IT8761: SECURITY LABORATORY

INDEX

11.a	Defeating Malware i) Building Trojans			
11. b	ii) Rootkit Hunter			
Content Beyond Syllabus				
12	Setup a honey pot and monitor the honeypot on network (KF Sensor)			
13	Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures (GnuPG).			

SYLLABUS
IT8761 SECURITYLABORATORY

LIST OF EXPERIMENTS

1. Perform encryption, decryption using the following substitution techniques
   (i) Ceaser cipher, (ii) playfair cipher iii) Hill Cipher iv) Vigenere cipher
2. Perform encryption and decryption using following transposition techniques
   i) Rail fence ii) row & Column Transformation
3. Apply DES algorithm for practical applications.
4. Apply AES algorithm for practical applications.
5. Implement RSA Algorithm using HTML and JavaScript
6. Implement the Diffie-Hellman Key Exchange algorithm for a given problem.
7. Calculate the message digest of a text using the SHA-1 algorithm.
8. Implement the SIGNATURE SCHEME – Digital Signature Standard.
9. Demonstrate intrusion detection system (ids) using any tool eg. Snort or any other s/w.
10. Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool
11. Defeating Malware
    i) Building Trojans ii) Rootkit Hunter

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

HARDWARE: Standalone desktops – 30 Nos. (or) Server supporting 30 terminals or more.

SoftwareDownloadLinks:
Visual Studio Code: https://code.visualstudio.com/download
Snort - https://www.snort.org/downloads
N-Stalker - https://www.nstalker.com/products/editions/free/download/
GMER - http://www.gmer.net/
JAVA - https://www.java.com/en/download/

SECURITY LABORATORY INTRODUCTION

Security means different things to different people.It is important in all protocols notjust protocols in the security area. Security Services

♣ Confidentiality (privacy)

♣ Authentication (who created or sent the data)

♣ Integrity (has not been altered) ♣ Non-repudiation (parties cannot later deny)

♣ Access control (prevent misuse of resources) Availability (permanence, non-erasure) Cryptography Terminologies Most important concept behind network security is encryption.

Two forms of encryption: Private (or Symmetric)

Single key shared by sender and receiver.

Public-key (or Asymmetric) Separate keys for sender and receiver

Symmetric Key Cryptography Basic ingredients of the scheme:

♣ Plaintext (P)

♣ Message to be encrypted

♣ Secret Key (K)

♣ Shared among the two parties

♣ Cipher text (C)

♣ Message after encryption

♣ Encryption algorithm (EA)

♣ Uses P and K

♣ Decryption algorithm (DA) Uses C and K Security of the scheme

♣ Depends on the secrecy of the key.

♣ Does not depend on the secrecy of thealgorithm. Assumptions that we make:

♣ Algorithms for encryption/decryptionare known to the public.

Ex.No:1(a) Date :	Encryption and Decryption Using Caesar Cipher

AIM:

To encrypt and decrypt the given message by using Caesar Cipher.
encryption algorithm.

ALGORITHM:

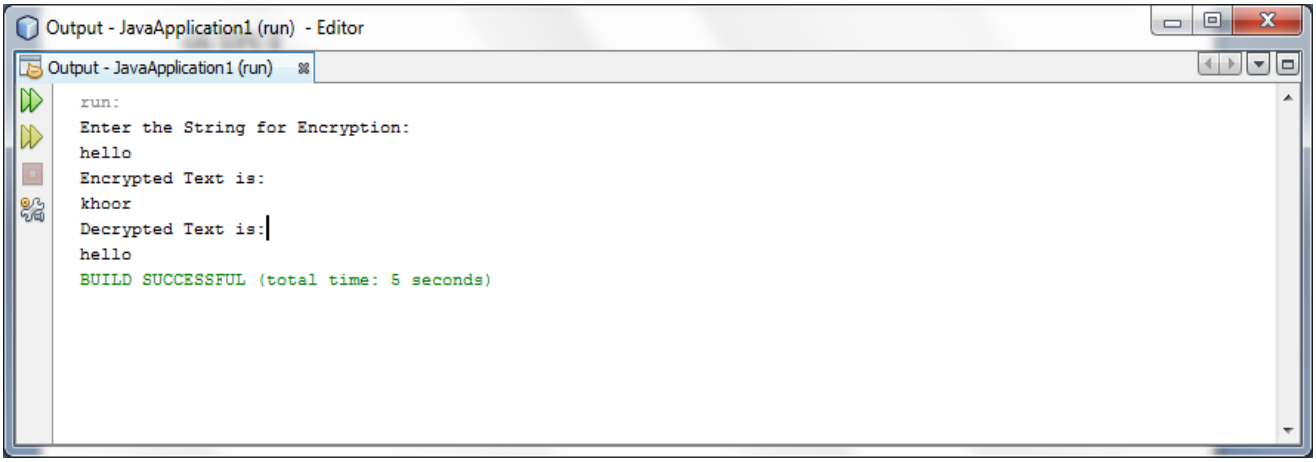
Step 1: Include the header files for implementing Caesar Cipher technique.
Step 2: Declare the necessary variables.
Step 3: Initially get the Plain text and Key value.
Step 4: Now convert the plain text into cipher text using the key value.
Cipher text=Plain text + key % 26
Step 5: Display the cipher text

PROGRAM:

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

public class CaeserCipher
{
   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();
Scanner sc1 = new Scanner(System.in);
System.out.println("Enter the key length: ");
int key=sc1.nextInt();
        System.out.println(encrypt(message, key));
        System.out.println(decrypt(encrypt(message, key), key));
        sc.close();
    }
}
```

SAMPLE OUTPUT:



RESULT:

Thus, the Java program to implement the Caesar Cipher has been compiled and executed successfully.

Ex.No:1(b) Date:	Playfair Cipher

AIM:

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

ALGORITHM:

Step 1: Include the header files for implementing Play fair Cipher technique.
Step 2: Declare the necessary variables.
Step 3: Initially get the Plain text (word).
Step 4: Convert the plain text into characters (a-z) and arrange them in a 5 x 5 matrix.
Step 5: Fill the remaining rows in the matrix with a non-repeating characters.
Step 6: Follow the rules to convert the plaintext to cipher text.
Step 7: Display the cipher text.

PROGRAM:

```
import java.util.*;
class Basic{
  String allChar="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
  boolean indexOfChar(char c)
 {
  for(int i=0;i < allChar.length();i++)
  {
   if(allChar.charAt(i)==c)
                        return true;
  }
  return false;
 }
}

class PlayFair1{
      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.charAt(k)))
            {
                keyMatrix[a][b++]=key.charAt(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.charAt(1);
   String cipherChar="";
   int r1,c1,r2,c2;
   r1=rowPos(a);
```

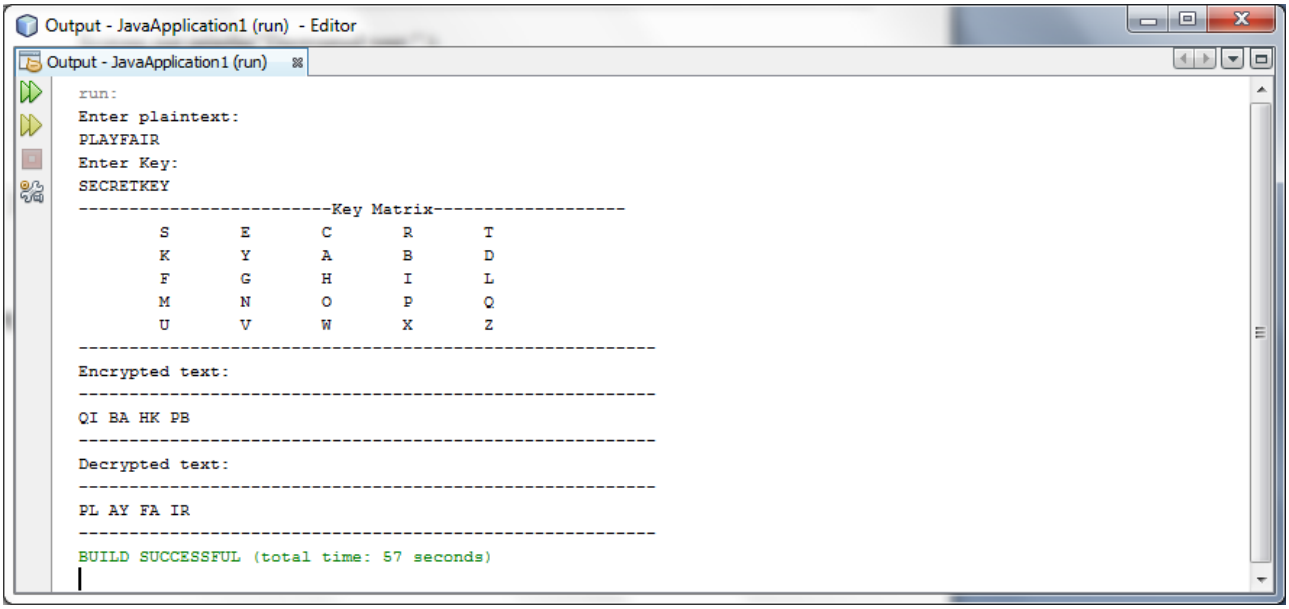
```
    c1=columnPos(a);
    r2=rowPos(b);
    c2=columnPos(b);

    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;
       cipherChar+=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();
```

```
   // System.out.println(plainText.substring(1,2+1));
    if(len/2!=0)
    {
       plainText+="X";
       ++len;
    }

    for(int i=0;i < len-1;i=i+2)
    {
      cipherText+=encryptChar(plainText.substring(i,i+2));
      cipherText+=" ";
    }
    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 PlayFair{
    public static void main(String args[])throws Exception
    {
        PlayFair1 p=new PlayFair1();
        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("-----------------------------------------------------");

    }
}
```

SAMPLE OUTPUT:



RESULT:

Thus, the java program to implement the Play Fair Cipher has been compiled and executed successfully.

Ex.No:1(c) Date:	Hill Cipher

AIM:

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

ALGORITHM:

1. In the Hill cipher Each letter is represented by a number modulo 26.
2. To encrypt a message, each block of n letters is multiplied by an invertible ***nx n*** matrix, again ***modulus 26***.
3. To decrypt the message, each block is multiplied by the inverse of the matrixused for encryption.
4. The matrix used for encryption is the cipher key, and it should be chosenrandomly from the ***set of invertible n × n matrices (modulo 26).***
5. The cipher can, be adapted to an alphabet with any number of letters.
6. All arithmetic just needs to be done modulo the number of letters instead ofmodulo 26.

PROGRAM:

HillCipher.java

```
class hillCipher {
    /* 3x3 key matrix for 3 characters at once */
    public static int[][] keymat = new int[][]
    {{ 1, 2, 1 }, { 2, 3, 2 },{ 2, 2, 1 } };      /* key inverse matrix */
    public static int[][] invkeymat = new int[][]
    { { -1, 0, 1 }, { 2, -1, 0 }, { -2, 2, -1} };
    public static String key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    private static String encode(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 * 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.charAt(x % 26);
        b = key.charAt(y % 26);
```

```
    c = key.charAt(z % 26);
    ret = "" + a + b + c;
    return ret;
}

private static String decode(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.charAt((x % 26 < 0) ? (26 + x % 26) : (x % 26));

    b = key.charAt((y % 26 < 0) ? (26 + y % 26) : (y % 26));
    c = key.charAt((z % 26 < 0) ? (26 + z % 26) : (z % 26));
    ret = "" + a + b + c;
    return ret;
}

public static void main(String[] args) throws java.lang.Exception
{
String msg;
    String  enc  =  "";
    String dec = "";
    int n;
    msg = ("SecurityLaboratory");
    System.out.println("simulation of Hill Cipher\n---------------------------------- ");
    System.out.println("Input message : " + msg);
    msg = msg.toUpperCase();
    msg = msg.replaceAll("\\s", "");
    /* remove spaces */
    n = msg.length() % 3;
    /* append padding text X */
     if (n != 0)
    {
    for (int i = 1; i <= (3 - n); i++)
```

```
        {
            msg += 'X';
          }
        }
        System.out.println("padded message : " + msg);
        char[] pdchars = msg.toCharArray();
        for (int i = 0; i < msg.length(); i += 3)
        {
          enc += encode(pdchars[i], pdchars[i + 1], pdchars[i + 2]);
        }
        System.out.println("encoded message : " + enc);
        char[] dechars = enc.toCharArray();
        for (int i = 0; i < enc.length(); i += 3)
        {
          dec += decode(dechars[i], dechars[i + 1], dechars[i + 2]);
        }
        System.out.println("decoded message : " + dec);
    }
}
```

OUTPUT:

```
Simulating Hill Cipher
----------------------------------

Input Message : SecurityLaboratory
Padded Message : SECURITYLABORATORY Encrypted
Message : EACSDKLCAEFQDUKSXUDecrypted Message
: SECURITYLABORATORY
```

RESULT:

Thus the program for hill cipher encryption and decryption algorithm hasbeen implemented and the output verified successfully.

Ex.No:1(d) Date:	Vigenere Cipher

AIM:

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

ALGORITHM

Step 1: Start the program.

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

Step 3: 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 and decipher() to decrypt the encrypted text.

Step 6: Display the results.

Step 7: Stop the program.

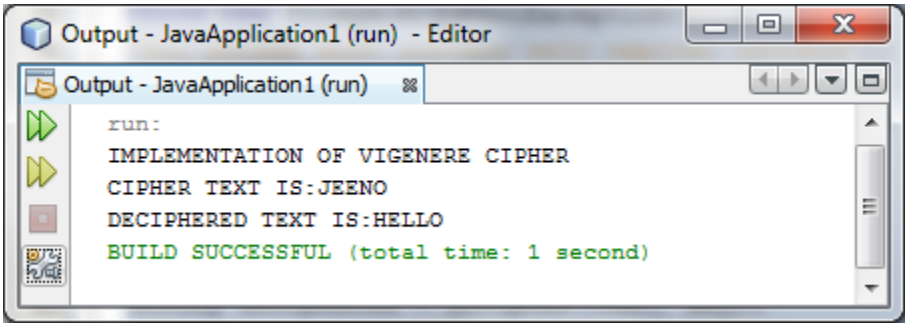
PROGRAM

```
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 modularly
```

```
                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.charAt(i) < 65 || s.charAt(i) > 90)
                { //ASCII character (capital letter)
                        throw new IllegalArgumentException("" +"Ciphertext must contain only
                        capital letters");
                }
                //subtract shift modularly
                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.charAt(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



RESULT

Thus java program to implement Vigenere Cipher was written, executed and output is verified successfully.

REVIEW QUESTIONS

1. **Differentiate between Active attacks and Passive Attacks.**

2. **Compare Substitution and Transposition techniques.**

3. **Define cryptography**

4. **Why network need security?**

5. **Why Modular arithmetic has been used in cryptography?**

6. **Why Random numbers are used in Network Security**

Ex. No : 2(a) Date:	Rail Fence Cipher Transposition Technique

AIM:

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

ALGORITHM:

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

PROGRAM:

railFenceCipher.java

```
class railfenceCipherHelper
{
int depth;

    String encode(String msg, int depth) throws Exception
    {
    int r = depth;
        int l = msg.length();
        int c = l / depth;
        int k = 0;
        char mat[][] = new char[r][c];
        String enc = "";
        for (int i = 0; i < c; i++)
        {
        for (int j = 0; j < r; j++)
        {
                if (k != l)
                {
                    mat[j][i] = msg.charAt(k++);
                }
                 else
                {
                    mat[j][i] = 'X';
                }
```

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

        }
        return enc;
    }

    String decode(String encmsg, int depth) throws Exception
    {
    int r = depth;
        int l = encmsg.length();
        int c = l / depth;
        int k = 0;
        char mat[][] = new char[r][c];
        String dec = "";
        for (int i = 0; i < r; i++)
        {
         for (int j = 0; j < c; j++)
        {
                mat[i][j] = encmsg.charAt(k++);
            }
        }
        for (int i = 0; i < c; i++)
        {
         for (int j = 0; j < r; j++)
        {
                dec += mat[j][i];
            }
        }
        return dec;
    }
}

class railFenceCipher {
    public static void main(String[] args) throws java.lang.Exception
     {
    railfenceCipherHelper rf = new railfenceCipherHelper();
```

```
        String msg, enc, dec;
        msg = "Anna University, Chennai";
        int depth = 2;
        enc = rf.encode(msg, depth);
        dec = rf.decode(enc, depth);
        System.out.println("Simulating Railfence Cipher\n---------------------------------- ");
        System.out.println("Input Message : " + msg);
        System.out.println("Encrypted Message : " + enc);
        System.out.printf("Decrypted Message : " + dec);
    }
}
```

OUTPUT:

Simulating Railfence Cipher

_____-

Input Message : Anna University, Chennai Encrypted
Message : An nvriy hnanaUiest,CeniDecrypted
Message : Anna University, Chennai

RESULT:

Thus the java program for Rail Fence Transposition Technique has been implemented and the output verified successfully.

Ex. No : 2(b) Date:	Row and Column Transformation Technique

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 iscreated with vertical format as: **holewdlo lr**.
3. Now, the receiver has to use the same table to decrypt the cipher text toplain text.

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
SecurityLab
Sreictuy
```

RESULT:

Thus the java program for Row and Column Transposition Technique hasbeen implemented and the output verified successfully.

Ex.No:3 Date:	Data Encryption Standard (DES) Algorithm (User Message Encryption)

AIM:

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

ALGORITHM:

1. Create a DES Key.
2. Create a Cipher instance from Cipher class, specify the following information and separated by a slash (/).
   a. Algorithm name
   b. Mode (optional)
   c. Padding scheme (optional)
3. Convert String into ***Byte[]*** array format.
4. Make Cipher in encrypt mode, and encrypt it with ***Cipher.doFinal()*** method.
5. Make Cipher in decrypt mode, and decrypt it with ***Cipher.doFinal()*** method.

PROGRAM:

DES.java

```
import   java.security.InvalidKeyException;
import  java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

public class DES
{
        public static void main(String[] argv) {

                try{
          System.out.println("Message Encryption Using DES Algorithm\n ---------- ");
                    KeyGenerator keygenerator = KeyGenerator.getInstance("DES");
                    SecretKey myDesKey = keygenerator.generateKey();
                    Cipher desCipher;
                    desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
                    desCipher.init(Cipher.ENCRYPT_MODE,  myDesKey);
                    byte[] text = "Secret Information ".getBytes();
                    System.out.println("Message [Byte Format] : " + text);
```

```
                    System.out.println("Message : " + new String(text));
                    byte[] textEncrypted = desCipher.doFinal(text);
                    System.out.println("Encrypted Message: " + textEncrypted);
                    desCipher.init(Cipher.DECRYPT_MODE, myDesKey); byte[]
                    textDecrypted = desCipher.doFinal(textEncrypted);
                    System.out.println("Decrypted Message: " + new
String(textDecrypted));
                }
                catch(NoSuchAlgorithmException e)
                {
                e.printStackTrace();
                }
                catch(NoSuchPaddingException e)
                {
                e.printStackTrace();
                }
                catch(InvalidKeyException e)
                {
                e.printStackTrace();
                }
                catch(IllegalBlockSizeException e)
                {
                e.printStackTrace();
                }
                catch(BadPaddingException e)
                {
                e.printStackTrace();
                }

        }
}
```

OUTPUT:

```
Message Encryption Using DES Algorithm
------------------------------------------------------

Message [Byte Format] : [B@4dcbadb4
Message : Secret Information Encrypted
Message: [B@504bae78 Decrypted Message:
Secret Information
```

RESULT:

Thus the java program for DES Algorithm has been implemented and theoutput verified successfully.

Ex. No : 4 Date:	Advanced Encryption Standard (AES) Algorithm (URL Encryption)

AIM:

To use Advanced Encryption Standard (AES) Algorithm for a practical application like URL Encryption.

