INDEX

EX NO	DATE OF EXPERIMENT	TITLE	DATE OF COMPLETION	SIGN
1		Implementation of Substitution Techniques		
1(a)		Caesar Cipher		
1(b)		Playfair Cipher		
1(c)		Hill Cipher		
1(d)		Vignere Cipher		
2		Implementation of Transposition Technique		
2(a)		Rail fence		
2(b)		Row and Column Transformation		
3		DES Algorithm		
4		AES Algorithm		
5		RSA Algorithm		
6		Diffie-Hellman Key Exchange Algorithm		
7		SHA-1 Algorithm		
8		Digital Signature Standard		

9		Intrusion Detection System		
10		Automated Attack and Penetration Tools		
11(a)		Defeating Malware- Building Trojans		
11 (b)		Rootkit Hunter		

CONTENT BEYOND SYLLABUS

EX NO	DATE OF EXPERIMENT	TITLE	DATE OF COMPLETION	SIGN
12		Mono Alphabetic Cipher		
13		Triple DES		

EX NO:1(A)	CAESAR CIPHER
DATE:	

AIM:

To create a program Caeser cipher using C++ or Java Programming.

ALGORITHM:

Step 1: Start the program.

Step 2: Create Caesar cipher (shift cipher) is a simple substitution cipher based on a replacement of every single character of the open text with a character, which is fixed number of positions further down the alphabet.

Step 3: Declare and define the structure Julius Caesar was used only the shift of 3 characters, but nowadays the term Caesar cipher refers to all variants (shifts) of this cryptosystem.

Step 4: Develop can define transformation as:

a b c d e f g h i j k l m n o p q r s t u v w x y z

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

Step 5: Demonstrate the mathematically give each letter a number

a b c d e f g h i j k l m

0 1 2 3 4 5 6 7 8 9 10 11 12

n o p q r s t u v w x y Z

13 4 15 16 17 18 19 20 21 22 23 24 25

Step 6: Initialize the Caesar cipher as:

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

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

Step 7: Stop the program

PROGRAM:

```
import java.util.Scanner;
public class CaesarCipher
{
  public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";
   public static String encrypt(String plainText, int shiftKey)
          {
```

```
        plainText = plainText.toLowerCase();
        String cipherText = "";
        for (int i = 0; i<plainText.length(); i++)
                {
                 int charPosition = ALPHABET.indexOf(plainText.charAt(i));
                int keyVal = (shiftKey + charPosition) % 26;
                char replaceVal = ALPHABET.charAt(keyVal);
                cipherText += replaceVal;
                }
            return cipherText;
         }
 public static String decrypt(String cipherText, int shiftKey)
 {
        cipherText = cipherText.toLowerCase();
        String plainText = "";
        for (int i = 0; i<cipherText.length(); i++)
        {
         int charPosition = ALPHABET.indexOf(cipherText.charAt(i));
         int keyVal = (charPosition - shiftKey) % 26;
        if (keyVal< 0)
        {
        keyVal = ALPHABET.length() + keyVal;
        }
         char replaceVal = ALPHABET.charAt(keyVal);
         plainText += replaceVal;
    }
    return plainText;
}
public static void main(String[] args)
{
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the String for Encryption: ");
        String message = new String();
        message = sc.next();
        System.out.println(encrypt(message, 3));
        System.out.println(decrypt(encrypt(message, 3), 3));
        sc.close();
  }
}
```

OUTPUT:

Enter the String for Encryption:
Hello World
Khoor Zruog

Hello World

INFERENCE:

RESULT:

EX NO:1(B)	PLAYFAIR CIPHER
DATE:	

AIM:

To create program play fair using C++ Programming.

ALGORITHM:

Step 1: Start the program.

Step 2: Create not even the large number of keys in a monoalphabetic cipher provides security.

Step 3: Develop one approach to improving security was to encrypt multiple letters.

Step 4: Initialize a 5x5 matrix of letters based on a keyword (I and J aren't distinguished)
fill in letters of keyword (sans duplicates)
fill rest of matrix with other letters
eg. Using the keyword MONARCHY
MONAR
CHYBD
EFGIK
LPQST
UVWXZ
The frequency analysis used for simple substitution ciphers does not work with it.

Step 5: Stop the program

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char v,w,ch,string[100],arr[5][5],key[10],a,b,enc[100];
int temp,i,j,k,l,r1,r2,c1,c2,t,var;
FILE * fp;
fp=fopen("sk.txt","r");   //keep message in sk.txt (e.g. jamia)
clrscr();
printf("Enter the key\n");
fflush(stdin);
scanf("%s",&key);
l=0;
```

```
while(1)
{
ch=fgetc(fp);
if(ch!=EOF)
        {

        string[l++]=ch;
        }
if(ch==EOF)
break;
}
string[l]='\0';
puts(string);
for(i=0;key[i]!='\0';i++)
{
for(j=i+1;key[j]!='\0';j++)
    {
if(key[i]==key[j])
        {
           temp=1;
break;
        }
    }
}
if(temp==1)
printf("invalid key");
else
{
k=0;
a='a';
//printf("%c",b);
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
    {
if(k<strlen(key))
arr[i][j]=key[k];
elseif(k==strlen(key))
    {
    b:
for(l=0;l<strlen(key);l++)
```

```
    {
if(key[l]==a)
        {
  a++;
goto b;
        }
    }
arr[i][j]=a;
if(a=='i')
     a=a+2;
else
     a++;
    }
if(k<strlen(key))
     k++;
    }
}
printf("\n");
printf("The matrix is\n");
for(i=0;i<5;i++)
{
for(j=0;j<5;j++)
    {
printf("%c",arr[i][j]);
    }
printf("\n");
}
t=0;
if(strlen(string)%2!=0)
var=strlen(string)-1;
for(i=0;i<var;)
{
    v=string[i++];
    w=string[i++];
if(v==w)
    {
     enc[t++]=v;
     enc[t++]='$';
    }
else
    {
```

```
for(l=0;l<5;l++)
        {
for(k=0;k<5;k++)
            {
if(arr[l][k]==v||v=='j'&&arr[l][k]=='i')
       {
        r1=l;
        c1=k;
       }
if(arr[l][k]==w||w=='j'&&arr[l][k]=='i')
       {
        r2=l;
        c2=k;
       }
            }
        }
if(c1==c2)
        {
  r1++;
  r2++;
if(r1==5||r2==5)
  {
   r1=0;
   r2=0;
  }
        }
elseif(r1==r2)
        {
  c1++;
  c2++;
if(c1==5||c2==5)
  {
   c1=0;
   c2=0;
  }
        }
else
        {
  temp=r1;
  r1=r2;
  r2=temp;
```

```
        }
        enc[t++]=arr[r1][c1];
        enc[t++]=arr[r2][c2];

    }
}
if(strlen(string)%2!=0)
 enc[t++]=string[var];
enc[t]='\0';
}
printf("The encrypted text is\n");
puts(enc);
getch();
}
```

OUTPUT:

```
Enter the key
3421
mohsin

The matrix is
3421a
bcdef
ghikl
mnopq
rstuv
The encrypted text is
np4$
```

INFERENCE:

RESULT:

EX NO:1(C)	HILL CIPHER
DATE:	

AIM:

To create program Hill cipher using C++ Programming.

ALGORITHM:

Step 1: Start the program
Step 2: To create the technique encrypts pairs of letters
Step 3: Develop instead of single letters as in the simple substitution cipher and rather more complex Vignere cipher systems then in use.
Step 4: Declare the Play fair is thus significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it.
Step 5: Stop the program

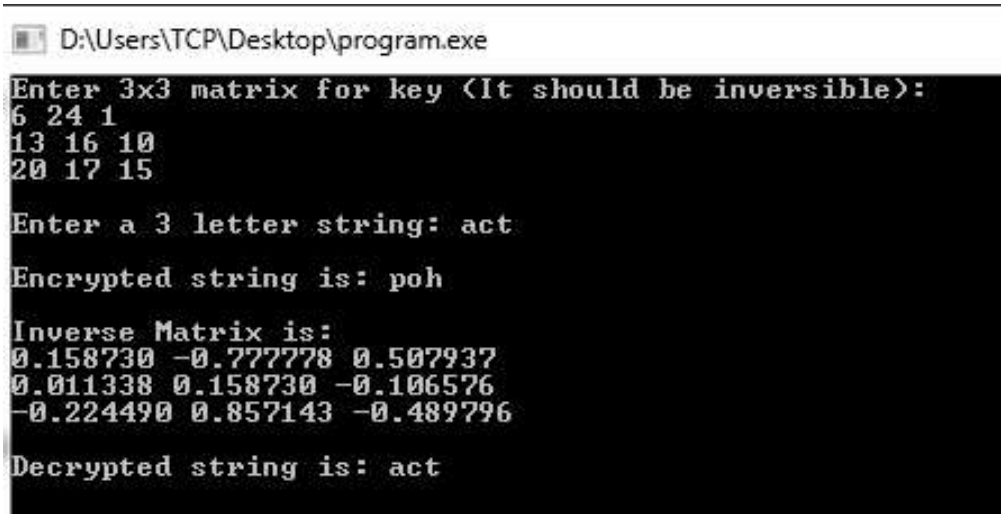
PROGRAM:

```
#include<stdio.h>
#include<math.h>

float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];

void encryption();          //encrypts the message
void decryption();          //decrypts the message
void getKeyMessage(); //gets key and message from user
void inverse();             //finds inverse of key matrix
void main() {
  getKeyMessage();
  encryption();
  decryption();
}

void encryption() {
  int i, j, k;

  for(i = 0; i< 3; i++)
          for(j = 0; j < 1; j++)
                  for(k = 0; k < 3; k++)
                          encrypt[i][j] = encrypt[i][j] + a[i][k] * mes[k][j];

  printf("\nEncrypted string is: ");
```

```
  for(i = 0; i< 3; i++)
        printf("%c", (char)(fmod(encrypt[i][0], 26) + 97));

