Experiment No. 2

Aim : Identify The Basic Cryptographic Techniques Using Playfair Cipher

Theory :

When it was first put to the British Foreign Office as a cipher, it was rejected due to its perceived complexity. However, it was later adopted as a military cipher due to it being reasonably fast to use, and it requires no special equipment, whilst also providing a stronger cipher than a Monoalphabetic Substitution Cipher. It was used in the Second Boer War, and both World War I and World War II to different degrees. It is no longer used by military forces since the advent of powerful computers, but in its day it provided a relatively secure cipher which was easy to implement quite quickly.

In order to encrypt using the Playfair Cipher, we must first draw up a Polybius Square (but without the need for the number headings). This is usually done using a keyword, and either combining "i" and "j" or omitting "q" from the square.

Steps to implement the Playfair chipper :

1. We must now split the plaintext up into digraphs (that is pairs of letters). On each digraph we perform the following encryption steps:
2. If the digraph consists of the same letter twice (or there is only one letter left by itself at the end of the plaintext) then insert the letter "X" between the same letters (or at the end), and then continue with the rest of the steps.
3. If the two letters appear on the same row in the square, then replace each letter by the letter immediately to the right of it in the square (cycling round to the left hand side if necessary).
4. If the two letters appear in the same column in the square, then replace each letter by the letter immediately below it in the square (cycling round to the top of the square if necessary).
5. Otherwise, form the rectangle for which the two plaintext letters are two opposite corners. Then replace each plaintext letter with the letter that forms the other corner of

the rectangle that lies on the same row as that plaintext letter (being careful to maintain the order).

EXAMPLE:

Plain Text: This is Final Exam Key : President .

TH IS FI NA LE AX MX

(I) (II)

P R E S I/J

D N T A B

C F G H K

L M O Q U

V W X Y Z

P R E S I/J

D N T A B

C F G H K

L M O Q U

V W X Y Z

P R E S I/J

D N T A B

C F G H K

L M O Q U

V W X Y Z

P R E S I/J

D N T A B

C F G H K

L M O Q U

V W X Y Z

(III) (IV)

(V) (VI)

(VII) (VIII)

Plain Text: This is Final Exam

Chipper Text : AGPIPIKRTBOPYTOW

C++ PROGRAM :

```
#include<iostream>
#include<vector>
using namespace std;

void get_pos(char, int&, int&);
void same_row(int, vector<char>&, int, int);
void same_column(int, vector<char>&, int,
int);
void diff_col_row(int, int, vector<char>&, int,
int);
void encode(vector<char>, int);
void get_input(vector<char>&);

void convert_string(vector<char>&,
vector<char>&);

const char encoder[5][5]={{'A','B','C','D','E'},

{'F','G','H','I','K'},
{'L','M','N','O','P'},
{'Q','R','S','T','U'},
{'V','W','X','Y','Z'}};

void get_pos(char p, int& r, int& c)
{
if (p < 'J')
{
P R E S I/J
D N T A B
C F G H K
L M O Q U
```

```
V W X Y Z

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

r = (p - 65) / 5;
c = (p - 65) % 5;
}

}

void encode(vector<char> msgx, int len)


else if (p > 'J')

{

r = (p - 66) / 5;

c = (p - 66) % 5;

}

return;

}

void same_row(int r, vector<char>& code, int

c1, int c2)

{

code.push_back(encoder[r][(c1 + 1) % 5]);

code.push_back(encoder[r][(c2 + 1) % 5]);

return;

}

void same_column(int c, vector<char>& code,

int r1, int r2)
```

```
{
vector<char> code;
int i = 0, j = 0;
int r1, c1, r2, c2;
while (i < len)
{
get_pos(msgx[i], r1, c1);
i++;
get_pos(msgx[i], r2, c2);
if (r1 == r2)
{
same_row(r1, code, c1, c2);
}
else if (c1 == c2)
{
same_column(c1, code, r1, r2);
}
{
code.push_back(encoder[(r1 + 1) % 5][c]);
code.push_back(encoder[(r2 + 1) % 5][c]);
return;
}
else
{
diff_col_row(r1, c1, code, r2, c2);
}
i++;
```

```
}
void diff_col_row(int r1, int c1,
vector<char>& code, int r2, int c2)
{
code.push_back(encoder[r1][c2]);
code.push_back(encoder[r2][c1]);
return;

cout<<"\nCODE: ";
for (j = 0;j < code.size();j++)
{
cout<<code[j];
}
return;
}
void get_input(vector<char>& a)
{
char c;
while (1)
{
c = getchar();
if (c >= 97 && c <= 122)
c -= 32;
if (c == '\n')
break;
else if (c==' ')
continue;
```

```
else if (c == 'J')
a.push_back('I');
a.push_back(c);
}
return;
}

void convert_string(vector<char>& msg,
vector<char>& msgx)
{
int i, j;
i = j = 0;
while (i < msg.size())
{
msgx.push_back(msg[i]);
i++;
cout<<"\n\n";
for (i = 0;i < len;i++)
if (i == msg.size())
{
msgx.push_back('X');
break;
}
if (msg[i] == msgx[j])
{
msgx.push_back('X');
j++;
```

```
}
else if(msg[i] != msgx[j])
{
j++;
msgx.push_back(msg[i]);
i += 1;
}
j++;
}
}

int main()
{
vector<char> msg;
vector<char> msgx;
int i, j;
cout<<"Enter Message to Encrypt:";
get_input(msg);
convert_string(msg, msgx);
int len = msgx.size();
/*

cout<<msgx[i];
*///this is the string after making pairs of 2
encode(msgx, len);

return 0;
```

}

Output :

JAVA PORGRAM :

```
import java.awt.Point;
import java.util.Scanner;
public class PlayfairCipher
{
//length of digraph array
private int length = 0;
//creates a matrix for Playfair cipher
private String [][] table;
//main() method to test Playfair method
public static void main(String args[])
{
PlayfairCipher pf = new PlayfairCipher();
}
//main run of the program, Playfair method
//constructor of the class
private PlayfairCipher()
```

```
{
//prompts user for the keyword to use for
encoding & creates tables

System.out.print("Enter the key for playfair
cipher: ");
Scanner sc = new Scanner(System.in);
String key = parseString(sc);
while(key.equals(""))
key = parseString(sc);
table = this.cipherTable(key);
//prompts user for message to be encoded
System.out.print("Enter the plaintext to be
encipher: ");
//System.out.println("using the previously
given keyword");
String input = parseString(sc);
while(input.equals(""))
input = parseString(sc);
//encodes and then decodes the encoded
message
String output = cipher(input);
String decodedOutput = decode(output);
//output the results to user
this.keyTable(table);

this.printResults(output,decodedOutput);
```

```
}
//parses an input string to remove numbers,
punctuation,
//replaces any J's with I's and makes string all
caps
private String parseString(Scanner sc)
{
String parse = sc.nextLine();
//converts all the letters in upper case
parse = parse.toUpperCase();
//the string to be substituted by space for each
match (A to Z)
parse = parse.replaceAll("[^A-Z]", "");
//replace the letter J by I
parse = parse.replace("J", "I");
return parse;
}
//creates the cipher table based on some input
string (already parsed)
private String[][] cipherTable(String key)
{
//creates a matrix of 5*5
String[][] playfairTable = new String[5][5];
String keyString = key +
"ABCDEFGHIKLMNOPQRSTUVWXYZ";
//fill string array with empty string
for(int i = 0; i < 5; i++)
```

```
for(int j = 0; j < 5; j++)
playfairTable[i][j] = "";
for(int k = 0; k < keyString.length(); k++)
{
boolean repeat = false;
boolean used = false;

for(int i = 0; i < 5; i++)
{
for(int j = 0; j < 5; j++)
{
if(playfairTable[i][j].equals("" +
keyString.charAt(k)))
{
repeat = true;
}
else if(playfairTable[i][j].equals("") &&
!repeat && !used)
{
playfairTable[i][j] = "" + keyString.charAt(k);
used = true;
}
}
}
}
return playfairTable;
}
```

```
//cipher: takes input (all upper-case), encodes
it, and returns the output
private String cipher(String in)
{
length = (int) in.length() / 2 + in.length() % 2;
//insert x between double-letter digraphs &
redefines "length"

for(int i = 0; i < (length - 1); i++)
{
if(in.charAt(2 * i) == in.charAt(2 * i + 1))
{
in = new StringBuffer(in).insert(2 * i + 1,
'X').toString();

length = (int) in.length() / 2 + in.length() % 2;
}
}
//------------makes plaintext of even length------
--------
//creates an array of digraphs
String[] digraph = new String[length];
//loop iterates over the plaintext
for(int j = 0; j < length ; j++)
{
//checks the plaintext is of even length or not
if(j == (length - 1) && in.length() / 2 ==
```

```
(length - 1))
//if not addends X at the end of the plaintext
in = in + "X";
digraph[j] = in.charAt(2 * j) +""+ in.charAt(2
* j + 1);
}
//encodes the digraphs and returns the output
String out = "";
String[] encDigraphs = new String[length];
encDigraphs = encodeDigraph(digraph);
for(int k = 0; k < length; k++)
out = out + encDigraphs[k];
return out;
}
//---------------encryption logic-----------------
//encodes the digraph input with the cipher's
specifications
private String[] encodeDigraph(String di[])
{
String[] encipher = new String[length];
for(int i = 0; i < length; i++)
{

char a = di[i].charAt(0);
char b = di[i].charAt(1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
```

```
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
//executes if the letters of digraph appear in the
same row
//in such case shift columns to right
if(r1 == r2)
{
c1 = (c1 + 1) % 5;
c2 = (c2 + 1) % 5;
}
//executes if the letters of digraph appear in the
same column
//in such case shift rows down
else if(c1 == c2)
{
r1 = (r1 + 1) % 5;
r2 = (r2 + 1) % 5;
}
//executes if the letters of digraph appear in the
different row and different column
//in such case swap the first column with the
second column
else
{
int temp = c1;
c1 = c2;
c2 = temp;
```

```
}
//performs the table look-up and puts those
values into the encoded array

encipher[i] = table[r1][c1] + "" + table[r2][c2];
}
return encipher;
}
//----------------------decryption logic------------
---------
// decodes the output given from the cipher and
decode methods (opp. of encoding process)
private String decode(String out)
{
String decoded = "";
for(int i = 0; i < out.length() / 2; i++)
{
char a = out.charAt(2*i);
char b = out.charAt(2*i+1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
{
c1 = (c1 + 4) % 5;
c2 = (c2 + 4) % 5;
```

```
}
else if(c1 == c2)
{
r1 = (r1 + 4) % 5;
r2 = (r2 + 4) % 5;
}
else
{
//swapping logic
int temp = c1;

c1 = c2;
c2 = temp;
}
decoded = decoded + table[r1][c1] +
table[r2][c2];
}
//returns the decoded message
return decoded;
}
// returns a point containing the row and
column of the letter
private Point getPoint(char c)
{
Point pt = new Point(0,0);
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
```

```
if(c == table[i][j].charAt(0))
pt = new Point(i,j);
return pt;
}
//function prints the key-table in matrix form
for playfair cipher
private void keyTable(String[][] printTable)
{
System.out.println("Playfair Cipher Key
Matrix: ");
System.out.println();
//loop iterates for rows
for(int i = 0; i < 5; i++)
{
//loop iterates for column
for(int j = 0; j < 5; j++)
{
//prints the key-table in matrix form

System.out.print(printTable[i][j]+" ");
}
System.out.println();
}
System.out.println();
}
//method that prints all the results
private void printResults(String encipher,
```

```
String dec)
{
System.out.print("Encrypted Message: ");
//prints the encrypted message
System.out.println(encipher);
System.out.println();
System.out.print("Decrypted Message: ");
//prints the decryted message
System.out.println(dec);
}
}
```

Output:

PYTHON PROGRAM :

```
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 r
ange(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 charac
ter
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 matri
x
for i in range(0,5): #making matrix
for j in range(0,5):
my_matrix[i][j]=result[k]

k+=1

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

```
acter
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[(lo
c[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]][(lo
c1[1]+1)%5]),end=' ')
else:
print("{}{}".format(my_matrix[loc
[0]][loc1[1]],my_matrix[loc1[0]][loc[1]]),e
nd=' ')
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[(lo
c[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]]),e
nd=' ')
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")
```

Conclusion : Thus implement playfair chipper using different techniques

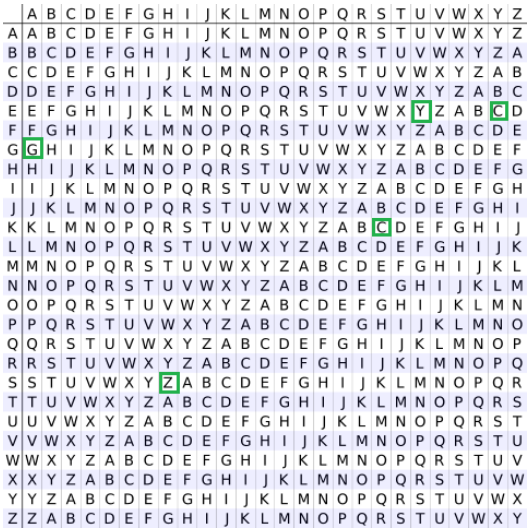
Experiment No. 3

Aim : To study and implement Vigenere Cipher in fundamental programming languages .

Theory :

Vigenere Cipher - Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenère square or Vigenère table.

- The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.
- At different points in the encryption process, the cipher uses a different alphabet from one of the rows.
- The alphabet used at each point depends on a repeating keyword.



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

Algorithm For Shift Cipher :

Encryption:
The first letter of the plaintext, G is paired with A, the first letter of the key. So use row G and column A of the Vigenère square, namely G. Similarly, for the second letter of the plaintext, the second letter of the key is used, the letter at row E, and column Y is C. The rest of the plaintext is enciphered in a similar fashion.

Decryption:
Decryption is performed by going to the row in the table corresponding to the key, finding the position of the ciphertext letter in this row, and then using the column's label as the plaintext.

A more **easy implementation** could be to visualize Vigenère algebraically by converting [A-Z] into numbers [0–25].

Encryption

The plaintext(P) and key(K) are added modulo 26.

$E_i = (P_i + K_i) \bmod 26$

Decryption

$D_i = (E_i - K_i + 26) \bmod 26$

Program & Codes : So we have implemented The Vigenere Encryption technique in C++ , Java and Python the 3 generally used programming languages ;

#1 In C++

```
#include<bits/stdc++.h>
using namespace std;


string generateKey(string str, string key)
{
	int x = str.size();

	for (int i = 0; ; i++)
	{
		if (x == i)
			i = 0;
		if (key.size() == str.size())
			break;
		key.push_back(key[i]);
	}
	return key;
}


string cipherText(string str, string key)
{
	string cipher_text;

	for (int i = 0; i < str.size(); i++)
	{

		char x = (str[i] + key[i]) %26;
```

```
                x += 'A';

                cipher_text.push_back(x);
        }
        return cipher_text;
}


string originalText(string cipher_text, string key)
{
        string orig_text;

        for (int i = 0 ; i < cipher_text.size(); i++)
        {

                char x = (cipher_text[i] - key[i] + 26) %26;

                // convert into alphabets(ASCII)
                x += 'A';
                orig_text.push_back(x);
        }
        return orig_text;
}


int main()
{
        string str ;
        string keyword ;
   cout<<"Enter a string";
   cin>>str;
   cout<<"enter a key";
   cin>>keyword;


        string key = generateKey(str, keyword);
        string cipher_text = cipherText(str, key);

        cout << "Ciphertext : "
                << cipher_text << "\n";

        cout << "Original/Decrypted Text : "
                << originalText(cipher_text, key);
```

```
        return 0;
}
```

OUTPUT :

#2 In Java

```
import java.util.*;
public class Vigenere {

   static String generateKey(String str, String key) {
      int x = str.length();

      for (int i = 0;; i++) {
         if (x == i)
            i = 0;
         if (key.length() == str.length())
            break;
         key += (key.charAt(i));
      }
      return key;
   }

   static String cipherText(String str, String key) {
      String cipher_text = "";

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

         int x = (str.charAt(i) + key.charAt(i)) % 26;
```

```
        // convert into alphabets(ASCII)
        x += 'A';

        cipher_text += (char) (x);
    }
    return cipher_text;
}

static String originalText(String cipher_text, String key) {
    String orig_text = "";

    for (int i = 0; i < cipher_text.length() &&
            i < key.length(); i++) {

        int x = (cipher_text.charAt(i) -
                key.charAt(i) + 26) % 26;

        // convert into alphabets(ASCII)
        x += 'A';
        orig_text += (char) (x);
    }
    return orig_text;
}

static String LowerToUpper(String s) {
    StringBuffer str = new StringBuffer(s);
    for (int i = 0; i < s.length(); i++) {
        if (Character.isLowerCase(s.charAt(i))) {
            str.setCharAt(i, Character.toUpperCase(s.charAt(i)));
        }
    }
    s = str.toString();
    return s;
}

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);

    System.out.println("Enter a string\n");
    String Str = sc.nextLine();

    System.out.println("Enter a a keyword\n");
```

```
        String Keyword = sc.nextLine();

        String str = LowerToUpper(Str);
        String keyword = LowerToUpper(Keyword);

        String key = generateKey(str, keyword);
        String cipher_text = cipherText(str, key);

        System.out.println("Ciphertext : "
                + cipher_text + "\n");

        System.out.println("Original/Decrypted Text : "
                + originalText(cipher_text, key));
    }
}
```

OUTPUT :

#3 in Python

```
from ast import keyword

from pyparsing import Keyword

def generateKey(string, key):
    key = list(key)
    if len(string) == len(key):
        return(key)
    else:
        for i in range(len(string) -
                       len(key)):
```

```
            key.append(key[i % len(key)])
    return("" . join(key))


def cipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = (ord(string[i]) +
             ord(key[i])) % 26
        x += ord('A')
        cipher_text.append(chr(x))
    return("" . join(cipher_text))


def originalText(cipher_text, key):
    orig_text = []
    for i in range(len(cipher_text)):
        x = (ord(cipher_text[i]) -
             ord(key[i]) + 26) % 26
        x += ord('A')
        orig_text.append(chr(x))
    return("" . join(orig_text))


if __name__ == "__main__":
    string = input("Enter a string")
    keyword = input("Enter a keyword")

    key = generateKey(string, keyword)
    cipher_text = cipherText(string,key)
    print("Ciphertext :", cipher_text)
    print("Original/Decrypted Text :",
          originalText(cipher_text, key))
```

OUTPUT :

SELF TESTING QUESTIONS :

Q1 List down all the substitution type Cipher methods in Cryptology .

Q2 What is the difference between The Vigenere Cipher and Shift Cipher .

Q3 Give an application of Vigenere Cipher in Cybersecurity.

Q4 State drawbacks of Vigenere Cipher .

_____
_____
_____
_____
____
_____

Q5 Judging on the working and algorithm of all and only substitution Ciphers do you feel vigenere cipher is a better option for real time deployment ?

_____
_____
_____
_____
____
_____

Conclusion : So therefore we learnt vigenere cryptology in Computer Networking , studied its cryptanalysis and implemented its algorithm methodology successfully in the big three languages – C++ , Java & Python

Experiment No. 4

Aim: To use Advanced Encrytion Standard (AES) Algorithm for a practical application like User Message Encryption.

Theory: The **AES algorithm** (also known as the **Rijndael algorithm**) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits. Since the AES algorithm is considered secure, it is in the worldwide standard.

Algorithm:

Step – 1: Derive the set of round keys from the cipher key.

Step – 2: Initialize the state array with the block data (plaintext).

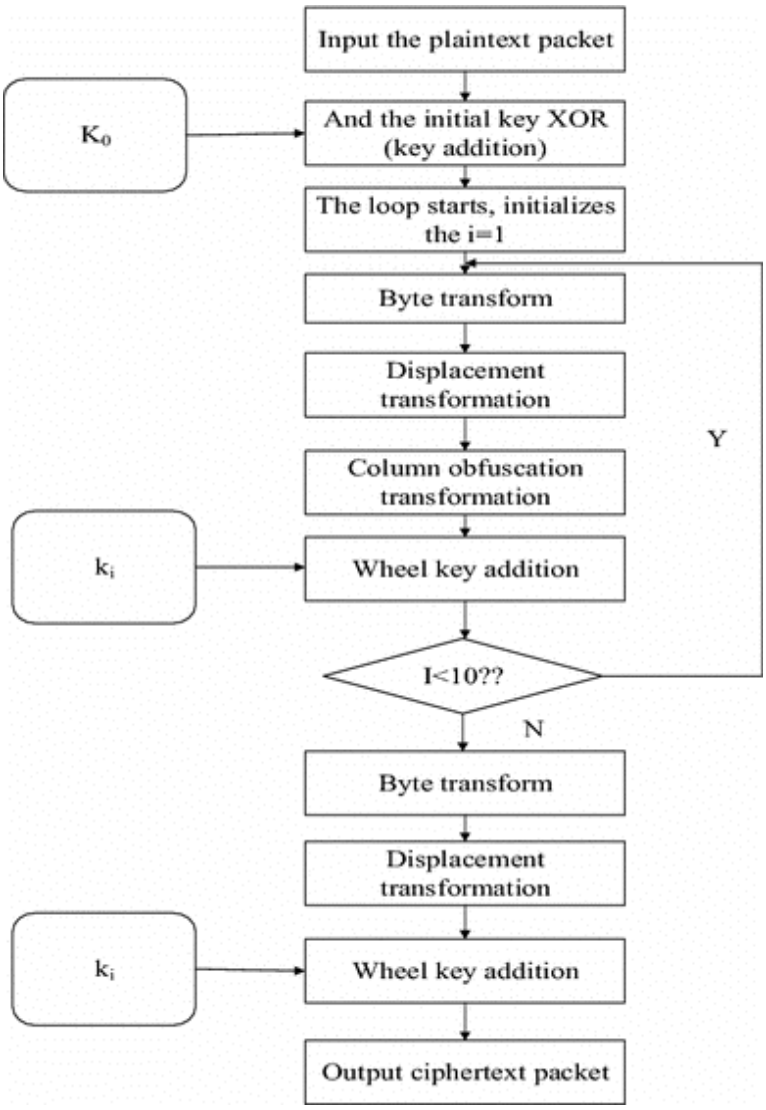
Step – 3: Add the initial round key to the starting state array.

Step – 4: Perform nine rounds of state manipulation.

Step – 5: Perform the tenth and final round of state manipulation.

Step – 6: Copy the final state array out as the encrypted data (ciphertext).

Flowchart:



Program & Codes: So we have implemented the advanced encryption technique in C++, Java 7 Python the 3 generally used programming languages.

#1 In C++ (Encryption):

