HINDUSTHAN

COLLEGE OF ENGINEERING AND TECHNOLOGY
COIMBATORE – 641 032

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

16CS7001 – CRYPTOGRAPHY AND NETWORK SECURITY LAB

NAME OF THE STUDENT : _____

REGISTER NUMBER : _____

BRANCH : _____

YEAR / SEMESTER : _____

HINDUSTHAN

COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE – 641 032

(AN AUTONOMOUS INSTITUTION)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certified that this is the bonafide record of work done

By

.....

in the

16CS7001 – Cryptography And Network Security Lab

of this institution, as prescribed by the college for the ***VII Semester***

during the Academic year ***2019 – 2020.***

Place: Coimbatore

Date:

Faculty in-Charge **Head of the Department**

Register Number: .....

Submitted for the End Semester Practical Examination conducted on .....

Internal Examiner **External Examiner**

CONTENTS

FACULTY INCHARGE

Ex. No.: 1 - a	IMPLEMENTATION OF CAESAR CIPHER
Date:	

AIM:

To write a Java Program for implementing Caesar Cipher.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class Caesar Cipher.

Step 2 : Declare a string ALPHABET.

Step 4 : Define a function encrypt () to produce a cipher text and decrypt () to reproduce the plain text.

Step 5 : Define a main (), get the string and call encrypt () to encrypt the string and decrypt () to reproduce the plain text and display it.

Step 6 : Stop the program.

PROGRAM:

```
package javaapplication1;
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 + char Position) % 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.charA t(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)
{
try (Scanner sc = new Scanner(System.in))
{
System.out.println("Enter the String for Encryption: ");
String message = new String();
message = sc.next();
System.out.println("Encrypted Text is:");
System.out.println(encrypt(message, 3));
System.out.println("Decrypted Text is:");
System.out.println(decrypt(encrypt( message, 3), 3));
```

```
}
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of Caesar cipher was executed and output is verified successfully.

Ex. No.: 1 - b	IMPLEMENTATION OF PLAYFAIR CIPHER
Date:	

AIM:

To write a Java program for implementing Playfair Cipher.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class Basic to find the index of a char.

Step 3 : Define a class PlayFair to define the key matrix, find the row position, column position, encrypt the text and decrypt the text.

Step 4 : Define a class PlayFair Cipher, to get the plain text and then to encrypt and decrypt the text.

Step 5 : Display the encrypted text and decrypted text.

Step 6 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.util.*;
class Basic
{
String allChar="ABCDEFGHIJKLM NOPQRSTUVWXYZ”;
boolean indexOfChar(char c)
{
for(int i=0;i < allChar.length();i++)
{
if(allChar.charA t(i)==c)
return true;
}
return false;
}
}
```

```
class PlayFair
{
Basic b=new Basic();
char keyMatrix[][]=new char[5][5];
boolean repeat(char c)
{
if(!b.indexOfChar(c))
{
return true;
}
for(int i=0;i < keyMatrix.length;i++)
{
for(int j=0;j < keyMatrix[i].length;j++)
{
if(keyMatrix[i][j]==c || c=='J')
return true;
}
}
return false;
}
void insertKey(String key)
{
key= key.toUpperCase();
key= key.replaceAll("J", "I");
key= key.replaceAll(" ", "");
int a=0,b= 0;
for(int k= 0;k < key.length();k++)
{
if(!repeat(key.charA t(k)))
{
keyMatrix[a][b++]=key.charA t(k);
```

```
if(b> 4)
{
b= 0;
a++;
}
}
}
char p= 'A';
while(a < 5)
{
while(b < 5)
{
if(!repeat(p))
{
keyMatrix[a][b++]=p;
}
p+ +;
}
b= 0;
a++;
}
System.out.print(" -------------------------Key Matrix--------------------");
for(int i=0;i < 5;i++)
{
System.out.println();
for(int j=0;j < 5;j++)
System.out.print(" \t"+ keyMatrix[i][j]);
}
System.out.println(" \n-----------------------------------------------------------");
}
int rowPos(char c)
```

```
{
for(int i=0;i < keyMatrix.length;i++)
{
for(int j=0;j < keyMatrix[i].length;j++)
{
if(keyMatrix[i][j]==c)
return i;
}
}
return -1;
}
int columnPos(char c)
{
for(int i=0 ;i < keyMatrix.length;i++)
{
for(int j=0;j < keyMatrix[i].length;j++)
{
if(keyMatrix[i][j]==c)
return j;
}
}
return -1;
}
String encryptChar(String plain)
{
plain=plain.toUpperCase();
char a=plain.charAt(0),b= plain.charA t(1);
String cipherChar="";
int r1,c 1,r2,c2;
r1=rowPos(a);
c1=columnPos(a);
```

```
r2=rowPos(b);
c2=columnPos(b);
if(c1==c2)
{
++r1;
++r2;
if(r1>4)
r1=0;
if(r2>4)
r2=0;
cipherChar+=keyMatrix[r1][c2];
cipherChar+=keyMatrix[r2][c1];
}
else if(r1= =r2)
{
++c1;
++c2;
if(c1>4)
c1=0;
if(c2>4)
c2=0;
cipher Char+=keyMatrix[r1][c1];
cipherChar+=keyMatrix[r2][c2];
}
else
{
cipherChar+=keyMatrix[r1][c2];
cipherChar+=keyMatrix[r2][c1];
}
return cipherChar;
}
```

```
String Encrypt(String plainText,String key)
{
insertKey(key);
String cipherText="";
plainText=plainText.replaceAll("j", "i");
plainText=plainText.replaceAll(" ", "");
plainText=plainText.toUpperCase();
int len=plainText.length();
if(len/2!=0)
{
plainText+="X";
++len;
}
for(int i=0;i < len -1;i=i+2)
{
cipherText+=encryptChar(plainText.substring(i,i+2));
cipherTe xt+=" ";
}
return cipherText;
}
String decryptChar(String cipher)
{
cipher=cipher.toUpperCase();
char a=cipher.charAt(0),b= cipher.charAt(1);
String plainChar="";
int r1,c1,r2,c2;
r1=rowPos(a);
c1=columnPos(a);
r2=rowPos(b);
c2=columnPos(b);
if(c1==c2)
```

```
{
--r1;
--r2;
if(r1 < 0)
r1=4;
if(r2 < 0)
r2=4;
plainChar+=keyMatrix[r1][c2];
plainChar+=keyMatrix[r2][c1];
}
else if(r1==r2)
{
--c1;
--c2;
if(c1 < 0)
c1=4;
if(c2 < 0)
c2=4;
plainChar+=keyMatrix[r1][c1];
plainChar+=keyMatrix[r2][c2];
}
else
{
plainChar+ =keyMatrix[r1][c2];
plainChar+=keyMatrix[r2][c1];
}
return plainChar;
}
String Decrypt(String cipherText,String key)
{
String plainText="";
```

```
cipherText=cipherText.replaceAll("j", "i");
cipherText=cipherText.replaceAll(" ", "");
cipherText=cipherText.toUpperCase();
int len=cipherText.length();
for(int i=0;i < len -1;i=i+2)
{
plainText+=decryptChar(cipherText.substring(i,i+2));
plainText+=" ";
}
return plainText;
}
}
class PlayfairCipher
{
public static void main(String args[])throws Exception
{
PlayFair p= new PlayFair();
Scanner scn=new Scanner(System.in);
String key,cipherText,plainText;
System.out.println("Enter plaintext:");
plainText=scn.nextLine();
System.out.println("Enter Key:");
key= scn.nextLine();
cipherText=p.Encrypt(plainText,key);
System.out.println("Encrypted text:");
System.out.println(" ----------\n"+cipherText);
System.out.println(" ----------");
String encryptedText=p.Decrypt(cipherText, key);
System.out.println("Decrypted text:" );
System.out.println(" ----------\n"+encryptedText);
System.out.println(" ----------");
```

```
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of PlayFair Cipher was executed and output is verified successfully.

Ex. No.: 2 - a	IMPLEMENTATION OF HILL CIPHER
Date:	

AIM:

To write a Java Program for implementing Hill Cipher.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class HillCipher, in that declare 2 array one for key, other for inverse key and define a string key.

Step 2 : In this class, define a main (), get the choice to encrypt or decrypt.

Step 4 : Based on the choice call encrypt () and decrypt () to find cipher and plain text.

Step 5 : Display the result.

Step 6 : Stop the program.

PROGRAM:

```
package javaapplication1;
import javax.swing.JOptionPane;
public class HillCipher
{
//the 3x3 key matrix for 3 characters at on ce
public static int[][] keymat = new int[][]
{
{1, 2, 1},
{2, 3, 2},
{2, 2, 1},
};
public static int[][] invkeymat = new int[][]
{
{-1, 0, 1},
{2, -1, 0},
{-2, 2, -1},
```

```
};
public static String key = "ABCDEFGHIJKLM NOPQRSTUVWXYZ" ;
public static void main(String[] args)
{ // TOD O code application logic here
String text,outtext ="";
int ch, n;
ch = Integer.parseInt(JOptionP ane.showInputDialog(null, "Enter 1 to Encrypt and 2 to Decrypt!"));
text = JOptionPane.showInputDialog(null, "Enter plain/cipher text to encrypt?");
text = text.toUpperCase();
text = text.replaceAll(" \\s",""); //removing spaces
n = text.length() % 3;
if(n!=0)
{
for(int i = 1; i<= (3 -n);i++)
{
text+= 'X ';
}
}
System.out.println("Padded Text:" + text);
char[] ptextchars = text.toCharArray();
switch(ch)
{
case 1:
for(int i=0;i< text.length(); i+=3)
{
outtext += encrypt(ptextchars[i],ptextchars[i+1],ptextchars[i+2]);
}
break;
case 2:
for(int i=0;i< text.length(); i+=3)
```

```
{
outtext += decrypt(ptextchars[i],ptextchars[i+1],ptextchars[i+2]);
}
break;
default: System.out.println("Invalid Choice!");
}
System.out.println("Output: " + outtext);
}
private static String encrypt(char a, char b, char c)
{
Strin g ret = "";
int x,y, z;
int posa = (int)a - 65;
int posb = (int)b - 65;
int posc = (int)c - 65;
x = posa * keymat[0][0] + posb * keymat[1][0] + posc * keymat[2][0];
y = posa * keymat[0][1] + posb * keymat[1][1] + posc * keymat[2][1];
z = posa * keymat[0][2 ] + posb * keymat[1][2] + posc * keymat[2][2];
a = key.charA t(x%26);
b = key.charA t(y%26);
c = key.charA t(z%26);
ret = "" + a + b + c;
return ret;
}
private static String decrypt(char a, char b, char c)
{
String ret = "";
int x,y,z;
int posa = (int)a - 65;
int posb = (int)b - 65;
int posc = (int)c - 65;
```

```
x = posa * invkeymat[0][0]+ posb * invkeymat[1][0] + posc * invkeymat[2][0];
y = posa * invkeymat[0][1]+ posb * invkeymat[1][1] + posc * invkeymat[2][1];
z = posa * invkeymat[0][2]+ posb * invkeymat[1][2] + posc * invkeymat[2][2];
a = key.charA t((x% 26<0)?(26+ x% 26):(x% 26));
b = key.charA t((y% 26<0)?(26+ y% 26):(y% 26));
c = key.charA t((z%26<0)?(26+ z%26):(z%26));
ret = "" + a + b + c;
return ret;
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of Hill Cipher was executed and output is verified successfully.

Ex. No.: 2 - b	IMPLEMENTATION OF VIGENERE CIPHER
Date:	

AIM:

To write a Java Program for implementing Vigenere Cipher.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class VC1, in that define encipher () to produce a cipher text.

Step 2 : Define decipher () to reproduce the plain text.

Step 4 : Define a shift () to shift the values.

Step 5 : In main (), define the text and key values and call the encipher () to encrypt decipher () to decrypt the encrypted text.

Step 6 : Display the results.

Step 6 : Stop the program.

PROGRAM:

```
package javaapplication1;
public class VC1
{
public static String encipher(String s, String key)
{
StringBuilder builder = new StringBuilder();
for(int i = 0; i < s.length(); i ++)
{
if(s.charAt(i) < 65 || s.charAt(i) > 90)
{ //ASCII character (capital letter)
throw new IllegalArgumentException("" + "Open text must contain only capital letters");
}
//add shift mod ularly
char encyphered = s.charAt(i) + getShift(key, i) > 90 ? (char)((s.charAt(i) + getShift(key, i)) - 26)
: (char)(s.charAt(i) + getShift(key, i));
```

```
builder.append(encyphered);
}
return builder.toString();
}
public static String decipher(String s, String key)
{
StringBuilder builder = new StringBuilder();
for(int i = 0; i < s.length(); i ++)
{
if(s.charA t(i) < 65 || s.charA t(i) > 90)
{ //ASCII character (capital letter)
throw new IllegalArgumentException("" +"Ciphertext must contain only capital letters");
}
//subtract shift mod ularly
char decyphered = s.charAt(i) - getShift(key, i) < 65 ? (char)((s.charAt(i) - getShift(key, i)) + 26)
: (char)(s.charAt(i) - getShift(key, i));
builder.append(decyphered);
}
return builder.toString();
}
private static int getShift(String key, int i)
{
if(key.charAt(i % key.length()) < 65 || key.charA t(i % key.length()) > 90)
{
throw new IllegalArgumentException("" +"Key phrase must contain only capital letters");
}
return ((int)key.charAt(i % key.length())) - 65;
}
public static void main(String[] args)
{
String text = “HELLO";
```

```
String key = "CAT";
String enciphered = encipher (text, key);
System.out.println("IMPLEMENTATION OF VIGENERE CIPHER");
System.out.println("CIPHER TEXT IS:"+ enciphered);
System.out.println("DECIPHERED TEXT IS:"+ decipher(enciphered, key));
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of Vigenere Cipher was executed and output is verified successfully.

Ex. No.: 2 - c	IMPLEMENTATION OF RAIL FENCE ROW & COLUMN TRANSFORMATION
Date:	

AIM:

To write a Java Program for implementing Rail Fence Row & Column Transformation.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class rail fence basic, in that define Encryption () to produce cipher text.

Step 3 : Define decryption () to produce decipher text.

Step 4 : Define a class rail fence cipher, in that define main () and input the text to cipher and decipher it.

Step 5 : Display the encrypted text and decrypted text.

Step 6 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.util.*;
class RailFenceBasic
{
int depth;
String Encryption(String plainText,int depth)throw s 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)throw s 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 railfencecipher
{
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"+decryptedTex t);
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of Rail Fence Row & Column Transformation was executed and output is verified successfully.

Ex. No.: 3	IMPLEMENTATION OF DES
Date:	

AIM:

To write a Java Program for implementing DES.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class DES, in that define a constructor to display the encrypted and decrypted message.

Step 3 : Define generate Symmetric Key (), to generate the key.

Step 4 : Define encrypt () to encrypt the plain text.

Step 5 : Define decrypt () to decrypt the ciphered text.

Step 6 : Define main () to declare object for the class.

Step 7 : Stop the program.

PROGRAM:

```
package javaapplication1;
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import jav ax.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 "+encryptedD ata);
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();
int num = 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) throw s Exception
{
KeyGenerator kgen = KeyGenerator.getInstance("DES");
SecureRandom sr = SecureRandom.getInstance("SHA1P RNG");
sr.setSeed(seed);
kgen.init(56, sr);
SecretKey skey = kgen.generateKey();
raw = skey.getEncoded();
return raw;
}
private static byte[] encrypt(byte[] ra w, byte[] clear) throws Exception
{
SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher.init(Cipher.EN CRYPT_MODE, skeySpec);
byte[] encrypted = cipher.doFinal(clear);
return encrypted;
}
private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception
{
SecretKeySpec skeySpec = new SecretKeySpec(raw, "DES");
Cipher cipher = Cipher.getInstance("DES");
cipher .init(Cipher.DECRYPT_MODE, skeyS pec);
byte[] decrypted = cipher.doFinal(encrypted);
return decrypted;
```

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

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of DES was executed and output is verified successfully.

Ex. No.: 4	IMPLEMENTATION OF RSA
Date:	

AIM:

To write a Java Program for implementing RSA.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a default constructor RSA to compare 2 prime numbers.

Step 3 : Define a parameterized constructor to assign values.

Step 4 : Define bytetostring() to convert byte to string.

Step 5 : Define encrypt () to encrypt the text and decrypt () to reproduce the plain text.

Step 6 : Define main () to call the encrypt () and decrypt () to perform respective operations and display the results.

Step 7 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA
{
private BigInteger p;
private BigInteger q;
private BigInteger N;
private BigInteger phi;
private BigInteger e;
private BigInteger d;
private int bitlength = 1024;
private Random r;
```

```
public RSA()
{
r = new Random();
p = BigInteger.probablePrime(bitlength, r);
q = BigInteger.probablePrime(bitlength, r);
N = p.multiply(q);
phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
e = BigInteger.probablePrime(bitlength / 2, r);
while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
{
e.add(BigInteger.ONE);
}
d = e.modInverse(phi);
}
public RSA(BigInteger e, BigInteger d, BigInteger N)
{
this.e = e;
this.d = d;
this.N = N ;
}
@SuppressWarnings("deprecation")
public static void main(String[] args) throw s IOException
{
RSA rsa = new RSA();
DataInputStream in = new DataInputStream(System.in);
String teststring;
System.out.println("Enter the plain text:");
teststring = in.readLine();
System.out.println("Encrypting String: " + teststring);
System.out.println("String in Bytes: " + bytesToString(teststring.getBytes()));
// encrypt
```

```
byte[] encrypted = rsa.encrypt(teststring.getBytes());
// decrypt
byte[] decrypted = rsa.decrypt(encrypted);
System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
System.out.println("Decrypted String: " + new String(decrypted));
}
private static String bytesToString(byte[] encrypted)
{
String test = "";
for (byte b : encrypted)
{
test += Byte.toString(b);
}
return test;
}
// Encrypt message
public byte[] encrypt(byte[] message)
{
return (new BigInteger (message)).modPow (e, N ).toByteArray();
}
// Decrypt message
public byte[] decrypt(byte[] message)
{
return (new BigInteger(message)).modPow (d, N ).toByteArray();
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of RSA was executed and output is verified successfully.

Ex. No.: 5	IMPLEMENTATION OF DIFFIEE - HELLMAN
Date:	

AIM:

To write a Java Program for implementing Diffiee - Hellman.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define class Diffe Hellman Big Int, in that main () get the details and pass the se key.

Step 3 : Calculate the keys and display it.

Step 4 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.util.*;
import java.math.BigInteger;
public class DiffeHellmanBigInt
{
final static BigInteger one = new BigInteger("1");
public static void main(String args[])
{
Scanner stdin = new Scanner(System.in);
BigInteger p;
// Get a start spot to pick a prime from the user.
System.out.println("Enter the approximate value of p you want.");
String ans = stdin.next();
p = getNextPrime(ans);
System.out.println("Your prime is "+p+ ".");
// Get the base for exponentiation from the user.
System.out.println("Now, enter a number in between 2 and p -1.");
Big Integer g = new BigInteger(stdin.next());
```

```
// Get A's secret number.
System.out.println("Person A : enter your secret number now.");
BigInteger a = new BigInteger(stdin.next());
// Make A 's calculation .
BigInteger resulta = g.modPow (a,p);
// This is the value that will get sent from A to B.
// This value does NOT compromise the value of a easily.
System.out.println("Person A sends to person B "+resulta+ ".");
// Get B's secret number.
System.out.println("Person B: enter your secret number now .");
BigInteger b = new BigInteger(stdin.next());
// Make B's calculation.
BigInteger resultb = g.modPow (b,p);
// This is the value that will get sent from B to A.
// This value does NOT compromise the value of b easily.
System.out.println("Person B sends to person A "+resultb+".");
// Once A and B receive their values, they make their new calculation s.
// This involved getting their new numbers and raising them to the
// same power as before, their secret number.
BigInteger KeyACalculates = resultb.modPow (a,p);
BigInteger KeyBCalculates = resulta.modPow (b,p);
// Print out the Key A calculates.
System.out.println("A takes "+resultb+" raises it to the power "+a+" mod "+p);
System.out.println("The Key A calculates is "+KeyACalculates+".");
// Print out the Key B calculates.
System.out.println("B takes "+resulta+ " raises it to the power "+b+ " mod "+p);
System.out.println("The Key B calculates is "+KeyBCalculates+".");
}
public static BigInteger getNextPrime(String ans)
{
BigInteger test = new BigInteger(ans);
```

```
while (!test.isProbablePrime(99))
test = test.add(one);
return test;
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of Diffiee Hellman was executed and output is verified successfully.

Ex. No.: 6 - a	IMPLEMENTATION OF DES
Date:	

AIM:

To write a Java Program for implementing MD5.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class Java MD5 Hash, in that define a function MD5 to find the hash values of three different inputs.

Step 3 : Display the hash values.

Step 4 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchSAlgorithmException;
public class Java MD5 Hash
{
public static void main(String[] args)
{
System.out.println("For null " + md5(""));
System.out.println("For simple text "+ md5("This is my text"));
System.out.println("For simple numbers " + md5("12345"));
}
public static String md5(String input)
{
String md5 = null;
if (null == input) return null;
try
{
```

```
//Create MessageDigest object for M D5
MessageDigest digest = MessageDigest.getInstance("MD5");
//Update input string in message digest
digest.update(input.getBytes(), 0, input.length());
//Converts message digest value in base 16 (hex)
md5 = new BigInteger(1, digest.digest()).toString(16);
}
catch (NoSuchAlgorithmException e)
{
e.printStackTrace();
}
return md5;
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of MD5 was executed and output is verified successfully.

Ex. No.: 6 - b	IMPLEMENTATION OF SHA -1
Date:	

AIM:

To write a Java Program for implementing SHA-1.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class HashTextTest, in that main () call sha1 () to display secured hash value.

Step 3 : Define sha1 (), in that define the instances and generate the hash value.

Step 4 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class HashTextTest
{
public static void main(String[] args) throw s NoSuchAlgorithmException
{
System.out.println("Shared H ash Key Value is:"+ sha1("shared Hashing"));
}
static String sha1(String input) throws NoSuchAlgorithmException
{
MessageDigest mDigest = MessageDigest.getInstance("SHA1");
byte[] result = mDigest.digest(input.getBytes());
StringBuffer sb = new StringBuffer();
for (int i = 0; i < result.length; i++)
{
sb.append(Integer.toString((result[i] & 0xff) + 0x100, 16).substring(1));
}
```

```
return sb.toString();
}
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of SHA was executed and output is verified successfully.

Ex. No.: 7	IMPLEMENTATION OF SIGN ATURE SCHEME DIGITAL SIGNATURE STANDARD
Date:	

AIM:

To implement Signature Scheme of Digital Signature Standard using java.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Define a class DSS, in that define a main().

Step 3 : Create instance for Keypair Generator class.

Step 4 : Generate Key Pair using KeyPair Class.

Step 5 : Send the data to encrypt and sign.

Step 6 : Sign the data using Signature class.

Step 7 : Display the signature and verification status.

Step 8 : Stop the program.

PROGRAM:

```
package javaapplication1;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.Signature;
import sun.misc.BASE64Encoder;
public class DSS
{
public static void main(String[] args) throw s Exception
{
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(1024);
KeyPair keyPair = kpg.genKeyPair();
byte[] data = "test".getBytes("UTF8");
Signature sig = Signature.getInstance("MD5WithRSA");
sig.initSign(keyPair.getPrivate());
```

```
sig.update(data);
byte[] signatureBytes = sig.sign();
System.out.println("Singature:" + new BASE64Encoder().encode(signatureBytes));
sig.initVerify(keyPair.getPublic());
sig.update(data);
System.out.println(sig.verify(signatureBytes));
}
```

OUTPUT:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, java program for implementation of Signature Scheme of Digital Signature Standard was executed and output is verified successfully.

Ex. No.: 8	CREATION OF DIGITAL SIGNATURE, SECURE DATA STORAGE, SECURE DATA TRANSMISSION USING GNUPG
Date:	

AIM:

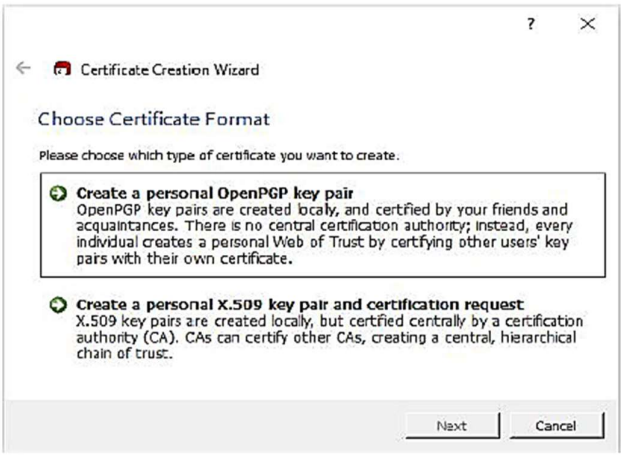
To create Digit al Signature, secure Data Storage & transmission using GnuPG.

PROCEDURE:

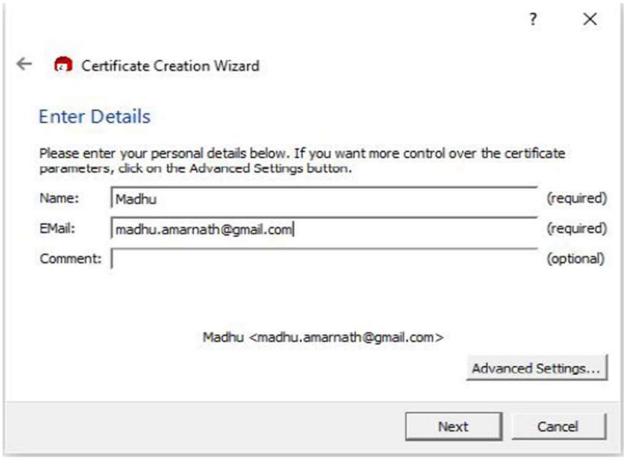
GENERATING KEYPAIR

Step 1: Open up Kleopatra.

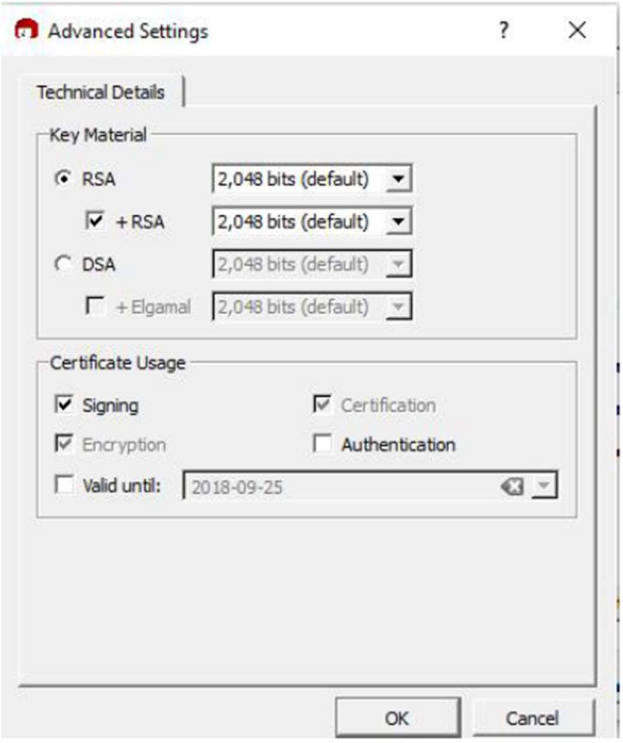
Step 2: Click File ->New Certificate -> A Certificate Creation Wizard appears as follows:



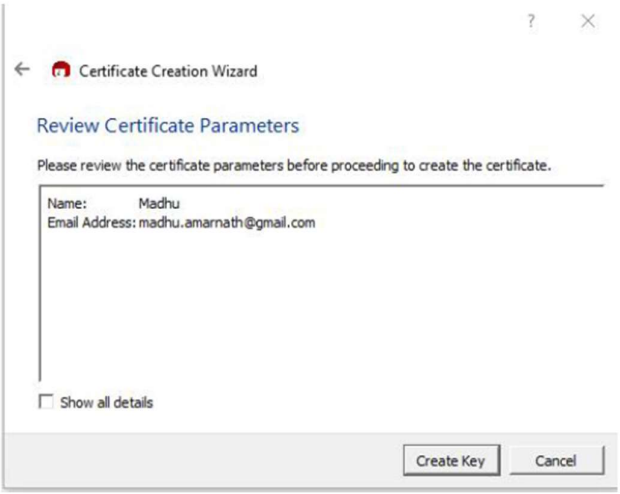
Step 3: Choose and Click Create a personal OpenPGP key pair. It will prompt to enter name, email and comments



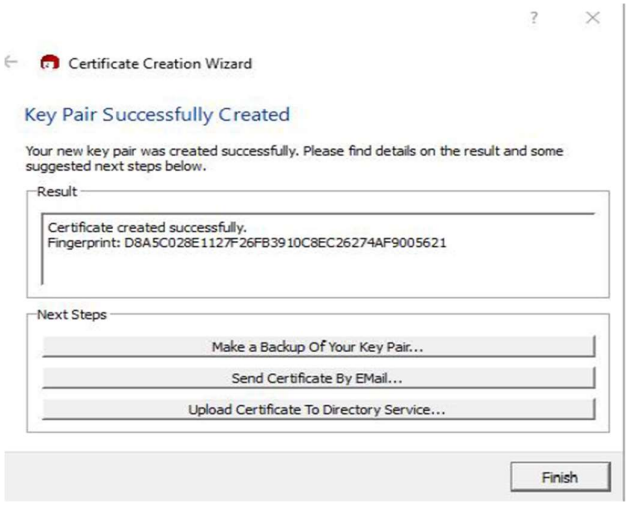
Step 4: Click Advanced Settings to choose appropriate key and key size. Click OK after choosing the key details.

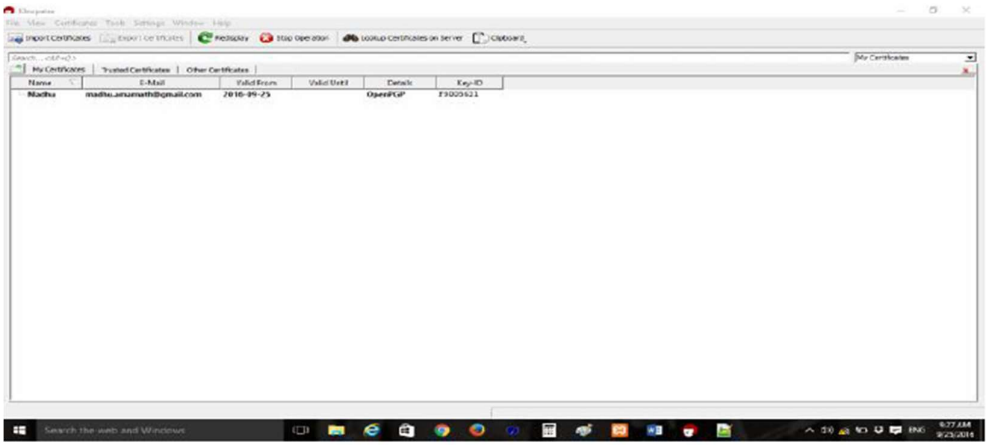


Step 5: Click Next in the Certificate Creation Wizard. It will prompt to review certificate parameters. Click Create Key button.



Step 6: The pinentry dialog box will appear and it will prompt to enter passphrase details. After entering passphrase (use alphanumeric characters), it will prompt to re-enter the passphrase. After re-entering, Key Pair Successfully Created message appears. Click Finish

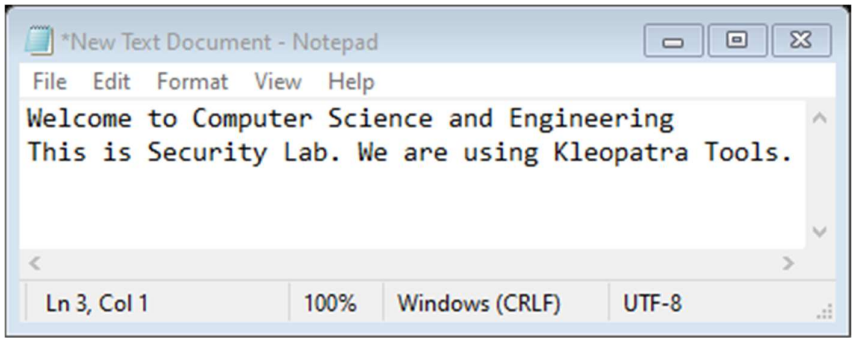




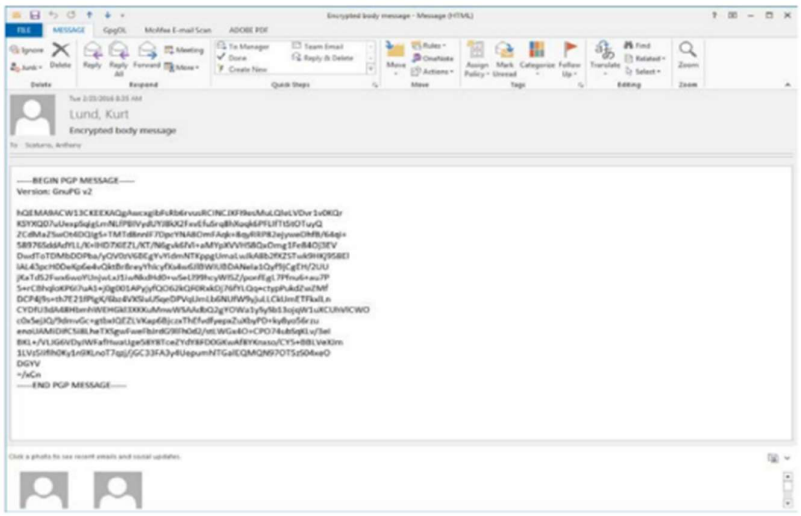
ENCRYPTING MESSAGE:

Step 1: Create another certificate.

Step 2: Create a text file containing plaintext message.



Step 3: Click File ->Sign/Encrypt Files and choose the text file containing plaintext message.



Step 4: Make sure that Encrypt radio button is checked and Click Next.

Step 5: Choose receiver Certificate file details and click Add button. Details of selected file appears in the box below. Then click Encrypt.

Step 6: Encryption succeeded message appears. Then click Finish.

DECRYPTING MESSAGE

Step 1: Click File ->Decrypt/Verify Files and choose the file containing encrypted cipher Text.

Step 2: Click Decrypt/Verify button. Now the tool will prompt to enter receiver's passphrase. Enter the passphrase.

Step 3: Decryption succeeded message appears after saving the file in appropriate folder. Click OK.

Step 4: Now see the text file containing recovered plaintext message.

SIGNING A MESSAGE:

Step 1: Click File ->Sign/Encrypt Files and choose the text file containing plaintext message.

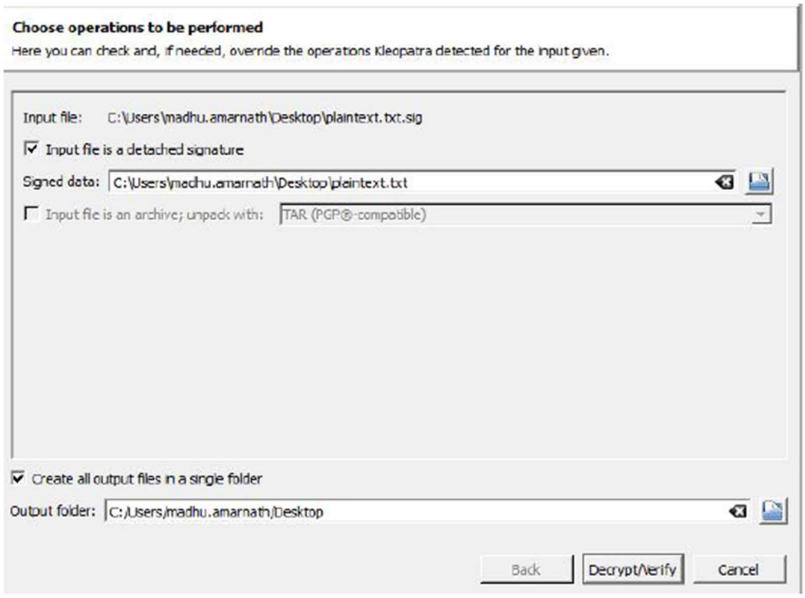
Step 2: Make sure that Sign radio button is checked and Click Next.

Step 3: Choose the mail id of sender under OpenPGP Signing Certificate. Then click Sign button.

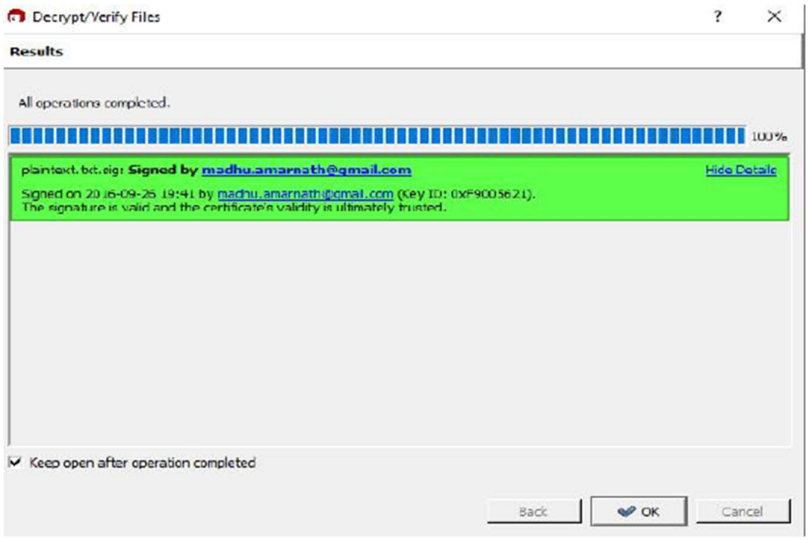
Step 4: Now the tool will prompt to enter passphrase of sender. Enter the passphrase. Now the message is signed.

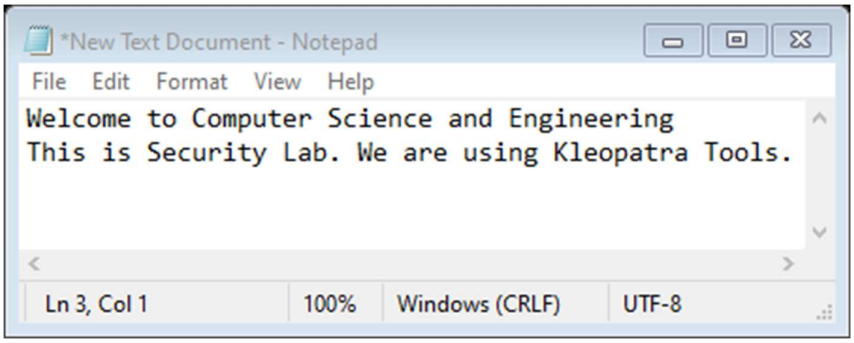
VERIFYING MESSAGE:

Step 1: Click File -> Decrypt/Verify File and choose the signature file.



Step 2: Click Decrypt/Verify button. The following window appears. Click Show Details.





HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus the secure data storage, secure data transmission and for creating digital signatures (GnuPG) was developed successfully.

Ex. No.: 9	WORKING WITH KF SENSOR TOOL FOR CREATING AND MONITORING HONEYPOT
Date:	

AIM:

Honey Pot is a device placed on Computer Network specifically designed to capture malicious network traffic. KF Sensor is the tool to setup as honeypot when KF Sensor is running it places a siren icon in the windows system tray in the bottom right of the screen. If there are no alerts then green icon is displayed.

INTRODUCTION:

HONEY POT:

A honeypot is a computer system that is set up to act as a decoy to lure cyber attackers, and to detect, deflect or study attempts to gain unauthorized access to information systems. Generally, it consists of a computer, applications, and data that simulate the behavior of a real system that appears to be part of a network but is actually isolated and closely monitored.

All communications with a honeypot are considered hostile, as there is no reason for legitimate users to access a honeypot. Viewing and logging this activity can provide an insight into the level and types of threat a network infrastructure faces while distracting attackers away from assets of real value. Honeypots can be classified based on their deployment (use/action) and based on their level of involvement. Based on deployment, honeypots may be classified as:

1. Production honeypots
2. Research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

Research honeypots are run to gather information about the motives and tactics of the Black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats.

KF SENSOR:

KFSensor is a Windows based honeypot Intrusion Detection System (IDS). It acts as a honeypot to attract and detect hackers and worms by simulating vulnerable system services and trojans. By acting as a decoy server it can divert attacks from critical systems and provide a higher level of information than can be achieved by using firewalls and NIDS alone. KFSensor is a system installed in a network in order to divert and study an attacker's behavior. This is a new technique that is very effective in detecting attacks.

The main feature of KFSensor is that every connection it receives is a suspect hence it results in very few false alerts. At the heart of KFSensor sits a powerful internet daemon service that is built to handle multiple ports and IP addresses. It is written to resist denial of service and buffer overflow attacks. Building on this flexibility KFSensor can respond to connections in a variety of ways, from simple port listening and basic services (such as echo), to complex simulations of standard system services. For the HTTP protocol KFSensor accurately simulates the way Microsoft's web server (IIS) responds to both valid and invalid requests. As well as being able to host a website it also handles complexities such as range requests and client side cache negotiations. This makes it extremely difficult for an attacker to fingerprint, or identify KFSensor as a honeypot.

PROCEDURE:

STEP 1 : Download KF Sensor Evaluation Setup File from KF Sensor Website.

STEP 2 : Install with License Agreement and appropriate directory path.

STEP 3 : Reboot the Computer now. The KF Sensor automatically starts during windows boot.

STEP 4 : Click Next to setup wizard.

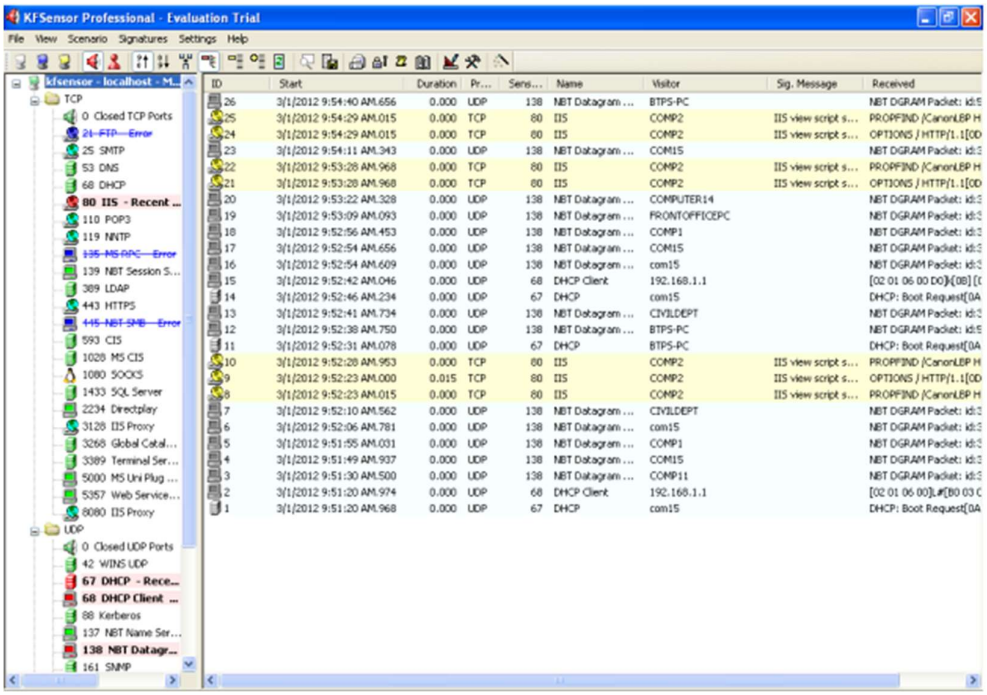
STEP 5 : Select all port classes to include and Click Next.

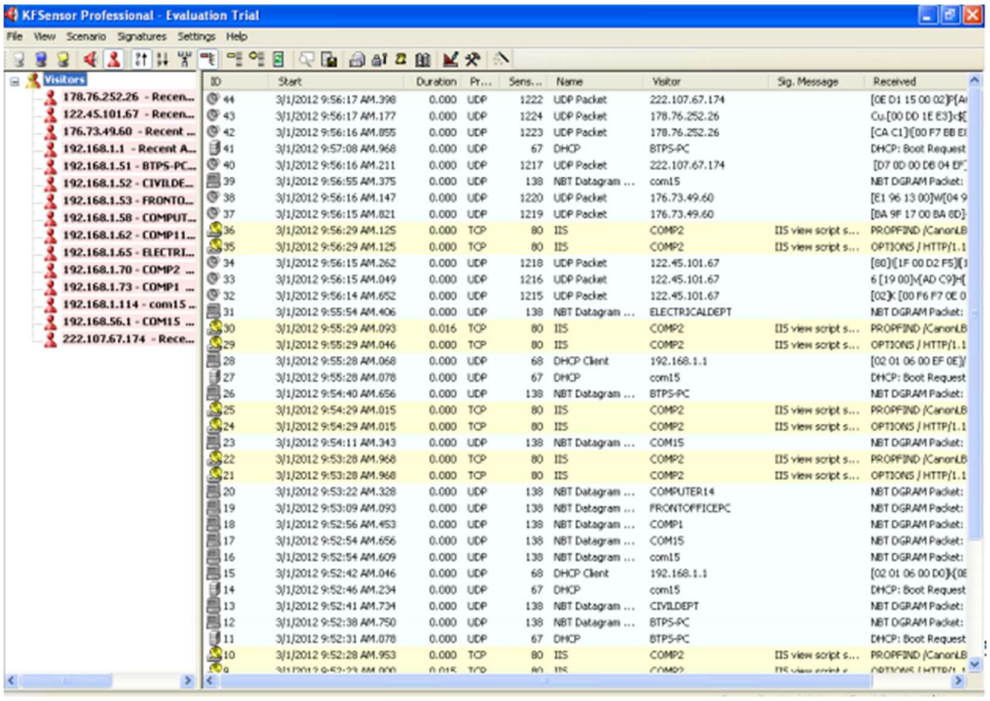
STEP 6 : "Send the email and Send from email", enter the ID and Click Next.

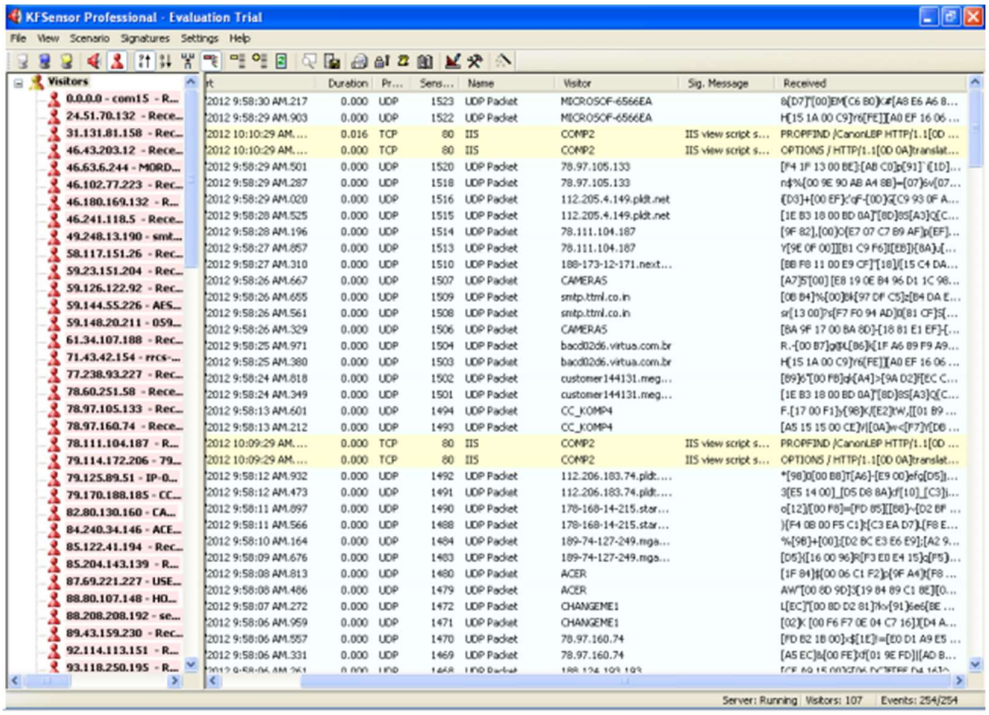
STEP 7 : Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next.

STEP 8 : Select Install as System service and Click Next.

STEP 9 : Click finish.


KFSensor Professional - Evaluation Trial
File View Scenario Signatures Settings Help
kfsensor - localhost - M...
TCP
0 Closed TCP Ports
21 FTP - Error
25 SMTP
53 DNS
68 DHCP
80 IIS - Recent ...
110 POP3
119 NNTP
135 MS RPC - Error
139 NBT Session S...
389 LDAP
443 HTTPS
445 NBT SMB - Error
593 CIS
1028 MS CIS
1080 SOCKS
1433 SQL Server
2234 Directplay
3128 IIS Proxy
3268 Global Catal...
3389 Terminal Ser...
5000 MS Uni Plug ...
5357 Web Service...
8080 IIS Proxy
UDP
0 Closed UDP Ports
42 WINS UDP
67 DHCP - Rece...
68 DHCP Client ...
88 Kerberos
137 NBT Name Ser...
138 NBT Datagr...
161 SNMP
ID Start Duration Pr... Sens... Name Visitor Sig. Message Received


KFSensor Professional - Evaluation Trial
File View Scenario Signatures Settings Help
Visitors
178.76.252.26 - Recen...
122.45.101.67 - Recen...
176.73.49.60 - Recent ...
192.168.1.1 - Recent A...
192.168.1.51 - BTPS-PC...
192.168.1.52 - CIVILDE...
192.168.1.53 - FRONTO...
192.168.1.58 - COMPUT...
192.168.1.62 - COMP11...
192.168.1.65 - ELECTRI...
192.168.1.70 - COMP2 ...
192.168.1.73 - COMP1 ...
192.168.1.114 - com15 ...
192.168.56.1 - COM15 ...
222.107.67.174 - Rece...
ID Start Duration Pr... Sens... Name Visitor Sig. Message Received



HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus the study of setup a hotspot and monitor the hotspot on network has been developed successfully.

Ex. No.: 10	INSTALLATION AND STUDY OF ROOTKITS
Date:	

AIM:

Rootkit 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.

INTRODUCTION:

Breaking the term rootkit into the two component words, root and kit, is a useful way to define it. Root is a UNIX/Linux term that is the equivalent of Administrator in Windows. The word kit denotes programs that allow someone to obtain root/admin-level access to the computer by executing the programs in the kit - all of which is done without end-user consent or knowledge.

A rootkit is a type of malicious software that is activated each time your system boots up. Rootkits are difficult to detect because they are activated before your system's Operating System has completely booted up. A rootkit often allows the installation of hidden files, processes, hidden user accounts, and more in the systems OS. Rootkits are able to intercept data from terminals, network connections, and the keyboard.

Rootkits have two primary functions: remote command/control (back door) and software eavesdropping. Rootkits allow someone, legitimate or otherwise, to administratively control a computer. This means executing files, accessing logs, monitoring user activity, and even changing the computer's configuration. Therefore, in the strictest sense, even versions of VNC are rootkits.

This surprises most people, as they consider rootkits to be solely malware, but in of themselves they aren't malicious at all. The presence of a rootkit on a network was first documented in the early 1990s. At that time, Sun and Linux operating systems were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems, including Windows, and are increasingly difficult to detect on any network.

PROCEDURE:

STEP 1 : Download Rootkit Tool from GMER website www.gmer.net.

STEP 2 : This displays the Processes, Modules, Services, Files, Registry, RootKit Malwares, Autostart, CMD of local host.

STEP 3 : Select Processes menu and kill any unwanted process if any.

STEP 4 : Modules menu displays the various system files like .sys, .dll

STEP 5 : Services menu displays the complete services running with Autos Enable, Disable, System, Boot.

STEP 6 : Files menu displays full files on Hard-Disk volumes.

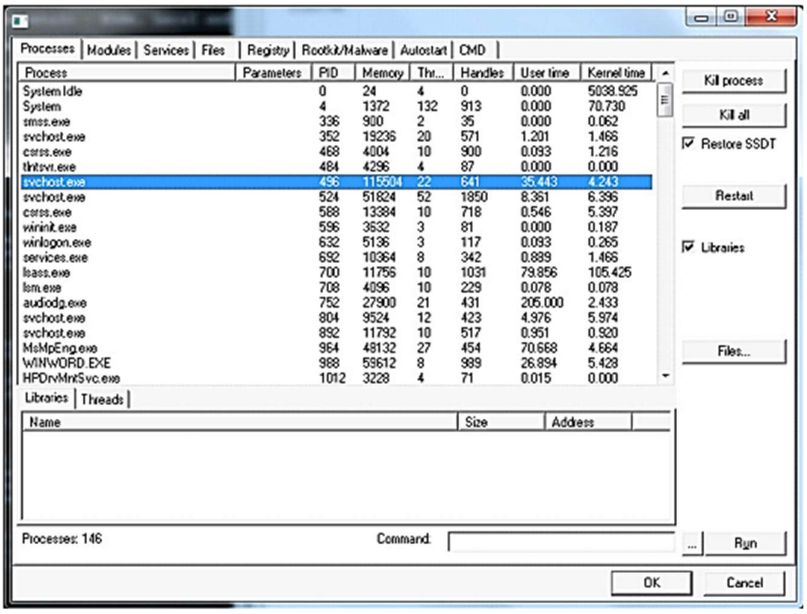
STEP 7 : Registry displays Hkey_Current_user and Hkey_Local_Machine.

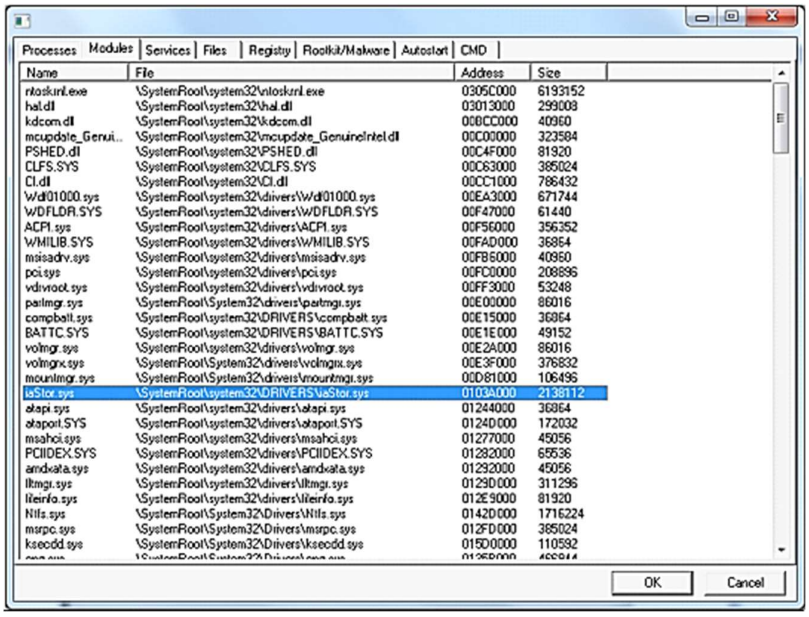
STEP 8 : Rootkits / Malwares scans the local drives selected.

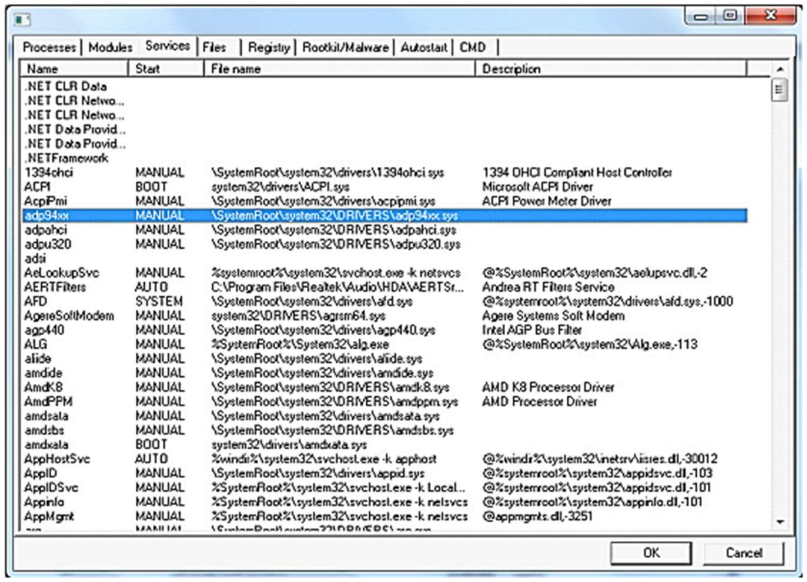
STEP 9 : Autostart displays the registry base Autostart applications.

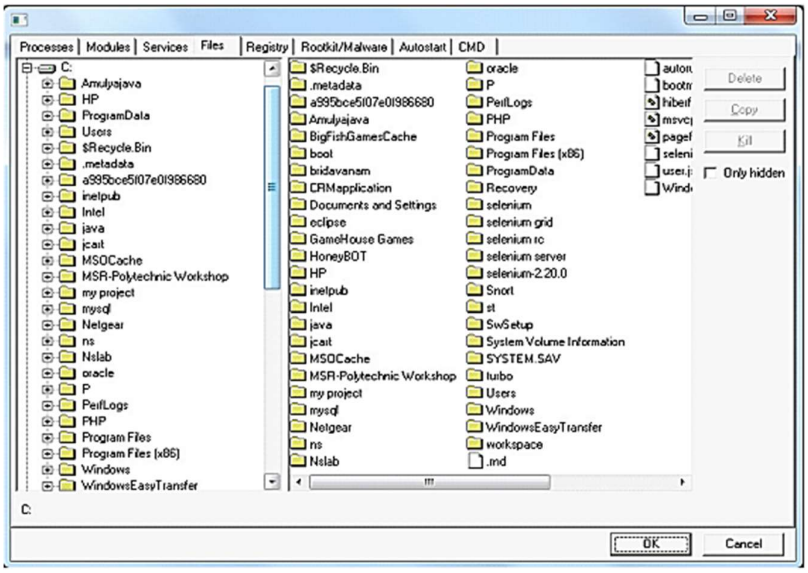
STEP 10 : CMD allows the user to interact with command line utilities or Registry

STEP 11 : Click finish.




Processes | Modules | Services | Files | Registry | Rootkit/Malware | Autostart | CMD
Name | File | Address | Size
ntoskrnl.exe | \SystemRoot\system32\ntoskrnl.exe | 0305C000 | 6193152
hal.dll | \SystemRoot\system32\hal.dll | 03013000 | 299008
kdcom.dll | \SystemRoot\system32\kdcom.dll | 00BCC000 | 40960
mcupdate_Genui... | \SystemRoot\system32\mcupdate_GenuineIntel.dll | 00C00000 | 323584
PSHED.dll | \SystemRoot\system32\PSHED.dll | 00C4F000 | 81920
CLFS.SYS | \SystemRoot\system32\CLFS.SYS | 00C63000 | 385024
CI.dll | \SystemRoot\system32\CI.dll | 00CC1000 | 786432
Wdf01000.sys | \SystemRoot\system32\drivers\Wdf01000.sys | 00EA3000 | 671744
WDFLDR.SYS | \SystemRoot\system32\drivers\WDFLDR.SYS | 00F47000 | 61440
ACPI.sys | \SystemRoot\system32\drivers\ACPI.sys | 00F56000 | 356352
WMILIB.SYS | \SystemRoot\system32\drivers\WMILIB.SYS | 00FAD000 | 36864
msisadrv.sys | \SystemRoot\system32\drivers\msisadrv.sys | 00FB6000 | 40960
pci.sys | \SystemRoot\system32\drivers\pci.sys | 00FC0000 | 208896
vdrvroot.sys | \SystemRoot\system32\drivers\vdrvroot.sys | 00FF3000 | 53248
partmgr.sys | \SystemRoot\System32\drivers\partmgr.sys | 00E00000 | 86016
compbatt.sys | \SystemRoot\system32\DRIVERS\compbatt.sys | 00E15000 | 36864
BATTC.SYS | \SystemRoot\system32\DRIVERS\BATTC.SYS | 00E1E000 | 49152
volmgr.sys | \SystemRoot\system32\drivers\volmgr.sys | 00E2A000 | 86016
volmgrx.sys | \SystemRoot\System32\drivers\volmgrx.sys | 00E3F000 | 376832
mountmgr.sys | \SystemRoot\System32\drivers\mountmgr.sys | 00D81000 | 106496
iaStor.sys | \SystemRoot\system32\DRIVERS\iaStor.sys | 0103A000 | 2138112
atapi.sys | \SystemRoot\system32\drivers\atapi.sys | 01244000 | 36864
ataport.SYS | \SystemRoot\system32\drivers\ataport.SYS | 0124D000 | 172032
msahci.sys | \SystemRoot\system32\drivers\msahci.sys | 01277000 | 45056
PCIIDEX.SYS | \SystemRoot\system32\drivers\PCIIDEX.SYS | 01282000 | 65536
amdxata.sys | \SystemRoot\system32\drivers\amdxata.sys | 01292000 | 45056
iltmgr.sys | \SystemRoot\system32\drivers\iltmgr.sys | 0129D000 | 311296
lfeinfo.sys | \SystemRoot\system32\drivers\lfeinfo.sys | 012E9000 | 81920
Ntfs.sys | \SystemRoot\System32\Drivers\Ntfs.sys | 0142D000 | 1716224
msrpc.sys | \SystemRoot\System32\Drivers\msrpc.sys | 012FD000 | 385024
ksecdd.sys | \SystemRoot\System32\Drivers\ksecdd.sys | 015D0000 | 110592
OK
Cancel


Processes | Modules | Services | Files | Registry | Rootkit/Malware | Autostart | CMD
Name | Start | File name | Description
.NET CLR Data
.NET CLR Netwo...
.NET CLR Netwo...
.NET Data Provid...
.NET Data Provid...
.NETFramework
1394ohci | MANUAL | \SystemRoot\system32\drivers\1394ohci.sys | 1394 OHCI Compliant Host Controller
ACPI | BOOT | system32\drivers\ACPI.sys | Microsoft ACPI Driver
AcpiPmi | MANUAL | \SystemRoot\system32\drivers\acpipmi.sys | ACPI Power Meter Driver
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\svchost.exe -k netsvcs | @%SystemRoot%\system32\aelupsvc.dll,-2
AERTFilters | AUTO | C:\Program Files\Realtek\Audio\HDA\AERTSr... | Andrea RT Filters Service
AFD | SYSTEM | \SystemRoot\system32\drivers\afd.sys | @%systemroot%\system32\drivers\afd.sys,-1000
AgereSoftModem | MANUAL | system32\DRIVERS\agrsm64.sys | Agere Systems Soft Modem
agp440 | MANUAL | \SystemRoot\system32\drivers\agp440.sys | Intel AGP Bus Filter
ALG | MANUAL | %SystemRoot%\System32\alg.exe | @%SystemRoot%\system32\Alg.exe,-113
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\svchost.exe -k apphost | @%windir%\system32\inetsrv\iisres.dll,-30012
AppID | MANUAL | \SystemRoot\system32\drivers\appid.sys | @%systemroot%\system32\appidsvc.dll,-103
AppIDSvc | MANUAL | %SystemRoot%\system32\svchost.exe -k Local... | @%systemroot%\system32\appidsvc.dll,-101
Appinfo | MANUAL | %SystemRoot%\system32\svchost.exe -k netsvcs | @%systemroot%\system32\appinfo.dll,-101
AppMgmt | MANUAL | %SystemRoot%\system32\svchost.exe -k netsvcs | @appmgmts.dll,-3251
OK
Cancel



HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, the study of installation of Rootkit software and its variety of options were developed successfully.

Ex. No.: 11	WORKING WITH NET STUMBLER TO PERFORM WIRELESS AUDIT ON A ROUTER
Date:	

AIM:

To perform wireless audit on an access point or a router and decrypt WEP and WPA (Net Stumbler).

INTRODUCTION:

NET STUMBLER:

NetStumbler (Network Stumbler) is one of the Wi-Fi hacking tool which only compatible with windows, this tool also a freeware. With this program, we can search for wireless network which open and infiltrate the network. Its having some compatibility and network adapter issues. NetStumbler is a tool for Windows that allows you to detect Wireless Local Area Networks (WLANs) using 802.11b, 802.11a and 802.11g. It runs on Microsoft Windows operating systems from Windows 2000 to Windows XP. A trimmed-down version called MiniStumbler is available for the handheld Windows CE operating system.

It has many uses:

a. Verify that your network is set up the way you intended
b. Find locations with poor coverage in your WLAN.
c. Detect other networks that may be causing interference on your network
d. Detect unauthorized "rogue" access points in your workplace
e. Help aim directional antennas for long-haul WLAN links.
f. Use it recreationally for WarDriving.

PROCEDURE:

STEP 1 : Download and install Netstumbler.

STEP 2 : It is highly recommended that the PC should have wireless network card in order to access wireless router.

STEP 3 : Now Run Netstumbler in record mode and configure wireless card.

STEP 4 : There are several indicators regarding the strength of the signal, such as GREEN indicates Strong, YELLOW and other color indicates a weaker signal, RED indicates a very weak and GREY indicates a signal loss.

STEP 5 : Lock symbol with GREEN bubble indicates the Access point has encryption enabled.

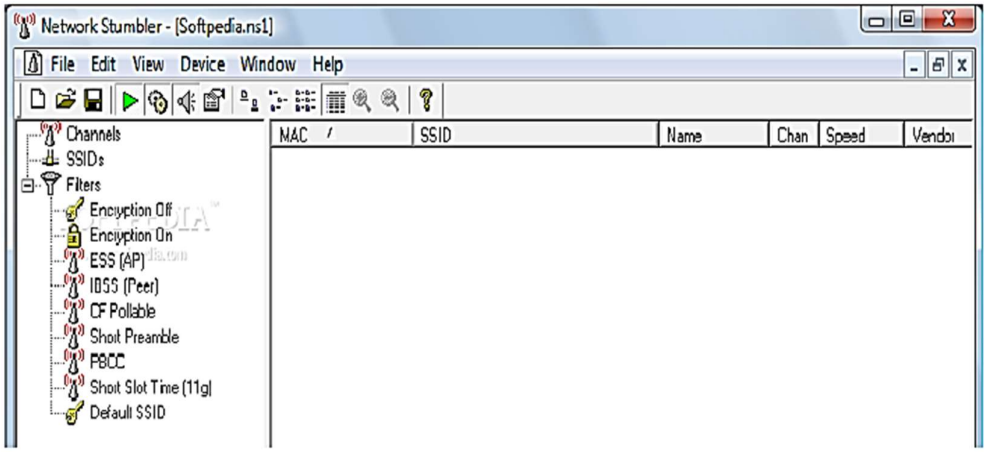
STEP 6 : MAC assigned to Wireless Access Point is displayed on right hand pane.

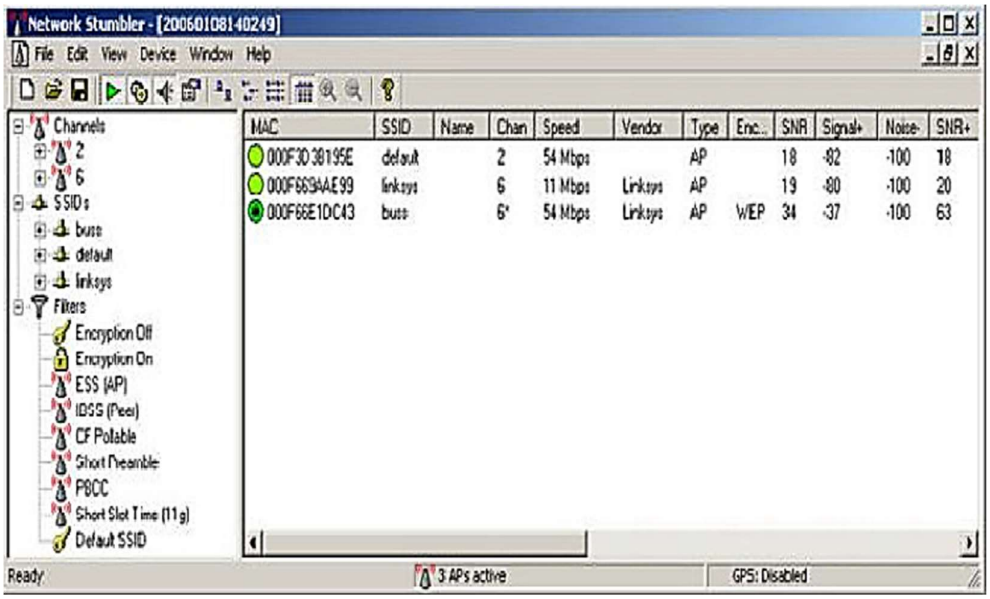
STEP 7 : The next column displays the Access points Service Set Identifier[SSID] which is useful to crack the password.

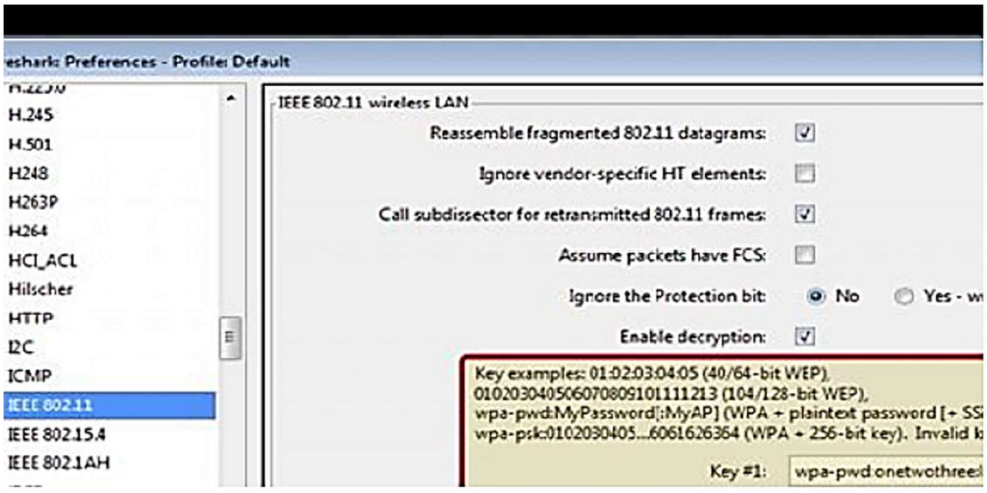
STEP 8 : To decrypt use WireShark tool by selecting Edit preferences IEEE 802.

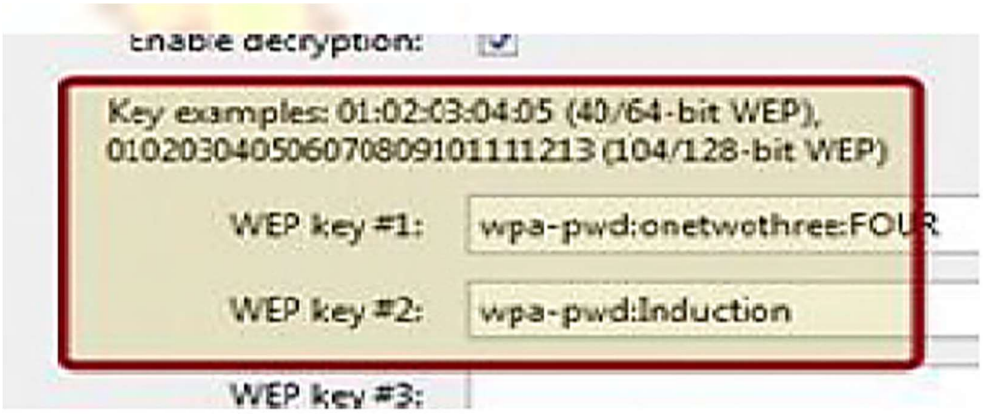
STEP 9 : Enter the WEP keys as a string of hexadecimal numbers as A1B2C3D4E5.

STEP 10 : Click finish.









Adding Keys: Wireless Toolbar

a. If the system is having the Windows version of Wireshark and have an AirPcap adapter, then we can add decryption keys using the wireless toolbar.

b. If the toolbar isn't visible, you can show it by selecting View ——————————Wireless Toolbar.

c. Click on the Decryption Keys button on the toolbar:

➢ This will open the decryption key management window. As shown in the window you can select between three decryption modes: None, Wireshark and Driver:

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus the wireless audit on an access point or a router and decrypt WEP and WPA (Net Stumbler) was done successfully.

Ex. No.: 11	WORKING WITH SNORT TOOL TO DEMONSTRATE INTRUSION DETECTION SYSTEM
Date:	

AIM:

Snort is an open source network intrusion detection system (NIDS) and it is a packet sniffer that monitors network traffic in real time.

INTRODUCTION:

INTRUSION DETECTION SYSTEM:

Intrusion detection is a set of techniques and methods that are used to detect suspicious activity both at the network and host level. Intrusion detection systems fall into two basic categories:

a. Signature-based intrusion detection systems
b. Anomaly detection systems.

Intruders have signatures, like computer viruses, that can be detected using software. You try to find data packets that contain any known intrusion-related signatures or anomalies related to Internet protocols. Based upon a set of signatures and rules, the detection system is able to find and log suspicious activity and generate alerts.

Anomaly-based intrusion detection usually depends on packet anomalies present in protocol header parts. In some cases these methods produce better results compared to signature-based IDS. Usually an intrusion detection system captures data from the network and applies its rules to that data or detects anomalies in it. Snort is primarily a rule-based IDS, however input plug-ins are present to detect anomalies in protocol headers.

SNORT TOOL:

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. When suspicious behavior is detected, Snort sends a real-time alert to syslog, a separate 'alerts' file, or to a pop-up window. Snort is currently the most popular free network intrusion detection software. The advantages of Snort are numerous. According to the snort web site, “It can perform protocol analysis, content searching/matching, and can be used to

detect a variety of attacks and probes, such as buffer overflow, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more" (Caswell).

One of the advantages of Snort is its ease of configuration. Rules are very flexible, easily written, and easily inserted into the rule base. If a new exploit or attack is found a rule for, the attack can be added to the rule base in a matter of seconds. Another advantage of snort is that it allows for raw packet data analysis.

SNORT can be configured to run in three modes:

a. Sniffer mode
b. Packet Logger mode
c. Network Intrusion Detection System mode

1. Sniffer mode:

- ✓ Snort - v Print out the TCP/IP packets header on the screen
- ✓ Snort - vd show the TCP/IP ICMP header with application data in transmit

2. Packet Logger mode:

- ✓ Snort –dev - l c:\log [create this directory in the C drive] and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.
- ✓ Snort –dev - l c:\log –h ipaddress/24:This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory. Snort –l c:\log –b this is binary mode logs everything into a single file.

3. Network Intrusion Detection System mode:

- ✓ Snort –d c:\log –h ipaddress/24 –c snort.conf this is a configuration file applies rule to each packet to decide it an action based upon the rule type in the file.
- ✓ Snort –d –h ipaddress/24 –l c:\log –c snort.conf this will configure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.

PROCEDURE:

STEP 1 : Sniffer mode→ snort -v → Print out the TCP/IP packets header on the screen.

STEP 2 : Snort -vd → Show the TCP/IP ICMP header with application data in transit.

STEP 3 : Packet Logger mode → snort -dev -l c:\log [create this directory in the C drive] and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.

STEP 4 : Snort -dev -l c:\log -h ipaddress/24 → This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory.

STEP 5 : Snort -l c:\log -b → this binary mode logs everything into a single file.

STEP 6 : Network Intrusion Detection System mode → snort -d c:\log -h ipaddress/24 -c snort.conf → This is a configuration file that applies rule to each packet to decide it an action based upon the rule type in the file.

STEP 7 : Snort -d -h ip address/24 –l c:\log –c snort.conf → This will configure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.

STEP 8 : Download SNORT from snort.org. Install snort with or without database support.

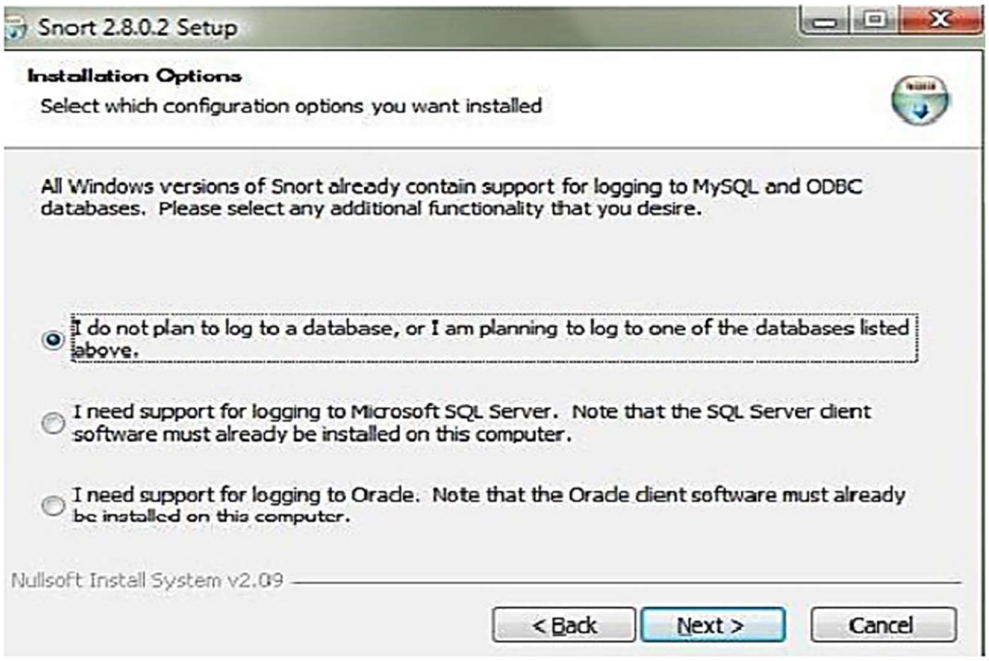
STEP 9 : Select all the components and Click Next. Install and Close.

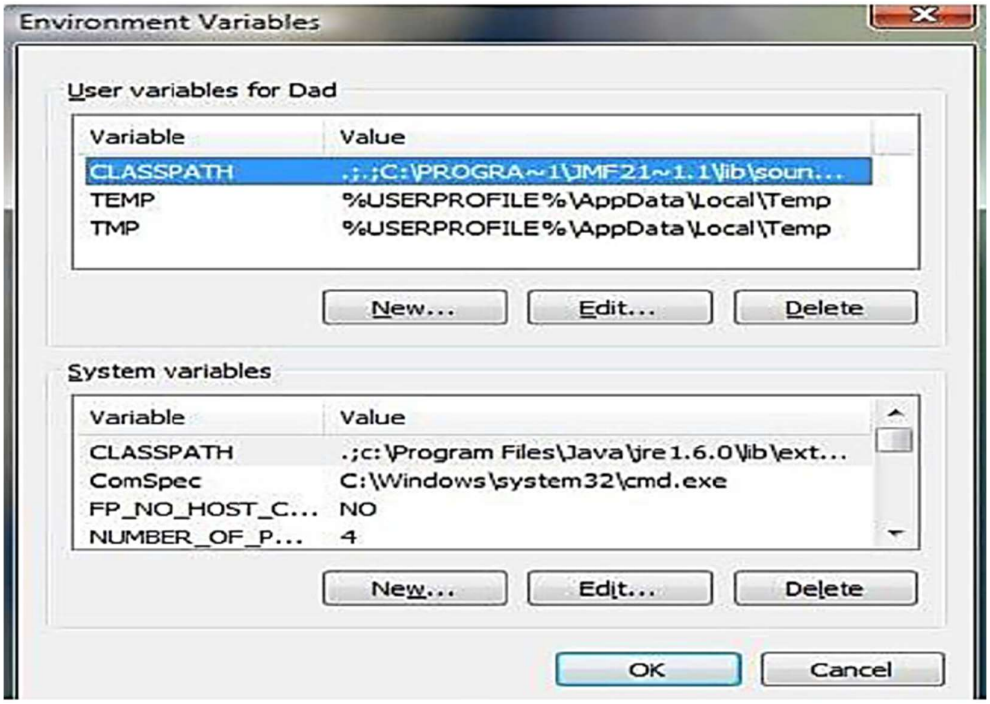
STEP 10 : Skip the WinPcap driver installation.

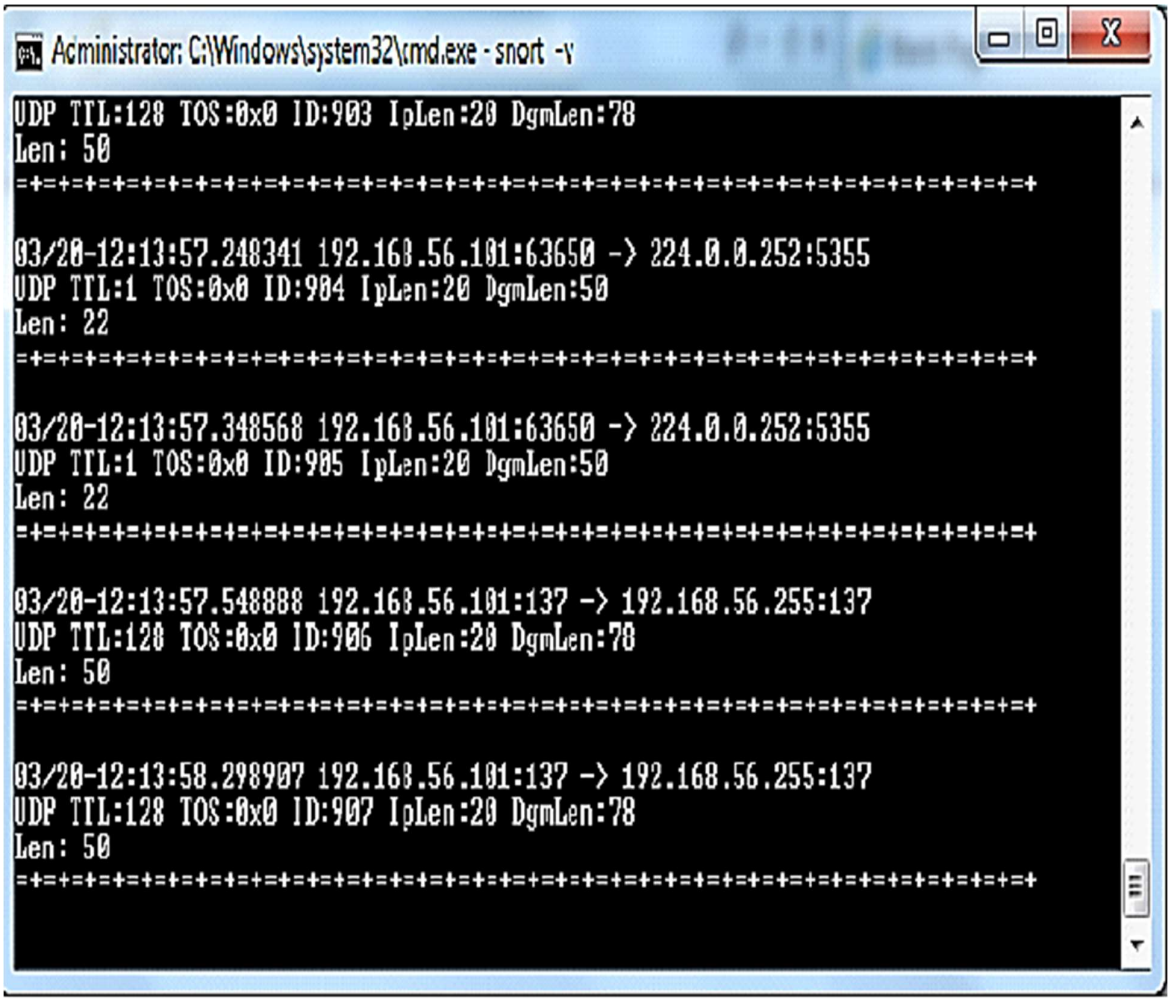
STEP 11 : Add the path variable in windows environment variable by selecting new classpath.

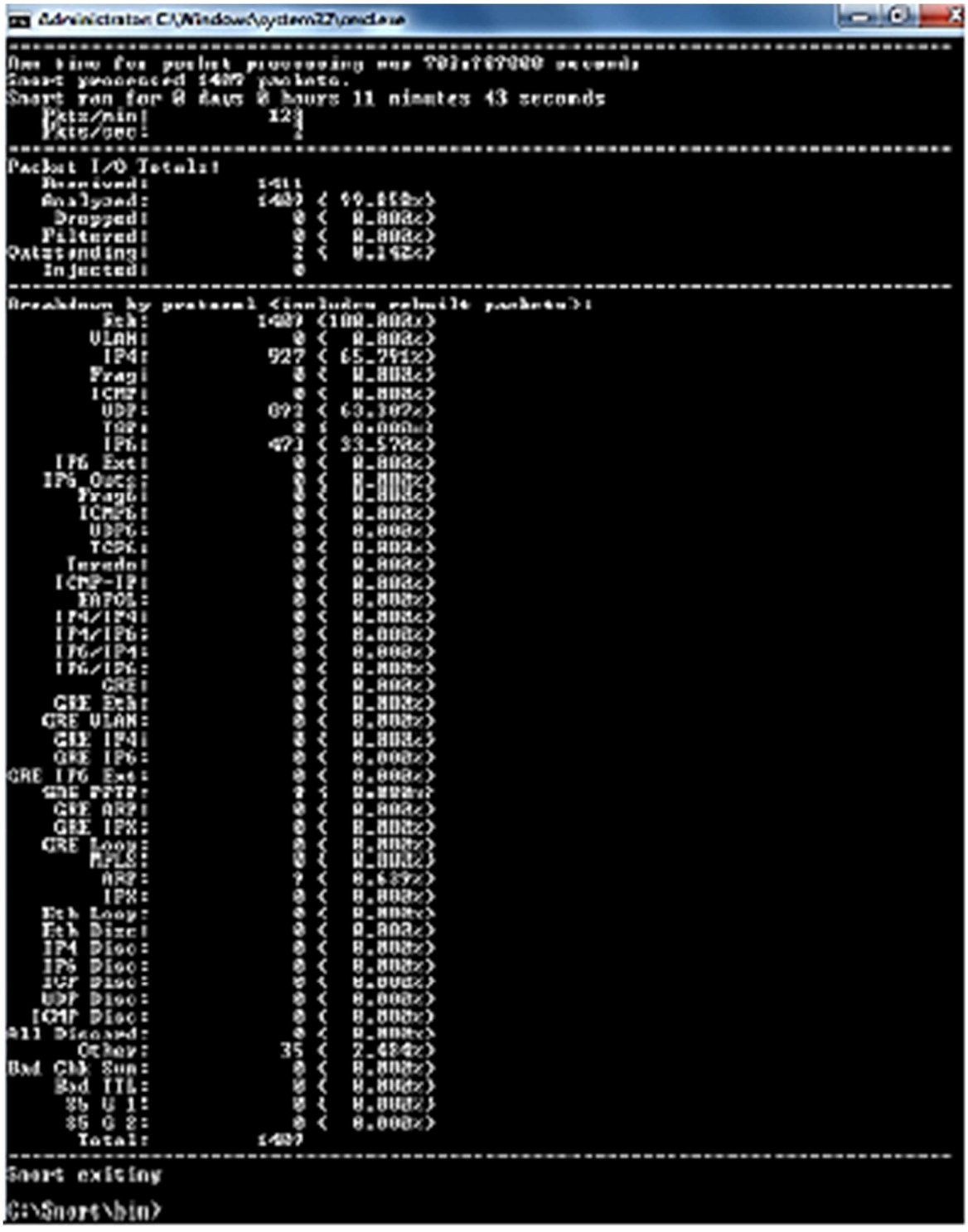
STEP 12 : Create a path variable and point it at snort.exe variable name path variable value → c:\snort\bin.

STEP 13 : Click OK button and then close all dialog boxes. Open command prompt type the following commands:


Snort 2.8.0.2 Setup
Installation Options
Select which configuration options you want installed
All Windows versions of Snort already contain support for logging to MySQL and ODBC databases. Please select any additional functionality that you desire.
I do not plan to log to a database, or I am planning to log to one of the databases listed above.
I need support for logging to Microsoft SQL Server. Note that the SQL Server client software must already be installed on this computer.
I need support for logging to Oracle. Note that the Oracle client software must already be installed on this computer.
Nullsoft Install System v2.09
< Back
Next >
Cancel


Environment Variables
User variables for Dad
Variable
Value
CLASSPATH
.;.;C:\PROGRA~1\JMF21~1.1\lib\soun...
TEMP
%USERPROFILE%\AppData\Local\Temp
TMP
%USERPROFILE%\AppData\Local\Temp
New...
Edit...
Delete
System variables
Variable
Value
CLASSPATH
.;c:\Program Files\Java\jre1.6.0\lib\ext...
ComSpec
C:\Windows\system32\cmd.exe
FP_NO_HOST_C...
NO
NUMBER_OF_P...
4
New...
Edit...
Delete
OK
Cancel


Administrator: C:\Windows\system32\cmd.exe - snort -v
UDP TTL:128 TOS:0x0 ID:903 IpLen:20 DgmLen:78
Len: 50
03/20-12:13:57.248341 192.168.56.101:63650 -> 224.0.0.252:5355
UDP TTL:1 TOS:0x0 ID:904 IpLen:20 DgmLen:50
Len: 22
03/20-12:13:57.348568 192.168.56.101:63650 -> 224.0.0.252:5355
UDP TTL:1 TOS:0x0 ID:905 IpLen:20 DgmLen:50
Len: 22
03/20-12:13:57.548888 192.168.56.101:137 -> 192.168.56.255:137
UDP TTL:128 TOS:0x0 ID:906 IpLen:20 DgmLen:78
Len: 50
03/20-12:13:58.298907 192.168.56.101:137 -> 192.168.56.255:137
UDP TTL:128 TOS:0x0 ID:907 IpLen:20 DgmLen:78
Len: 50


Administrator: C:\Windows\system32\cmd.exe
Snort exiting
C:\Snort\bin>

HICET CSE DEPT CBE 32 (AUTONOMOUS INSTITUTION)	
RECORD OBSERVATION NOTE	
AIM ALGORITHM PROCEDURE	**:**
CODING QUERY DESCRIPTION	**:**
COMPILATION TEST CASES	**:**
EXECUTION AND RESULT	**:**
DOCUMENTATION AND VIVA	**:**
TOTAL	**:**
FACULTY SIGNATURE	

RESULT:

Thus, the demonstration of the instruction detection using Snort tool was done successfully.