}

void decryption() {
  int i, j, k;

  inverse();

  for(i = 0; i< 3; i++)
        for(j = 0; j < 1; j++)
              for(k = 0; k < 3; k++)
                    decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];

  printf("\nDecrypted string is: ");
  for(i = 0; i< 3; i++)
        printf("%c", (char)(fmod(decrypt[i][0], 26) + 97));

  printf("\n");
}

void getKeyMessage() {
  int i, j;
  char msg[3];

  printf("Enter 3x3 matrix for key (It should be inversible):\n");

  for(i = 0; i< 3; i++)
        for(j = 0; j < 3; j++) {
              scanf("%f", &a[i][j]);
              c[i][j] = a[i][j];
        }

  printf("\nEnter a 3 letter string: ");
  scanf("%s", msg);

  for(i = 0; i< 3; i++)
        mes[i][0] = msg[i] - 97;
}

void inverse() {
  int i, j, k;
  float p, q;

  for(i = 0; i< 3; i++)
        for(j = 0; j < 3; j++) {
              if(i == j)
```

```
                b[i][j]=1;
            else
                b[i][j]=0;
        }

    for(k = 0; k < 3; k++) {
        for(i = 0; i< 3; i++) {
            p = c[i][k];
            q = c[k][k];

            for(j = 0; j < 3; j++) {
                if(i != k) {
                    c[i][j] = c[i][j]*q  - p*c[k][j];
                    b[i][j] = b[i][j]*q  - p*b[k][j];
                }
            }
        }
    }

    for(i = 0; i< 3; i++)
        for(j = 0; j < 3; j++)
            b[i][j] = b[i][j] / c[i][i];

    printf("\n\nInverse Matrix is:\n");
    for(i = 0; i< 3; i++) {
        for(j = 0; j < 3; j++)
            printf("%d ", b[i][j]);

        printf("\n");
    }
}
```

OUTPUT:



INFERENCE:

RESULT:

EX NO:1(D)	VIGNERE CIPHER
DATE:	

AIM:

To create program Vignere using C++ Programming.

ALGORITHM:

Step 1: Start the program
Step 2: Create the simplest polyalphabetic substitution cipher is the Vignere Cipher effectively multiple Caesar ciphers
Step 3: key is multiple letters long K = k1 k2 ... kd
Step 4: develop the i^{th} letter specifies i^{th} alphabet to use each alphabet in turn
Step 5: repeat from start after d letters in message
Step 6: decryption simply works in reverse
Step 7: Develop to write the plaintext out
Step 8: Develop write the keyword repeated above it
Step 9: initialize use each key letter as a Caesar cipher key and encrypt the corresponding plaintext letter
eg using keyword *deceptive*
Key: deceptivedeceptivedeceptive
Plaintext: wearediscoveredsaveyourself
ciphertext:ZICVTWQNGRZGVTWAVZHCQYGLMGJ
Step 10: Stop the program

PROGRAM:

```
import java.util.*;
public class vigenereCipher
{
static String encode(String text, final String key)
{
String res = "";
text = text.toUpperCase();
for (int i = 0, j = 0; i<text.length(); i++)
{
char c = text.charAt(i);
if (c < 'A' || c > 'Z')
{
continue;
}
```

```
res += (char)((c + key.charAt(j) - 2 * 'A') % 26 + 'A');
j = ++j % key.length();
}
return res;

}
static String decode(String text, final String key)
{
String res = "";
text = text.toUpperCase();
for (int i = 0, j = 0; i<text.length();
i++
)
{
char c = text.charAt(i);
if (c < 'A' || c > 'Z')
{
continue;
}
res += (char)((c - key.charAt(j) + 26) % 26 + 'A');
j = ++j % key.length();
}
return res;
}
public static void main (String[] args) throws java.lang.Exception
{
String key = "VIGENERECIPHER";
String msg = "SecurityLaboratory";
System.out.println("simulation of Vigenere Cipher");
System.out.println("input message : " + msg);
String enc = encode(msg, key);
System.out.println("encoded message : " + enc);
System.out.println("decoded message : " + decode(enc, key));
}
```

OUTPUT:

$javac vigenereCipher.java
$java -Xmx128M -Xms16M vigenereCipher
simulation of Vigenere Cipher
input message :SecurityLaboratory
encoded message : NMIYEMKCNIQVVROWXC
decoded message : SECURITYLABORATORY

INFERENCE:

RESULT:

EX NO: 2 (a)	RAIL FENCE
DATE:	

AIM:

To create program Rail fence using C++ Programming.

ALGORITHM:

Step 1: Start the program.

Step 2: Create the write message letters out diagonally over a number of rows

Step 3: Develop then read off cipher row by row
eg. write message out as:
m e m a t r h t g p r y
e t e f e t e o a a t

Step 4: giving ciphertext
MEMATRHTGPRYETEFETEOAAT

Step 5: initialize a more complex scheme then write letters of message out in rows over a specified number of columns then reorder the columns according to some key before reading off the rows
Key: 4 3 1 2 5 6 7
Plaintext: 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 y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Step 6: Stop the program

PROGRAM:

```
import java.util.*;
class RailFenceBasic{
 int depth;
String Encryption(String plainText,int depth)throws Exception
```

```
{
 int r=depth,len=plainText.length();
 int c=len/depth;
 char mat[][]=new char[r][c];
 int k=0;

 String cipherText="";

 for(int i=0;i<c;i++)
 {
  for(int j=0;j<r;j++)
  {
   if(k!=len)
    mat[j][i]=plainText.charAt(k++);
   else
    mat[j][i]='X';
  }
 }
 for(int i=0;i<r;i++)
 {
  for(int j=0;j<c;j++)
  {
   cipherText+=mat[i][j];
  }
 }
 return cipherText;
}


String Decryption(String cipherText,int depth)throws Exception
{
 int r=depth,len=cipherText.length();
 int c=len/depth;
 char mat[][]=new char[r][c];
 int k=0;

 String plainText="";


 for(int i=0;i<r;i++)
 {
  for(int j=0;j<c;j++)
  {
   mat[i][j]=cipherText.charAt(k++);
  }
 }
 for(int i=0;i<c;i++)
 {
```

```
   for(int j=0;j<r;j++)
   {
    plainText+=mat[j][i];
   }
  }

  return plainText;
 }
}

class RailFence{
 public static void main(String args[])throws Exception
 {
  RailFenceBasic rf=new RailFenceBasic();
                Scanner scn=new Scanner(System.in);
                int depth;

                String plainText,cipherText,decryptedText;

                System.out.println("Enter plain text:");
                plainText=scn.nextLine();

                System.out.println("Enter depth for Encryption:");
                depth=scn.nextInt();

  cipherText=rf.Encryption(plainText,depth);
  System.out.println("Encrypted text is:\n"+cipherText);

                decryptedText=rf.Decryption(cipherText, depth);

  System.out.println("Decrypted text is:\n"+decryptedText);

 }
}
```

OUTPUT:

```
Enter plain text:
railfencecipher
Enter depth for Encryption:
3
Encrypted text is:
rlnchafcieieepr
Decrypted text is:
railfencecipher
```

INFERENCE:

RESULT:

EX NO:2(b)	ROW & COLUMN TRANSFORMATION
DATE:	

AIM:

To implement a program for encryption and decryption by using row and column transformation technique.

ALGORITHM:

1. Consider the plain text hello world, and let us apply the simple columnar transposition technique as shown below

h	**e**	**l**	**l**
o	**w**	**o**	**r**
l	**d**		

2. The plain text characters are placed horizontally and the cipher text is created with vertical format as: holewdlolr.

Now, the receiver has to use the same table to decrypt the cipher text to plaintext

PROGRAM:

TransCipher.java

```
import java.util.*;
class TransCipher {
public static void main(String args[]) { Scanner sc = new
    Scanner(System.in); System.out.println("Enter the plain text");
    String pl = sc.nextLine();
    sc.close(); String s = ""; int
    start = 0;
    for (int i = 0; i<pl.length(); i++) { if (pl.charAt(i) == '
        ') {
            s = s + pl.substring(start, i); start = i + 1;
        }
    }
    s = s + pl.substring(start);
    System.out.print(s); System.out.println();
    // end of space deletion
```

```
        int k = s.length(); int l = 0;
        int col = 4;
        int row = s.length() / col;
        char ch[][] = new char[row][col]; for (int i = 0; i<row;
        i++) {
            for (int j = 0; j < col; j++) { if (l < k) {
                    ch[i][j] = s.charAt(l); l++;
                } else {
                    ch[i][j] = '#';
                }
            }
        }

        // arranged in matrix

        char trans[][] = new
        char[col][row]; for (int i = 0; i< row; i++) {
            for (int j = 0; j < col; j++) { trans[j][i] = ch[i][j];
            }
        }

        for (int i = 0; i< col; i++) { for(int j = 0; j < row;
            j++){
                System.out.print(trans[i][j]);
            }
        }
        // display System.out.println();
    }
}
```

OUTPUT:

```
Enter the plain text: Security
Lab SecurityLabSreictuy
```

INFERENCE:

RESULT:

EX NO: 3	DES ALGORITHM
DATE:	

AIM:

To implement DES algorithm using Java or C++ programming.

ALGORITHM:

Step 1: Start the program using Java or C++

Step 2: Create block cipher symmetric algorithm

Step 3: Develop a 64-bit plain text as given I/p to initial permutation function (IPF)

Step 4: Initialize IPF Produce two halves one is left plain text (LPF) and two is Right plain text

Step 5: Execute at the end LPT and RPT are rejoined and a final permutation (FP) is performed which is being the inverse of IP on the combined box

Step 6: Run finally the result produced 64 bits Cipher text.

Step 7: Stop the program

PROGRAM:

```
import java.util.*;
class Main {
private static class DES {
  // CONSTANTS
  // Initial Permutation Table
  int[] IP = { 58, 50, 42, 34, 26, 18,
           10, 2, 60, 52, 44, 36, 28, 20,
           12, 4, 62, 54, 46, 38,
           30, 22, 14, 6, 64, 56,
           48, 40, 32, 24, 16, 8,
           57, 49, 41, 33, 25, 17,
           9, 1, 59, 51, 43, 35, 27,
           19, 11, 3, 61, 53, 45,
           37, 29, 21, 13, 5, 63, 55,
           47, 39, 31, 23, 15, 7 };

  // Inverse Initial Permutation Table
  int[] IP1 = { 40, 8, 48, 16, 56, 24, 64,
```

```
            32, 39, 7, 47, 15, 55,
            23, 63, 31, 38, 6, 46,
            14, 54, 22, 62, 30, 37,
            5, 45, 13, 53, 21, 61,
            29, 36, 4, 44, 12, 52,
            20, 60, 28, 35, 3, 43,
            11, 51, 19, 59, 27, 34,
            2, 42, 10, 50, 18, 58,
            26, 33, 1, 41, 9, 49,
            17, 57, 25 };

// first key-hePermutation Table
int[] PC1 = { 57, 49, 41, 33, 25,
            17, 9, 1, 58, 50, 42, 34, 26,
            18, 10, 2, 59, 51, 43, 35, 27,
            19, 11, 3, 60, 52, 44, 36, 63,
            55, 47, 39, 31, 23, 15, 7, 62,
            54, 46, 38, 30, 22, 14, 6, 61,
            53, 45, 37, 29, 21, 13, 5, 28,
            20, 12, 4 };

// second key-Permutation Table
int[] PC2 = { 14, 17, 11, 24, 1, 5, 3,
            28, 15, 6, 21, 10, 23, 19, 12,
            4, 26, 8, 16, 7, 27, 20, 13, 2,
            41, 52, 31, 37, 47, 55, 30, 40,
            51, 45, 33, 48, 44, 49, 39, 56,
            34, 53, 46, 42, 50, 36, 29, 32 };

// Expansion D-box Table
int[] EP = { 32, 1, 2, 3, 4, 5, 4,
           5, 6, 7, 8, 9, 8, 9, 10,
           11, 12, 13, 12, 13, 14, 15,
           16, 17, 16, 17, 18, 19, 20,
           21, 20, 21, 22, 23, 24, 25,
           24, 25, 26, 27, 28, 29, 28,
           29, 30, 31, 32, 1 };

// Straight Permutation Table
int[] P = { 16, 7, 20, 21, 29, 12, 28,
          17, 1, 15, 23, 26, 5, 18,
          31, 10, 2, 8, 24, 14, 32,
          27, 3, 9, 19, 13, 30, 6,
          22, 11, 4, 25 };

// S-box Table
int[][][] sbox = {
```

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

    { { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10 },
      { 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5 },
      { 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15 },
      { 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 } },
    { { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8 },
      { 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1 },
      { 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7 },
      { 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 } },
    { { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15 },
      { 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9 },
      { 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4 },
      { 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 } },
    { { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9 },
      { 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6 },
      { 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14 },
      { 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 } },
    { { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11 },
      { 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8 },
      { 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6 },
      { 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 } },
    { { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1 },
      { 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6 },
      { 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2 },
      { 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 } },
    { { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7 },
      { 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2 },
      { 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8 },
      { 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 } }
};
int[] shiftBits = { 1, 1, 2, 2, 2, 2, 2, 2,
                    1, 2, 2, 2, 2, 2, 2, 1 };

// hexadecimal to binary conversion
String hextoBin(String input)
{
    int n = input.length() * 4;
    input = Long.toBinaryString(
        Long.parseUnsignedLong(input, 16));
    while (input.length() < n)
        input = "0" + input;
    return input;
}
```

```
// binary to hexadecimal conversion
String binToHex(String input)
{
    int n = (int)input.length() / 4;
    input = Long.toHexString(
        Long.parseUnsignedLong(input, 2));
    while (input.length() < n)
        input = "0" + input;
    return input;
}

// per-mutate input hexadecimal
// according to specified sequence
String permutation(int[] sequence, String input)
{
    String output = "";
    input = hextoBin(input);
    for (int i = 0; i<sequence.length; i++)
        output += input.charAt(sequence[i] - 1);
    output = binToHex(output);
    return output;
}

// xor 2 hexadecimal strings
String xor(String a, String b)
{
    // hexadecimal to decimal(base 10)
    long t_a = Long.parseUnsignedLong(a, 16);
    // hexadecimal to decimal(base 10)
    long t_b = Long.parseUnsignedLong(b, 16);
    // xor
    t_a = t_a ^ t_b;
    // decimal to hexadecimal
    a = Long.toHexString(t_a);
    // prepend 0's to maintain length
    while (a.length() <b.length())
        a = "0" + a;
    return a;
}

// left Circular Shifting bits
String leftCircularShift(String input, int numBits)
{
    int n = input.length() * 4;
    int perm[] = new int[n];
    for (int i = 0; i< n - 1; i++)
```

```
        perm[i] = (i + 2);
    perm[n - 1] = 1;
    while (numBits-- > 0)
        input = permutation(perm, input);
    return input;
}

// preparing 16 keys for 16 rounds
String[] getKeys(String key)
{
    String keys[] = new String[16];
    // first key permutation
    key = permutation(PC1, key);
    for (int i = 0; i< 16; i++) {
        key = leftCircularShift(
                key.substring(0, 7), shiftBits[i])
            + leftCircularShift(key.substring(7, 14),
                                shiftBits[i]);
        // second key permutation
        keys[i] = permutation(PC2, key);
    }
    return keys;
}

// s-box lookup
String sBox(String input)
{
    String output = "";
    input = hextoBin(input);
    for (int i = 0; i< 48; i += 6) {
        String temp = input.substring(i, i + 6);
        int num = i / 6;
        int row = Integer.parseInt(
            temp.charAt(0) + "" + temp.charAt(5), 2);
        int col = Integer.parseInt(
            temp.substring(1, 5), 2);
        output += Integer.toHexString(
            sbox[num][row][col]);
    }
    return output;
}

String round(String input, String key, int num)
{
    // fk
    String left = input.substring(0, 8);
    String temp = input.substring(8, 16);
```

```
    String right = temp;
    // Expansion permutation
    temp = permutation(EP, temp);
    // xor temp and round key
    temp = xor(temp, key);
    // lookup in s-box table
    temp = sBox(temp);
    // Straight D-box
    temp = permutation(P, temp);
    // xor
    left = xor(left, temp);
    System.out.println("Round "
                    + (num + 1) + " "
                    + right.toUpperCase()
                    + " " + left.toUpperCase() + " "
                    + key.toUpperCase());

    // swapper
    return right + left;
}

String encrypt(String plainText, String key)
{
    int i;
    // get round keys
    String keys[] = getKeys(key);

    // initial permutation
    plainText = permutation(IP, plainText);
    System.out.println(
        "After initial permutation: "
        + plainText.toUpperCase());
    System.out.println(
        "After splitting: L0="
        + plainText.substring(0, 8).toUpperCase()
        + " R0="
        + plainText.substring(8, 16).toUpperCase() + "\n");

    // 16 rounds
    for (i = 0; i< 16; i++) {
        plainText = round(plainText, keys[i], i);
    }

    // 32-bit swap
    plainText = plainText.substring(8, 16)
              + plainText.substring(0, 8);
```

```
    // final permutation
    plainText = permutation(IP1, plainText);
    return plainText;
}

String decrypt(String plainText, String key)
{
    int i;
    // get round keys
    String keys[] = getKeys(key);

    // initial permutation
            plainText = permutation(IP, plainText);
            System.out.println(
                "After initial permutation: "
                + plainText.toUpperCase());
            System.out.println(
                "After splitting: L0="
                + plainText.substring(0, 8).toUpperCase()
                + " R0=" + plainText.substring(8, 16).toUpperCase()
                + "\n");

            // 16-rounds
            for (i = 15; i> -1; i--) {
                plainText = round(plainText, keys[i], 15 - i);
            }

            // 32-bit swap
            plainText = plainText.substring(8, 16)
                    + plainText.substring(0, 8);
            plainText = permutation(IP1, plainText);
            return plainText;
        }
    }
    public static void main(String args[])
    {
        String text = "123456ABCD132536";
        String key = "AABB09182736CCDD";

        DES cipher = new DES();
        System.out.println("Encryption:\n");
        text = cipher.encrypt(text, key);
        System.out.println(
            "\nCipher Text: " + text.toUpperCase() + "\n");
        System.out.println("Decryption\n");
        text = cipher.decrypt(text, key);
        System.out.println(
```

```
            "\nPlain  Text: "
            + text.toUpperCase());
    }
}
```

OUTPUT:

Encryption:

After initial permutation: 14A7D67818CA18AD

After splitting: L0=14A7D678 R0=18CA18AD

Round 1 18CA18AD 5A78E394 194CD072DE8C

Round 2 5A78E394 4A1210F6 4568581ABCCE

Round 3 4A1210F6 B8089591 06EDA4ACF5B5

Round 4 B8089591 236779C2 DA2D032B6EE3

Round 5 236779C2 A15A4B87 69A629FEC913

Round 6 A15A4B87 2E8F9C65 C1948E87475E

Round 7 2E8F9C65 A9FC20A3 708AD2DDB3C0

Round 8 A9FC20A3 308BEE97 34F822F0C66D

Round 9 308BEE97 10AF9D37 84BB4473DCCC

Round 10 10AF9D37 6CA6CB20 02765708B5BF

Round 11 6CA6CB20 FF3C485F 6D5560AF7CA5

Round 12 FF3C485F 22A5963B C2C1E96A4BF3

Round 13 22A5963B 387CCDAA 99C31397C91F

Round 14 387CCDAA BD2DD2AB 251B8BC717D0

Round 15 BD2DD2AB CF26B472 3330C5D9A36D

Round 16 19BA9212 CF26B472 181C5D75C66D

Cipher Text: C0B7A8D05F3A829C

Decryption

After initial permutation: 19BA9212CF26B472

After splitting: L0=19BA9212 R0=CF26B472

Round 1 CF26B472 BD2DD2AB 181C5D75C66D

Round 2 BD2DD2AB 387CCDAA 3330C5D9A36D

Round 3 387CCDAA 22A5963B 251B8BC717D0

Round 4 22A5963B FF3C485F 99C31397C91F

Round 5 FF3C485F 6CA6CB20 C2C1E96A4BF3

Round 6 6CA6CB20 10AF9D37 6D5560AF7CA5

Round 7 10AF9D37 308BEE97 02765708B5BF

Round 8 308BEE97 A9FC20A3 84BB4473DCCC

Round 9 A9FC20A3 2E8F9C65 34F822F0C66D

Round 10 2E8F9C65 A15A4B87 708AD2DDB3C0

Round 11 A15A4B87 236779C2 C1948E87475E

Round 12 236779C2 B8089591 69A629FEC913

Round 13 B8089591 4A1210F6 DA2D032B6EE3

Round 14 4A1210F6 5A78E394 06EDA4ACF5B5

Round 15 5A78E394 18CA18AD 4568581ABCCE

Round 16 14A7D678 18CA18AD 194CD072DE8C

Plain Text: 123456ABCD132536

INFERENCE:

RESULT:

EX NO: 4	AES ALGORITHM
DATE:	

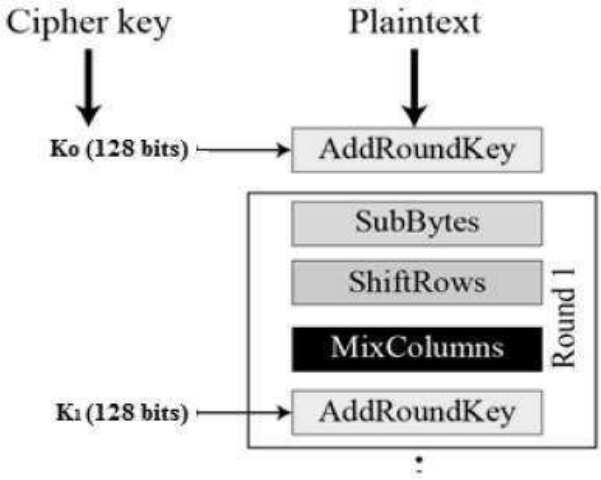
AIM:

To Write a Java program to implement the AES Algorithm.

ALGORITHM:

STEP 1: Initialize the variable

STEP 2: Encryption Process a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below



STEP 3: Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

STEP 4: Shift rows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

STEP 5: Mix Columns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

STEP 6: Add round key

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

STEP 7: Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order

- Add round key
- Mix columns
- Shift rows
- Byte substitution

PROGRAM:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.xml.bind.DatatypeConverter;

/**
 * This example program shows how AES encryption and decryption can be done in Java.
 * Please note that secret key and encrypted text is unreadable binary and hence
 * in the following program we display it in hexadecimal format of the underlying bytes.
 * @author Jayson
 */
public class AESEncryption {

    /**
     * 1. Generate a plain text for encryption
     * 2. Get a secret key (printed in hexadecimal form). In actual use this must
     * by encrypted and kept safe. The same key is required for decryption.
     * 3.
     */
```

```
    public static void main(String[] args) throws Exception {
        String plainText = "Hello World";
SecretKeysecKey = getSecretEncryptionKey();
byte[] cipherText = encryptText(plainText, secKey);
        String decryptedText = decryptText(cipherText, secKey);

System.out.println("Original Text:" + plainText);
System.out.println("AES Key (Hex Form):"+bytesToHex(secKey.getEncoded()));
System.out.println("Encrypted Text (Hex Form):"+bytesToHex(cipherText));
System.out.println("Descrypted Text:"+decryptedText);

    }

    /**
     * gets the AES encryption key. In your actual programs, this should be safely
     * stored.
     * @return
     * @throws Exception
     */
    public static SecretKeygetSecretEncryptionKey() throws Exception{
KeyGenerator generator = KeyGenerator.getInstance("AES");
generator.init(128); // The AES key size in number of bits
SecretKeysecKey = generator.generateKey();
        return secKey;
    }

    /**
     * Encrypts plainText in AES using the secret key
     * @param plainText
     * @param secKey
     * @return
     * @throws Exception
     */
    public static byte[] encryptText(String plainText,SecretKeysecKey) throws Exception{
    // AES defaults to AES/ECB/PKCS5Padding in Java 7
        Cipher aesCipher = Cipher.getInstance("AES");
aesCipher.init(Cipher.ENCRYPT_MODE, secKey);
byte[] byteCipherText = aesCipher.doFinal(plainText.getBytes());
        return byteCipherText;
    }

    /**
     * Decrypts encrypted byte array using the key used for encryption.
     * @param byteCipherText
     * @param secKey
     * @return
```

```
     * @throws Exception
     */
    public static String decryptText(byte[] byteCipherText, SecretKeysecKey) throws Exception {
    // AES defaults to AES/ECB/PKCS5Padding in Java 7
        Cipher aesCipher = Cipher.getInstance("AES");
aesCipher.init(Cipher.DECRYPT_MODE, secKey);
byte[] bytePlainText = aesCipher.doFinal(byteCipherText);
        return new String(bytePlainText);
    }

    /**
     * Convert a binary byte array into readable hex form
     * @param hash
     * @return
     */
    private static String  bytesToHex(byte[] hash) {
        return DatatypeConverter.printHexBinary(hash);
    }
}
```

OUTPUT:

$javac AESEncryption.java
$java -Xmx128M -Xms16M AESEncryption
Original Text: Hello World
AES Key (Hex Form):3B8A5850CBB16FB6D846A9ABCD1350F0
Encrypted Text (Hex Form):93FD59D738F00E5EC2938A3A88349E75
Decrypted Text: Hello World

INFERENCE:

RESULT:

EX NO: 5	RSA ALGORITHM
DATE:	

AIM:

To implement RSA algorithm using Java or C++ programming.

ALGORITHM:

Step 1: Start the program using Java or C++

Step 2: Create a RSA in a nutshell:

Step 3: Develop the Key generation:

- Select random prime numbers p and q, and check that p != q
- Compute modulus n = pq
- Compute phi = (p - 1)(q - 1)
- Select public exponent e, 1 < e < phi(n) such that gcd(e, phi(n)) = 1
- Compute private exponent d = e - 1 mod phi(n)
- Public key is {n, e}, private key is d

Step 4: Encryption: c = me mod n, decryption: m = cd mod n

Step 5: Digital signature: s = H (m) d mod n, verification: m' = se mod n,

If m' = H (m) signature is correct. H is a publicly known hash Function.

Step 6: Stop the program

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;
char msg[100];
int prime(long int);
```

```
void ce();
long int cd(long int);
void encrypt();
void decrypt();
void main()
{
clrscr();
printf("\nENTER FIRST PRIME NUMBER\n");
scanf("%d",&p);
flag=prime(p);
if(flag==0)

{
printf("\nWRONG INPUT\n");
getch();
exit(1);
}
printf("\nENTER ANOTHER PRIME NUMBER\n");
scanf("%d",&q);
flag=prime(q);
if(flag==0||p==q)
{
printf("\nWRONG INPUT\n");
getch();
exit(1);
}
printf("\nENTER MESSAGE\n");
fflush(stdin);
scanf("%s",msg);
for(i=0;msg[i]!=NULL;i++)
m[i]=msg[i];
n=p*q;
t=(p-1)*(q-1);
ce();
printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
for(i=0;i<j-1;i++)
printf("\n%ld\t%ld",e[i],d[i]);
encrypt();
decrypt();
getch();
}
```

```
int prime(long int pr)
{
int i;
j=sqrt(pr);
for(i=2;i<=j;i++)
{
if(pr%i==0)
return 0;
}
return 1;
}
void ce()
{
int k;
k=0;
for(i=2;i<t;i++)
{
if(t%i==0)
continue;
   flag=prime(i);
if(flag==1&&i!=p&&i!=q)
   {
      e[k]=i;
      flag=cd(e[k]);
if(flag>0)
      {
         d[k]=flag;
         k++;
      }
if(k==99)
break;
   }
}
}
long int cd(long int x)
{
long int k=1;
while(1)
{
   k=k+t;
if(k%x==0)
```

```
return(k/x);
}
}
void encrypt()
{
long int pt,ct,key=e[0],k,len;
i=0;
len=strlen(msg);
while(i!=len)
{
pt=m[i];
pt=pt-96;
   k=1;
for(j=0;j<key;j++)
   {
      k=k*pt;
      k=k%n;
   }
   temp[i]=k;
ct=k+96;
en[i]=ct;
i++;
}
en[i]=-1;
printf("\nTHE ENCRYPTED MESSAGE IS\n");
for(i=0;en[i]!=-1;i++)
printf("%c",en[i]);
}
void decrypt()
{
long int pt,ct,key=d[0],k;
i=0;
while(en[i]!=-1)
{
ct=temp[i];
   k=1;
for(j=0;j<key;j++)
   {
      k=k*ct;
      k=k%n;
   }
```

```
pt=k+96;
   m[i]=pt;
i++;
}
m[i]=-1;
printf("\nTHE DECRYPTED MESSAGE IS\n");
for(i=0;m[i]!=-1;i++)
printf("%c",m[i]);
}
```

OUTPUT:

RSA Input

```
ENTER FIRST PRIME NUMBER
7

ENTER ANOTHER PRIME NUMBER
17

ENTER MESSAGE
hello
POSSIBLE VALUES OF e AND d ARE

5       77
11      35
13      37
19      91
23      71
29      53
31      31
37      13
THE ENCRYPTED MESSAGE IS
ïɷccä
THE DECRYPTED MESSAGE IS
hello
```

INFERENCE:

RESULT:

EX NO: 6	DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM
DATE:	

AIM:

To implement Diffie-Hellman algorithm using Java or C++ programming.