```
#include "pch.h"

#include <iostream>

#include "aes.h"

#include <Windows.h>

#include "osrng.h"

using CryptoPP::AutoSeededRandomPool;
```

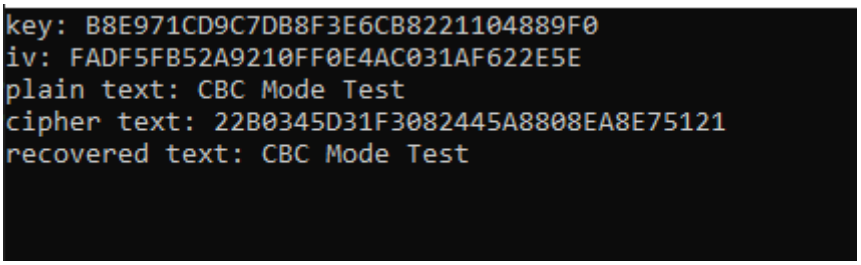
```
#include <iostream>
using std::cout;
using std::cerr;
using std::endl;
#include <string>
using std::string;
#include <cstdlib>
using std::exit;
#include "cryptlib.h"
using CryptoPP::Exception;
#include "hex.h"
using CryptoPP::HexEncoder;
using CryptoPP::HexDecoder;
#include "filters.h"
using CryptoPP::StringSink;
using CryptoPP::StringSource;
using CryptoPP::StreamTransformationFilter;
#include "aes.h"
using CryptoPP::AES;
#include "ccm.h"
using CryptoPP::CBC_Mode;
#include "assert.h"
int main(int argc, char* argv[])
{
AutoSeededRandomPool prng;
byte key[AES::DEFAULT_KEYLENGTH];
prng.GenerateBlock(key, sizeof(key));
byte iv[AES::BLOCKSIZE];
prng.GenerateBlock(iv, sizeof(iv));
string plain = "CBC Mode Test";
```

```
string cipher, encoded, recovered;
/*********************************\
\*********************************/
// Pretty print key
encoded.clear();
StringSource(key, sizeof(key), true,
new HexEncoder(
new StringSink(encoded)
) // HexEncoder
); // StringSource
cout << "key: " << encoded << endl;
// Pretty print iv
encoded.clear();
StringSource(iv, sizeof(iv), true,
new HexEncoder(
new StringSink(encoded)
) // HexEncoder
); // StringSource
cout << "iv: " << encoded << endl;
/*********************************\
\*********************************/
try
{
cout << "plain text: " << plain << endl;
CBC_Mode< AES >::Encryption e;
e.SetKeyWithIV(key, sizeof(key), iv);
// The StreamTransformationFilter removes
// padding as required.
StringSource s(plain, true,
new StreamTransformationFilter(e,
```

```
new StringSink(cipher)
) // StreamTransformationFilter
); // StringSource
#if 0
StreamTransformationFilter filter(e);
filter.Put((const byte*)plain.data(), plain.size());
filter.MessageEnd();
const size_t ret = filter.MaxRetrievable();
cipher.resize(ret);
filter.Get((byte*)cipher.data(), cipher.size());
#endif
}
catch (const CryptoPP::Exception& e)
{
cerr << e.what() << endl;
exit(1);
}
/*********************************\
\*********************************/
// Pretty print
encoded.clear();
StringSource(cipher, true,
new HexEncoder(
new StringSink(encoded)
) // HexEncoder
); // StringSource
cout << "cipher text: " << encoded << endl;
/*********************************\
\*********************************/
try
```

```
{
CBC_Mode< AES >::Decryption d;
d.SetKeyWithIV(key, sizeof(key), iv);
// The StreamTransformationFilter removes
// padding as required.
StringSource s(cipher, true,
new StreamTransformationFilter(d,
new StringSink(recovered)
) // StreamTransformationFilter
); // StringSource
#if 0
StreamTransformationFilter filter(d);
filter.Put((const byte*)cipher.data(), cipher.size());
filter.MessageEnd();
const size_t ret = filter.MaxRetrievable();
recovered.resize(ret);
filter.Get((byte*)recovered.data(), recovered.size());
#endif
cout << "recovered text: " << recovered << endl;
}
catch (const CryptoPP::Exception& e)
{
cerr << e.what() << endl;
exit(1);
}
/**********************************\
\**********************************/
return 0;
}
```

Output:



2. **Java:**

```
import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.SecretKeySpec;

import java.nio.charset.StandardCharsets;

import java.security.InvalidAlgorithmParameterException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import java.security.spec.KeySpec;

import java.util.Base64;

import javax.crypto.BadPaddingException;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;
```

```
public class AESExample
{
   /* Private variable declaration */
   private static final String SECRET_KEY = "123456789";
   private static final String SALTVALUE = "abcdefg";

   /* Encryption Method */
   public static String encrypt(String strToEncrypt)
   {
   try
   {
     /* Declare a byte array. */
     byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
     IvParameterSpec ivspec = new IvParameterSpec(iv);
     /* Create factory for secret keys. */
     SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
     /* PBEKeySpec class implements KeySpec interface. */
     KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(), SALTVALUE.getBytes(),
65536, 256);
     SecretKey tmp = factory.generateSecret(spec);
     SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
     Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
     cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);
     /* Retruns encrypted value. */
     return Base64.getEncoder()
```

```
.encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_8)));
    }
    catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e)
    {
      System.out.println("Error occured during encryption: " + e.toString());
    }
    return null;
    }

    /* Decryption Method */
    public static String decrypt(String strToDecrypt)
    {
    try
    {
      /* Declare a byte array. */
      byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
      IvParameterSpec ivspec = new IvParameterSpec(iv);
      /* Create factory for secret keys. */
      SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
      /* PBEKeySpec class implements KeySpec interface. */
      KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(), SALTVALUE.getBytes(), 65536, 256);
      SecretKey tmp = factory.generateSecret(spec);
      SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
```

```
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
    cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);
    /* Retruns decrypted value. */
    return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
  }
  catch (InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | InvalidKeySpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException e)
  {
    System.out.println("Error occured during decryption: " + e.toString());
  }
  return null;
  }
  /* Driver Code */
  public static void main(String[] args)
  {
    /* Message to be encrypted. */
    String originalval = "AES Encryption";
    /* Call the encrypt() method and store result of encryption. */
    String encryptedval = encrypt(originalval);
    /* Call the decrypt() method and store result of decryption. */
    String decryptedval = decrypt(encryptedval);
    /* Display the original message, encrypted message and decrypted message on the console. */
    System.out.println("Original value: " + originalval);
    System.out.println("Encrypted value: " + encryptedval);
```

```
        System.out.println("Decrypted value: " + decryptedval);
    }
}
```

Output:

```
Decrypted value: AES Encryption
```

3. Python:

```
import base64
import hashlib
from Crypto.Cipher import AES
from Crypto import Random
BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) %
BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]
def encrypt(plain_text, key):
    private_key = hashlib.sha256(key.encode("utf-8")).digest()
    plain_text = pad(plain_text)
    print("After padding:", plain_text)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return base64.b64encode(iv + cipher.encrypt(plain_text))
def decrypt(cipher_text, key):
    private_key = hashlib.sha256(key.encode("utf-8")).digest()
    cipher_text = base64.b64decode(cipher_text)
    iv = cipher_text[:16]
    cipher = AES.new(private_key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(cipher_text[16:]))
message=input("Enter message to encrypt: ");
key = input("Enter encryption key: ")
encrypted_msg = encrypt(message, key)
```

```
print("Encrypted Message:", encrypted_msg)
decrypted_msg = decrypt(encrypted_msg, key)
print("Decrypted Message:", bytes.decode(decrypted_msg))
```

Output:

```
Enter message to encrypt: hello
Enter encryption key: 12345
After padding: hello

Encrypted Message: b'r3V0A0Ssjw/4ZOKL42/hWSQOLKy7lt9bOVt7D75RA3E='
Decrypted Message: hello
```

Conclusion: We have successfully implemented Advanced Encryption Standard (AES) Algorithm using various modes of operations.

Experiment No. 5

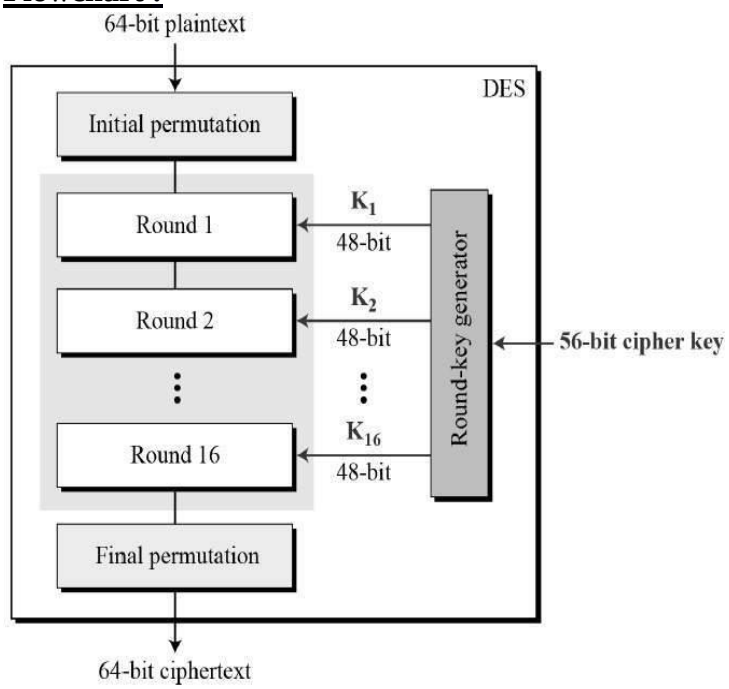
Aim : To use Data Encryption Standard (DES) Algorithm for a practical application like User Message Encryption.

Theory :

Data encryption standard (DES) has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found slightly on the decline. DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bit

ALGORITHM:
STEP-1: Read the 64-bit plain text.
STEP-2: Split it into two 32-bit blocks and store it in two different arrays.
STEP-3: Perform XOR operation between these two arrays.
STEP-4: The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.
STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

Flowchart :



Program & Codes : So we have implemented The Shift Encryption technique in C++ , Java and Python the 3 generally used programming languages ;

#1 In C++

```
#include <bits/stdc++.h>
using namespace std;
```

```
string hex2bin(string s)
{
	// hexadecimal to binary conversion
	unordered_map<char, string> mp;
	mp['0'] = "0000";
	mp['1'] = "0001";
	mp['2'] = "0010";
	mp['3'] = "0011";
	mp['4'] = "0100";
	mp['5'] = "0101";
	mp['6'] = "0110";
	mp['7'] = "0111";
	mp['8'] = "1000";
	mp['9'] = "1001";
	mp['A'] = "1010";
	mp['B'] = "1011";
	mp['C'] = "1100";
	mp['D'] = "1101";
	mp['E'] = "1110";
	mp['F'] = "1111";
	string bin = "";
	for (int i = 0; i < s.size(); i++) {
		bin += mp[s[i]];
	}
	return bin;
}
string bin2hex(string s)
{
	// binary to hexadecimal conversion
	unordered_map<string, string> mp;
	mp["0000"] = "0";
	mp["0001"] = "1";
	mp["0010"] = "2";
	mp["0011"] = "3";
	mp["0100"] = "4";
	mp["0101"] = "5";
	mp["0110"] = "6";
	mp["0111"] = "7";
	mp["1000"] = "8";
	mp["1001"] = "9";
	mp["1010"] = "A";
	mp["1011"] = "B";
	mp["1100"] = "C";
	mp["1101"] = "D";
```

```
	mp["1110"] = "E";
	mp["1111"] = "F";
	string hex = "";
	for (int i = 0; i < s.length(); i += 4) {
		string ch = "";
		ch += s[i];
		ch += s[i + 1];
		ch += s[i + 2];
		ch += s[i + 3];
		hex += mp[ch];
	}
	return hex;
}

string permute(string k, int* arr, int n)
{
	string per = "";
	for (int i = 0; i < n; i++) {
		per += k[arr[i] - 1];
	}
	return per;
}

string shift_left(string k, int shifts)
{
	string s = "";
	for (int i = 0; i < shifts; i++) {
		for (int j = 1; j < 28; j++) {
			s += k[j];
		}
		s += k[0];
		k = s;
		s = "";
	}
	return k;
}

string xor_(string a, string b)
{
	string ans = "";
	for (int i = 0; i < a.size(); i++) {
		if (a[i] == b[i]) {
			ans += "0";
		}
```

```
                else {
                        ans += "1";
                }
        }
        return ans;
}
string encrypt(string pt, vector<string> rkb, vector<string> rk)
{
        // Hexadecimal to binary
        pt = hex2bin(pt);

        // Initial Permutation Table
        int initial_perm[64] = { 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 };
        // Initial Permutation
        pt = permute(pt, initial_perm, 64);
        cout << "After initial permutation: " << bin2hex(pt) << endl;

        // Splitting
        string left = pt.substr(0, 32);
        string right = pt.substr(32, 32);
        cout << "After splitting: L0=" << bin2hex(left)
                << " R0=" << bin2hex(right) << endl;

        // Expansion D-box Table
        int exp_d[48] = { 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 };

        // S-box Table
        int s[8][4][16] = { { 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 } };

// Straight Permutation Table
int per[32] = { 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 };

cout << endl;
for (int i = 0; i < 16; i++) {
        // Expansion D-box
        string right_expanded = permute(right, exp_d, 48);

        // XOR RoundKey[i] and right_expanded
```

```
		string x = xor_(rkb[i], right_expanded);

		// S-boxes
		string op = "";
		for (int i = 0; i < 8; i++) {
			int row = 2 * int(x[i * 6] - '0') + int(x[i * 6 + 5] - '0');
			int col = 8 * int(x[i * 6 + 1] - '0') + 4 * int(x[i * 6 + 2] - '0') + 2 * int(x[i * 6
+ 3] - '0') + int(x[i * 6 + 4] - '0');
			int val = s[i][row][col];
			op += char(val / 8 + '0');
			val = val % 8;
			op += char(val / 4 + '0');
			val = val % 4;
			op += char(val / 2 + '0');
			val = val % 2;
			op += char(val + '0');
		}
		// Straight D-box
		op = permute(op, per, 32);

		// XOR left and op
		x = xor_(op, left);

		left = x;

		// Swapper
		if (i != 15) {
			swap(left, right);
		}
		cout << "Round " << i + 1 << " " << bin2hex(left) << " "
			<< bin2hex(right) << " " << rk[i] << endl;
	}

	// Combination
	string combine = left + right;

	// Final Permutation Table
	int final_perm[64] = { 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 };

        // Final Permutation
        string cipher = bin2hex(permute(combine, final_perm, 64));
        return cipher;
}
int main()
{
        // pt is plain text
        string pt, key;
        /*cout<<"Enter plain text(in hexadecimal): ";
        cin>>pt;
        cout<<"Enter key(in hexadecimal): ";
        cin>>key;*/

        pt = "123456ABCD132536";
        key = "AABB09182736CCDD";
        // Key Generation

        // Hex to binary
        key = hex2bin(key);

        // Parity bit drop table
        int keyp[56] = { 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 };

        // getting 56 bit key from 64 bit using the parity bits
        key = permute(key, keyp, 56); // key without parity

        // Number of bit shifts
        int shift_table[16] = { 1, 1, 2, 2,
                                                2, 2, 2, 2,
                                                1, 2, 2, 2,
                                                2, 2, 2, 1 };

        // Key- Compression Table
        int key_comp[48] = { 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 };

        // Splitting
        string left = key.substr(0, 28);
        string right = key.substr(28, 28);

        vector<string> rkb; // rkb for RoundKeys in binary
        vector<string> rk; // rk for RoundKeys in hexadecimal
        for (int i = 0; i < 16; i++) {
                // Shifting
                left = shift_left(left, shift_table[i]);
                right = shift_left(right, shift_table[i]);

                // Combining
                string combine = left + right;

                // Key Compression
                string RoundKey = permute(combine, key_comp, 48);

                rkb.push_back(RoundKey);
                rk.push_back(bin2hex(RoundKey));
        }

        cout << "\nEncryption:\n\n";
        string cipher = encrypt(pt, rkb, rk);
        cout << "\nCipher Text: " << cipher << endl;

        cout << "\nDecryption\n\n";
        reverse(rkb.begin(), rkb.end());
        reverse(rk.begin(), rk.end());
        string text = encrypt(cipher, rkb, rk);
        cout << "\nPlain Text: " << text << endl;
}
```

Output:

#2 Java 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-hePermutation 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:

#3 Python

```
# Hexadecimal to binary conversion
def hex2bin(s):
        mp = {'0' : "0000",
              '1' : "0001",
              '2' : "0010",
              '3' : "0011",
              '4' : "0100",
              '5' : "0101",
              '6' : "0110",
              '7' : "0111",
              '8' : "1000",
              '9' : "1001",
              'A' : "1010",
              'B' : "1011",
              'C' : "1100",
              'D' : "1101",
              'E' : "1110",
              'F' : "1111" }
        bin = ""
        for i in range(len(s)):
              bin = bin + mp[s[i]]
        return bin

# Binary to hexadecimal conversion
def bin2hex(s):
        mp = {"0000" : '0',
              "0001" : '1',
              "0010" : '2',
              "0011" : '3',
              "0100" : '4',
              "0101" : '5',
              "0110" : '6',
              "0111" : '7',
              "1000" : '8',
              "1001" : '9',
              "1010" : 'A',
              "1011" : 'B',
              "1100" : 'C',
              "1101" : 'D',
              "1110" : 'E',
              "1111" : 'F' }
        hex = ""
        for i in range(0,len(s),4):
```

```
                ch = ""
                ch = ch + s[i]
                ch = ch + s[i + 1]
                ch = ch + s[i + 2]
                ch = ch + s[i + 3]
                hex = hex + mp[ch]

        return hex

# Binary to decimal conversion
def bin2dec(binary):

        binary1 = binary
        decimal, i, n = 0, 0, 0
        while(binary != 0):
                dec = binary % 10
                decimal = decimal + dec * pow(2, i)
                binary = binary//10
                i += 1
        return decimal

# Decimal to binary conversion
def dec2bin(num):
        res = bin(num).replace("0b", "")
        if(len(res)%4 != 0):
                div = len(res) / 4
                div = int(div)
                counter =(4 * (div + 1)) - len(res)
                for i in range(0, counter):
                        res = '0' + res
        return res

# Permute function to rearrange the bits
def permute(k, arr, n):
        permutation = ""
        for i in range(0, n):
                permutation = permutation + k[arr[i] - 1]
        return permutation

# shifting the bits towards left by nth shifts
def shift_left(k, nth_shifts):
        s = ""
        for i in range(nth_shifts):
                for j in range(1,len(k)):
```

```
                s = s + k[j]
            s = s + k[0]
            k = s
            s = ""
        return k

