SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

1904008 SECURITY LABORATORY

LAB MANUAL

VII SEMESTER

Regulation – 2019

Academic Year 2025 – 2026 (Odd Semester)

Prepared by

Mr. N. Leo Bright Tennisson, Assistant Professor (Sr. G) /CSE
Ms. M. Priyadharshini, Assistant Professor (O. G) /CSE
Ms. R. Anbuvizhi, Assistant Professor (O. G) /CSE

INDEX

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. To afford the necessary background in the field of Information Technology to deal with engineering problems to excel as engineering professionals in industries.
2. To improve the qualities like creativity, leadership, teamwork and skill thus contributing towards the growth and development of society.
3. To develop ability among students towards innovation and entrepreneurship that caters to the needs of Industry and society.
4. To inculcate and attitude for life-long learning process through the use of information technology sources.
5. To prepare then to be innovative and ethical leaders, both in their chosen profession and in other activities.

PROGRAMME OUTCOMES (POs)

After going through the four years of study, Information Technology Graduates will exhibit ability to:

PO#	Graduate Attribute	Programme Outcome
1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2	Problem analysis	Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	Design/development of solutions	Design solutions for complex engineering problemsand design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4	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
5	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.

6	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
7	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.
8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
9	Individual and team work	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings
10	Communication	Communicate effectively on complex engineering activities with the engineering community and with the 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
11	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
12	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

PROGRAM SPECIFIC OUTCOMES (PSOs)

By the completion of Information Technology program the student will have following Program specific outcomes

1. Design secured database applications involving planning, development and maintenance using state of the art methodologies based on ethical values.
2. Design and develop solutions for modern business environments coherent with the advanced technologies and tools.
3. Design, plan and setting up the network that is helpful for contemporary business environments using latest hardware components.
4. Planning and defining test activities by preparing test cases that can predict and correct errors ensuring a socially transformed product catering all technological needs.

SYLLABUS

IT8761 SECURITY LABORATORY

L T P C
0 0 4 2

OBJECTIVES:

The student should be made to:

- To learn different cipher techniques
- To implement the algorithms DES, RSA,MD5,SHA-1
- To use network security tools and vulnerability assessment tools

LIST OF EXPERIMENTS:

1. Perform encryption, decryption using the following substitution techniques
   - i. Ceaser cipher
   - ii. Playfair cipher
   - iii. Hill Cipher
   - iv. Vigenere cipher
2. Perform encryption and decryption using following transposition techniques

   Rail fence - Row & Column Transformation
3. Apply DES algorithm for practical applications.
4. Apply AES algorithm for practical applications
5. Implement RSA Algorithm using HTML and JavaScript
6. Implement the Diffie-Hellman Key Exchange algorithm for a given problem.
7. Calculate the message digest of a text using the SHA-1 algorithm
8. Implement the SIGNATURE SCHEME - Digital Signature Standard.
9. Demonstrate intrusion detection system (ids) using any tool eg. Snort or any other s/w.
10. Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool
11. Defeating Malware - Building Trojans, Rootkit Hunter

TOTAL: 60 PERIODS

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:

SOFTWARE:

- C / C++ / Java or equivalent compiler
- GnuPG, Snort, N-Stalker or Equivalent

HARDWARE:

- Standalone desktops - 30 Nos. (or) Server supporting 30 terminals or more.

OUTCOMES:

At the end of the course, the student should be able to

- Develop code for classical Encryption Techniques to solve theproblems.
- Build cryptosystems by applying symmetric and public key encryption algorithms.
- Construct code for authentication algorithms.
- Develop a signature scheme using Digital signature standard.
- Demonstrate the network security system using open source tools

COURSE OUTCOMES:

	CS476.1 To learn different cipher techniques.
	CS476.2 To implement the algorithms DES
	CS476.3 To implement the RSA Algorithm
	CS476.4 To implement the MD5, Digital Signature Algorithms.
	CS476.5 To use network security tools and vulnerability assessment tools

CO-PO MATRIX:

CO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CS476.1	3	-	3	-	-	-	-	-	-	-	-	-
CS476.2	-	3	-	-	3	-	-	-	-	-	-	-
CS476.3	-	3	3	-	3	-	-	-	-	-	-	-
CS476.4	-	-	-	2	3	-	-	-	-	-	1	-
CS476.5	-	-	-	2	3	3	-		-	1	1	1
CS476.N	3	3	3	2	-	3	-	-	-	-	-	-

CO-PSO MATRIX:

CO	PSO2
CS476.1	**2**
CS476.2	**2**
CS476.3	**2**
CS476.4	**2**
CS476.5	**2**
CS476.N	**2**

MODE OF ASSESSMENT

EVALUATION PROCEDURE FOR EACH EXPERIMENT

S.No	Description	Mark
1.	Aim & Pre-Lab discussion	20
2.	Observation	20
3.	Conduction and Execution	30
4.	Output & Result	10
5.	Viva	20
Total		**100**

INTERNAL ASSESSMENT FOR LABORATORY

S.No	Description	Mark
1.	Conduction & Execution of Experiment	10
2.	Record	5
3.	Model Test	5
Total		**20**

DESCRIPTION OF MAJOR SOFTWARE USED

JAVA

Java is a high-level programming language originally developed by Sun Microsystems. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.

Installation requires you to download an executable file available at the manual Java download page, which includes all the files needed for the complete installation at the user's discretion. There is no need to remain connected to the Internet during the installation. The file can also be copied to and installed on another computer that is not connected to the Internet. Administrative permission is required in order to install Java on Microsoft Windows

The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being more strict than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Unlike Windows executables (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE), which is available as a free download.

SNORT

Snort really isn’t very hard to use, but there are a lot of command line options to play with, and it’s not always obvious which ones go together well. This file aims to make using Snort easier for new users. Before we proceed, there are a few basic concepts you should understand about Snort. Snort can be configured to run in three modes:

- Sniffer mode, which simply reads the packets off of the network and displays them for you in a continuous stream on the console (screen).

- Packet Logger mode, which logs the packets to disk.

- Network Intrusion Detection System (NIDS) mode, which performs detection and analysis on network traffic. This is the most complex and configurable mode.

Snort is based on libpcap (for library packet capture), a **tool** that is widely **used** in TCP/IP traffic sniffers and analyzers. Through protocol analysis and content searching and matching, **Snort** detects attack methods, including denial of service, buffer overflow, CGI attacks, stealth port scans, and SMB probes

N-STALKER

N-STALKER is a world leader in Web Application Security solutions since 2000. It has started providing the first commercial and most complete HTTP Security Scanner, holding the largest signatures database available in the market – more than 39,000 attack signatures. Our products are delivered to hundreds of customers distributed in more than 30 different countries around the world. Back in 2000, N-Stalker's challenge was to provide complete solutions for your Web server infrastructure, which ended up with the release of N-Stealth HTTP Security Scanner. Nowadays, N-Stalker is seeking to provide the most complete solution for your enterprise web applications, the N-Stalker Web Application Security Scanner Suite.

EX.No.:1(a)	**CAESAR CIPHER**

AIM:

To implement a program for encrypting a plain text and decrypting a cipher text using Caesar Cipher (shift cipher) substitution technique

PRELAB DISCUSSION:

The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places 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 named after Julius Caesar, who apparently used it to communicate with his generals. More complex encryption schemes such as the Vigenere employ the Caesar cipher as one element of the encryption process. The widely known ROT13 'encryption' is simply a Caesar cipher with an offset of 13. The Caesar cipher offers essentially no communication security, and it will be shown that it can be easily broken even by hand.

To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. For the caesar cipher, the key is the number of characters to shift the cipher alphabet.

First we translate all of our characters to numbers, 'a'=0, 'b'=1, 'c'=2, ... , 'z'=25. We can now represent the caesar cipher encryption function, e(x), where x is the character we are encrypting, as:

$$e(x) = (x + k) \pmod{26}$$

Where k is the key (the shift) applied to each letter. After applying this function the result is a number which must then be translated back into a letter. The decryption function is :

$$e(x) = (x - k) \pmod{26}$$

ALGORITHM:

1. Create and initialize a string ALPHABET that holds the alphabet characters. The index position of the string represents the numeric representation for the corresponding characters in the string ALPHABET.
2. Read the input plain text to be encrypted and also the Caeser cipher key an integer between 0 and 25.
3. Encrypt the plain text using the Caeser cipher key and the ALPHABET string.
    a. For every character in the plain text
        i. Search the ALPHABET string for the character and assign the numeric representation of the character (plainnumeric) as the index position of the character in the ALPHABET string.

```
            ii. Perform encryption using
                ciphernumeric = ( plainnumeric + Caeser cipher key ) mod 26
           iii. Use ciphernumeric as the index position and get the corresponding character
                from the ALPHABET string as the equivalent cipher text character for the plain
                text character
       b. Print the equivalent cipher text
    4. Decrypt the cipher text using the Caeser cipher key and the ALPHABET string.
       a. For every character in the cipher text
             i. Search the ALPHABET string for the character and assign the numeric
                representation of the character (ciphernumeric) as the index position of the
                character in the ALPHABET string.
            ii. Perform decryption using
                Plainnumeric = ( ciphernumeric - Caeser cipher key ) mod 26,
                if plainnumeric < 0 , plainnumeric = plainnumeric + 26
           iii. Use plainnumeric as the index position and get the corresponding character
                from the ALPHABET string as the equivalent plain text character for the cipher
                text character
       b. Print the equivalent plain text

    5. Stop
```

PROGRAM:

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

public class Caesercipher
{
   public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";
   public static String  encrypt(String ptext, int cserkey)
   {
      String ctext = "";
      for (int i = 0; i < ptext.length(); i++)
      {

         int plainnumeric = ALPHABET.indexOf(ptext.charAt(i));
         int ciphernumeric =  (plainnumeric+cserkey) % 26;
         char cipherchar = ALPHABET.charAt(ciphernumeric);
         ctext += cipherchar;
      }
      return ctext;
   }

   public static String decrypt(String ctext, int cserkey)
   {
      String ptext = "";
```

```
        for (int i = 0; i < ctext.length(); i++)
        {
            int ciphernumeric = ALPHABET.indexOf(ctext.charAt(i));
            int plainnumeric=  (ciphernumeric-cserkey) % 26;
            if (plainnumeric < 0)
            {
                plainnumeric = ALPHABET.length() + plainnumeric;
            }
            char plainchar = ALPHABET.charAt(plainnumeric);
            ptext += plainchar;
        }
        return ptext;
    }

    public static void main(String[] args)
     throws IOException
     {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the PLAIN TEXT for Encryption: ");
        String plaintext = new String();
        String ciphertext = new String();
        String key;
        int cserkey;
        plaintext = br.readLine();
        System.out.println("Enter the CAESERKEY between 0 and 25:");
        key = br.readLine();
        cserkey = Integer.parseInt(key);

        System.out.println("ENCRYPTION");
        ciphertext = encrypt(plaintext,cserkey);
        System.out.println("CIPHER TEXT :"+ ciphertext);

        System.out.println("DECRYPTION");
        plaintext = decrypt(ciphertext,cserkey);
        System.out.println("PLAIN TEXT :" + plaintext);
    }
}
```

OUTPUT:

```
C:\Program Files\Java\jdk1.8.0_71\bin>javac Caesercipher.java

C:\Program Files\Java\jdk1.8.0_71\bin>java Caesercipher

Enter the PLAIN TEXT for Encryption:
information
Enter the CAESERKEY between 0 and 25:
7

ENCRYPTION
CIPHER TEXT :pumvythapvu

DECRYPTION
PLAIN TEXT :information
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. Crack the following plaintext TRVJRI TZGYVIJ RIV HLZKV VRJP KF TIRTB
2. What encryption key was used?
3. Make you own cipher text using the Caesar cipher.
4. Can you crack other people’s ciphertexts?
5. What key do we need to make “CAESAR” become “MKOCKB”?
6. What key do we need to make “CIPHER” become “SYFXUH”?
7. Use the Caesar cipher to encrypt your first name
8. How can we find the decryption key from the encryption key?

RESULT:

Thus the program to implement caeser cipher encryption technique was developed and executed.

EX.No.:1(b)	PLAY FAIR CIPHER

AIM:

To implement a program to encrypt a plain text and decrypt a cipher text using play fair Cipher substitution technique.

PRELAB DISCUSSION:

The Playfair cipher or Playfair square is a manual symmetric encryption technique and was the first literal digraph substitution cipher. Playfair cipher is a multi- alphabet letter encryption cipher, which deals with letters in plaintext as single units and renders these units into Ciphertext letters. The Playfair algorithm is based on the use of a 5X5 matrix of letters built using a keyword. The playfair cipher starts with creating a key table. The key table is a 5×5 grid of letters that will act as the key for encrypting your plaintext. Each of the 25 letters must be unique and one letter of the alphabet (usually Q) is omitted or treat I and J as the same alphabet from the table (as there are 25 spots and 26 letters in the alphabet). Let's say we wanted to use the phrase "Hello World" as our key. The first characters (going left to right) in the table will be the phrase, with duplicate letters removed. The rest of the table will be filled with the remaining letters of the alphabet, in order.

The text can only contain alphabets (i.e. no spaces or punctuation). Also this cipher is case-insensitive. We start by removing spaces from the text and duplicate letters from the key then converting them into uppercase. If any double letters occurring in the plaintext, an 'x' is inserted between the occurrences of the letters. In a playfair cipher the message is split into digraphs, pairs of two letters. If there is an odd number of letters, a Z or X is added to the last letter. Now for the actual encryption process, the Playfair cipher uses a few simple rules relating to where the letters of each digraph are in relation to each other. Performing this quick encryption process for each digraph in the message eventually results in the entire plaintext being encrypted. Decrypting the Playfair cipher (assuming you have the key) is as simple as doing the same process in reverse. Assuming you have the same key you will always be able to create the same key table, and then decrypt any messages made using that key.

The Playfair cipher was used mainly to protect important, yet non-critical secrets, as it is quick to use and requires no special equipment. By the time enemy cryptanalysts could break the code the information it was protecting would often no longer be relevant.

ALGORITHM:

1. Generate the (5 x 5) key table or matrix using the key.
    a. Fill the spaces in the table with the letters of the key without any duplication of letters.
    b. Fill the remaining spaces with the rest of the letters of the alphabet in order by having 'I' & 'J' in the same space or omitting 'Q' to reduce the alphabet to fit.

2. Remove any punctuation or characters from the plain text that are not present in the key square.
3. Identify any double letters in the plaintext and insert 'X' between the two occurrences.
4. Split the plain text into digraphs (groups of 2 letters)
5. Encryption: Locate the digraph letters in the key table
    a. If the letters appear on the same row of the table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).
    b. If the letters appear on the same column of the table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).
    c. If the letters are in different rows and columns, replace the pair with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first encrypted letter of the pair is the one that lies on the same row as the first plaintext letter.
    d. Append the letters referred from the key table using the steps 5.a, 5.b and 5.c to generate Cipher text.
6. Decryption: Split the cipher text into digraphs and locate the digraph letters in the key table.
    a. If the letters appear on the same row of the table, replace them with the letters to their immediate left respectively (wrapping around to the right side of the row if a letter in the original pair was on the left side of therow).
    e. If the letters appear on the same column of the table, replace them with the letters immediately above respectively (wrapping around to the bottom side of the column if a letter in the original pair was on the top side of the column).
    f. If the letters are in different rows and columns, replace the pair with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first encrypted letter of the pair is the one that lies on the same row as the first plaintext letter.
    g. Append the letters referred from the key table using the steps 6.a, 6.b and 6.c to generate Plain text.
7. Stop

PROGRAM:

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

public class Playfair
{
        private char pfmatrix[][] = new char[5][5];
        public String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        String plain,cipher;
        int jflg=0,xpad=0;
        int row,col;
```

```
public void matrixgen(String key)
{
        char keychar;
        int count=0;
        int alphacount=0;
        int p,k,flg=1;
        for(int i=0; i<5; i++)
        {
        for(int j=0;j<5;j++)
        {
        if (count<key.length())
        {
                keychar=key.charAt(count);
                if (keychar == 'J')
                        keychar='I';
                p=0;
                while (p<count)
                {
                        flg=1;
                        if (keychar==key.charAt(p))
                        {
                                count++;
                                if (count == key.length())
                                {
                                        flg=0;
                                        break;
                                }
                                keychar=key.charAt(count);
                                p=0;
                        }
                        else
                                p++;
                }

                if (flg!=0)
                {
                        pfmatrix[i][j]=keychar;
                        count++;
                }
                if ((count==key.length()) && (flg==0))

        {
                        if(alphacount<26)
                        {
                                        keychar=ALPHABET.charAt(alphacount);
                                        k=0;
                                        while (k<key.length())
                                        {
                                                if ((keychar==key.charAt(k)) || (keychar=='J'))
                                                {
                                                      alphacount++;
                                                      keychar=ALPHABET.charAt(alphacount);
                                                      k=0;
                }
                else
                    k++;
            }

            //if (keychar!='J' && k==key.length())
            pfmatrix[i][j]=keychar;
            alphacount++;
        }
    }
}
else
{

    if(alphacount<26)
    {
        keychar=ALPHABET.charAt(alphacount);
        k=0;
        while  (k<key.length())
        {
            if  ((keychar==key.charAt(k))  ||  (keychar=='J'))
            {
                alphacount++;
                keychar=ALPHABET.charAt(alphacount);
                k=0;
            }
            else
                k++;
        }
```

```
                    pfmatrix[i][j]=keychar;
                    alphacount++;
                }
            }
        }
    }

}


public  void matrixdisplay()
{
    for(int i=0;i<5;i++)
    {
        for(int  j=0;j<5;j++)
            System.out.print(pfmatrix[i][j] + " ");
        System.out.println();
    }
}

public  String  pfencryption(String  txt)
{
    int i,j,k;
    int ch1row,ch2row,ch1col,ch2col;
    char ch1,ch2,tmp1,tmp2;
    String  nutext="";String text=""; i=0;
    while   (i<(txt.length()-1))
    {
        text += txt.charAt(i);
        if  (txt.charAt(i)  ==  txt.charAt(i+1))
        {
            text += 'X';
            xpad++;
        }
        i++;
    }

    text += txt.charAt(txt.length()-1);
    System.out.println("TEXT : " + text);

    if  (text.length()%2  !=  0)
    {
        text += 'X';
        xpad++;
    }
    System.out.println("FINAL TEXT : "+ text);
```

```
        for(k=0;k<text.length();k=k+2)
        {
                ch1=text.charAt(k);
                ch2=text.charAt(k+1);
                System.out.println("CHARACTER PAIR :" + ch1 +" " + ch2);
                matsearch(ch1);
                ch1row=row;
                ch1col=col;
                matsearch(ch2);
                ch2row=row;
                ch2col=col;
//              System.out.println("ch1row:" + ch1row + "ch1col:" + ch1col);
//              System.out.println("ch2row:" +  ch2row + "ch2col:" + ch2col);
                if (ch1row==ch2row)
                {
                        tmp1=pfmatrix[ch1row][(ch1col+1)%5];
                        tmp2=pfmatrix[ch2row][(ch2col+1)%5];
                }
                else if (ch1col==ch2col)
                {
                        tmp1=pfmatrix[(ch1row+1)%5][ch1col];
                        tmp2=pfmatrix[(ch2row+1)%5][ch2col];
                }
                else
                {
                        tmp1=pfmatrix[ch1row][ch2col];
                        tmp2=pfmatrix[ch2row][ch1col];
                }

                nutext += tmp1;
                nutext += tmp2;
                System.out.println("TRANSLATED TEXT :" + tmp1 + " " + tmp2);
        }
```

```
return nutext;
 }

 public String pfdecryption(String text)
 {
        int i,j,k;
        int ch1row,ch2row,ch1col,ch2col;
        char ch1,ch2,tmp1,tmp2;
        String nutext="";
        String txt="";
        for(k=0;k<text.length();k=k+2)
        {
                ch1=text.charAt(k);
                ch2=text.charAt(k+1);
                System.out.println("CHARACTER PAIR :" + ch1 +" " + ch2);
                matsearch(ch1);
                ch1row=row;
                ch1col=col;
                matsearch(ch2);
                ch2row=row;
                ch2col=col;
//              System.out.println("ch1row:" + ch1row + "ch1col:" + ch1col);
//              System.out.println("ch2row:" + ch2row + "ch2col:" + ch2col);
                if (ch1row==ch2row)
                {
                        int c1,c2;
                        c1 = ch1col-1;
                        if (c1<0)
                                c1 = c1+5;
                        c2 = ch2col-1;
                        if (c2<0)
                                c2=c2+5;
                        tmp1=pfmatrix[ch1row][c1];
                        tmp2=pfmatrix[ch2row][c2];
                }
                else if (ch1col==ch2col)
                {
                        int r1,r2;
                        r1=ch1row-1;
                        r2=ch2row-1;
                        if (r1<0)
                                r1=r1+5;
                        if (r2<0)
                                r2=r2+5;
                        tmp1=pfmatrix[r1][ch1col];
                        tmp2=pfmatrix[r2][ch2col];
                }
                else
                {
                        tmp1=pfmatrix[ch1row][ch2col];
                        tmp2=pfmatrix[ch2row][ch1col];
                }
                nutext += tmp1;


                nutext += tmp2;
                System.out.println("TRANSLATED TEXT :" + tmp1 + " " + tmp2);
        }
        if (xpad != 0)
        {
                i=0;
                while (i<nutext.length())
                {
                        if (nutext.charAt(i) == 'X')
                        {
                                i++;
                                continue;
                        }
                        txt += nutext.charAt(i);
                        i++;
                }
                System.out.println("TEXT :" + txt);
                return txt;
        }
        else
        {
                System.out.println("TEXT : " + nutext);
                return nutext;
        }
}


public void matsearch(char ch)
{
        int i,j;
        if (ch=='J')
        {
                ch='I';
                jflg=1;
        }
        for(i=0;i<5;i++)
        {
                for(j=0;j<5;j++)
                {
                        if (pfmatrix[i][j] == ch)
                        {
                                row=i;
                                col=j;
                        }
                }
        }
}
```

```
public static void main(String[] args)
{
        Playfair pf = new Playfair();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the PLAYFAIR KEY: ");
        String pfkey = new String();
        pfkey = sc.next();
        System.out.println("PLAYFAIR MATRIX");
        pf.matrixgen(pfkey);
        pf.matrixdisplay();
        String ptext = new String();
        System.out.println("Enter PLAIN TEXT");
        ptext = sc.next();
        String ctext = new String();
        ctext = pf.pfencryption(ptext);
        System.out.println();
        System.out.println("CIPHER TEXT :" + ctext);
        System.out.println();
        String plaintext = new String();
        plaintext = pf.pfdecryption(ctext);
        System.out.println();
        System.out.println("PLAIN TEXT :" + plaintext);
        sc.close();
}
}
```

OUTPUT:

```
C:\Program Files\Java\jdk1.8.0_71\bin>javac Playfair.java

C:\Program  Files\Java\jdk1.8.0_71\bin>java  Playfair

Enter the PLAYFAIR KEY:

INFORMATION

PLAYFAIR MATRIX

I   N F O  R

M A T  B  C D

E G  H  K L  P

Q  S  U V  W

X  Y  Z

Enter PLAIN TEXT

GOODMORNING

TEXT  :  GOXODMORNING

FINAL TEXT : GOXODMORNING

CHARACTER PAIR      :G O

TRANSLATED TEXT :H F

CHARACTER PAIR      :X O

TRANSLATED TEXT :Y F

CHARACTER PAIR :D M

TRANSLATED TEXT :L D
```

```
CHARACTER PAIR :O R
TRANSLATED TEXT :R I
CHARACTER PAIR : N I
TRANSLATED TEXT :F N
CHARACTER PAIR : N G
TRANSLATED TEXT :F E

CIPHER TEXT :HFYFLDRIFNFE

CHARACTER PAIR :H F
TRANSLATED TEXT :G O
CHARACTER PAIR :Y F
TRANSLATED TEXT :X O
CHARACTER PAIR :L D
TRANSLATED TEXT :D M
CHARACTER PAIR :R I
TRANSLATED TEXT :O R
CHARACTER PAIR :F N
TRANSLATED TEXT :N I
CHARACTER PAIR :F E
TRANSLATED TEXT :N G
TEXT :GOODMORNING

PLAIN TEXT :GOODMORNING
C:\Program Files\Java\jdk1.8.0_71\bin>
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. What is difference between a monoalphabetic and a polyalphabetic cipher?
2. What are stream cipher and block cipher and how are they different?
3. How many possible keys does the playfair cipher have?
4. How to find the keyword of playfair cipher, given the plain text and ciphertext?
5. Construct a table for the Playfair Cipher with the keyword EFFECTIVENESS?

RESULT:

Thus the program to implement Play Fair Cipher technique was developed and executed successfully.

EX.No.:1(c)	HILL CIPHER

AIM:

To develop a program to encrypt and decrypt using the Hill cipher substitution technique

PRELAB DISCUSSION:

The Hill cipher is a substitution cipher invented by Lester S. Hill in 1929. Hill's major contribution was the use of mathematics to design and analyse cryptosystems. The **Hill cipher** is a polygraphic substitution **cipher** based on linear algebra. Hill ciphers are applications of linear algebra because a Hill cipher is simply a linear transformation represented by a matrix with respect to the standard basis. Groups of letters are represented by vectors. The domain of the linear transformation is all plaintext vectors, while the codomain is made up of all ciphertext vectors. Matrix multiplication is involved in the encoding and decoding process. And when trying to find the inverse key, we will use elementary row operations to row reduce the key matrix in order to find its inverse in the standard manner. Each letter is represented by a number modulo 26. To encrypt a message, each block of n letters is multiplied by an invertible n × n matrix, again modulus 26.To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).The cipher can, be adapted to an alphabet with any number of letters. All arithmetic just needs to be done modulo the number of letters instead of modulo 26.
Encryption : Cipher text = (Plain text * Key) mod 26
Decryption : Plain text = (Cipher text * Key^{-1}) mod 26

ALGORITHM:

1. Get the n-by-n key matrix with vectors of length n.
2. Separate the plaintext from left to right into some number k of groups of n letters each. If you run out of letters when forming the final group, repeat the last plaintext letter or ‘X’ as many times as needed to fill out the final group of n letters.
3. Replace each letter by the corresponding number of its position (from 0 through m-1) in the alphabet to get k groups of n integers each.
4. Encryption:
   a. Reshape each of the k groups of integers into an n-row column vector called Plain text matrix
   b. Cipher Text Matrix = (Plain Text Matrix * Key Matrix) mod 26.
   c. Using step 4.b, perform matrix multiplication of Plain text matrix and Key matrix and modulus 26 to yield Cipher text matrix.
   d. Translate the Cipher text matrix to string representation by replacing each of the entries with the corresponding letter of the alphabet, after arranging all k of the resulting column vectors in order into a single vector of length k x n
5. Decryption:
   a. Find the inverse of the key matrix
      i. Find the determinant of the key matrix

```
                ii.  Transpose the key matrix
               iii.  Find minor matrix and then Cofactor of the key matrix
                iv.  Key$^{-1}$ = [ [ Det(key matrix) ]$^{-1}$ * Cofactor ] mod 26 using modulus
                     arithmetic
       b. Plain Text = ( Cipher Text * Key$^{-1}$ ) mod 26
       c. Using step 5.b, perform matrix multiplication of Cipher text matrix and Key inverse
          matrix and modulus 26 to yield Plain text matrix.
       d. Translate the Plain text matrix to string representation by replacing each of the entries
          with the corresponding letter of the alphabet, after arranging all k of the resulting
          column vectors and removing any letters padded in order into a single vector of length
          k x n
   6. Stop
```

PROGRAM:

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

public class Hillcipher
{
	public int keyinverse[][] = new int[3][3];
	public int key[][] = { {17, 17, 5}, {21, 18, 21}, {2, 2, 19} };
	public int plainmat[][] = new int[8][3];
	public int ciphermat[][] = new int[8][3];
	public String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
	String plain,cipher;
	int row, flag=0, decrypt=0;

	public void matdisplay(int mat[][])
	{
		int i, j;
		for(i=0;i<row;i++)
		{
			for(j=0;j<3;j++)
				System.out.print(mat[i][j] + " ");
		System.out.println();
		}
	}

	public void keydisplay(int mat[][])
	{
		int i, j;
		for(i=0;i<3;i++)
		{
			for(j=0;j<3;j++)
				System.out.print(mat[i][j] + " ");
		System.out.println();
		}
	}

	public void inverse()
	{
		int dtrmnt = 0;
		int mulinvdtrmnt = 0;
		int x, y, z, i, j, a, tmp, p, q;
		int transkey[][] = new int[3][3];
		int minormat[][] = new int[3][3];
		int temp[][] = new int[2][2];

		System.out.println("HILL CIPHER KEY");
		keydisplay(key);


		 x = key[0][0] * (( key[1][1] * key[2][2]) - (key[1][2] * key[2][1]));
		 y = key[0][1] * (( key[1][0] * key[2][2]) - (key[1][2] * key[2][0]));
		 z = key[0][2] * (( key[1][0] * key[2][1]) - (key[1][1] * key[2][0]));

		dtrmnt = (x-y+z)%26;
		if ( dtrmnt < 0 )
			dtrmnt = dtrmnt + 26;
		System.out.println("DETERMINANT :"+ dtrmnt);

		a = dtrmnt;
		for(i=0;i<25;i++)
		{
			tmp = ( a * i ) % 26;
			if ( tmp == 1 )
			{
				mulinvdtrmnt = i;
				break;
			}
		}
		System.out.println("MULTIPLICATIVE INVERSE OF DETERMINANT:" +
mulinvdtrmnt);


		for(i=0;i<3;i++)
		{
			 for(j=0;j<3;j++)
				transkey[i][j] = key[j][i];
		}
//		keydisplay(transkey);


		for(i=0;i<3;i++)
		{
			for(j=0;j<3;j++)
			{
				p=0;
				q=0;
				for(x=0;x<3;x++)
				{
					for(y=0;y<3;y++)
					{


						if ((x!=i) && (y!=j))
```

```
                                        {
                                                temp[p][q] = transkey[x][y];
                                                q++;

                                                if ( q == 2)
                                                {
                                                        q=0;
                                                        p++;
                                                }
                                        }
                                }
                        }
                minormat[i][j]=(temp[0][0]*temp[1][1])-(temp[0][1]*temp[1][0]);
                minormat[i][j] = minormat[i][j] * (int)Math.pow(-1,(i+j));
                }
        }

        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                {
                        keyinverse[i][j]=(mulinvdtrmnt * minormat[i][j])%26;
                        if (keyinverse[i][j] < 0)
                                keyinverse[i][j] = keyinverse[i][j] + 26;
                }
        }
        System.out.println("KEY INVERSE");
        keydisplay(keyinverse);
}


public void str2matrix(String text)
{

        int k, p, n;


        if ((text.length() % 3) == 1)
        {
                n=text.length();
                text += 'X';
                text += 'X';
                flag = 2;
        }
        if ((text.length() % 3) == 2)
        {
                text += 'X';
                flag = 1;
        }
        row = (text.length()) / 3;

        k = 0;
        for(int i=0; i<row; i++)


        {
                for(int j=0;j<3;j++)
                {
                    for (p=0;p<26;p++)
                    {


                        if (text.charAt(k) == ALPHABET.charAt(p))
                        {
                            plainmat[i][j] = p;
                            k++;
                            break;
                        }
                    }
                }
            }
//          System.out.println("PLAIN TEXT MATRIX");
            matdisplay(plainmat);
            System.out.println();
        }

        public String matrix2str(int mat[][])
        {
            int i, j, k;
            String txt="";
            String tmp="";

            for(i=0;i<row;i++)
            {
                for(j=0;j<3;j++)
                {
                    k = mat[i][j];
                    txt += ALPHABET.charAt(k);
                }
            }
            if (decrypt == 1)
            {
                if ( flag == 1 )
                {
                    tmp = txt.substring(0, (txt.length()-1));
                    return tmp;
                }
                if ( flag == 2 )
                {
                    tmp = txt.substring(0, (txt.length()-2));
                    return tmp;
                }
            }
            return txt;
        }

        public String hcencryption(String ptxt)
        {
            int i,j,k;
            int sum=0;
            String ctxt="";

            decrypt=0;
            System.out.println("HILL CIPHER ENCRYPTION");
            System.out.println("PLAIN TEXT MATRIX");
            str2matrix(ptxt);

            for(i=0;i<row;i++)
            {
                        for(j=0;j<3;j++)
                        {
                                    for(k=0;k<3;k++)
                                                sum += plainmat[i][k] * key[k][j];
                                    ciphermat[i][j] = sum % 26;
                                    sum = 0;
                        }
            }

            System.out.println("CIPHER TEXT MATRIX");
            matdisplay(ciphermat);
            ctxt = matrix2str(ciphermat);
            return ctxt;
}

public String hcdecryption(String ctxt)
{
            int i,j,k;
            int sum=0;
            String ptxt="";

            decrypt=1;
            System.out.println("HILL CIPHER DECRYPTION");
            System.out.println("CIPHER TEXT MATRIX");
            str2matrix(ctxt);

            for(i=0;i<row;i++)
            {
                        for(j=0;j<3;j++)
                        {
                                    for(k=0;k<3;k++)
                                                sum += ciphermat[i][k] * keyinverse[k][j];
                                    plainmat[i][j] = sum % 26;
                                    sum = 0;
                        }
            }
            System.out.println("PLAIN TEXT MATRIX");
            matdisplay(plainmat);
            ptxt = matrix2str(plainmat);
            return ptxt;
}

public static void main(String[] args)
{
```

```
                Hillcipher hc = new Hillcipher();
        Scanner sc = new Scanner(System.in);

                hc.inverse();


                String ptext = new String();
                System.out.println("Enter PLAIN TEXT");
                ptext = sc.next();

                String ctext = new String();
                ctext = hc.hcencryption(ptext);
                System.out.println();
                System.out.println("CIPHER TEXT :" + ctext);
                System.out.println();

                String plaintext = new String();
                plaintext =hc.hcdecryption(ctext);
                System.out.println();
                System.out.println("PLAIN  TEXT  :"  +  plaintext);

                sc.close();
        }
}
```

OUTPUT:

```
C:\Program Files\Java\jdk1.8.0_71\bin>javac Hillcipher.java
C:\Program  Files\Java\jdk1.8.0_71\bin>java  Hillcipher

HILL CIPHER KEY
17 17  5
21 18  21
2     2 19

DETERMINANT  :  23

MULTIPLICATIVE INVERSE OF DETERMINANT : 17

KEY INVERSE
4     9  15
15  17   6
24   0  17

Enter PLAIN TEXT
PAYMOREMONEY

HILL CIPHER ENCRYPTION
PLAIN TEXT MATRIX
15 0  24
12  14  17
4    12  14
13   4   24
```

```
CIPHER TEXT MATRIX
17 17  11
12 22  1
10 0 18
15   3    7

CIPHER  TEXT  :RRLMWBKASPDH

HILL CIPHER DECRYPTION
CIPHER TEXT MATRIX
17 17  11
12 22  1
10 0 18
15   3    7

PLAIN  TEXT  MATRIX
 15   0  24
 12  14  17
  4   12  14
 13    4 24

PLAIN  TEXT  :PAYMOREMONEY

C:\Program Files\Java\jdk1.8.0_71\bin>javac Hillcipher.java
C:\Program  Files\Java\jdk1.8.0_71\bin>java  Hillcipher

HILL CIPHER KEY
17 17  5
21 18  21
2    2 19

DETERMINANT  :23
MULTIPLICATIVE  INVERSE  OF  DETERMINANT:17

KEY INVERSE
4     9 15
15  17   6
24 0 17

Enter PLAIN TEXT
TECHNOLOGY

HILL CIPHER ENCRYPTION
PLAIN TEXT MATRIX
19 4 2
7    13  14
11 14  6
24 23  23
```

```
CIPHER TEXT MATRIX
 21  9   9
 4  17  2
 25  9  21
 1  10   0

CIPHER TEXT :VJJERCZJVBKA

HILL CIPHER DECRYPTION
CIPHER TEXT MATRIX
21   9 9
4    17  2
25   9 21
1    10  0

PLAIN TEXT MATRIX
19 4 2
7    13  14
11 14  6
24 23  23

PLAIN TEXT :TECHNOLOGY
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. Name the aspects to be considered of information security.
2. What is meant by deciphering?
3. What are the two different uses of public key cryptography related to key distribution?
4. How many bit keys areused in S-DES algorithm?
5. What are the parameters are include the certificate request message?
6. What is S/MIME?
7. What is the purpose of dual signature?
8. What are the two common techniques used to protect a password file?
9. What is the size of the key for substitution block cipher?
10. Define Stream Cipher.

RESULT:

Thus the program to implement Hill cipher encryption technique was developed and executed successfully.

EX.No.:1(d)	VIGENERE CIPHER

AIM:

To develop a program to implement encryption and decryption using vigenere cipher substitution technique

PRELAB DISCUSSION:

The Vigenère Cipher was developed by mathematician Blaise de Vigenère in the 16th century. The Vigenère Cipher was adapted as a twist on the standard Caesar cipher to reduce the effectiveness of performing frequency analysis on the ciphertext. The cipher accomplishes this using uses a text string (for example, a word) as a key, which is then used for doing a number of alphabet shifts on the plaintext. Similar to the Caesar Cipher, but instead of performing a single alphabet shift across the entire plaintext, the Vigenère cipher uses a key to determine several different shift amounts across the entirety of the message. The Vigenère cipher uses a 26×26 table with **A** to **Z** as the row heading and column heading This table is usually referred to as the *Vigenère Tableau*, *Vigenère Table* or *Vigenère Square*. We shall use *Vigenère Table*. The first row of this table has the 26 English letters. Starting with the second row, each row has the letters shifted to the left one position in a cyclic way. For example, when **B** is shifted to the first position on the second row, the letter **A** moves to the end.

In addition to the plaintext, the Vigenère cipher also requires a keyword, which is repeated so that the total length is equal to that of the plaintext. In this way, each letter in the plaintext is shifted by the alphabet number of the corresponding letter in the key. To encrypt, pick a letter in the plaintext and its corresponding letter in the keyword, use the keyword letter and the plaintext letter as the row index and column index, respectively, and the entry at the row-column intersection is the letter in the ciphertext. Repeating this process until all plaintext letters are processed.

To decrypt, pick a letter in the ciphertext and its corresponding letter in the keyword, use the keyword letter to find the corresponding row, and the letter heading of the column that contains the ciphertext letter is the needed plaintext letter. For example, to decrypt the first letter **T** in the ciphertext, we find the corresponding letter **H** in the keyword. Then, the row of **H** is used to find the corresponding letter **T** and the column that contains **T** provides the plaintext letter **M** (see the above figures). Consider the fifth letter **P** in the ciphertext. This letter corresponds to the keyword letter **H** and row **H** is used to find **P**. Since **P** is on column **I**, the corresponding plaintext letter is **I**.

ALGORITHM :

1. Vigenere table consists of the alphabet 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.
2. The Key is repeated so that the total length is equal to that of the plaintext. In this way, each letter in the plaintext is shifted by the alphabet number of the corresponding letter in the key.
3. At different points in the encryption process, the cipher uses a different alphabet from one of the rows used.

4. **Encryption:** The the plaintext(P) and key(K) are added modulo 26.

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

5. **Decryption: S**ubtract the key from the Encrypted (Cipher) text and perform modulo 26 to arrive back at the original, plaintext value

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

6. Stop

PROGRAM:

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

public class Vigenerecipher
{
        public static String key = new String();
        public String extndkey;
        public String plaintxt, ciphertxt;
        public String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
//      String plain,cipher;
        int row, flag=0, decrypt=0;

        public String keyextnsn(String ptxt,String keytxt)
        {
                int i,j=0,n;
                String nukey="";

                for(i=0;i<ptxt.length();i++)
                {
                        nukey += keytxt.charAt(j);
                        j++;
                        if (j == keytxt.length())
                                j=0;
                }
                return nukey;
        }

        public int valueofchar(char x)
        {
                int i,pos=0;

                for(i=0;i<26;i++)
                {
                        if ( x == ALPHABET.charAt(i))
                        {
```

```
                pos = i;
                break;
            }
        }
        return pos;
    }

    public char charofvalue(int y)
    {
        int i;
        char ch;

        ch = ALPHABET.charAt(y);
        return ch;
    }

    public String vcencryption(String txt)
    {
        int i,j,p=0,k=0,tmp1=0;
        char tmp;
        String ctxt="";

        extndkey = keyextnsn(txt,key);
        System.out.println("VIGENERE ENCRYPTION");
        System.out.println("PLAIN TEXT : " + txt);
        System.out.println("VIGENERE KEY : " + extndkey);
        for(i=0;i<txt.length();i++)
        {
            p = valueofchar(txt.charAt(i));
            k = valueofchar(extndkey.charAt(i));

        //  System.out.println("p : " + p + " k : " + k);
            tmp1 = ( p + k )%26;
            tmp = charofvalue(tmp1);
        //  System.out.println("tmp1 : " + tmp1 + "tmp : " + tmp);

            ctxt += tmp;
//          System.out.println("CTXT : " + ctxt);
        }
        return ctxt;
    }

    public String vcdecryption(String txt)
    {
        int i,c=0,k=0,tmp1=0;
        char ch;
        String ptxt="";

        System.out.println("VIGENERE DECRYPTION");
        System.out.println("CIPHER TEXT : " + txt);
        System.out.println("VIGENERE KEY : " + extndkey);
        for(i=0;i<txt.length();i++)
        {
            c = valueofchar(txt.charAt(i));
            k = valueofchar(extndkey.charAt(i));

            tmp1 = ( c - k + 26 )%26;
            ch = charofvalue(tmp1);

            ptxt += ch;
        }
        return ptxt;
    }

    public static void main(String[] args)
    {
        Vigenerecipher vc = new Vigenerecipher();
        Scanner sc = new Scanner(System.in);

        System.out.println("ENTER KEY");
        key = sc.next();
        String text = new String();
        System.out.println("Enter PLAIN TEXT");
        text = sc.next();

        String ciphertext = new String();
        ciphertext = vc.vcencryption(text);
        System.out.println();
        System.out.println("CIPHER TEXT :" + ciphertext);
        System.out.println();

        String plaintext = new  String();
        plaintext =vc.vcdecryption(ciphertext);
        System.out.println();
        System.out.println("PLAIN  TEXT  :"  +  plaintext);

        sc.close();
    }
}
```

OUTPUT:

```
C:\Program Files\Java\jdk1.8.0_71\bin>javac Vigenerecipher.java
C:\Program Files\Java\jdk1.8.0_71\bin>java Vigenerecipher
ENTER KEY
DECEPTIVE
Enter PLAIN TEXT
WEAREDISCOVEREDSAVEYOURSELF
VIGENERE ENCRYPTION
PLAIN TEXT : WEAREDISCOVEREDSAVEYOURSELF
VIGENERE  KEY  :  DECEPTIVEDECEPTIVEDECEPTIVE

CIPHER TEXT :ZICVTWQNGRZGVTWAVZHCQYGLMGJ

VIGENERE DECRYPTION
CIPHER  TEXT  :  ZICVTWQNGRZGVTWAVZHCQYGLMGJ
VIGENERE KEY : DECEPTIVEDECEPTIVEDECEPTIVE

PLAIN  TEXT  :WEAREDISCOVEREDSAVEYOURSELF

C:\Program Files\Java\jdk1.8.0_71\bin>java Vigenerecipher
ENTER KEY
LEMON
Enter PLAIN TEXT
ATTACKATDAWN
VIGENERE ENCRYPTION
PLAIN TEXT : ATTACKATDAWN
VIGENERE KEY : LEMONLEMONLE

CIPHER  TEXT  :LXFOPVEFRNHR

VIGENERE  DECRYPTION
CIPHER TEXT : LXFOPVEFRNHR
VIGENERE KEY : LEMONLEMONLE

PLAIN TEXT :ATTACKATDAWN

C:\Program  Files\Java\jdk1.8.0_71\bin>
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. What is convert channel?
2. Give the principle advantages of elliptical curve cryptography.
3. What are the counter measures for timing attach?
4. Give all the generic types of attacks.
5. How secure is DES?
6. What is the necessity of firewalls?
7. Between symmetric Vs Public Key cryptography, which method is more convenient?
8. What are the requirements of a hash function?
9. For what purpose the hash function is used?
10. What is Kerberos?

RESULT:

Thus the program to implement Vigenere cipher encryption technique was developed and executed successfully.

EX.No.: 2	RAIL FENCE CIPHER

AIM:

To develop a program for implementing encryption and decryption using rail fence transposition technique.

PRELAB DISCUSSION:

The rail fence is the simplest example of a class of transposition ciphers, known as route ciphers. In general, the elements of the plaintext (usually single letters) are written in a prearranged order (route) into a geometric array (matrix)—typically a rectangle—agreed upon in advance by the transmitter and receiver and then read off by following another prescribed route through the matrix to produce the cipher. The key in a route cipher consists of keeping secret the geometric array, the starting point, and the routes. Clearly both the matrix and the routes can be much more complex than in this example; but even so, they provide little security. One form of transposition (permutation) that was widely used depends on an easily remembered key word for identifying the route in which the columns of a rectangular matrix are to be read. For example, using the key word AUTHOR and ordering the columns by the lexicographic order of the letters in the key word.

In decrypting a route cipher, the receiver enters the cipher text symbols into the agreed-upon matrix according to the encryption route and then reads the plaintext according to the original order of entry. A significant improvement in crypto security can be achieved by reencrypting the cipher obtained from one transposition with another transposition. Because the result (product) of two transpositions is also a transposition, the effect of multiple transpositions is to define a complex route in the matrix, which in itself would be difficult to describe by any simple mnemonic.

ALGORITHM DESCRIPTION:

In the rail fence cipher, the plaintext is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail. When we reach the top rail, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows.

1. Generate numerical key from the word key by the characters of the word in alphabetical order.

2. Encryption
   i. The plain text is written in the matrix form, where the column of the matrix is number of characters in the word key and row of the matrix is to accommodate the characters of the plain text and the space left after the plain text characters in the last row is filled with any character. (eg. x or z)
   ii. The cipher text is generated by reading the characters column by column in the order specified in the numerical key.

3. Decryption
    i. The characters in the cipher text are filled in the matrix of same order used for encryption, but in the order specified in the key. The characters from cipher text equal to the number of rows in matrix are taken and filled in the matrix column based on the order specified in the key
    ii. The plain text is generated from cipher text by reading the characters from the matrix row by row.
4. Stop

PROGRAM:

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

public class Railfence
{
        public static int key[] = new int[8];
        public char mat[][] = new char[10][8];
        public char pmat[][] = new char[10][8];
        public char cmat[][] = new char[10][8];
        String plain="";
        String cipher="";
        int rows=0, col;

        public String rfencryption(String text1)
        {
                int i,j,len,ch,k,p=0;
                String enctxt="";
                String text = "";
                len = text1.length();

                for(i=0;i<len;i++)
                        text += text1.charAt(i);

                if (( len % 7 ) != 0)
                {
                        rows = ( len / 7 ) + 1;
                        ch = len % 7;
                        for (i=0;i<(7-ch);i++)
                                text += 'X';
                }
                else
                        rows = len / 7;
                k=0;

                for(i=1;i<=rows;i++)
                {
```

```
        for(j=1;j<=7;j++)
            mat[i][j] = text.charAt(k++);
    }

    for(i=1;i<=rows;i++)
    {
        for(j=1;j<=7;j++)
            System.out.print(mat[i][j] + " ");
        System.out.println();
    }

    k = 1;
    j = 1;
    while ( k <= 7 )
    {
        for(p=0;p<7;p++)
        {
            if ( k == key[p] )
            {
                j=p+1;
                k++;
                break;
            }
        }
        for(i=1;i<=rows;i++)
            enctxt+=mat[i][j];
    }

    System.out.println(enctxt);
    return enctxt;
}

public String rfdecryption(String txt,int plength)
{
    int i,j=1,len,k=1,p,q=0;
    String dectxt="";
    String ptext="";

    while (k<=7)
    {
        for(p=0;p<7;p++)
        {
            if (key[p] == k)
            {
                j = p+1;
                k++;
```

```
                        break;
                    }
                }
                for(i=1;i<=rows;i++)
                    cmat[i][j] = txt.charAt(q++);
            }

            for(i=1;i<=rows;i++)
            {
                for(j=1;j<=7;j++)
                    System.out.print(cmat[i][j] + " ");
                System.out.println();
            }

            for(i=1;i<=rows;i++)
            {
                for(j=1;j<=7;j++)
                    dectxt += cmat[i][j];
            }

            len = dectxt.length();
            if (plength < len)
            {
                for(i=0;i<plength;i++)
                    ptext += dectxt.charAt(i);
            }

            return ptext;
        }

        public static voidmain(String[] args)
        throws IOException
        {
            int i=0;
            int k;
            String c;
            BufferedReader br = new BufferedReader(new  InputStreamReader(System.in));
            Railfence rf = new Railfence();
            Scanner sc = new Scanner(System.in);

            System.out.println("Enter key");
            for(i=0;i<7;i++)
            {
                c = br.readLine();
                key[i] = Integer.parseInt(c);
            }
```

```
                for(i=0;i<7;i++)
                        System.out.print(key[i]  +  "  ");

                String plain = new String();
                System.out.println("Enter PLAIN TEXT");
                plain = sc.next();
                k=plain.length();

                System.out.println(plain);
                String ctext = new String();
                ctext = rf.rfencryption(plain);
                System.out.println();
                System.out.println("CIPHER TEXT :" + ctext);
                System.out.println();

                String plaintext = new String();
                plaintext = rf.rfdecryption(ctext,k);
                System.out.println();
                System.out.println("PLAIN  TEXT  :"  +  plaintext);

                sc.close();
        }
}
```

OUTPUT:

```
C:\Program Files\Java\jdk1.8.0_71\bin>javac Railfence.java
C:\Program  Files\Java\jdk1.8.0_71\bin>java  Railfence
Enter key
 AUTHOR
 A U T H O R
 A H O R T U
 KEY : NUMERICAL REPRESENTATION
 0 3 4 5 2 1
 Enter PLAIN TEXT ATTACKPOSTPONEDUNTILTWOAM
 ATTACKPOSTPONEDUNTILTWOAM


 A T T A C K
 P O S T P O
 N E D U N T
 I L T W O A
 M X X X X X
 APNIMKOTAXCPNOXTOELXTSDTXATUWX
 CIPHER  TEXT  :APNIMKOTAXCPNOXTOELXTSDTXATUWX

A T T A C K
P O S T P O
N E D U N T
I L T W O A
M X X X X X

 PLAIN  TEXT  :ATTACKPOSTPONEDUNTILTWOAM

C:\Program  Files\Java\jdk1.8.0_71\bin>
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. Where do you apply PGP?
2. List out the basic tasks in Public Key Encryption in key distribution.
3. Give an example for Simple Hash Function.
4. List out the two methods of operations in Authentication Header (AH) and Encapsulating Security Payload (ESP).
5. Enumerate the functions provided by S/MIME.
6. List out the two ways in which password can be protected.
7. Which attack is related to integrity?
8. Which public key cryptosystem can be used for digital signature?
9. Expand: S/MIME.
10. What is the use of trusted system?

RESULT:

Thus the program for Railfence cipher was executed and verified successfully.

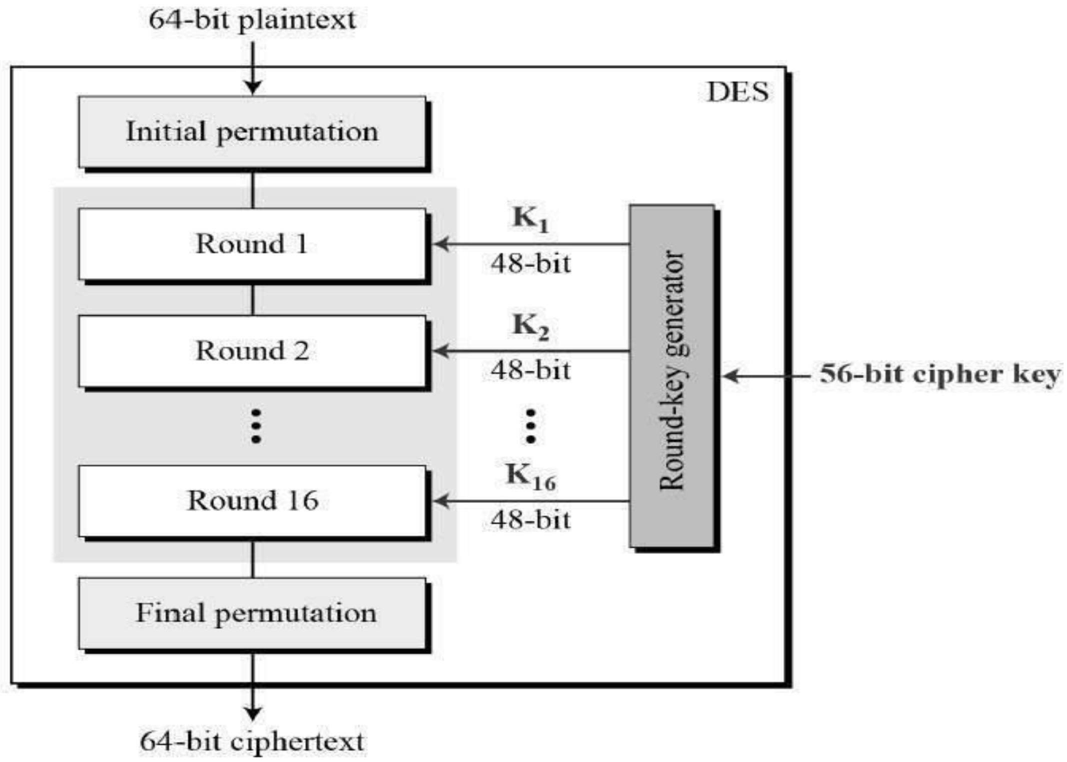
EX.No.: 3	DATA ENCRYPTION STANDARD (DES)

AIM:

To develop a program to implement Data Encryption Standard for encryption and decryption.

PRELAB DISCUSSION:

- The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
- DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit.
- Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).
- General Structure of DES is depicted in the following illustration



ALGORITHM:

1. Process the key.
    i. Get a 64-bit key from the user.
    ii. Calculate the key schedule.
        1. Perform the following permutation on the 64-bit key. The parity bits are discarded, reducing the key to 56 bits. Bit 1 of the permuted block is bit 57 of the original key, bit 2 is bit 49, and so on with bit 56 being bit 4 of the original key.
        2. Split the permuted key into two halves. The first 28 bits are called C[0] and the last 28 bits are called D[0].

3. Calculate the 16 subkeys. Start with i = 1.

    1. Perform one or two circular left shifts on both C[i-1] and D[i-1] to get C[i] and D[i], respectively. The number of shifts per iteration are given in the table below.

        Iteration # 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

        Left Shifts 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 1

    2. Permute the concatenation C[i]D[i] as indicated below. This will yield K[i], which is 48 bits long.

    3. Loop back to 1.ii.c.1 until K[16] has been calculated.

2 Process a 64-bit data block.

i. Get a 64-bit data block. If the block is shorter than 64 bits, it should be padded as appropriate for the application.

ii. Perform the initial permutation on the data block.

iii. Split the block into two halves. The first 32 bits are called L[0], and the last 32 bits are called R[0].

iv. Apply the 16 subkeys to the data block. Start with i = 1.

a. Expand the 32-bit R[i-1] into 48 bits according to the bit-selection function Expansion (E)

b. Exclusive-or E(R[i-1]) with K[i].

c. Break E(R[i-1]) xor K[i] into eight 6-bit blocks. Bits 1-6 are B[1], bits 7-12 are B[2], and so on with bits 43-48 being B[8].

d. Substitute the values found in the S-boxes for all B[j]. Start with j = 1. All values in the S-boxes should be considered 4 bits wide.

i. Take the 1st and 6th bits of B[j] together as a 2-bit value (call it m) indicating the row in S[j] to look in for the substitution.

ii. Take the 2nd through 5th bits of B[j] together as a 4-bit value(call it n) indicating the column in S[j] to find the substitution.

iii. Replace B[j] with S[j][m][n].

iv. Loop back to 2.iv.d.i until all 8 blocks have been replaced.

e. Permute the concatenation of B[1] through B[8]

f. Exclusive-or the resulting value with L[i-1]. Thus, all together, your R[i] = L[i-1] xor P(S[1](B[1])...S[8](B[8])), where B[j] is a 6-bit block of E(R[i-1]) xor K[i]. (The function for R[i] is written as, R[i] = L[i-1] xor f(R[i-1], K[i]).)

g. L[i] = R[i-1].

h. Loop back to 2.iv.a until K[16] has been applied.

v. Perform the final permutation on the block R[16]L[16].

3. Decryption : Use the keys K[i] in reverse order. That is, instead of applying K[1] for the first iteration, apply K[16], and then K[15] for the second, on down to K[1]

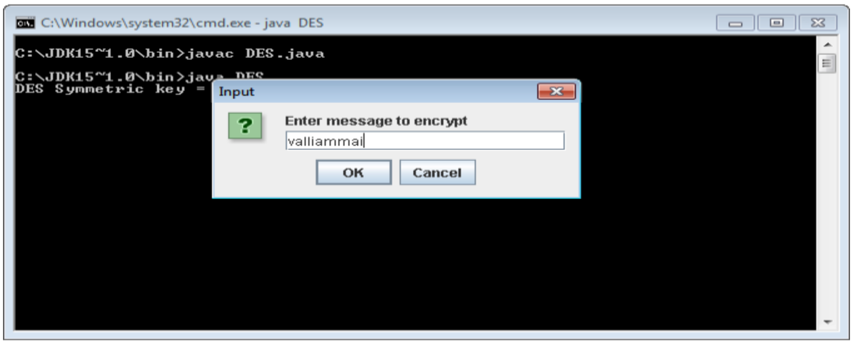
PROGRAM:

DES :-

```
import javax.swing.*;
import java.security.SecureRandom;
import  javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
class DES  {
byte[] skey = new byte[1000];
String skeyString;
static byte[] raw;
String inputMessage, encryptedData, decryptedMessage;



public DES() {
try {
generateSymmetricKey();
inputMessage=JOptionPane.showInputDialog(null,"Enter message to encrypt");
byte[] ibyte = inputMessage.getBytes();
byte[]  ebyte=encrypt(raw,  ibyte);
String encryptedData = new String(ebyte);
System.out.println("Encrypted  message  "+encryptedData);
JOptionPane.showMessageDialog(null,"Encrypted  Data  "+"\n"+encryptedData);
byte[] dbyte= decrypt(raw,ebyte);
String decryptedMessage = new String(dbyte);
System.out.println("Decrypted  message  "+decryptedMessage);
JOptionPane.showMessageDialog(null,"Decrypted  Data  "+"\n"+decryptedMessage);
}
catch(Exception e) {
System.out.println(e);
}
}

void generateSymmetricKey() {
try {
Random r = new Random();
intnum = r.nextInt(10000);
String knum = String.valueOf(num);
byte[] knumb = knum.getBytes();
skey=getRawKey(knumb);
skeyString = new String(skey);
System.out.println("DES  Symmetric  key  =  "+skeyString);
}
catch(Exception e) {
System.out.println(e);
}
}
```

```
private static byte[] getRawKey(byte[] seed) throws Exception
{
 KeyGeneratorkgen = KeyGenerator.getInstance("DES");
SecureRandomsr= SecureRandom.getInstance("SHA1PRNG");
sr.setSeed(seed);



kgen.init(56, sr);
SecretKeyskey = kgen.generateKey();
raw = skey.getEncoded();
return raw;
}


private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
SecretKeySpecskeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal(clear);
return encrypted;
}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {
SecretKeySpecskeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.DECRYPT_MODE, skeySpec);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
}

public static void main(String args[]) {
DES des = new DES();
}
}
```

OUTPUT:

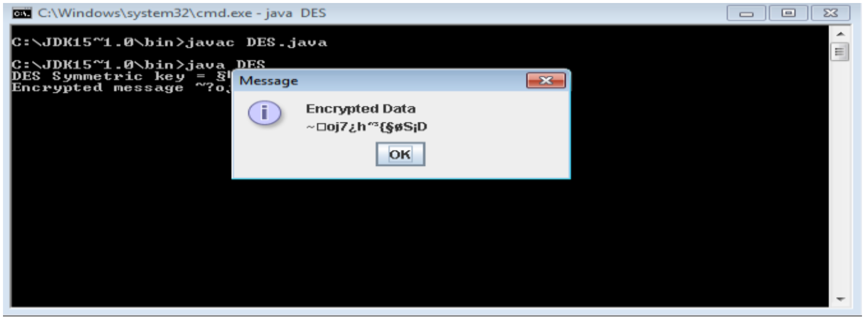


```
C:\JDK15~1.0\bin>javac DES.java

C:\JDK15~1.0\bin>java DES
DES Symmetric key =
```

Input

Enter message to encrypt

valliammai

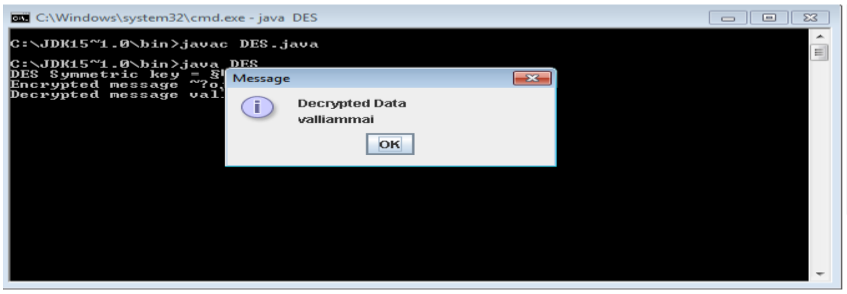


C:\Windows\system32\cmd.exe - java DES

```
C:\JDK15~1.0\bin>javac DES.java

C:\JDK15~1.0\bin>java DES
DES Symmetric key = §
Encrypted message ~?o
```

Message

Encrypted Data
~□oj7¿h^³{§øS¡D



C:\Windows\system32\cmd.exe - java DES

```
C:\JDK15~1.0\bin>javac DES.java

C:\JDK15~1.0\bin>java DES
DES Symmetric key = §
Encrypted message ~?o
Decrypted message val
```

Message

Decrypted Data
valliammai

VIVA QUESTIONS (PRELAB and POSTLAB):

1. DES follows which basic stream cipher?
2. The DES Algorithm Cipher System consists of how many rounds (iterations) each with a round key?
3. What is the key length of the DESalgorithm?
4. In the DES algorithm, although the key size is 64 bits only 48bits are used for the encryption procedure, the rest are parity bits. Is it true or false?
5. In the DES algorithm, what is the size of the round key and the Round Input?
6. In the DES algorithm how the Round Input is expanded to 48 ?
7. What is size of the Initial Permutation table/matrix
8. How many unique substitution boxes are in DES after the 48 bit XOR operation?
9. In the DES algorithm the 64 bit key input is shortened to 56 bits by ignoring every 4th bit. Is it true or false?

RESULT:

Thus the program to implement DES encryption technique was developed and executed successfully

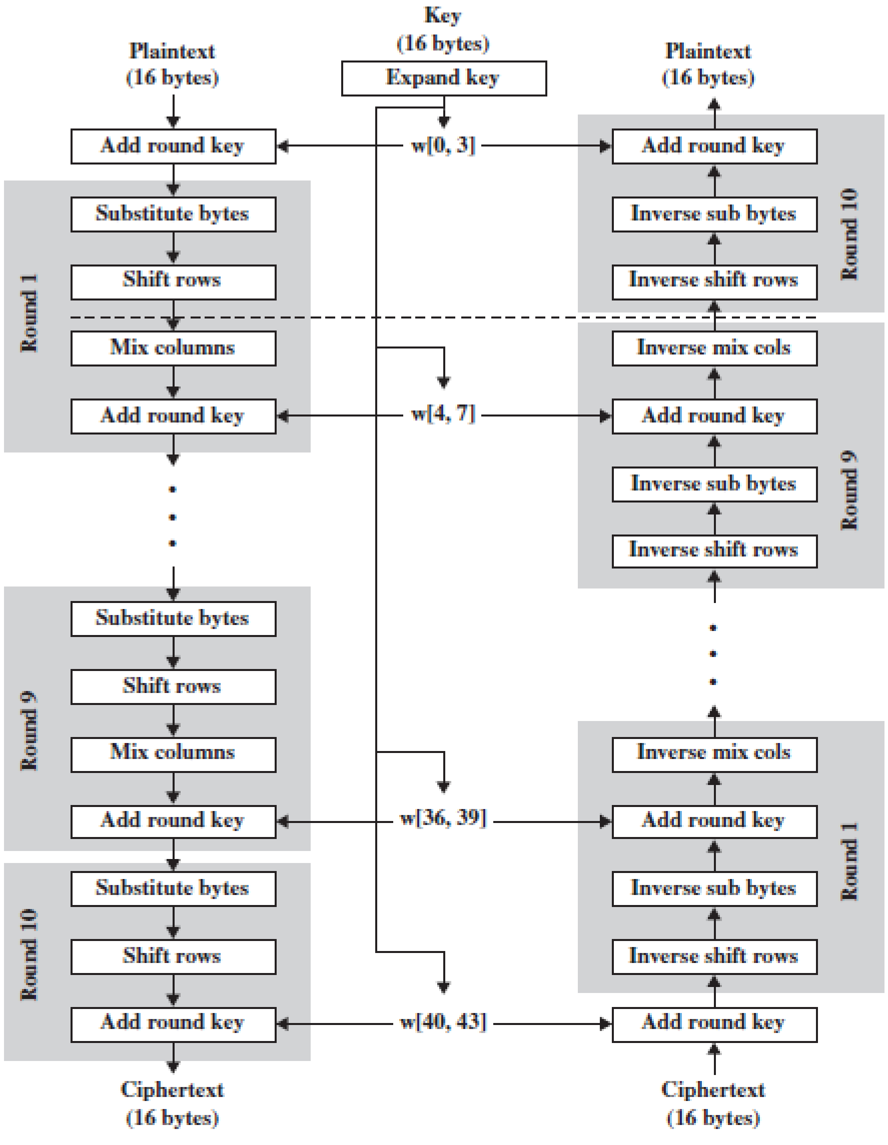
EX.No.: 4	AES ALGORITHM

AIM:

To develop a program to implement Advanced Encryption Standard for encryption and decryption.

PRELAB DISCUSSION:

The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.



(a) Encryption (b) Decryption

The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a 4 * 4 square matrix of bytes. This block is copied into the **State** array, which is modified at each stage of encryption or decryption. After the final stage, **State** is copied to an output matrix. Similarly, the key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key. Note that the ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the **in** matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the **w** matrix.

The cipher consists of *N* rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key. The first *N* - 1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The final round contains only three transformations, and there is a initial single transformation (AddRoundKey) before the first round, which can be considered Round 0. Each transformation takes one or more 4 * 4 matrices

PROGRAM:

```
package com.includehelp.stringsample;

import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

/**
 * Program to Encrypt/Decrypt String Using AES 128 bit Encryption Algorithm
 */
public class EncryptDecryptString
{
        private static final String encryptionKey           = "ABCDEFGHIJKLMNOP";
        private static final String characterEncoding        = "UTF-8";
        private static final String cipherTransformation     = "AES/CBC/PKCS5PADDING";
        private static final String aesEncryptionAlgorithem = "AES";
   /**
   * Method for Encrypt Plain String Data
   * @param plainText
   * @return encryptedText
   */
        public static String encrypt(String plainText)
        {
                String encryptedText = "";
                try
                {
                     Cipher cipher = Cipher.getInstance(cipherTransformation);
                     byte[] key      = encryptionKey.getBytes(characterEncoding);
                     SecretKeySpec secretKey = new SecretKeySpec(key, aesEncryptionAlgorithem);
                     IvParameterSpec ivparameterspec = new IvParameterSpec(key);
                     cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivparameterspec);
                     byte[] cipherText = cipher.doFinal(plainText.getBytes("UTF8"));
```

```
              Base64.Encoder encoder = Base64.getEncoder();
              encryptedText = encoder.encodeToString(cipherText);
          } catch (Exception E)
          {
                System.err.println("Encrypt Exception : "+E.getMessage());
           }
          return encryptedText;
      }
/**

 * Method For Get encryptedText and Decrypted provided String
 * @param encryptedText
 * @return decryptedText
 */
    public static String decrypt(String encryptedText)
    {
          String decryptedText = "";
          try
          {
                Cipher cipher = Cipher.getInstance(cipherTransformation);
                byte[] key = encryptionKey.getBytes(characterEncoding);
              SecretKeySpec secretKey = new SecretKeySpec(key, aesEncryptionAlgorithem);
                IvParameterSpec ivparameterspec = new IvParameterSpec(key);
                cipher.init(Cipher.DECRYPT_MODE, secretKey, ivparameterspec);
                Base64.Decoder decoder = Base64.getDecoder();
                byte[] cipherText = decoder.decode(encryptedText.getBytes("UTF8"));
                decryptedText = new String(cipher.doFinal(cipherText), "UTF-8");
          } catch (Exception E)
          {
                System.err.println("decrypt Exception : "+E.getMessage());
          }
          return decryptedText;
    }

    public static void main(String[] args)
    {
          Scanner sc = new Scanner(System.in);
          System.out.println("Enter String : ");
          String plainString = sc.nextLine();

          String encyptStr = encrypt(plainString);
          String decryptStr = decrypt(encyptStr);

          System.out.println("Plain   String   :   "+plainString);
          System.out.println("Encrypt   String   :   "+encyptStr);
```

```
        System.out.println("Decrypt String : "+decryptStr);
} }
```

OUTPUT:

```
Enter String : Hello World
Plain String : Hello World
Encrypt String : IMfL/ifkuvkZwG/v2bn6Bw==
Decrypt String : Hello World
```

VIVA QUESTIONS (PRELAB and POSTLAB):

This set of Cryptography Multiple Choice Questions & Answers (MCQs) focuses on “The Data Encryption Standard (DES) and It’s Strength”.

1. DES follows

a) Hash Algorithm b)Caesars Cipher c)Feistel Cipher Structure d)SP Networks

2. The DES Algorithm Cipher System consists of ___________ rounds (iterations) each with a round key

a) 12 b) 18 c) 9 d) 16

3. The DES algorithm has a key length of

a) 128 Bits b) 32 Bits c) 64 Bits d) 16 Bits

4. In the DES algorithm, although the key size is 64 bits only 48bits are used for the encryption procedure, the rest are parity bits.

a) True b) False

5. In the DES algorithm the round key is___________bit and the Round Input is _____________bits.

a) 48, 32 b) 64,32 c) 56, 24 d) 32, 32

6. In the DES algorithm the Round Input is 32 bits, which is expanded to 48 bits via ______ ___

a) Scaling of the existing bits b) Duplication of the existing bits

c) Addition of zeros d) Addition of ones

7. The Initial Permutation table/matrix is of size

a) 16×8 b) 12×8 c) 8×8 d) 4×8

8. The number of unique substitution boxes in DES after the 48 bit XOR operation are

a) 8 b) 4 c) 6 d) 12

9. In the DES algorithm the 64 bit key input is shortened to 56 bits by ignoring every 4th bit.

a) True b) False

RESULT:

Thus the program to implement AES encryption technique was developed and executed successfully.

EX.No.: 5	RSA ALGORITHM

AIM:

Develop a program to implement RSA algorithm for encryption and decryption. This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars **Ron Rivest, Adi Shamir,** and **Len Adleman** and hence, it is termed as RSA cryptosystem. The two aspects of the RSA cryptosystem, firstly generation of key pair and secondly encryption-decryption algorithms

PRELAB DISCUSSION:

Generation of RSA Key Pair

- Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key.
- The process followed in the generation of keys is described below −
- Generate the RSA modulus (n)

  Select two large primes, p and q.

  Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.
- Find Derived Number (e)

  Number e must be greater than 1 and less than (p − 1)(q − 1).

  There must be no common factor for e and (p − 1)(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.
- Form the public key

  The pair of numbers (n, e) form the RSA public key and is made public.

  Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.
- Generate the private key

  Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.

  Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1).
- This relationship is written mathematically as follows ed = 1 mod (p − 1)(q − 1)
- The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

ALGORITHM:

1. Key Generation
   1. Choose two distinct prime numbers p and q.
   2. Find n such that n = pq, n will be used as the modulus for both the public and private keys.
   3. Find the totient of n, $\phi(n)$ $\phi(n)=(p-1)(q-1)$
   4. Choose an e such that $1 < e < \phi(n)$, and such that e and $\phi(n)$ share no divisors other than 1 (e and $\phi(n)$ are relatively prime). e is kept as the public

key exponent

v. Determine d (using modular arithmetic) which satisfies the congruence relation

$de \equiv 1 \pmod{\phi(n)}$.

The public key has modulus n and the public (or encryption) exponent e. The private key has modulus n and the private (or decryption) exponent d, which is kept secret.

2. Encryption

   $c \equiv m^e \pmod{n}$.

3. Decryption:

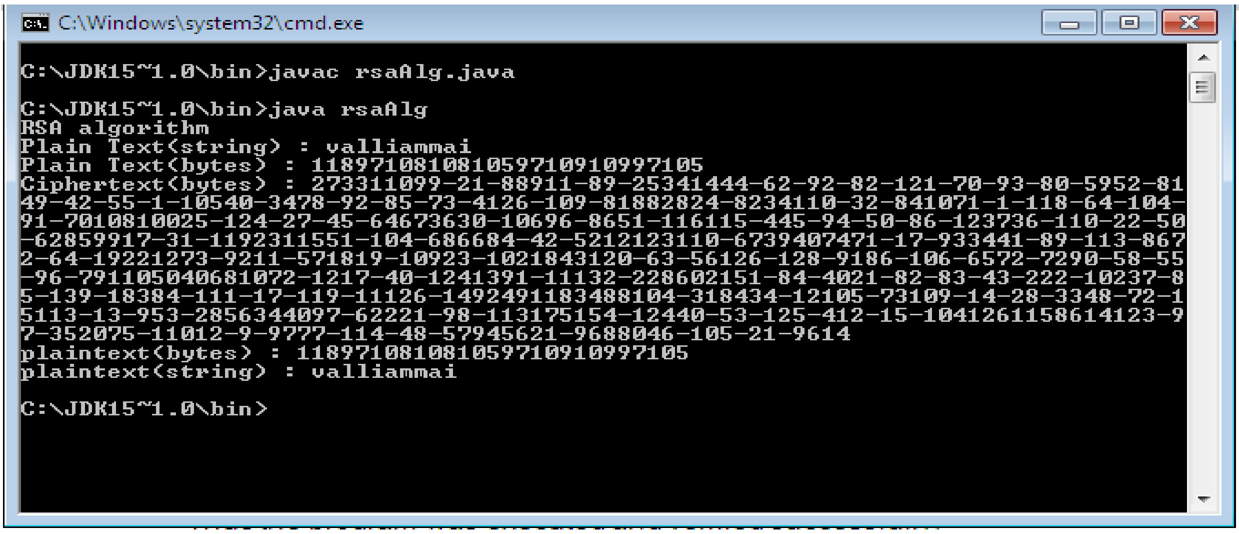
   $m \equiv c^d \pmod{n}$.

4. Stop.

PROGRAM:

```
import java.math.BigInteger;
import java.util.Random;
import java.io.*;
class rsaAlg
{
private BigInteger p, q, n, phi, e, d; /* public key components */
private int bitLen = 1024;
private int blkSz = 256; /* block size in bytes */
private Random rand;
/* convert bytes to string */
private static String bytesToString(byte[] encrypted)
{
String str = "";
for (byte b : encrypted)
{
str += Byte.toString(b);
}
return str;
}
/* encrypt message */
public byte[] encrypt(byte[] msg)
{
return (new BigInteger(msg)).modPow(e, n).toByteArray();
}
/* decrypt message */
public byte[] decrypt(byte[] msg)
{
return (new BigInteger(msg)).modPow(d, n).toByteArray();
}
/* calculate public key components p, q, n, phi, e, d */
public rsaAlg()
{


rand = new Random();
p  =  BigInteger.probablePrime(bitLen, rand);
q = BigInteger.probablePrime(bitLen, rand);
n = p.multiply(q);
phi  =  p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
e = BigInteger.probablePrime(bitLen/2, rand);
while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 &&
e.compareTo(phi) < 0)
{
e.a dd(BigInteger.ONE);
}
d = e.modInverse(phi);
}
public rsaAlg (BigInteger e, BigInteger d, BigInteger n)
{
this.e = e;
this.d = d;
this.n = n;
}
public static void main (String[] args) throws java.lang.Exception
{
rsaAlg rsaObj = new rsaAlg();
String msg = "Hello world! Security Laboratory";
System.out.println("simulation of RSA algorithm");
System.out.println("message(string) : " + msg);
System.out.println("message(bytes) : " +
bytesToString(msg.getBytes()));
/* encrypt test message */
byte[] ciphertext = rsaObj.encrypt(msg.getBytes());
System.out.println("ciphertext(bytes)  :  "  +  bytesToString(ciphertext));
/* decrypt ciphertext */
byte[] plaintext = rsaObj.decrypt(ciphertext);
System.out.println("plaintext(bytes) : " + bytesToString(plaintext));
System.out.println("plaintext(string)  :  "  +  new  String(plaintext));  }  }
```

Output:



```
C:\JDK15~1.0\bin>javac rsaAlg.java

C:\JDK15~1.0\bin>java rsaAlg
RSA algorithm
Plain Text(string) : valliammai
Plain Text(bytes) : 11897108108105971091099 7105
Ciphertext(bytes) : 273311099-21-88911-89-25341444-62-92-82-121-70-93-80-5952-81
49-42-55-1-10540-3478-92-85-73-4126-109-81882824-8234110-32-841071-1-118-64-104-
91-7010810025-124-27-45-64673630-10696-8651-116115-445-94-50-86-123736-110-22-50
-62859917-31-1192311551-104-686684-42-5212123110-6739407471-17-933441-89-113-867
2-64-19221273-9211-571819-10923-1021843120-63-56126-128-9186-106-6572-7290-58-55
-96-791105040681072-1217-40-1241391-11132-228602151-84-4021-82-83-43-222-10237-8
5-139-18384-111-17-119-11126-1492491183488104-318434-12105-73109-14-28-3348-72-1
5113-13-953-2856344097-62221-98-113175154-12440-53-125-412-15-1041261158614123-9
7-352075-11012-9-9777-114-48-57945621-9688046-105-21-9614
plaintext(bytes) : 1189710810810597109109 97105
plaintext(string) : valliammai

C:\JDK15~1.0\bin>
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. What Is RSA?
2. Are Strong Primes Necessary In RSA?
3. Can Users Of RSA Run Out Of Distinct Primes?
4. What Are The Alternatives To RSA?
5. How fast is RSA?
6. What would it take to break RSA?
7. How large a modulus key should be used in RSA?
8. How large should the primes be?
9. How do you know if a number is prime?
10. How is RSA used for authentication in practice?
11. What do you mean by Secret Key Cryptography and Public Key Cryptography? How they are different from one another?
12. What exactly do you know about RSA?

RESULT:

Thus the program for implementation of RSA algorithm was executed and verified successfully.

EX.No.: 6	DIFFIEE HELLMAN KEY EXCHANGE ALGORITHM

AIM:

Develop a program to implement Diffie Hellman Key Exchange Algorithm for encryption and Decryption.

PRELAB DISCUSSION:

Diffie–Hellman key exchange (D–H) is a specific method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. This algorithm uses arithmetic modulus as the basis of its calculation. Suppose Alice and Bob follow this key exchange procedure with Eve acting as a man in middle interceptor (or the bad guy).

Here are the calculation steps followed in this algorithm that make sure that eve never gets to know the final keys through which actual encryption of data takes place. First, both Alice and Bob agree upon a prime number and another number that has no factor in common. Lets call the prime number as **p** and the other number as **g**. Note that **g** is also known as the generator and **p** is known as prime modulus. Now, since eve is sitting in between and listening to this communication so eve also gets to know **p** and **g**. Now, the modulus arithmetic says that **r** = (**g** to the power **x**) mod **p.** So **r** will always produce an integer between 0 and **p**. The first trick here is that given **x** (with **g** and **p** known) , its very easy to find **r**. But given **r** (with **g** and **p** known) its difficult to deduce **x**. One may argue that this is not that difficult to crack but what if the value of **p**is a very huge prime number? Well, if this is the case then deducing **x** (if **r** is given) becomes almost next to impossible as it would take thousands of years to crack this even with supercomputers. This is also called the discrete logarithmic problem. Coming back to the communication, all the three Bob, Alice and eve now know **g** and **p**. Now, Alice selects a random private number **xa** and calculates (**g** to the power **xa**) mod **p** = **ra**. This resultant **ra** is sent on the communication channel to Bob. Intercepting in between, eve also comes to know **ra**. Similarly Bob selects his own random private number **xb**, calculates (**g** to the power **xb**) mod **p** = **rb** and sends this **rb** to Alice through the same communication channel. Obviously eve also comes to know about **rb**. So eve now has information about **g**, **p**, **ra** and **rb**. Now comes the heart of this algorithm. Alice calculates (**rb** to the power **xa**) mod **p** = **Final key** which is equivalent to **(g to the power (xa*xb)) mod p**. Similarly Bob calculates **(ra to the power xb) mod p** = **Final key** which is again equivalent to **(g to the power(xb * xa)) mod p**. So both Alice and Bob were able to calculate a common **Final key** without sharing each others private random number and eve sitting in between will not be able to determine the **Final key** as the private numbers were never transferred.

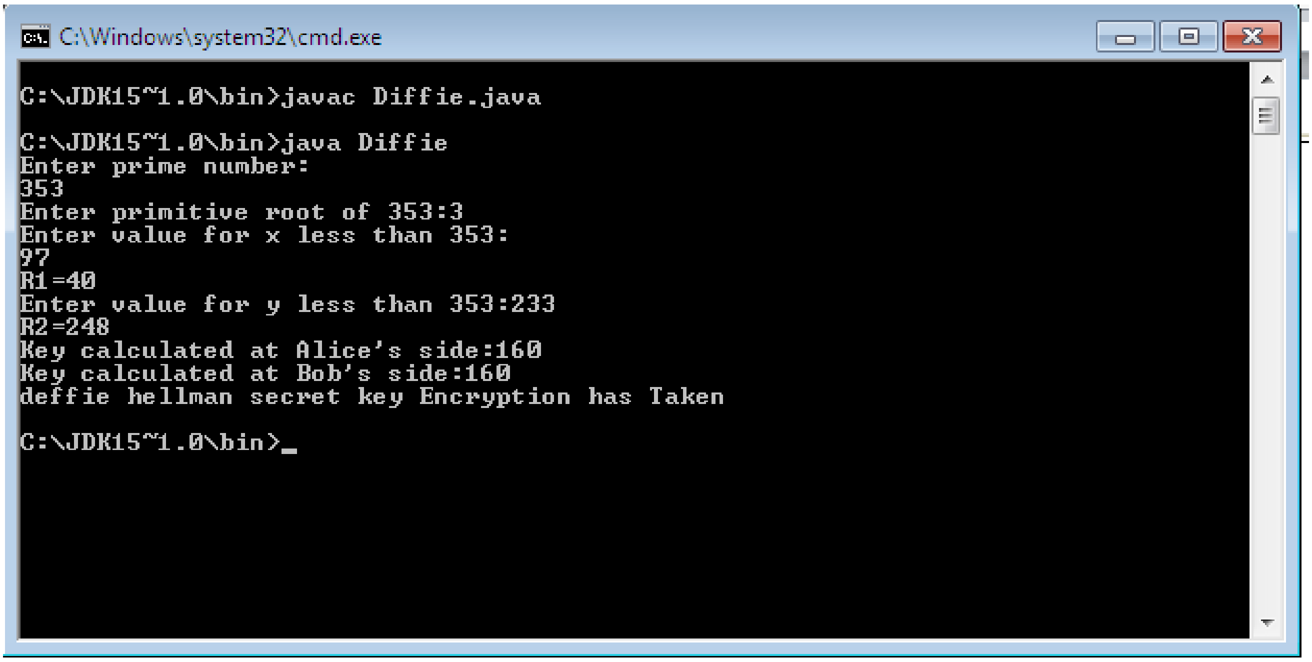
ALGORITHM

```
1. Global Public Elements:
        Let q be a prime number and $\alpha$ where $\alpha < q$ and $\alpha$ is a primitive root of q.
2. User A Key Generation:
    i.  Select private $X_A$ where $X_A < q$
    ii. Calculate public $Y_A$ where $Y_A = \alpha^{XA}$ mod q
3. User B Key Generation:
    i.  Select private $X_B$ where $X_B < q$
    ii. Calculate public $Y_B$ where $Y_B = \alpha^{XB}$ mod q
4. Calculation of Secret Key by User
        A: $K = (Y_B)^{XA}$ mod q
5. Calculation of Secret Key by User
        B: $K = (Y_A)^{XB}$ mod q
```

PROGRAM:

```
import java.io.*;
import java.math.BigInteger;
class Diffie
{
   public static void main(String[]args)throws IOException
   {
      BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Enter prime number:");
      BigInteger p=new BigInteger(br.readLine());
      System.out.print("Enter primitive root of "+p+":");
      BigInteger g=new BigInteger(br.readLine());
      System.out.println("Enter value for x less than "+p+":");
      BigInteger x=new BigInteger(br.readLine());
      BigInteger R1=g.modPow(x,p);
      System.out.println("R1="+R1);
      System.out.print("Enter value for y less than "+p+":");
      BigInteger y=new BigInteger(br.readLine());
      BigInteger R2=g.modPow(y,p);
      System.out.println("R2="+R2);
      BigInteger k1=R2.modPow(x,p);
      System.out.println("Key calculated at Sender's side:"+k1);
      BigInteger k2=R1.modPow(y,p);
      System.out.println("Key calculated at Receiver's side:"+k2);
      System.out.println("deffie hellman secret key Encryption has Taken");
   }
}
```

OUTPUT:



```
C:\JDK15~1.0\bin>javac Diffie.java

C:\JDK15~1.0\bin>java Diffie
Enter prime number:
353
Enter primitive root of 353:3
Enter value for x less than 353:
97
R1=40
Enter value for y less than 353:233
R2=248
Key calculated at Alice's side:160
Key calculated at Bob's side:160
deffie hellman secret key Encryption has Taken

C:\JDK15~1.0\bin>_
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. What's the difference between Diffie-Hellman and RSA?
2. Does Diffie Hellman guarantee secrecy?
3. Why is RSA preferred over Diffie-Hellman if they are both used to establish shared key?
4. Are there any one way operations that could be used for Diffie-Hellman post quantum?
5. Why is Diffie-Hellman required whenRSA is already used for key exchange in TLS?
6. What is Authenticated Diffie-Hellman Key Agreement?
7. How secure is ECDH if the public keys are never shared?
8. Which is better when the secret is leaked, RSA or Diffie-Hellman?
9. What role does RSA play in DH-RSA cipher suite?
10. Why is Diffie-Hellman used alongside public keys?
11. Is Diffie-Hellman key exchange based on one-way function or trapdoor function?

RESULT:

Thus the program to implement Diffie-Hellman Key Exchange algorithm was developed and executed successfully

EX.No.: 7	IMPLEMENT SECURE HASH FUNCTION (SHA)

AIM:

Develop a program to implement Secure Hash Algorithm (SHA-1)

PRELAB DISCUSSION:

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest - typically rendered as a hexadecimal number, 40 digits long.

Secure Hashing Algorithms, also known as SHA, are a family of cryptographic functions designed to keep data secured. It works by transforming the data using a hash function: an algorithm that consists of bitwise operations, modular additions, and compression functions. The hash function then produces a fixed size string that looks nothing like the original. These algorithms are designed to be one-way functions, meaning that once they're transformed into their respective hash values, it's virtually impossible to transform them back into the original data. A few algorithms of interest are SHA-1, SHA-2, and SHA-5, each of which was successively designed with increasingly stronger encryption in response to hacker attacks. SHA-0, for instance, is now obsolete due to the widely exposed vulnerabilities.

A common application of SHA is to encrypting passwords, as the server side only needs to keep track of specific user's hash value, rather than the actual password. This is helpful in case an attacker hacks the database, as they will only find the hashed functions and not the actual passwords, so if they were to input the hashed value as a password, the hash function will convert it into another string and subsequently deny access. Additionally, SHA exhibit the avalanche effect, where the modification of very few letters being encrypted cause a big change in output; or conversely, drastically different strings produce similar hash values. This effect causes hash values to not give any information regarding the input string, such as its original length. In addition, SHAs are also used to detect the tampering of data by attackers, where if a text file is slightly changed and barely noticeable, the modified file's hash value will be different than the original file's hash value, and the tampering will be rather noticeable.

ALGORITHM:

1. Append Padding Bits: Message is "padded" with a 1 and as many 0's as necessary to bring the message length to 64 bits less than an even multiple of 512.
2. Append Length: 64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the originalmessage.
3. Prepare Processing Functions: SHA1 requires 80 processing functions defined as:

```
f(t;B,C,D) = (B AND C) OR ((NOT B) AND D)          ( 0 <= t <= 19)
f(t;B,C,D) = B XOR C XOR D                         (20 <= t <= 39)
f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t<=59)
f(t;B,C,D) = B XOR C XOR D                         (60 <= t <= 79)
```

4. Prepare Processing Constants: SHA1 requires 80 processing constant words defined as:

   K(t) = 0x5A827999 (0 <= t <= 19)
   K(t) = 0x6ED9EBA1 (20 <= t <= 39)
   K(t) = 0x8F1BBCDC (40 <= t <= 59)
   K(t) = 0xCA62C1D6 (60 <= t <= 79)

5. Initialize Buffers: SHA1 requires 160 bits or 5 buffers of words (32 bits):

   H0 = 0x67452301   H1 = 0xEFCDAB89
   H2 = 0x98BADCFE   H3 = 0x10325476
   H4 = 0xC3D2E1F0

6. Processing Message in 512-bit blocks (L blocks in total message)

   i. This is the main task of SHA1 algorithm which loops through the padded and appended message in 512-bit blocks.

   ii. Input and predefined functions: M[1, 2, ..., L]: Blocks of the padded and appended message f(0;B,C,D), f(1,B,C,D), ..., f(79,B,C,D): 80 Processing Functions

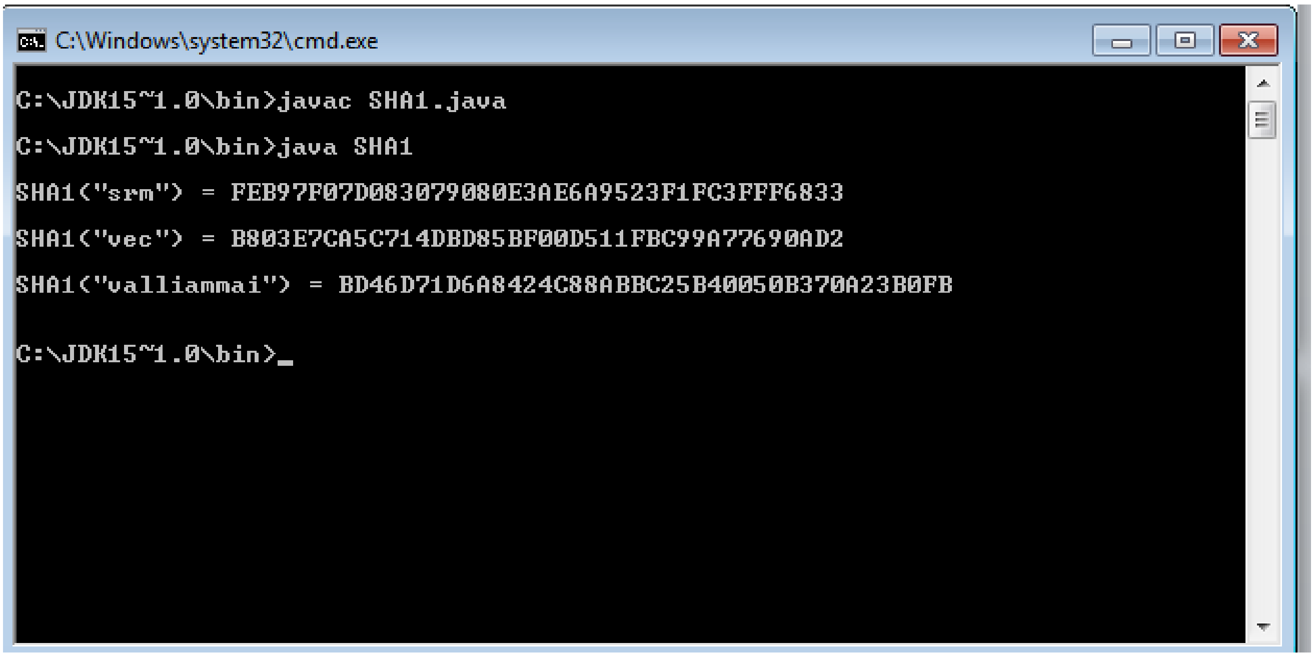
   K(0), K(1), ..., K(79): 80 Processing Constant Words
   H0, H1, H2, H3, H4, H5: 5 Word buffers with initial values

```
7. For loop on k = 1 to L
           1. (W(0),W(1),...,W(15)) = M[k] /* Divide M[k] into 16 words */
8. For t = 16 to 79 do:
       W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1
       A = H0, B = H1, C = H2, D = H3, E = H4
       For t = 0 to 79 do:
           TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t)
           E = D, D = C, C = B<<<30, B = A, A = TEMP
       End of for loop
       H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E
   End of for loop
```

PROGRAM

```
import java.security.*;
public class SHA1 {
public static void main(String[] a) {
try {
MessageDigest md = MessageDigest.getInstance("SHA1");
 String input = "srm";
md.update(input.getBytes());
byte[] output = md.digest();
System.out.println();
System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "vec";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("SHA1(\""+input+"\") = " +bytesToHex(output));
input = "valliammai";
md.update(input.getBytes());
output = md.digest();
System.out.println();
System.out.println("SHA1(\"" +input+"\") = " +bytesToHex(output));
System.out.println(""); }
catch (Exception e) {
System.out.println("Exception: " +e);
 }
 }
public static String bytesToHex(byte[] b) {
 char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
StringBuffer buf = new StringBuffer();
for (int j=0; j<b.length; j++) {
buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
buf.append(hexDigit[b[j] & 0x0f]); }
return buf.toString(); }
}
```

OUTPUT:



```
C:\JDK15~1.0\bin>javac SHA1.java

C:\JDK15~1.0\bin>java SHA1

SHA1("srm") = FEB97F07D083079080E3AE6A9523F1FC3FFF6833

SHA1("vec") = B803E7CA5C714DBD85BF00D511FBC99A77690AD2

SHA1("valliammai") = BD46D71D6A8424C88ABBC25B40050B370A23B0FB

C:\JDK15~1.0\bin>_
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. SHA-1 produces a hash value of how many bits?
2. What is the number of round computation steps in the SHA-256 algorithm?
3. In SHA-512, the message is divided into blocks size of how many bits for the hash computation.
4. What is the maximum length of the message (in bits) that can be taken by SHA-512?
5. What is the length of the message in SHA-512 after it ispadded?
6. Describe the big-endian format?
7. In SHA-512, the registers 'a' to 'h' are obtained by taking the first 64 bits of the fractional parts of the cube roots of the first 8 prime numbers. Is it true or false?
8. What is the size of W (in bits) in the SHA-512 processing of a single 1024- bit block?
9. In the SHA-512 processing of a single 1024- bit block, how the round constants are obtained?
10. What is the maximum length of the message (in bits) that can be taken by SHA-512?
11. What does the figure represent?
12. Among the registers 'a' to 'h' how many involve permutation in each round?

RESULT:

Thus the program to implement Secure Hash Algorithm was developed and executed successfully.

EX.No.: 8	IMPLEMENT DIGITAL SIGNATURE SCHEME

AIM:

To write a program to implement the digital signature scheme in java

PRELAB DISCUSSION:

Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm such as RSA, one can generate two keys that are mathematically linked: one private and one public. To create a digital signature, signing software (such as an email program) creates a one-way hash of the electronic data to be signed. The private key is then used to encrypt the hash. The encrypted hash -- along with other information, such as the hashing algorithm -- is the digital signature. The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter. This saves time since hashing is much faster than signing. The value of the hash is unique to the hashed data. Any change in the data, even changing or deleting a single character, results in a different value. This attribute enables others to validate the integrity of the data by using the signer's public key to decrypt the hash. If the decrypted hash matches a second computed hash of the same data, it proves that the data hasn't changed since it was signed. If the two hashes don't match, the data has either been tampered with in some way (integrity) or the signature was created with a private key that doesn't correspond to the public key presented by the signer (authentication).

A digital signature can be used with any kind of message -- whether it is encrypted or not -- simply so the receiver can be sure of the sender's identity and that the message arrived intact. Digital signatures make it difficult for the signer to deny having signed something (non-repudiation)
-- assuming their private key has not been compromised -- as the digital signature is unique to both the document and the signer, and it binds them together. A digital certificate, an electronic document that contains the digital signature of the certificate-issuing authority, binds together a public key with an identity and can be used to verify a public key belongs to a particular person or entity. Most modern email programs support the use of digital signatures and digital certificates, making it easy to sign any outgoing emails and validate digitally signed incoming messages. Digital signatures are also used extensively to provide proof of authenticity, data integrity and non- repudiation of communications and transactions conducted over the Internet.

ALGORITHM:

1. Choose a prime number q, which is called the prime divisor.
2. Choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus.
3. Choose an integer g, such that 1 < g < p, g**q mod p = 1 and g = h**((p–1)/q) mod p. q is also called g's multiplicative order modulo p.
4. Choose an integer, such that 0 < x < q.
5. Compute y as g**x mod p.
6. Package the public key as {p,q,g,y}, {p,q,g,x}.

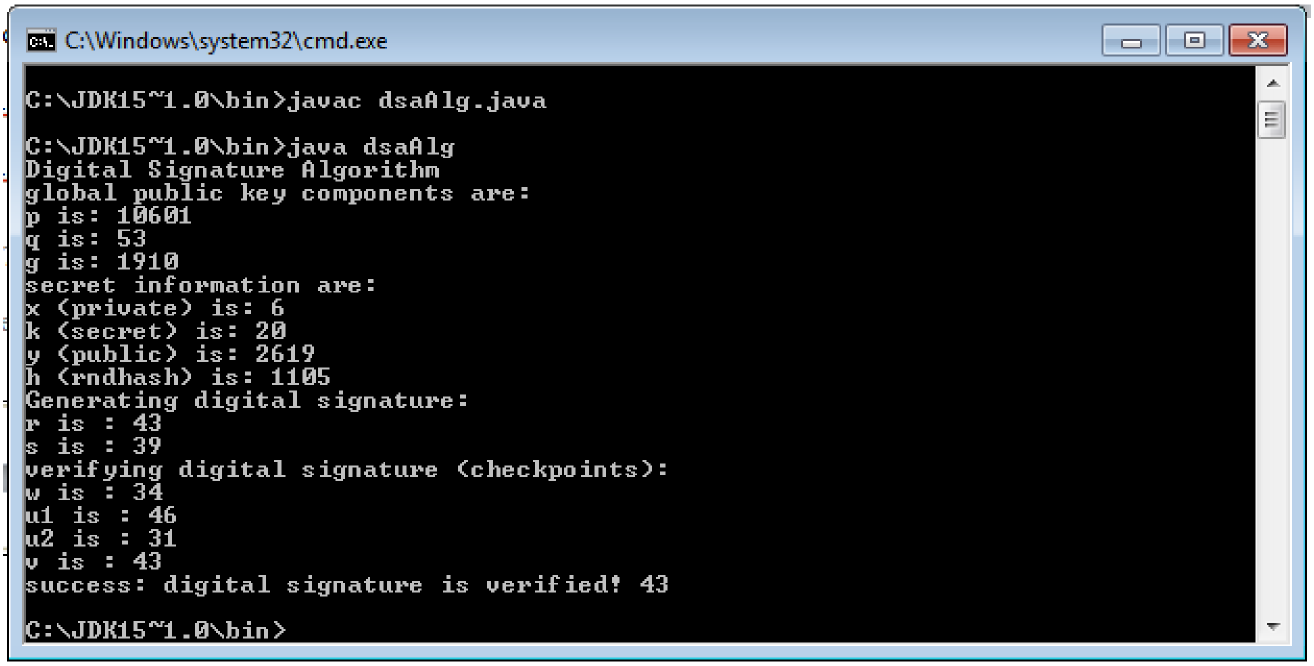
7. Generate the message digest h, using a hash algorithm like SHA1.
8. Generate a random number k, such that 0 < k < q.
9. Compute r as (g**k mod p) mod q. If r = 0, select a different k.
10. Compute i, such that k*i mod q = 1. i is called the modular multiplicative inverse of k modulo q.
11. Compute s = i*(h+r*x) mod q. If s = 0, select a different k.
12. Package the digital signature as {r,s}.
13. Generate the message digest h, using the same hash algorithm.
14. Compute w, such that s*w mod q = 1. w is called the modular multiplicative inverse of s modulo q.
15. Compute u1 = h*w mod q. Compute u2 = r*w mod q.
16. Compute v = (((g**u1)*(y**u2)) mod p) mod q.
17. If v == r, the digital signature is valid.

PROGRAM

```
import java.util.*;
import java.math.BigInteger;
class dsaAlg
{
        final static BigInteger one = new BigInteger("1");
        final static BigInteger zero = new BigInteger("0");

        /* incrementally tries for next prime */
        public static BigInteger getNextPrime(String ans)
        {
                BigInteger test = new BigInteger(ans);
                while (!test.isProbablePrime(99))
                {
                        test = test.add(one);
                }
                return test;
        }

        /* finds largest prime factor of n */
        public static BigInteger findQ(BigInteger n)
        {
                BigInteger start = new BigInteger("2");
                while (!n.isProbablePrime(99))
                {
                        while (!((n.mod(start)).equals(zero)))
                        {
                                start = start.add(one);
                        }
                        n = n.divide(start);
                }
                return n;
        }

        /* finds a generator mod p */
        public static BigInteger getGen(BigInteger p, BigInteger q, Random r)
        {
                BigInteger h = new BigInteger(p.bitLength(), r);
                h = h.mod(p);
                return h.modPow((p.subtract(one)).divide(q), p);
        }

        public static void main (String[] args) throws java.lang.Exception
        {
                Random randObj = new Random();

                /* establish the global public key components */
                BigInteger p = getNextPrime("10600"); /* approximate prime */
                BigInteger q = findQ(p.subtract(one));
                BigInteger g = getGen(p,q,randObj);

                /* public key components */
                System.out.println("Digital Signature Algorithm");
                System.out.println("global public key components are:");
                System.out.println("p is: " + p);
                System.out.println("q is: " + q);
                System.out.println("g is: " + g);

                /* find the private key */
                BigInteger x = new BigInteger(q.bitLength(), randObj);
                x = x.mod(q);

                /* corresponding public key */
                BigInteger y = g.modPow(x,p);

                /* random value message */
                BigInteger k = new BigInteger(q.bitLength(), randObj);
                k = k.mod(q);

                /* randomly generated hash value and digital signature */
                BigInteger r = (g.modPow(k,p)).mod(q);
                BigInteger hashVal = new BigInteger(p.bitLength(), randObj);
                BigInteger kInv = k.modInverse(q);
                BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
                s = s.mod(q);
```

```
                /*  secret  information  */
                System.out.println("secret information are:");
                System.out.println("x (private) is: " + x);
                System.out.println("k (secret) is: " + k);
                System.out.println("y (public) is: " + y);
                System.out.println("h (rndhash) is: "  + hashVal);
                System.out.println("Generating digital signature:");
                System.out.println("r is : " +  r);
                System.out.println("s is : " + s);

                /*verify the digital signature */
                BigInteger w = s.modInverse(q);
                BigInteger u1 = (hashVal.multiply(w)).mod(q);
                BigInteger u2 = (r.multiply(w)).mod(q);
                BigInteger  v  =  (g.modPow(u1,p)).multiply(y.modPow(u2,p));
                v = (v.mod(p)).mod(q);
                System.out.println("verifying digital signature (checkpoints):");
                System.out.println("w is : " + w);
                System.out.println("u1  is:  "  +  u1);
                System.out.println("u2  is:  "  +  u2);
                System.out.println("v is : " + v);
                if (v.equals(r))
                {
                        System.out.println("success:  digital  signature  is  verified!  "  +  r);
                }
                else
                {
                        System.out.println("error:  incorrect  digital  signature");
                }
        }
}
```

OUTPUT



```
C:\JDK15~1.0\bin>javac dsaAlg.java

C:\JDK15~1.0\bin>java dsaAlg
Digital Signature Algorithm
global public key components are:
p is: 10601
q is: 53
g is: 1910
secret information are:
x (private) is: 6
k (secret) is: 20
y (public) is: 2619
h (rndhash) is: 1105
Generating digital signature:
r is : 43
s is : 39
verifying digital signature (checkpoints):
w is : 34
u1 is : 46
u2 is : 31
v is : 43
success: digital signature is verified! 43

C:\JDK15~1.0\bin>
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. What is a digital signature?
2. What does a digital signature look like?
3. What is an electronic document?
4. Does that mean that the authenticity of any electronic document can be verified by a digital signature?
5. What is it like to actually sign an electronic document?
6. Can you actually see the signer's handwritten signature?
7. How do I get a digital signature certificate?
8. What is a certificate? What does it mean to "publish" a certificate?
9. How am I identified as the signer?
10. If my private key is stored on my computer, can't someone sign the documents without my permission by getting access to the computer?
11. Can a digital signature be forged?
12. What are the responsibilities and the liability of a digital signature certificate subscriber?
13. What are the practical uses of a digital signature?

RESULT:

Thus the program to implement Digital Signature was developed and executed successfully

EX.No.: 9	**DEMONSTRATE INTRUSION DETECTION SYSTEM (IDs) USING ANY TOOL (SNORT OR ANY OTHER S/W)**

AIM:

To demonstrate intrusion detection system (ids) using the tool snort

PRELAB DISCUSSION and PROCEDURE:

1. Configure and Use Snort IDS on Windows

Steps to configure Snort on Widnows machine and how to use it for detection of attacks.

2. Steps:

1. Download Snort from "http://www.snort.org/" website.
2. Also download Rules from the same website. You need to sign up to get rules for registered users.
3. Click on the Snort_(version-number)_Installer.exe file to install it. By-default it will install snort in the "C:\Snort" directory.
4. Extract downloaded Rules file: snortrules-snapshot-(number).tar.gz
5. Copy all files from the "rules" directory of the extracted folder and paste them into "C:\Snort\rules" directory.
6. Copy "snort.conf" file from the "etc" directory of the extracted folder and paste it into "C:\Snort\etc" directory. Overwrite existing file if there is any.
7. Open command prompt (cmd.exe) and navigate to directory "C:\Snort\bin" directory.
8. To execute snort in sniffer mode use following command:

```
snort -dev -i 2
```

-i indicate interface number.

-dev is used to run snort to capture packets.

To check interface list use following command: snort -W

9. To execute snort in IDS mode, we need to configure a file "snort.conf" according to our network environment.
10. Set up network address we want to protect in snort.conf file. To dothat look for "HOME_NET" and add your IP address.

```
var HOME_NET 10.1.1.17/8
```

11. You can also set addresses or DNS_SERVERS, if you have any. otherwise go to the next step.
12. Change RULE_PATH variable with the path of rules directory.

```
var RULE_PATH c:\snort\rules
```

13. Change the path of all libraries with the name and path on your system. or change path of snort_dynamicpreprocessorvariable.

```
sor file C:\Snort\lib\snort_dynamiccpreprocessor\sf_dcerpc.dll
```

You need to do this to all library files in the "C:\Snort\lib" directory. The old path might be something like: "/usr/local/lib/...". you need to replace that path with you system path.

14. Change path of the "dynamicengine" variable value in the "snort.conf" file with the path of your system. Such as:

```
dynamicengine  C:\Snort\lib\snort_dynamicengine\sf_engine.dll
```

15 Add complete path for "include classification.config" and "include reference.config" files.

```
include c:\snort\etc\classification.config
include c:\snort\etc\reference.config
```

16. Remove the comment on the line to allow **ICMP** rules, if it is alredy commented.

```
include $RULE_PATH/icmp.rules
```

17. Similary, remove the comment of ICMP-info rules comment, if it is already commented.

```
include $RULE_PATH/icmp-info.rules
```

18 To add log file to store alerts generated by snort, search for "output log" test and add following line:

```
output alert_fast: snort-alerts.ids
```

19. Comment whitelist $WHITE_LIST_PATH/white_list.rules and blacklist $BLACK_LIST_PATH/black_list.rules lines. Also ensure that you add change the line above $WHITE_LIST_PATH

Change nested_ip inner , \ to nested_ip inner #, \

20. Comment following lines:

```
#preprocessor  normalize_ip4
#preprocessor normalize_tcp: ips ecn stream
#preprocessor normalize_icmp4
#preprocessor normalize_ip6
#preprocessor  normalize_icmp6
```

21. Save the "snort.conf" file and close it.

22. Go to the "C:\Snort\log" directory and create a file: snort-alerts.ids

23. To start snort in IDS mode, run following command:

```
snort -c c:\snort\etc\snort.conf -l c:\snort\log -i 2
```

Above command will generate log file that will not be readable without using a tool. To read it use following command:

```
C:\Snort\Bin\> snort -r ..\log\log-filename
```

To generate Log files in ASCII mode use following command while running snort in IDS mode:

```
snort -A console -i2 -c c:\Snort\etc\snort.conf -l c:\Snort\log -K ascii
```

24. Scan the computer running snort from another computer using PING or launch attack. Then check snort-alerts.ids file the log folder.

VIVA QUESTIONS (PRE LAB and POSTLAB)

1. When discussing IDS/IPS, what is a signature?
2. "Semantics-aware" signatures automatically generated by Nemean are based on traffic at which two layers?
3. Which of the following is used to provide a baseline measure for comparison of IDSes?
4. What Is Ips And Ids?
5. What Are The Functions Of Intrusion Detection?
6. What Is Ids In Networking?
7. Explain Host Based (hids)?
8. What Is An Intrusion Detection System?
9. What is the advantage of anomaly detection?
10. Define a false positive.
11. One of the most obvious places to put an IDS sensor is near the firewall. Where exactly in relation to the firewall is the most productive placement?
12. What is the purpose of a shadow honeypot?
13. At which two traffic layers do most commercial IDSes generate signatures?
14. An IDS follows a two-step process consisting of a passive component and an active component.
15. Which of the following is part of the active component?

RESULT:

Thus the intrusion detection system (ids) using the tool snort program was demonstrated and verified successfully.

EX.No.: 10	AUTOMATED ATTACK AND PENETRATION TOOLS EXPLORING N-STALKER, A VULNERABILITY ASSESSMENT TOOL

AIM:

To explore automated and penetration tools on network (KF Sensor)

PRELAB DISCUSSION:

HONEYPOTS

When it comes to computer security, honeypots are all the rage. Honeypots can detect unauthorized activities that might never be picked up by a traditional intrusion detection system. Furthermore, since almost all access to a honeypot is unauthorized, nearly everything in a honeypot's logs is worth paying attention to. Honeypots can act as a decoy to keep hackers away from your production servers. At the same time though, a honeypot can be a little tricky to deploy. In this article, I will walk you through the process of deploying a honeypot.

INTRODUCTION

There are many different types of honeypot systems. Honeypots can be hardware appliances or they can be software based. Software based firewalls can reside on top of a variety of operating systems. For the most part though, honeypots fall into two basic categories; real and virtual.

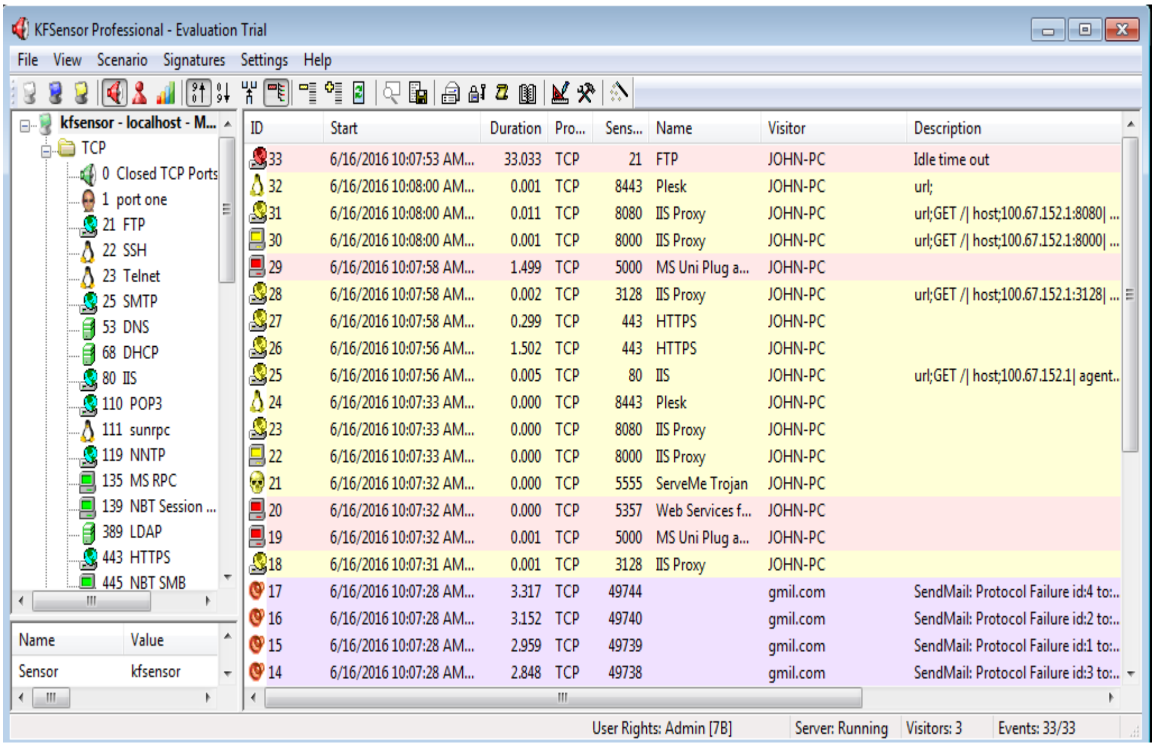
A virtual honeypot is essentially an emulated server. There are both hardware and software implementations of virtual honeypots. For example, if a network administrator was concerned that someone might try to exploit an FTP server, the administrator might deploy a honeypot appliance that emulates an FTP server.

Downloading and installing KF Sensor

- The KF Sensor download consists of a 1.7 MB self-extracting executable file.
- Download the file and copy it into an empty folder on your computer.
- When you double click on the file, it will launch a very basic Setup program.
- The only thing special that you need to know about the Setup process is that it will require a reboot

Using KFSensor

Step1: You will see the main KFSensor screen shown

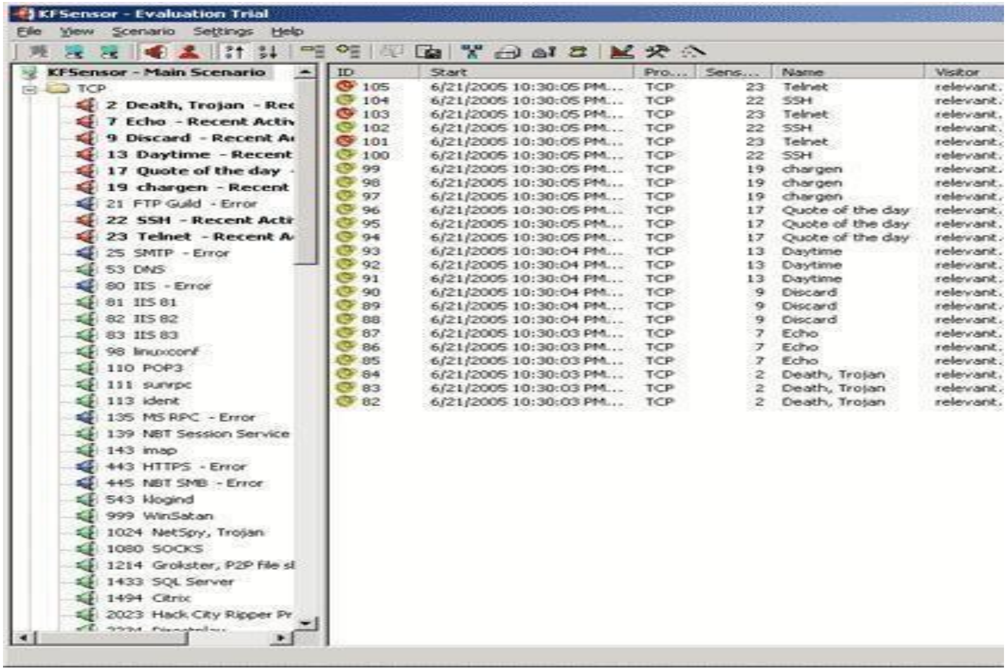


ID	Start	Duration	Pro...	Sens...	Name	Visitor	Description
33	6/16/2016 10:07:53 AM...	33.033	TCP	21	FTP	JOHN-PC	Idle time out
32	6/16/2016 10:08:00 AM...	0.001	TCP	8443	Plesk	JOHN-PC	url;
31	6/16/2016 10:08:00 AM...	0.011	TCP	8080	IIS Proxy	JOHN-PC	url;GET /| host;100.67.152.1:8080| ...
30	6/16/2016 10:08:00 AM...	0.001	TCP	8000	IIS Proxy	JOHN-PC	url;GET /| host;100.67.152.1:8000| ...
29	6/16/2016 10:07:58 AM...	1.499	TCP	5000	MS Uni Plug a...	JOHN-PC	
28	6/16/2016 10:07:58 AM...	0.002	TCP	3128	IIS Proxy	JOHN-PC	url;GET /| host;100.67.152.1:3128| ...
27	6/16/2016 10:07:58 AM...	0.299	TCP	443	HTTPS	JOHN-PC	
26	6/16/2016 10:07:56 AM...	1.502	TCP	443	HTTPS	JOHN-PC	
25	6/16/2016 10:07:56 AM...	0.005	TCP	80	IIS	JOHN-PC	url;GET /| host;100.67.152.1| agent..
24	6/16/2016 10:07:33 AM...	0.000	TCP	8443	Plesk	JOHN-PC	
23	6/16/2016 10:07:33 AM...	0.000	TCP	8080	IIS Proxy	JOHN-PC	
22	6/16/2016 10:07:33 AM...	0.000	TCP	8000	IIS Proxy	JOHN-PC	
21	6/16/2016 10:07:32 AM...	0.000	TCP	5555	ServeMe Trojan	JOHN-PC	
20	6/16/2016 10:07:32 AM...	0.000	TCP	5357	Web Services f...	JOHN-PC	
19	6/16/2016 10:07:32 AM...	0.001	TCP	5000	MS Uni Plug a...	JOHN-PC	
18	6/16/2016 10:07:31 AM...	0.001	TCP	3128	IIS Proxy	JOHN-PC	
17	6/16/2016 10:07:28 AM...	3.317	TCP	49744		gmil.com	SendMail: Protocol Failure id:4 to:..
16	6/16/2016 10:07:28 AM...	3.152	TCP	49740		gmil.com	SendMail: Protocol Failure id:2 to:..
15	6/16/2016 10:07:28 AM...	2.959	TCP	49739		gmil.com	SendMail: Protocol Failure id:1 to:..
14	6/16/2016 10:07:28 AM...	2.848	TCP	49738		gmil.com	SendMail: Protocol Failure id:3 to:..

- ➢ As you can see, the column on the left contains a list of port numbers and what the port is typically used for.
- ➢ If the icon to the left of a port listing is green, it means that KFSensor is actively monitoring that port for attacks.
- ➢ If the icon is blue, it means that there has been an error and KFSensor is not watching for exploits aimed at that particular port.

Testing the software

- ➢ Once you've got the software up and running, one of the best things that you can do is to test the software by launching a port scan against the machine that's running KFSensor.
  - ➢ For the port scan, we using the HostScan.
- ➢ It simply scans a block of IP addresses, looking for open ports. Figure B shows how the KFSensor reacts to a partial port scan.
- ➢ If you look at Figure B, you will notice that the icons next to ports that were scanned turn red to indicate recent activity.

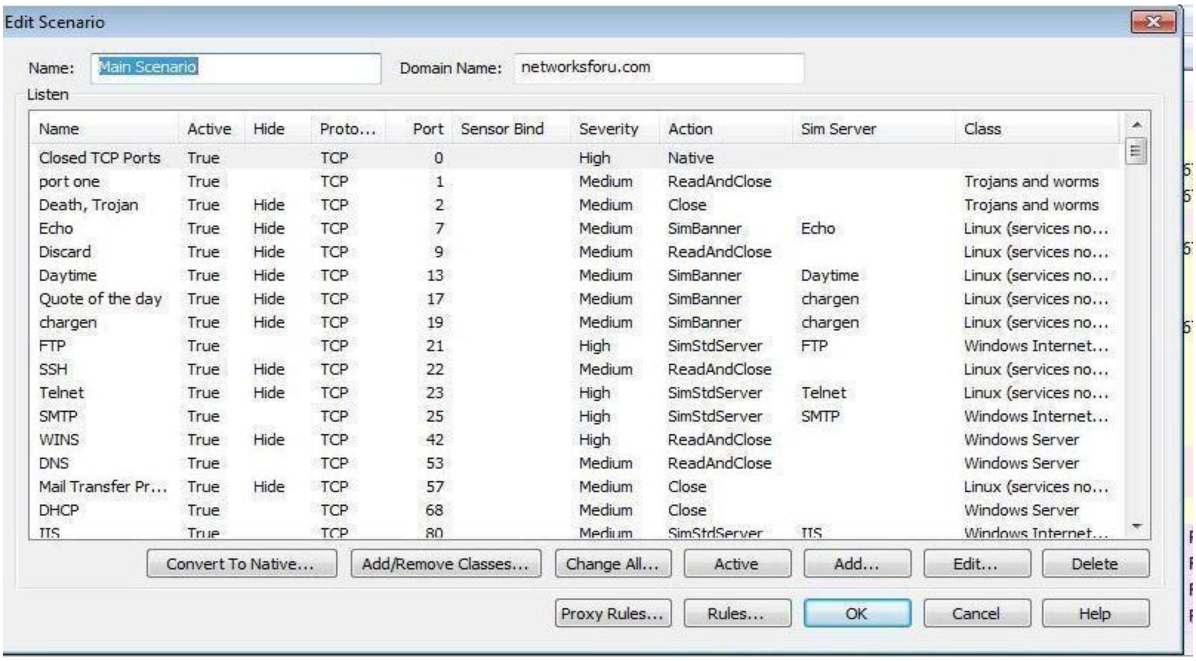


KFSensor - Main Scenario
- TCP
  - 2 Death, Trojan - Rec
  - 7 Echo - Recent Activ
  - 9 Discard - Recent Ac
  - 13 Daytime - Recent
  - 17 Quote of the day -
  - 19 chargen - Recent
  - 21 FTP Guild - Error
  - 22 SSH - Recent Acti
  - 23 Telnet - Recent A
  - 25 SMTP - Error
  - 53 DNS
  - 80 IIS - Error
  - 81 IIS 81
  - 82 IIS 82
  - 83 IIS 83
  - 98 linuxconf
  - 110 POP3
  - 111 sunrpc
  - 113 ident
  - 135 MS RPC - Error
  - 139 NBT Session Service
  - 143 imap
  - 443 HTTPS - Error
  - 445 NBT SMB - Error
  - 543 klogind
  - 999 WinSatan
  - 1024 NetSpy, Trojan
  - 1080 SOCKS
  - 1214 Grokster, P2P file sl
  - 1433 SQL Server
  - 1494 Citrix
  - 2023 Hack City Ripper Pr

ID	Start	Pro...	Sens...	Name	Visitor
105	6/21/2005 10:30:05 PM...	TCP	23	Telnet	relevant.
104	6/21/2005 10:30:05 PM...	TCP	22	SSH	relevant.
103	6/21/2005 10:30:05 PM...	TCP	23	Telnet	relevant.
102	6/21/2005 10:30:05 PM...	TCP	22	SSH	relevant.
101	6/21/2005 10:30:05 PM...	TCP	23	Telnet	relevant.
100	6/21/2005 10:30:05 PM...	TCP	22	SSH	relevant.
99	6/21/2005 10:30:05 PM...	TCP	19	chargen	relevant.
98	6/21/2005 10:30:05 PM...	TCP	19	chargen	relevant.
97	6/21/2005 10:30:05 PM...	TCP	19	chargen	relevant.
96	6/21/2005 10:30:05 PM...	TCP	17	Quote of the day	relevant.
95	6/21/2005 10:30:05 PM...	TCP	17	Quote of the day	relevant.
94	6/21/2005 10:30:05 PM...	TCP	17	Quote of the day	relevant.
93	6/21/2005 10:30:04 PM...	TCP	13	Daytime	relevant.
92	6/21/2005 10:30:04 PM...	TCP	13	Daytime	relevant.
91	6/21/2005 10:30:04 PM...	TCP	13	Daytime	relevant.
90	6/21/2005 10:30:04 PM...	TCP	9	Discard	relevant.
89	6/21/2005 10:30:04 PM...	TCP	9	Discard	relevant.
88	6/21/2005 10:30:04 PM...	TCP	9	Discard	relevant.
87	6/21/2005 10:30:03 PM...	TCP	7	Echo	relevant.
86	6/21/2005 10:30:03 PM...	TCP	7	Echo	relevant.
85	6/21/2005 10:30:03 PM...	TCP	7	Echo	relevant.
84	6/21/2005 10:30:03 PM...	TCP	2	Death, Trojan	relevant.
83	6/21/2005 10:30:03 PM...	TCP	2	Death, Trojan	relevant.
82	6/21/2005 10:30:03 PM...	TCP	2	Death, Trojan	relevant.

Modifying the Honeypot's behavior

- To create or modify rules, select the Edit Active Scenario command from the scenario menu.
- When you do, you will see a dialog box which contains a summary of all of the existing rules.
- You can either select a rule and click the Edit button to edit a rule, or you can click the Add button to create a new rule.
- Both procedures work similarly.



Name: Main Scenario

Domain Name: networksforu.com

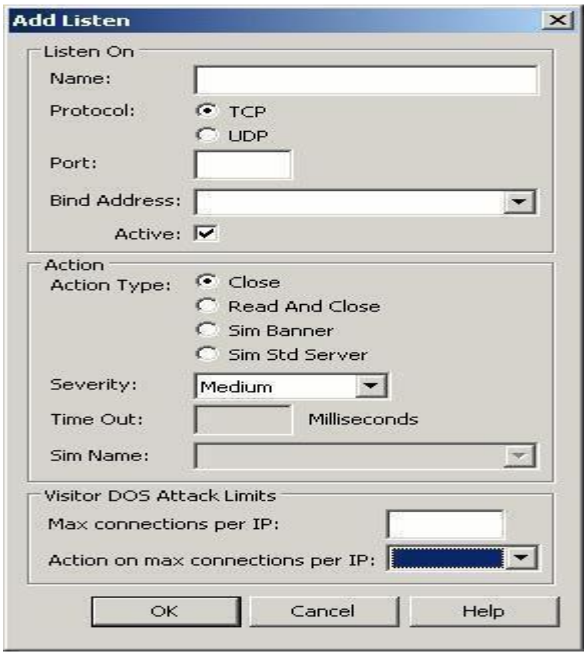
Listen

Name	Active	Hide	Proto...	Port	Sensor Bind	Severity	Action	Sim Server	Class
Closed TCP Ports	True		TCP	0		High	Native		
port one	True		TCP	1		Medium	ReadAndClose		Trojans and worms
Death, Trojan	True	Hide	TCP	2		Medium	Close		Trojans and worms
Echo	True	Hide	TCP	7		Medium	SimBanner	Echo	Linux (services no...
Discard	True	Hide	TCP	9		Medium	ReadAndClose		Linux (services no...
Daytime	True	Hide	TCP	13		Medium	SimBanner	Daytime	Linux (services no...
Quote of the day	True	Hide	TCP	17		Medium	SimBanner	chargen	Linux (services no...
chargen	True	Hide	TCP	19		Medium	SimBanner	chargen	Linux (services no...
FTP	True		TCP	21		High	SimStdServer	FTP	Windows Internet...
SSH	True	Hide	TCP	22		Medium	ReadAndClose		Linux (services no...
Telnet	True	Hide	TCP	23		High	SimStdServer	Telnet	Linux (services no...
SMTP	True		TCP	25		High	SimStdServer	SMTP	Windows Internet...
WINS	True	Hide	TCP	42		High	ReadAndClose		Windows Server
DNS	True		TCP	53		Medium	ReadAndClose		Windows Server
Mail Transfer Pr...	True	Hide	TCP	57		Medium	Close		Linux (services no...
DHCP	True		TCP	68		Medium	Close		Windows Server
IIS	True		TCP	80		Medium	SimStdServer	IIS	Windows Internet...

Click the Add button and you will see the Add Listen dialog box, shown in Figure D.

- The first thing that this dialog box asks for is a name. This is just a name for the rule.
- Pick something descriptive though, because the name that you enter is what will show up in the logs whenever the rule is triggered.

Click on Add Button



Add Listen

Listen On

Name:

Protocol: ● TCP ○ UDP

Port:

Bind Address:

Active: ☑

Action

Action Type: ● Close ○ Read And Close ○ Sim Banner ○ Sim Std Server

Severity: Medium

Time Out: Milliseconds

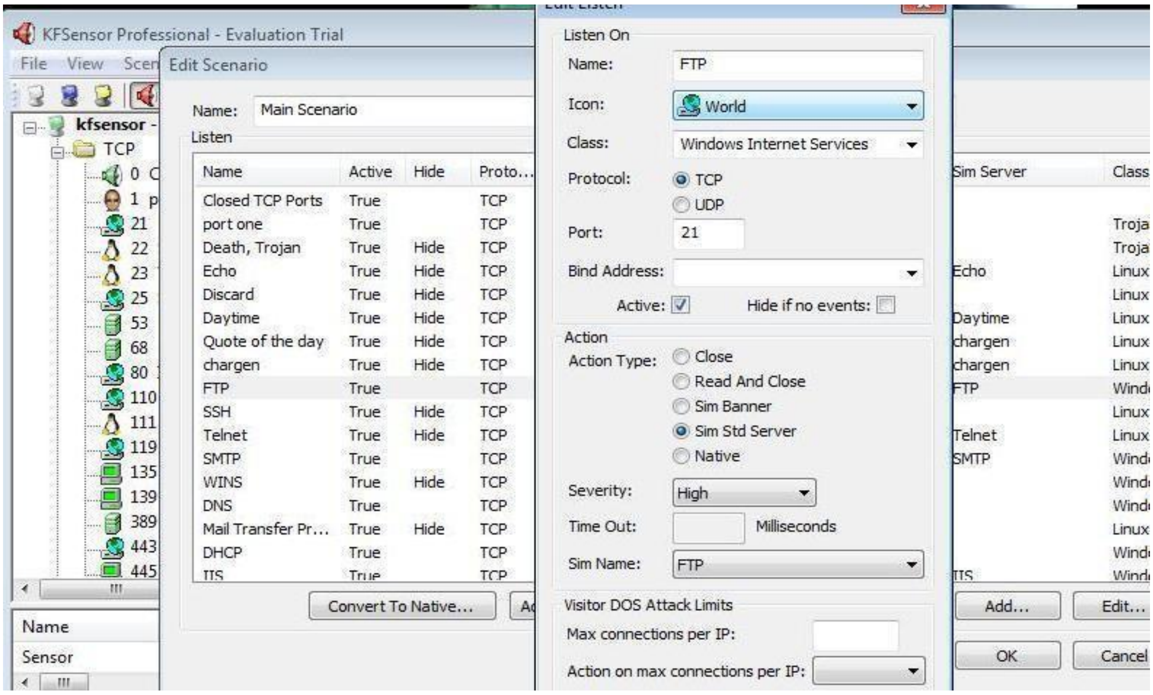
Sim Name:

Visitor DOS Attack Limits

Max connections per IP:

Action on max connections per IP:

Click on Edit Button



KFSensor Professional - Evaluation Trial

Edit Scenario

Name: Main Scenario

Listen

Name	Active	Hide	Proto...	Sim Server	Class
Closed TCP Ports	True		TCP		
port one	True		TCP		Troja
Death, Trojan	True	Hide	TCP		Troja
Echo	True	Hide	TCP	Echo	Linux
Discard	True	Hide	TCP		Linux
Daytime	True	Hide	TCP	Daytime	Linux
Quote of the day	True	Hide	TCP	chargen	Linux
chargen	True	Hide	TCP	chargen	Linux
FTP	True		TCP	FTP	Wind
SSH	True	Hide	TCP		Linux
Telnet	True	Hide	TCP	Telnet	Linux
SMTP	True		TCP	SMTP	Wind
WINS	True	Hide	TCP		Wind
DNS	True		TCP		Wind
Mail Transfer Pr...	True	Hide	TCP		Linux
DHCP	True		TCP		Wind
IIS	True		TCP	IIS	Wind

Edit Listen

Listen On

Name: FTP

Icon: World

Class: Windows Internet Services

Protocol: ● TCP ○ UDP

Port: 21

Bind Address:

Active: ☑ Hide if no events: ☐

Action

Action Type: ○ Close ○ Read And Close ○ Sim Banner ● Sim Std Server ○ Native

Severity: High

Time Out: Milliseconds

Sim Name: FTP

Visitor DOS Attack Limits

Max connections per IP:

Action on max connections per IP:

- The next few fields are protocol, port, and Bind Address. These fields allow you to choose what the rule is listening for. For example, you could configure the rule to listen to TCP port 1023 on IP address 192.168.1.100. The bind address portion of the rule is optional though. If you leave the bind address blank, the rule will listen across all of the machine's NICs.

- Now that you have defined the listener, it's time to configure the action that the rule takes when traffic is detected on the specified port. Your options are close, read and close, Sim Banner, and SimStd Server.

- The close option tells the rule to just terminate the connection. Read and close logs the information and then terminates the connection. The SimStd Server and Sim Banner options pertain to server emulation. The Sim Banner option allows you to perform a very simple server emulation, such as what you might use to emulate an FTP server.

- The Sim STD Server option allows you to emulate a more complex server, such as an IIS server.
- If you choose to use one of the sim options, you will have to fill in the simulator's name just below the Time Out field.
- The other part of the Action section that's worth mentioning is the severity section. KFSensor treated some events as severe and other events as a more moderate threat. The dialog box's Severity drop down list allows you to determine what level of severity should be associated with the event that you are logging.
- The final portion of the Add Listen dialog box is the Visitor DOS Attack Limits section. This section allows you to prevent denial of service attacks against KFSensor. You can determine the maximum number of connections to the machine per IP address (remember that this applies on a per rule basis).
- If your threshold is exceeded, you can choose to either ignore the excessive connections or you can lock out the offending IP address.
- Now that you have configured the new rule, select the Active Button to Enable/Disable. The new rule should now be in effect.

VIVA QUESTIONS (PRE LAB and POSTLAB):

1. How to build and use a Honeypot
2. List the types of interactions
3. Define honey tokens
4. Give the advantages of honey pot
5. Mention the protocols used by IPSec to provide security.
6. How arethe passwords stored in password file in UNIX operating system?
7. Which system is used to protect credit card transactions on the internet?
8. What is meant by Block Cipher?
9. Define Plain Text.

RESULT:

Thus the program to explore automated attack and penetration tools has been completed successfully

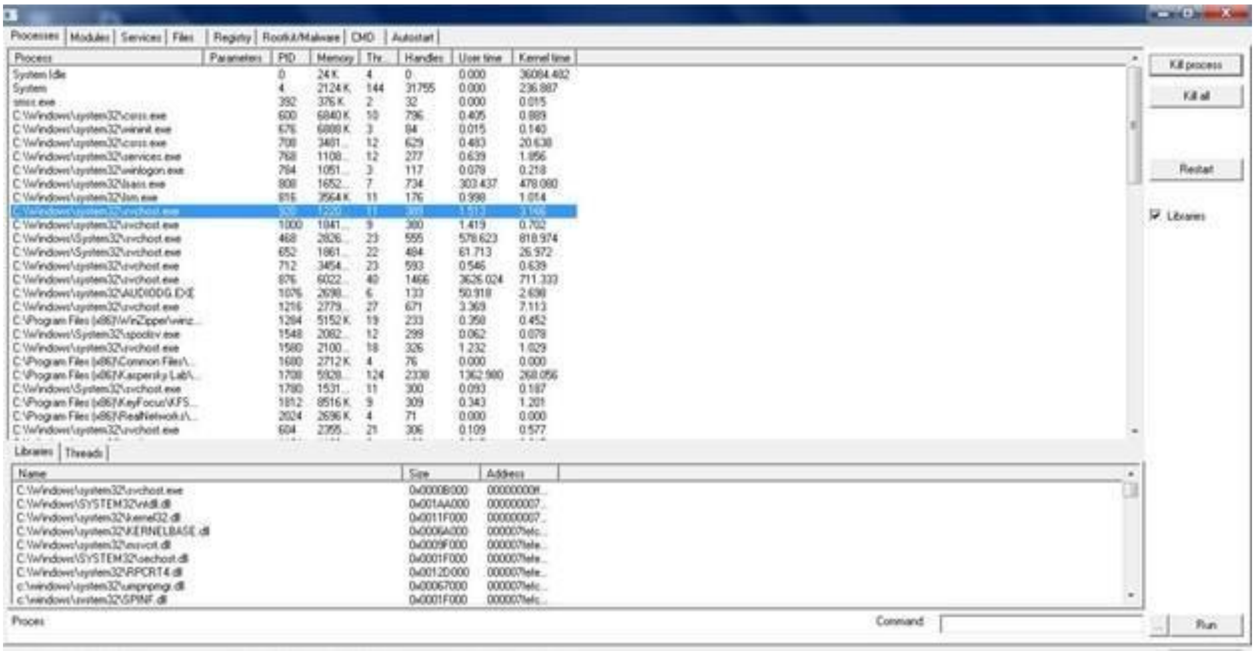
EX.No.: 11	DEFEATING MALWARE – ROOTKIT HUNTER

AIM

Root kit is a stealth type of malicious software designed to hide the existence of certain process from normal methods of detection and enables continued privileged access to a computer.

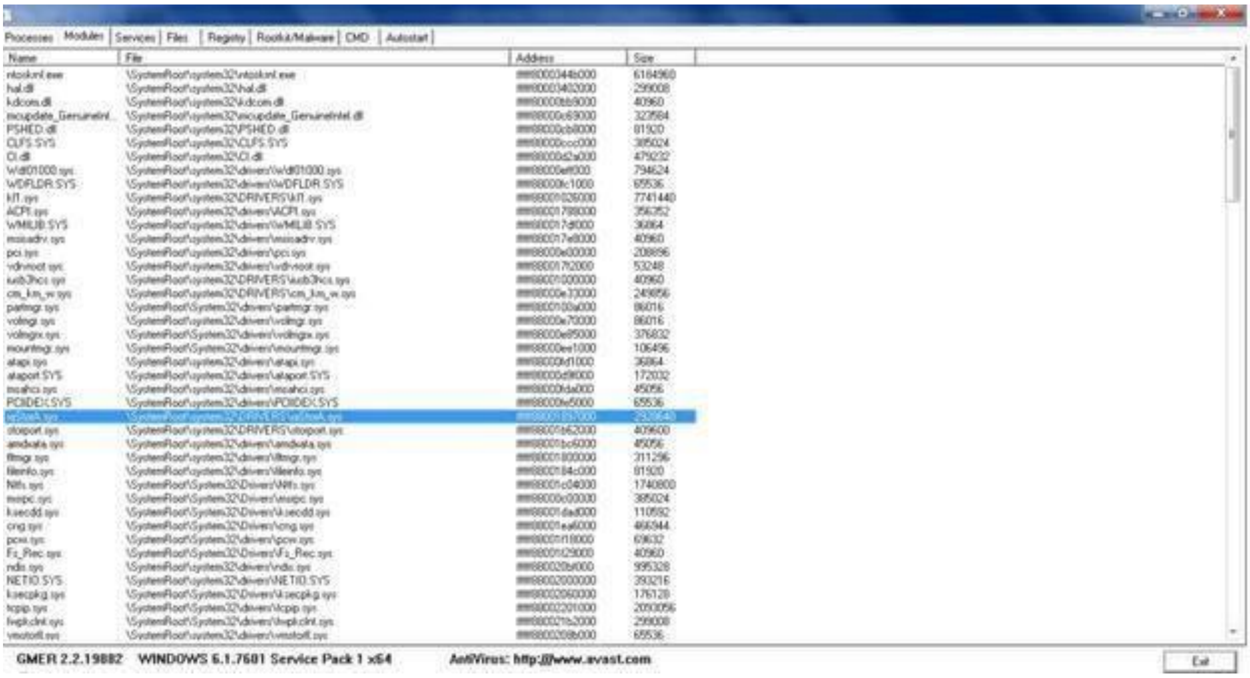
PRELAB DISCUSSION and PROCEDURE :

- Download Rootkit Tool from GMER website. www.gmer.net
- This displays the Processes, Modules, Services, Files, Registry, RootKit/Malwares, Autostart, CMD of local host.
- Select Processes menu and kill any unwanted process if any. Modules menu displays the various system files like .sys, .dll
- Services menu displays the complete services running with Autostart, Enable, Disable, System, Boot.
- Files menu displays full files on Hard-Disk volumes.
- Registry displays **Hkey_Current_user** and **Hkey_Local_Machine**. Rootkits/Malawares scans the local drives selected.
- **Autostart** displays the registry base Autostart applications.
- CMD allows the user to interact with command line utilities or Registry.



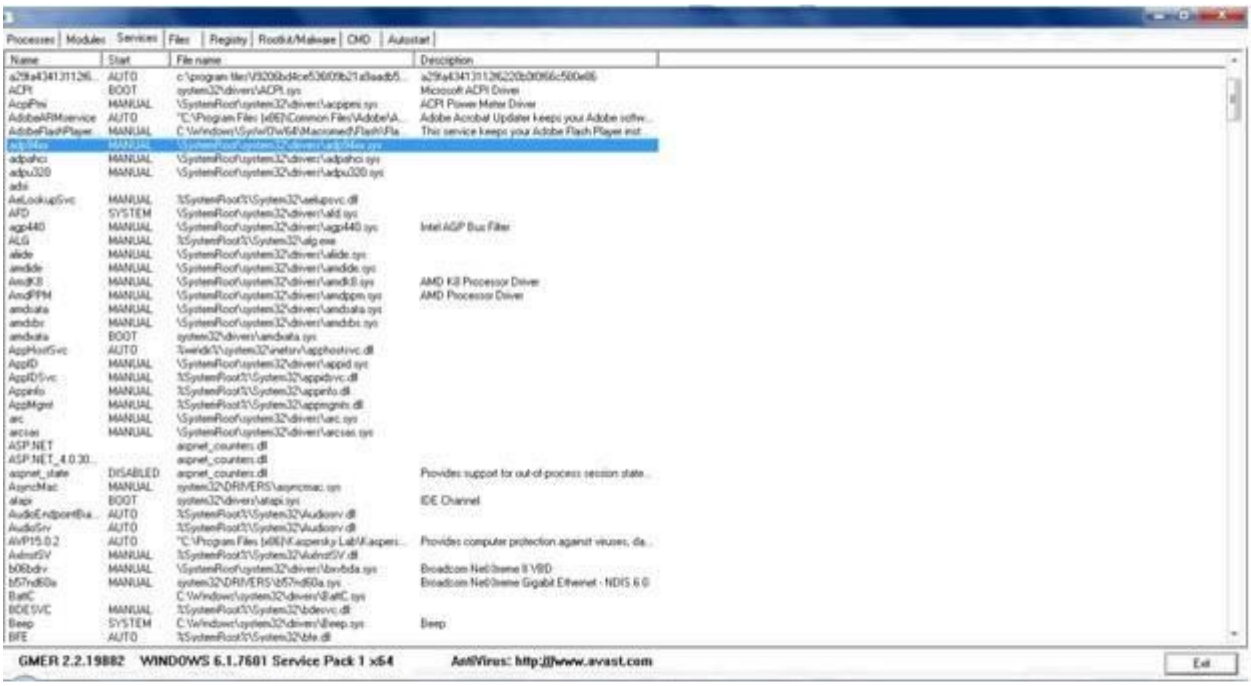
Process	Parameters	PID	Memory	Thr...	Handles	User time	Kernel time
System Idle		0	24 K	4	0	0.000	36084.402
System		4	2124 K	144	31755	0.000	236.887
smss.exe		392	376 K	2	32	0.000	0.015
C:\Windows\system32\csrss.exe		600	6840 K	10	796	0.405	0.889
C:\Windows\system32\wininit.exe		676	6808 K	3	84	0.015	0.140
C:\Windows\system32\csrss.exe		708	3481...	12	629	0.483	20.638
C:\Windows\system32\services.exe		768	1108...	12	277	0.639	1.856
C:\Windows\system32\winlogon.exe		784	1051...	3	117	0.078	0.218
C:\Windows\system32\lsass.exe		808	1652...	7	734	303.437	478.080
C:\Windows\system32\lsm.exe		816	3564 K	11	176	0.998	1.014
C:\Windows\system32\svchost.exe		920	1220...	11	389	3.513	3.166
C:\Windows\system32\svchost.exe		1000	1041...	9	380	1.419	0.702
C:\Windows\System32\svchost.exe		468	2826...	23	555	578.623	818.974
C:\Windows\System32\svchost.exe		652	1861...	22	494	61.713	26.972
C:\Windows\system32\svchost.exe		712	3454...	23	593	0.546	0.639
C:\Windows\system32\svchost.exe		876	6022...	40	1466	3626.024	711.333
C:\Windows\system32\AUDIODG.EXE		1076	2698...	6	133	50.918	2.698
C:\Windows\system32\svchost.exe		1216	2779...	27	671	3.369	7.113
C:\Program Files (x86)\WinZipper\winz...		1284	5152 K	19	233	0.358	0.452
C:\Windows\System32\spoolsv.exe		1548	2082...	12	299	0.062	0.078
C:\Windows\system32\svchost.exe		1580	2100...	18	326	1.232	1.029
C:\Program Files (x86)\Common Files\...		1600	2712 K	4	76	0.000	0.000
C:\Program Files (x86)\Kaspersky Lab\...		1708	5928...	124	2338	1362.980	268.056
C:\Windows\System32\svchost.exe		1780	1531...	11	300	0.093	0.187
C:\Program Files (x86)\KeyFocus\KFS...		1812	8516 K	9	309	0.343	1.201
C:\Program Files (x86)\RealNetworks\...		2024	2696 K	4	71	0.000	0.000
C:\Windows\system32\svchost.exe		604	2355...	21	306	0.109	0.577

Name	Size	Address
C:\Windows\system32\svchost.exe	0x0000B000	00000000ff...
C:\Windows\SYSTEM32\ntdll.dll	0x001AA000	000000007...
C:\Windows\system32\kernel32.dll	0x0011F000	000000007...
C:\Windows\system32\KERNELBASE.dll	0x0006A000	000007fefc...
C:\Windows\system32\msvcrt.dll	0x0009F000	000007fefe...
C:\Windows\SYSTEM32\sechost.dll	0x0001F000	000007fefe...
C:\Windows\system32\RPCRT4.dll	0x0012D000	000007fefe...
c:\windows\system32\umpnpmgr.dll	0x00067000	000007fefc...
c:\windows\system32\SPINF.dll	0x0001F000	000007fefc...



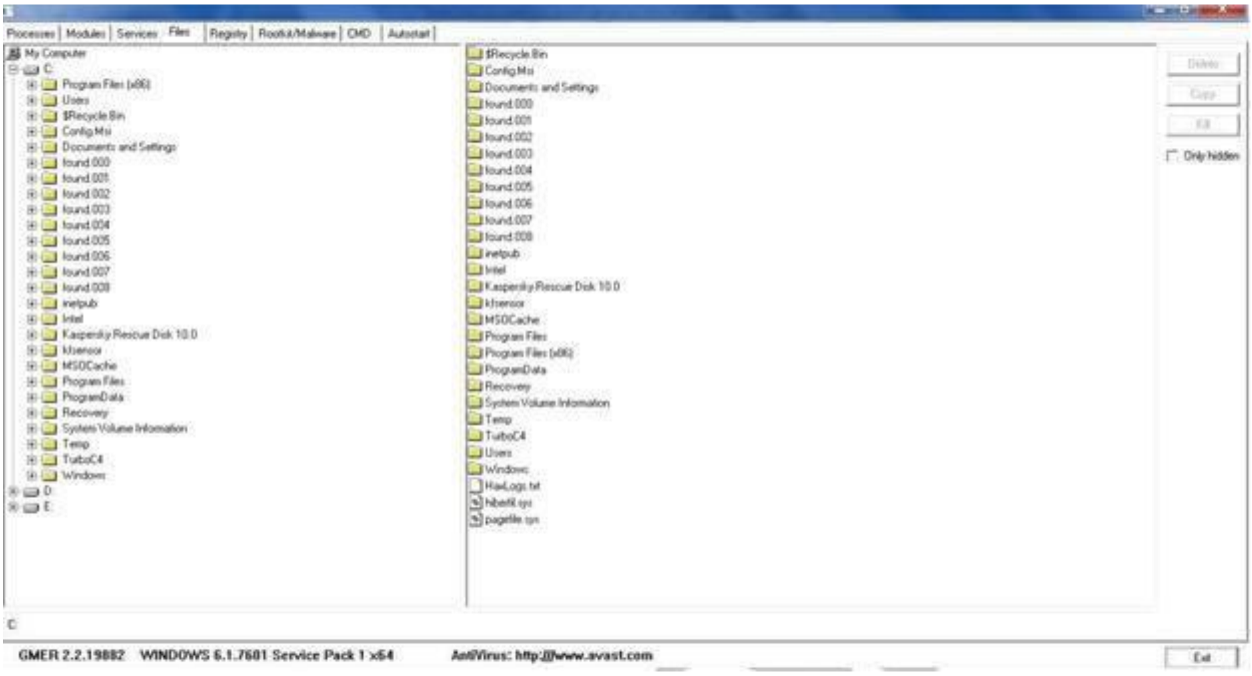
Name	File	Address	Size
ntoskrnl.exe	\SystemRoot\system32\ntoskrnl.exe	fffff8000344b000	6184960
hal.dll	\SystemRoot\system32\hal.dll	fffff80003402000	299008
kdcom.dll	\SystemRoot\system32\kdcom.dll	fffff80000bb9000	40960
mcupdate_GenuineInt...	\SystemRoot\system32\mcupdate_GenuineIntel.dll	fffff88000c69000	323584
PSHED.dll	\SystemRoot\system32\PSHED.dll	fffff88000cb8000	81920
CLFS.SYS	\SystemRoot\system32\CLFS.SYS	fffff88000ccc000	385024
CI.dll	\SystemRoot\system32\CI.dll	fffff88000d2a000	479232
Wdf01000.sys	\SystemRoot\system32\drivers\Wdf01000.sys	fffff88000eff000	794624
WDFLDR.SYS	\SystemRoot\system32\drivers\WDFLDR.SYS	fffff88000fc1000	65536
kl1.sys	\SystemRoot\system32\DRIVERS\kl1.sys	fffff88001026000	7741440
ACPI.sys	\SystemRoot\system32\drivers\ACPI.sys	fffff88001798000	356352
WMILIB.SYS	\SystemRoot\system32\drivers\WMILIB.SYS	fffff880017ef000	36864
msisadrv.sys	\SystemRoot\system32\drivers\msisadrv.sys	fffff880017f8000	40960
pci.sys	\SystemRoot\system32\drivers\pci.sys	fffff88000e00000	208896
vdrvroot.sys	\SystemRoot\system32\drivers\vdrvroot.sys	fffff88001712000	53248
iusb3hcs.sys	\SystemRoot\system32\DRIVERS\iusb3hcs.sys	fffff88001000000	40960
cm_km_w.sys	\SystemRoot\system32\DRIVERS\cm_km_w.sys	fffff88000e33000	249856
partmgr.sys	\SystemRoot\System32\drivers\partmgr.sys	fffff8800100a000	86016
volmgr.sys	\SystemRoot\system32\drivers\volmgr.sys	fffff88000e70000	86016
volmgrx.sys	\SystemRoot\System32\drivers\volmgrx.sys	fffff88000e85000	376832
mountmgr.sys	\SystemRoot\System32\drivers\mountmgr.sys	fffff88000ee1000	106496
atapi.sys	\SystemRoot\system32\drivers\atapi.sys	fffff88000fd1000	36864
ataport.SYS	\SystemRoot\system32\drivers\ataport.SYS	fffff88000d9f000	172032
msahci.sys	\SystemRoot\system32\drivers\msahci.sys	fffff88000fda000	45056
PCIIDEX.SYS	\SystemRoot\system32\drivers\PCIIDEX.SYS	fffff88000fe5000	65536
iaStorA.sys	\SystemRoot\system32\DRIVERS\iaStorA.sys	fffff88001897000	2928640
storport.sys	\SystemRoot\system32\DRIVERS\storport.sys	fffff88001b62000	409600
amdxata.sys	\SystemRoot\system32\drivers\amdxata.sys	fffff88001bc6000	45056
fltmgr.sys	\SystemRoot\system32\drivers\fltmgr.sys	fffff88001800000	311296
fileinfo.sys	\SystemRoot\system32\drivers\fileinfo.sys	fffff8800184c000	81920
Ntfs.sys	\SystemRoot\System32\Drivers\Ntfs.sys	fffff88001c04000	1740800
msrpc.sys	\SystemRoot\System32\Drivers\msrpc.sys	fffff88000c00000	385024
ksecdd.sys	\SystemRoot\System32\Drivers\ksecdd.sys	fffff88001dad000	110592
cng.sys	\SystemRoot\System32\Drivers\cng.sys	fffff88001ea6000	466944
pcw.sys	\SystemRoot\System32\drivers\pcw.sys	fffff88001f18000	69632
Fs_Rec.sys	\SystemRoot\System32\Drivers\Fs_Rec.sys	fffff88001f29000	40960
ndis.sys	\SystemRoot\system32\drivers\ndis.sys	fffff88001f0d000	995328
NETIO.SYS	\SystemRoot\system32\drivers\NETIO.SYS	fffff88002000000	393216
ksecpkg.sys	\SystemRoot\System32\Drivers\ksecpkg.sys	fffff88002060000	176128
tcpip.sys	\SystemRoot\System32\drivers\tcpip.sys	fffff88002201000	2093056
fwpkclnt.sys	\SystemRoot\System32\drivers\fwpkclnt.sys	fffff880021b2000	299008
vmstorfl.sys	\SystemRoot\system32\drivers\vmstorfl.sys	fffff8800208b000	65536

GMER 2.2.19882 WINDOWS 6.1.7601 Service Pack 1 x64 AntiVirus: http://www.avast.com



Name	Start	File name	Description
a29a4341311126...	AUTO	c:\program files\V9206bd4ce536f09b21a9aadb5...	a29a434131126220b0f066c580e86
ACPI	BOOT	system32\drivers\ACPI.sys	Microsoft ACPI Driver
AcpiPmi	MANUAL	\SystemRoot\system32\drivers\acpipmi.sys	ACPI Power Meter Driver
AdobeARMservice	AUTO	"C:\Program Files (x86)\Common Files\Adobe\A...	Adobe Acrobat Updater keeps your Adobe softw...
AdobeFlashPlayer...	MANUAL	C:\Windows\SysWOW64\Macromed\Flash\Fla...	This service keeps your Adobe Flash Player inst...
adp94xx	MANUAL	\SystemRoot\system32\drivers\adp94xx.sys	
adpahci	MANUAL	\SystemRoot\system32\drivers\adpahci.sys	
adpu320	MANUAL	\SystemRoot\system32\drivers\adpu320.sys	
adsi			
AeLookupSvc	MANUAL	%SystemRoot%\System32\aelupsvc.dll	
AFD	SYSTEM	\SystemRoot\system32\drivers\afd.sys	
agp440	MANUAL	\SystemRoot\system32\drivers\agp440.sys	Intel AGP Bus Filter
ALG	MANUAL	%SystemRoot%\System32\alg.exe	
aliide	MANUAL	\SystemRoot\system32\drivers\aliide.sys	
amdide	MANUAL	\SystemRoot\system32\drivers\amdide.sys	
AmdK8	MANUAL	\SystemRoot\system32\drivers\amdk8.sys	AMD K8 Processor Driver
AmdPPM	MANUAL	\SystemRoot\system32\drivers\amdppm.sys	AMD Processor Driver
amdsata	MANUAL	\SystemRoot\system32\drivers\amdsata.sys	
amdsbs	MANUAL	\SystemRoot\system32\drivers\amdsbs.sys	
amdxata	BOOT	system32\drivers\amdxata.sys	
AppHostSvc	AUTO	%windir%\system32\inetsrv\apphostsvc.dll	
AppID	MANUAL	\SystemRoot\system32\drivers\appid.sys	
AppIDSvc	MANUAL	%SystemRoot%\System32\appidsvc.dll	
Appinfo	MANUAL	%SystemRoot%\System32\appinfo.dll	
AppMgmt	MANUAL	%SystemRoot%\System32\appmgmts.dll	
arc	MANUAL	\SystemRoot\system32\drivers\arc.sys	
arcsas	MANUAL	\SystemRoot\system32\drivers\arcsas.sys	
ASP.NET		aspnet_counters.dll	
ASP.NET_4.0.30...		aspnet_counters.dll	
aspnet_state	DISABLED	aspnet_counters.dll	Provides support for out-of-process session state...
AsyncMac	MANUAL	system32\DRIVERS\asyncmac.sys	
atapi	BOOT	system32\drivers\atapi.sys	IDE Channel
AudioEndpointBu...	AUTO	%SystemRoot%\System32\Audiosrv.dll	
AudioSrv	AUTO	%SystemRoot%\System32\Audiosrv.dll	
AVP15.0.2	AUTO	"C:\Program Files (x86)\Kaspersky Lab\Kaspers...	Provides computer protection against viruses, da...
AxInstSV	MANUAL	%SystemRoot%\System32\AxInstSV.dll	
b06bdrv	MANUAL	\SystemRoot\system32\drivers\bxvbda.sys	Broadcom NetXtreme II VBD
b57nd60a	MANUAL	system32\DRIVERS\b57nd60a.sys	Broadcom NetXtreme Gigabit Ethernet - NDIS 6.0
BattC		C:\Windows\system32\drivers\BattC.sys	
BDESVC	MANUAL	%SystemRoot%\System32\bdesvc.dll	
Beep	SYSTEM	C:\Windows\system32\drivers\Beep.sys	Beep
BFE	AUTO	%SystemRoot%\System32\bfe.dll	

GMER 2.2.19882 WINDOWS 6.1.7601 Service Pack 1 x64 AntiVirus: http://www.avast.com

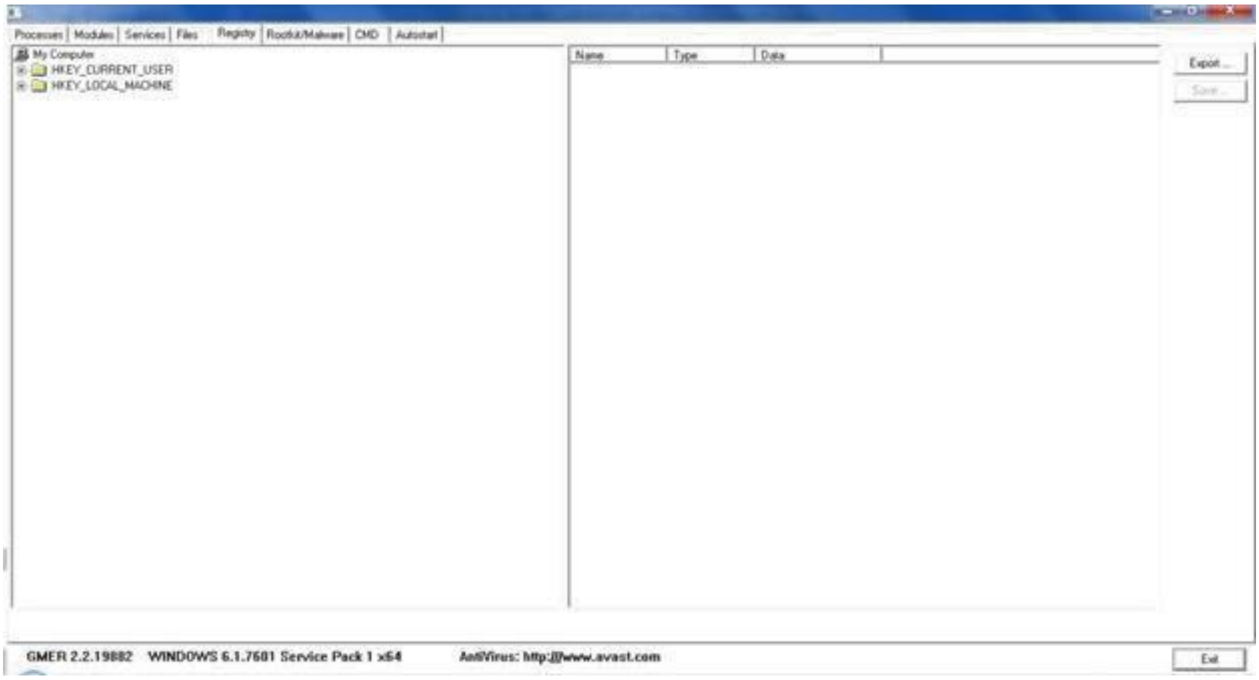


```
My Computer
⊟ C:
    ⊞ Program Files (x86)
    ⊞ Users
    ⊞ $Recycle.Bin
    ⊞ Config.Msi
    ⊞ Documents and Settings
    ⊞ found.000
    ⊞ found.001
    ⊞ found.002
    ⊞ found.003
    ⊞ found.004
    ⊞ found.005
    ⊞ found.006
    ⊞ found.007
    ⊞ found.008
    ⊞ inetpub
    ⊞ Intel
    ⊞ Kaspersky Rescue Disk 10.0
    ⊞ kfsensor
    ⊞ MSOCache
    ⊞ Program Files
    ⊞ ProgramData
    ⊞ Recovery
    ⊞ System Volume Information
    ⊞ Temp
    ⊞ TurboC4
    ⊞ Windows
⊞ D:
⊞ E:
```

```
$Recycle.Bin
Config.Msi
Documents and Settings
found.000
found.001
found.002
found.003
found.004
found.005
found.006
found.007
found.008
inetpub
Intel
Kaspersky Rescue Disk 10.0
kfsensor
MSOCache
Program Files
Program Files (x86)
ProgramData
Recovery
System Volume Information
Temp
TurboC4
Users
Windows
HaxLogs.txt
hiberfil.sys
pagefile.sys
```

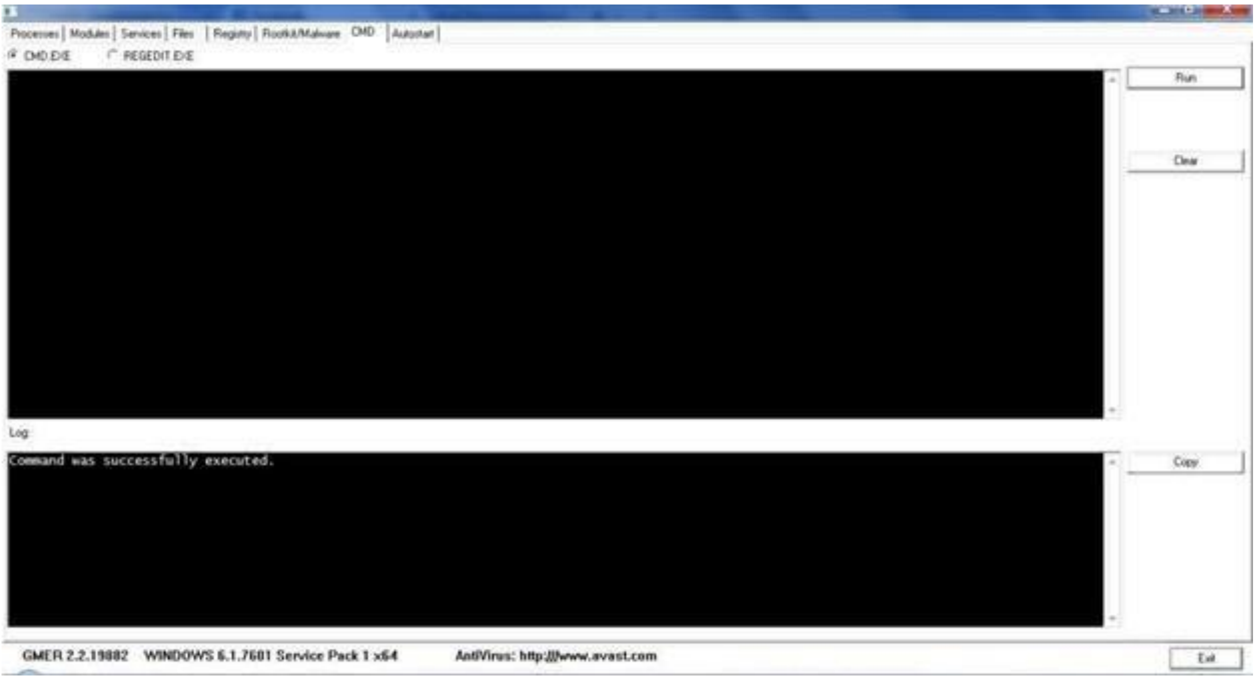
C:

GMER 2.2.19882 WINDOWS 6.1.7601 Service Pack 1 x64 AntiVirus: http://www.avast.com



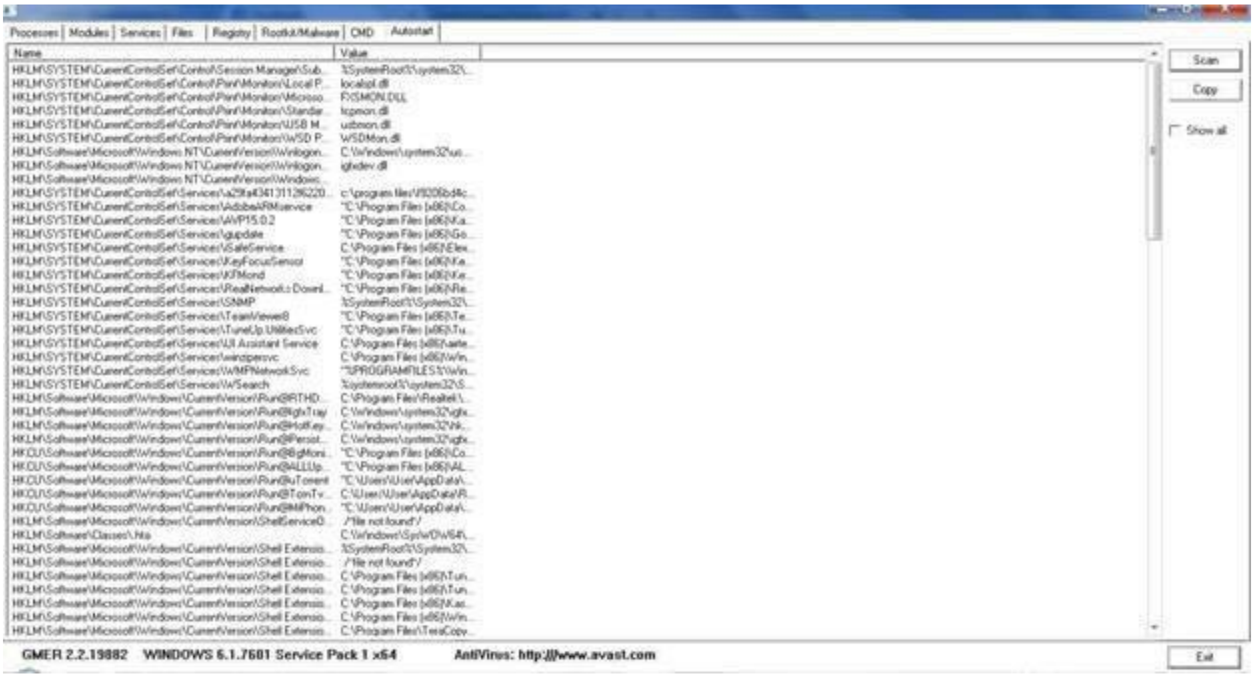
```
My Computer
⊞ HKEY_CURRENT_USER
⊞ HKEY_LOCAL_MACHINE
```

GMER 2.2.19882 WINDOWS 6.1.7601 Service Pack 1 x64 AntiVirus: http://www.avast.com



```
Command was successfully executed.
```

GMER 2.2.19882 WINDOWS 6.1.7601 Service Pack 1 x64 AntiVirus: http://www.avast.com



Name	Value
HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Sub...	%SystemRoot%\system32\...
HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\Local P...	localspl.dll
HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\Microso...	FXSMON.DLL
HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\Standar...	tcpmon.dll
HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\USB M...	usbmon.dll
HKLM\SYSTEM\CurrentControlSet\Control\Print\Monitors\WSD P...	WSDMon.dll
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon...	C:\Windows\system32\us...
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon...	igfxdev.dll
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows...	
HKLM\SYSTEM\CurrentControlSet\Services\a29fa4341311286220...	c:\program files\f9206bd4c...
HKLM\SYSTEM\CurrentControlSet\Services\AdobeARMservice	"C:\Program Files (x86)\Co...
HKLM\SYSTEM\CurrentControlSet\Services\AVP15.0.2	"C:\Program Files (x86)\Ka...
HKLM\SYSTEM\CurrentControlSet\Services\gupdate	"C:\Program Files (x86)\Go...
HKLM\SYSTEM\CurrentControlSet\Services\iSafeService	C:\Program Files (x86)\Elex...
HKLM\SYSTEM\CurrentControlSet\Services\KeyFocusSensor	"C:\Program Files (x86)\Ke...
HKLM\SYSTEM\CurrentControlSet\Services\KFMond	"C:\Program Files (x86)\Ke...
HKLM\SYSTEM\CurrentControlSet\Services\RealNetworks Downl...	"C:\Program Files (x86)\Re...
HKLM\SYSTEM\CurrentControlSet\Services\SNMP	%SystemRoot%\System32\...
HKLM\SYSTEM\CurrentControlSet\Services\TeamViewer8	"C:\Program Files (x86)\Te...
HKLM\SYSTEM\CurrentControlSet\Services\TuneUp.UtilitiesSvc	"C:\Program Files (x86)\Tu...
HKLM\SYSTEM\CurrentControlSet\Services\UI Assistant Service	C:\Program Files (x86)\aste...
HKLM\SYSTEM\CurrentControlSet\Services\winzipersvc	C:\Program Files (x86)\Win...
HKLM\SYSTEM\CurrentControlSet\Services\WMPNetworkSvc	"%PROGRAMFILES%\Win...
HKLM\SYSTEM\CurrentControlSet\Services\WSearch	%systemroot%\system32\S...
HKLM\Software\Microsoft\Windows\CurrentVersion\Run@RTHD...	C:\Program Files\Realtek\...
HKLM\Software\Microsoft\Windows\CurrentVersion\Run@IgfxTray	C:\Windows\system32\igfx...
HKLM\Software\Microsoft\Windows\CurrentVersion\Run@HotKey...	C:\Windows\system32\hk...
HKLM\Software\Microsoft\Windows\CurrentVersion\Run@Persist...	C:\Windows\system32\igfx...
HKCU\Software\Microsoft\Windows\CurrentVersion\Run@BgMoni...	"C:\Program Files (x86)\Co...
HKCU\Software\Microsoft\Windows\CurrentVersion\Run@ALLUp...	"C:\Program Files (x86)\AL...
HKCU\Software\Microsoft\Windows\CurrentVersion\Run@uTorrent	"C:\Users\User\AppData\...
HKCU\Software\Microsoft\Windows\CurrentVersion\Run@TomTv...	C:\Users\User\AppData\R...
HKCU\Software\Microsoft\Windows\CurrentVersion\Run@MPhon...	"C:\Users\User\AppData\...
HKLM\Software\Microsoft\Windows\CurrentVersion\ShellServiceO...	/*file not found*/
HKLM\Software\Classes\.hta	C:\Windows\SysWOW64\...
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	%SystemRoot%\System32\...
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	/*file not found*/
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	C:\Program Files (x86)\Tun...
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	C:\Program Files (x86)\Tun...
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	C:\Program Files (x86)\Kas...
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	C:\Program Files (x86)\Win...
HKLM\Software\Microsoft\Windows\CurrentVersion\Shell Extensio...	C:\Program Files\TeraCopy...

GMER 2.2.19882 WINDOWS 6.1.7601 Service Pack 1 x64 AntiVirus: http://www.avast.com

VIVA QUESTIONS (PRE LAB and POSTLAB):

1. List any two web security threats.
2. List any two design goals for a firewall.
3. Define Security Mechanism.
4. What is S/MIME?
5. Define IPSec.
6. What are the key features of SET?
7. What is the use of public key encryptionscheme?
8. Identify the possible threats for RSA algorithm.
9. List out the general schemes for the distribution of public keys.
10. What are the areas where Kerberos Version 5 addresses the limitation of Version 4?

RESULT:

Thus the Rootkits tool was installed and its various options were verified successfully

EX.No.: 12	TRIPLE DES

AIM:

To implement the TRIPLE DES in java.

ALGORITHM:

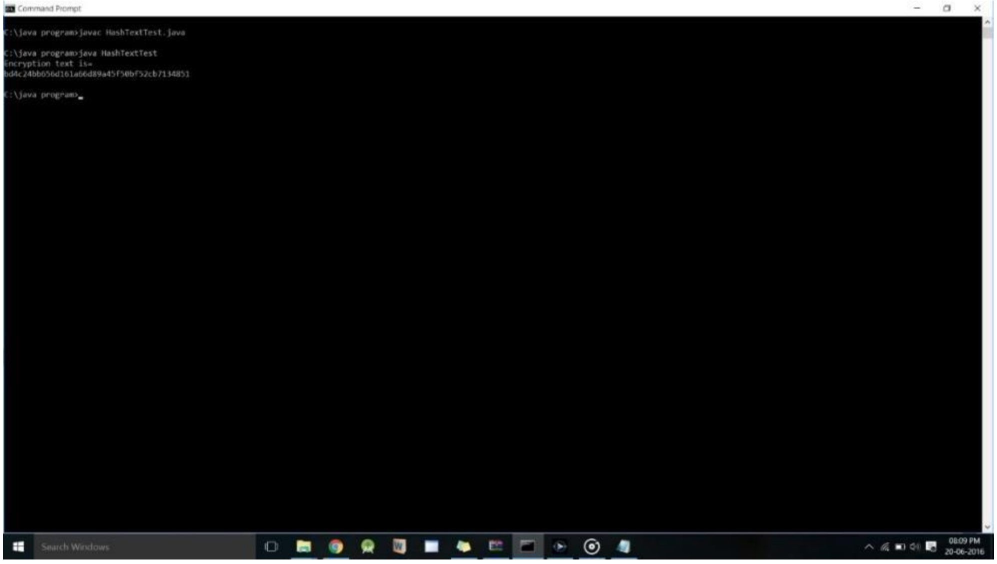
1. Start the program.
2. Encrypt the plaintext blocks using single DES with key K_1.
3. Now decrypt the output of step 1 using single DES with key K_2.
4. Finally, encrypt the output of step 2 using single DES with key K_3.
5. The output of step 3 is the ciphertext.
6. Decryption of a ciphertext is a reverse process. User first decrypt using $K_{3,}$ then encrypt with $K_{2,}$ and finally decrypt with K_1.
7. Stop the program.

PROGRAM:

```
import java.util.Arrays;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;
public class TripleDESTest
{
          public static void main(String[] args) throws Exception
      {
         String text = "textToEncrypt";
         String codedtext = new TripleDESTest()._encrypt(text,"SecretKey");
         String decodedtext = new TripleDESTest()._decrypt(codedtext,"SecretKey");
              System.out.println(codedtext + " ---> " + decodedtext);
       }
      private String _encrypt(String message, String secretKey) throws Exception
      {
       MessageDigest md = MessageDigest.getInstance("SHA-1");
              byte[] digestOfPassword = md.digest(secretKey.getBytes("utf-8"));
              byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);

              SecretKey key = new SecretKeySpec(keyBytes, "DESede");
              Cipher cipher = Cipher.getInstance("DESede");
              cipher.init(Cipher.ENCRYPT_MODE, key);

              byte[] plainTextBytes = message.getBytes("utf-8");
         byte[] buf = cipher.doFinal(plainTextBytes);
```

```
            byte [] base64Bytes = Base64.encodeBase64(buf);
            String base64EncryptedString = new String(base64 Bytes);

            return base64EncryptedString;
        }
        private String _decrypt(String encryptedText, String secretKey) throws Exception {
            byte[] message = Base64.decodeBase64(encryptedText.getBytes("utf-8"));
                  MessageDigest md = MessageDigest.getInstance("SHA-1");
                  byte[] digestOfPassword = md.digest(secretKey.getBytes("utf-8"));
                  byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24); SecretKey
                  key = new SecretKeySpec(keyBytes, "DESede");

                  Cipher decipher = Cipher.getInstance("DESede");
                  decipher.init(Cipher.DECRYPT_MODE, key);

                  byte[] plainText = decipher.doFinal(message);

                  return new String(plainText, "UTF-8");
        }
}
```

OUTPUT:



```
C:\java program>javac HashTextTest.java

C:\java program>java HashTextTest
Encryption text is=
bd4c24bb656d161a66d89a45f50bf52cb7134851

C:\java program>_
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. Is Dsa Secure?
2. Is The Use Of Dsa Covered By Any Patents?
3. What Are Special Signature Schemes?
4. The DES algorithm has a key length of
5. In the DES algorithm, although the key size is 64 bits only 48bits are used for the encryption procedure, the rest are parity bits.
6. The Initial Permutation table/matrix is of size
7. The number of unique substitution boxes in DES after the 48 bit XOR operationare
8. In the DES algorithm the 64 bit key input is shortened to 56 bits by ignoring every 4th bit. True or False?
9. Which structure follows DES?
10. During decryption, we use the Inverse Initial Permutation (IP-1) before the IP. True or false?
11. A preferable cryptographic algorithm should have a good avalanche effect. True or false?
12. The number of tests required to break the DES algorithm are
13. The number of tests required to break the Double DES algorithm are
14. How many keys does the Triple DES algorithm use?

RESULT:

Thus the program to implement the TRIPLE DES in java has been executed and the output was verified successfully.