ALGORITHM:

1. AES is based on a design principle known as a substitution–permutation.
2. AES does not use a Feistel network like DES, it uses variant of Rijndael.
3. It has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.
4. AES operates on a 4 × 4 column-major order array of bytes, termed the state

PROGRAM:

AES.java

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AES {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey) {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        }
         catch (NoSuchAlgorithmException e)
        {
            e.printStackTrace();
```

```
        }
         catch (UnsupportedEncodingException e)
         {
        e.printStackTrace();
        }
    }

    public static String encrypt(String strToEncrypt, String secret)
    {
    try
    {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF
            -8")));
        }
         catch (Exception e)
        {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    public static String decrypt(String strToDecrypt, String secret)
    {
    try
    {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        }
         catch (Exception e)
        {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }
```

```
    public static void main(String[] args)
    {
        final String secretKey = "annaUniversity";
        String originalString = "www.annauniv.edu";
        String encryptedString = AES.encrypt(originalString, secretKey);
        String decryptedString = AES.decrypt(encryptedString, secretKey);
        System.out.println("URL Encryption Using AES Algorithm\n      ");
        System.out.println("Original URL : " + originalString);
        System.out.println("Encrypted URL : " + encryptedString);
        System.out.println("Decrypted URL : " + decryptedString);
    }
}
```

OUTPUT:

URL Encryption Using AES Algorithm

Original URL : www.annauniv.edu
Encrypted URL : vibpFJW6Cvs5Y+L7t4N6YWWe07+JzS1d3CU2h3mEvEg=Decrypted URL : www.annauniv.edu

RESULT:

Thus the java program for AES Algorithm has been implemented for URLEncryption and the output verified successfully.

Ex. No : 5 Date:	RSA Algorithm

AIM:

To implement RSA (Rivest–Shamir–Adleman) algorithm by using HTML and Javascript.

ALGORITHM:

1. Choose two prime number p and q
2. Compute the value of n and **p**
3. Find the value of ***e*** (public key)
4. Compute the value of ***d*** (private key) using gcd()
5. Do the encryption and decryption
   a. Encryption is given as,
      $c = t^e \bmod n$
   b. Decryption is given as,
      $t = c^d \bmod n$

PROGRAM:

rsa.html

```
<html>

<head>
    <title>RSA Encryption</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>

<body>
    <center>
        <h1>RSA Algorithm</h1>
        <h2>Implemented Using HTML & Javascript</h2>
        <hr>
        <table>
            <tr>
                <td>Enter First Prime Number:</td>
                <td><input type="number" value="53" id="p"></td>
            </tr>
            <tr>
                <td>Enter Second Prime Number:</td>
```

```
                <td><input type="number" value="59" id="q"></p>
                </td>
            </tr>
            <tr>
                <td>Enter the Message(Plain Text):<br>[A=1, B=2,...]</td>
                <td><input type="number" value="89" id="msg"></p>
                </td>
            </tr>
            <tr>
                <td>n:</td>
                <td>
                    <p id="publickey"></p>
                </td>
            </tr>
            <tr>
                <td>Exponent:</td>
                <td>
                    <p id="exponent"></p>
                </td>
            </tr>
            <tr>
                <td>Private Key:</td>
                <td>
                    <p id="privatekey"></p>
                </td>
            </tr>
            <tr>
                <td>Cipher Text:</td>
                <td>
                    <p id="ciphertext"></p>
                </td>
            </tr>
            <tr>
                <td><button onclick="RSA();">Apply RSA</button></td>
             </tr>
        </table>
    </center>
</body>
<script type="text/javascript">
    function RSA() {
```

```
        var gcd, p, q, no, n, t, e, i, x;
        gcd = function (a, b) { return (!b) ? a : gcd(b, a % b); };
        p = document.getElementById('p').value;
        q = document.getElementById('q').value;
        no = document.getElementById('msg').value;
        n = p * q;
        t = (p - 1) * (q - 1);

        for (e = 2; e < t; e++)
        {
        if (gcd(e, t) == 1)
        {
                break;
            }
        }

        for (i = 0; i < 10; i++)
        {
        x = 1 + i * t
            if (x % e == 0)
             {
            d = x / e;
            break;
            }
        }

        ctt = Math.pow(no, e).toFixed(0);
        ct = ctt % n;

        dtt = Math.pow(ct, d).toFixed(0);
        dt = dtt % n;

        document.getElementById('publickey').innerHTML = n;
        document.getElementById('exponent').innerHTML = e;
        document.getElementById('privatekey').innerHTML = d;
        document.getElementById('ciphertext').innerHTML =  ct;
    }
</script>
</html>
```

OUTPUT:

RSA Algorithm

Implemented Using HTML & Javascript

Enter First Prime Number:	53
Enter Second Prime Number:	59
Enter the Message(cipher text): [A=1, B=2,...]	89
Public Key:	3127
Exponent:	3
Private Key:	2011
Cipher Text:	1394

Apply RSA

RESULT:

Thus the RSA algorithm has been implemented using HTML & CSS and theoutput has been verified successfully.

Ex. No : 6 Date:	Diffie-Hellman key exchange algorithm

AIM:

To implement the Diffie-Hellman Key Exchange algorithm for a given problem .

ALGORITHM:

Step 1: Include the requires header files for implementing Diffiee Hellman
Step 2: Declare the necessary variables.
Step 3: Calculate the length of the key, by declaring key as a long int.
Step 4: Now, the two persons are aware of the key values n and g.
Step 5: Get the values for the two persons from the user.
Step 6: Print the Key for the two persons separately

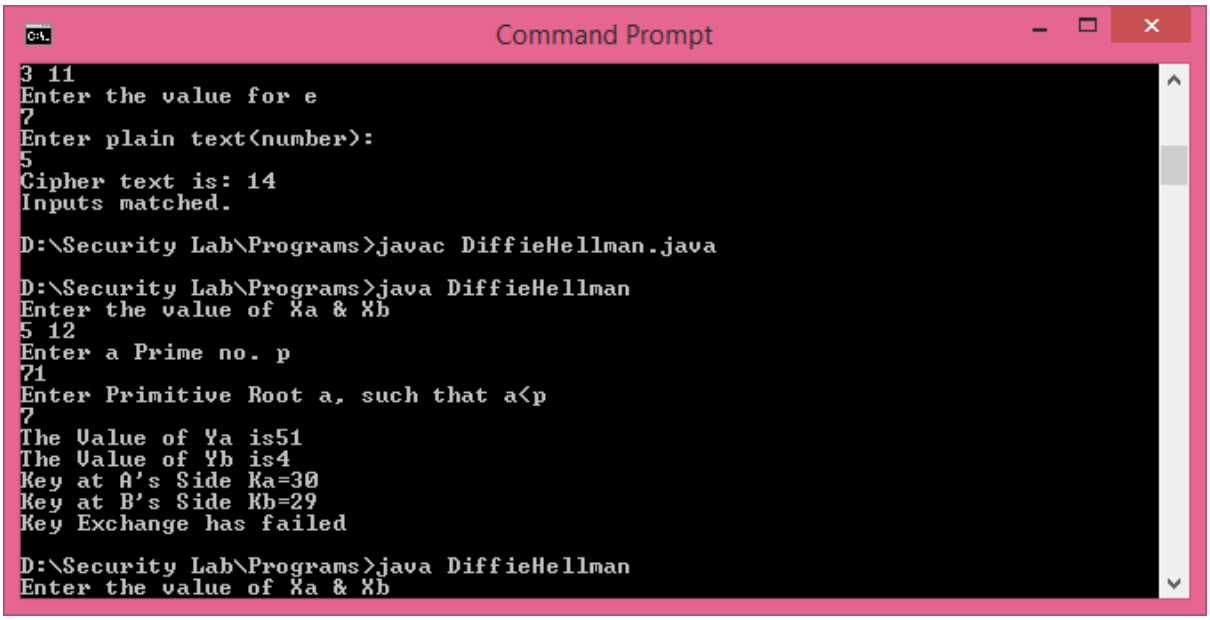
PROGRAM:

DiffieHellman.java

```
import java.util.*;
class DiffieHellman
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter the value of Xa & Xb");
int Xa=sc.nextInt();
int Xb=sc.nextInt();
System.out.println("Enter a Prime no. p");
int p=sc.nextInt();
System.out.println("Enter Primitive Root a, such that a<p");
int a=sc.nextInt();
int Ya=(int)((Math.pow(a,Xa))%p);
int Yb=(int)((Math.pow(a,Xb))%p);
int Ka=(int)((Math.pow(Yb,Xa))%p);
int Kb=(int)((Math.pow(Ya,Xb))%p);
System.out.println("The Value of Ya is"+Ya);
System.out.println("The Value of Yb is" +Yb);
System.out.println("Key at A's Side Ka="+Ka);
System.out.println("Key at B's Side Kb="+Kb);
if(Ka==Kb)
{
System.out.println("Diffie-Hellman Key Exchange has successful");
}
else
```

```
{
System.out.println("Key Exchange has failed");
}
}
}
```

OUTPUT:



RESULT:

Thus the *Diffie-Hellman key exchange algorithm* has been implementedusing Java Program and the output has been verified successfully.

REVIEW QUESTIONS:

1. **Differentiate public key and conventional encryption.**

2. **What is the purpose of Diffie Hellman key exchange?**

3. **Name the principle elements of a public key crypto system?**

4. **What would it take to break RSA?**

5. **Are strong primes necessary in RSA?**

6. **What are Brute Force Attacks?**

Ex. No : 7 Date:	SHA-1 Algorithm

AIM:

To Calculate the message digest of a text using the SHA-1 algorithm.

ALGORITHM:

1. Append Padding Bits
2. Append Length - 64 bits are appended to the end
3. Prepare Processing Functions
4. Prepare Processing Constants
5. Initialize Buffers
6. Processing Message in 512-bit blocks (L blocks in total message)

PROGRAM:

sha1.java

```
import java.security.*;

public class sha1 {
    public static void main(String[] a)
    {
    try
    {
            MessageDigest md = MessageDigest.getInstance("SHA1");
            System.out.println("Message digest object info:\n----------------------------");
            System.out.println("Algorithm=" + md.getAlgorithm());
            System.out.println("Provider=" + md.getProvider());
            System.out.println("ToString=" + md.toString());
            String input = "";
            md.update(input.getBytes());
            byte[] output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output));
            input = "abc";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output));
            input = "abcdefghijklmnopqrstuvwxyz";
            md.update(input.getBytes());
```

```
            output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" + input + "\")=" + bytesToHex(output));
            System.out.println();
        }
         catch (Exception e)
        {
        System.out.println("Exception:" + e);
        }
    }

    private static String bytesToHex(byte[] b)
    {
        char hexDigit[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
        StringBuffer buf = new StringBuffer();

        for (byte aB : b)
        {
            buf.append(hexDigit[(aB >> 4) & 0x0f]);
            buf.append(hexDigit[aB & 0x0f]);
        }
        return buf.toString();
    }
}
```

OUTPUT:

Message digest object info:

Algorithm=SHA1
Provider=SUN version 12
ToString=SHA1 Message Digest from SUN, <initialized>

SHA1("")=DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

SHA1("abc")=A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz")=32D10C7B8CF96570CA04CE37F2A19 D84240D3A89

RESULT:

Thus the *Secure Hash Algorithm (SHA-1)* has been implemented and theoutput has been verified successfully.

REVIEW QUESTIONS:

1. **Why is SHA more secure than MD5?**

2. **List any three hash algorithm**

3. **What are the two approaches of digital signature?**

4. **What is blow fish?**

5. **What is one time password?**

6. **What is birthday attack?**

Ex. No : 8 Date:	Digital Signature Standard

AIM:

To implement the SIGNATURE SCHEME - Digital Signature Standard.

ALGORITHM:

1. Create a KeyPairGenerator object.
2. Initialize the KeyPairGenerator object.
3. Generate the KeyPairGenerator.
4. Get the private key from the pair.
5. Create a signature object.
6. Initialize the Signature object.
7. Add data to the Signature object
8. Calculate the Signature

PROGRAM:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import  java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;

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

      Scanner sc = new Scanner(System.in);
      System.out.println("Enter some text");
      String msg = sc.nextLine();

      KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");

      keyPairGen.initialize(2048);

      KeyPair pair = keyPairGen.generateKeyPair();

      PrivateKey privKey = pair.getPrivate();
```

```
        Signature sign = Signature.getInstance("SHA256withDSA");
        sign.initSign(privKey);
        byte[] bytes = "msg".getBytes();

        sign.update(bytes);

        byte[] signature = sign.sign();

        System.out.println("Digital signature for given text: "+new String(signature,"UTF8"));
    }
}
```

OUTPUT:

```
Enter some textHi
how are you
Digital signature for given text: 0=@gRD???-?.???? /yGL?i??a!?
```

RESULT:

Thus the Digital Signature Standard Signature Scheme has been implemented and the output has been verified successfully.

REVIEW QUESTIONS:

1. **What is the difference between weak and strong collision resistance?**

2. **Distinguish between direct and arbitrated digital signature?**

3. **What are the properties a digital signature should have?**

4. **What requirements should a digital signature scheme should satisfy?**

5. **What is the role of compression function in hash function?**

6. **Differentiate internal and external error control**

Ex. No : 9 Date:	Demonstration of Intrusion Detection System(IDS)

AIM

To demonstrate intrusion detection system (ids) using snort.

PROCEDURE

SNORT can be configured to run in three modes:

1. Sniffer mode
2. Packet Logger mode
3. Network Intrusion Detection System mode

Sniffer mode→snort –v Print out the TCP/IP packets header on the screen

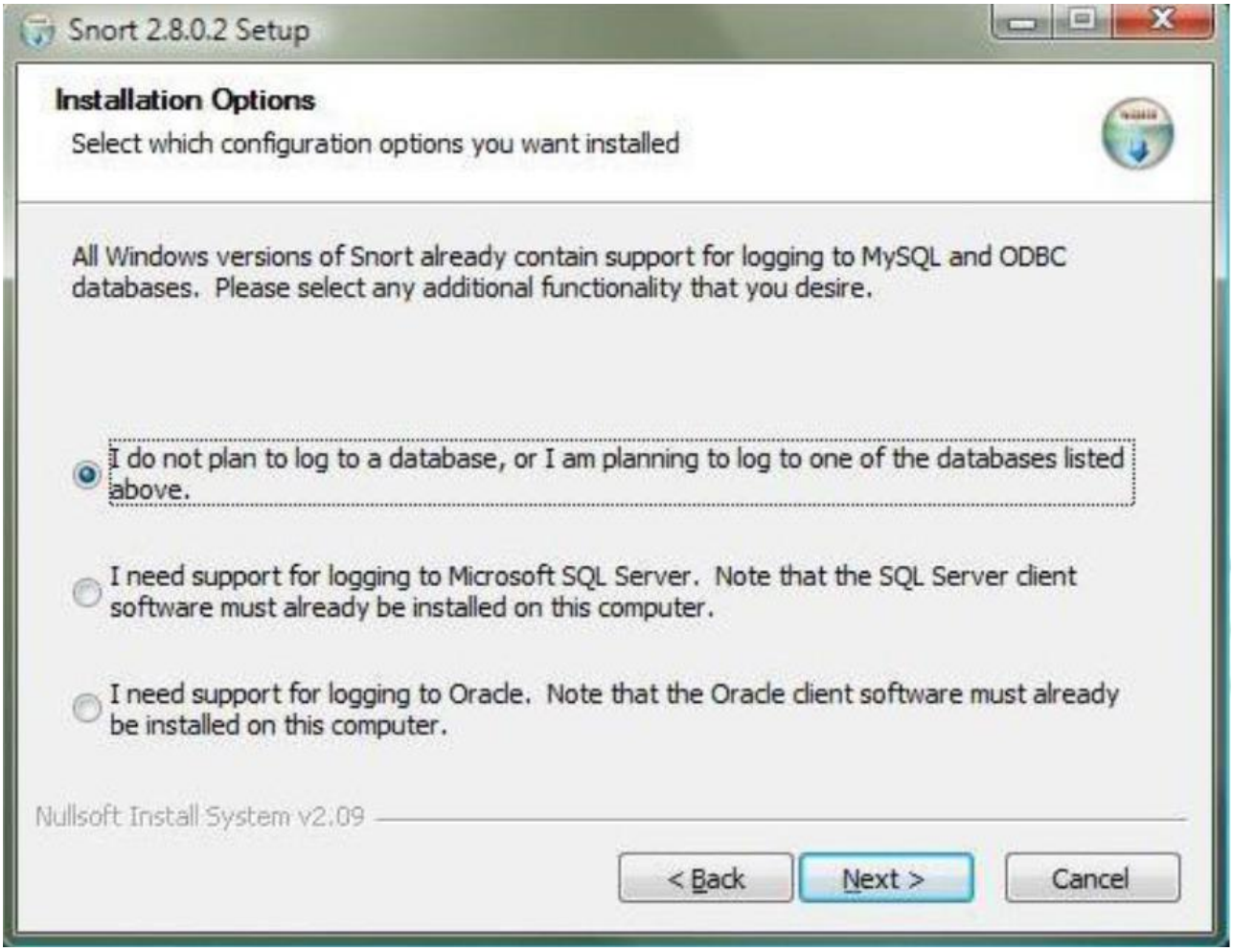
Snort –vd show the TCP/IP ICMP header with application data in transit.

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.

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.

Step 1: Download SNORT from snort.org

Step 2: Install snort with or without database support.



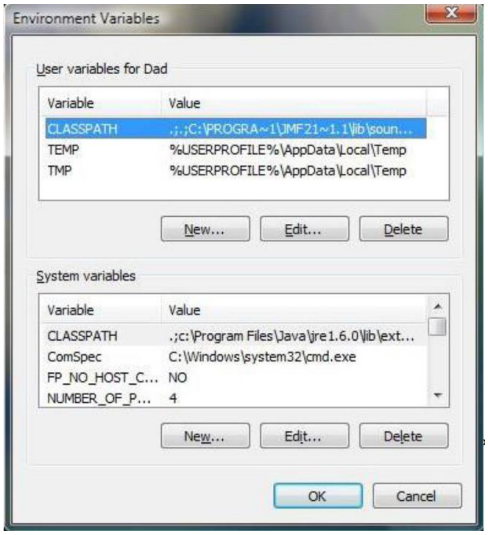
Step 3: Select all the components and Click Next.

Step 4: Install and Close.

Step 5: Skip the WinPcap driver installation

Step 6; Add the path variable in windows environment variable by selecting new classpath.

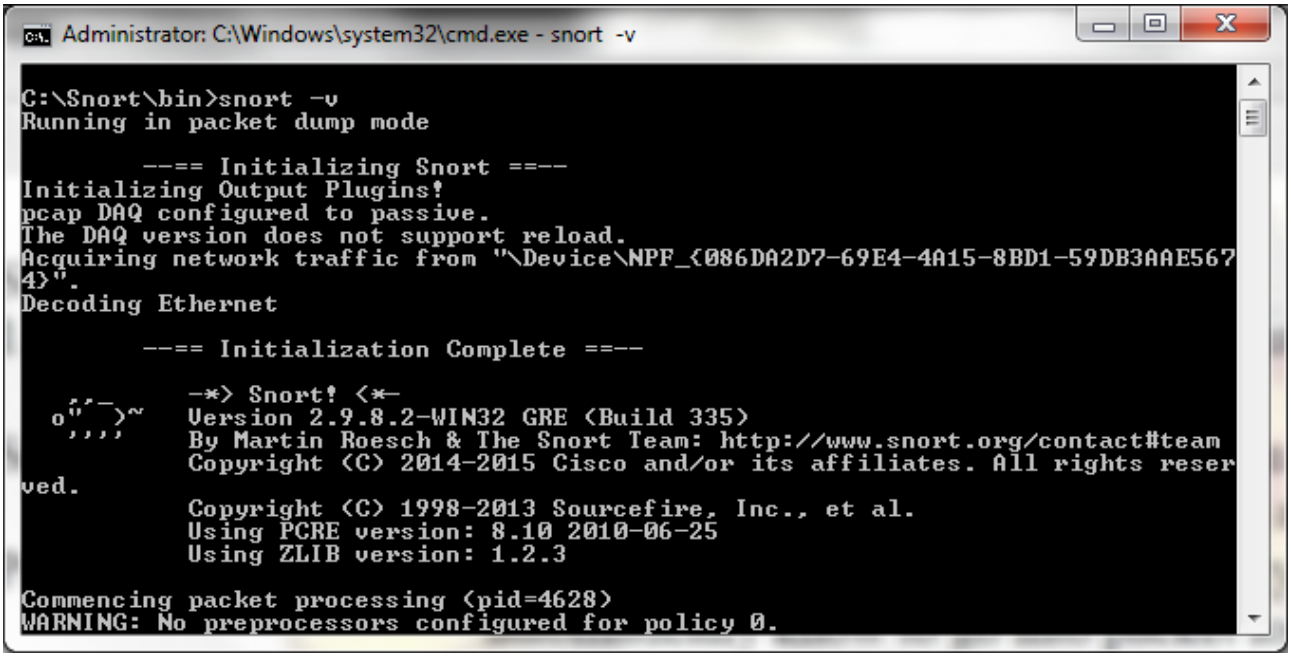
Step 7: Create a path variable and point it at snort.exe variable name→path and variable value→c:\snort\bin.