ALGORITHM:

Step 1: Start the program using Java or C++

Step 2: Create the Global Public Elements

Step 3: Develop the User A Key Generation

Step 4: Develop the User B Key Generation

Step 5: Generation of Secret Key by User A

Step 6: Generation of Secret Key by User B

q	Prime number
α	$\alpha < q$ and α is a primitive root of q
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

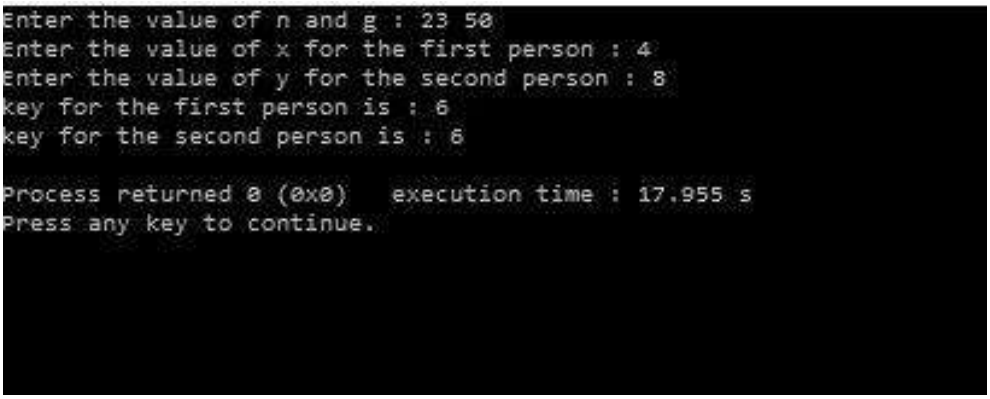
Step 7: Stop the program

PROGRAM:

```
#include<stdio.h>
long int power(int a,intb,int mod)
 {
  long long int t;
  if(b==1)
  return a;
  t=power(a,b/2,mod);
  if(b%2==0)
   return (t*t)%mod;
  else
```

```
  return (((t*t)%mod)*a)%mod;
 }
 long long int calculateKey(int a,intx,int n)
 {
  return power(a,x,n);
 }
 int main()
 {
  int n,g,x,a,y,b;
// both the persons will be agreed upon the common n and g
printf("Enter the value of n and g : ");

scanf("%d%d",&n,&g);
// first person will choose the x
printf("Enter the value of x for the first person : ");
scanf("%d",&x);  a=power(g,x,n);
// second person will choose the y
printf("Enter the value of y for the second person : ");
scanf("%d",&y);  b=power(g,y,n);
printf("key for the first person is : %lld\n",power(b,x,n));
printf("key for the second person is : %lld\n",power(a,y,n));
  return 0;
 }
```

OUTPUT:



INFERENCE:

RESULT:

EX NO:7	SHA-1 ALGORITHM
DATE:	

AIM:

To implement SHA-1 algorithm using Java or C++ programming.

ALGORITHM:

Step 1: Start the program using Java or C++

Step 2: Create Append Padding Bit

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.

Step 3: Develop the 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 original message.

Step 4: 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)

Step 5: 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)

Step 6: Initialize Buffer:

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

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

Step 7: Processing Message in 512-bit blocks

Step 8: Stop the program

PROGRAM :

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class GFG {
    public static String encryptThisString(String input)
    {
        try {
            // getInstance() method is called with algorithm SHA-1
MessageDigest md = MessageDigest.getInstance("SHA-1");
            // digest() method is called
            // to calculate message digest of the input string
            // returned as array of byte
byte[] messageDigest = md.digest(input.getBytes());
            // Convert byte array into signum representation
BigInteger no = new BigInteger(1, messageDigest);
            // Convert message digest into hex value
            String hashtext = no.toString(16);
            // Add preceding 0s to make it 32 bit
            while (hashtext.length() < 32) {
hashtext = "0" + hashtext;
            }
            // return the HashText
            return hashtext;
        }
        // For specifying wrong message digest algorithms
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
    // Driver code
    public static void main(String args[]) throws NoSuchAlgorithmException
    {
System.out.println("HashCode Generated by SHA-1 for: ");
        String s1 = "SHA1";
System.out.println("\n" + s1 + " : " + encryptThisString(s1));
        String s2 = "hello world";
System.out.println("\n" + s2 + " : " + encryptThisString(s2));
}
```

OUTPUT:

$javac GFG.java
$java -Xmx128M -Xms16M GFG
HashCode Generated by SHA-1 for:

SHA1 : e1744a525099d9a53c0460ef9cb7ab0e4c4fc939

hello world : 2aae6c35c94fcfb415dbe95f408b9ce91ee846ed

INFERENCE:

RESULT:

EX NO: 8	DIGITAL SIGNATURE STANDARD
DATE:	

AIM:

Create the signature scheme of Digital Signature Standard.

ALOGRITHM:

Step 1: Start the program using Java or C++

Step 2: Create a prime q with the same number of bits as hash and Select a L-bit prime p such that p-1 is a multiple of q

Step 3: Implement to Generator g such that $g^q = 1 \bmod p$ This can be done by $g=h^{(p-1)/q} \bmod p$ for some arbitrary h $1<h<p-1$.Algorithm parameters (p, q, g) may be shared among users.

Step 4: User Keys: public and private key for a user

Choose S randomly $0 <S <p$

$T = g^S \bmod p$

Public key is (p, q, g, T). Private key is S.

Step 5: Signing: Generate per message key S_m, $0<S_m<q$

$T_m = (g^{Sm} \bmod p) \bmod q$

Compute $S_m^{-1} \bmod q$

Calculate message digest d_m

Signature $X = S_m^{-1} (d_m + ST_m) \bmod q$

Transmit message m, per message public number T_m, and signature X

Step 6: Verification:

Calculate inverse of signature $X^{-1} \bmod q$ and Calculate message digest d_m

Calculate $x = d_m X^{-1} \bmod q$

$y = T_m x^{-1} \bmod q$

$z = (g^x . T^y \bmod p) \bmod q$

If $z = T_m$ then signature is verified.

Step 7: Stop the program

PROGRAM:

```
package java_cryptography;
// Imports
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.util.Scanner;
import javax.xml.bind.DatatypeConverter;
public class Digital_Signature{

    // Signing Algorithm
    private static final String
        SIGNING_ALGORITHM
        = "SHA256withRSA";
    private static final String RSA = "RSA";
    private static Scanner sc;

    // Function to implement Digital signature
    // using SHA256 and RSA algorithm
    // by passing private key.
    public static byte[] Create_Digital_Signature(
        byte[] input,
        PrivateKey Key)
        throws Exception
    {
        Signature signature
            = Signature.getInstance(
                SIGNING_ALGORITHM);
        signature.initSign(Key);
        signature.update(input);
        return signature.sign();
    }
    // Generating the asymmetric key pair
    // using SecureRandom class
    // functions and RSA algorithm.
    public static KeyPairGenerate_RSA_KeyPair()
        throws Exception
    {
        SecureRandomsecureRandom
            = new SecureRandom();
        KeyPairGeneratorkeyPairGenerator
            = KeyPairGenerator
                .getInstance(RSA);
        keyPairGenerator
```

```
        .initialize(
            2048, secureRandom);
    return keyPairGenerator
        .generateKeyPair();
}
// Function for Verification of the
// digital signature by using the public key
public static boolean
Verify_Digital_Signature(
    byte[] input,
    byte[] signatureToVerify,
    PublicKey key)
    throws Exception
{
    Signature signature
        = Signature.getInstance(
            SIGNING_ALGORITHM);
    signature.initVerify(key);
    signature.update(input);
    return signature
        .verify(signatureToVerify);
}

// Driver Code
public static void main(String args[])
    throws Exception
{

    String input
        = "COMPUTER SCIENCE";
    KeyPairkeyPair
        = Generate_RSA_KeyPair();

    // Function Call
    byte[] signature
        = Create_Digital_Signature(
            input.getBytes(),
            keyPair.getPrivate());

    System.out.println(
        "Signature Value:\n "
        + DatatypeConverter
            .printHexBinary(signature));

    System.out.println(
        "Verification: "
        + Verify_Digital_Signature(
            input.getBytes(),
```

```
            signature, keyPair.getPublic()));
    }
}
```

OUTPUT:

```
Signature Value:
2492035AE7782EEB75E18C1C76651384FDE30178DBE806A67DA4C884D52BF15A35CB8D1F
Verification: true
```

INFERENCE:

RESULT:

EX NO: 9	INTRUSION DETECTION SYSTEM
DATE:	

AIM:

To Implementation intrusion detection system (ids) using any tool.

ALOGRITHM:

Step 1: create a key technique in network security domain, Intrusion Detection System (IDS) plays vital role of detecting various kinds of attacks and secures the networks.

Step 2: Go to the web site www.snort.org/start/download

Step 3: Click on download option and support path to save the setup file.

Step 4: Double click on Snort Installation icon to run setup.

Step 5: Accept License agreement and Specify path for installation, then Click on Next.

Step 6: Install snort with or without database support.

Step 7: Skip the WinPcap driver installation

Step 8: Select all the components and Click Next.

Step 9: Install and Close.

Step 10: Add the path variable in windows environment variable by selecting new class path.

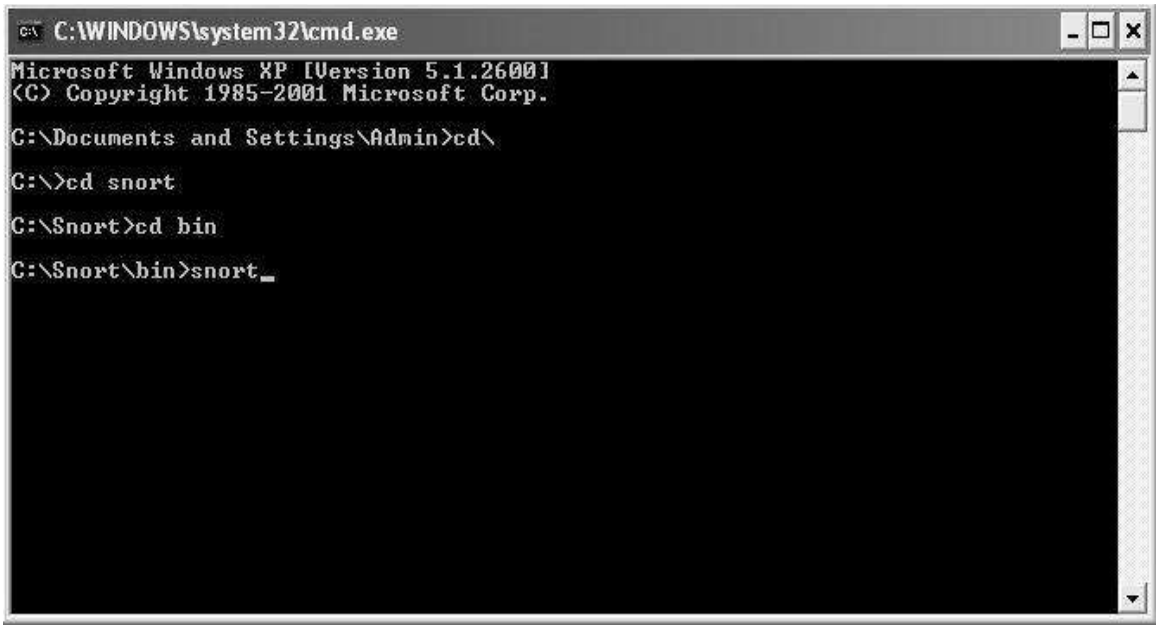
Step 11: Create a path variable and point it at snort.exe variable name: path and variable value as c:\snort\bin.

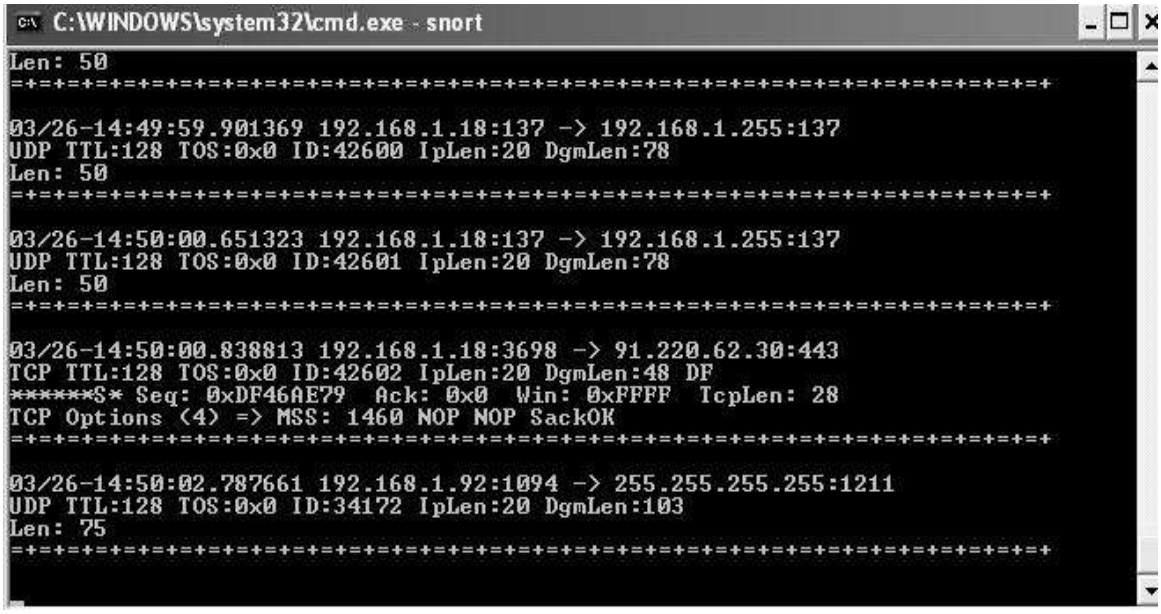
Step 12: Click OK button and then close all dialog boxes and type the following commands:

Step 13: Go to command prompt and get into Snort/bin directory and run Snort.exe file.

Step14: An editor window displays the complete details of packets flowing across the system, the IP Address of packet generator, date &Time, length of Packet, Time to live(TTL)Etc. at Real-time.

Step 15: By analyzing these details Intruders can be traced at real time.
Step 16: These details can be documents by using a print screen option.





INFERENCE:

RESULT:

EX NO: 10	AUTOMATED ATTACK AND PENETRATION TOOLS
DATE:	

AIM:

To download the N-Stalker Vulnerability Assessment Tool and exploring the features.

Exploring N-Stalker:

- N-Stalker Web Application Security Scanner is a Web security assessment tool.
- It incorporates with a well-known N-Stealth HTTP Security Scanner and35,000 Web attack signature database.
- This tool also comes in both free and paid version.
- Before scanning the target, go to "License Manager" tab, perform the update.
- Once update, you will note the status as up to date.
- You need to download and install N-Stalker from www.nstalker.com.

ALOGRITHM DESCRIPTION:

1. Start N-Stalker from a Windows computer. The program is installed under Start⇨Programs⇨N-Stalker ⇨N-Stalker Free Edition.
2. Enter a host address or a range of addresses to scan.
3. Click Start Scan.
4. After the scan completes, the N-Stalker Report Manager will prompt
5. You to select a format for the resulting report as choose Generate HTML.
6. Review the HTML report for vulnerabilities.


N-Stalker Scanner
Scan Options
Start
Scan Session
Policy Editor
Global Options
Report Manager
Macro Recorder
Scan Tools
Web Proxy
HTTP Brute Force
Web Discovery
Encoder Tool
GHDB Tool
HTTP Load Tester
Miscellaneous Tools
License Manager
-Stalker
THE WEB SECURITY SPECIALISTS
Web Security Intelligence Service
Service will expire on :
Current Status
Update Settings
Check available updates upon scanner initialization
Enable automatic updates upon scanner initialization
N-Stalker Update Status
Name
Version
Status
XSS Assessment Free DB 11012501 Up to date
Backup Finder Free 2012 11011901 Up to date
Sensitive Files Finder Free 20 11110901 Up to date
WebDAV Assessment Free 2 10102501 Up to date
Info Leak Assessment Free 2 11052401 Up to date
HTTP Method Finder Free 201 10091601 Up to date
Webserver Infrastructure Fre 11110905 Up to date
Update

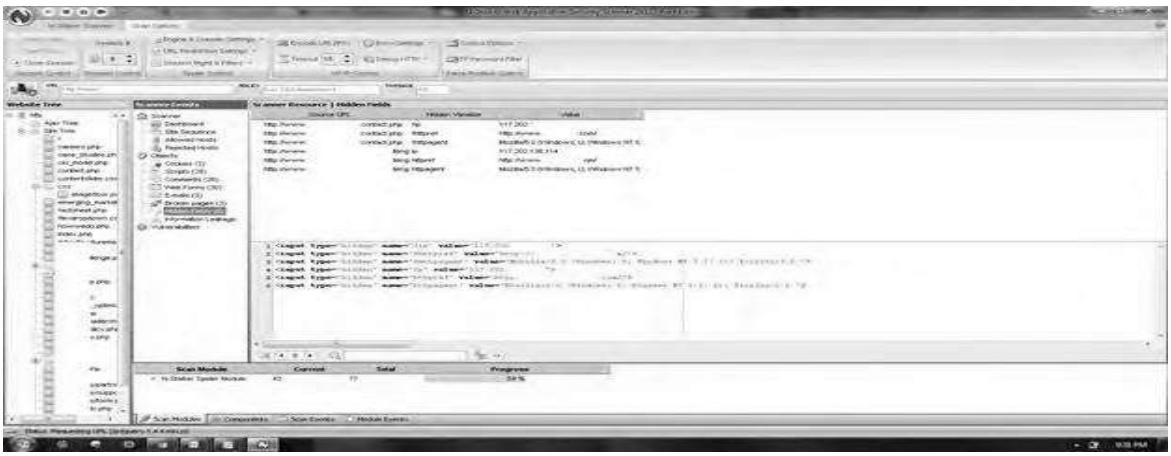
Now go to “Scan Session”, enter the target URL.

In scan policy, you can select from the four options,

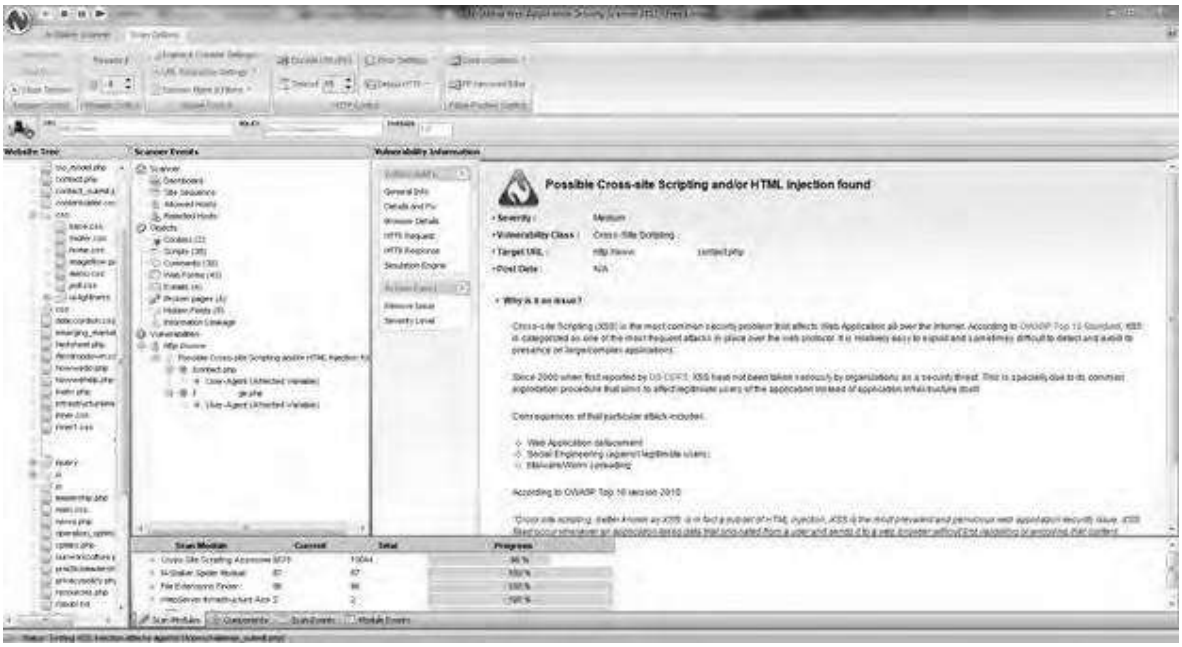
- Manual test which will crawl the website and will be waiting for manual attacks.
- full xssassessment
- owasp policy
- Web server infrastructure analysis.

Once, the option has been selected, next step is “Optimize settings” which will crawl the whole website for further analysis.

In review option, you can get all the information like host information, technologies used, policy name, etc.



Once the scan is completed, the N-Stalker scanner will show details like severity level, vulnerability class, why is it an issue, the fix for the issue and the URL which is vulnerable to the particular vulnerability?



INFERENCE:

RESULT:

EX NO: 11(i)	DEFEATING MALWARE
DATE:	

AIM:

To build a Trojan and know the harmness of the trojan malwares in a computer system.

ALOGRITHM:

1. Create a simple trojan by using Windows Batch File (***.bat***)
2. Type these below code in notepad and save it as **Trojan.bat**
3. Double click on ***Trojan.bat*** file.
4. When the trojan code executes, it will open MS-Paint, Notepad, Command Prompt, Explorer, etc., infinitely.
5. Restart the computer to stop the execution of this trojan.

Trojan:

- In computing, a Trojan horse, or trojan, is any malware which misleads users of its true intent.
- Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or anywhere else.
- Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer.
- Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity.
- ***Example:*** *Ransomware* attacks are often carried out using a *trojan*

PROGRAM:

Trojan.bat