# calculating xow of two strings of binary number a and b
def xor(a, b):
        ans = ""
        for i in range(len(a)):
            if a[i] == b[i]:
                ans = ans + "0"
            else:
                ans = ans + "1"
        return ans

# Table of Position of 64 bits at initial level: Initial Permutation Table
initial_perm = [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]

# Expansion D-box Table
exp_d = [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
per = [ 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
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] ] ]

# Final Permutation Table
final_perm = [ 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 ]

def encrypt(pt, rkb, rk):
        pt = hex2bin(pt)

        # Initial Permutation
        pt = permute(pt, initial_perm, 64)
        print("After initial permutation", bin2hex(pt))

        # Splitting
        left = pt[0:32]
        right = pt[32:64]
        for i in range(0, 16):
                # Expansion D-box: Expanding the 32 bits data into 48 bits
                right_expanded = permute(right, exp_d, 48)

                # XOR RoundKey[i] and right_expanded
                xor_x = xor(right_expanded, rkb[i])

                # S-boxex: substituting the value from s-box table by calculating row and column
                sbox_str = ""
                for j in range(0, 8):
                        row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
                        col = bin2dec(int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] + xor_x[j * 6 + 3] +
xor_x[j * 6 + 4]))
                        val = sbox[j][row][col]
                        sbox_str = sbox_str + dec2bin(val)

                # Straight D-box: After substituting rearranging the bits
                sbox_str = permute(sbox_str, per, 32)

                # XOR left and sbox_str
                result = xor(left, sbox_str)
                left = result

                # Swapper
                if(i != 15):
                        left, right = right, left
                print("Round ", i + 1, " ", bin2hex(left), " ", bin2hex(right), " ", rk[i])
```

```
        # Combination
        combine = left + right

        # Final permutation: final rearranging of bits to get cipher text
        cipher_text = permute(combine, final_perm, 64)
        return cipher_text

pt = "123456ABCD132536"
key = "AABB09182736CCDD"

# Key generation
# --hex to binary
key = hex2bin(key)

# --parity bit drop table
keyp = [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 ]

# getting 56 bit key from 64 bit using the parity bits
key = permute(key, keyp, 56)

# Number of bit shifts
shift_table = [1, 1, 2, 2,
                        2, 2, 2, 2,
                        1, 2, 2, 2,
                        2, 2, 2, 1 ]

# Key- Compression Table : Compression of key from 56 bits to 48 bits
key_comp = [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 ]
```

```
# Splitting
left = key[0:28] # rkb for RoundKeys in binary
right = key[28:56] # rk for RoundKeys in hexadecimal

rkb = []
rk = []
for i in range(0, 16):
        # Shifting the bits by nth shifts by checking from shift table
        left = shift_left(left, shift_table[i])
        right = shift_left(right, shift_table[i])

        # Combination of left and right string
        combine_str = left + right

        # Compression of key from 56 to 48 bits
        round_key = permute(combine_str, key_comp, 48)

        rkb.append(round_key)
        rk.append(bin2hex(round_key))

print("Encryption")
cipher_text = bin2hex(encrypt(pt, rkb, rk))
print("Cipher Text : ",cipher_text)

print("Decryption")
rkb_rev = rkb[::-1]
rk_rev = rk[::-1]
text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))
print("Plain Text : ",text)
```

Output:

SELF TESTING QUESTIONS :

Q1 What is the main vulnerability of DES?

Q2 Which mode of operation is used in DES?

Q3 What is strength of DES?

Q4 What is the avalanche effect in DES?

Q5 What is expansion table in DES?

Conclusion : DES uses the same key to encrypt and decrypt a message, so both the sender and the receiver must know and use the same private key. DES was once the go-to, symmetric key algorithm for the encryption of electronic data, but it has been superseded by the more secure Advanced Encryption Standard (AES) algorithm.

Experiment No. 6

Aim: To use Hashing Techniques: HMAC and SHA for a practical application like User Message Encryption.

Theory:

A hashing algorithm is a mathematical function that garbles data and makes it unreadable.

Hashing algorithms are one-way programs, so the text can't be unscrambled and decoded by anyone else. And that's the point. Hashing protects data at rest, so even if someone gains access to your server, the items stored there remain unreadable.

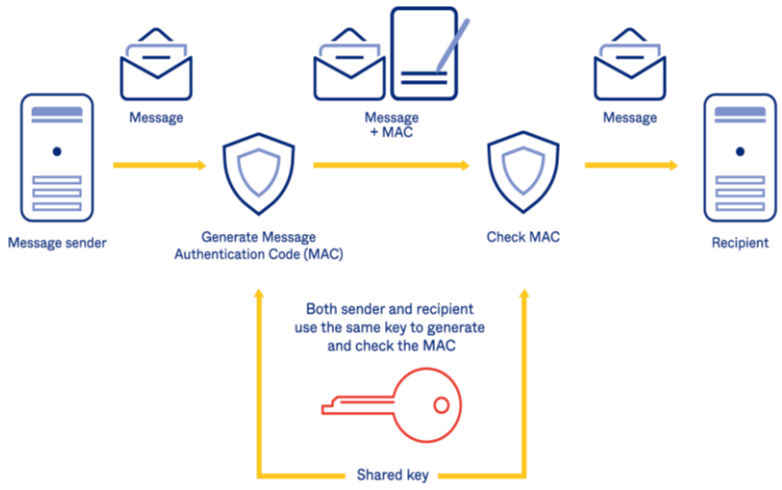
Hashing can also help you prove that data isn't adjusted or altered after the author is finished with it. And some people use hashing to help them make sense of reams of data.

Hash Function is a function that has a huge role in making a System Secure as it converts normal data given to it as an irregular value of fixed length. We can imagine it to be a Shaker in our homes.

When we put data into this function it outputs an irregular value. The Irregular value it outputs is known as the **"Hash Value"**. Hash Values are simply numbers but are often written in Hexadecimal. Computers manage values as Binary. The hash value is also data and is often managed in Binary.

HMAC:

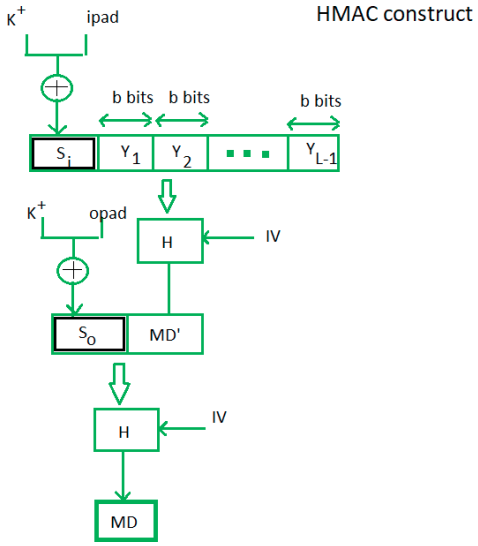
HMAC algorithm stands for Hashed or Hash-based Message Authentication Code. It is a result of work done on developing a MAC derived from cryptographic hash functions. HMAC is a great resistant to cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC and thus is more secure than any other authentication code. RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security. The FIPS 198 NIST standard has also issued HMAC.



Objectives –

- As the Hash Function, HMAC is also aimed to be one way, i.e, easy to generate output from input but complex the other way round.
- It aims at being less affected by collisions than the hash functions.
- HMAC reuses the algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- HMAC tries to handle the Keys in a more simple manner.

- **HMAC algorithm –**
  The working of HMAC starts with taking a message M containing blocks of length *b* bits. An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a temporary message-digest MD'. MD' again is appended to an output signature and the whole is applied a hash function again, the result is our final message digest MD.
- Here is a simple structure of HMAC:

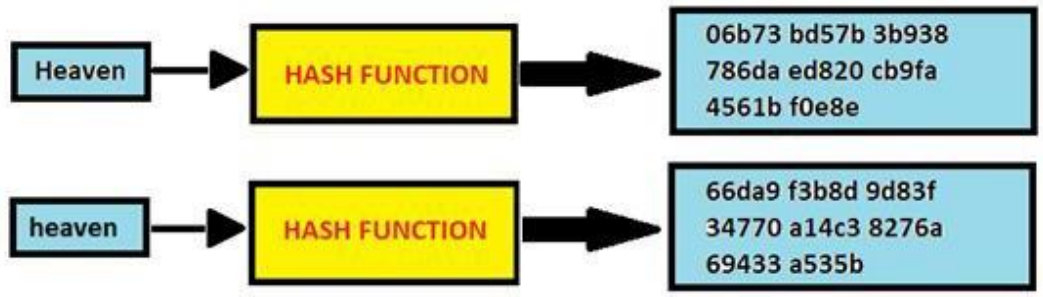


Here, H stands for Hashing function,
M is the original message
Si and So are input and output signatures respectively,
Yi is the ith block in original message M, where I ranges from [1, L)
L = the count of blocks in M
K is the secret key used for hashing
IV is an initial vector (some constant)
The generation of input signature and output signature *Si* and *So* respectively.

SHA

SHA stands for secure hashing algorithm. SHA is a modified version of MD5 and used for hashing data and certificates. A hashing algorithm shortens the input data into a smaller form that cannot be understood by using bitwise operations, modular additions, and compression functions. You may be wondering, can hashing be cracked or decrypted? Hashing is similar to encryption, the only difference between hashing and encryption is that hashing is one-way, meaning once the data is hashed, the resulting hash digest cannot be cracked, unless a brute force attack is used.

SHA works in such a way even if a single character of the message changed, then it will generate a different hash. For example, hashing of two similar, but different messages i.e., Heaven and heaven is different. However, there is only a difference of a capital and small letter.



Different SHA Forms

When learning about SHA forms, several different types of SHA are referenced. Examples of SHA names used are SHA-1, SHA-2, SHA-256, SHA-512, SHA-224, and SHA-384, but in actuality, there are only two types: SHA-1 and SHA-2. The other larger numbers, like SHA-256, are just versions of SHA-2 that note the bit lengths of the SHA-2. SHA-1 was the original secure hashing algorithm, returning a 160-bit hash digest after hashing.

Program and code:

SHA-256

- **Java:**

```
import java.math.BigInteger;

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

// Java program to calculate SHA hash value

class GFG2 {

    public static byte[] getSHA(String input) throws NoSuchAlgorithmException

    {

        // Static getInstance method is called with hashing SHA
```

```
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        // digest() method called
        // to calculate message digest of an input
        // and return array of byte
        return md.digest(input.getBytes(StandardCharsets.UTF_8));
    }

    public static String toHexString(byte[] hash)
    {

        // Convert byte array into signum representation
        BigInteger number = new BigInteger(1, hash);
        // Convert message digest into hex value
        StringBuilder hexString = new StringBuilder(number.toString(16));
        // Pad with leading zeros
        while (hexString.length() < 64)
        {

            hexString.insert(0, '0');
        }


        return hexString.toString();
    }

    // Driver code
```

```
    public static void main(String args[])
      {
            try
            {
                  System.out.println("HashCode Generated by SHA-256 for:\n");
                  String s1 = "Batchfour";
                  System.out.println("\n" + s1 + " : " + toHexString(getSHA(s1)));
                  String s2 = "\n hello world";
                  System.out.println("\n" + s2 + " : " + toHexString(getSHA(s2)));
                  String s3 = "K1t4fo0V";
                  System.out.println("\n" + s3 + " : " + toHexString(getSHA(s3)));
            }

            // For specifying wrong message digest algorithms
            catch (NoSuchAlgorithmException e) {
                  System.out.println("Exception thrown for incorrect algorithm: " + e);
            }

      }
}
```

Output:

```
HashCode Generated by SHA-256 for:

Batchfour : 8f53ca117a48ba3a0e78fd9889fb103eca5bc9554b6d197e0e86523889735105
hello world : 7b80c4913687d5bda92d876fb8252ddd74592c7815e4f1c7d976f40538608835
K1t4fo0V : 0a979e43f4874eb24b740c0157994e34636eed0425688161cc58e8b26b1dcf4e
```

- **Python:**

```
import hashlib

str = "TEITBATCHFOUR"

result = hashlib.sha256(str.encode())

print("The hexadecimal equivalent of SHA256 is : ")

print(result.hexdigest())


print ("\r")


str = "TEITBATCHFOUR"

result = hashlib.sha384(str.encode())

print("\nThe hexadecimal equivalent of SHA384 is : ")

print(result.hexdigest())


print ("\r")


str = "TEITBATCHFOUR"

result = hashlib.sha224(str.encode())

print("\nThe hexadecimal equivalent of SHA224 is : ")

print(result.hexdigest())
```

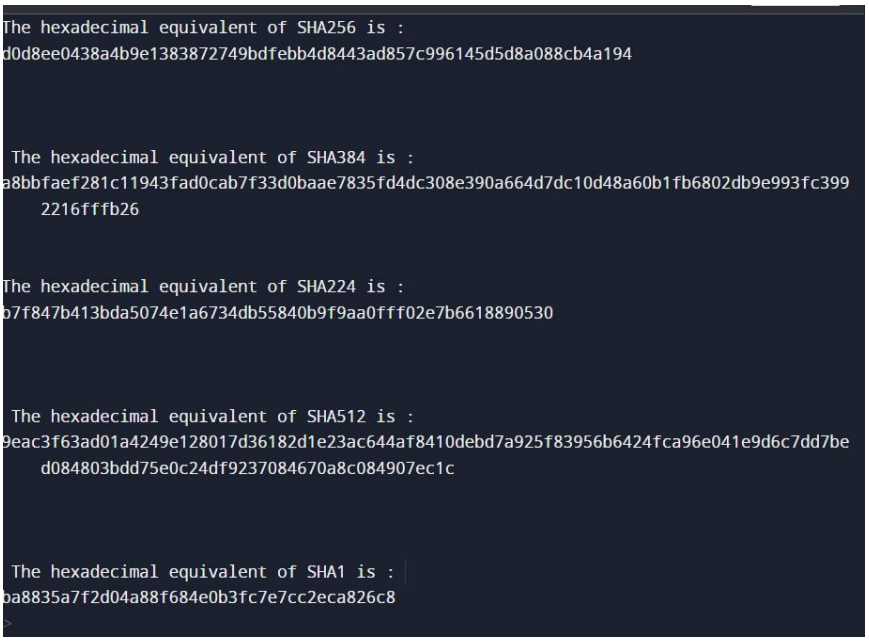
```
print ("\r")


str = "TEITBATCHFOUR"
result = hashlib.sha512(str.encode())
print("\nThe hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

print ("\r")

str = "TEITBATCHFOUR"
result = hashlib.sha1(str.encode())
print("\nThe hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```

Output:



- **C++:**

```
#include
<cstring>
#include
<fstream>
#include
"sha256.h"
const unsigned int SHA256::sha256_k[64] = //UL = uint32
        {0x428a2f98,        0x71374491,        0xb5c0fbcf,
         0xe9b5dba5,        0x3956c25b,        0x59f111f1,
         0x923f82a4,        0xab1c5ed5,        0xd807aa98,
         0x12835b01, 0x243185be, 0x550c7dc3,
         0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,

         0xe49b69c1,        0xefbe4786,        0x0fc19dc6,
         0x240ca1cc,        0x2de92c6f,        0x4a7484aa,
         0x5cb0a9dc,        0x76f988da,        0x983e5152,
         0xa831c66d,        0xb00327c8,        0xbf597fc7,
         0xc6e00bf3,        0xd5a79147,        0x06ca6351,
         0x14292967,
         0x27b70a85,        0x2e1b2138,        0x4d2c6dfc,
         0x53380d13,        0x650a7354,        0x766a0abb,
         0x81c2c92e, 0x92722c85,
```

```
        0xa2bfe8a1,     0xa81a664b,     0xc24b8b70,
        0xc76c51a3,     0xd192e819,     0xd6990624,
        0xf40e3585, 0x106aa070,
        0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
        0x391c0cb3,     0x4ed8aa4a,     0x5b9cca4f,
        0x682e6ff3,     0x748f82ee,     0x78a5636f,
        0x84c87814, 0x8cc70208,
        0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2};
void SHA256::transform(const unsigned char *message, unsigned int block_nb)
{
   uint32
   w[64];
   uint32
   wv[8];
   uint32  t1,
   t2;
   const     unsigned     char
   *sub_block; int i;
   int j;
   for (i = 0; i < (int) block_nb; i++)
      { sub_block = message + (i <<
      6); for (j = 0; j < 16; j++) {
         SHA2_PACK32(&sub_block[j << 2], &w[j]);
```

```
}

for (j = 16; j < 64; j++) {
    w[j] =  SHA256_F4(w[j -  2]) + w[j -  7] + SHA256_F3(w[j - 15]) + w[j - 16];
}

for (j = 0; j < 8; j++)
    { wv[j] = m_h[j];
}

for (j = 0; j < 64; j++) {
    t1 = wv[7] + SHA256_F2(wv[4]) + SHA2_CH(wv[4], wv[5], wv[6])
        + sha256_k[j] + w[j];
    t2 = SHA256_F1(wv[0]) + SHA2_MAJ(wv[0], wv[1], wv[2]);
    wv[7] = wv[6];
    wv[6] = wv[5];
    wv[5] = wv[4];
    wv[4] = wv[3] + t1;
    wv[3] = wv[2];
    wv[2] = wv[1];
    wv[1] = wv[0];
    wv[0] = t1 + t2;
}

for (j = 0; j < 8; j++)
```

```
            { m_h[j] += wv[j];
        }
    }
}
void SHA256::init()
{
    m_h[0] = 0x6a09e667;
    m_h[1] = 0xbb67ae85;
    m_h[2] = 0x3c6ef372;
    m_h[3] = 0xa54ff53a;
    m_h[4] = 0x510e527f;
    m_h[5] = 0x9b05688c;
    m_h[6] = 0x1f83d9ab;
    m_h[7] = 0x5be0cd19;
    m_len = 0;
    m_tot_len = 0;
}
void SHA256::update(const unsigned char *message, unsigned int len)
{
    unsigned int block_nb;
```

```
    unsigned int new_len, rem_len, tmp_len;
    const unsigned char *shifted_message;
    tmp_len = SHA224_256_BLOCK_SIZE -
    m_len; rem_len = len < tmp_len ? len :
    tmp_len; memcpy(&m_block[m_len], message,
    rem_len); if (m_len + len <
    SHA224_256_BLOCK_SIZE) {
        m_len += len;
        return;
    }

    new_len = len - rem_len;

    block_nb = new_len / SHA224_256_BLOCK_SIZE;
    shifted_message = message + rem_len;
    transform(m_block, 1);
    transform(shifted_message,   block_nb);
    rem_len = new_len % SHA224_256_BLOCK_SIZE;
    memcpy(m_block, &shifted_message[block_nb << 6], rem_len);
    m_len = rem_len;
    m_tot_len += (block_nb + 1) << 6;
}

void SHA256::final(unsigned char *digest)
{
```

```
    unsigned int block_nb;
    unsigned int pm_len;
    unsigned int len_b;
    int i;
    block_nb = (1 + ((SHA224_256_BLOCK_SIZE - 9)
                     < (m_len % SHA224_256_BLOCK_SIZE)));
    len_b = (m_tot_len + m_len) << 3;
    pm_len = block_nb << 6;
    memset(m_block + m_len, 0, pm_len - m_len);
    m_block[m_len] = 0x80;
    SHA2_UNPACK32(len_b, m_block + pm_len -
    4); transform(m_block, block_nb);
    for (i = 0 ; i < 8; i++) {
        SHA2_UNPACK32(m_h[i], &digest[i << 2]);


    }
}
std::string sha256(std::string input)
{
    unsigned char digest[SHA256::DIGEST_SIZE];
    memset(digest,0,SHA256::DIGEST_SIZE);
```

```
    SHA256 ctx = SHA256();
    ctx.init();
    ctx.update( (unsigned char*)input.c_str(), input.length());
    ctx.final(digest);
    char
    buf[2*SHA256::DIGEST_SIZE+1];
    buf[2*SHA256::DIGEST_SIZE] = 0;
    for (int i = 0; i < SHA256::DIGEST_SIZE; i++)
        sprintf(buf+i*2, "%02x", digest[i]);
    return std::string(buf);
}
```

Conclusion:

We have successfully implemented Hashing Techniques: HMAC and SHA for practical applications like User Message Encryption.

Experiment No. 7

AIM: To study and implement RSA asymmetric cryptography key algorithms.

THEORY: RSA algorithm is 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 Private key is kept private.

An example of asymmetric cryptography :

1. A client (for example browser) sends its public key to the server and requests for some data.

2.The server encrypts the data using client's public key and sends the encrypted data.Client receives this data and decrypts it.

3.Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser.

The idea! 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 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. But till now it seems to be an infeasible task.

The RSA Algorithm

1.Choose two different large random prime numbers p and q

2.Calculate n = p q n is the modulus for the public key and the private keys

3.Calculate $\phi (n) = (p - 1) (q - 1)$

4.Choose an integer k such that $1 < k < \phi (n)$ and k is co-prime to $\phi (n)$: k and $\phi (n)$

share no factors other than 1; gcd (k, ϕ (n))= 1.

5.k is released as the public key exponent

6.Compute d to satisfy the d k ≡ 1 (mod ϕ (n)) i.e.: d k = 1 + x ϕ (n) for some integer x

7.d is kept as the private key exponent.

Program and code:

//1.Python

```
#include<stdio.h>
#include<math.h>
// Returns gcd of a and b
int gcd(int a, int h)
{
int temp;
while (1)
{
temp = a%h;
if (temp == 0)
return h;
a = h;
h = temp;
}
}
// Code to demonstrate RSA algorithm
int main()
{
// Two random prime numbers
```

```
double p = 3;
double q = 7;
// First part of public key:
double n = p*q;
// Finding other part of public key.
// e stands for encrypt
double e = 2;
double phi = (p-1)*(q-1);
while (e < phi)
{
// e must be co-prime to phi and
// smaller than phi.
if (gcd(e, phi)==1)
break;
else
e++;
}
// Private key (d stands for decrypt)
// choosing d such that it satisfies
// d*e = 1 + k * totient
int k = 2; // A constant value
double d = (1 + (k*phi))/e;
// Message to be encrypted
double msg = 20;
printf("Message data = %lf", msg);
```

```
// Encryption c = (msg ^ e) % n
double c = pow(msg, e);
c = fmod(c, n);
printf("\nEncrypted data = %lf", c);
// Decryption m = (c ^ d) % n
double m = pow(c, d);
m = fmod(m, n);
printf("\nOriginal Message Sent = %lf", m);
return 0;
}
```

Output:

```
Message data = 12.000000
Encrypted data = 3.000000
Original Message Sent = 12.000000
```

//2 java

```
// Java Program to Implement the RSA Algorithm
import java.math.*;
import java.util.*;
class RSA {
public static void main(String args[])
{
int p, q, n, z, d = 0, e, i;
// The number to be encrypted and decrypted
```

```
int msg = 12;
double c;
BigInteger msgback;
// 1st prime number p
p = 3;
// 2nd prime number q
q = 11;
n = p * q;
z = (p - 1) * (q - 1);
System.out.println("the value of z = " + z);
for (e = 2; e < z; e++) {
// e is for public key exponent
if (gcd(e, z) == 1) {
break;
}
}
System.out.println("the value of e = " + e);
for (i = 0; i <= 9; i++) {
int x = 1 + (i * z);
// d is for private key exponent
if (x % e == 0) {
d = x / e;
break;
}
}
```

```
System.out.println("the value of d = " + d);
c = (Math.pow(msg, e)) % n;
System.out.println("Encrypted message is : " + c);
// converting int value of n to BigInteger
BigInteger N = BigInteger.valueOf(n);
// converting float value of c to BigInteger
BigInteger C = BigDecimal.valueOf(c).toBigInteger();
msgback = (C.pow(d)).mod(N);
System.out.println("Decrypted message is : "
+ msgback);
}
static int gcd(int e, int z)
{
if (e == 0)
return z;
else
return gcd(z % e, e);
}
}
```

Output:

The value of Z = 20

The value of e = 3

The value of d = 7

Encrypted message is : 12.0

Decrypted message is : 12

//3 RSA.cpp

```
#include<iostream>
#include<math.h>
using namespace std;
// find gcd
int gcd(int a, int b) {
int t;
while(1) {
t= a%b;
if(t==0)
return b;
a = b;
b= t;
}
}
int main() {
//2 random prime numbers
double p = 13;
double q = 11;
double n=p*q;//calculate n
double track;
double phi= (p-1)*(q-1);//calculate phi
//public key
```

```
//e stands for encrypt
double e=7;
//for checking that 1 < e < phi(n) and gcd(e, phi(n)) = 1; i.e., e and phi(n) are coprime.
while(e<phi) {
track = gcd(e,phi);
if(track==1)
break;
else
e++;
}
//private key
//d stands for decrypt
//choosing d such that it satisfies d*e = 1 mod phi
double d1=1/e;
double d=fmod(d1,phi);
double message = 9;
double c = pow(message,e); //encrypt the message
double m = pow(c,d);
c=fmod(c,n);
m=fmod(m,n);
cout<<"Original Message = "<<message;
cout<<"\n"<<"p = "<<p;
cout<<"\n"<<"q = "<<q;
cout<<"\n"<<"n = pq = "<<n;
cout<<"\n"<<"phi = "<<phi;
```

```
cout<<"\n"<<"e = "<<e;
cout<<"\n"<<"d = "<<d;
cout<<"\n"<<"Encrypted message = "<<c;
cout<<"\n"<<"Decrypted message = "<<m;
return 0;
}
```

Output:

```
p = 13
q = 11
n = pq = 143
phi = 120
e = 7
d = 0.142857
Original Message = 9
Encrypted message = 48
Decrypted message = 9
```

Conclution: So therefore we learnt RSA asymmetric cryptography algorithm successfully in the big three languages c++, java and python.

1. Hashed message is signed by a sender using which type of key. ?

_____
_____
_____
_____

2. Using the RSA public key cryptosystem, if p = 13, q = 31 and d = 7, then the value of 'e' is ?

3. Where RSA algorithm is used in real life?

4. What type of algorithm is RSA?

5. Why RSA is secure?

Experiment No. 8

AIM: to understand the functions and applications of a keylogger.

THEORY:

Keylogger:

A keylogger called a keystroke logger or keyboard capture is a type of surveillance technology used to monitor and record each keystroke on a specific computer. Keylogger software is also available for use on smartphones, such as the Apple iPhone and Android devices. Cybercriminals often use Keyloggers as a spyware tool to steal personally identifiable information (PII), login credentials, and sensitive enterprise data.

The basic functionality of a keylogger is that it records what you type and, in one way or another, reports that information back to whoever installed it on your computer. (We'll go into the details in a moment.) Since much of your interactions with your computer—and with the people you communicate with via your computer—are mediated through your keyboard, the range of potential information the snooper can acquire by this method is truly vast, from passwords and banking information to private correspondence.

Some keyloggers go beyond just logging keystrokes and recording text and snoop in several other ways as well. It's possible for advanced keyloggers to:

- Log clipboard text, recording information that you cut and paste from other documents
- Track activities like opening folders, documents, and applications
- Take and record randomly timed screenshots
- Request the text value of certain on-screen controls, which can be useful for grabbing passwords

Some uses of keyloggers could be considered ethical or appropriate to varying degrees. Keylogger recorders may also be used by:

- employers to observe employees' computer activities;
- parents to supervise their children's Internet usage;
- device owners to track the possible unauthorized activity on their devices; or
- law enforcement agencies to analyze incidents involving computer use.

There are basically two types of Keyloggers:

Hardware Keylogger: This is a thumb-size device. It records all the keystrokes you enter from the keyboard and then saves them in its memory. Later this data will be analyzed. The drawback

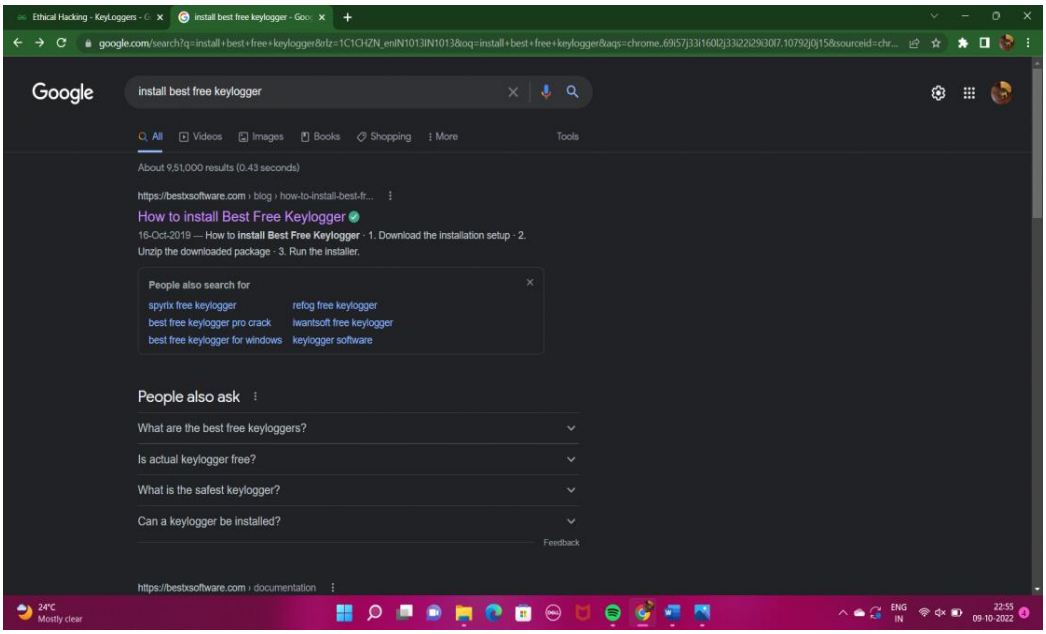
of this device is, It can’t record mouse clicks, can’t take screenshots, and even can’t email, more importantly, It requires physical access to the machine. Hardware Keylogger is advantageous because it’s not hooked into any software nor can it be detected by any software.

Software Keylogger: Software Keylogger can be installed in the victim’s system even if they use an updated Antivirus. There are lots of software available in the market which make a Keylogger undetectable by the latest antivirus, we are going to study them too in upcoming chapters. There are many keyloggers available in the market with various features. Some examples of Software Keyloggers are:

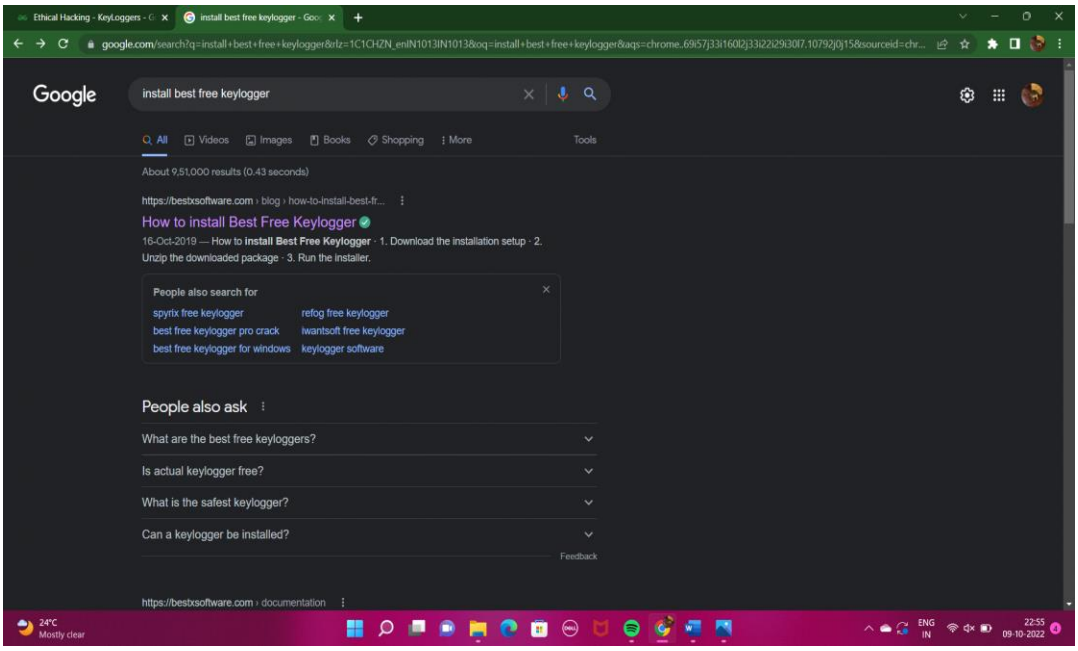
- Revealer HYPERLINK "https://www.logixoft.com/" HYPERLINK "https://www.logixoft.com/" HYPERLINK "https://www.logixoft.com/" HYPERLINK "https://www.logixoft.com/" HYPERLINK "https://www.logixoft.com/" HYPERLINK "https://www.logixoft.com/" HYPERLINK "https://www.logixoft.com/" Keylogger
- Ardamax HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.ardamax.com/keylogger" Keylogger
- WinSpy
- Invisible Keylogger
- Refog HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" HYPERLINK "https://www.geeksforgeeks.org/ethical-hacking-keyloggers/www.refog.com" Keylogger
- Best free key logger

PROCEDURE:

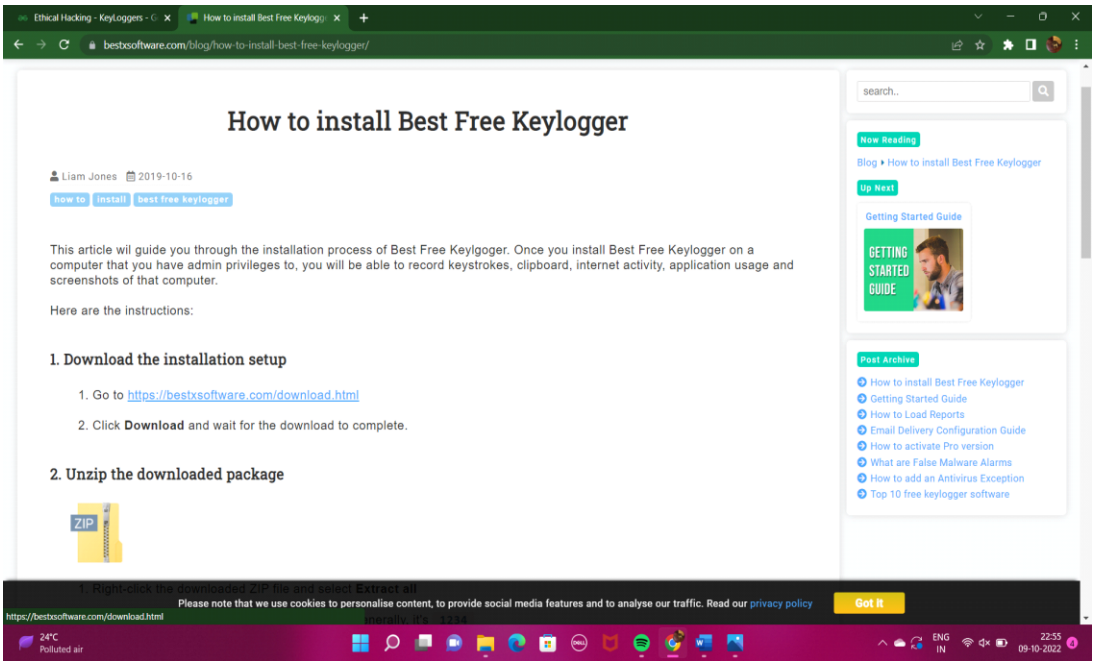
- Go to https://bestxsoftware.com/download.html



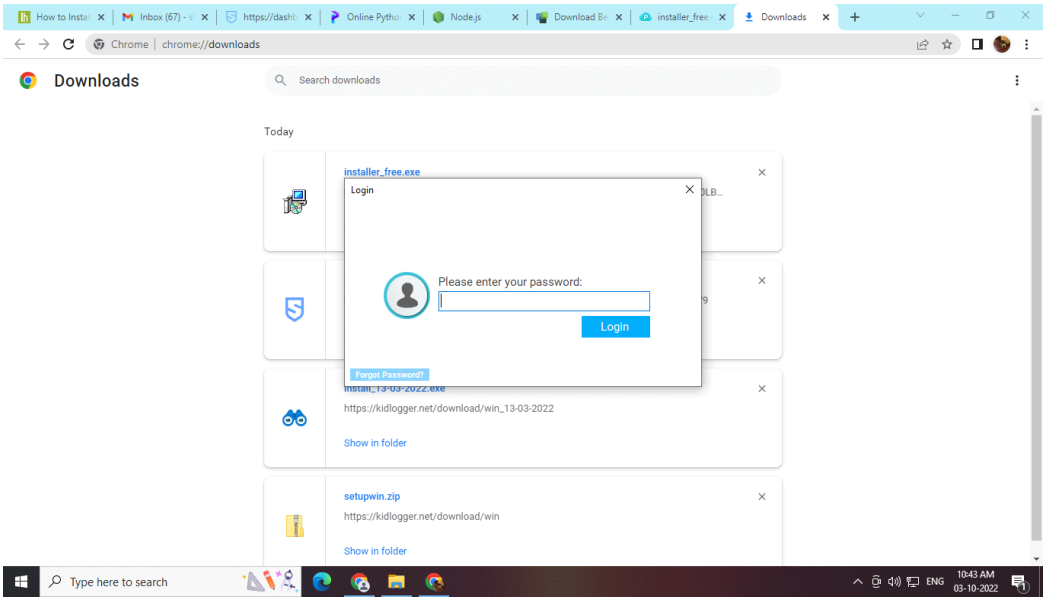
- Click **Download** and wait for the download to complete.



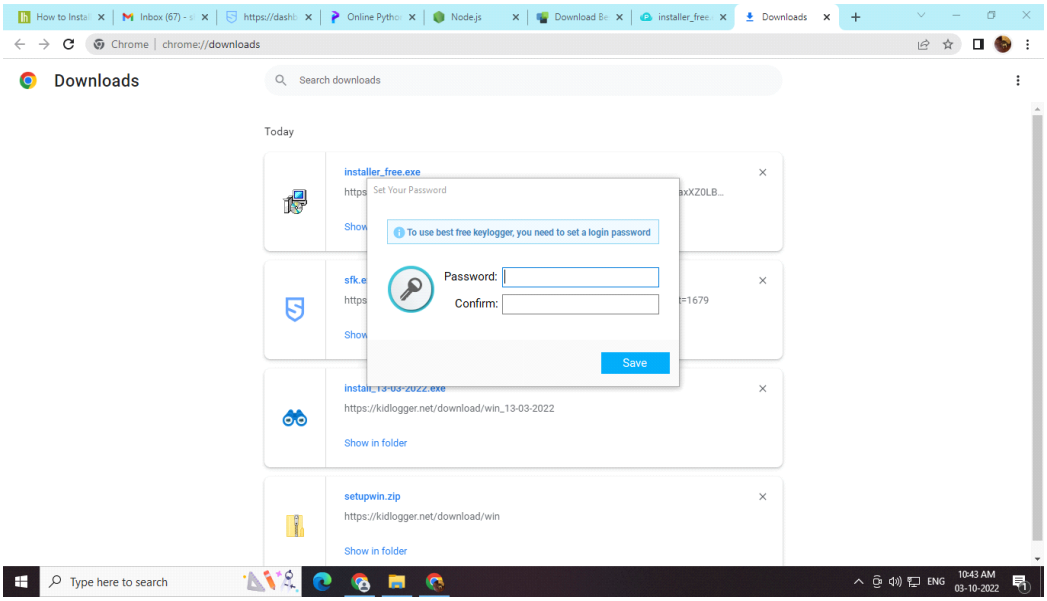
- Right-click the downloaded ZIP file and select **Extract all.**



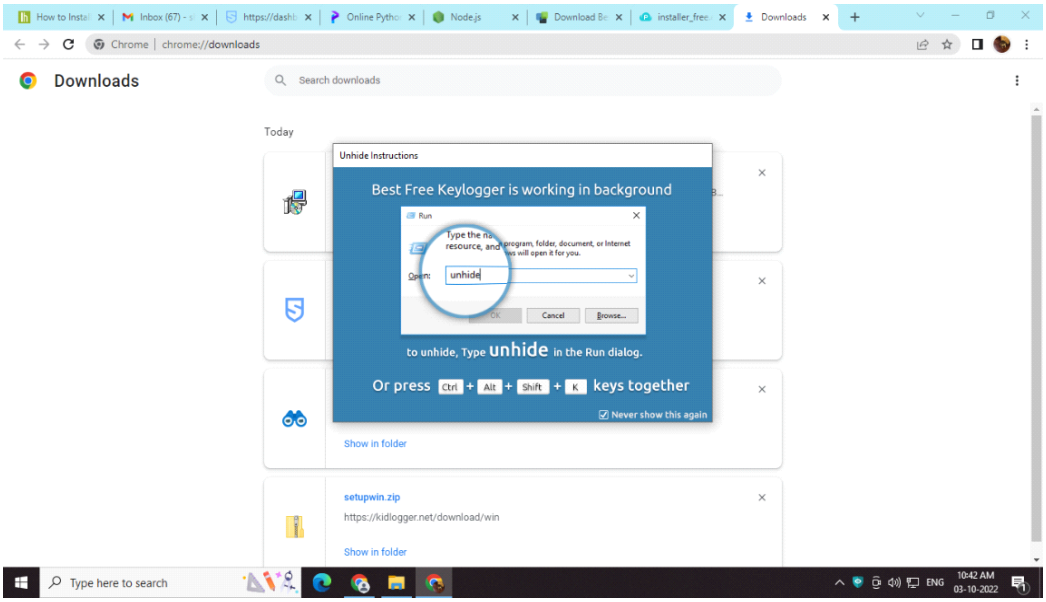
- Create a new password for the keylogger.



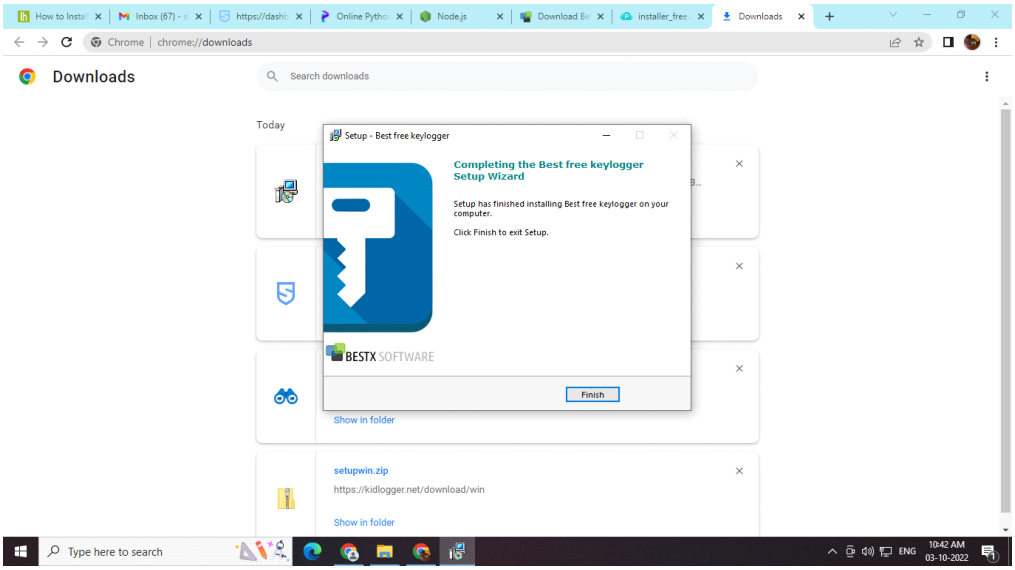
- Once the password is created, confirm and save it



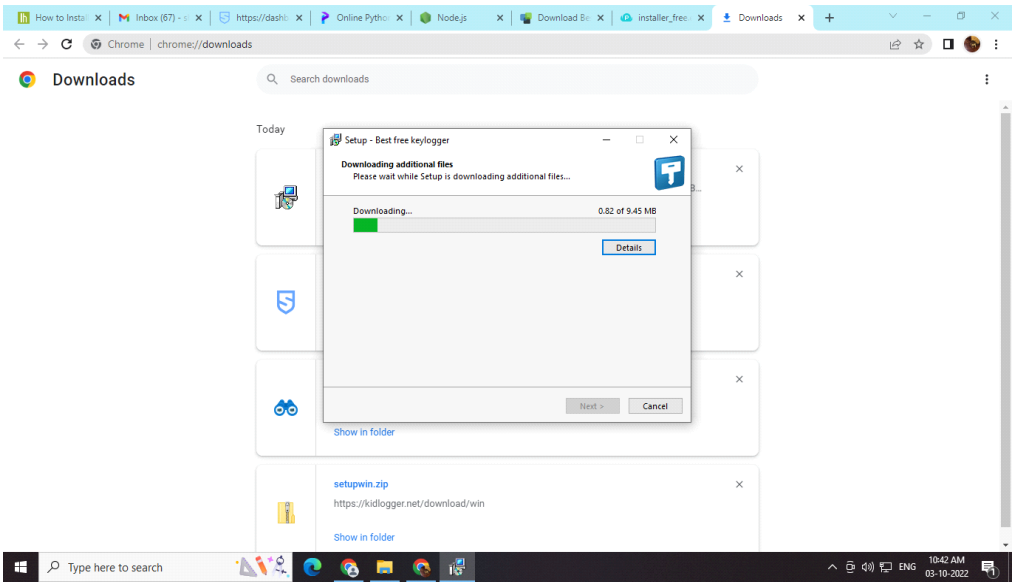
- The keylogger works in the background, so to unhide it press the ctrl + shift + alt + R keys together.



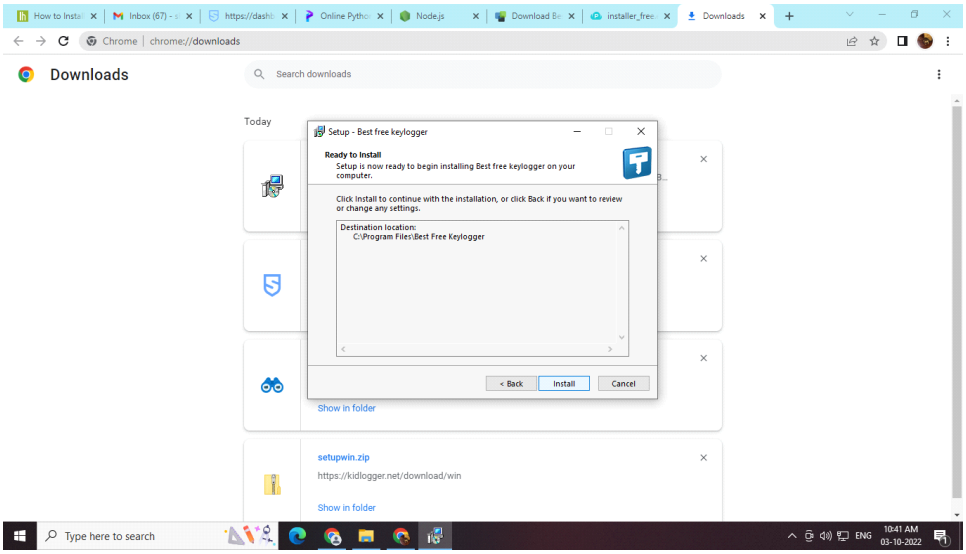
- A setup dialog box will open with a finish button, click finish.



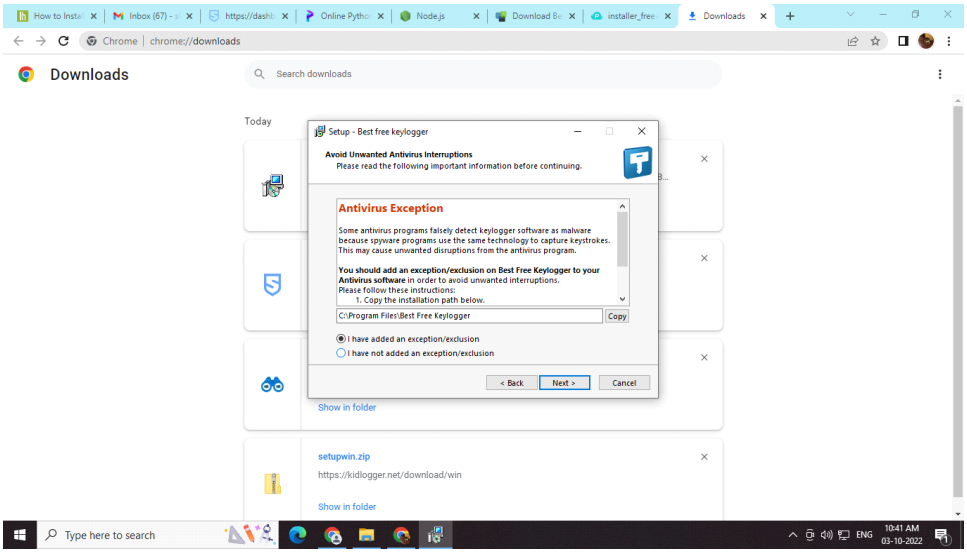
- Next, additional files will be installed for keylogger.



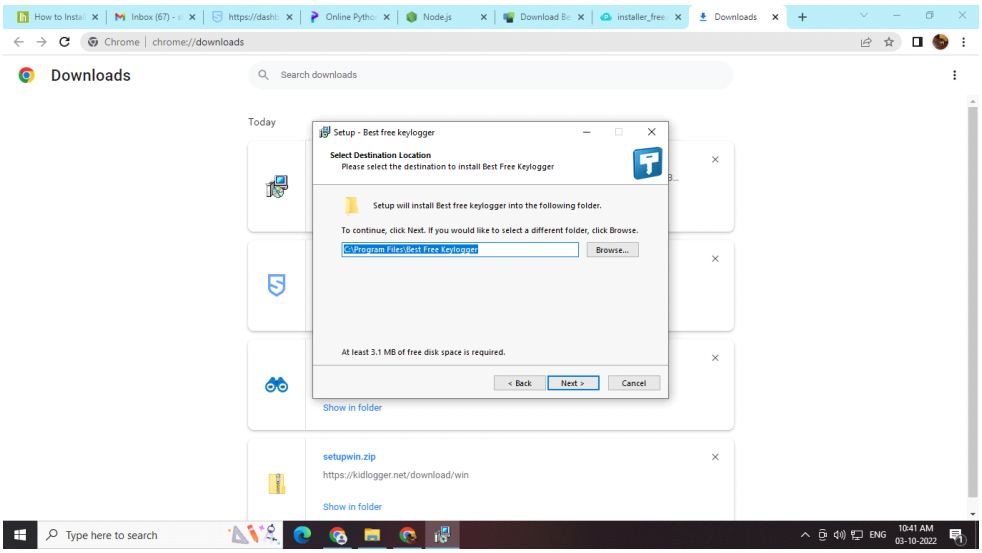
- Ready to install dialogue box will provide further details and address of installed files. Click install to continue.



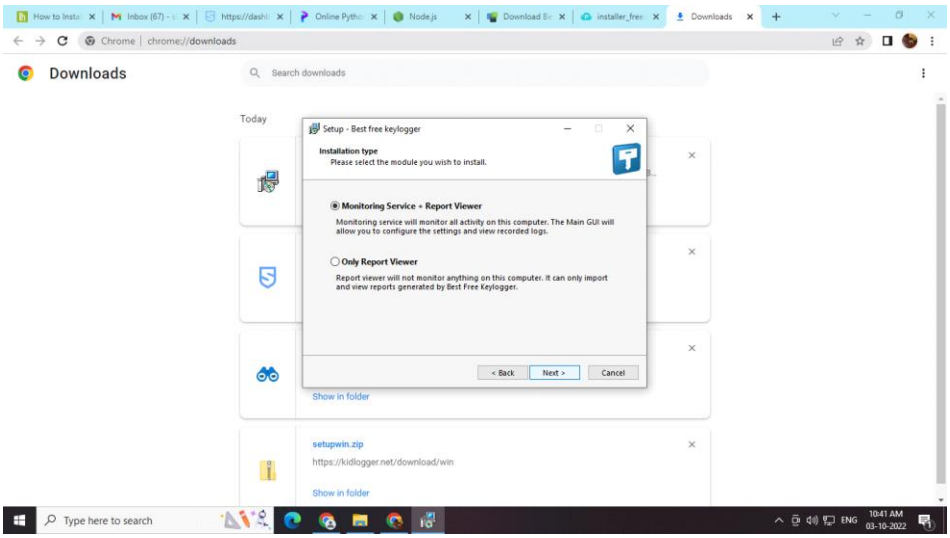
- An antivirus exception will appear, the user has to accept the antivirus exception to install the keylogger. Click next to continue.



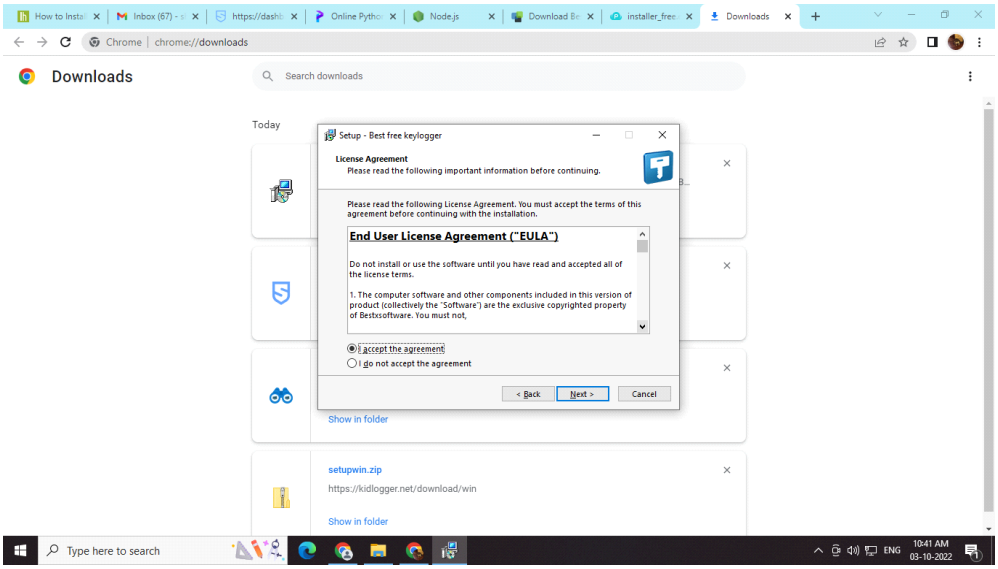
- The final location of the keylogger files will be displayed which is preferred to be unchanged. Click next to continue.



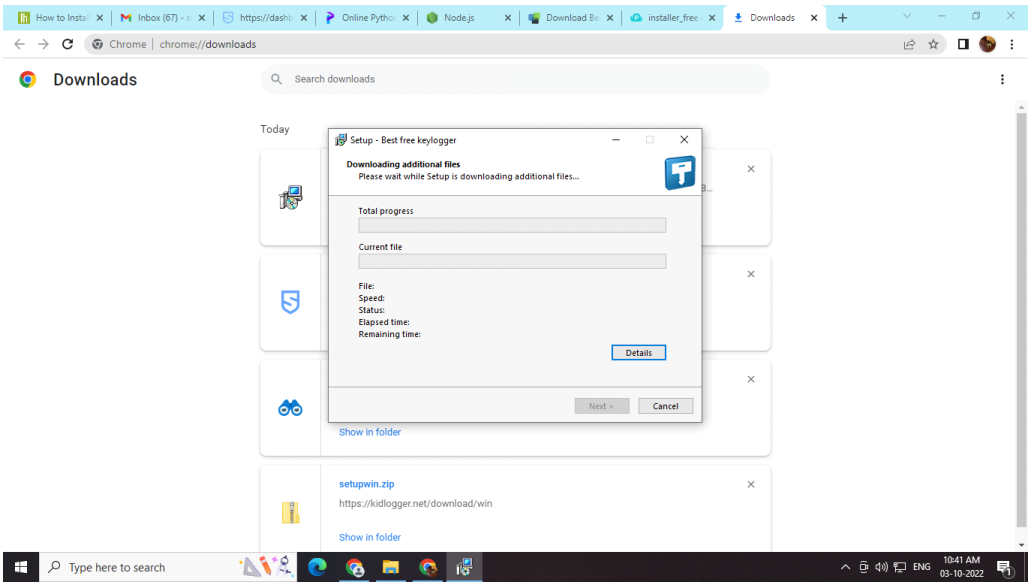
- The user will be asked about the installation type, in order to have a detailed analysis of keylogging it is preferred to go for the monitoring service + report Viewer option. Click next to continue.



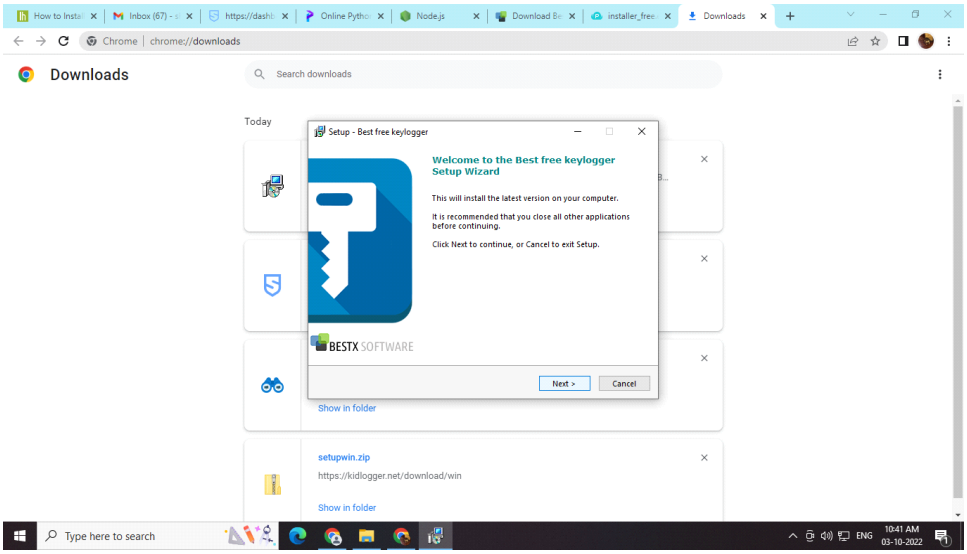
- License agreement window will ask to accept all the terms and conditions regarding the keylogger. Click next to continue.



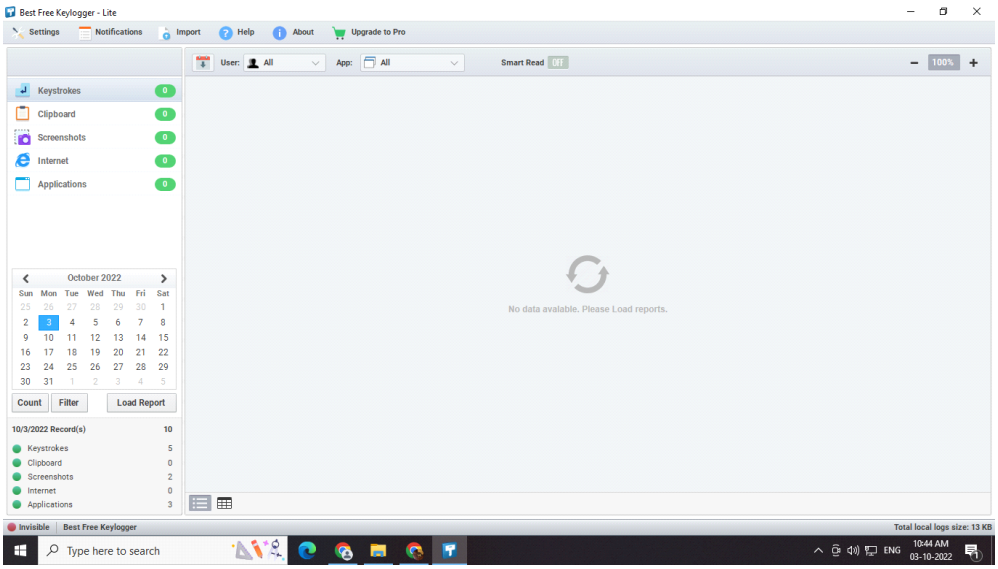
- Again a detailed installation of all the files necessary will occur. Click next again.



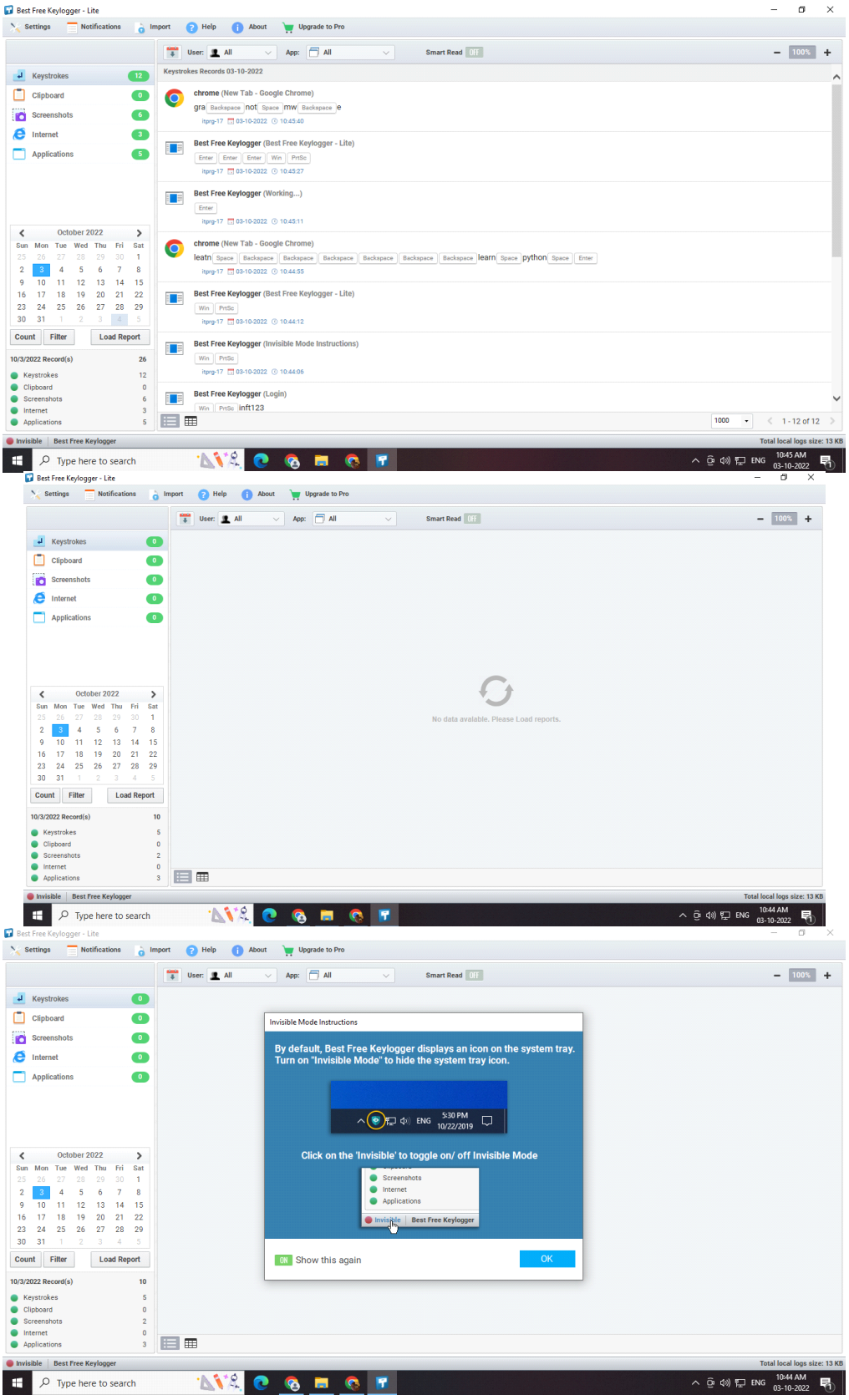
- Once all the installation procedures are done, user will find a desktop icon to access the records .



- The recods would be inintially empty.



- So as the user uses different applications and searches on the browser, all this information is accurately noted in the keylogger.


Best Free Keylogger - Lite
Settings
Notifications
Import
Help
About
Upgrade to Pro
Keystrokes
Clipboard
Screenshots
Internet
Applications
Count
Filter
Load Report
No data available. Please Load reports.
Invisible Mode Instructions
By default, Best Free Keylogger displays an icon on the system tray. Turn on "Invisible Mode" to hide the system tray icon.
Click on the 'Invisible' to toggle on/ off Invisible Mode
Show this again
OK
Type here to search

QUESTIONS:

1. What is a keylogger?_____

2. State any 3 funtions of a keylogger._____

3. Give examples of hardware keylogger._____

4. State few applications of a keylogger._____

5. Enlist software keyloggers?_____

CONCLUSION: The funtions and applications of a keylogger are studied and examined.

REFERENCES:

https://www.techtarget.com/searchsecurity/definition/keylogger

https://www.techtarget.com/searchsecurity/definition/keylogger

https://www.geeksforgeeks.org/introduction-to-keyloggers/

Experiment No. 9

Aim: To install and study Network Protocol Analyzer, analyse packet header using **Wireshark.**

API: Packet Analyzer, Ethereal, Wireshark, Ntop etc.

Theory :-

Version 3.6.2 (v3.6.2-0-g626020d9b3c3)

Copyright 1998-2022 Gerald Combs <gerald@wireshark.org> and contributors. License GPLv2+: GNU GPL version 2 or later <https://www.gnu.org/licenses/gpl-2.0.html> This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compiled (64-bit) using Microsoft Visual Studio 2019 (VC++ 14.29, build 30139), with Qt 5.15.2, with libpcap, with GLib 2.66.4, with zlib 1.2.11, with Lua 5.2.4, with GnuTLS 3.6.3 and PKCS #11 support, with Gcrypt 1.8.3, with MIT Kerberos, with MaxMind DB resolver, with nghttp2 1.44.0, with brotli, with LZ4, with Zstandard, with Snappy, with libxml2 2.9.10, with libsmi 0.4.8, with QtMultimedia, with automatic updates using WinSparkle 0.5.7, with AirPcap, with SpeexDSP (using bundled resampler), with Minizip.

Running on 64-bit Windows 10 (21H2), build 19044, with Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (with SSE4.2), with 7973 MB of physical memory, with GLib 2.66.4, with Qt 5.15.2, with Npcap version 1.55, based on libpcap version 1.10.2-PRE-GIT, with c-ares 1.17.0, with GnuTLS 3.6.3, with Gcrypt 1.8.3, with nghttp2 1.44.0, with brotli 1.0.9, with LZ4 1.9.3, with Zstandard 1.4.0, without AirPcap, with light display mode, without HiDPI, with LC_TYPE=English_India.utf8, binary plugins supported (21 loaded).

Wireshark is Open Source Software released under the GNU General Public License.

Check the man page and https://www.wireshark.org for more information.

Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

Wireshark has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis

- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis

Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others

Capture files compressed with gzip can be decompressed on the fly

Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)

Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2

Coloring rules can be applied to the packet list for quick, intuitive analysis

Output can be exported to XML, PostScript®, CSV, or plain text

Procedure: To download Wireshark:

1.Open a web browser.

2.Navigate to http://www.wireshark.org.

3.Select Download Wireshark.

4.Select the Wireshark Windows Installer matching your system type, either 32-bit or 64-bit as determined in Activity 1. Save the program in the Downloads folder.

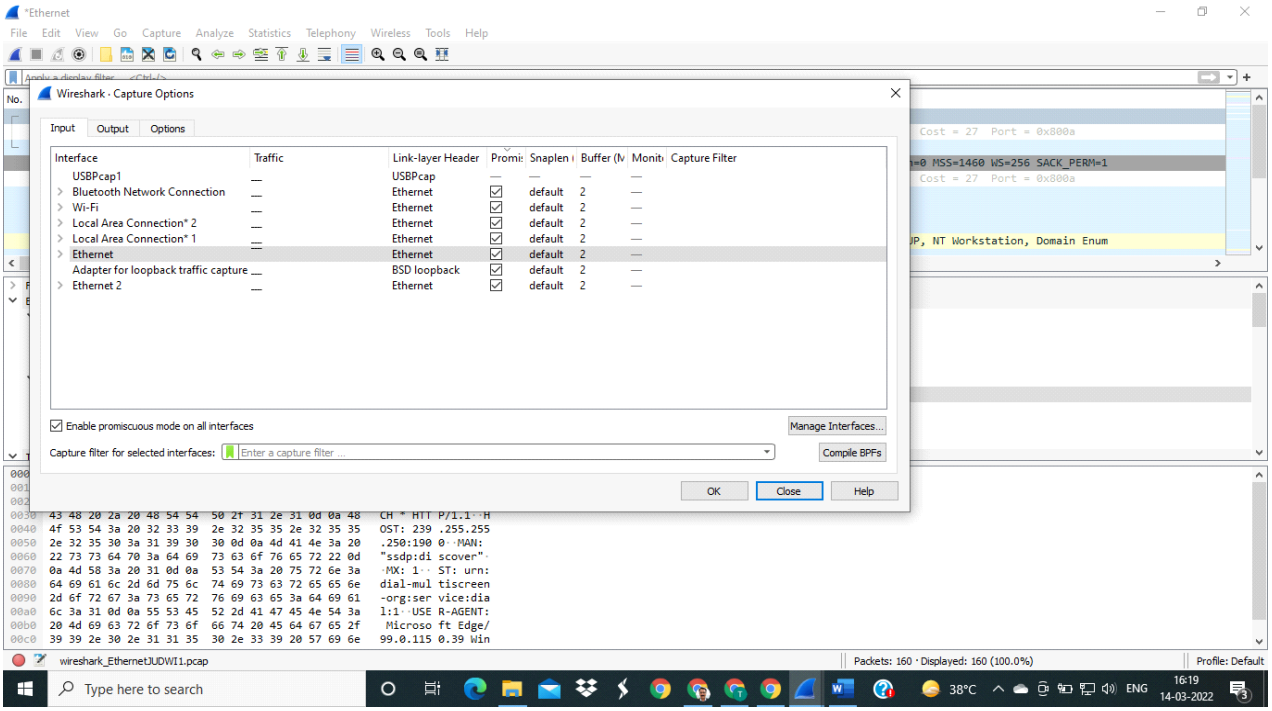
5.Close the web browser.

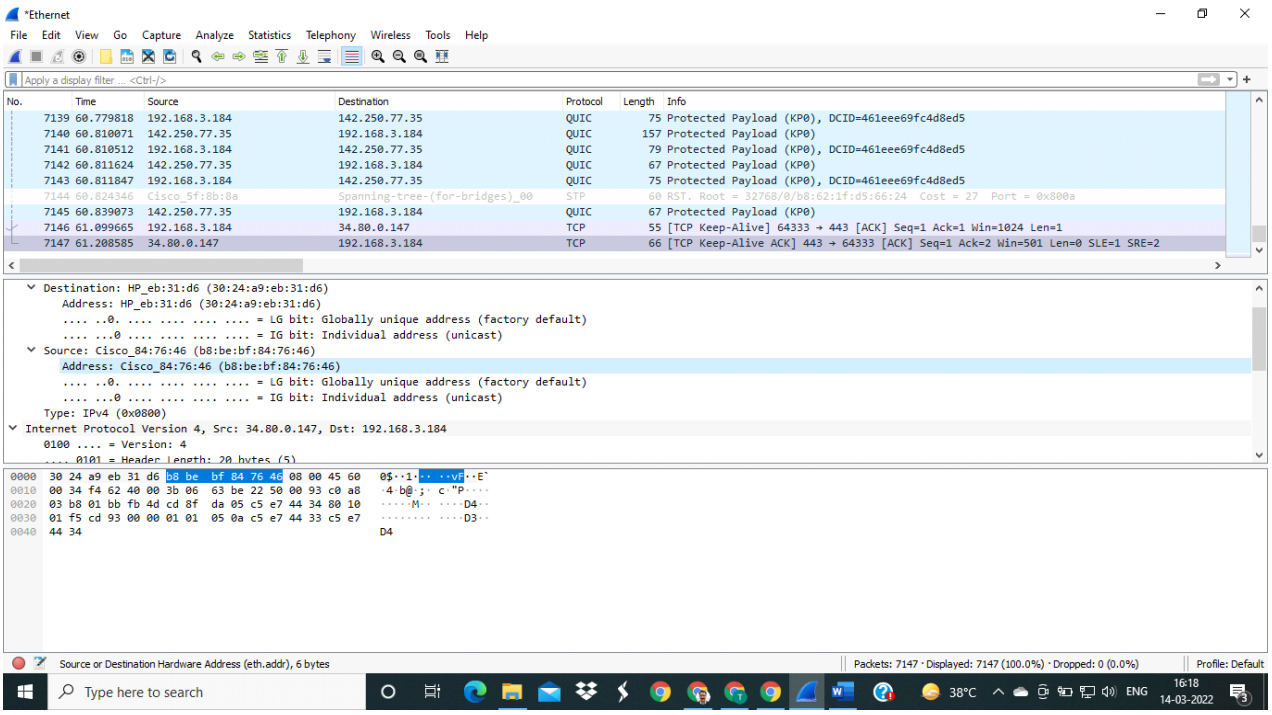
To install Wireshark:

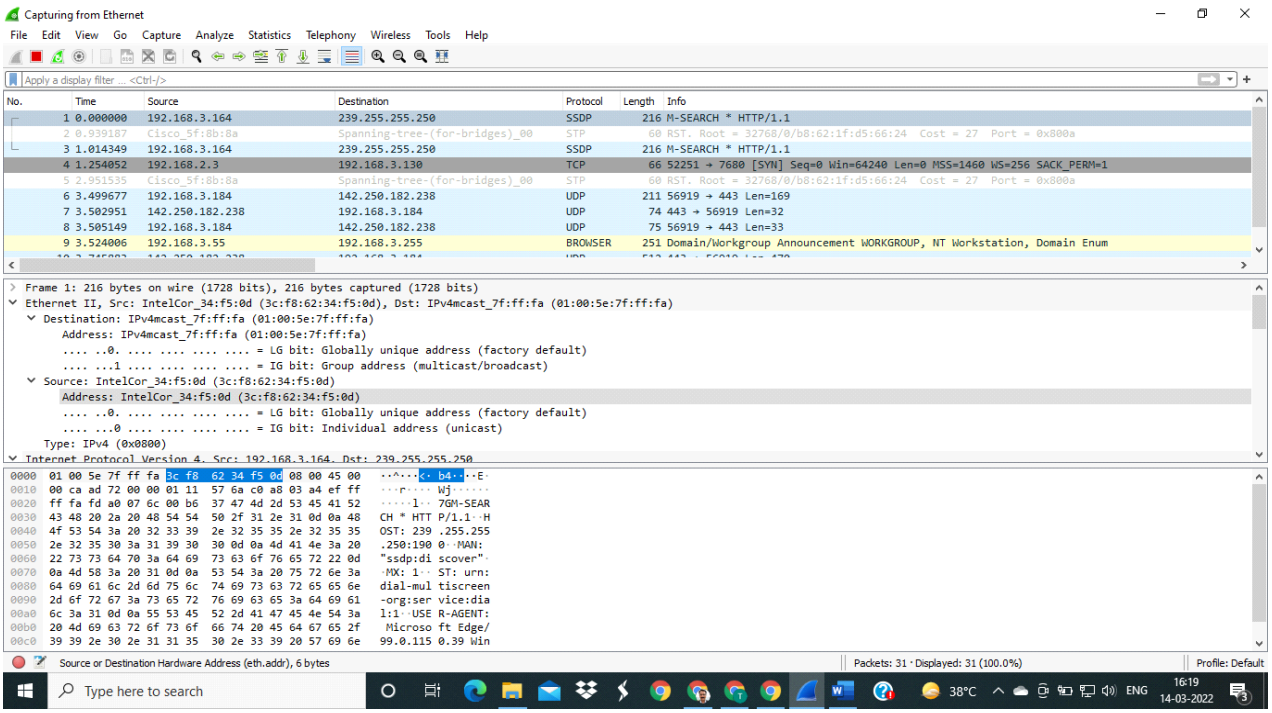
1.Open Windows Explorer.
2.Select the Downloads folder.
3.Locate the version of Wireshark you downloaded in Activity 2. Double-click on the file to open it.
4.If you see a User Account Control dialog box, select Yes to allow the program to make changes to this computer.

5.Select Next > to start the Setup Wizard.
6.Review the license agreement. If you agree, select I Agree to continue.
7.Select Next > to accept the default components.
8.Select the shortcuts you would like to have created. Leave the file extensions selected. Select Next > to continue.
9.Select Next > to accept the default install location.
10.Select Install to begin installation.
11.Select Next > to install WinPcap.
12.Select Next > to start the Setup Wizard.
13.Review the license agreement. If you agree, select I Agree to continue.
14.Select Install to begin installation.
15.Select Finish to complete the installation of WinPcap.
16.Select Next > to continue with the installation of Wireshark.
17.Select Finish to complete the installation of Wireshark.

Conclusion: Learnt how to install and configure Wireshark.




*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
7139 60.779818 192.168.3.184 142.250.77.35 QUIC 75 Protected Payload (KP0), DCID=461eee69fc4d8ed5
7140 60.810071 142.250.77.35 192.168.3.184 QUIC 157 Protected Payload (KP0)
7141 60.810512 192.168.3.184 142.250.77.35 QUIC 79 Protected Payload (KP0), DCID=461eee69fc4d8ed5
7142 60.811624 142.250.77.35 192.168.3.184 QUIC 67 Protected Payload (KP0)
7143 60.811847 192.168.3.184 142.250.77.35 QUIC 75 Protected Payload (KP0), DCID=461eee69fc4d8ed5
7144 60.824346 Cisco_5f:8b:8a Spanning-tree-(for-bridges)_00 STP 60 RST. Root = 32768/0/b8:62:1f:d5:66:24 Cost = 27 Port = 0x800a
7145 60.839073 142.250.77.35 192.168.3.184 QUIC 67 Protected Payload (KP0)
7146 61.099665 192.168.3.184 34.80.0.147 TCP 55 [TCP Keep-Alive] 64333 → 443 [ACK] Seq=1 Ack=1 Win=1024 Len=1
7147 61.208585 34.80.0.147 192.168.3.184 TCP 66 [TCP Keep-Alive ACK] 443 → 64333 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
Destination: HP_eb:31:d6 (30:24:a9:eb:31:d6)
Address: HP_eb:31:d6 (30:24:a9:eb:31:d6)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Source: Cisco_84:76:46 (b8:be:bf:84:76:46)
Address: Cisco_84:76:46 (b8:be:bf:84:76:46)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 34.80.0.147, Dst: 192.168.3.184
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
0000 30 24 a9 eb 31 d6 b8 be bf 84 76 46 08 00 45 60
0010 00 34 f4 62 40 00 3b 06 63 be 22 50 00 93 c0 a8
0020 03 b8 01 bb fb 4d cd 8f da 05 c5 e7 44 34 80 10
0030 01 f5 cd 93 00 00 01 01 05 0a c5 e7 44 33 c5 e7
0040 44 34
Source or Destination Hardware Address (eth.addr), 6 bytes
Packets: 7147 · Displayed: 7147 (100.0%) · Dropped: 0 (0.0%)
Profile: Default
Type here to search
38°C
ENG
16:18
14-03-2022


Capturing from Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.3.164 239.255.255.250 SSDP 216 M-SEARCH * HTTP/1.1
2 0.939187 Cisco_5f:8b:8a Spanning-tree-(for-bridges)_00 STP 60 RST. Root = 32768/0/b8:62:1f:d5:66:24 Cost = 27 Port = 0x800a
3 1.014349 192.168.3.164 239.255.255.250 SSDP 216 M-SEARCH * HTTP/1.1
4 1.254052 192.168.2.3 192.168.3.130 TCP 66 52251 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5 2.951535 Cisco_5f:8b:8a Spanning-tree-(for-bridges)_00 STP 60 RST. Root = 32768/0/b8:62:1f:d5:66:24 Cost = 27 Port = 0x800a
6 3.499677 192.168.3.184 142.250.182.238 UDP 211 56919 → 443 Len=169
7 3.502951 142.250.182.238 192.168.3.184 UDP 74 443 → 56919 Len=32
8 3.505149 192.168.3.184 142.250.182.238 UDP 75 56919 → 443 Len=33
9 3.524006 192.168.3.55 192.168.3.255 BROWSER 251 Domain/Workgroup Announcement WORKGROUP, NT Workstation, Domain Enum
Frame 1: 216 bytes on wire (1728 bits), 216 bytes captured (1728 bits)
Ethernet II, Src: IntelCor_34:f5:0d (3c:f8:62:34:f5:0d), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Destination: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
Address: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
.... ..0. .... = LG bit: Globally unique address (factory default)
....1 .... = IG bit: Group address (multicast/broadcast)
Source: IntelCor_34:f5:0d (3c:f8:62:34:f5:0d)
Address: IntelCor_34:f5:0d (3c:f8:62:34:f5:0d)
.... ..0. .... = LG bit: Globally unique address (factory default)
....0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.3.164, Dst: 239.255.255.250
0000 01 00 5e 7f ff fa 3c f8 62 34 f5 0d 08 00 45 00
0010 00 ca ad 72 00 00 01 11 57 6a c0 a8 03 a4 ef ff
0020 ff fa fd a0 07 6c 00 b6 37 47 4d 2d 53 45 41 52
0030 43 48 20 2a 20 48 54 54 50 2f 31 2e 31 0d 0a 48
0040 4f 53 54 3a 20 32 33 39 2e 32 35 35 2e 32 35 35
0050 2e 32 35 30 3a 31 39 30 30 0d 0a 4d 41 4e 3a 20
0060 22 73 73 64 70 3a 64 69 73 63 6f 76 65 72 22 0d
0070 0a 4d 58 3a 20 31 0d 0a 53 54 3a 20 75 72 6e 3a
0080 64 69 61 6c 2d 6d 75 6c 74 69 73 63 72 65 65 6e
0090 2d 6f 72 67 3a 73 65 72 76 69 63 65 3a 64 69 61
00a0 6c 3a 31 0d 0a 55 53 45 52 2d 41 47 45 4e 54 3a
00b0 20 4d 69 63 72 6f 73 6f 66 74 20 45 64 67 65 2f
00c0 39 39 2e 30 2e 31 31 35 30 2e 33 39 20 57 69 6e
Source or Destination Hardware Address (eth.addr), 6 bytes
Packets: 31 · Displayed: 31 (100.0%)
Profile: Default
Type here to search
38°C
ENG
16:19
14-03-2022

----------X-0-X----------

APIs: Packet Analyzer

Procedure: ***To capture packer: Set up the Packet Capture:***

1. Click View > Wireless Toolbar. The Wireless Toolbar will appear just below the Main toolbar.

2. Use the Wireless Toolbar to configure the desired channel and channel width.

3. Under Capture, click on AirPcap USB wireless capture adapter to select the capture interface.

Note: If the AirPcap isn't listed, press F5 to refresh the list of available packet capture interfaces. The AirPcap has been discontinued by RiverBed and is 802.11n only.

4. Click the Start Capture button to begin the capture.

5. When you are finished capturing, click the Stop button.

•***Saving the Capture:***

1. To save the capture, click File > Save.

2. Name the file, and click Save.

Note: .Pcap and .Pcap-ng are good filetypes to use for the capture if you plan to use Eye P.A. to open the capture.

3. Eye P.A. can now open the capture file.

Theory:

Packet sniffer \ Packet analyzer:

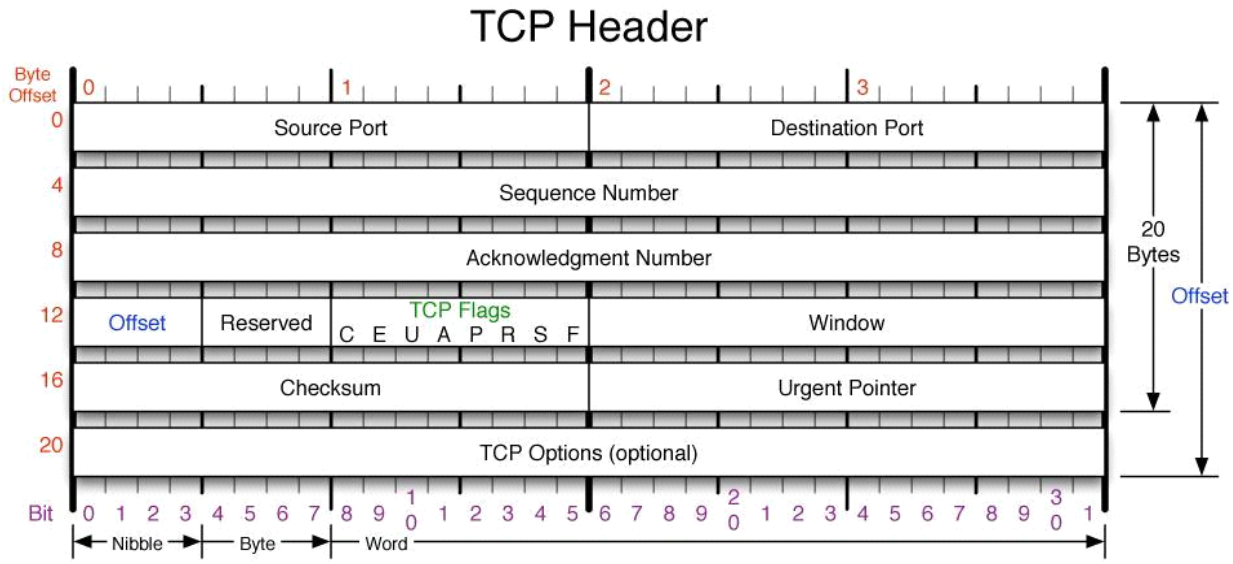
A packet analyzer (also known as a network analyzer, protocol analyzer or packet sniffer or for particular types of networks, an Ethernet sniffer or wireless sniffer) is a computer program or a piece of computer hardware that can intercept and log traffic passing over a digital network or part of a network. As data streams own across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate RFC or other specifications.

Different types of packet:

1. TCP:

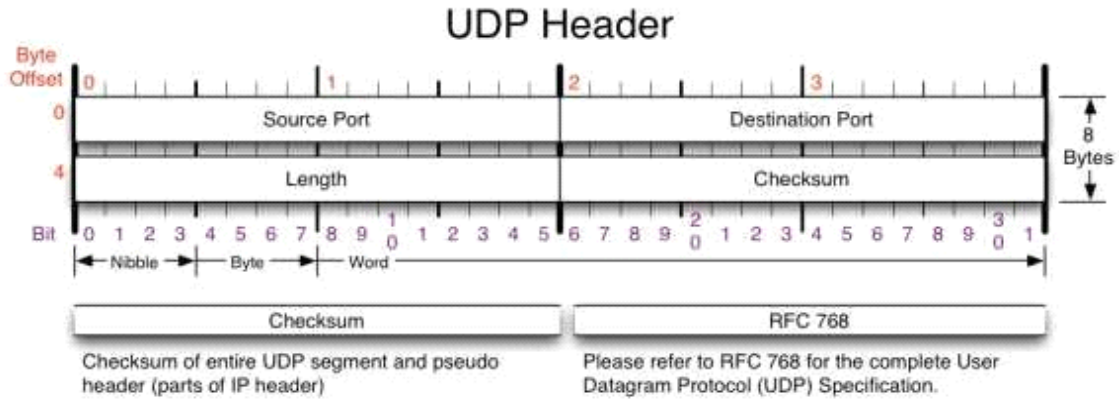
The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite (IP), and is so common that the entire suite is often called TCP/IP. TCP provides reliable, ordered and error-checked delivery (or notification of failure to deliver) of a stream of octets between programs running on computers connected to a local area network, intranet or the public Internet. It resides at the transport layer. Web browsers use TCP when they connect to servers on the World Wide Web, and it is used to deliver email and transfer files from one location to another. The protocol corresponds to the transport layer of TCP/IP suite. TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is, when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the

IP works by exchanging pieces of information called packets. A packet is a sequence of octets (bytes) and consists of a header followed by a body. The header describes the packet's source, destination and control information. The body contains the data IP is transmitting. Due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets can be lost, duplicated, or delivered out of order. TCP detects these problems, requests retransmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of the other problems. If the data still remains undelivered, its source is notified of this failure. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the receiving application. Thus, TCP abstracts the application's communication from the underlying networking details. TCP is a reliable stream delivery service that guarantees that all bytes received will be identical with bytes sent and in the correct order. Since packet transfer over many networks is not reliable, a technique known as positive acknowledgment with retransmission is used to guarantee reliability of packet transfers. This fundamental technique requires the receiver to respond with an acknowledgment message as it receives the data. The sender keeps a record of each packet it sends. The sender also maintains a timer from when the packet was sent, and retransmits a packet if the timer expires before the message has been acknowledged. The timer is needed in case a packet gets lost or corrupted.



2. UDP:

The User Datagram Protocol (UDP) is one of the core members of the Internet protocol Suite. UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. There is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths. UDP is suitable for purposes where error checking and correction is either not necessary or is performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.



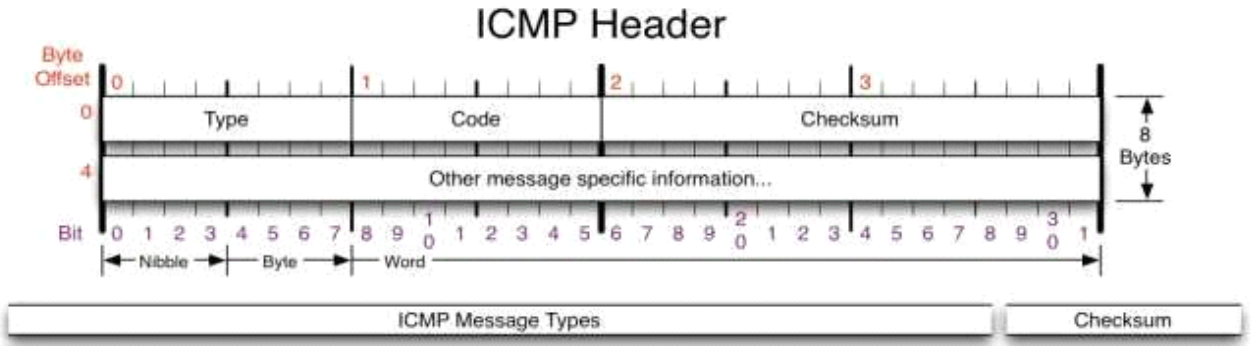
3.ICMP:

The Internet Control Message Protocol (ICMP) is one of the main protocols of the Internet Protocol Suite. It is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached.

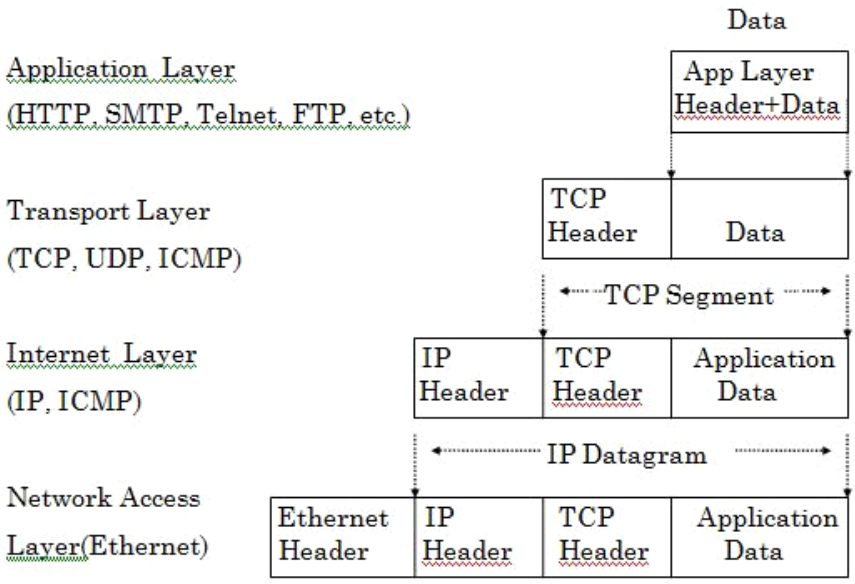
ICMP can also be used to relay query messages. It is assigned protocol number 1.ICMP differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems, nor is it regularly employed by end-user network applications (with the exception of some diagnostic tools like ping and trace route). ICMP for Internet Protocol version 4 (IPv4) is also known as ICMPv4. IPv6 has a similar protocol, ICMPv6. The Internet Control Message Protocol is part of the Internet Protocol Suite, as defined in RFC 792. ICMP messages

4.IGMP:

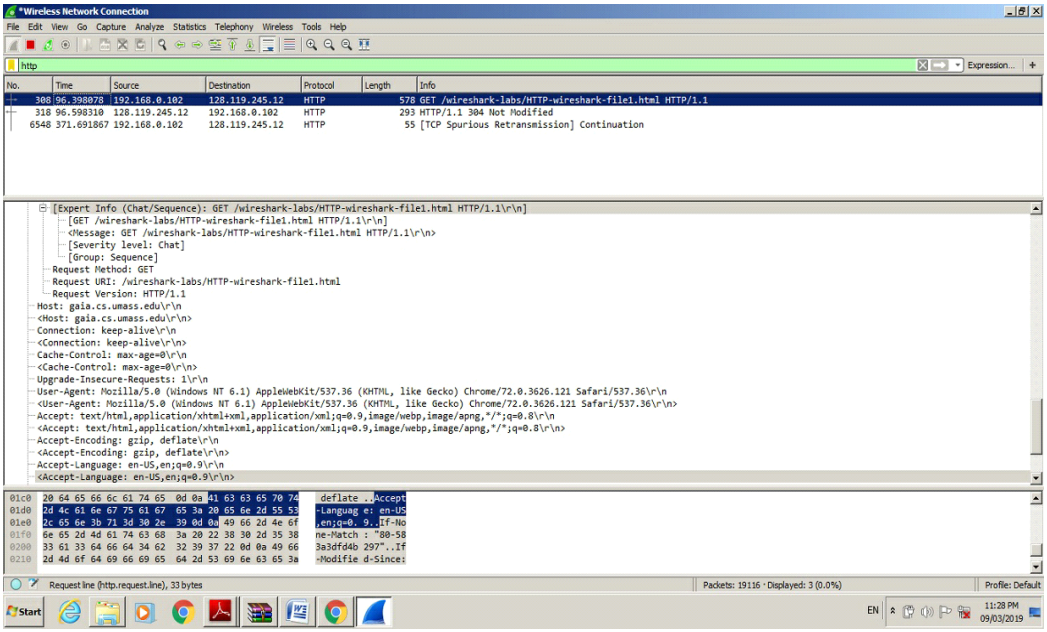
The Internet Group Management Protocol (IGMP) is a communications protocol used by hosts and adjacent routers on IP networks to establish multicast group memberships. IGMP is an integral part of IP multicast.IGMP can be used for one-to-many networking applications such as online streaming video and gaming, and allows more efficient use of resources when supporting these types of applications. IGMP messages are carried in bare IP packets with IP protocol. There is no transport layer used with IGMP messaging, similar to the Internet Control Message Protocol. Membership Queries are sent by multicast routers to determine which multicast addresses are of interest to systems attached to its network. Routers periodically send General Queries to refresh the group membership state for all systems on its network. Group-Specific Queries are used for determining the reception



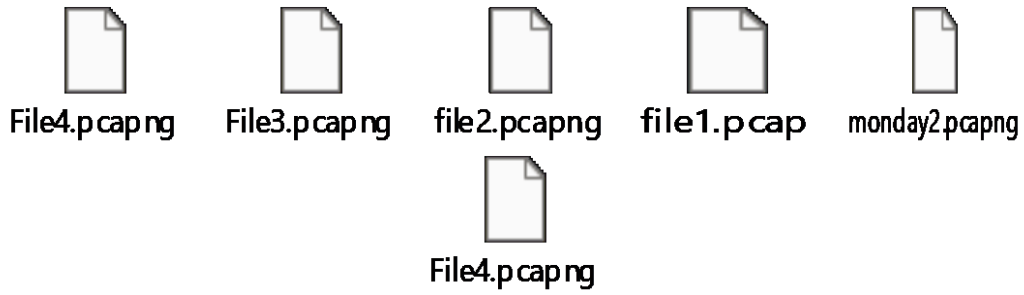
state for a particular multicast address.


Data
Application Layer
(HTTP, SMTP, Telnet, FTP, etc.)
App Layer Header+Data
Transport Layer
(TCP, UDP, ICMP)
TCP Header
Data
TCP Segment
Internet Layer
(IP, ICMP)
IP Header
TCP Header
Application Data
IP Datagram
Network Access Layer(Ethernet)
Ethernet Header
IP Header
TCP Header
Application Data

Output:



Conclusion: We have successfully analysed the captured packets from Wireshark.



Experiment No. 10

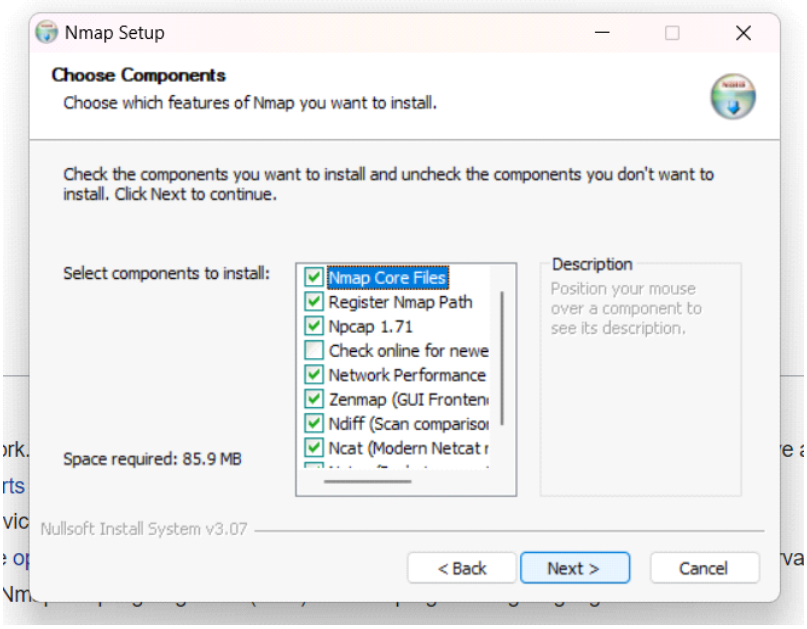
Aim : To install and implement NMAP.

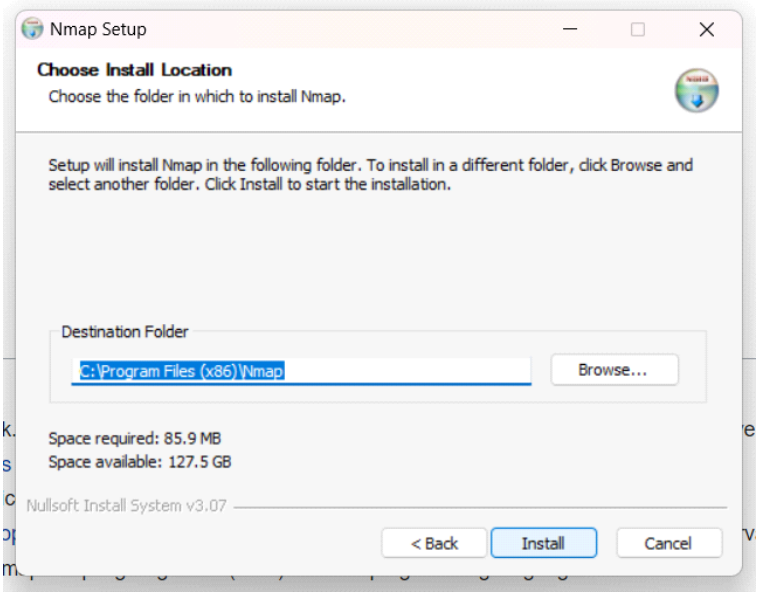
NMAP : Nmap (**Network Mapper**) is a network scanner created by Gordon Lyon. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses.

Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. Nmap can adapt to network conditions including latency and congestion during a scan.

INSTALLATION OF NMAP :

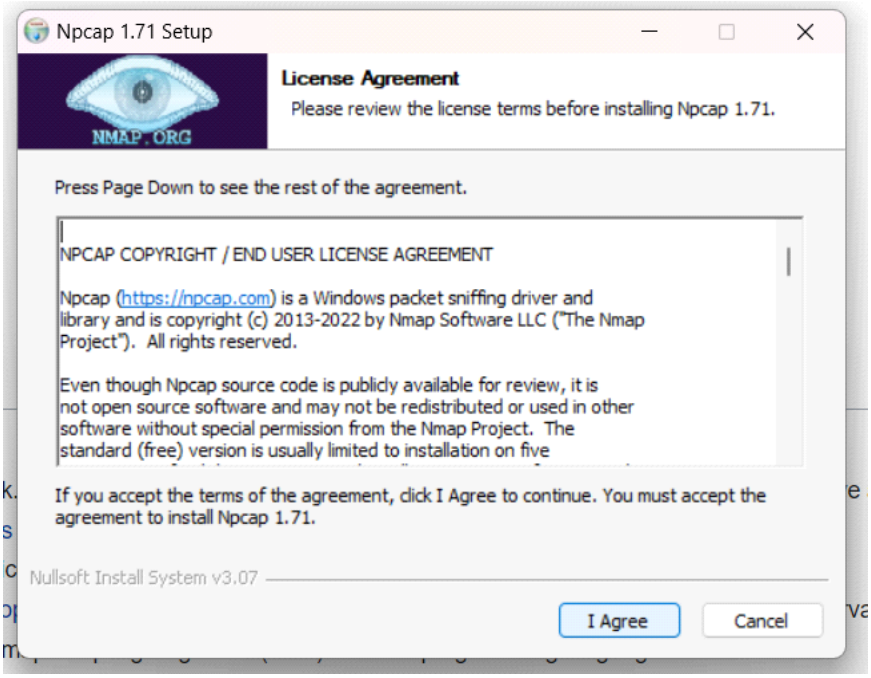
- Run the installer and click next :



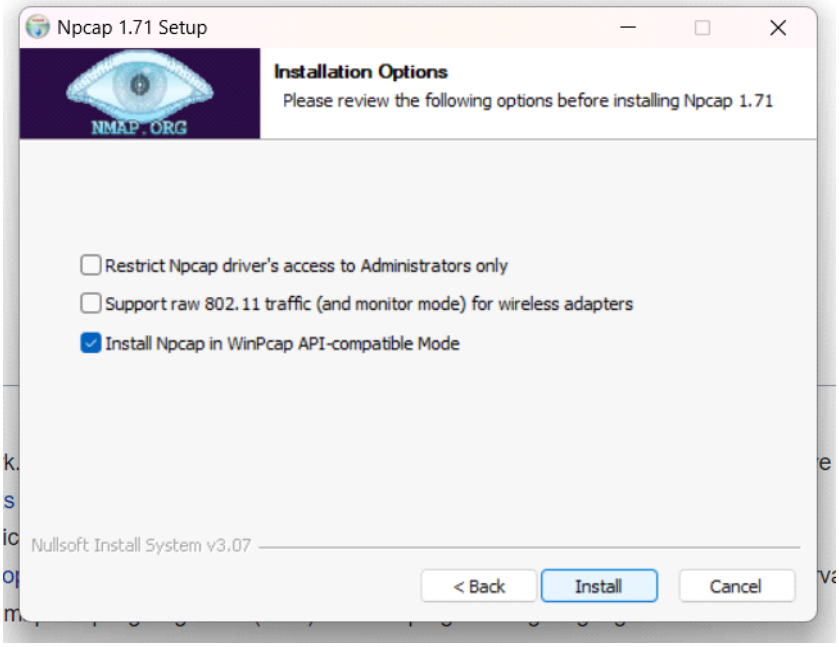
- 

Click on install :

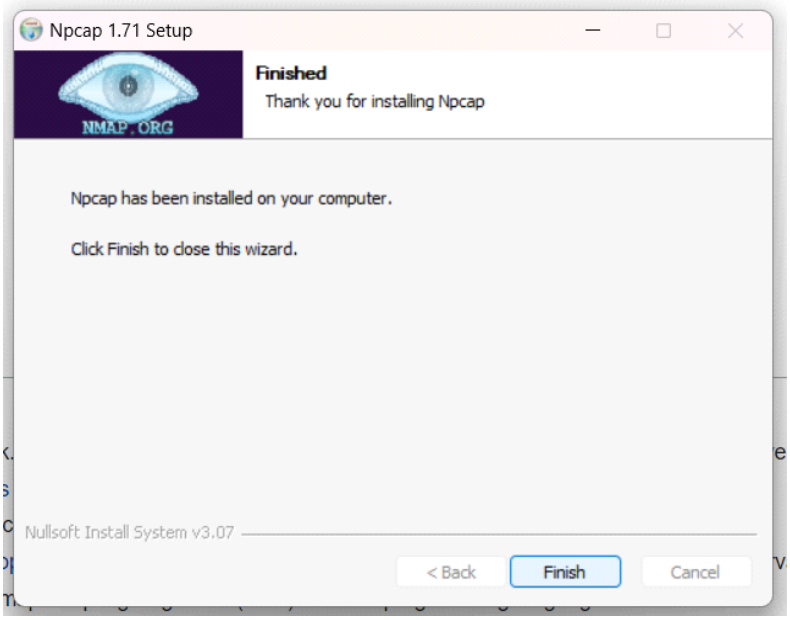
- **Click on I Agree** :



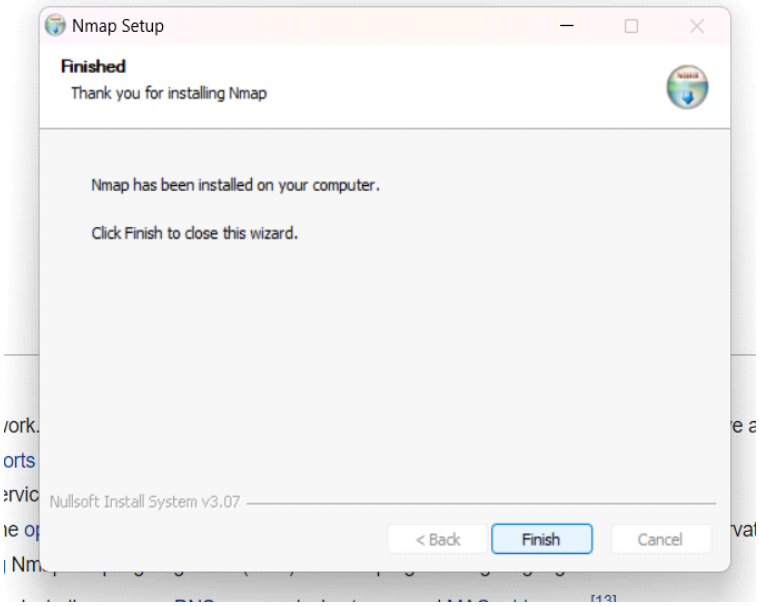
- **Click on Install :**



- Click on Finish :

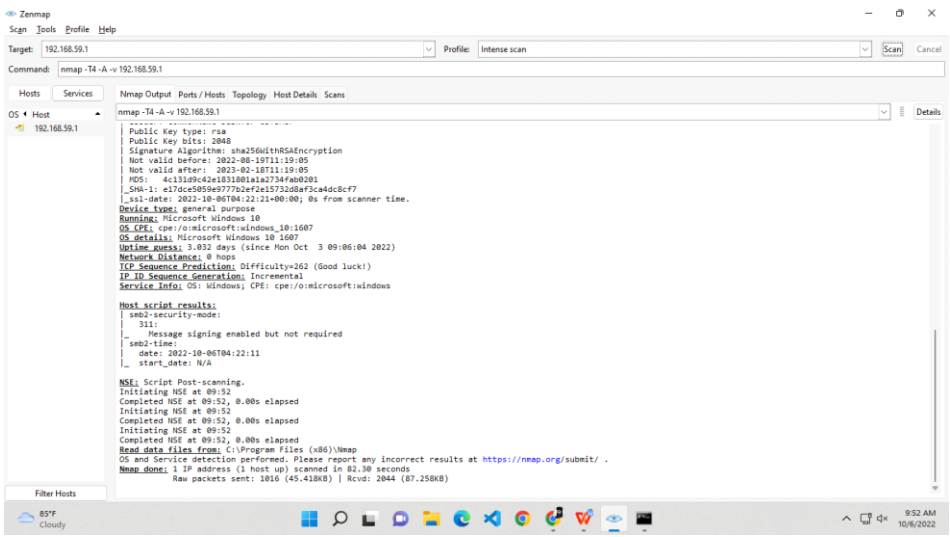


- **Click on Finish :**

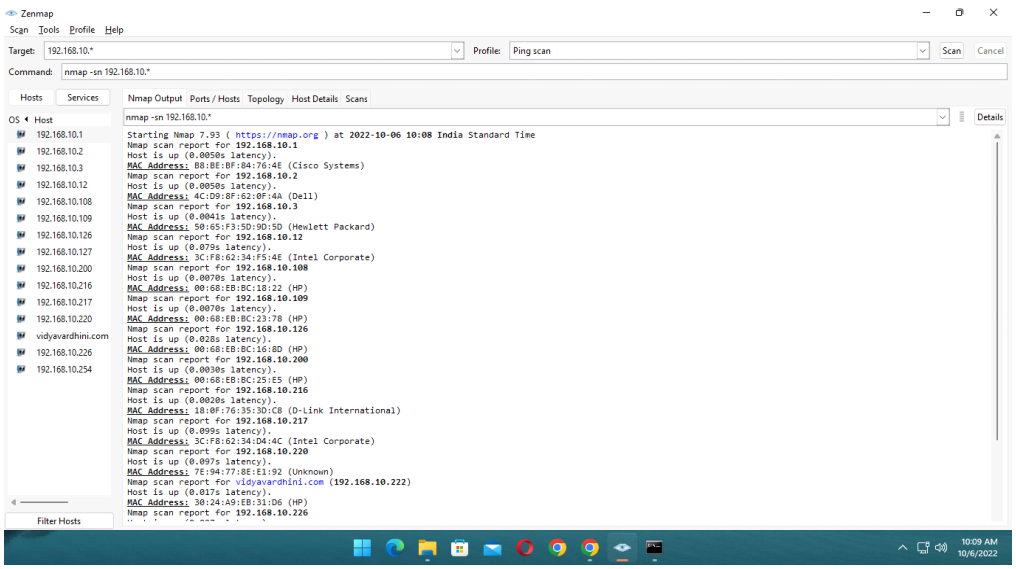


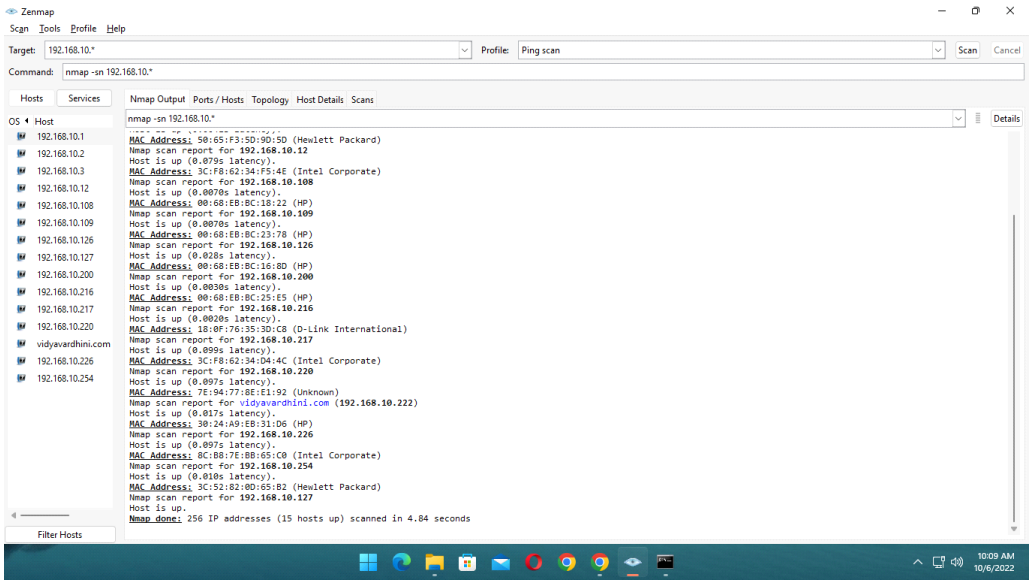
- **IMPLEMENTATION OF NMAP** :

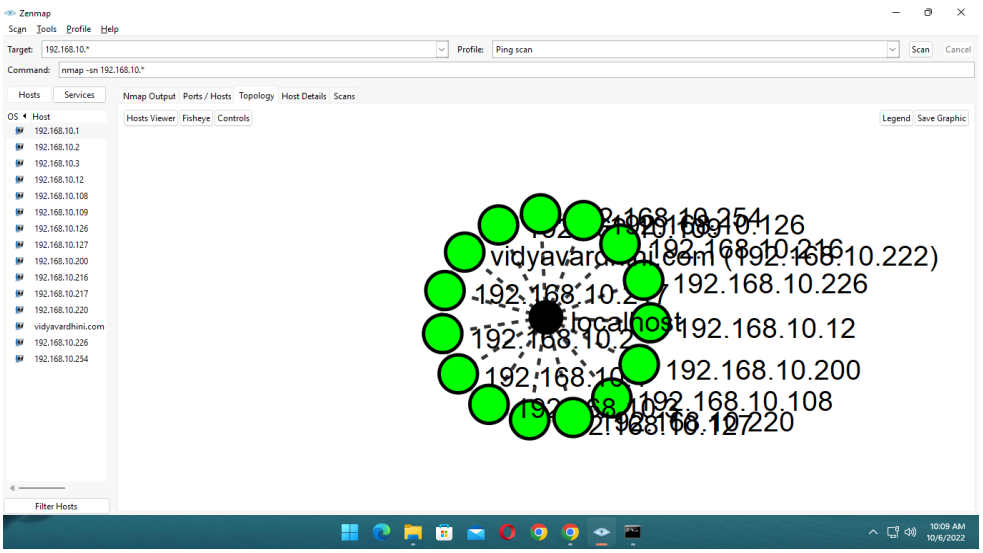
1. nmap-T4-A-v (ip address):



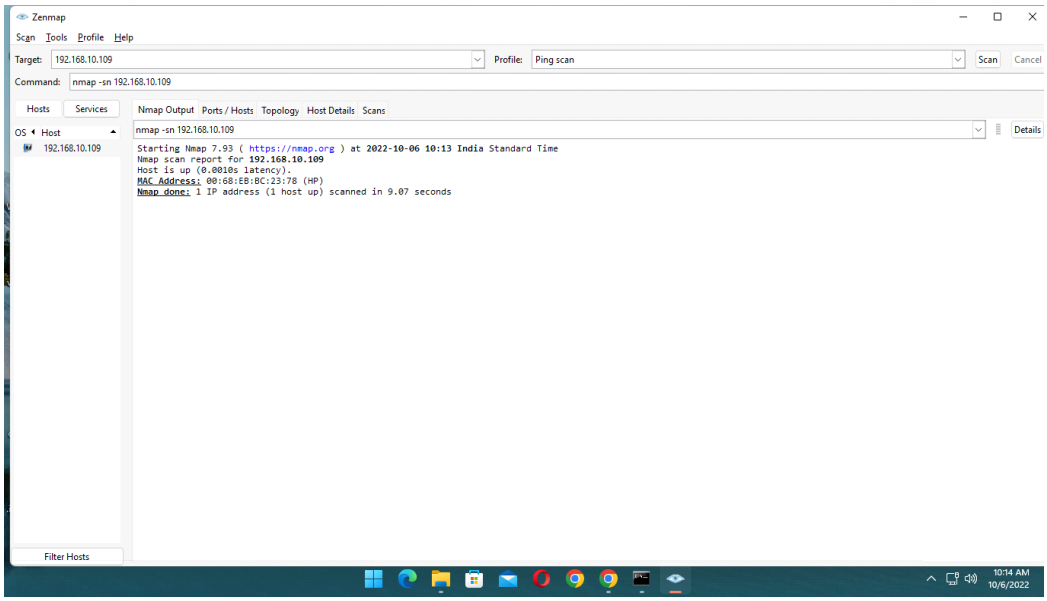
1. nmap -sn (ip address):

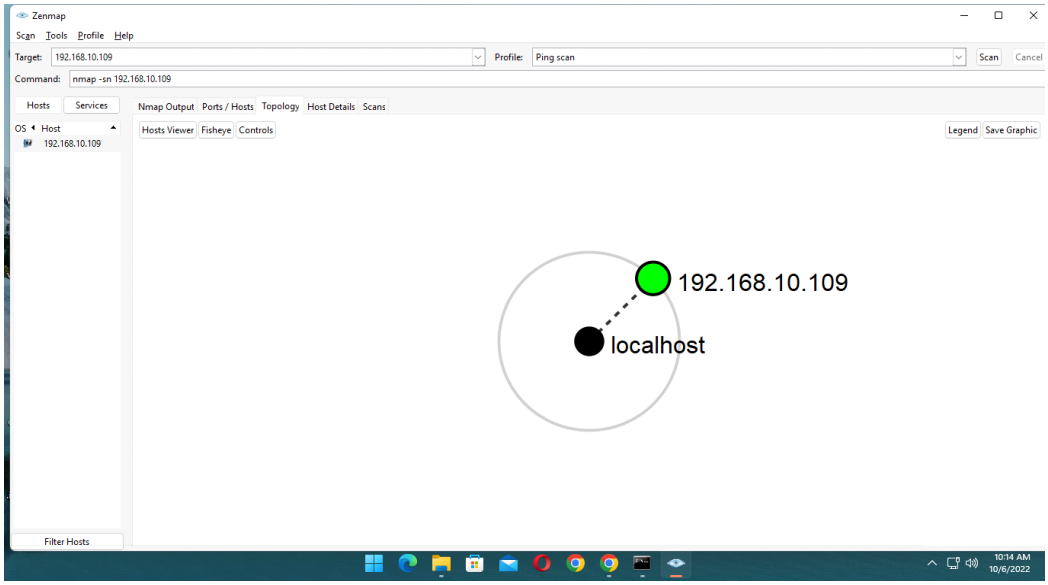

Zenmap
Scan Tools Profile Help
Target: 192.168.10.*
Profile: Ping scan
Scan Cancel
Command: nmap -sn 192.168.10.*
Hosts Services
Nmap Output Ports / Hosts Topology Host Details Scans
OS Host
192.168.10.1
192.168.10.2
192.168.10.3
192.168.10.12
192.168.10.108
192.168.10.109
192.168.10.126
192.168.10.127
192.168.10.200
192.168.10.216
192.168.10.217
192.168.10.220
vidyavardhini.com
192.168.10.226
192.168.10.254
Filter Hosts
nmap -sn 192.168.10.*
Details
Starting Nmap 7.93 (https://nmap.org) at 2022-10-06 10:08 India Standard Time
Nmap scan report for 192.168.10.1
Host is up (0.0050s latency).
MAC Address: B8:BE:BF:84:76:4E (Cisco Systems)
Nmap scan report for 192.168.10.2
Host is up (0.0050s latency).
MAC Address: 4C:D9:8F:62:0F:4A (Dell)
Nmap scan report for 192.168.10.3
Host is up (0.0041s latency).
MAC Address: 50:65:F3:5D:9D:5D (Hewlett Packard)
Nmap scan report for 192.168.10.12
Host is up (0.079s latency).
MAC Address: 3C:F8:62:34:F5:4E (Intel Corporate)
Nmap scan report for 192.168.10.108
Host is up (0.0070s latency).
MAC Address: 00:68:EB:BC:18:22 (HP)
Nmap scan report for 192.168.10.109
Host is up (0.0070s latency).
MAC Address: 00:68:EB:BC:23:78 (HP)
Nmap scan report for 192.168.10.126
Host is up (0.028s latency).
MAC Address: 00:68:EB:BC:16:8D (HP)
Nmap scan report for 192.168.10.200
Host is up (0.0030s latency).
MAC Address: 00:68:EB:BC:25:E5 (HP)
Nmap scan report for 192.168.10.216
Host is up (0.0020s latency).
MAC Address: 18:0F:76:35:3D:C8 (D-Link International)
Nmap scan report for 192.168.10.217
Host is up (0.099s latency).
MAC Address: 3C:F8:62:34:D4:4C (Intel Corporate)
Nmap scan report for 192.168.10.220
Host is up (0.097s latency).
MAC Address: 7E:94:77:8E:E1:92 (Unknown)
Nmap scan report for vidyavardhini.com (192.168.10.222)
Host is up (0.017s latency).
MAC Address: 30:24:A9:EB:31:D6 (HP)
Nmap scan report for 192.168.10.226
10:09 AM
10/6/2022


Zenmap
Scan Tools Profile Help
Target: 192.168.10.*
Profile: Ping scan
Scan Cancel
Command: nmap -sn 192.168.10.*
Hosts Services
Nmap Output Ports / Hosts Topology Host Details Scans
OS Host
192.168.10.1
192.168.10.2
192.168.10.3
192.168.10.12
192.168.10.108
192.168.10.109
192.168.10.126
192.168.10.127
192.168.10.200
192.168.10.216
192.168.10.217
192.168.10.220
vidyavardhini.com
192.168.10.226
192.168.10.254
Filter Hosts
nmap -sn 192.168.10.*
Details
MAC Address: 50:65:F3:5D:9D:5D (Hewlett Packard)
Nmap scan report for 192.168.10.12
Host is up (0.079s latency).
MAC Address: 3C:F8:62:34:F5:4E (Intel Corporate)
Nmap scan report for 192.168.10.108
Host is up (0.0070s latency).
MAC Address: 00:68:EB:BC:18:22 (HP)
Nmap scan report for 192.168.10.109
Host is up (0.0070s latency).
MAC Address: 00:68:EB:BC:23:78 (HP)
Nmap scan report for 192.168.10.126
Host is up (0.028s latency).
MAC Address: 00:68:EB:BC:16:8D (HP)
Nmap scan report for 192.168.10.200
Host is up (0.0030s latency).
MAC Address: 00:68:EB:BC:25:E5 (HP)
Nmap scan report for 192.168.10.216
Host is up (0.0020s latency).
MAC Address: 18:0F:76:35:3D:C8 (D-Link International)
Nmap scan report for 192.168.10.217
Host is up (0.099s latency).
MAC Address: 3C:F8:62:34:D4:4C (Intel Corporate)
Nmap scan report for 192.168.10.220
Host is up (0.097s latency).
MAC Address: 7E:94:77:8E:E1:92 (Unknown)
Nmap scan report for vidyavardhini.com (192.168.10.222)
Host is up (0.017s latency).
MAC Address: 30:24:A9:EB:31:D6 (HP)
Nmap scan report for 192.168.10.226
Host is up (0.097s latency).
MAC Address: 8C:B8:7E:BB:65:C0 (Intel Corporate)
Nmap scan report for 192.168.10.254
Host is up (0.010s latency).
MAC Address: 3C:52:82:0D:65:B2 (Hewlett Packard)
Nmap scan report for 192.168.10.127
Host is up.
Nmap done: 256 IP addresses (15 hosts up) scanned in 4.84 seconds
10:09 AM
10/6/2022



1. nmap -sn (reciever's ip address) :





- **Conclusion :**

Nmap is clearly the “Swiss Army Knife” of networking, thanks to its inventory of versatile commands. It lets you quickly scan and discover essential information about your network, hosts, ports, firewalls, and operating systems. Nmap has numerous settings, flags, and preferences that help system administrators analyze a network in detail.