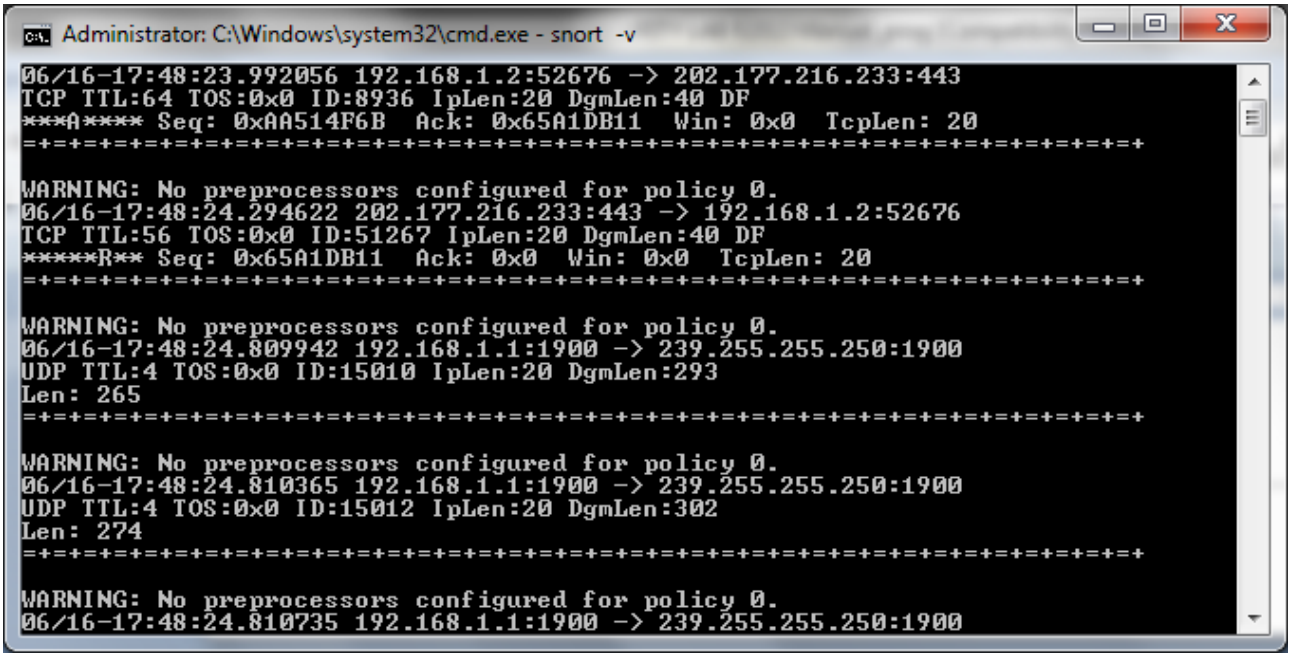


Step 8: Click OK button and then close all dialog boxes.

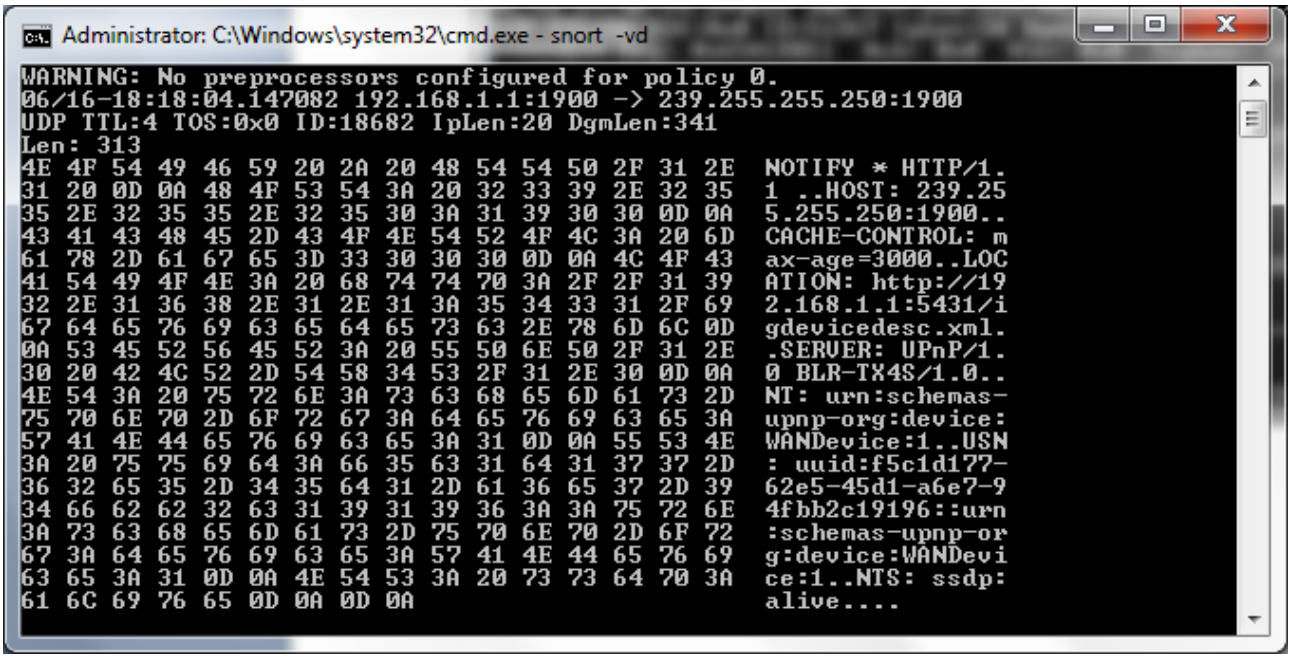
Step 9: Open command prompt and type the following commands:

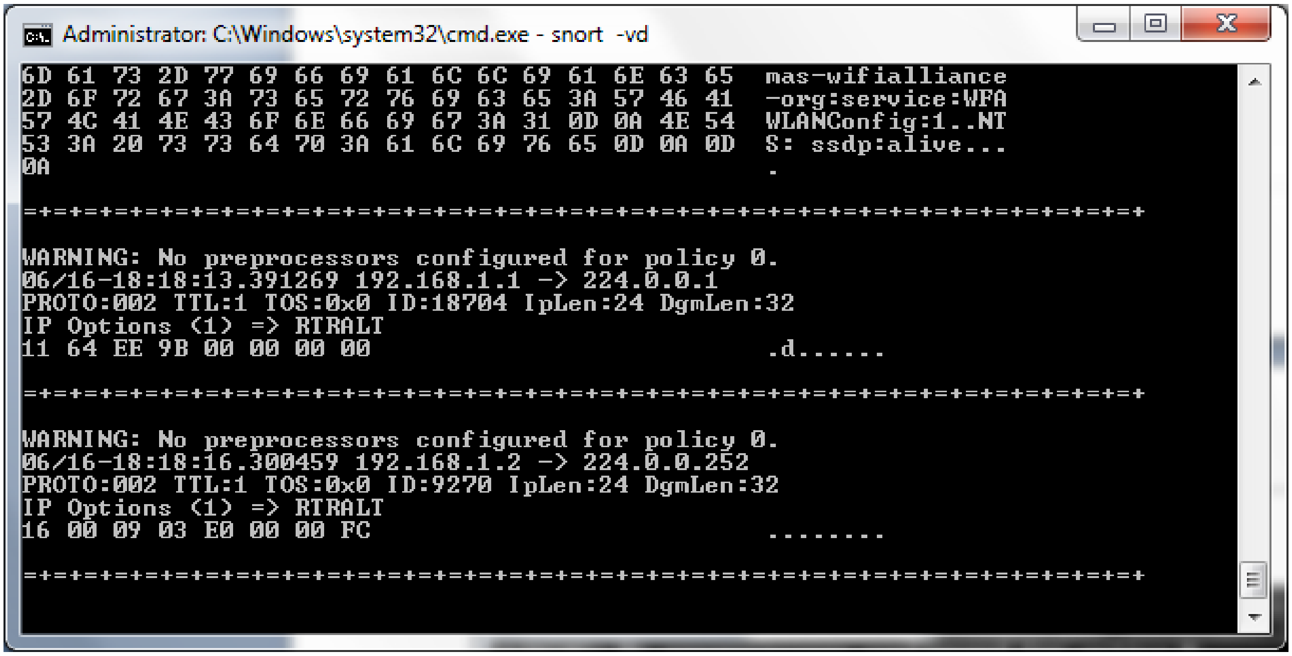
C:\Snort\bin>Snort - v





C:\Snort\bin>Snort – vd





RESULT

Thus Intrusion Detection System was demonstrated using Snort tool successfully.

Ex.No:10 Date:	Exploring N-Stalker, a Vulnerability Assessment Tool

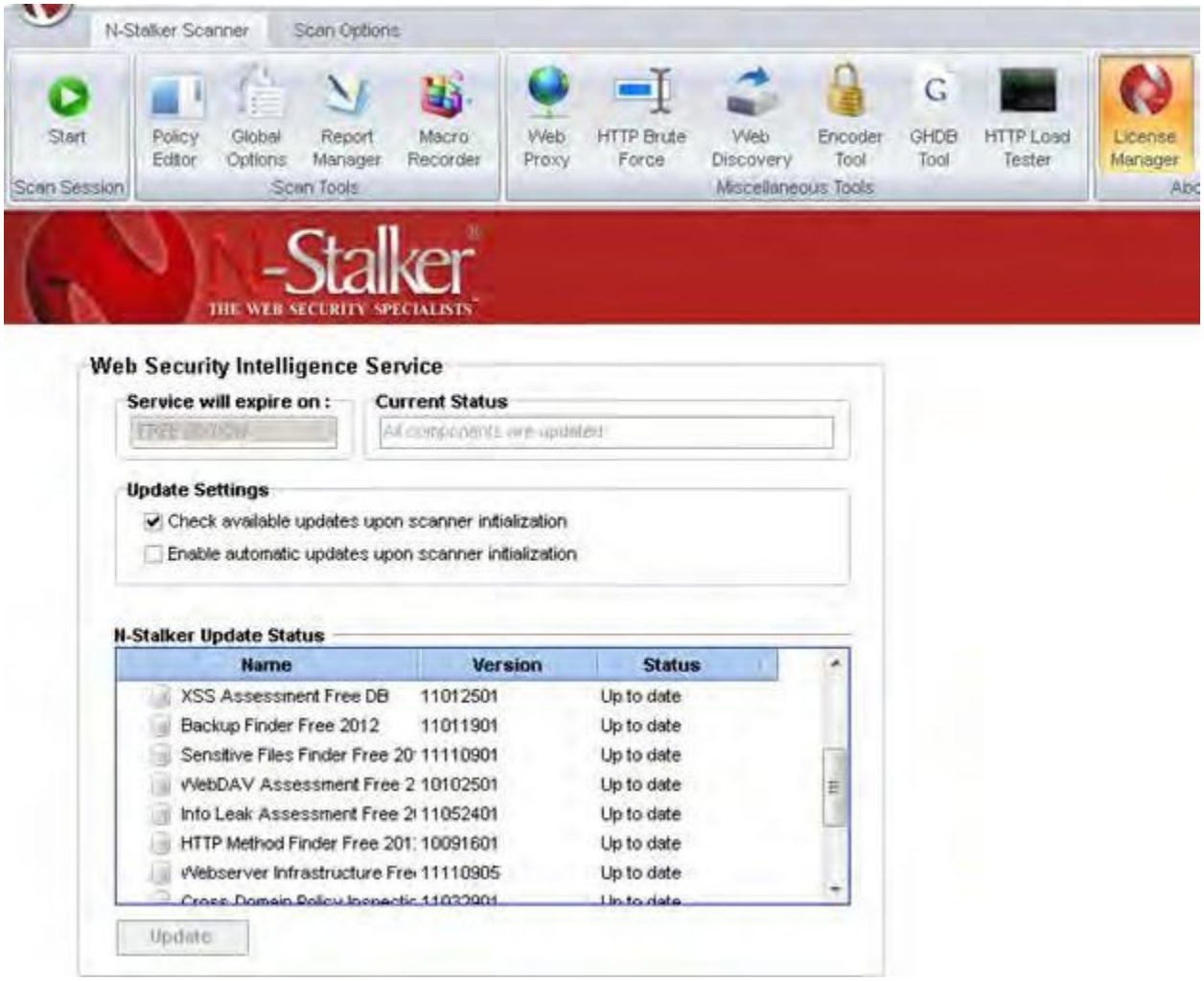
AIM:

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 and 35,000 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.

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.



Now goto “Scan Session”, enter the target URL.

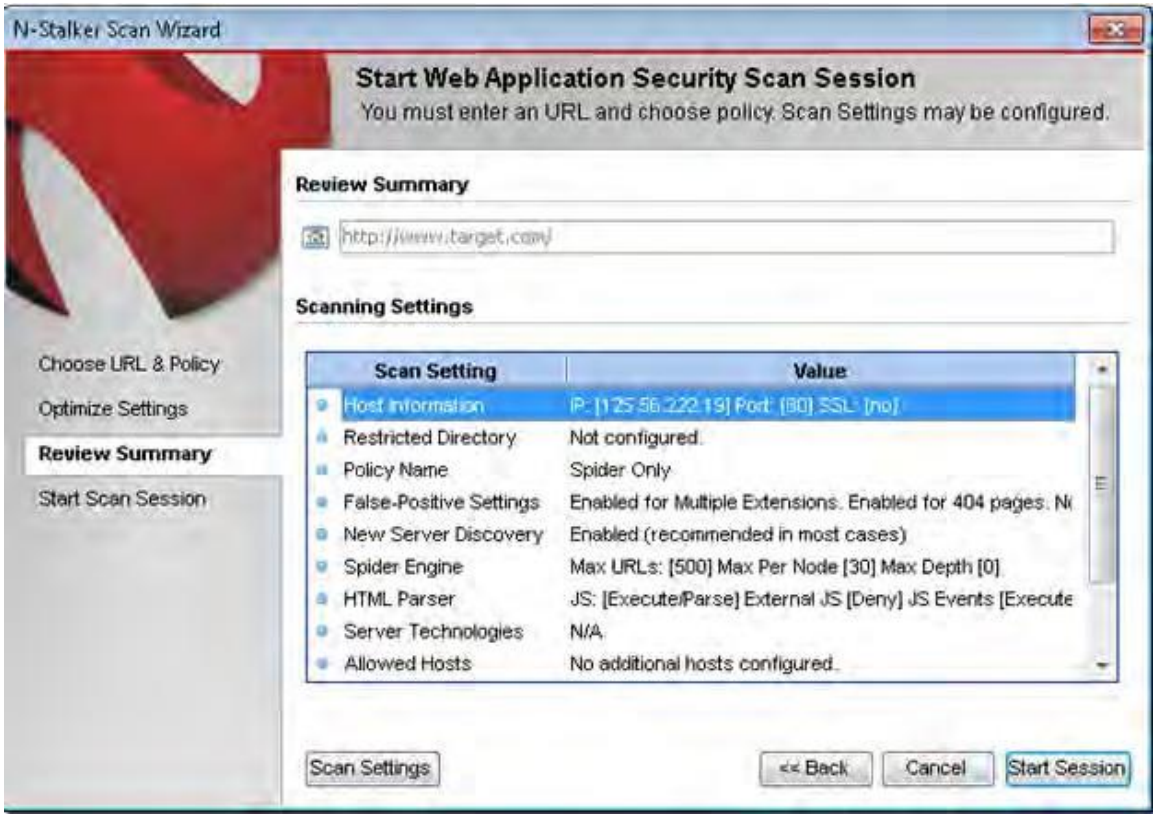
In scan policy, you can select from the four options,

- Manual test which will crawl the website and will be waiting for manual attacks.
- full xss assessment
- 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.

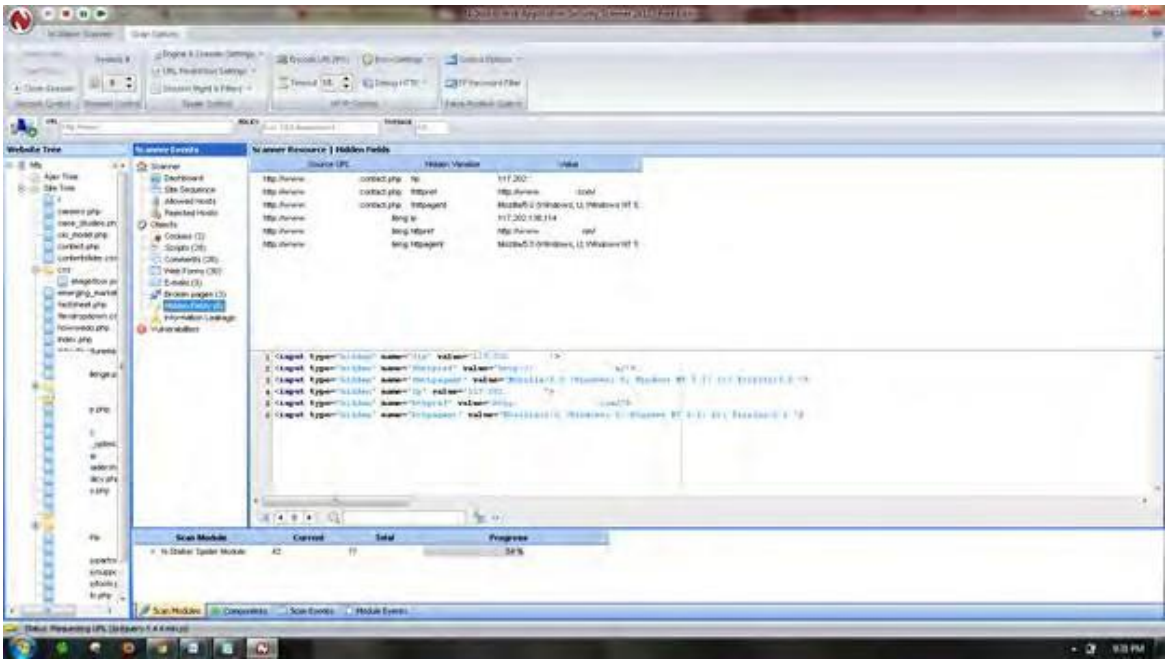
In review option, you can get all the information like host information, technologies used, policy name, etc.


N-Stalker Scan Wizard
Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.
Enter Web Application URL
www.target.com
(E.g: http://www.example.tl/, https://www.test.tl/VirtualDirectory/, etc)
Choose Scan Policy
(choose one)
Load Scan Session
(choose one)
(You may load scan settings from previously saved scan sessions)
Load Spider Data
(You may load spider data from previously saved scan sessions)
Choose URL & Policy
Optimize Settings
Review Summary
Start Scan Session
Scan Settings
Cancel
Next >>

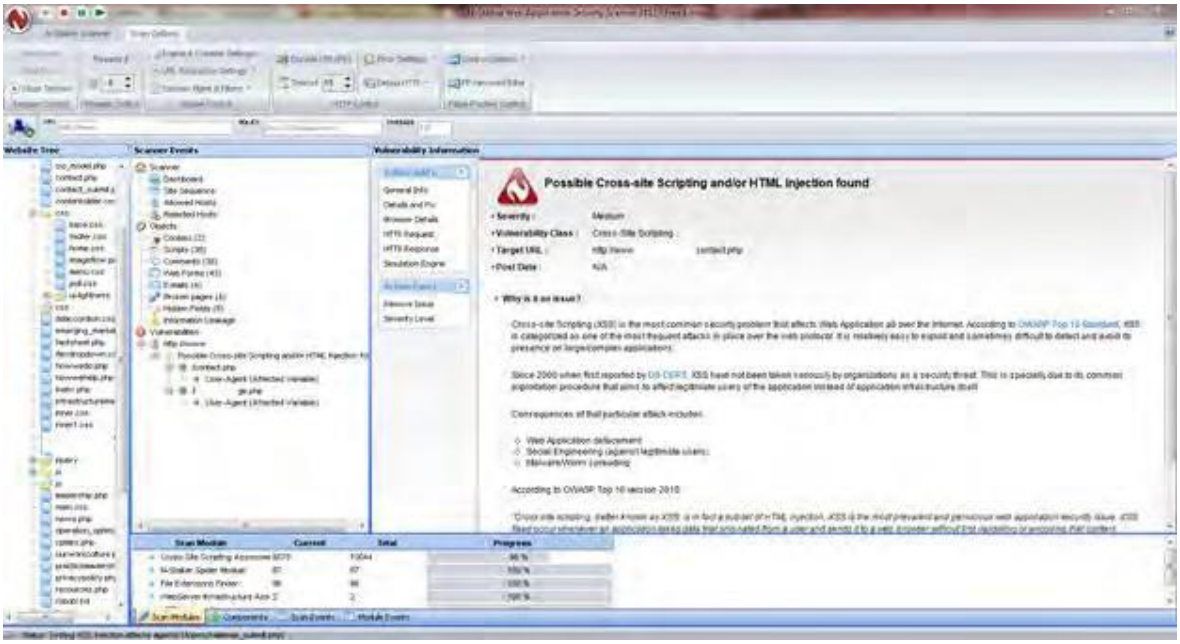

N-Stalker Scan Wizard
Start Web Application Security Scan Session
You must enter an URL and choose policy. Scan Settings may be configured.
Review Summary
http://www.target.com/
Scanning Settings
Scan Setting
Value
Host Information
IP: [125.56.222.19] Port: [80] SSL: [no]
Restricted Directory
Not configured.
Policy Name
Spider Only
False-Positive Settings
Enabled for Multiple Extensions. Enabled for 404 pages. Ni
New Server Discovery
Enabled (recommended in most cases)
Spider Engine
Max URLs: [500] Max Per Node [30] Max Depth [0]
HTML Parser
JS: [Execute/Parse] External JS [Deny] JS Events [Execute
Server Technologies
N/A
Allowed Hosts
No additional hosts configured.
Choose URL & Policy
Optimize Settings
Review Summary
Start Scan Session
Scan Settings
<< Back
Cancel
Start Session

Once done, start the session and start the scan.

The scanner will crawl the whole website and will show the scripts, broken pages, hidden fields, information leakage, web forms related information which helps to analyze further.



Once the scan is completed, the NStalker 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?



RESULT:

Thus the N-Stalker Vulnerability Assessment tool has been downloaded, installed and the features has been explored by using a vulnerable website.

Ex. No: 11(a) Date:	Defeating Malware – Building Trojans

AIM:

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

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.

Restart the PROCEDURE:

5. 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.

CODE:

Trojan.bat

```
@echo off
:x
start mspaint
start notepad
start cmd start
explorer start
control start
calc goto x
```

OUTPUT

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

RESULT:

Thus a trojan has been built and the harmness of the trojan viruses has been explored.

Ex.No:11(b) Date:	Defeating Malware-Rootkit hunter

AIM:

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

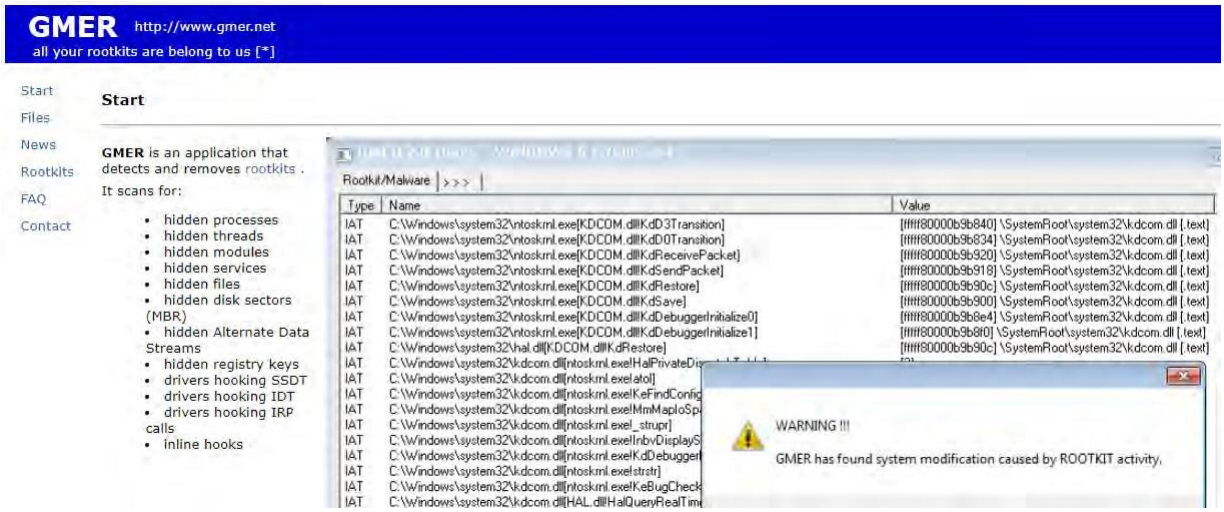
ROOTKIT HUNTER:

- rkhunter (Rootkit Hunter) is a Unix-based tool that scans for rootkits, backdoors and possible local exploits.
- It does this by comparing SHA-1 hashes of important files with known good ones in online databases, searching for default directories (of rootkits), wrong permissions, hidden files, suspicious strings in kernel modules, and special tests for Linux and FreeBSD.
- rkhunter is 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 can run on almost all UNIX-derived systems.

GMER ROOTKIT TOOL:

- GMER is a software tool written by a Polish researcher PrzemysławGmerek, for detecting and removing rootkits.
- 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.

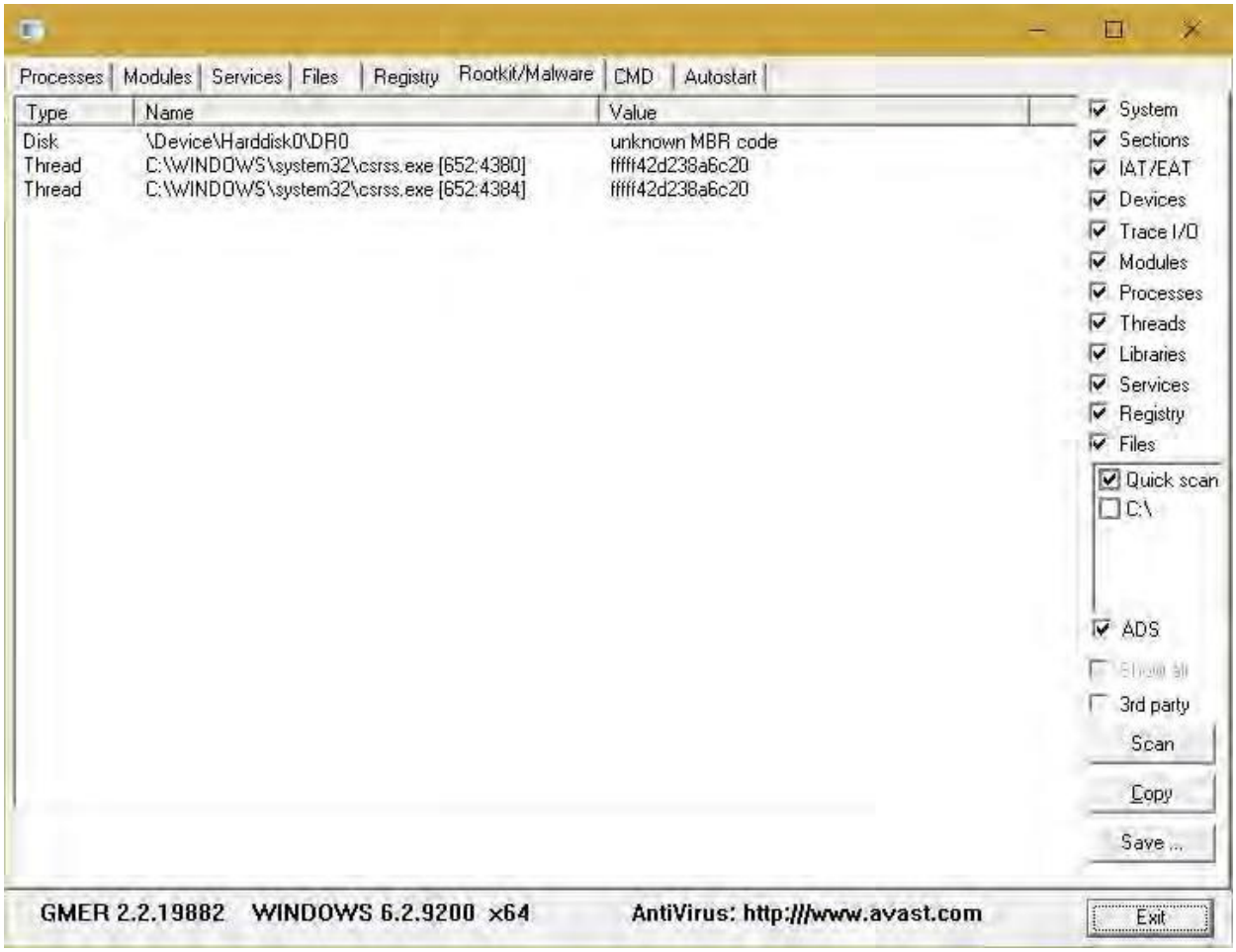
Step 1



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.

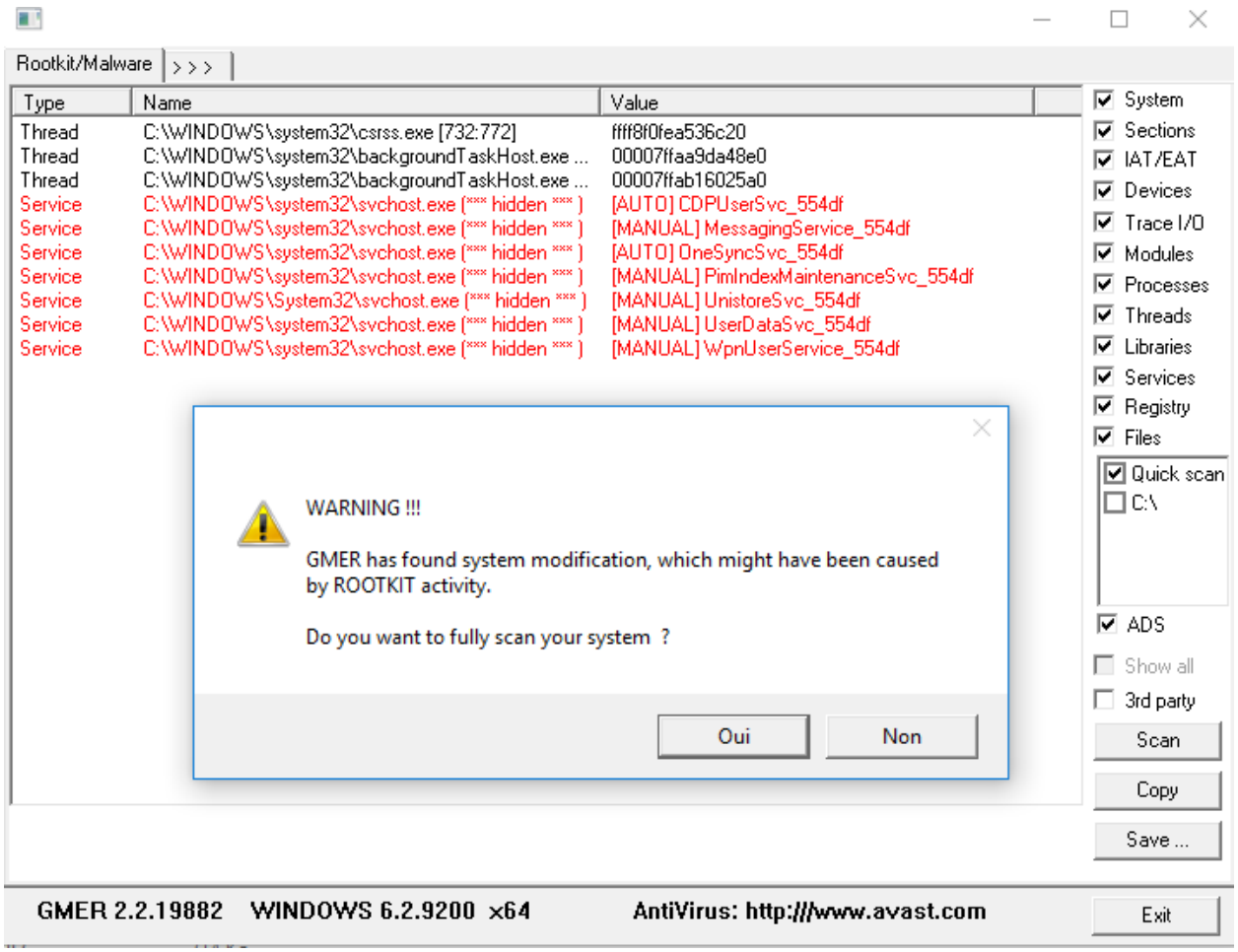
Step 2



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, close the program and restart your PC.

RESULT:

In this experiment a rootkit hunter software tool has been installed and the rootkits have been detected.

Ex.No:12 Date:	SETUPA HONEY POT AND MONITOR THE HONEYPOT ON NETWORK

SETUPA HONEY POT AND MONITOR THE HONEYPOT ON NETWORK

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.

PROCEDURE

Download KF Sensor Evaluation Setu File from KF Sensor Website. ϖ Install with License Agreement and appropriate directory path. Reboot the Computer now.

The KF Sensor automatically starts during windows boot Click Next to setup wizard. Select all port classes to include and Click Next.

Send the email and Send from email enter the ID and Click Next. ϖ Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next.

Select Install as System service and Click Next. Click finish.

STEPS

1. Install winpcap library (mandatory forkfsensor
2.Download kfsensor and install
3.Then restart your pc.Configure properly no change needs to do now go to setting option and configure according to your attack.
4.Now go to your home screen of kf sensor
5.You will get some logs about clients.And it will start working
6.Now just suppose you have to check how many ports are open of this network you can scan your network using nikto or nmap tools

Command for nmap

nmap -T4 -A -v 192.168.6.10 (intense scan)

you can go for other options

and you will see lots of ports are open but they are fake ports.

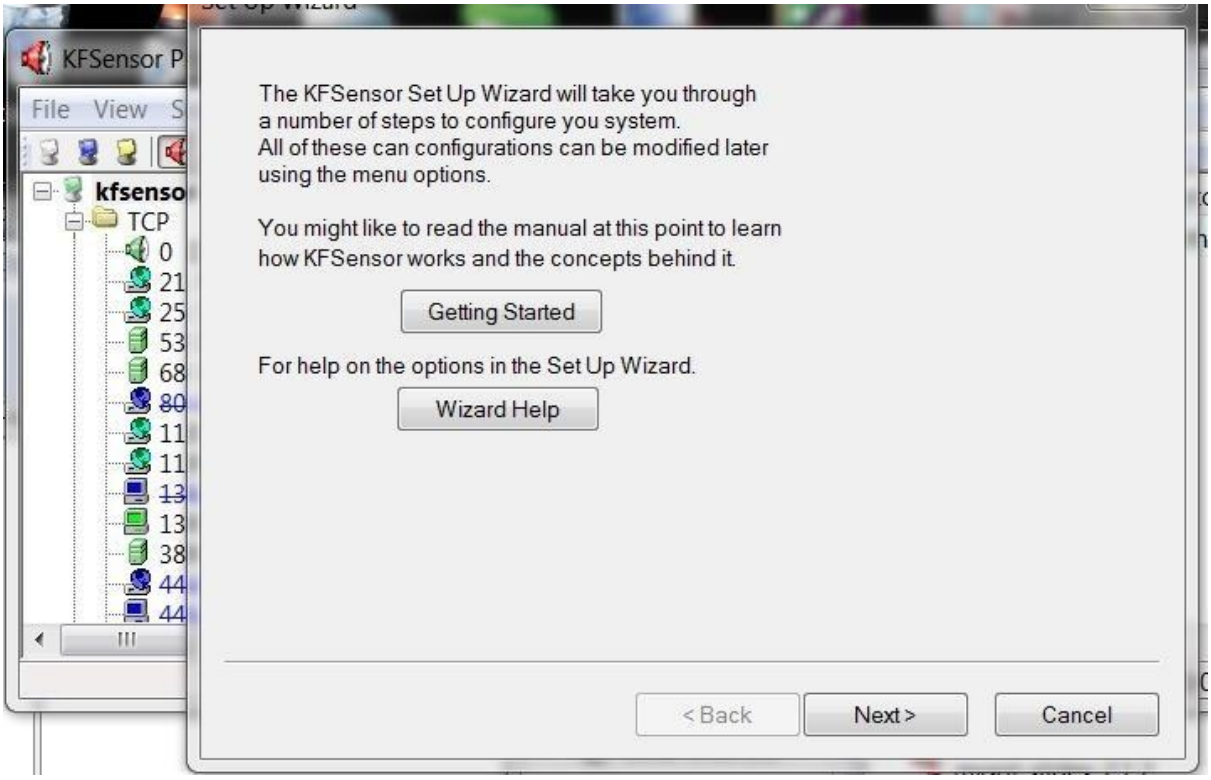
Note: nmap is already given in kali linux, you can download for ubuntu or windows

ALTERNATE METHOD:

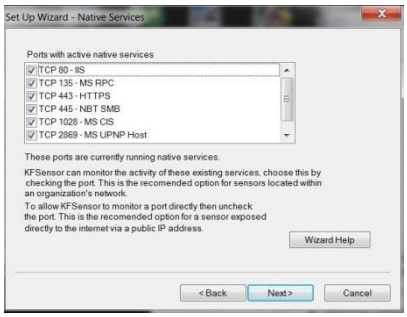
PROCEDURE:

Step1: Install KFSensor

Step 2:Once it is installed, right-click on the KFSensor icon and "run as administrator". You should get a set up wizard like this.

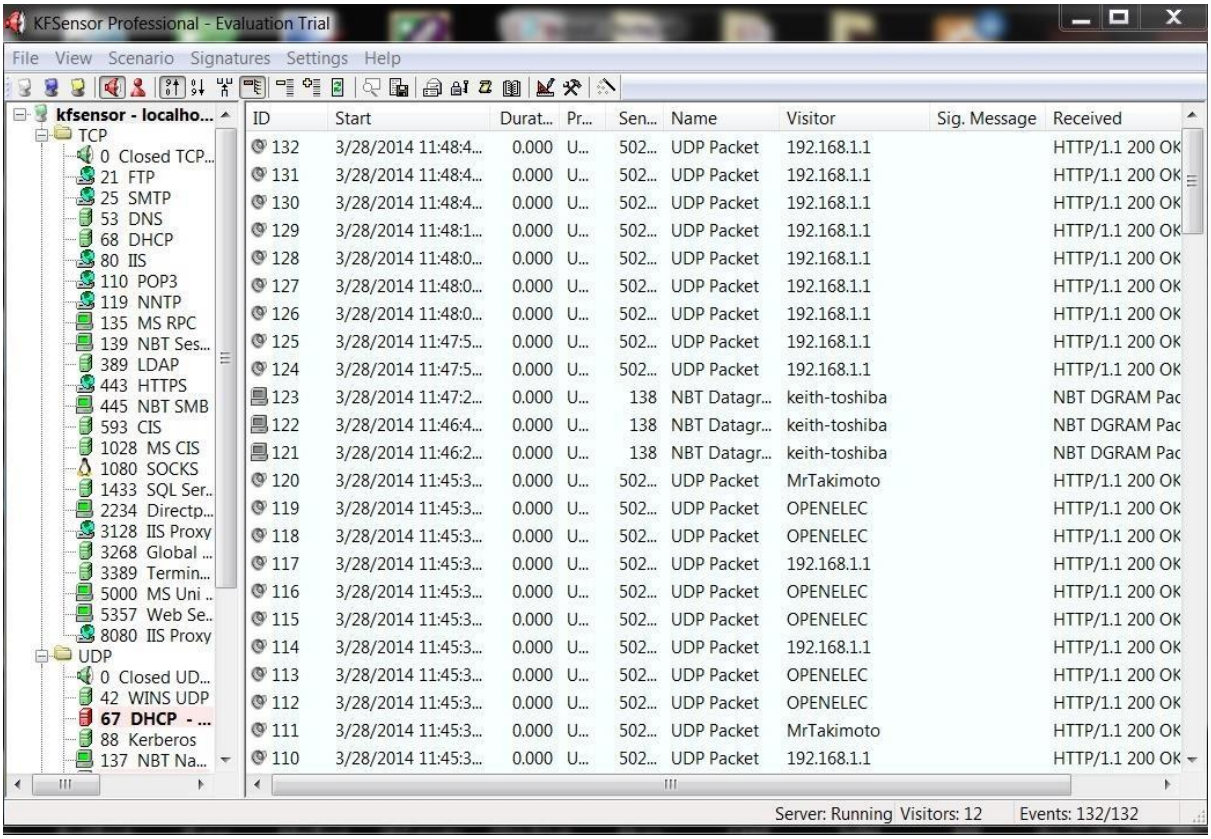


Step 3:After going through a few more screens in the wizard choosing the defaults, you come to the screen below that allows you to choose the native services. Let's choose all of them.



Step 4: choose your domain name. You might want to make it sound enticing. The default is *networksforu.com*

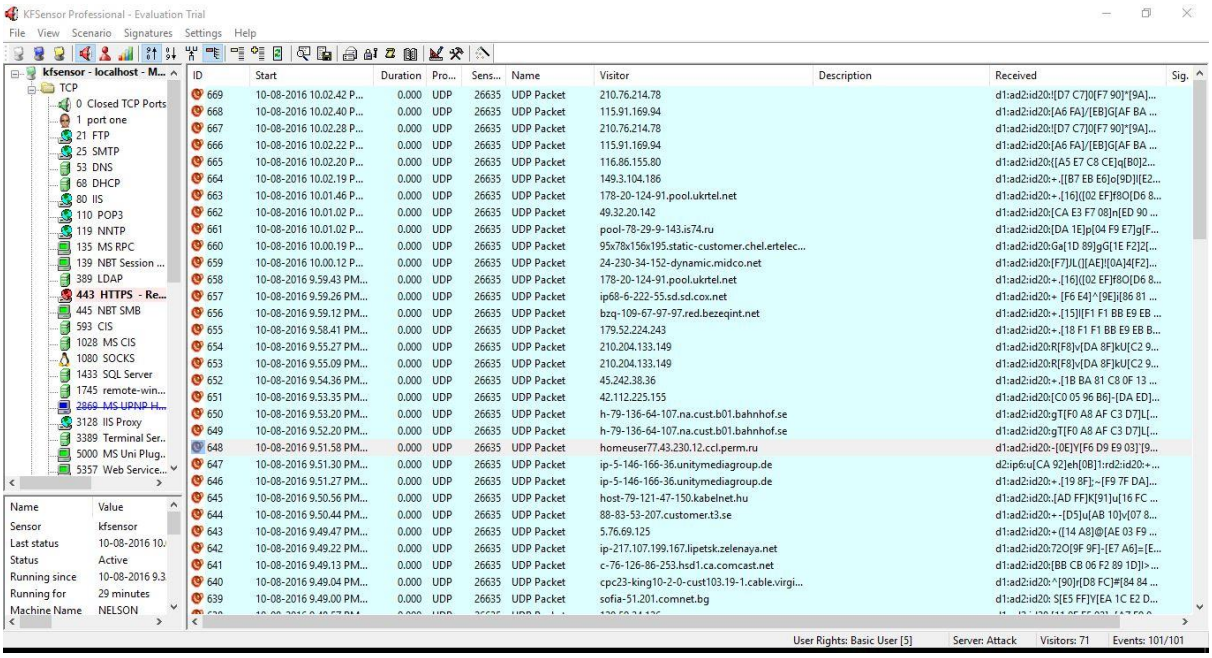
Step 5:: When you have completed the wizard, click *Finish* and you should have an application that looks like this.



Step 6:Scan with NmapLet's do a SYN scan:

- **nmap -sS 192.168.1.1**

Step 8: As you can see, we find numerous ports open. As a hacker, this is a big RED FLAG. Few commercial web servers would leave all these ports open.



RESULT

Thus Setting up a Honey Pot and Monitor the Honey pot on Network was executed successfully.

Ex.No:13 Date:	SECURED DATA TRANSMISSION USING GNUPG

AIM

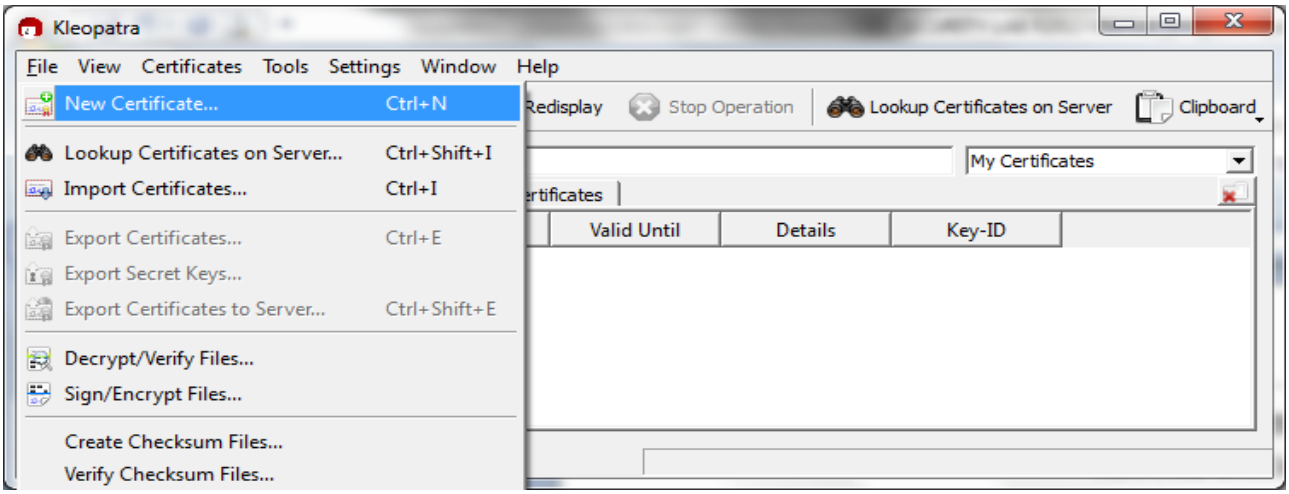
To create Digital Signature, secure Data Storage & transmission using GnuPG.

PROCEDURE

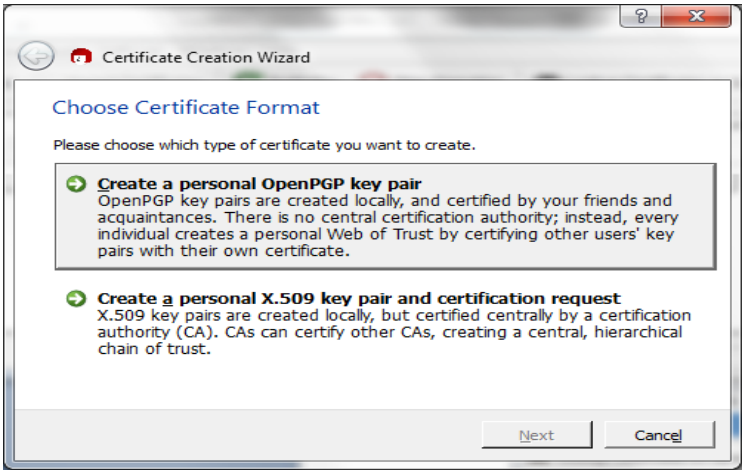
GENERATING KEYPAIR

Step 1: Open up Kleopatra.

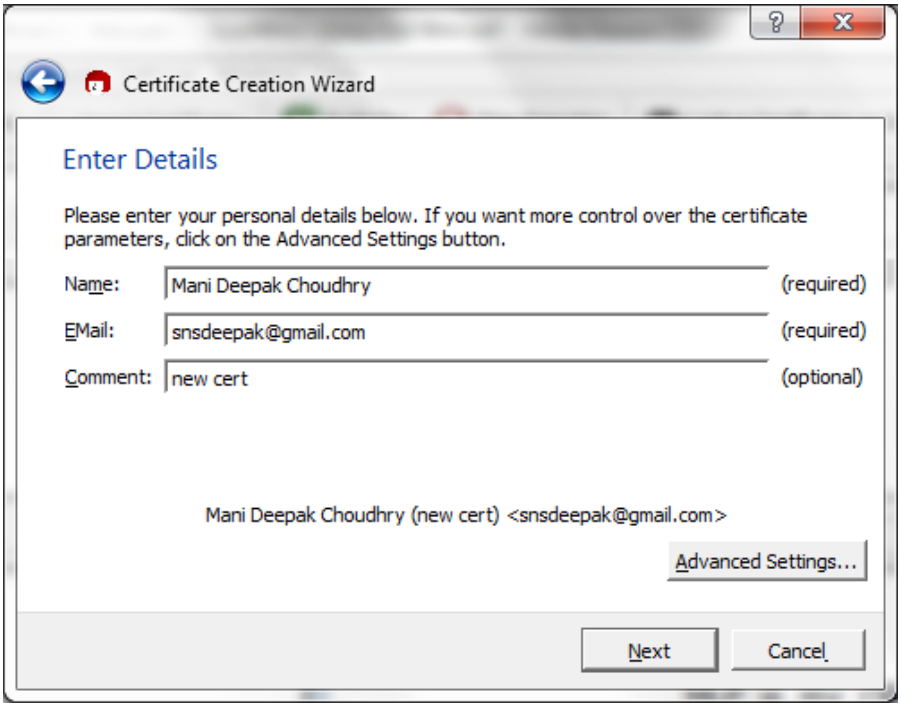
Step 2: Go to 'File', then 'New Certificate…'



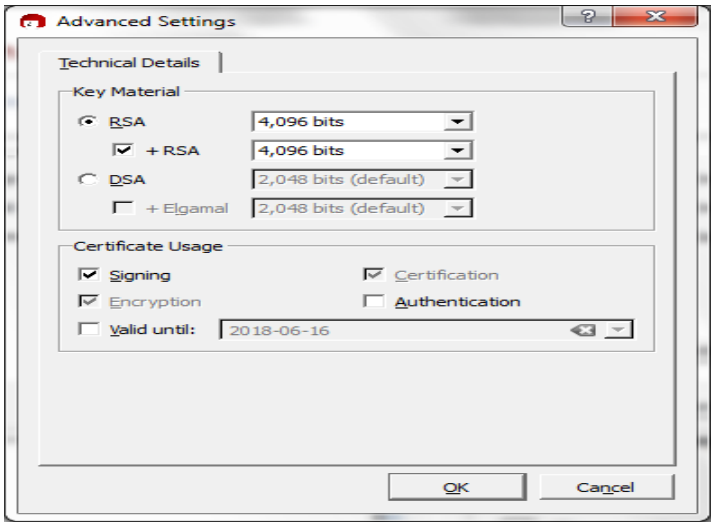
Step 3: The Certificate Creation Wizard should pop up, click on 'Create a personal OpenPGP key pair'



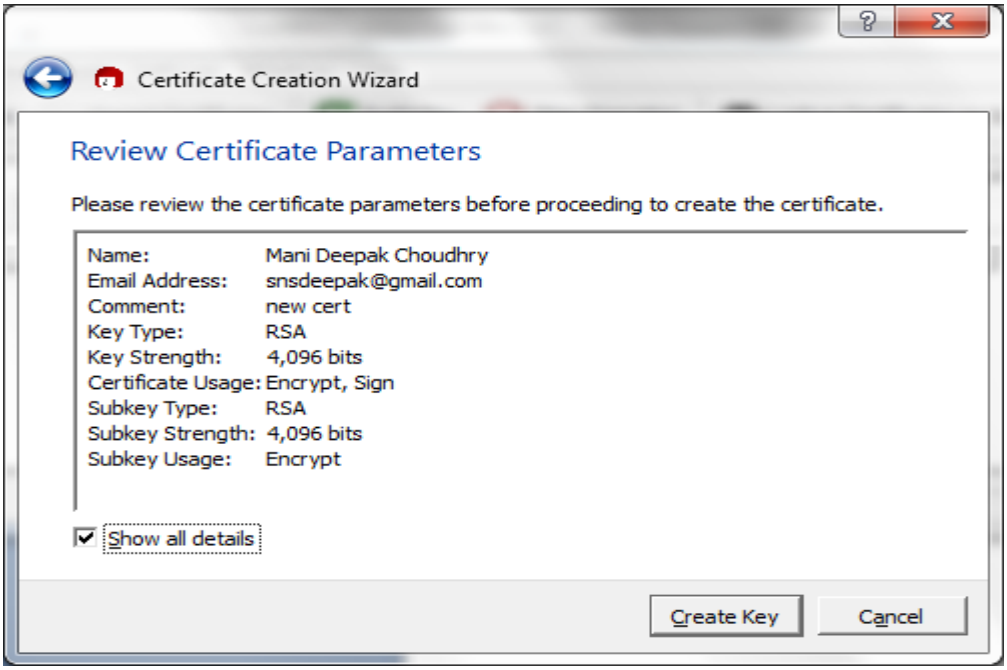
Step 4: Now you'll enter your details. Use your marketplace username as 'Name', and fill out the rest with whatever you want. You don't need to use a real email. Check the picture for an example on how it should look.



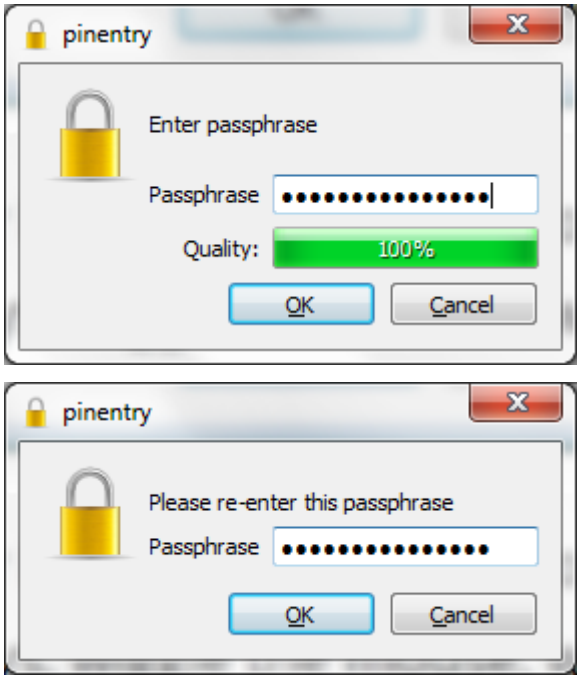
Step 5: Click 'Advanced Settings…', and another window should appear. Under 'Key Material', make sure 'RSA' is checked. In the drop down menu beside it, and select '4,096 bits'. Check the picture to confirm you have everything set correctly, then click 'Ok'



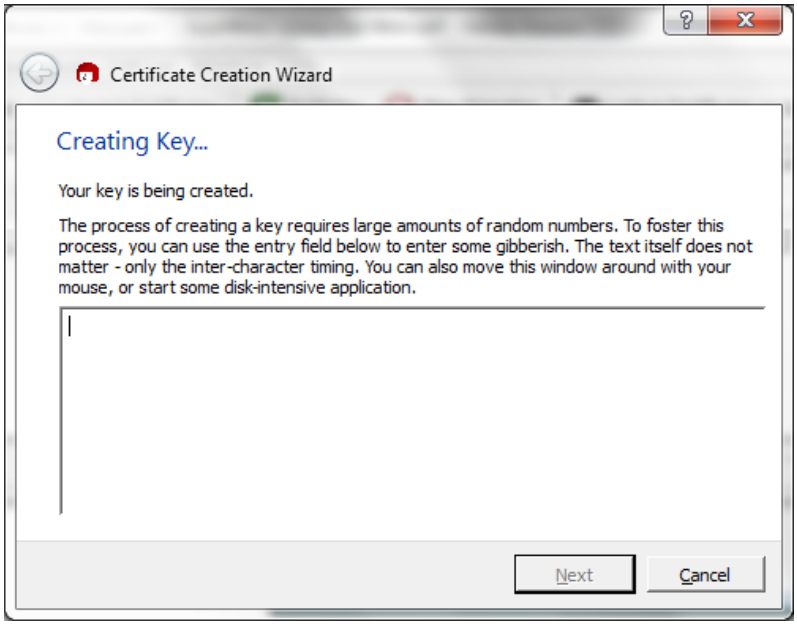
Step 6: Confirm you filled out all of your info correctly, then click ‘Create Key’



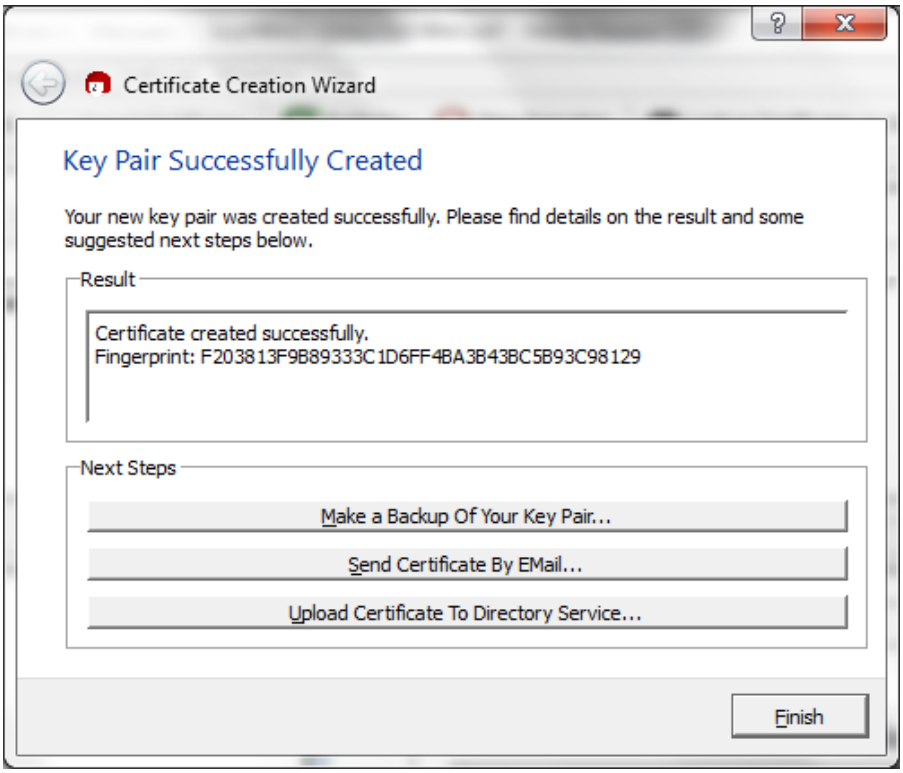
Step 7: Another window will pop up asking to enter a passphrase. Do so, then click ‘Ok’



Step 8: It will now generate your key. It will need you to do random things to create entropy. Mash keys, wiggle the mouse, watch porn, download torrents, whatever

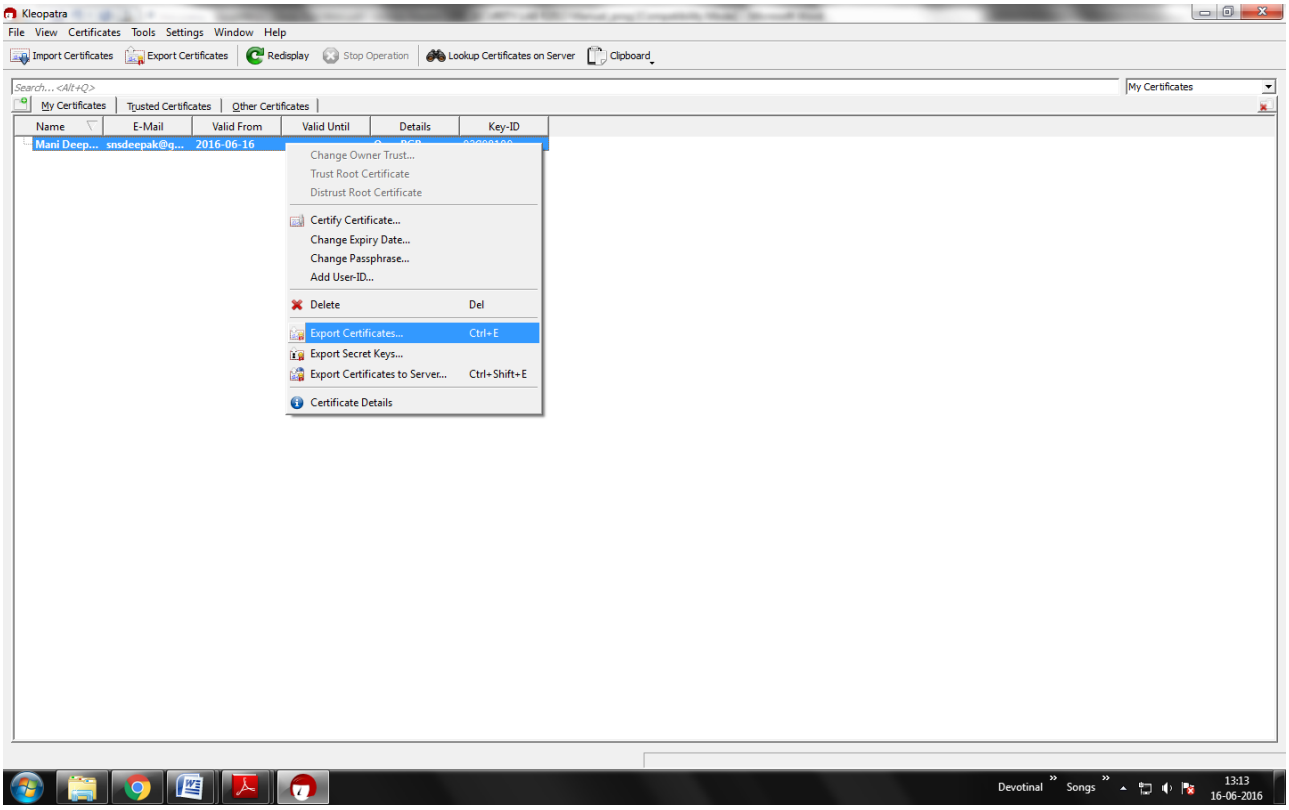


Step 9: Your key is now created. Go ahead and click 'Finish'

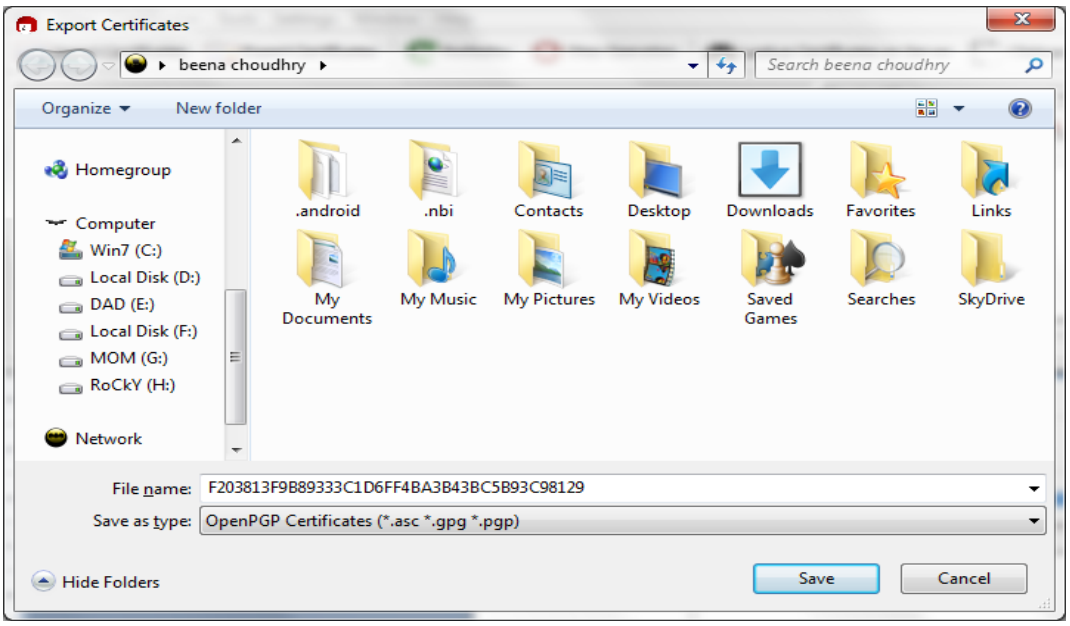


OBTAINING YOUR PUBLIC KEY

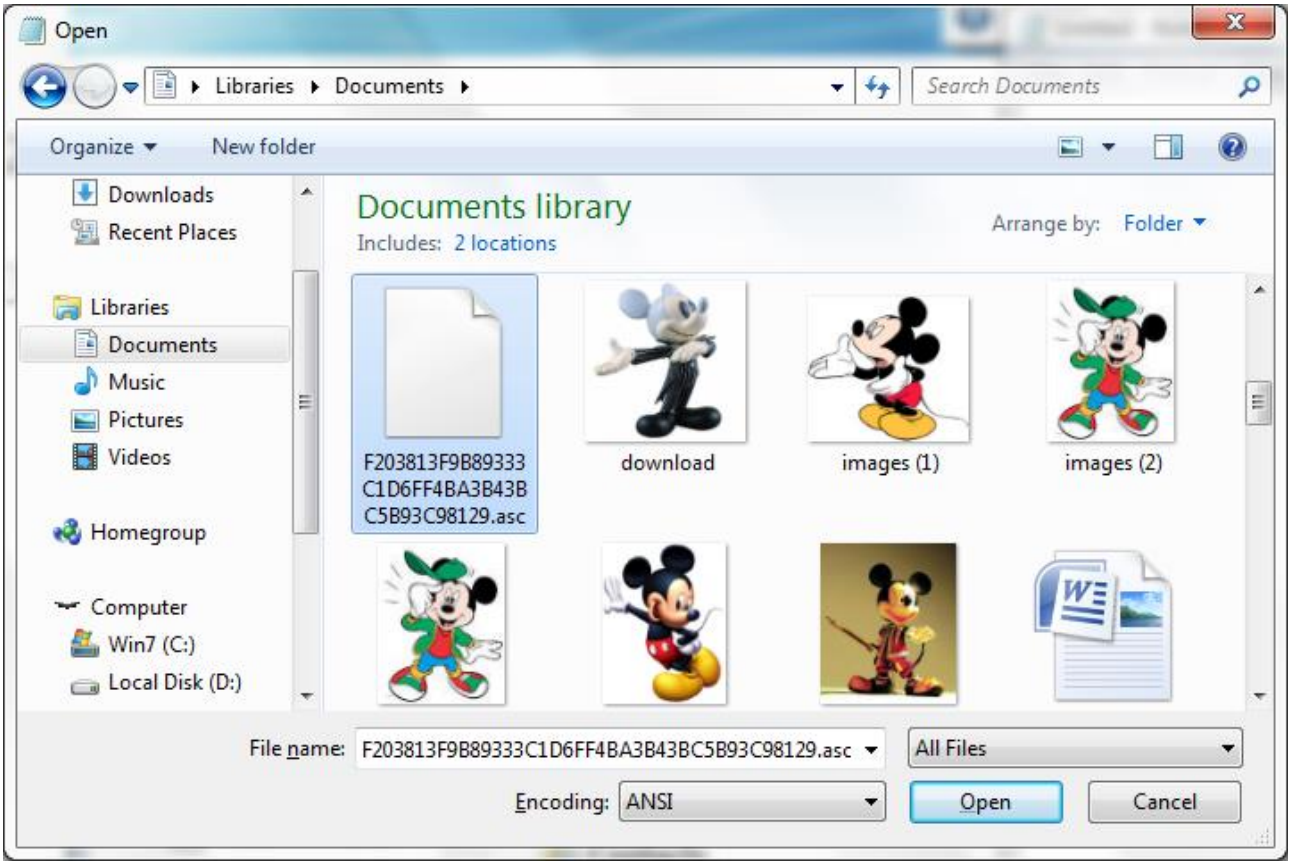
Step 1: Right click on your key, then click ‘Export Certificates…’



Step 2: Browse where you want to save, give it a name, then click ‘Save’

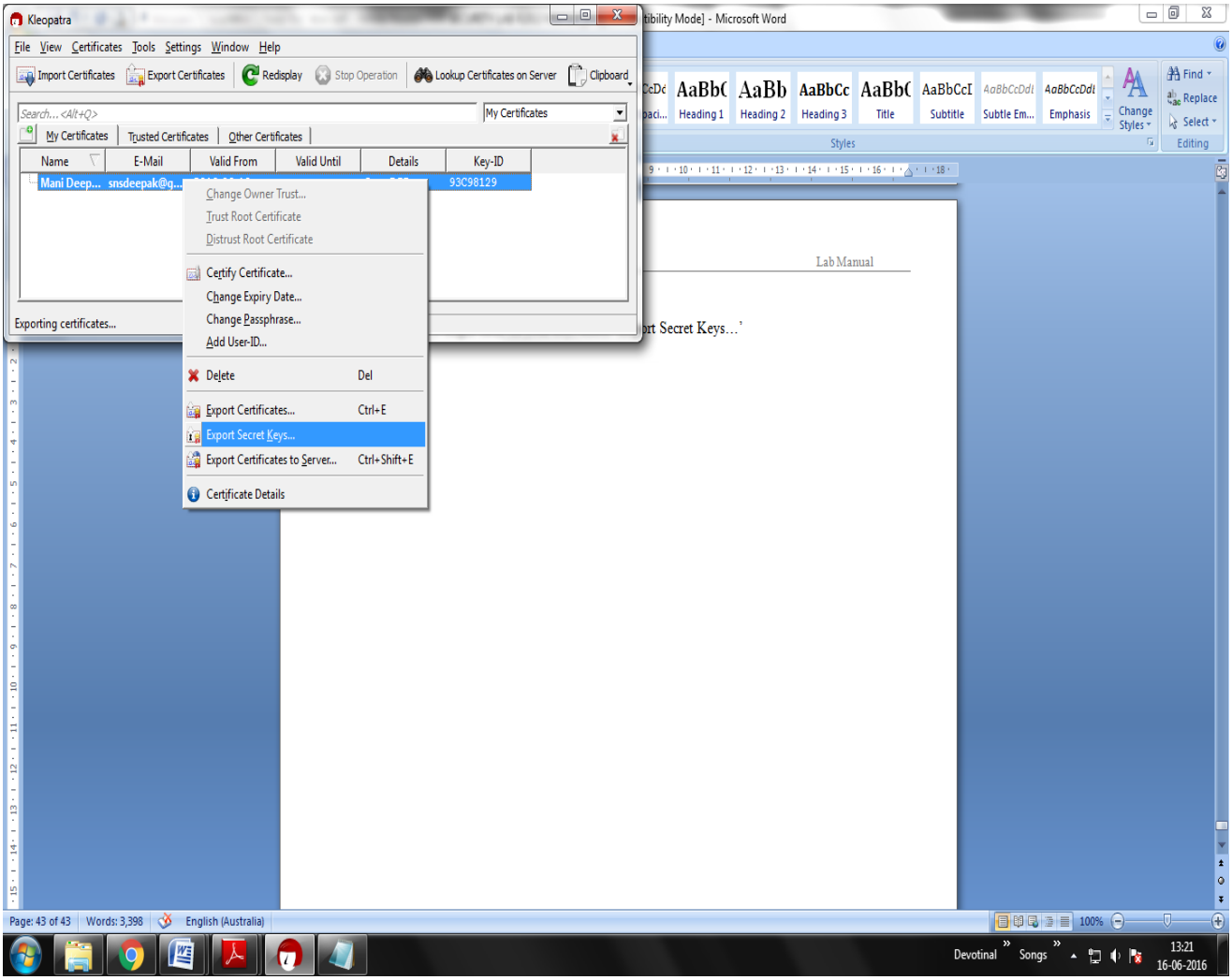


Step 3: Open your favourite text editor, browse to where the file is saved. You may have to select ‘All files’ from the dropdown menu. Click the file you saved, then open

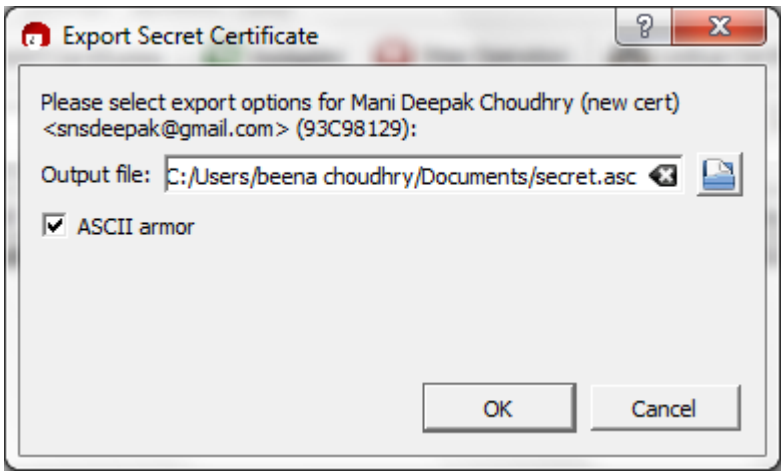


OBTAINING PRIVATE KEY

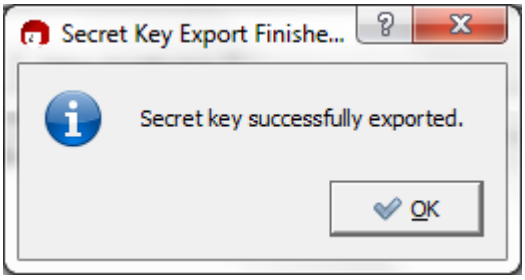
Step 1: Right click on your key, select ‘Export Secret Keys…’



Step 2: Select where you want it saved, give it a name, check ‘ASCII armor’, and click ‘Ok’



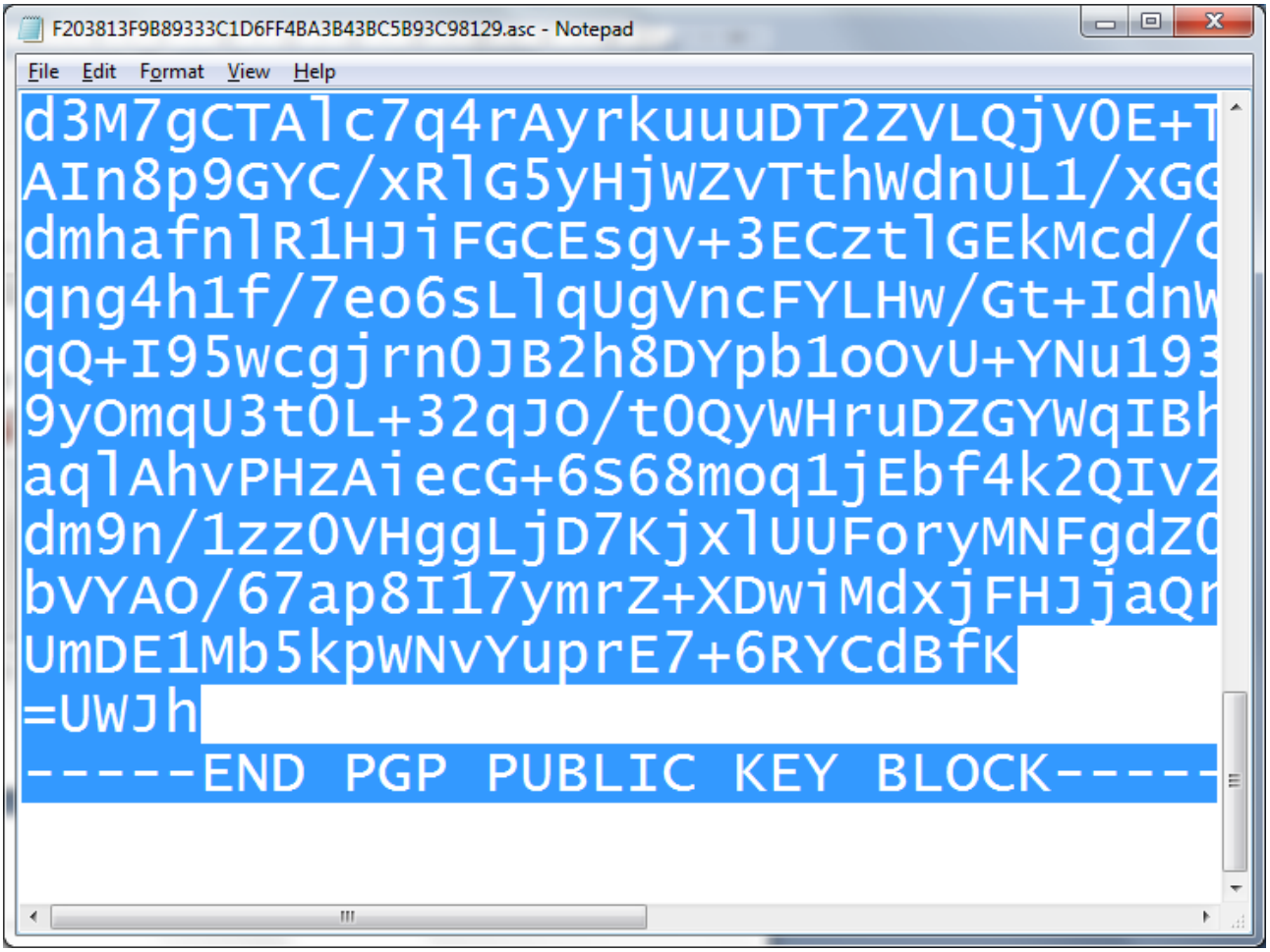
Step 3: You now have your private key



IMPORTING A PUBLIC KEY

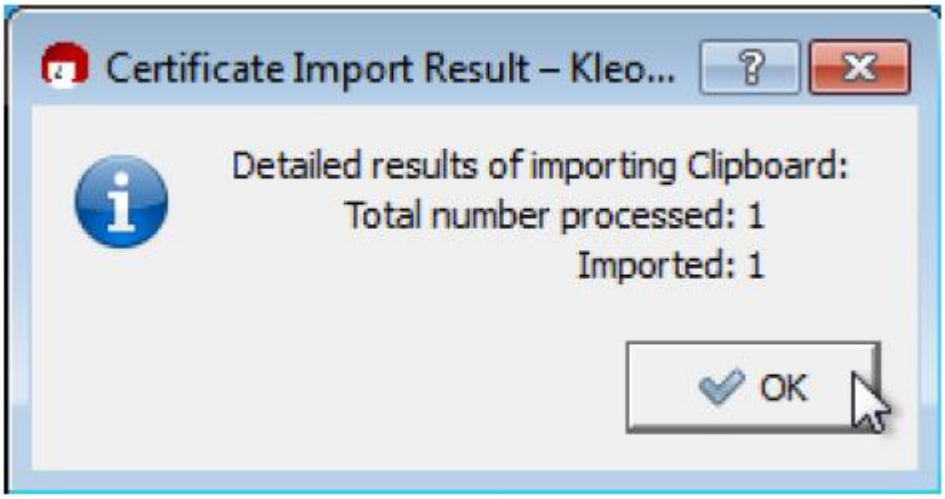
Step 1: Find a public key you want to import.

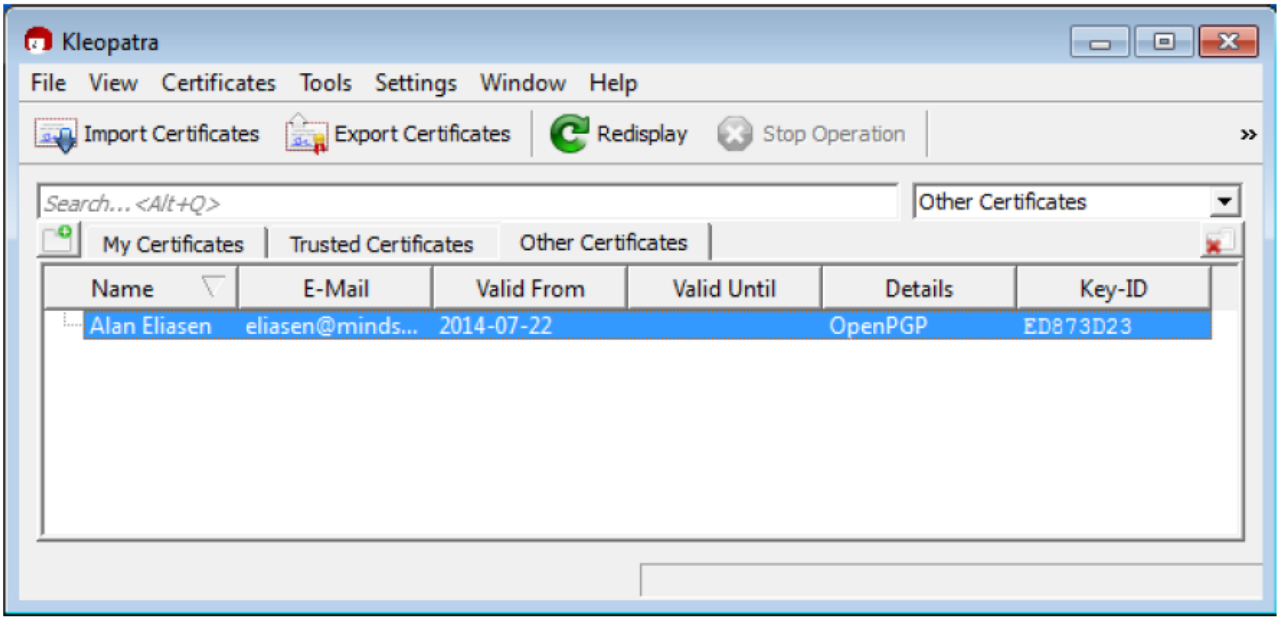
Step 2: Copy everything from '—BEGIN PGP PUBLIC KEY BLOCK—' to '— END PGP PUBLIC KEY BLOCK—'



Step 3: In your task bar, right click on the Kleopatra icon, go to 'Clipboard', then click 'Certificate Import'

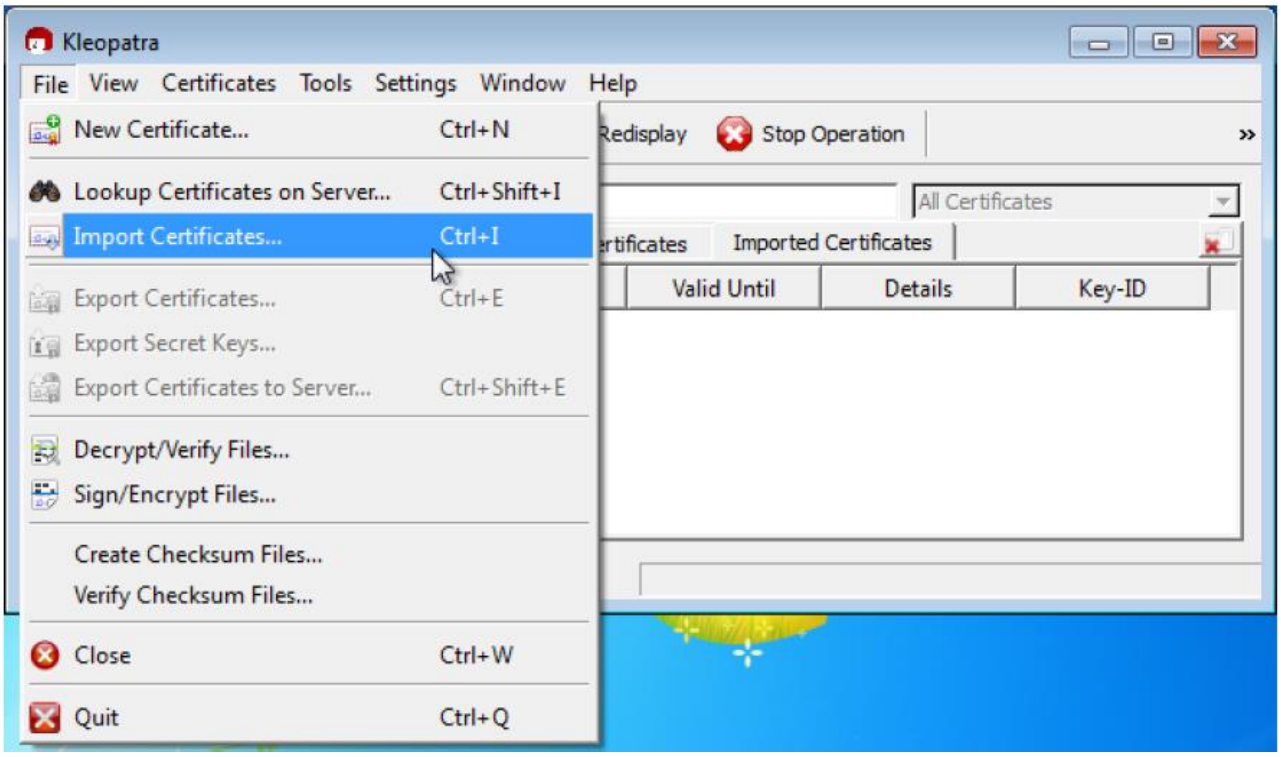
Step 4: If it worked, you should see a window pop up, click ‘Ok’.



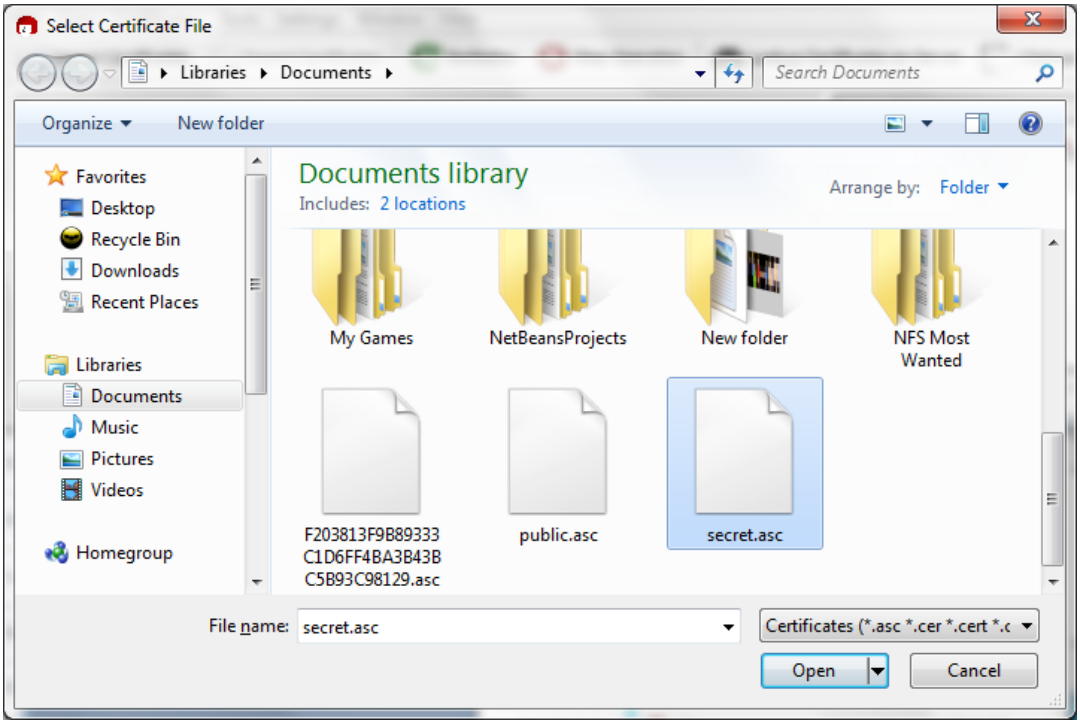


IMPORTING YOUR PRIVATE KEY

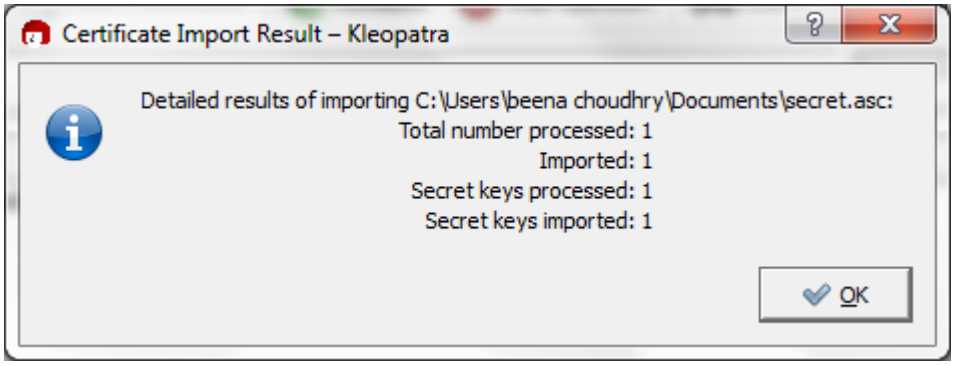
Step 1: Go to ‘File’, then click ‘Import Certificates…’



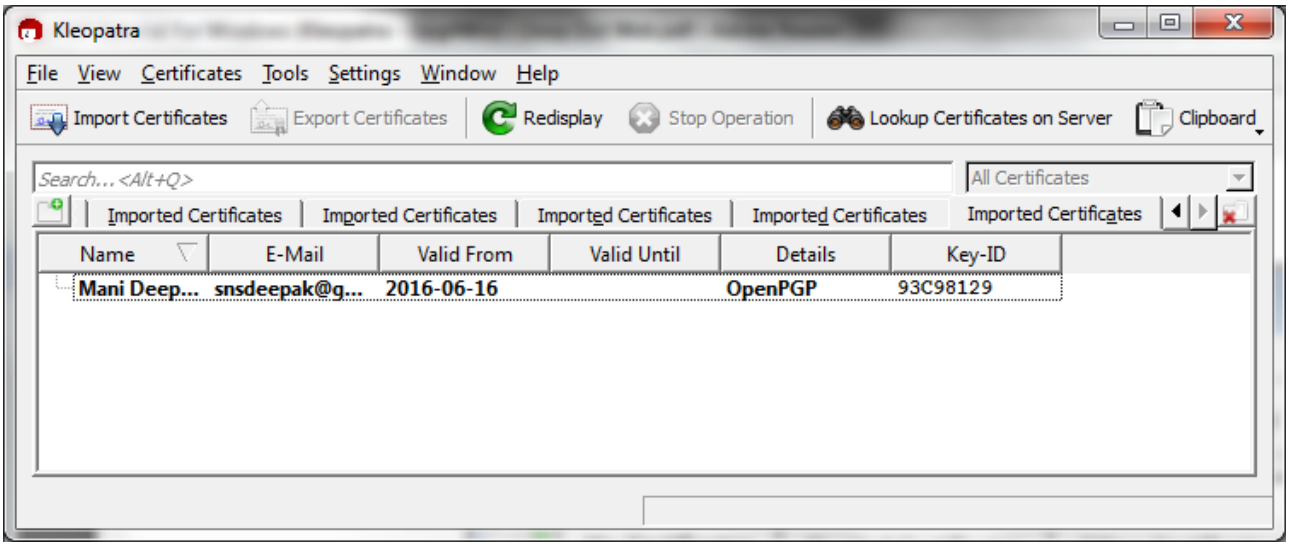
Step 2: Browse to where your private key is, select it, then click ‘Open’



Step 3: It will import your private key, and pop up a window to confirm. Click ‘Ok’



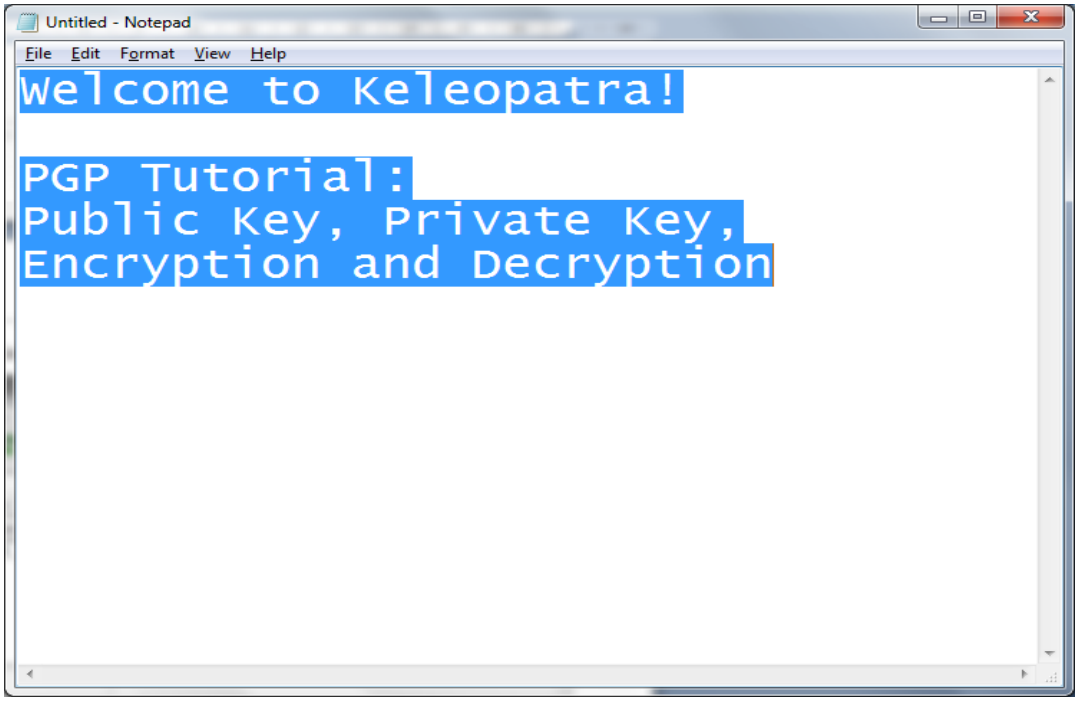
Step 4: You should now see your key information under the ‘My Certificates’ tab



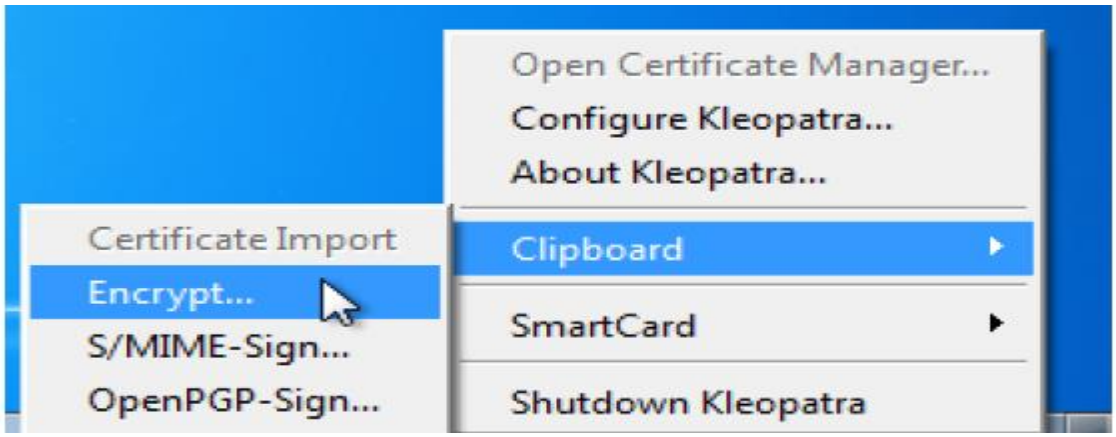
ENCRYPTING A MESSAGE

Step 1: Open up your text editor of choice.

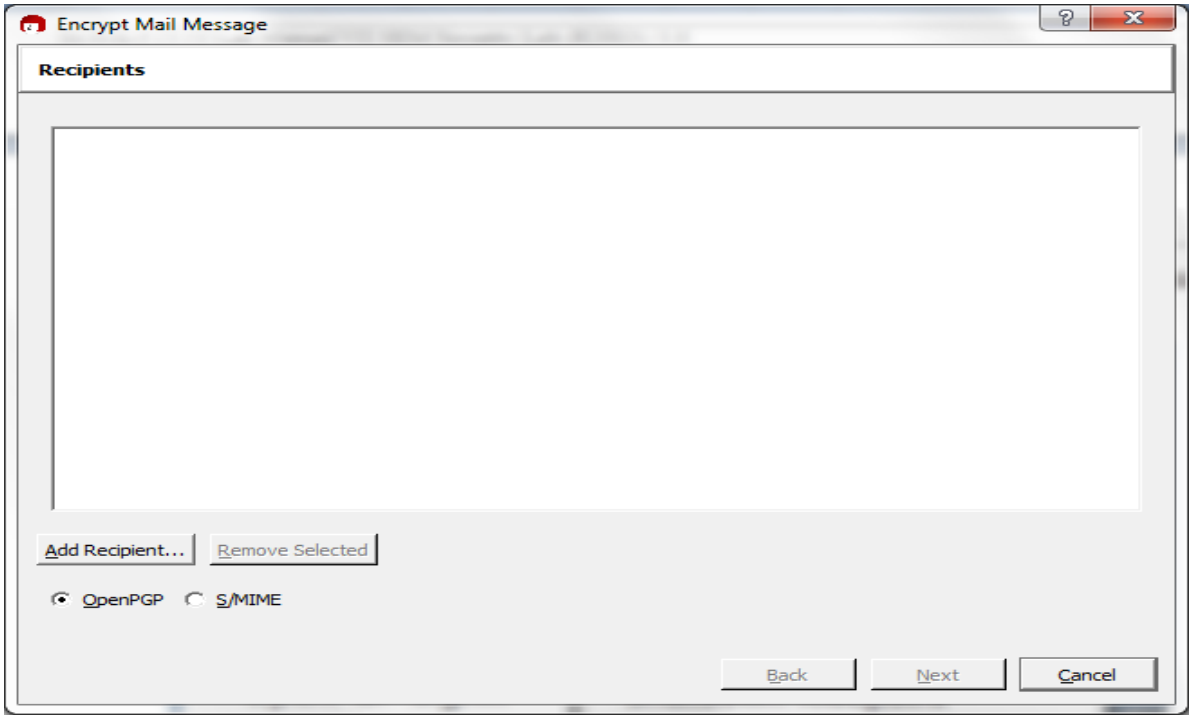
Step 2: Type out your message, select it all, and copy it.



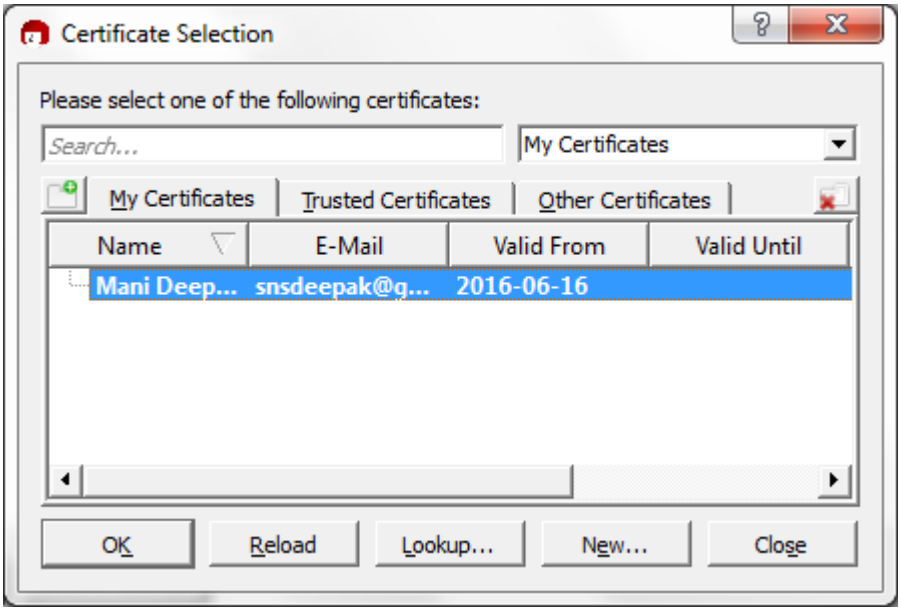
Step 3: In your task bar, right click on the Kleopatra icon, go to 'Clipboard', then click 'Encrypt…'



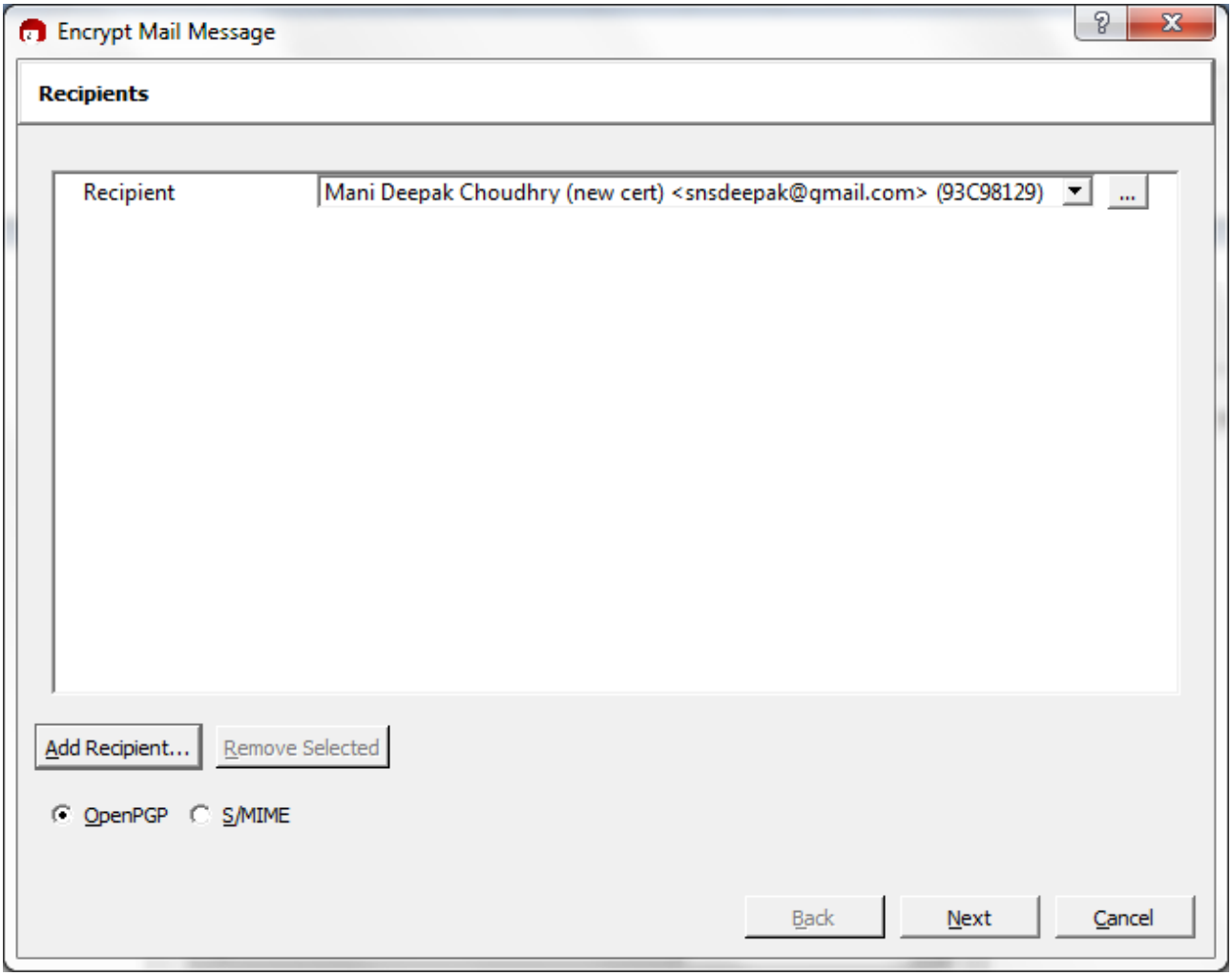
Step 4: window will open. Click 'Add Recipient…'



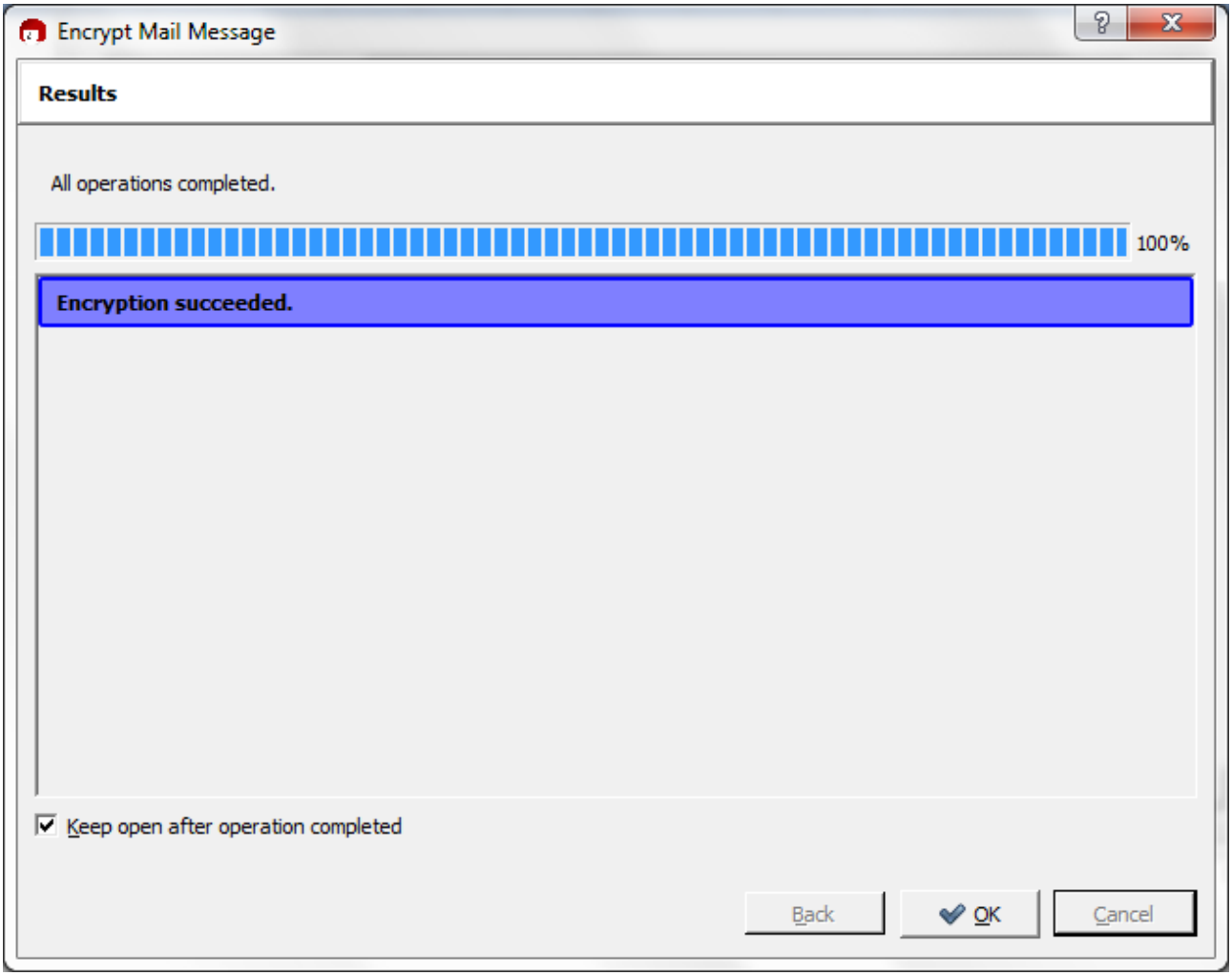
Step 5: Another window will appear. Click the 'Other Certificates' tab, then select who you want to send your message to, then click 'Ok'.



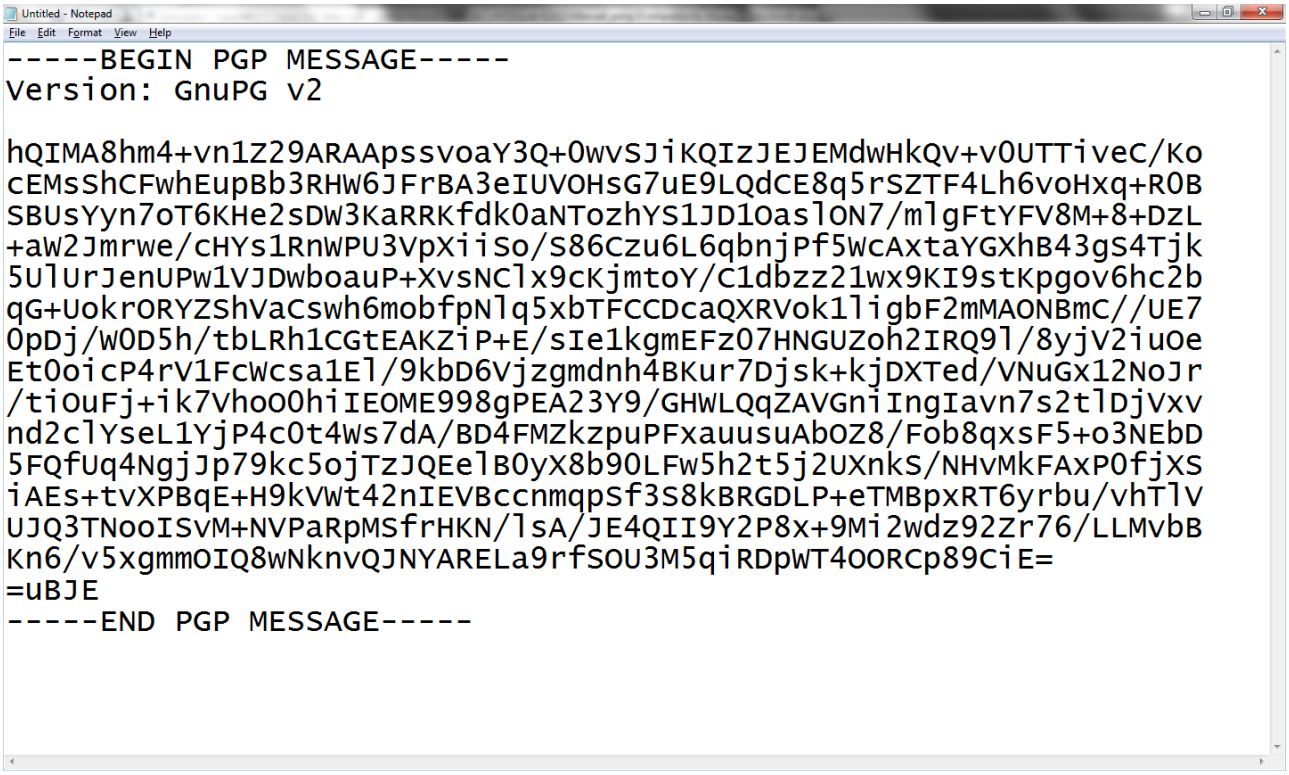
Step 6: You should be back at the previous window with the recipient listed. Click 'Next'



Step 7: If all went well, you should see this window. Click 'Ok'

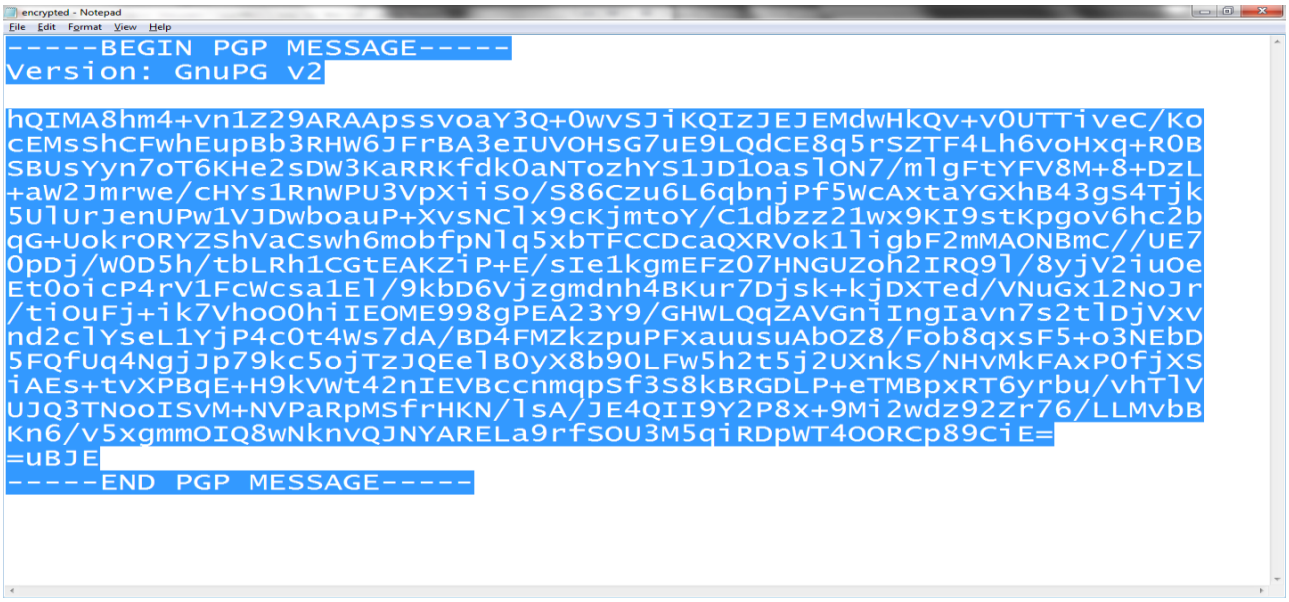


Step 8: Your encrypted message will be in your clipboard, all you need to do is paste it into the message box and send

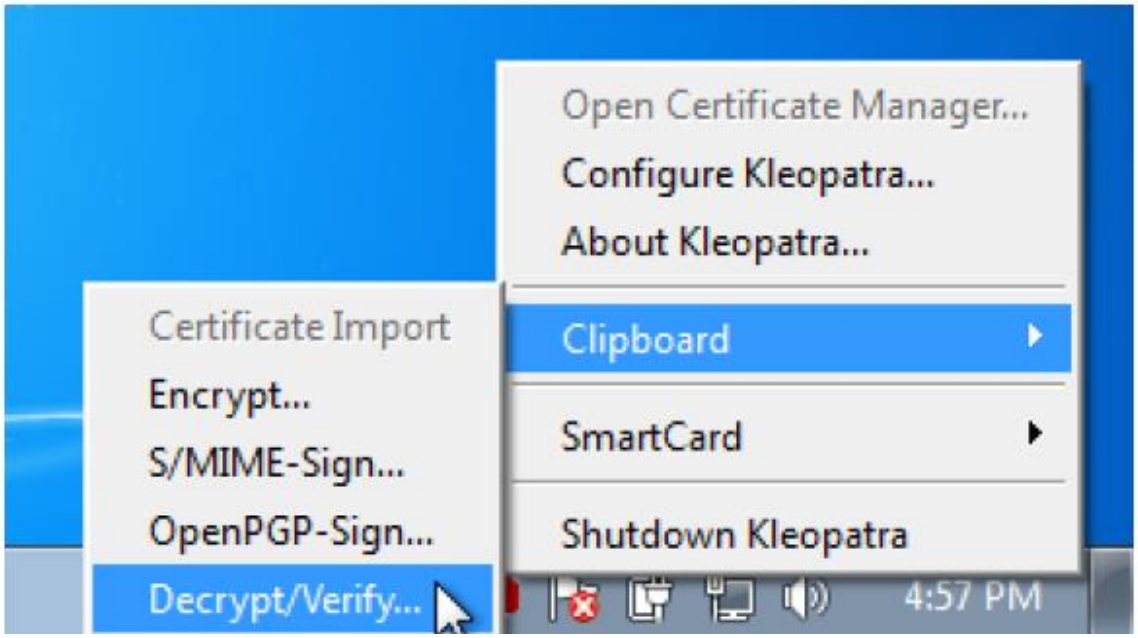


DECRYPTING A MESSAGE

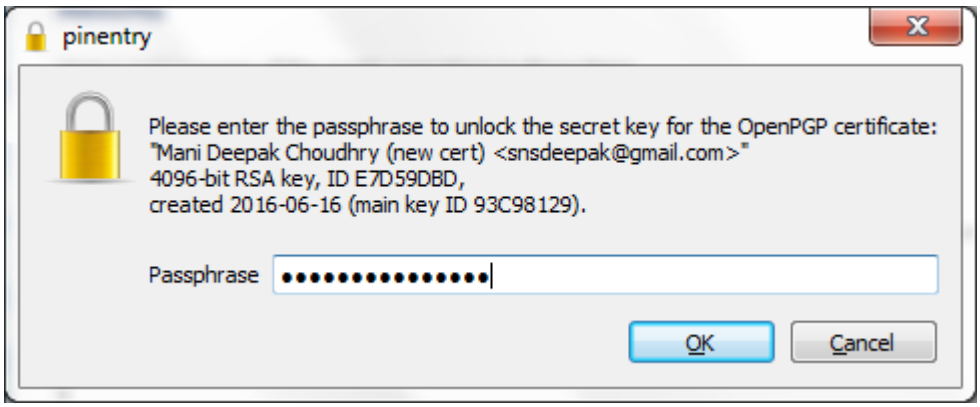
Step 1: Copy everything that was sent.

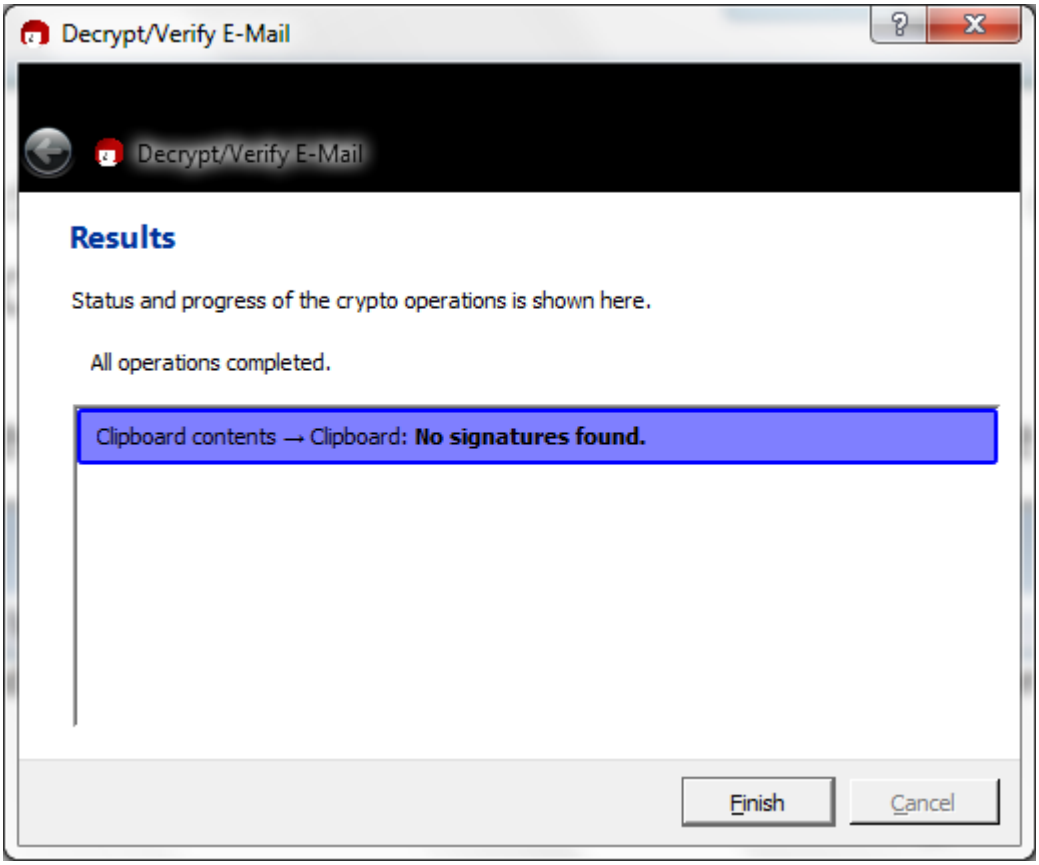


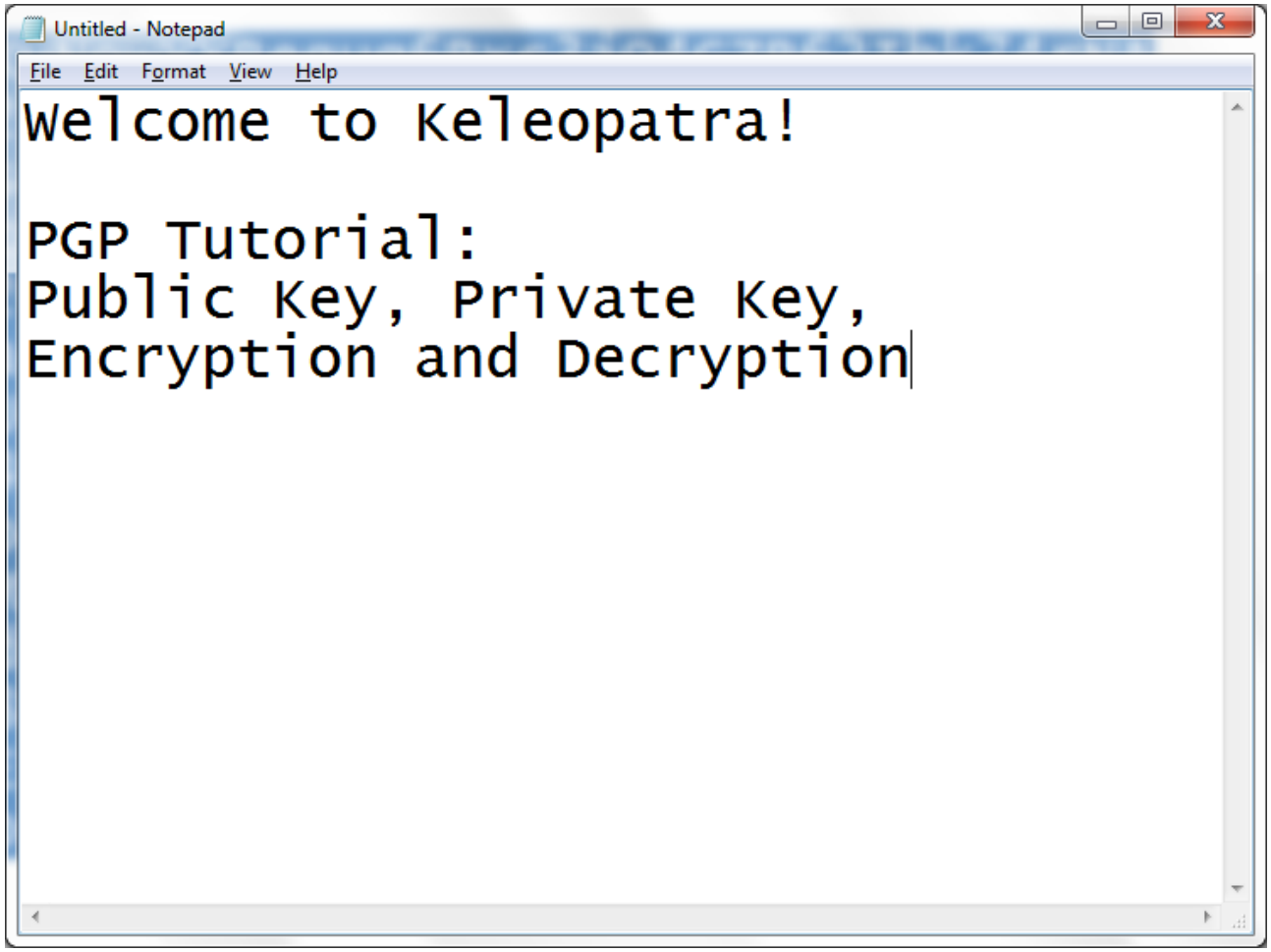
Step 2: In your task bar, right click on the Kleopatra icon, go to ‘Clipboard’, then click ‘Decrypt/Verify…’



Step 3: A window will pop up asking for your passphrase, enter that then click ‘Ok’.




Decrypt/Verify E-Mail
Decrypt/Verify E-Mail
Results
Status and progress of the crypto operations is shown here.
All operations completed.
Clipboard contents → Clipboard: No signatures found.
Finish
Cancel



RESULT

Thus creation of Digital Signature, secure data stirage and transmission was done using Kleopatra Tool using GnuPG was done and output is verified successfully.