```
@echooff
:x
Start mspaint
Start notepad
Start cmd
Start explorer
Start contrl
Start calc
goto x
```

OUTPUT:

(MS-Paint, Notepad, Command Prompt, Explorer will open infinitely)

INFERENCE:

RESULT:

EX NO: 11 (ii)	DEFEATING MALWARE - ROOTKIT HUNTER
DATE:	

AIM:

To install a rootkit hunter and find the malwares in a computer.

Rootkit Hunter:

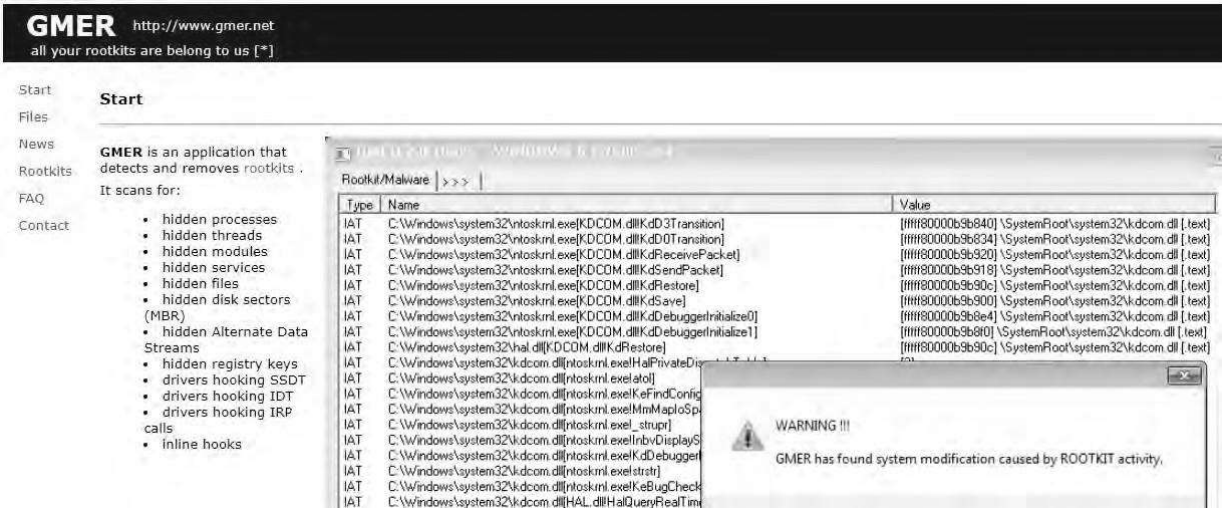
- rkhunter (Rootkit Hunter) is a Unix-based tool that scans for rootkits, backdoors and possible localexploits.
- It does this by comparing SHA-1 hashes of important files with knowngood ones in online databases, searching for default directories (of rootkits), wrong permissions, hidden files, suspicious strings in kernel modules, and special tests for Linux andFreeBSD.
- rkhunteris notable due to its inclusion in popular operating systems (Fedora, Debian,etc.)
- The tool has been written in Bourne shell, to allow for portability. It canrun on almost all UNIX-derivedsystems.

Gmer Rootkit Tool:

- GMER is a software tool written by a Polish researcherPrzemysławGmerek, for detecting and removingrootkits.
- It runs on Microsoft Windows and has support for Windows NT, 2000, XP, Vista, 7, 8 and 10. With version 2.0.18327 full support for Windows x64 is added.

ALOGRITHM DESCRIPTION:

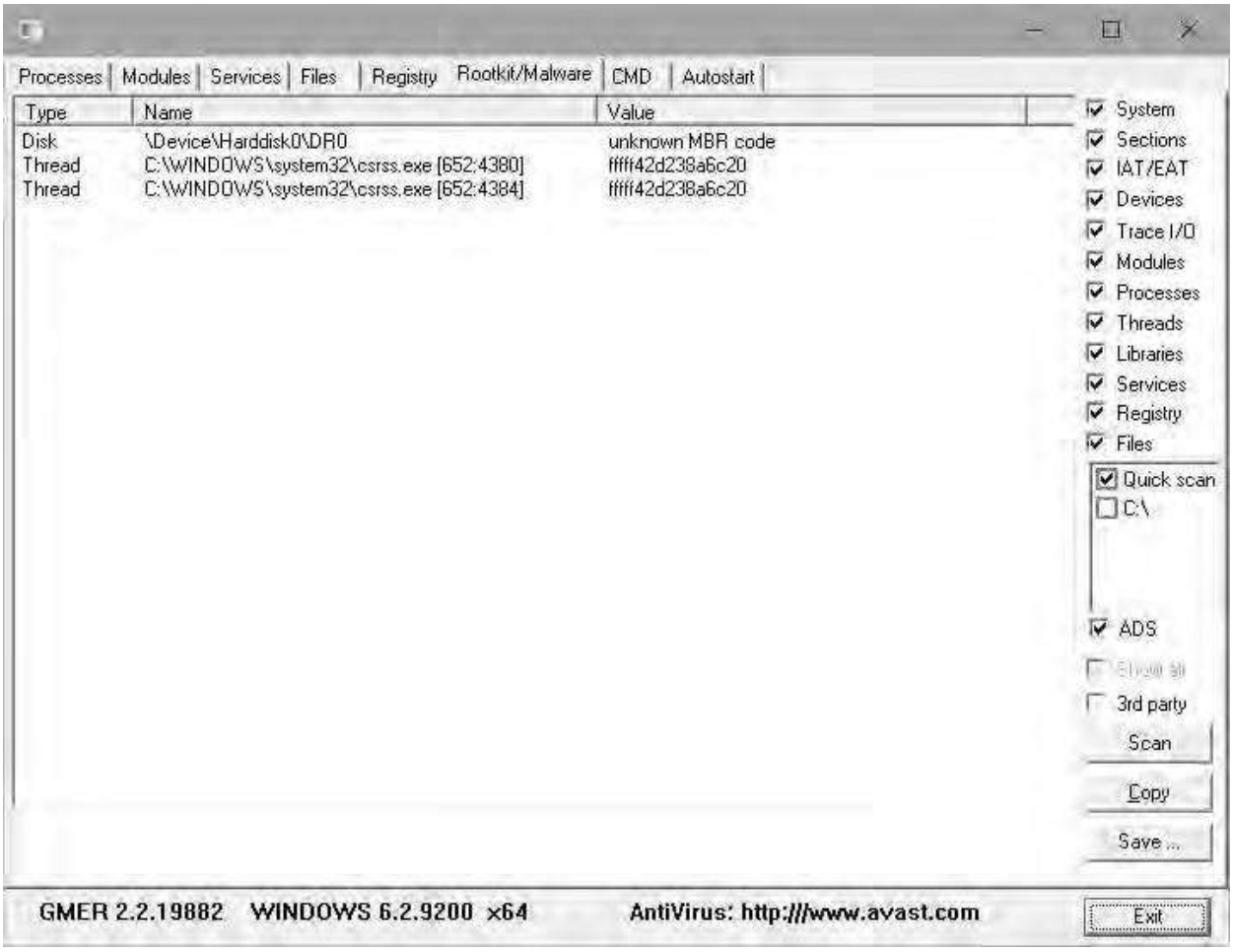
Step 1:



Visit GMER's website (see Resources) and download the GMER executable.

Click the "Download EXE" button to download the program with a random file name, as some rootkits will close “gmer.exe” before you can open it.

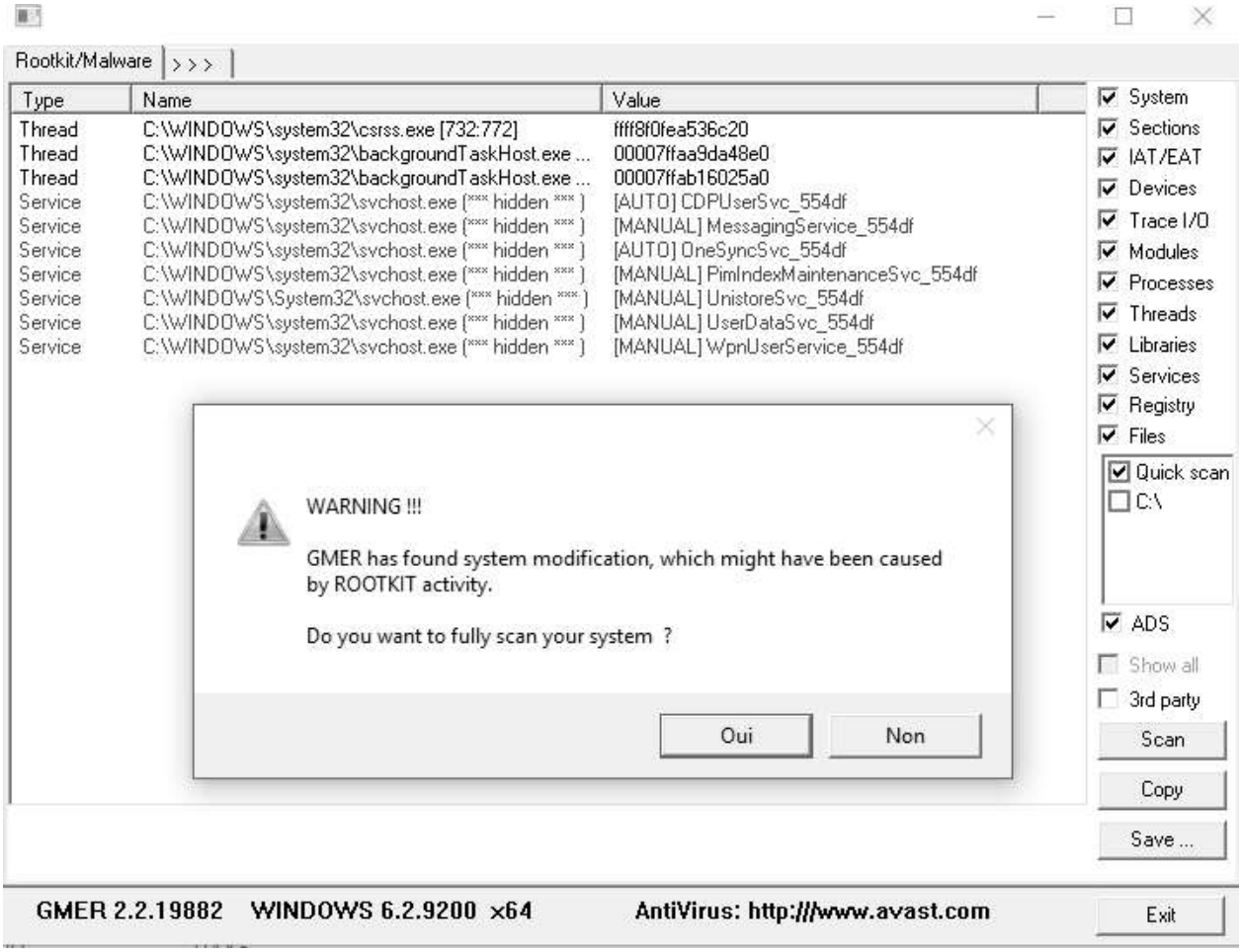
Step 2:



Double-click the icon for the program.

Click the "Scan" button in the lower-right corner of the dialog box. Allow the program to scan your entire hard drive.

Step 3:



When the program completes its scan, select any program or file listed in red. Right-click it and select "Delete."

If the red item is a service, it may be protected. Right-click the service and select "Disable." Reboot your computer and run the scan again, this time selecting "Delete" when that service is detected.
When your computer is free of Rootkits

INFERENCE:

RESULT:

EX NO: 12	MONOALPHABETIC CIPHER
DATE:	

AIM:

To implement the MONOALPHABETIC CIPHER in Java.

ALGORITHM:

Step 1: Declare the class and required variables.

Step 2: Implement the Monoalphabetic cipher in Java.

Step 3: Create the object for the class in the main program.

Step 4: Access the member functions using the objects.

Step 5: Execute the program.

PROGRAM:

```
package com.sanfoundry.setandstring;
import java.util.Scanner;
public class MonoalphabeticCipher
{
public static char p[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's',
't', 'u', 'v', 'w', 'x', 'y', 'z' };
public static char ch[] = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O',
'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L',
'Z', 'X', 'C', 'V', 'B', 'N', 'M' };
public static String doEncryption(String s)
{
```

```
 char c[] = new char[(s.length())];
for (int i = 0; i < s.length(); i++)
{
 for (int j = 0; j < 26; j++)
 {
if (p[j] == s.charAt(i))
{
c[i] = ch[j];
break;
}
 }
 }
 return (new String(c));
}
public static String doDecryption(String s)
{
 char p1[] = new char[(s.length())];
 for (int i = 0; i < s.length(); i++)
{
for (int j = 0; j < 26; j++)
 {
 if (ch[j] == s.charAt(i))
{
 p1[i] = p[j];
 break;
}
```

```
 }
}
 return (new String(p1));
}
public static void main(String args[])
{
 Scanner sc = new Scanner(System.in);
 System.out.println("Enter the message: ");
 String en = doEncryption(sc.next().toLowerCase());
 System.out.println("Encrypted message: " + en);
 System.out.println("Decrypted message: " + doDecryption(en));
 sc.close();
}
```

OUTPUT:

```
Enter the message:
helloworld
Encrypted message: ITSSGVGKSR
Decrypted message: helloworld
```

INFERENCE:

RESULT:

EX NO: 13	TRIPLE DES
DATE:	

AIM:

To implement the TRIPLE DES in Java.

ALGORITHM:

Step 1: Start the program.

Step 2: Declare the class and required variables.

Step 3: Declare the constructor and destructor of the class.

Step 4: Implement the TRIPLE DES in Java.

Step 5: Access the member functions using the objects.

PROGRAM:

```
import java.security.MessageDigest;
import java.util.Arrays;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class TripleDES {

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

```
    String text = "";
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the message to encrypt: ");
          text = sc.nextLine();

    byte[] codedtext = new TripleDES().encrypt(text);
    String decodedtext = new TripleDES().decrypt(codedtext);

    System.out.println(codedtext); // this is a byte array, you'll just see a reference to an array
    System.out.println(decodedtext); // This correctly shows "kyle boon"
}

public byte[] encrypt(String message) throws Exception {
    final MessageDigest md = MessageDigest.getInstance("md5");
    final byte[] digestOfPassword = md.digest("HG58YZ3CR9"
          .getBytes("utf-8"));
    final byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);
    for (int j = 0, k = 16; j < 8;) {
       keyBytes[k++] = keyBytes[j++];
    }

    final SecretKey key = new SecretKeySpec(keyBytes, "DESede");
    final IvParameterSpec iv = new IvParameterSpec(new byte[8]);
    final Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, key, iv);
```

```
    final byte[] plainTextBytes = message.getBytes("utf-8");
    final byte[] cipherText = cipher.doFinal(plainTextBytes);
    // final String encodedCipherText = new sun.misc.BASE64Encoder()
    // .encode(cipherText);

    return cipherText;
}

public String decrypt(byte[] message) throws Exception {
    final MessageDigest md = MessageDigest.getInstance("md5");
    final byte[] digestOfPassword = md.digest("HG58YZ3CR9"
            .getBytes("utf-8"));
    final byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);
    for (int j = 0, k = 16; j < 8;) {
        keyBytes[k++] = keyBytes[j++];
    }

    final SecretKey key = new SecretKeySpec(keyBytes, "DESede");
    final IvParameterSpec iv = new IvParameterSpec(new byte[8]);
    final Cipher decipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
    decipher.init(Cipher.DECRYPT_MODE, key, iv);

    // final byte[] encData = new
    // sun.misc.BASE64Decoder().decodeBuffer(message);
    final byte[] plainText = decipher.doFinal(message);
```

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

OUTPUT:

```
Enter the message to encrypt: hello world
[B@433c675d
hello world
```

INFERENCE:

RESULT: