OBJECTIVES:

The student should be made to:

- Learn to implement the algorithms DES, RSA,MD5,SHA-1
- Learn to use network security tools like GnuPG, KF sensor, Net Strumbler

LIST OF EXPERIMENTS:

1. Implement the following SUBSTITUTION & TRANSPOSITION TECHNIQUES concepts:
   a) Caesar Cipher
   b) Playfair Cipher
   c) Hill Cipher
   d) Vigenere Cipher
   e) Rail fence – row & Column Transformation
2. Implement the following algorithms
   a) DES
   b) RSA Algorithm
   c) Diffiee-Hellman
   d) MD5
   e) SHA-1
3. Implement the Signature Scheme - Digital Signature Standard
4. Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures (GnuPG)
5. Setup a honey pot and monitor the honeypot on network (KF Sensor)
6. Installation of rootkits and study about the variety of options
7. Perform wireless audit on an access point or a router and decrypt WEP and WPA. (Net Stumbler)
8. Demonstrate intrusion detection system (ids) using any tool (snort or any other s/w)

TOTAL: 45 PERIODS

OUTCOMES:

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

- Implement the cipher techniques
- Develop the various security algorithms
- Use different open source tools for network security and analysis

LIST OF HARDWARE REQUIREMENTS & SOFTWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

- C
- C++
- Java or equivalent compiler GnuPG
- KF Sensor or Equivalent
- Snort
- Net Stumbler or Equivalent

HARDWARE REQUIREMENTS

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

INDEX

EX. NO: 1(A)

IMPLEMENTATION OF CAESAR CIPHER

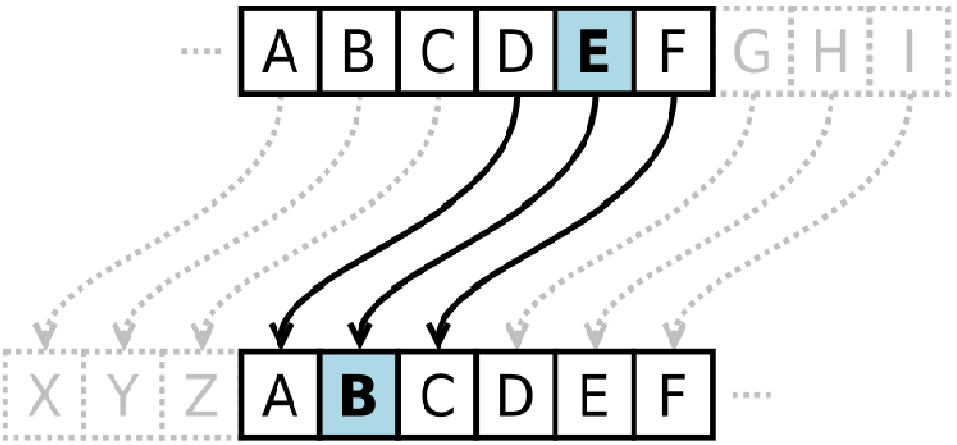
AIM:

To implement the simple substitution technique named Caesar cipher using C language.

DESCRIPTION:

To encrypt a message with a Caesar cipher, each letter in the message is changed using a simple rule: shift by three. Each letter is replaced by the letter three letters ahead in the alphabet. A becomes D, B becomes E, and so on. For the last letters, we can think of the alphabet as a circle and "wrap around". W becomes Z, X becomes A, Y becomes B, and Z becomes C. To change a message back, each letter is replaced by the one three before it.

EXAMPLE:



ALGORITHM:

STEP-1: Read the plain text from the user.

STEP-2: Read the key value from the user.

STEP-3: If the key is positive then encrypt the text by adding the key with each character in the plain text.

STEP-4: Else subtract the key from the plain text.

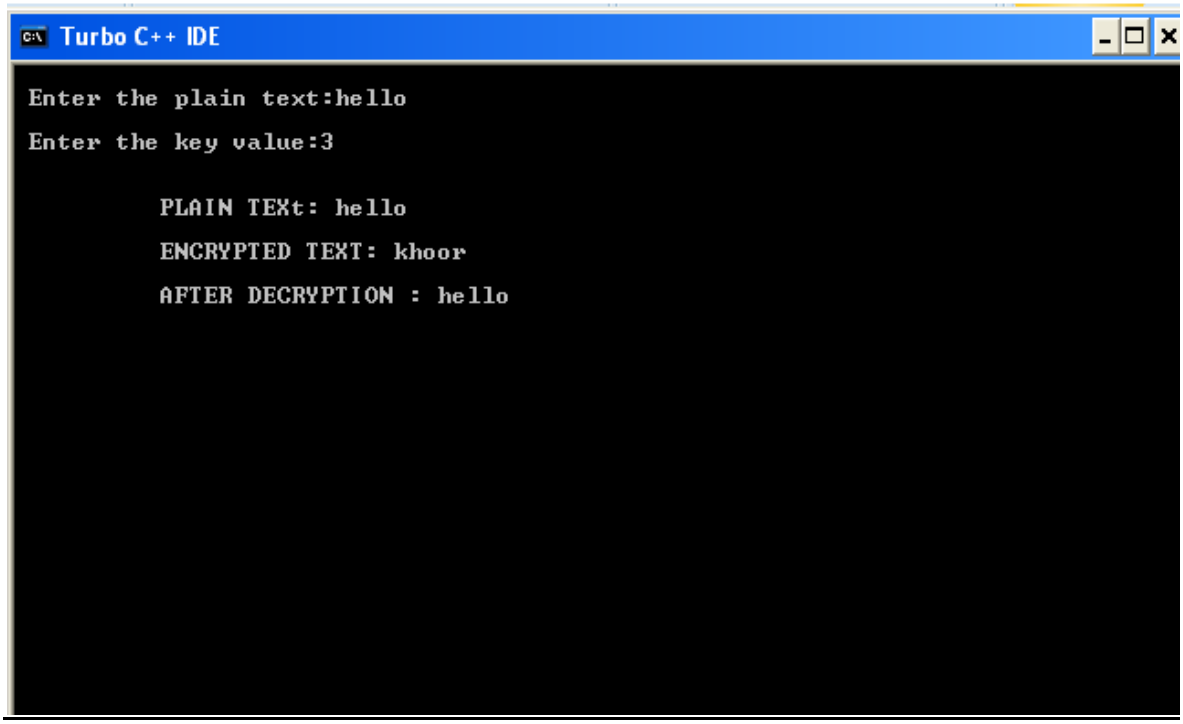
STEP-5: Display the cipher text obtained above.

PROGRAM: (Caesar Cipher)

```
#include <stdio.h>
#include <string.h>
#include<conio.h>
#include <ctype.h>
void main()
```

```
{
 char plain[10], cipher[10];
 int key,i,length;
 int result;
 clrscr();
 printf("\n Enter the plain text:");
 scanf("%s", plain);
 printf("\n Enter the key value:");
 scanf("%d", &key);
 printf("\n \n \t  PLAIN TEXt: %s",plain);
 printf("\n \n \t  ENCRYPTED TEXT: ");
 for(i = 0, length = strlen(plain); i < length; i++)
     {
     cipher[i]=plain[i] + key;
     if (isupper(plain[i]) && (cipher[i] > 'Z'))
     cipher[i] = cipher[i] - 26;
     if (islower(plain[i]) && (cipher[i] > 'z'))
     cipher[i] = cipher[i] - 26;
     printf("%c", cipher[i]);
     }
     printf("\n \n \t  AFTER DECRYPTION : ");
 for(i=0;i<length;i++)
     {
      plain[i]=cipher[i]-key;
     if(isupper(cipher[i])&&(plain[i]<'A'))
     plain[i]=plain[i]+26;
     if(islower(cipher[i])&&(plain[i]<'a'))
     plain[i]=plain[i]+26;
     printf("%c",plain[i]);
 }
getch();
}
```

OUTPUT:



RESULT:

Thus the implementation of Caesar cipher had been executed successfully.

EX. NO: 1(B)

IMPLEMENTATION OF PLAYFAIR CIPHER

AIM:

To write a C program to implement the Playfair Substitution technique.

DESCRIPTION:

The Playfair cipher starts with creating a key table. The key table is a 5×5 grid of letters that will act as the key for encrypting your plaintext. Each of the 25 letters must be unique and one letter of the alphabet is omitted from the table (as there are 25 spots and 26 letters in the alphabet).

To encrypt a message, one would break the message into digrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. The two letters of the diagram are considered as the opposite corners of a rectangle in the key table. Note the relative position of the corners of this rectangle. Then apply the following 4 rules, in order, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively
4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair.

EXAMPLE:

D. Playfair Cipher

Example1: Plaintext: CRYPTO IS TOO EASY **Key =** INFOSEC **Ciphertext:** ??

Grouped text: CR YP TO IS TO XO EA SY

Ciphertext: AQ TV YB NI YB YF CB OZ

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

ALGORITHM:

STEP-1: Read the plain text from the user.

STEP-2: Read the keyword from the user.

STEP-3: Arrange the keyword without duplicates in a 5*5 matrix in the row order and fill the remaining cells with missed out letters in alphabetical order. Note that 'i' and 'j' takes the same cell.

STEP-4: Group the plain text in pairs and match the corresponding corner letters by forming a rectangular grid.

STEP-5: Display the obtained cipher text.

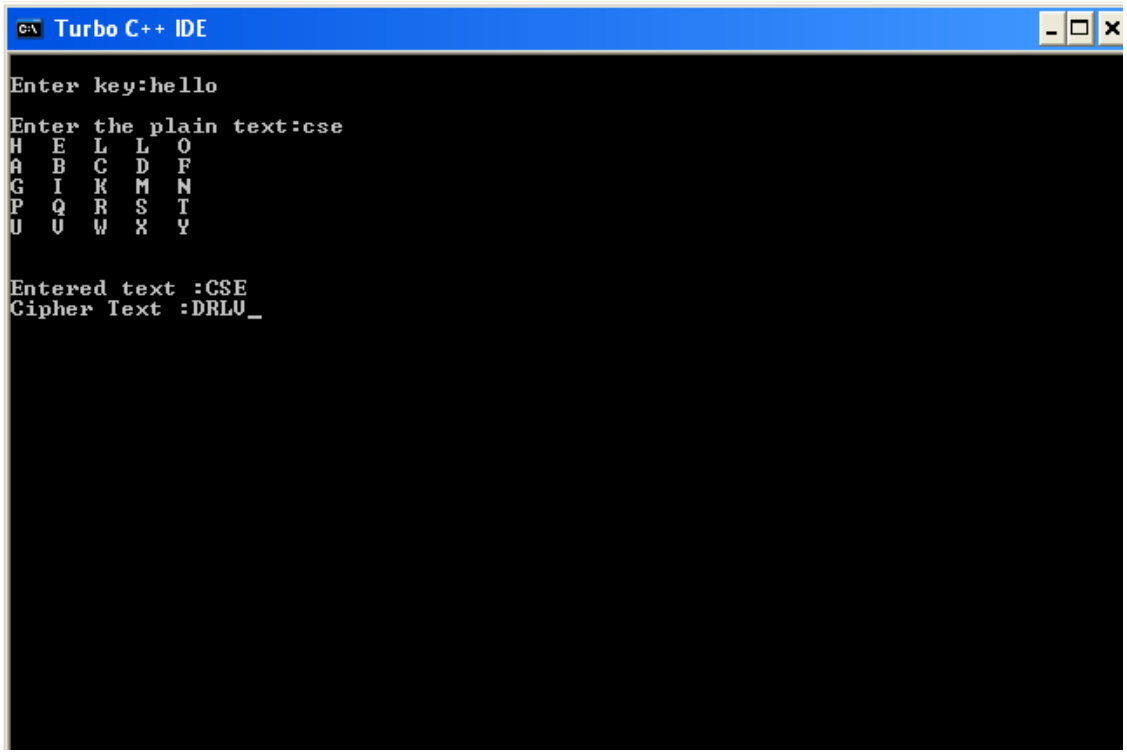
PROGRAM: (Playfair Cipher)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#define MX 5
void playfair(char ch1,char ch2, char key[MX][MX])
{
     int i,j,w,x,y,z;
     FILE *out;
if((out=fopen("cipher.txt","a+"))==NULL)
{
     printf("File Currupted.");
}
for(i=0;i<MX;i++)
{
     for(j=0;j<MX;j++)
     {
     if(ch1==key[i][j])
     {
     w=i;
     x=j;
     }
     else if(ch2==key[i][j])
     {
     y=i;
     z=j;
}}}
//printf("%d%d %d%d",w,x,y,z);
if(w==y)
{
     x=(x+1)%5;z=(z+1)%5;
     printf("%c%c",key[w][x],key[y][z]);
     fprintf(out, "%c%c",key[w][x],key[y][z]);
}
else if(x==z)
{
```

```
        w=(w+1)%5;y=(y+1)%5;
        printf("%c%c",key[w][x],key[y][z]);
        fprintf(out, "%c%c",key[w][x],key[y][z]);
    }
    else
    {
        printf("%c%c",key[w][z],key[y][x]);
        fprintf(out, "%c%c",key[w][z],key[y][x]);
    }
    fclose(out);
    }
    void main()
    {
        int i,j,k=0,l,m=0,n;
        char key[MX][MX],keyminus[25],keystr[10],str[25]={0};
        char
        alpa[26]={'A','B','C','D','E','F','G','H','I','J','K','L'
        ,'M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'}
        ;
        clrscr();
        printf("\nEnter key:");
        gets(keystr);
        printf("\nEnter the plain text:");
        gets(str);
        n=strlen(keystr);
        //convert the characters to uppertext
        for (i=0; i<n; i++)
        {
            if(keystr[i]=='j')keystr[i]='i';
            else if(keystr[i]=='J')keystr[i]='I';
            keystr[i] = toupper(keystr[i]);
        }
        //convert all the characters of plaintext  to uppertext
        for (i=0; i<strlen(str); i++)
        {
            if(str[i]=='j')str[i]='i';
            else if(str[i]=='J')str[i]='I';
            str[i] = toupper(str[i]);
        }
        j=0;
        for(i=0;i<26;i++)
        {
            for(k=0;k<n;k++)
            {
            if(keystr[k]==alpa[i])
            break;
            else if(alpa[i]=='J')
            break;
            }
            if(k==n)
            {
            keyminus[j]=alpa[i];j++;
            }
        }
```

```
      //construct key keymatrix
      k=0;
      for(i=0;i<MX;i++)
      {
            for(j=0;j<MX;j++)
            {
            if(k<n)
            {
            key[i][j]=keystr[k];
            k++;}
            else
            {
            key[i][j]=keyminus[m];m++;
            }
            printf("%c  ",key[i][j]);
            }
            printf("\n");
      }
      printf("\n\nEntered text :%s\nCipher Text :",str);
      for(i=0;i<strlen(str);i++)
      {
      if(str[i]=='J')str[i]='I';
      if(str[i+1]=='\0')
      playfair(str[i],'X',key);
      else
      {
            if(str[i+1]=='J')str[i+1]='I';
            if(str[i]==str[i+1])
            playfair(str[i],'X',key);
            else
            {
            playfair(str[i],str[i+1],key);i++;
            }}
      }
getch();
}
```

OUTPUT:



RESULT:

Thus the Playfair cipher substitution technique had been implemented successfully.

EX. NO: 1(C)

IMPLEMENTATION OF HILL CIPHER

AIM:

To write a C program to implement the hill cipher substitution techniques.

DESCRIPTION:

Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1... Z = 25, is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters is multiplied by an invertible $n \times n$ matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible $n \times n$ matrices (modulo 26).

EXAMPLE:

$$\begin{bmatrix} 2 & 4 & 5 \\ 9 & 2 & 1 \\ 3 & 17 & 7 \end{bmatrix} \begin{bmatrix} 0 \\ 19 \\ 19 \end{bmatrix} = \begin{bmatrix} 171 \\ 57 \\ 456 \end{bmatrix} \pmod{26} = \begin{bmatrix} 15 \\ 5 \\ 14 \end{bmatrix} = \text{'PFO'}$$

ALGORITHM:

STEP-1: Read the plain text and key from the user.

STEP-2: Split the plain text into groups of length three.

STEP-3: Arrange the keyword in a 3*3 matrix.

STEP-4: Multiply the two matrices to obtain the cipher text of length three.

STEP-5: Combine all these groups to get the complete cipher text.

PROGRAM: (Hill Cipher)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main(){
     unsigned int a[3][3]={{6,24,1},{13,16,10},{20,17,15}};
     unsigned int b[3][3]={{8,5,10},{21,8,21},{21,12,8}};
     int i,j, t=0;
     unsigned int c[20],d[20];
     char msg[20];
     clrscr();
     printf("Enter plain text\n ");
     scanf("%s",msg);
     for(i=0;i<strlen(msg);i++)
     {    c[i]=msg[i]-65;
```

```
            printf("%d ",c[i]);
      }
      for(i=0;i<3;i++)
      {    t=0;
      for(j=0;j<3;j++)
      {
            t=t+(a[i][j]*c[j]);
      }
            d[i]=t%26;
      }
printf("\nEncrypted Cipher Text :");
for(i=0;i<3;i++)
printf(" %c",d[i]+65);
for(i=0;i<3;i++)
{
      t=0;
      for(j=0;j<3;j++)
      {
            t=t+(b[i][j]*d[j]);
      }
      c[i]=t%26;
}
printf("\nDecrypted Cipher Text :");
for(i=0;i<3;i++)
printf(" %c",c[i]+65);
getch();
return 0;
}
```

OUTPUT:

RESULT:

Thus the hill cipher substitution technique had been implemented successfully in C.

EX. NO: 1(D)

IMPLEMENTATION OF VIGENERE CIPHER

AIM:

To implement the Vigenere Cipher substitution technique using C program.

DESCRIPTION:

To encrypt, a table of alphabets can be used, termed a tabula recta, Vigenère square, or Vigenère table. It consists of the alphabet written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

Each row starts with a key letter. The remainder of the row holds the letters A to Z. Although there are 26 key rows shown, you will only use as many keys as there are unique letters in the key string, here just 5 keys, {L, E, M, O, N}. For successive letters of the message, we are going to take successive letters of the key string, and encipher each message letter using its corresponding key row. Choose the next letter of the key, go along that row to find the column heading that matches the message character; the letter at the intersection of [key-row, msg-col] is the enciphered letter.

EXAMPLE:

ALGORITHM:

STEP-1: Arrange the alphabets in row and column of a 26*26 matrix.

STEP-2: Circulate the alphabets in each row to position left such that the first letter is attached to last.

STEP-3: Repeat this process for all 26 rows and construct the final key matrix.

STEP-4: The keyword and the plain text is read from the user.

STEP-5: The characters in the keyword are repeated sequentially so as to match with that of the plain text.

STEP-6: Pick the first letter of the plain text and that of the keyword as the row indices and column indices respectively.

STEP-7: The junction character where these two meet forms the cipher character.

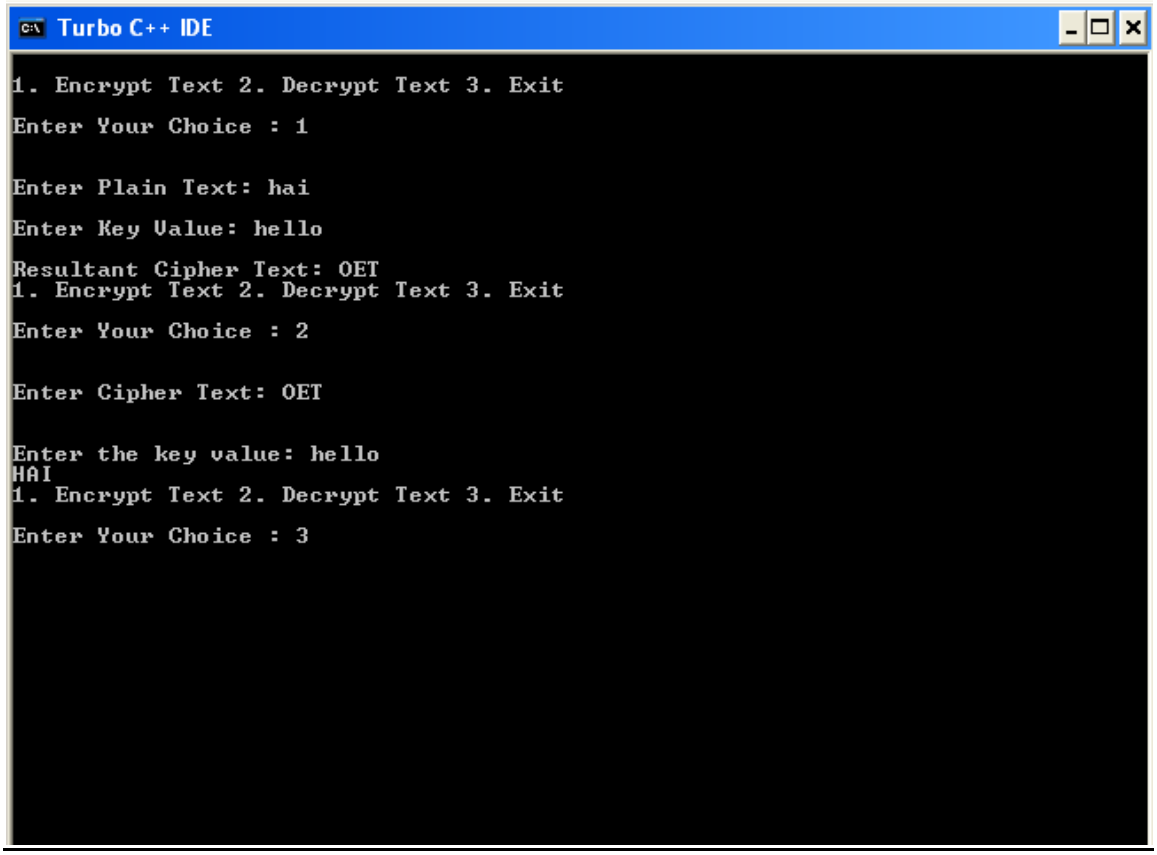
STEP-8: Repeat the above steps to generate the entire cipher text.

PROGRAM: (Vigenere Cipher)

```
#include <stdio.h>
#include<conio.h>
#include <ctype.h>
#include <string.h>
void encipher();
void decipher();
void main()
{
     int choice;
     clrscr();
     while(1)
     {
          printf("\n1. Encrypt Text");
          printf("\t2. Decrypt Text");
          printf("\t3. Exit");
          printf("\n\nEnter Your Choice : ");
          scanf("%d",&choice);
          if(choice == 3)
          exit(0);
          else if(choice == 1)
          encipher();
          else if(choice == 2)
          decipher();
          else
          printf("Please Enter Valid Option.");
     }
}
void encipher()
{
     unsigned int i,j;
     char input[50],key[10];
     printf("\n\nEnter Plain Text: ");
```

```
     scanf("%s",input);
     printf("\nEnter Key Value: ");
     scanf("%s",key);
     printf("\nResultant Cipher Text: ");
     for(i=0,j=0;i<strlen(input);i++,j++)
     {
          if(j>=strlen(key))
               {     j=0;
               }
     printf("%c",65+(((toupper(input[i])-65)+(toupper(key[j])-
     65))%26));
     }}
void decipher()
{
     unsigned int i,j;
     char input[50],key[10];
     int value;
     printf("\n\nEnter Cipher Text: ");
     scanf("%s",input);
     printf("\n\nEnter the key value: ");
     scanf("%s",key);
for(i=0,j=0;i<strlen(input);i++,j++)
{
     if(j>=strlen(key))
     {     j=0;       }
     value = (toupper(input[i])-64)-(toupper(key[j])-64);
     if( value < 0)
     {  value = value * -1;
     }
     printf("%c",65 + (value % 26));
}}
```

OUTPUT:



RESULT:

Thus the Vigenere Cipher substitution technique had been implemented successfully.

EX. NO: 1(E)

IMPLEMENTATION OF RAIL FENCE – ROW & COLUMN TRANSFORMATION TECHNIQUE

AIM:

To write a C program to implement the rail fence transposition technique.

DESCRIPTION:

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

EXAMPLE:

```
A U T H O R
1 6 5 2 3 4
-----------
W E A R E D
I S C O V E
R E D S A V
E Y O U R S
E L F A B C
```

yields the cipher

```
W I R E E R O S U A E V A R B D E V S C A C D O F E S E Y L .
```

ALGORITHM:

STEP-1: Read the Plain text.

STEP-2: Arrange the plain text in row columnar matrix format.

STEP-3: Now read the keyword depending on the number of columns of the plain text.

STEP-4: Arrange the characters of the keyword in sorted order and the corresponding columns of the plain text.

STEP-5: Read the characters row wise or column wise in the former order to get the cipher text.

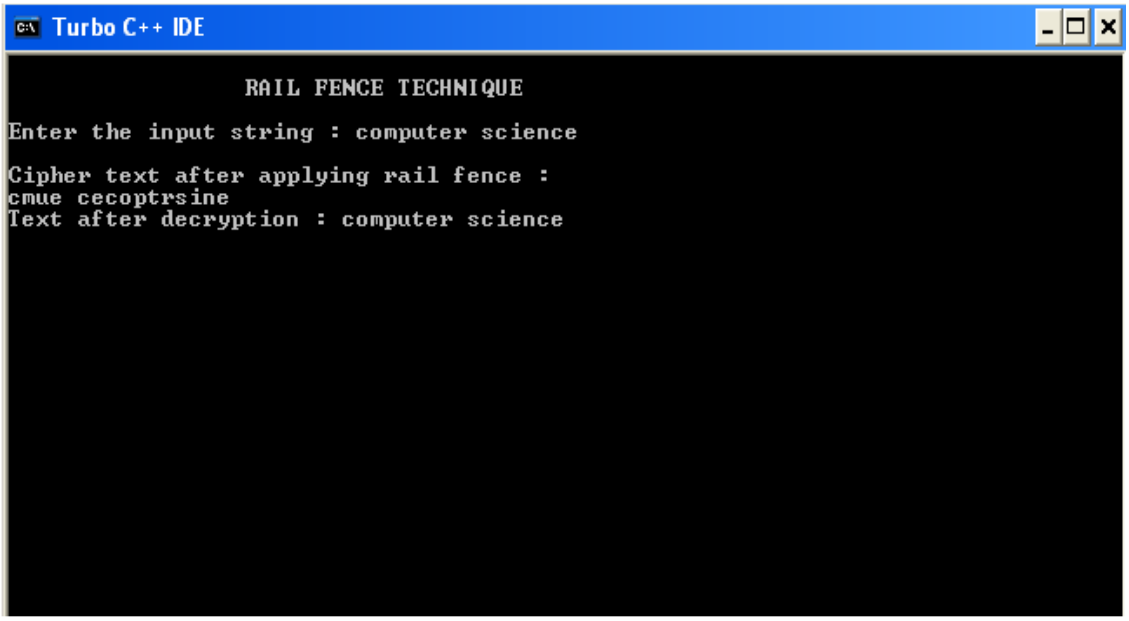
PROGRAM: (Rail Fence)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
     int i,j,k,l;
     char a[20],c[20],d[20];
     clrscr();
     printf("\n\t\t RAIL FENCE TECHNIQUE");
     printf("\n\nEnter the input string : ");
     gets(a);
     l=strlen(a);

/*Ciphering*/
for(i=0,j=0;i<l;i++)
{
     if(i%2==0)
     c[j++]=a[i];
}
for(i=0;i<l;i++)
{
     if(i%2==1)
     c[j++]=a[i];
}
c[j]='\0';
printf("\nCipher text after applying rail fence :");
printf("\n%s",c);

/*Deciphering*/
if(l%2==0)
     k=l/2;
else
     k=(l/2)+1;
for(i=0,j=0;i<k;i++)
{
    d[j]=c[i];
    j=j+2;
}
for(i=k,j=1;i<l;i++)
{
    d[j]=c[i];
    j=j+2;
}
d[l]='\0';
printf("\nText after decryption : ");
printf("%s",d);
getch();
}
```

OUTPUT:



RESULT:

Thus the rail fence algorithm had been executed successfully.

EX. NO: 2(A) **IMPLEMENTATION OF DES**

AIM:

To write a C program to implement Data Encryption Standard (DES) using C Language.

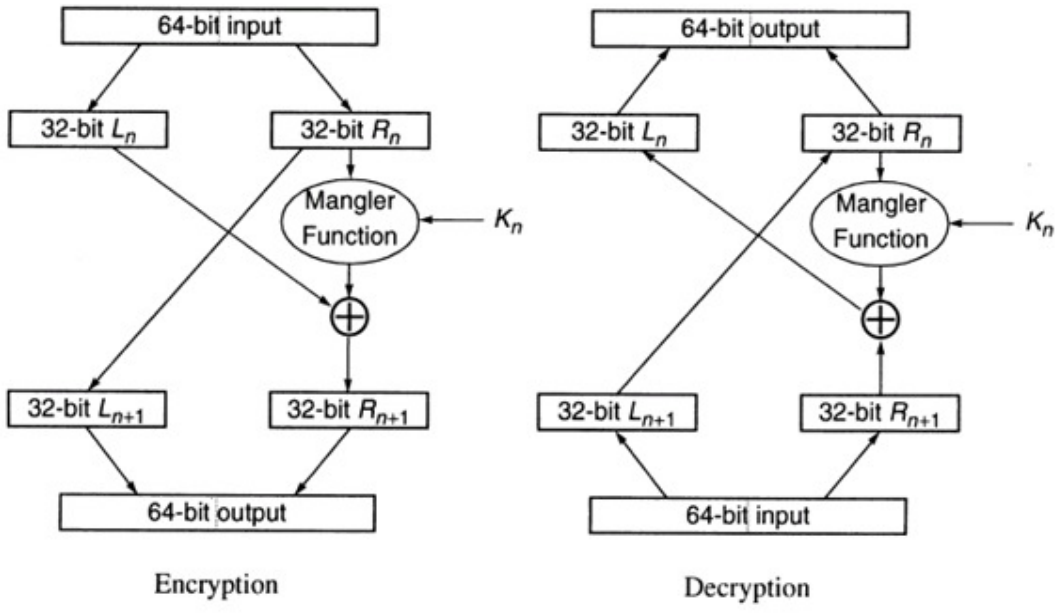
DESCRIPTION:

DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which are used for parity checks. The key therefore has a "useful" length of 56 bits, which means that only 56 bits are actually used in the algorithm. The algorithm involves carrying out combinations, substitutions and permutations between the text to be encrypted and the key, while making sure the operations can be performed in both directions. The key is ciphered on 64 bits and made of 16 blocks of 4 bits, generally denoted k_1 to k_{16}. Given that "only" 56 bits are actually used for encrypting, there can be 2^{56} different keys.

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit blocks
- Initial permutation of blocks
- Breakdown of the blocks into two parts: left and right, named L and R
- Permutation and substitution steps repeated 16 times
- Re-joining of the left and right parts then inverse initial permutation

EXAMPLE:



ALGORITHM:

STEP-1: Read the 64-bit plain text.

STEP-2: Split it into two 32-bit blocks and store it in two different arrays.

STEP-3: Perform XOR operation between these two arrays.

STEP-4: The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.

STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

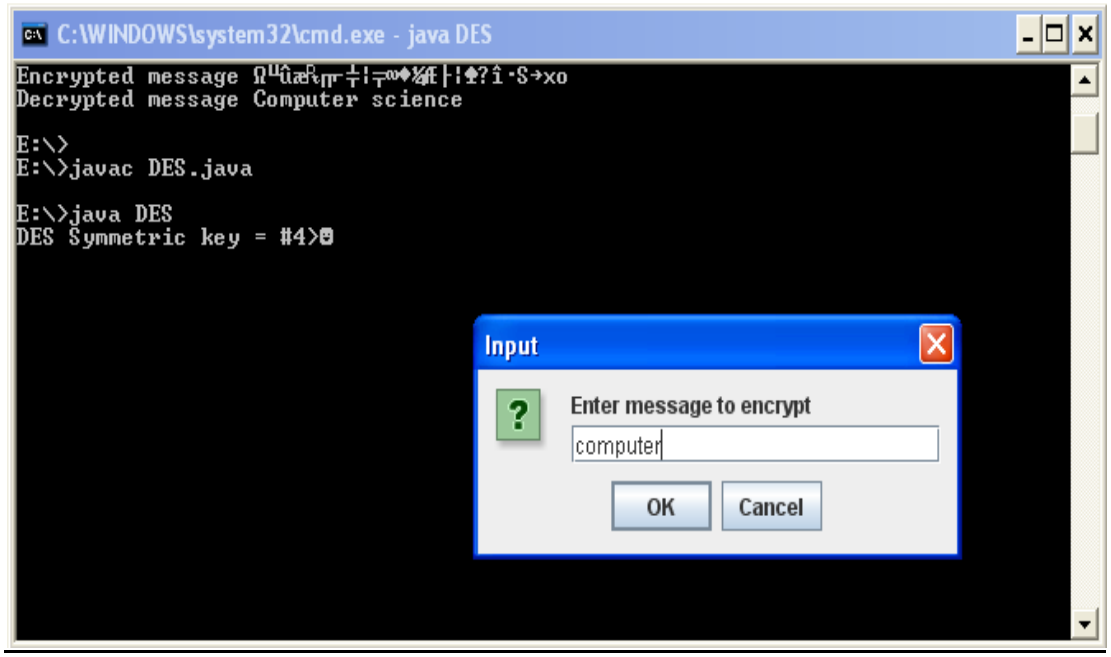
PROGRAM:

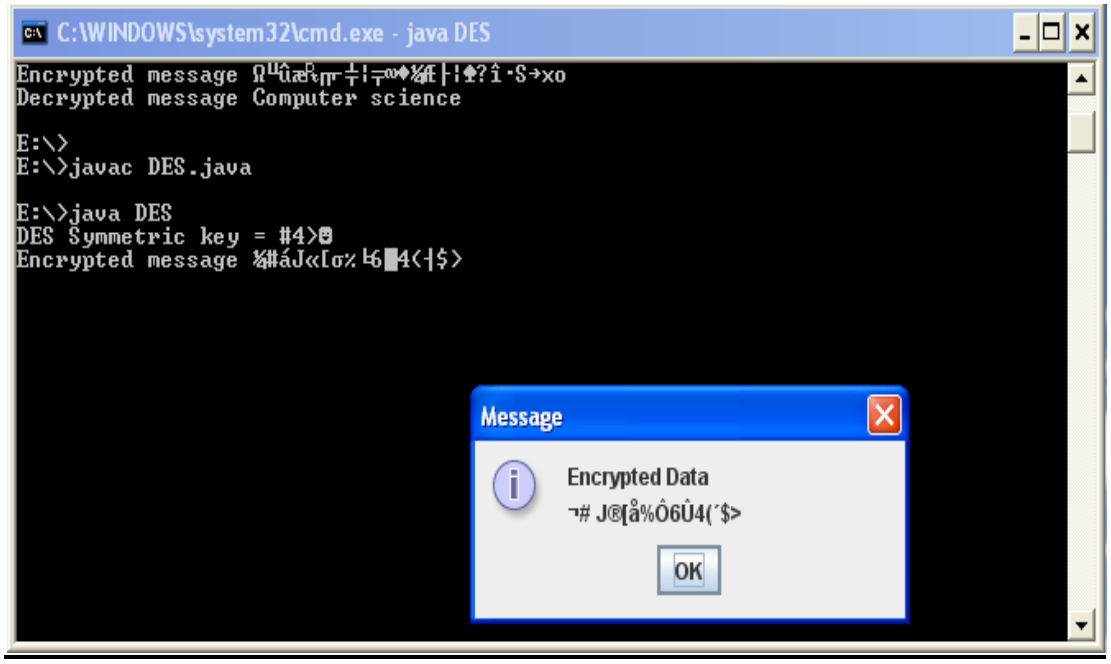
DES.java

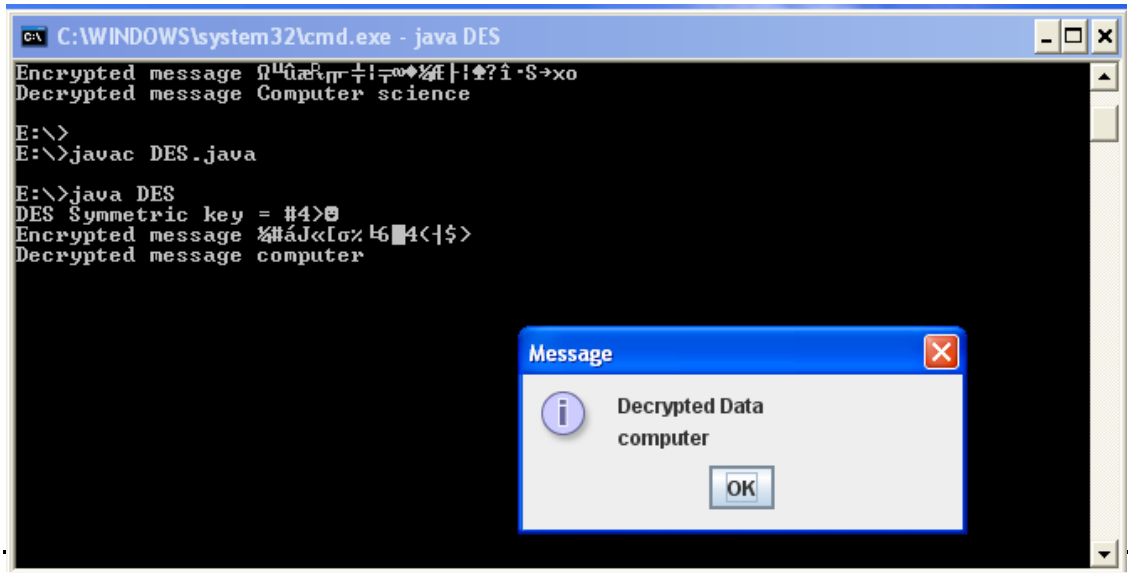
```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
class DES {
     byte[] skey = new byte[1000];
     String skeyString;
     static byte[] raw;
     String inputMessage,encryptedData,decryptedMessage;
public DES()
{
try
{
     generateSymmetricKey();
     inputMessage=JOptionPane.showInputDialog(null,"Enter
     message to encrypt");
     byte[] ibyte = inputMessage.getBytes();
     byte[] ebyte=encrypt(raw, ibyte);
     String encryptedData = new String(ebyte);
     System.out.println("Encrypted message "+encryptedData);
     JOptionPane.showMessageDialog(null,"Encrypted Data
     "+"\n"+encryptedData);
     byte[] dbyte= decrypt(raw,ebyte);
     String decryptedMessage = new String(dbyte);
     System.out.println("Decrypted message
     "+decryptedMessage);
     JOptionPane.showMessageDialog(null,"Decrypted Data
     "+"\n"+decryptedMessage);
}
catch(Exception e)
{
     System.out.println(e);
}
}
```

```
void generateSymmetricKey() {
try {
     Random r = new Random();
     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) throws Exception
{
     KeyGenerator kgen = KeyGenerator.getInstance("DES");
     SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
     sr.setSeed(seed);
     kgen.init(56, sr);
     SecretKey skey = kgen.generateKey();
     raw = skey.getEncoded();
     return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws
Exception {
          SecretKeySpec skeySpec = new SecretKeySpec(raw,
          "DES");
          Cipher cipher = Cipher.getInstance("DES");
          cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
          byte[] encrypted = cipher.doFinal(clear);
          return encrypted;
}
private static byte[] decrypt(byte[] raw, byte[] encrypted)
throws Exception
{
          SecretKeySpec skeySpec = new SecretKeySpec(raw,
          "DES");
          Cipher cipher = Cipher.getInstance("DES");
          cipher.init(Cipher.DECRYPT_MODE, skeySpec);
          byte[] decrypted = cipher.doFinal(encrypted);
          return decrypted;
}
public static void main(String args[]) {
     DES des = new DES();
}
}
```

OUTPUT:


C:\WINDOWS\system32\cmd.exe - java DES
Encrypted message
Decrypted message Computer science
E:\>
E:\>javac DES.java
E:\>java DES
DES Symmetric key = #4>
Input
Enter message to encrypt
computer
OK
Cancel


C:\WINDOWS\system32\cmd.exe - java DES
Encrypted message
Decrypted message Computer science
E:\>
E:\>javac DES.java
E:\>java DES
DES Symmetric key = #4>
Encrypted message
Message
Encrypted Data
OK



RESULT:

Thus the data encryption standard algorithm had been implemented successfully using C language.

EX. NO: 2(B) **IMPLEMENTATION OF RSA**

AIM:

To write a C program to implement the RSA encryption algorithm.

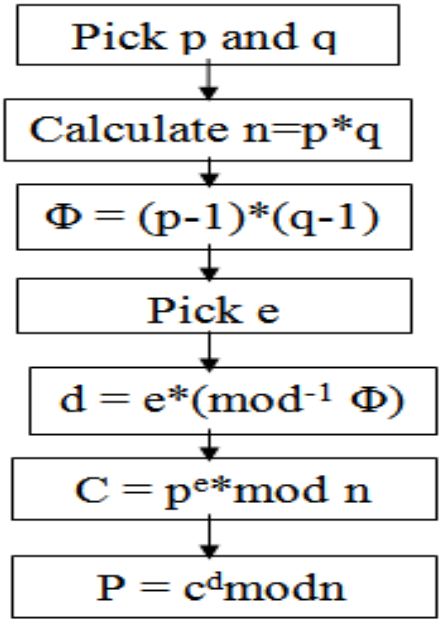
DESCRIPTION:

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. A basic principle behind RSA is the observation that it is practical to find three very large positive integers e, d and n such that with modular exponentiation for all integer m:

$$(m^e)^d = m \pmod n$$

The public key is represented by the integers n and e; and, the private key, by the integer d. m represents the message. RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. The intention is that messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

EXAMPLE:



ALGORITHM:

STEP-1: Select two co-prime numbers as p and q.

STEP-2: Compute n as the product of p and q.

STEP-3: Compute (p-1)*(q-1) and store it in z.

STEP-4: Select a random prime number e that is less than that of z.

STEP-5: Compute the private key, d as e * mod^{-1}(z).

STEP-6: The cipher text is computed as $message^{e}$ * mod n.

STEP-7: Decryption is done as $cipher^{d}$mod n.

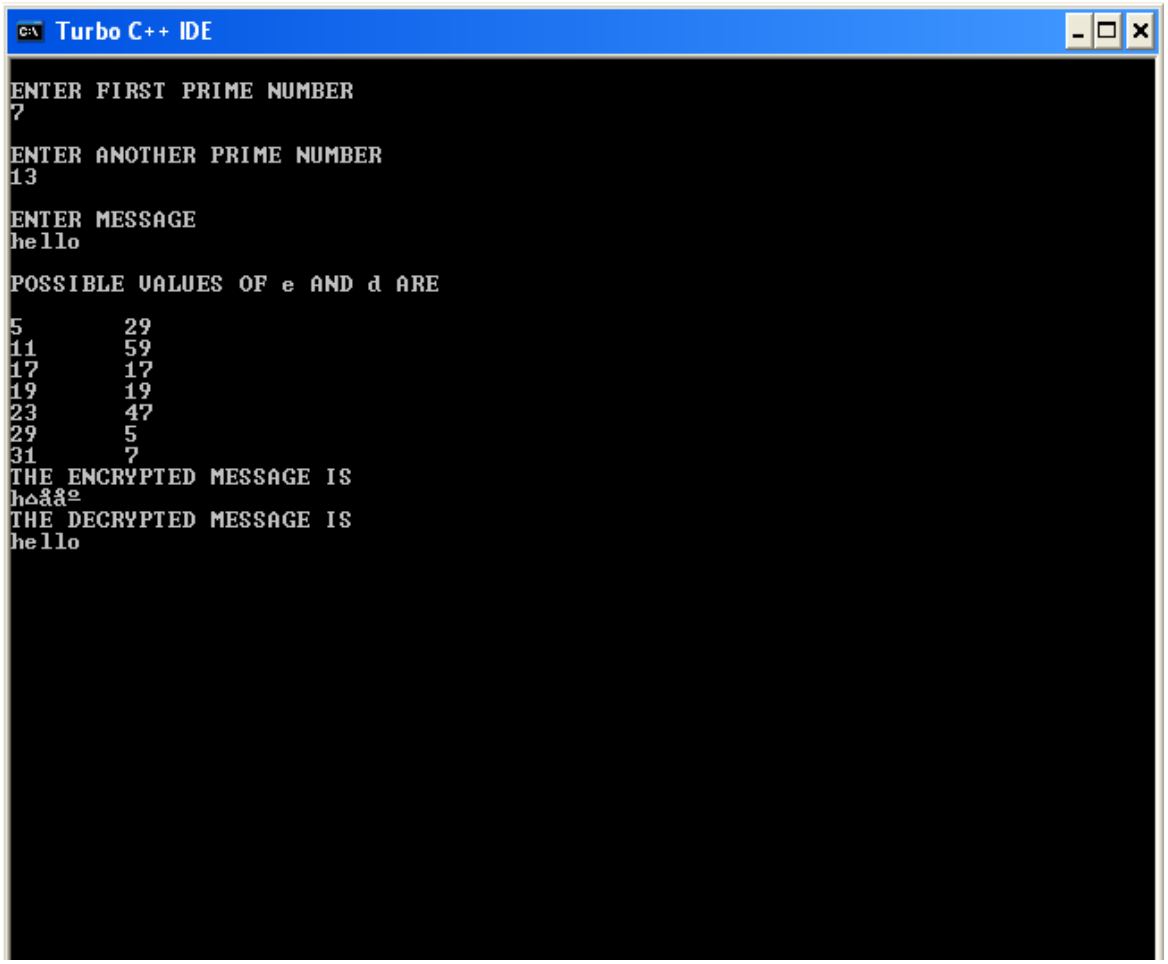
PROGRAM: (RSA)

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
     long int
     p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;
     char msg[100];
     int prime(long int);
     void ce();
     long int cd(long int);
     void encrypt();
     void decrypt();
void main()
{
     clrscr();
     printf("\nENTER FIRST PRIME NUMBER\n");
     scanf("%d",&p);
     flag=prime(p);
     if(flag==0)
      {
          printf("\nWRONG INPUT\n");
          getch();
      }
      printf("\nENTER ANOTHER PRIME NUMBER\n");
      scanf("%d",&q);
      flag=prime(q);
      if(flag==0||p==q)
      {
       printf("\nWRONG INPUT\n");
       getch();
      }
      printf("\nENTER MESSAGE\n");
      fflush(stdin);
      scanf("%s",msg);
      for(i=0;msg[i]!=NULL;i++)
      m[i]=msg[i];
      n=p*q;
```

```
        t=(p-1)*(q-1);
        ce();
        printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
        for(i=0;i<j-1;i++)
        printf("\n%ld\t%ld",e[i],d[i]);
        encrypt();
        decrypt();
        getch();
}
int prime(long int pr)
{
int i;
j=sqrt(pr);
for(i=2;i<=j;i++)
{
if(pr%i==0)
return 0;
}
return 1;
}
void ce()
{
int k;
k=0;
for(i=2;i<t;i++)
{
 if(t%i==0)
 continue;
 flag=prime(i);
 if(flag==1&&i!=p&&i!=q)
 {
 e[k]=i;
 flag=cd(e[k]);
 if(flag>0)
 {
 d[k]=flag;
 k++;
 }
 if(k==99)
 break;
 } } }
 long int cd(long int x)
 {
 long int k=1;
 while(1)
 {
 k=k+t;
 if(k%x==0)
 return(k/x);
 } }
 void encrypt() {
 long int pt,ct,key=e[0],k,len;
 i=0;
 len=strlen(msg);
```

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

OUTPUT:



RESULT:

Thus the C program to implement RSA encryption technique had been implemented successfully

EX. NO: 2(C)

IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

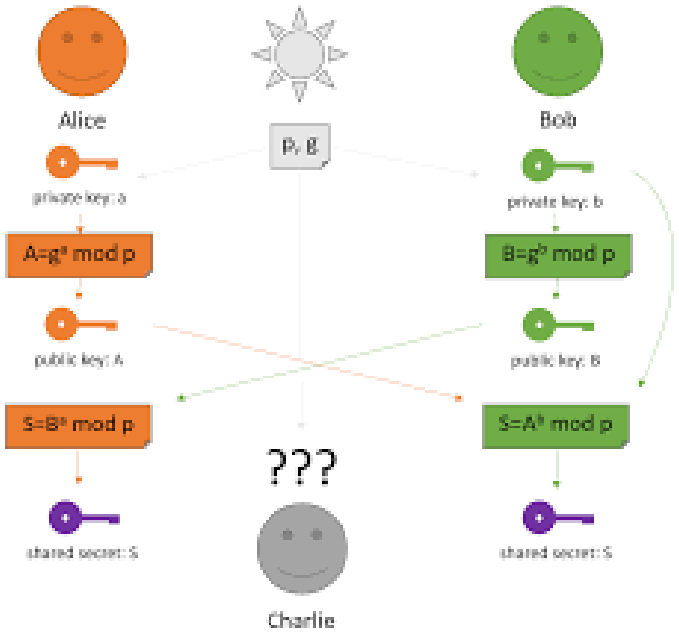
AIM:

To implement the Diffie-Hellman Key Exchange algorithm using C language.

DESCRIPTION:

Diffie–Hellman Key Exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. It is primarily used as a method of exchanging cryptography keys for use in symmetric encryption algorithms like AES. The algorithm in itself is very simple. The process begins by having the two parties, Alice and Bob. Let's assume that Alice wants to establish a shared secret with Bob.

EXAMPLE:



ALGORITHM:

STEP-1: Both Alice and Bob shares the same public keys g and p.

STEP-2: Alice selects a random public key a.

STEP-3: Alice computes his secret key A as g^a mod p.

STEP-4: Then Alice sends A to Bob.

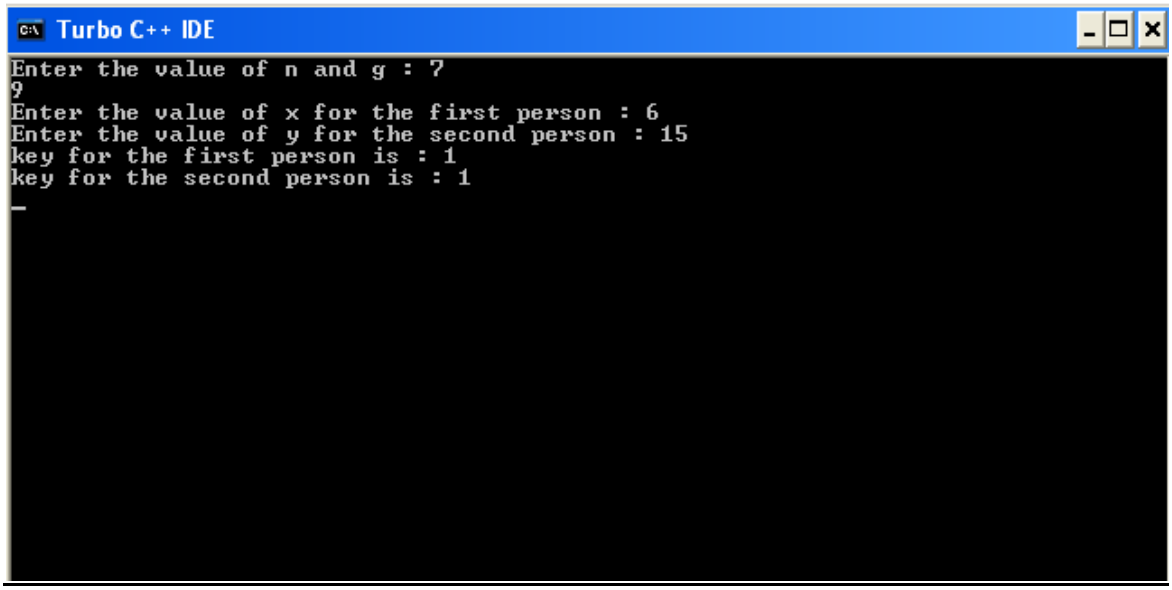
STEP-5: Similarly Bob also selects a public key b and computes his secret key as B and sends the same back to Alice.

STEP-6: Now both of them compute their common secret key as the other one's secret key power of a mod p.

PROGRAM: (Diffie Hellman Key Exchange)

```
#include<stdio.h>
#include<conio.h>
long long int power(int a, int b, int mod)
{
     long long int t;
     if(b==1)
     return a;
     t=power(a,b/2,mod);
     if(b%2==0)
     return (t*t)%mod;
     else
     return (((t*t)%mod)*a)%mod;
}
long int calculateKey(int a, int x, int n)
{
     return power(a,x,n);
}
void main()
{
     int n,g,x,a,y,b;
     clrscr();
     printf("Enter the value of n and g : ");
     scanf("%d%d",&n,&g);
     printf("Enter the value of x for the first person : ");
     scanf("%d",&x);
     a=power(g,x,n);
     printf("Enter the value of y for the second person : ");
     scanf("%d",&y);
     b=power(g,y,n);
     printf("key for the first person is :
     %lld\n",power(b,x,n));
     printf("key for the second person is :
     %lld\n",power(a,y,n));
     getch();
}
```

OUTPUT:



RESULT:

Thus the Diffie-Hellman key exchange algorithm had been successfully implemented using C.

EX. NO: 2(D)

IMPLEMENTATION OF MD5

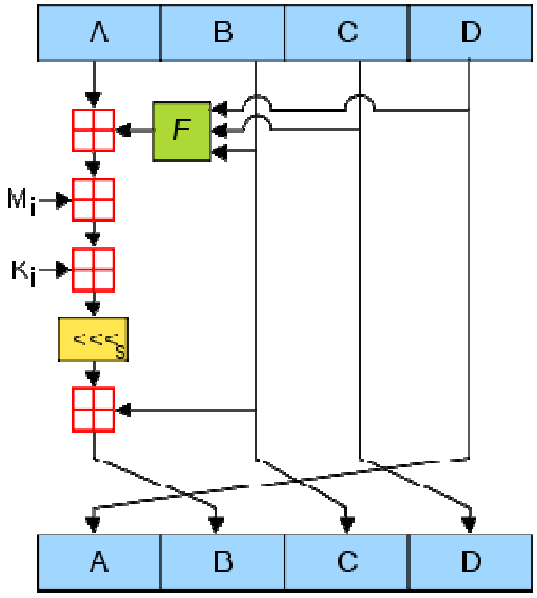
AIM:

To write a C program to implement the MD5 hashing technique.

DESCRIPTION:

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks. The message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with 64 bits representing the length of the original message, modulo 2^{64}.The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted A, B, C, and D. These are initialized to certain fixed constants. The main algorithm then uses each 512-bit message block in turn to modify the state.

EXAMPLE:



ALGORITHM:

STEP-1: Read the 128-bit plain text.

STEP-2: Divide into four blocks of 32-bits named as A, B, C and D.

STEP-3: Compute the functions f, g, h and i with operations such as, rotations, permutations, etc,.

STEP-4: The output of these functions are combined together as F and performed circular shifting and then given to key round.

STEP-5: Finally, right shift of 's' times are performed and the results are combined together to produce the final output.

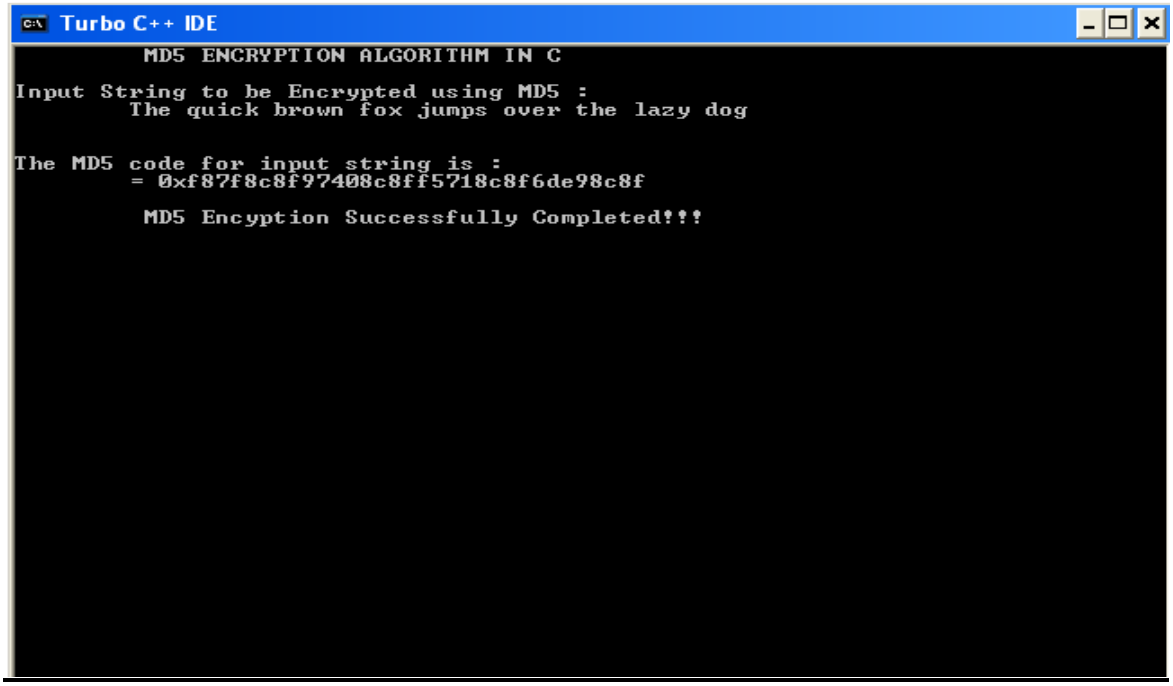
PROGRAM:(MD5)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include<conio.h>
typedef union uwb
{
    unsigned w;
    unsigned char b[4];
} MD5union;
     typedef unsigned DigestArray[4];
     unsigned func0( unsigned abcd[] ){
     return ( abcd[1] & abcd[2]) | (~abcd[1] & abcd[3]);}
     unsigned func1( unsigned abcd[] ){
     return ( abcd[3] & abcd[1]) | (~abcd[3] & abcd[2]);}
     unsigned func2( unsigned abcd[] ){
     return  abcd[1] ^ abcd[2] ^ abcd[3];}
     unsigned func3( unsigned abcd[] ){
     return abcd[2] ^ (abcd[1] |~ abcd[3]);}
     typedef unsigned (*DgstFctn)(unsigned a[]);
unsigned *calctable( unsigned *k)
{
     double s, pwr;
     int i;
     pwr = pow( 2, 32);
for (i=0; i<64; i++)
{
     s = fabs(sin(1+i));
     k[i] = (unsigned)( s * pwr );
}
     return k;
}
unsigned rol( unsigned r, short N )
{
    unsigned  mask1 = (1<<N) -1;
    return ((r>>(32-N)) & mask1) | ((r<<N) & ~mask1);
}
```

```
unsigned *md5( const char *msg, int mlen)
{
    static DigestArray h0 = { 0x67452301, 0xEFCDAB89,
    0x98BADCFE, 0x10325476 };
    static DgstFctn ff[] = { &func0, &func1, &func2, &func3};
    static short M[] = { 1, 5, 3, 7 };
    static short O[] = { 0, 1, 5, 0 };
    static short rot0[] = { 7,12,17,22};
    static short rot1[] = { 5, 9,14,20};
    static short rot2[] = { 4,11,16,23};
    static short rot3[] = { 6,10,15,21};
    static short *rots[] = {rot0, rot1, rot2, rot3 };
    static unsigned kspace[64];
    static unsigned *k;
    static DigestArray h;
    DigestArray abcd;
    DgstFctn fctn;
    short m, o, g;
    unsigned f;
    short *rotn;
    union
    {
         unsigned w[16];
         char     b[64];
    }mm;
         int os = 0;
         int grp, grps, q, p;
         unsigned char *msg2;
         if (k==NULL) k= calctable(kspace);
    for (q=0; q<4; q++) h[q] = h0[q];   // initialize
    {
          grps  = 1 + (mlen+8)/64;
          msg2 = malloc( 64*grps);
          memcpy( msg2, msg, mlen);
          msg2[mlen] = (unsigned char)0x80;
          q = mlen + 1;
     while (q < 64*grps){ msg2[q] = 0; q++ ; }
     {
          MD5union u;
          u.w = 8*mlen;
          q -= 8;
          memcpy(msg2+q, &u.w, 4 );
     }
     }
     for (grp=0; grp<grps; grp++)
     {
          memcpy( mm.b, msg2+os, 64);
```

```
        for(q=0;q<4;q++) abcd[q] = h[q];
        for (p = 0; p<4; p++)
        {
             fctn = ff[p];
             rotn = rots[p];
             m = M[p]; o= O[p];
        for (q=0; q<16; q++)
        {
            g = (m*q + o) % 16;
           f = abcd[1] + rol( abcd[0]+ fctn(abcd)+k[q+16*p]
           + mm.w[g], rotn[q%4]);
           abcd[0] = abcd[3];
           abcd[3] = abcd[2];
           abcd[2] = abcd[1];
          abcd[1] = f;
        }}
 for (p=0; p<4; p++)
 h[p] += abcd[p];
 os += 64;
 }
return h;}
void main()
{
     int j,k;
     const char *msg = "The quick brown fox jumps over
     the lazy dog";
     unsigned *d = md5(msg, strlen(msg));
     MD5union u;
     clrscr();
     printf("\t MD5 ENCRYPTION ALGORITHM IN C \n\n");
     printf("Input String to be Encrypted using MD5 :
     \n\t%s",msg);
     printf("\n\nThe MD5 code for input string is: \n");
     printf("\t= 0x");
     for (j=0;j<4; j++){
     u.w = d[j];
     for (k=0;k<4;k++) printf("%02x",u.b[k]);
}
printf("\n");
printf("\n\t MD5 Encyption Successfully
Completed!!!\n\n");
getch();
system("pause");
getch();}
```

OUTPUT:



RESULT:

Thus the implementation of MD5 hashing algorithm had been implemented successfully using C.

EX. NO: 2(E) **IMPLEMENTATION OF SHA-I**

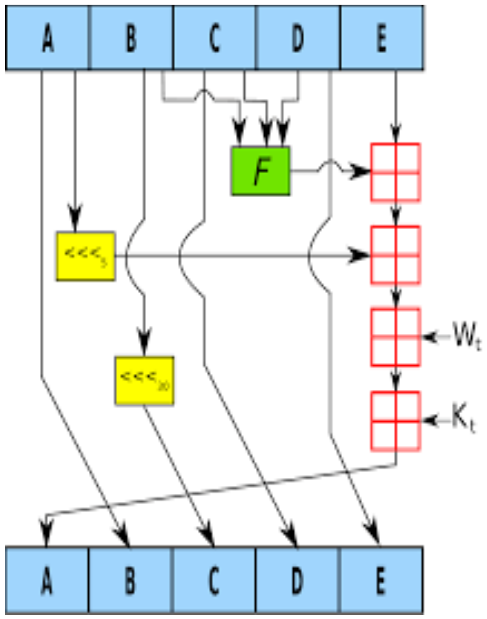
AIM:

To implement the SHA-I hashing technique using C program.

DESCRIPTION:

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function. SHA-1 produces a 160-bit hash value known as a message digest. The way this algorithm works is that for a message of size < 264 bits it computes a 160-bit condensed output called a message digest. The SHA-1 algorithm is designed so that it is practically infeasible to find two input messages that hash to the same output message. A hash function such as SHA-1 is used to calculate an alphanumeric string that serves as the cryptographic representation of a file or a piece of data. This is called a digest and can serve as a digital signature. It is supposed to be unique and non-reversible.

EXAMPLE:



ALGORITHM:

STEP-1: Read the 256-bit key values.

STEP-2: Divide into five equal-sized blocks named A, B, C, D and E.

STEP-3: The blocks B, C and D are passed to the function F.

STEP-4: The resultant value is permuted with block E.

STEP-5: The block A is shifted right by ‘s’ times and permuted with the result of step-4.

STEP-6: Then it is permuted with a weight value and then with some other key pair and taken as the first block.

STEP-7: Block A is taken as the second block and the block B is shifted by 's' times and taken as the third block.

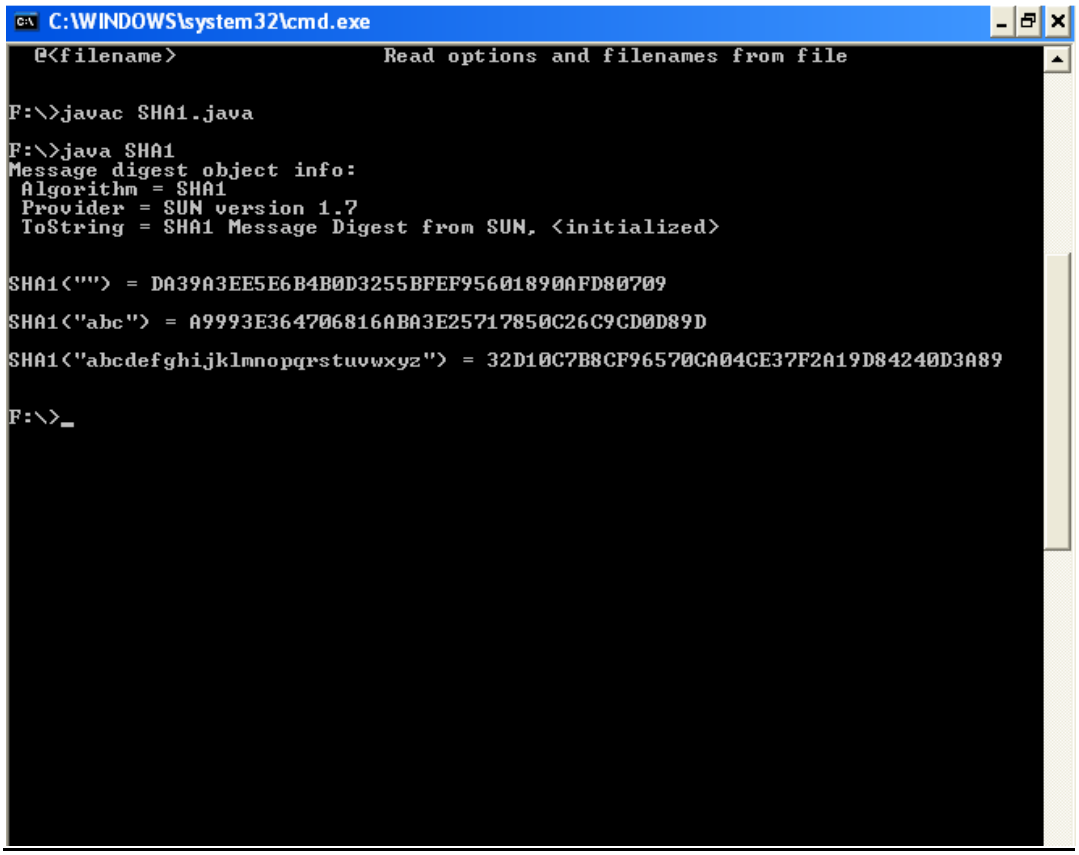
STEP-8: The blocks C and D are taken as the block D and E for the final output.

PROGRAM: (Secure Hash Algorithm)

```
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: ");
     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);
}
}
public static String bytesToHex(byte[] b)
{
     char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6',
     '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
     StringBuffer buf = new StringBuffer();
     for (int j=0; j<b.length; j++) {
```

```
        buf.append(hexDigit[(b[j] >> 4) & 0x0f]);
        buf.append(hexDigit[b[j] & 0x0f]); }
        return buf.toString(); }
}
```

OUTPUT:



RESULT:

Thus the SHA-1 hashing technique had been implemented successfully.

EX. NO: 3

IMPLEMENTATION OF DIGITAL SIGNATURE STANDARD

AIM:

To write a C program to implement the signature scheme named digital signature standard (Euclidean Algorithm).

ALGORITHM:

STEP-1: Alice and Bob are investigating a forgery case of x and y.

STEP-2: X had document signed by him but he says he did not sign that document digitally.

STEP-3: Alice reads the two prime numbers p and a.

STEP-4: He chooses a random co-primes alpha and beta and the x's original signature x.

STEP-5: With these values, he applies it to the elliptic curve cryptographic equation to obtain y.

STEP-6: Comparing this 'y' with actual y's document, Alice concludes that y is a forgery.

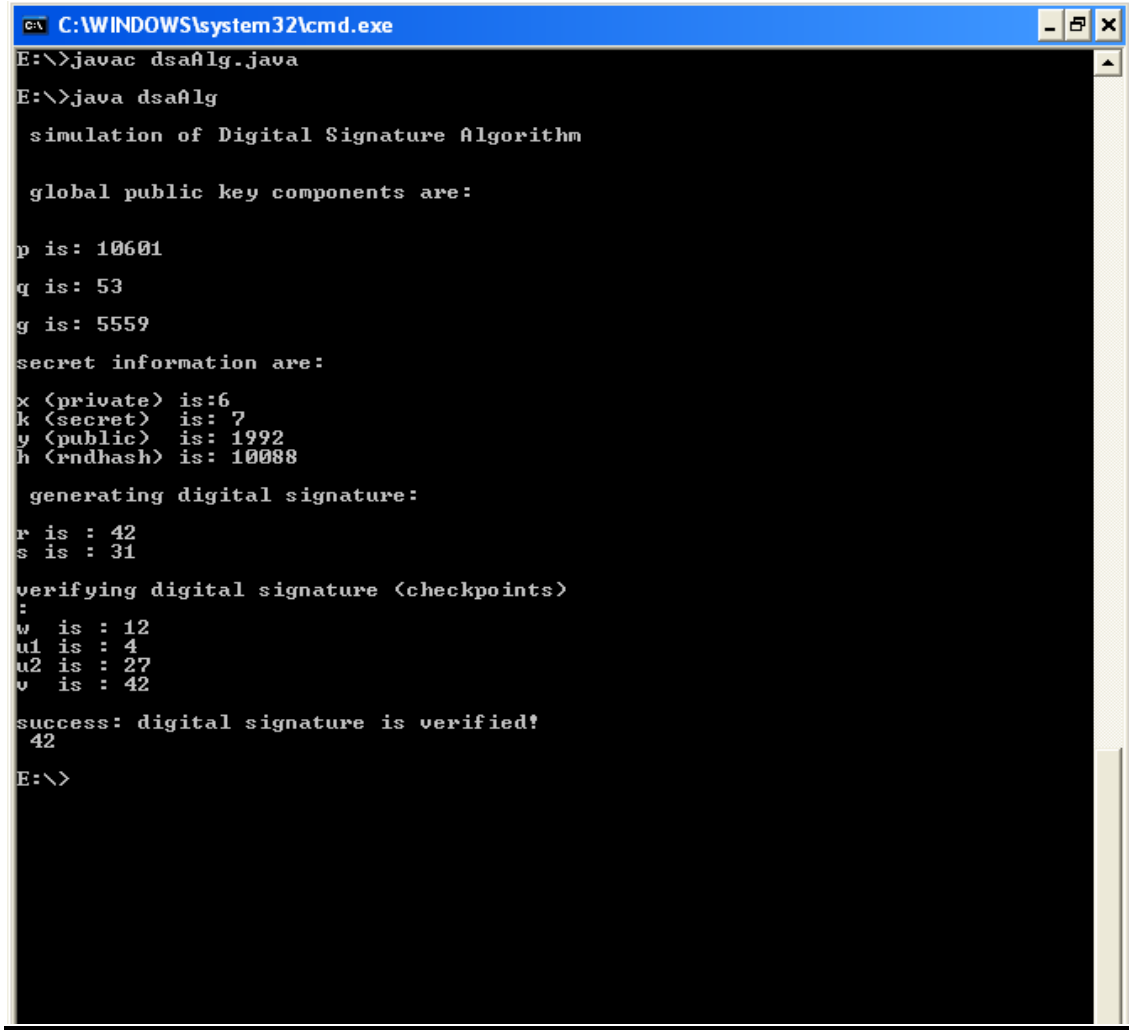
PROGRAM: (Digital Signature Standard)

```
import java.util.*;
import java.math.BigInteger;
class dsaAlg {
     final static BigInteger one  = new BigInteger("1");
     final static BigInteger zero = new BigInteger("0");
public static BigInteger getNextPrime(String ans)
{
     BigInteger test = new BigInteger(ans);
while (!test.isProbablePrime(99))
e:
{
     test = test.add(one);
}
     return test;
}
public static BigInteger findQ(BigInteger n)
{
     BigInteger start = new BigInteger("2");
while (!n.isProbablePrime(99))
{
     while (!((n.mod(start)).equals(zero)))
     {
          start = start.add(one);
```

```
        }
        n = n.divide(start);
    }
        return n;
    }
    public static BigInteger getGen(BigInteger p, BigInteger q,
    Random r)
    {
        BigInteger h = new BigInteger(p.bitLength(), r);
        h = h.mod(p);
        return h.modPow((p.subtract(one)).divide(q), p);
    }
    public static void main (String[] args) throws
    java.lang.Exception
    {
        Random randObj = new Random();
        BigInteger p = getNextPrime("10600");  /* approximate
        prime */
        BigInteger q = findQ(p.subtract(one));
        BigInteger g = getGen(p,q,randObj);
        System.out.println(" \n simulation of Digital Signature
        Algorithm \n");
        System.out.println(" \n global public key components
        are:\n");
        System.out.println("\np is: " + p);
        System.out.println("\nq is: " + q);
        System.out.println("\ng is: " + g);
        BigInteger x = new BigInteger(q.bitLength(), randObj);
        x = x.mod(q);
        BigInteger y = g.modPow(x,p);
        BigInteger k = new BigInteger(q.bitLength(), randObj);
        k = k.mod(q);
        BigInteger r = (g.modPow(k,p)).mod(q);
        BigInteger hashVal = new BigInteger(p.bitLength(),
        randObj);
        BigInteger kInv = k.modInverse(q);
        BigInteger s = kInv.multiply(hashVal.add(x.multiply(r)));
        s = s.mod(q);
        System.out.println("\nsecret information are:\n");
        System.out.println("x (private) is:" + x);
        System.out.println("k (secret)  is: " + k);
        System.out.println("y (public)  is: " + y);
        System.out.println("h (rndhash) is: " + hashVal);
        System.out.println("\n generating digital signature:\n");
        System.out.println("r is : " + r);
        System.out.println("s is : " + s);
        BigInteger w = s.modInverse(q);
        BigInteger u1 = (hashVal.multiply(w)).mod(q);
        BigInteger u2 = (r.multiply(w)).mod(q);
        BigInteger v = (g.modPow(u1,p)).multiply(y.modPow(u2,p));
        v = (v.mod(p)).mod(q);
        System.out.println("\nverifying digital signature
        (checkpoints)\n:");
        System.out.println("w  is : " + w);
```

```
        System.out.println("u1 is : " + u1);
        System.out.println("u2 is : " + u2);
        System.out.println("v  is : " + v);
if (v.equals(r))
{
        System.out.println("\nsuccess: digital signature is
        verified!\n " + r);
}
else
{
        System.out.println("\n error: incorrect digital
        signature\n ");
}
}
}
```

OUTPUT:



RESULT:

Thus the simple Code Optimization techniques had been implemented successfully.

EX. NO: 04

SECURE DATA STORAGE, SECURE DATA TRANSMISSION AND FOR CREATING DIGITAL SIGNATURES (GNUPG)

AIM:

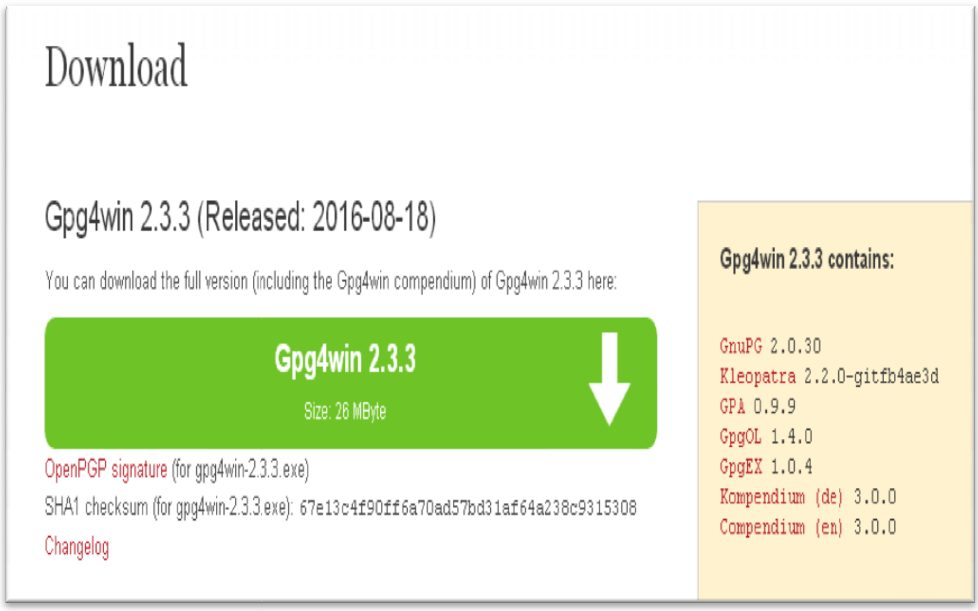
Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures (GnuPG).

INTRODUCTION:

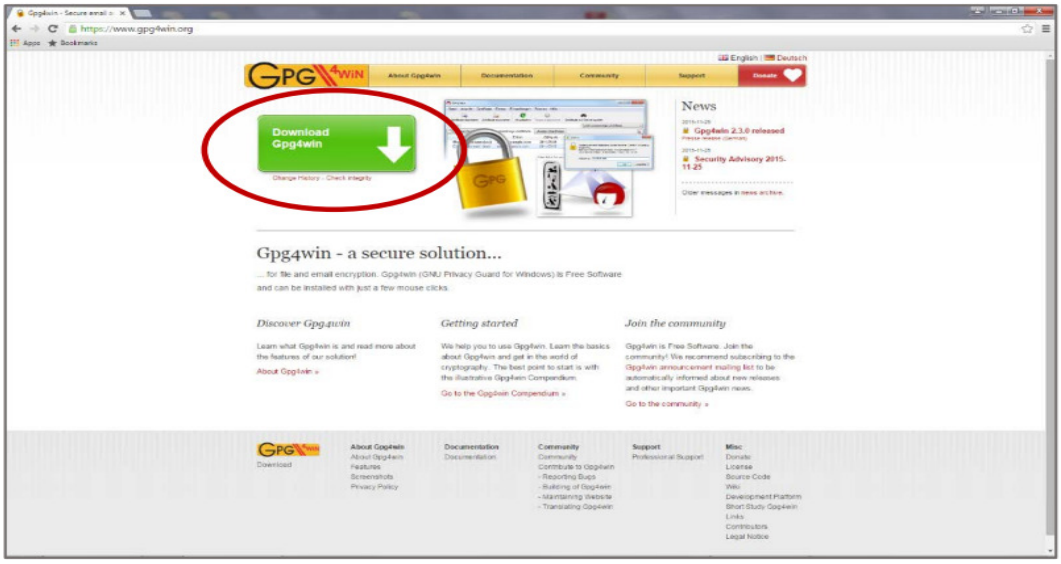
- Here's the final guide in my PGP basics series, this time focusing on Windows
- The OS in question will be Windows 7, but it should work for Win8 and Win8.1 as well
- Obviously it's not recommended to be using Windows to access the DNM, but I won't go into the reasons here.
- The tool well be using is GPG4Win

INSTALLING THE SOFTWARE:

1. Visit www.gpg4win.org. Click on the "Gpg4win 2.3.0" button

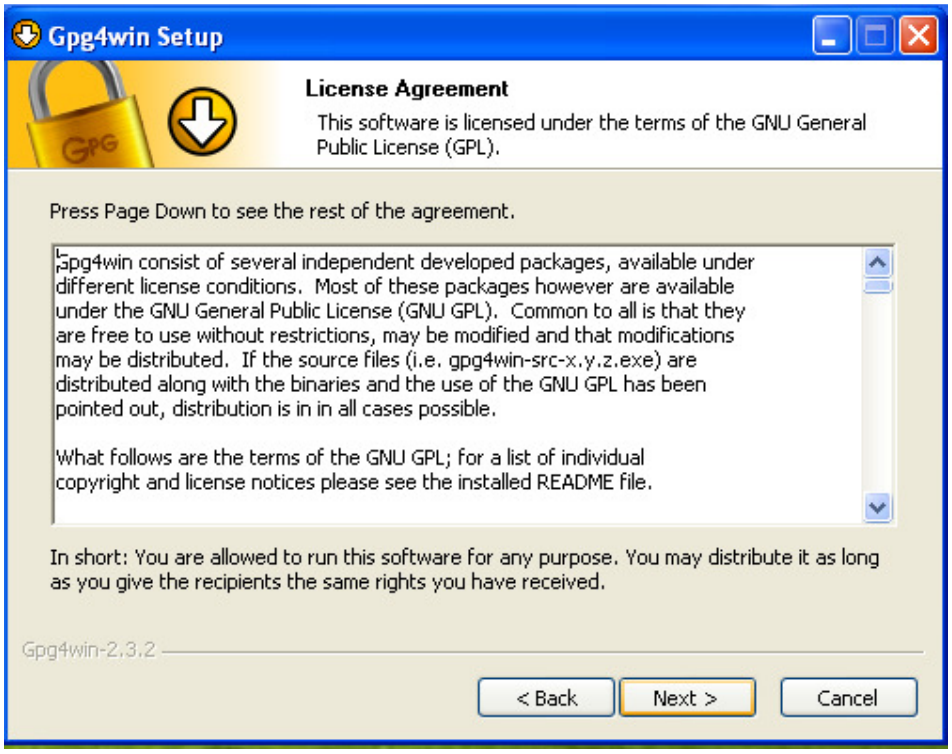


2. On the following screen, click the “Download Gpg4win” button.

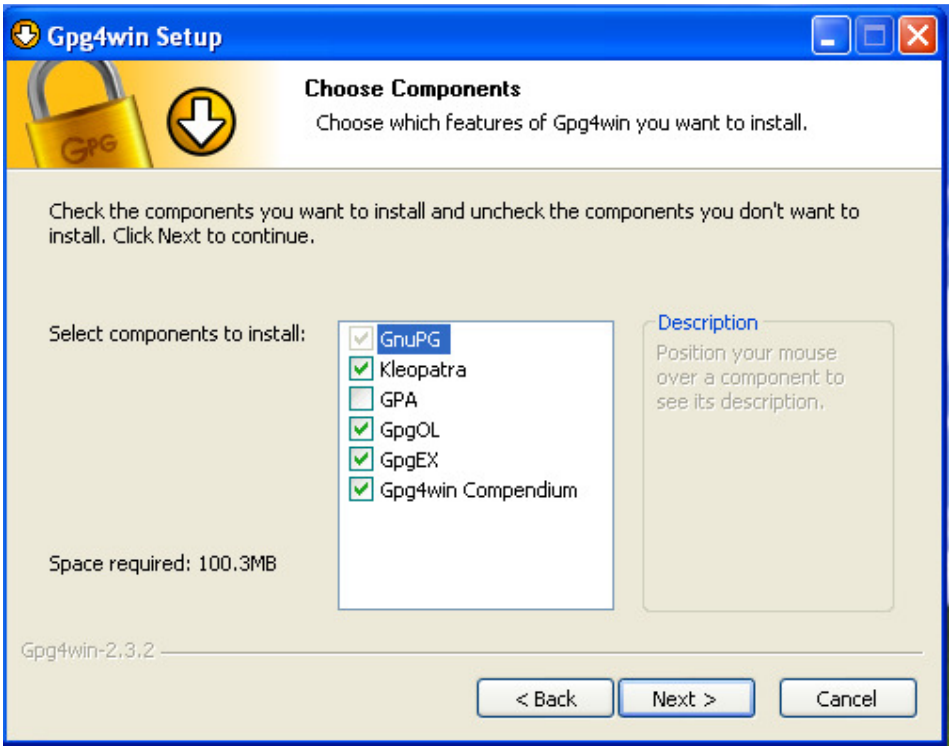


3. When the “Welcome” screen is displayed, click the “Next” button

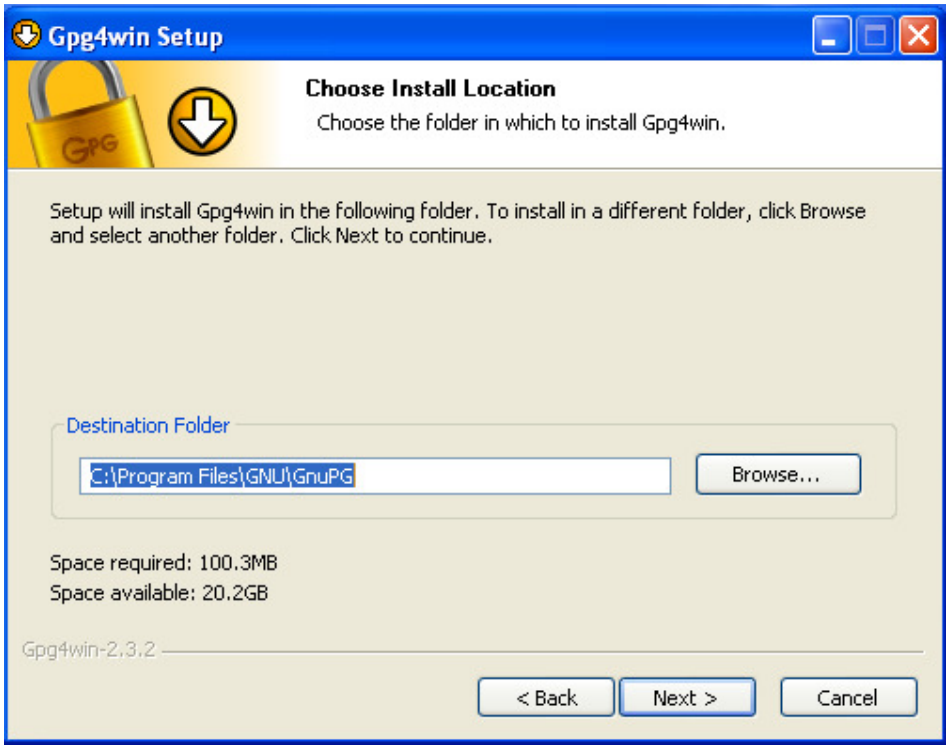
4. When the “License Agreement” page is displayed, click the “Next” button



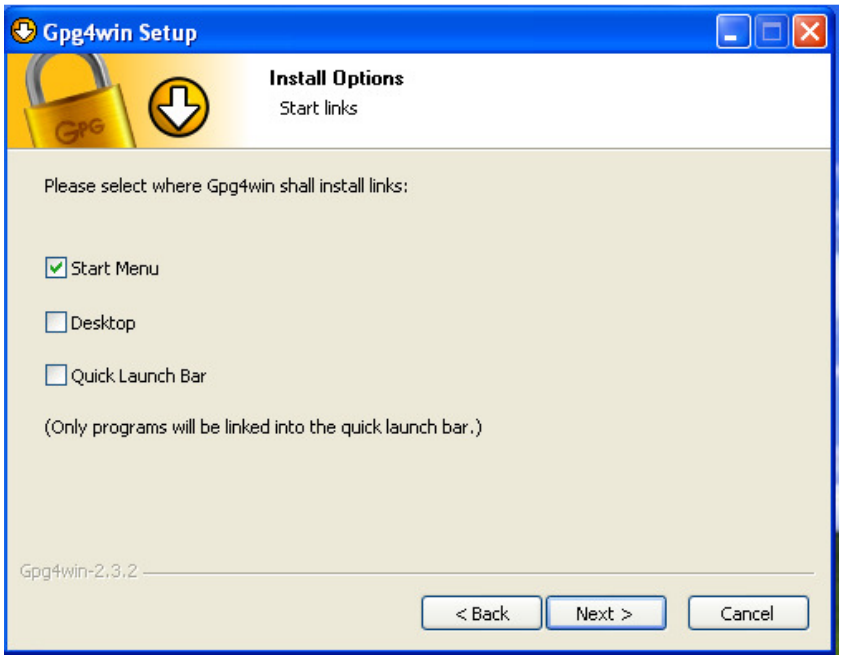
5. Set the check box values as specified below, then click the “Next” button



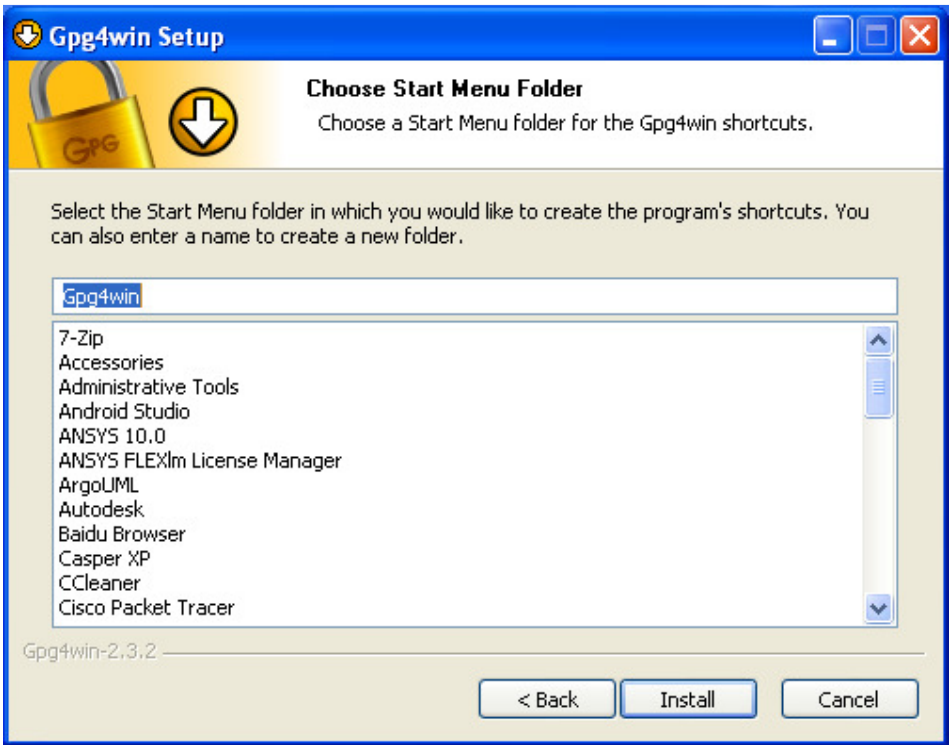
6. Set the location where you want the software to be installed. The default location is fine. Then, click the “Next” button.



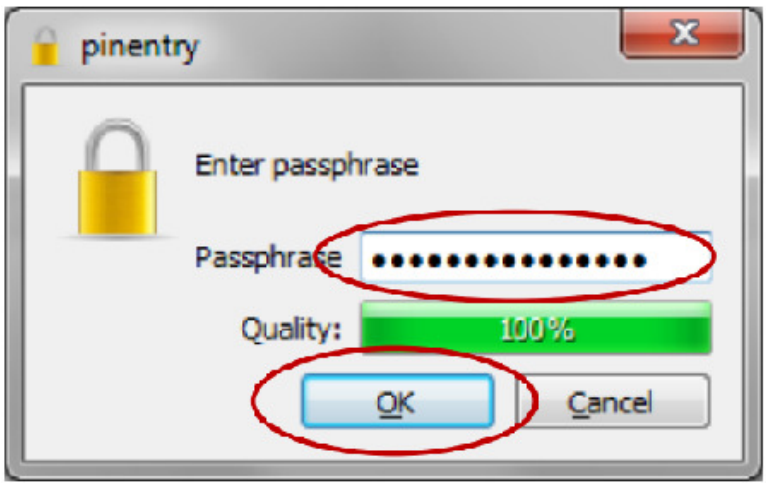
7. Specify where you want shortcuts to the software placed, then click the “Next” button.



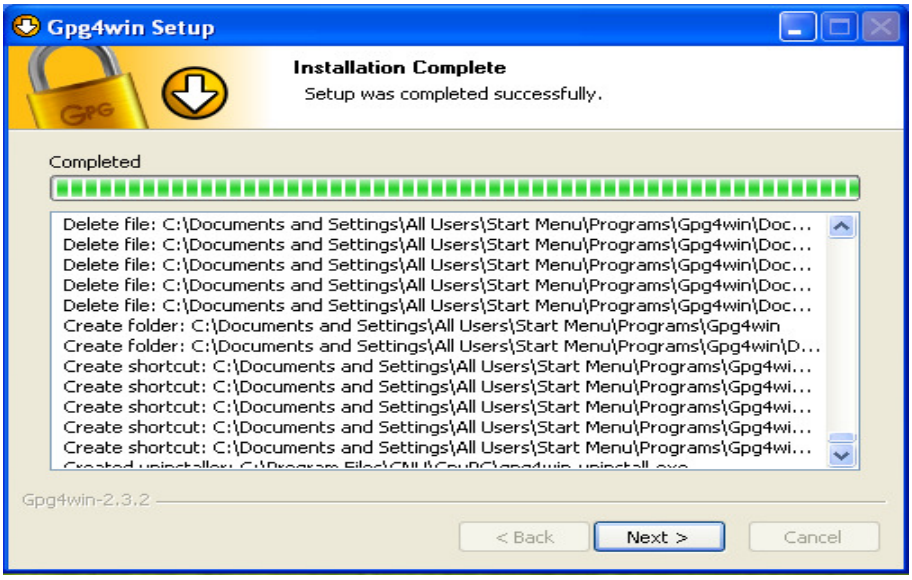
8. If you selected to have a GPG shortcut in your Start Menu, specify the folder in which it will be placed. The default "Gpg4win" is OK. Click the "Install" button to continue



9. A warning will be displayed if you have Outlook or Explorer opened. If this occurs, click the "OK" button.

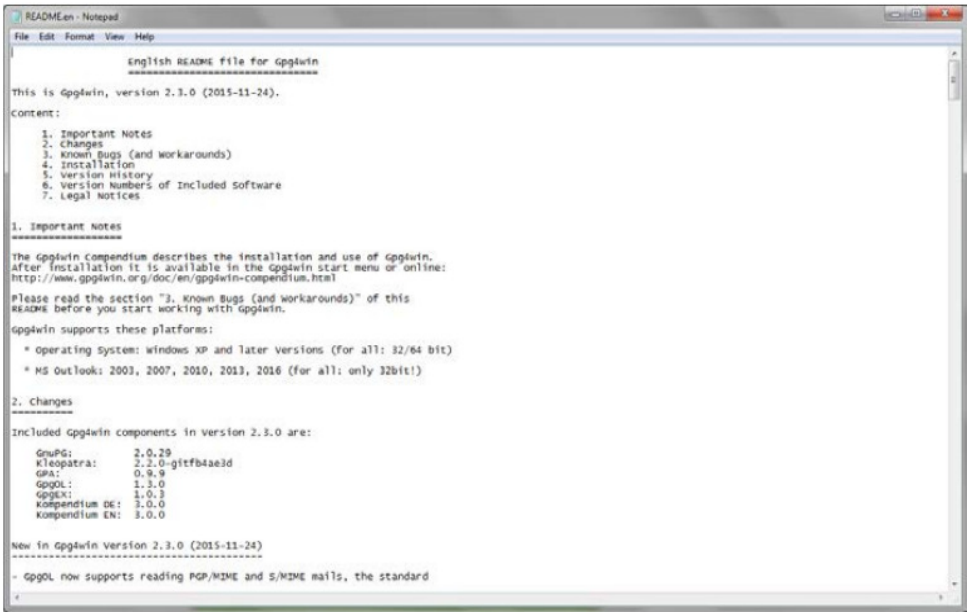


10. The installation process will tell you when it is complete. Click the "Next" button



11. Once the Gpg4win setup wizard is complete, the following screen will be displayed. Click the "Finish" button

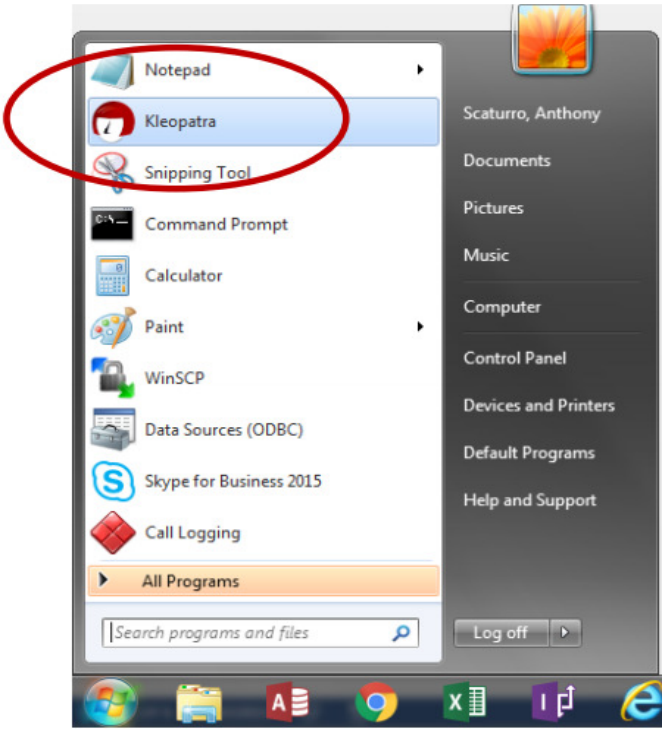
12. If you do not uncheck the "Show the README file" check box, the README file will be displayed. The window can be closed after you've reviewed it.



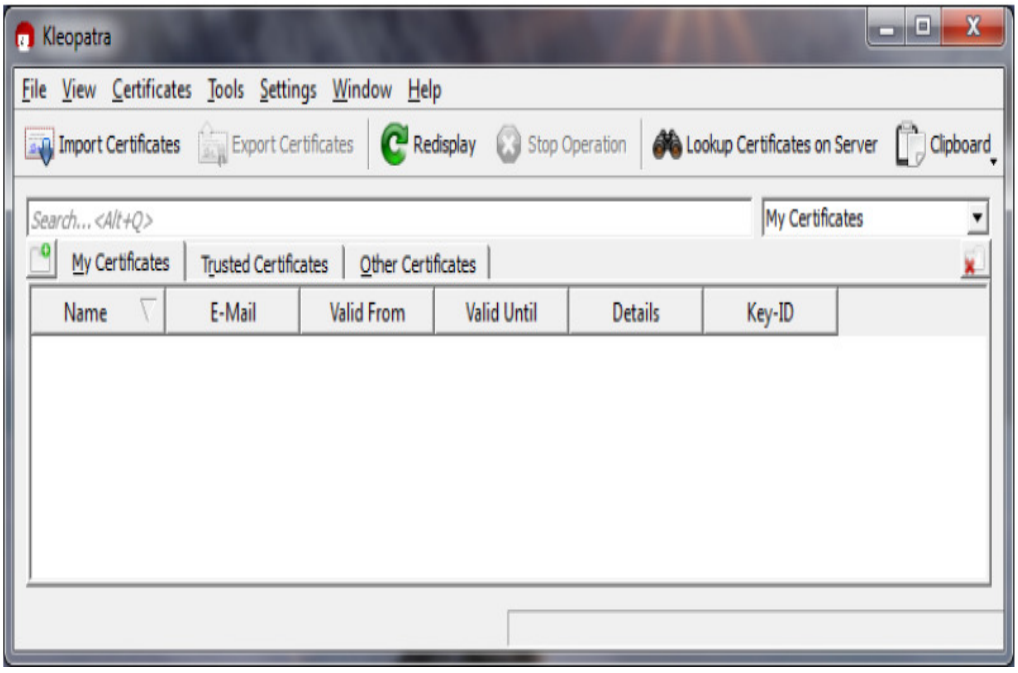
CREATING YOUR PUBLIC AND PRIVATE KEYS

GPG encryption and decryption is based upon the keys of the person who will be receiving the encrypted file or message. Any individual who wants to send the person an encrypted file or message must possess the recipient's public key certificate to encrypt the message. The recipient must have the associated private key, which is different than the public key, to be able to decrypt the file. The public and private key pair for an individual is usually generated by the individual on his or her computer using the installed GPG program, called "Kleopatra" and the following procedure:

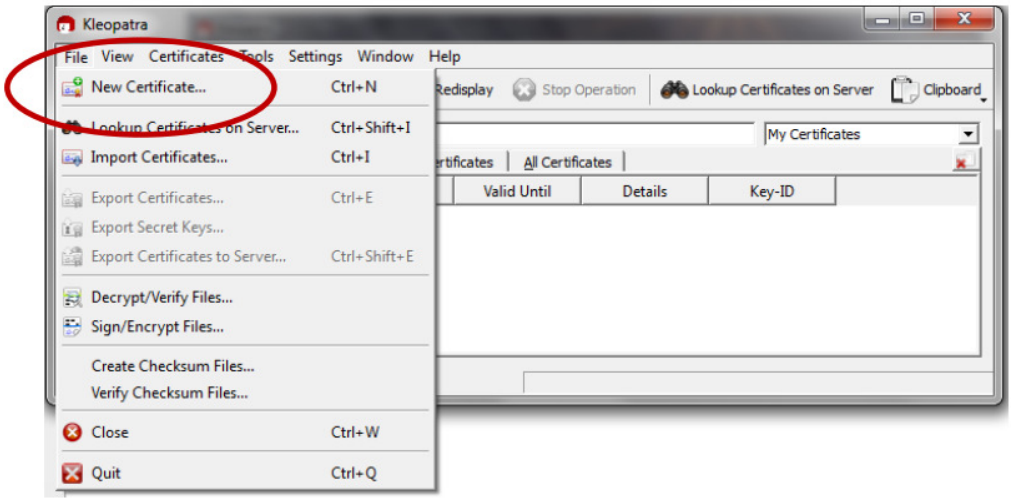
1. From your start bar, select the “Kleopatra” icon to start the Kleopatra certificate management software



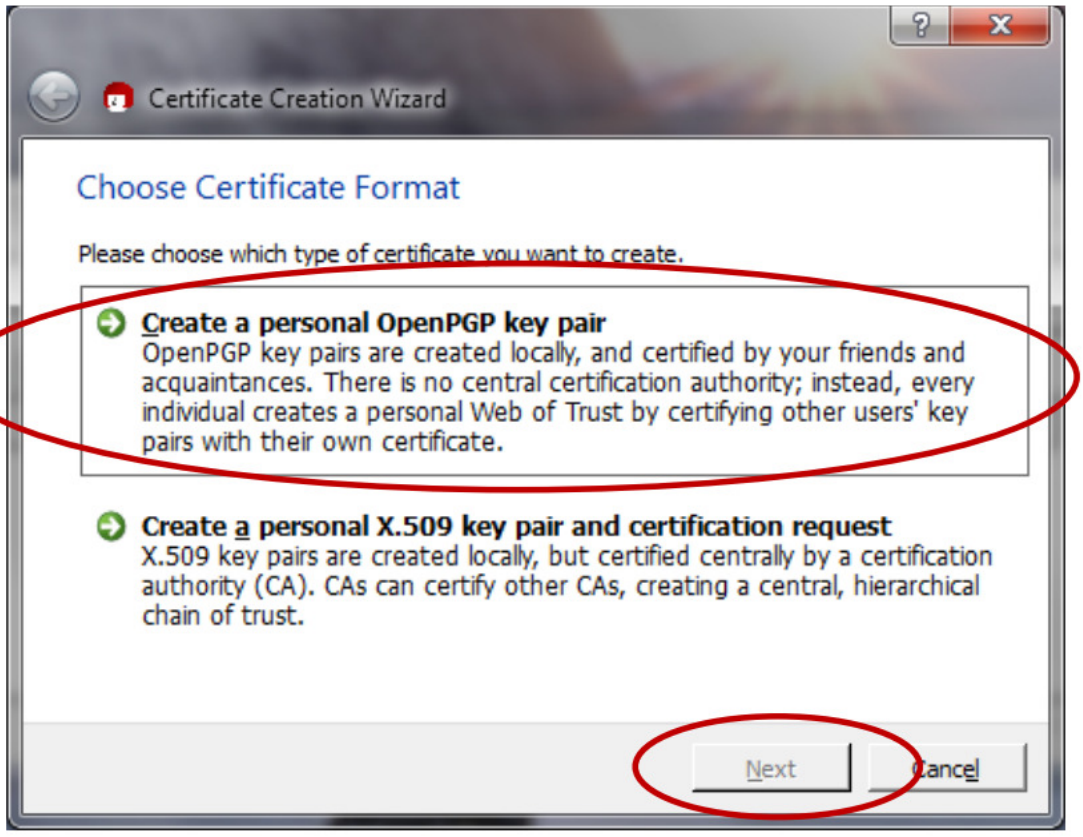
2. The following screen will be displayed



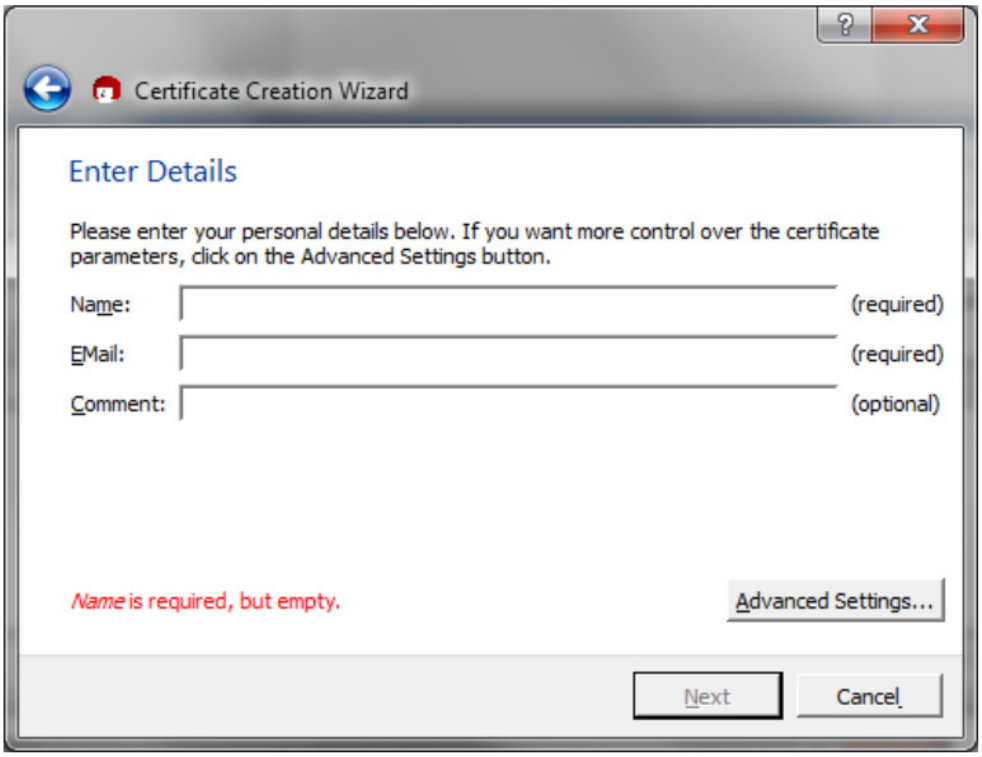
3. From the “File” dropdown, click on the “New Certificate” option



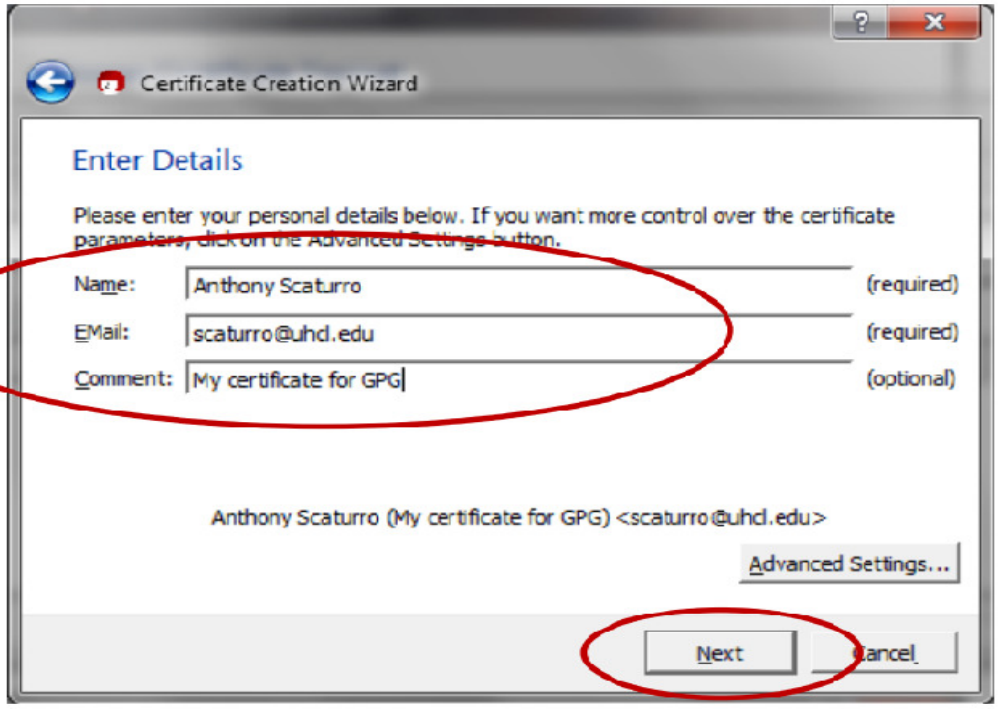
4. The following screen will be displayed. Click on “Create a personal OpenGPG key pair” and the “Next” button



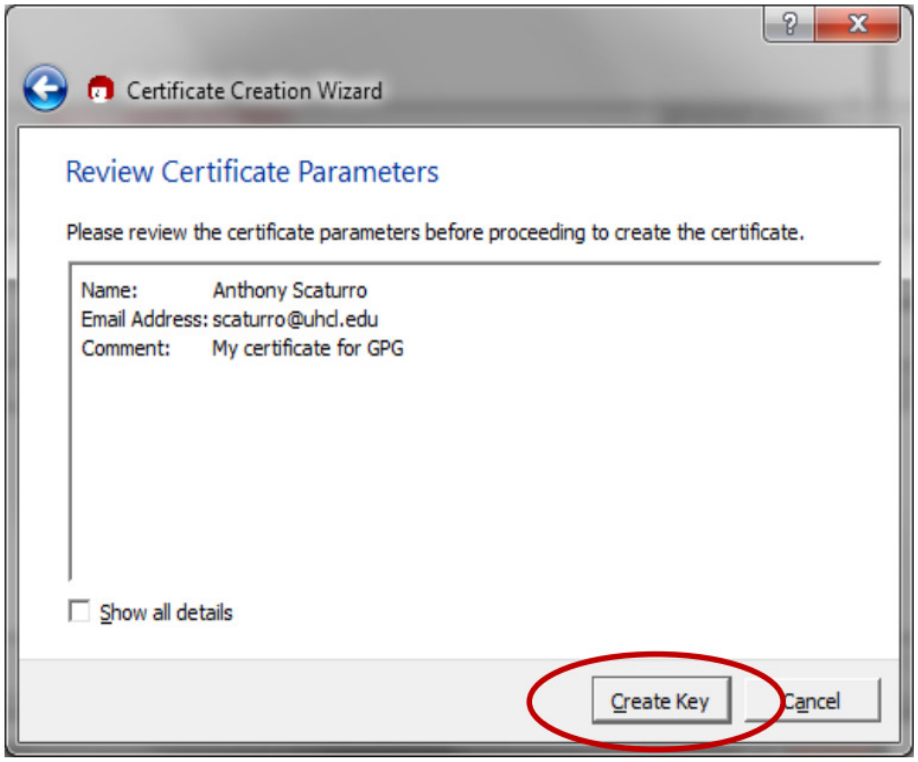
5. The Certificate Creation Wizard will start and display the following:



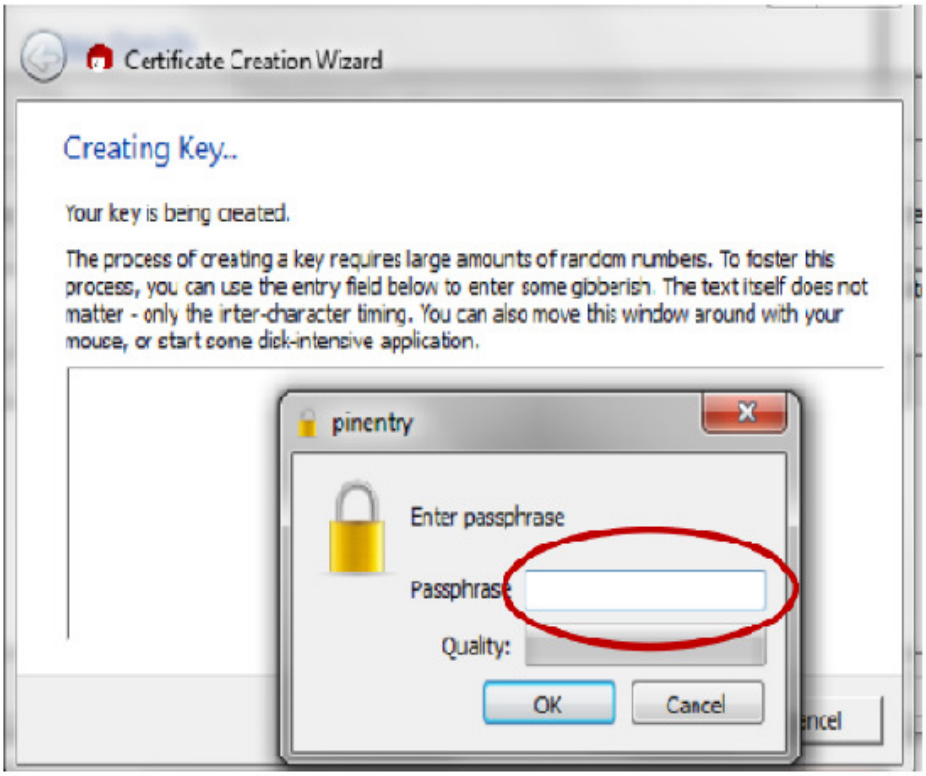
6. Enter your name and e-mail address. You may also enter an optional comment. Then, click the "Next" button



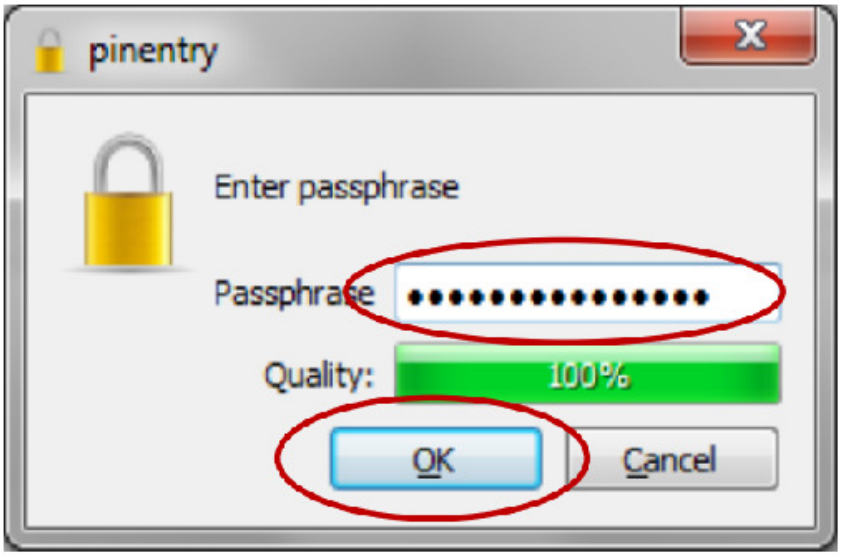
7. Review your entered values. If OK, click the "Create Key" button



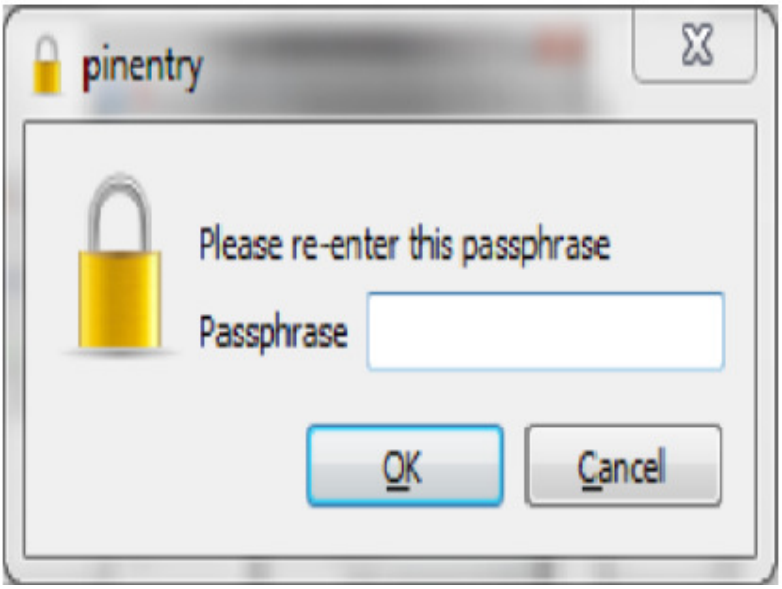
8. You will be asked to enter a passphrase



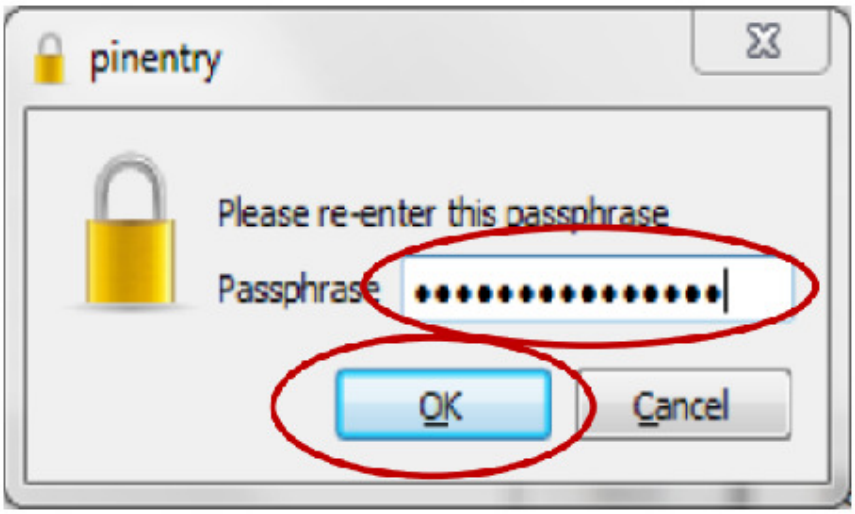
9. The passphrase should follow strong password standards. After you've entered your passphrase, click the "OK" button.



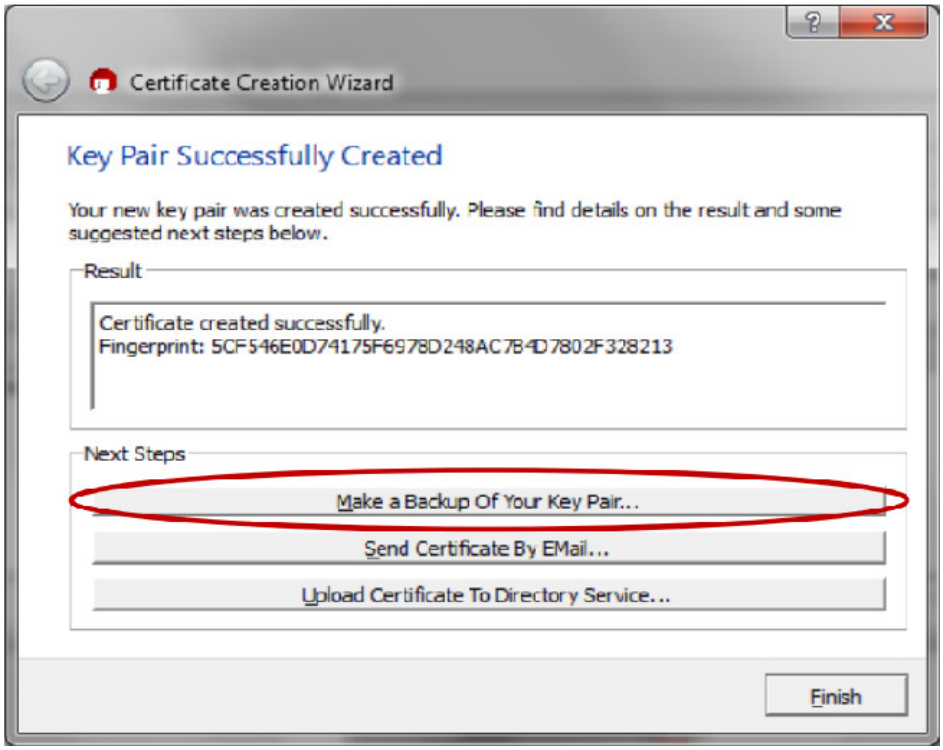
10. You will be asked to re-enter the passphrase



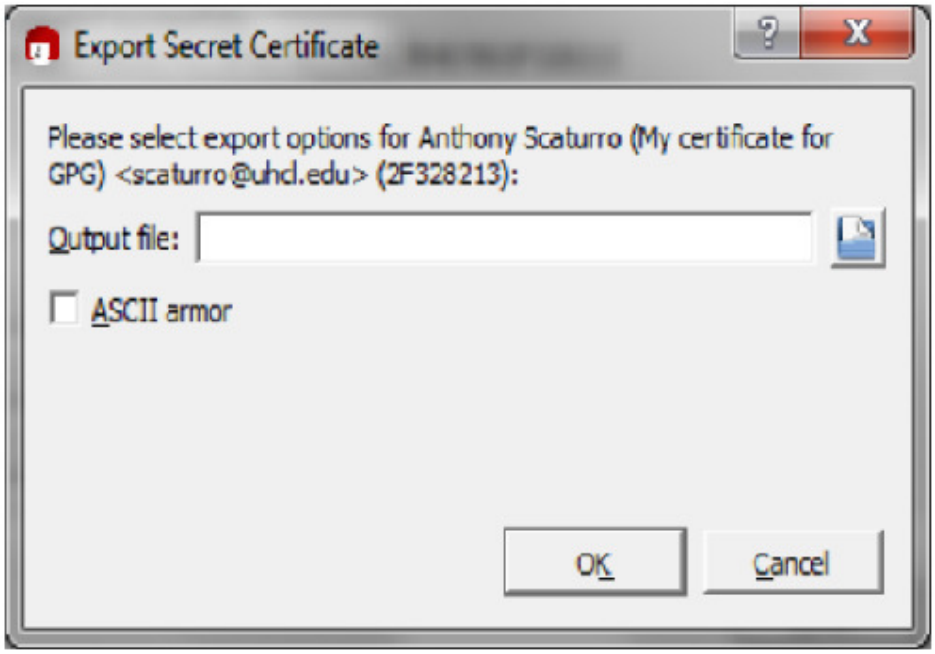
11. Re-enter the passphrase value. Then click the "OK" button. If the passphrases match, the certificate will be created.



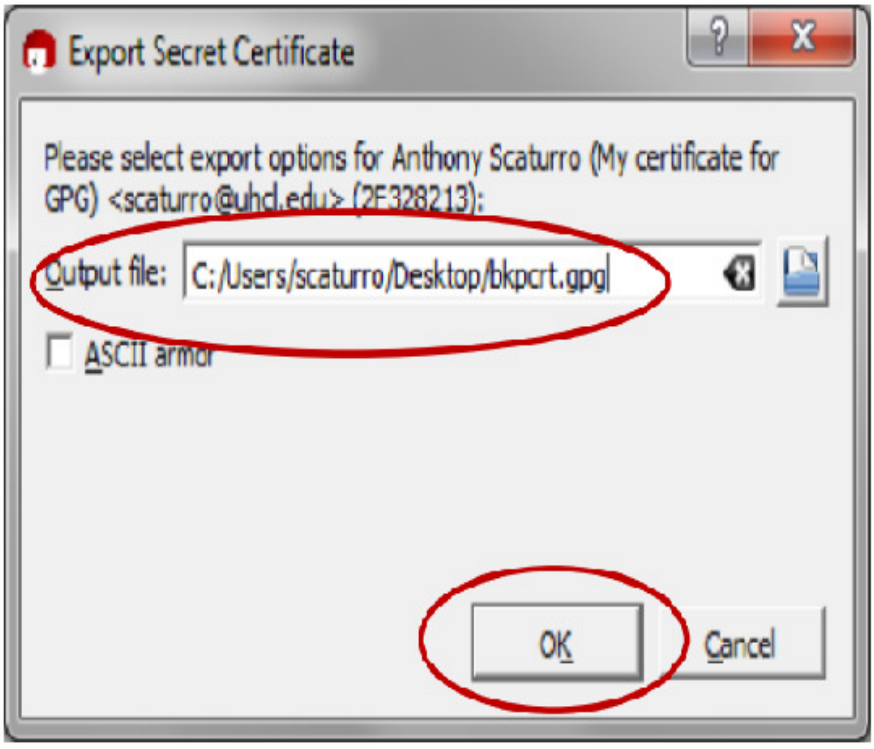
12. Once the certificate is created, the following screen will be displayed. You can save a backup of your public and private keys by clicking the "Make a backup Of Your Key Pair" button. This backup can be used to copy certificates onto other authorized computers.



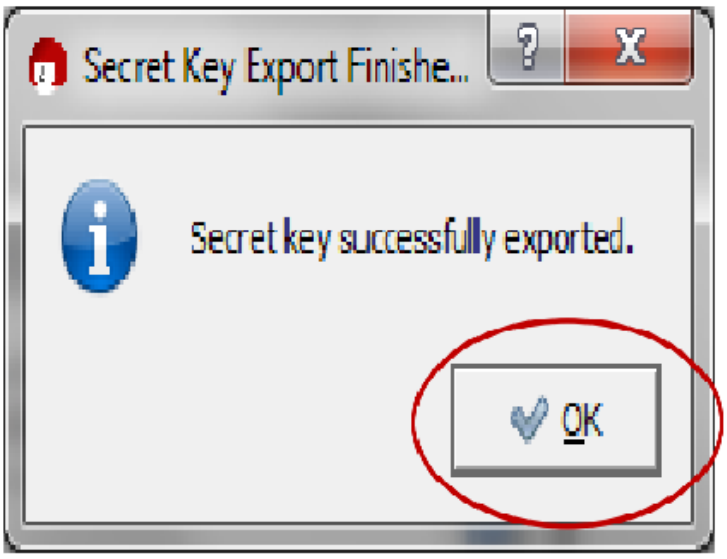
13. If you choose to backup your key pair, you will be presented with the following screen:



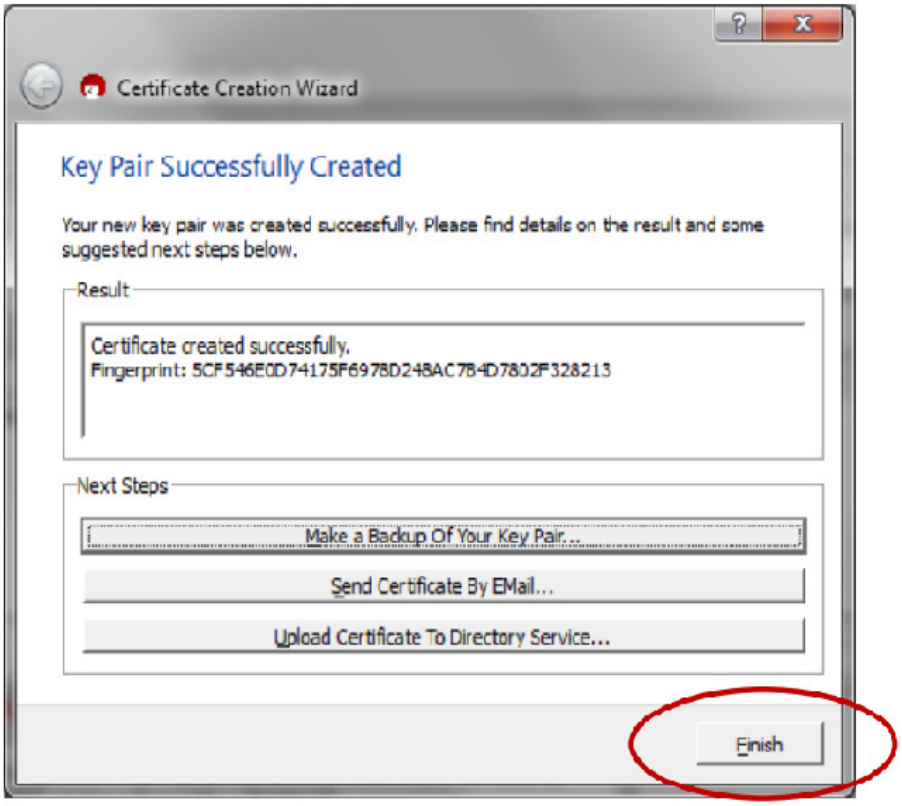
14. Specify the folder and name the file. Then click the "OK" button.



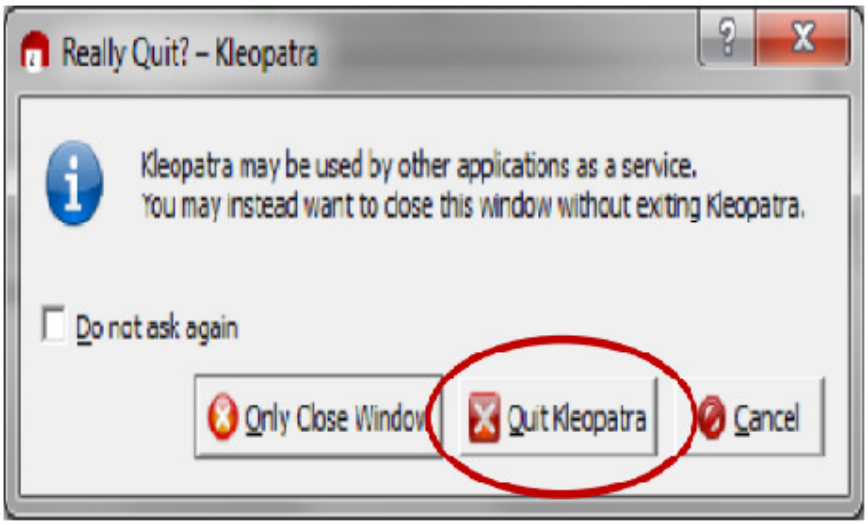
15. After the key is exported, the following will be displayed. Click the “OK” button.



16. You will be returned to the “Key Pair Successfully Created” screen. Click the “Finish” button.

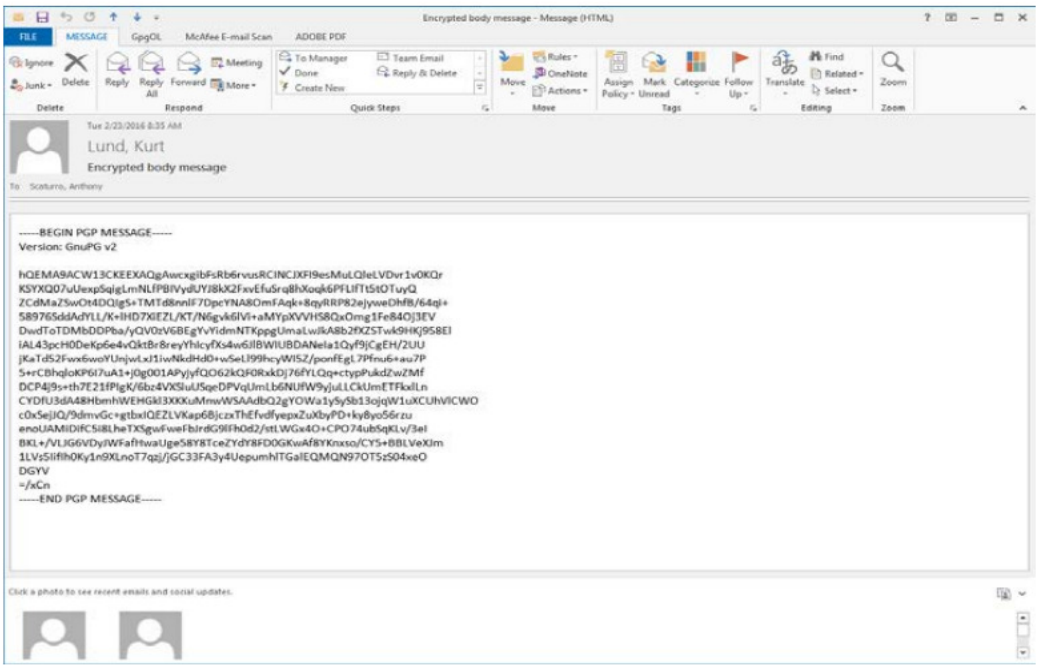


17. Before the program closes, you will need to confirm that you want to close the program by clicking on the "Quit Kleopatra" button

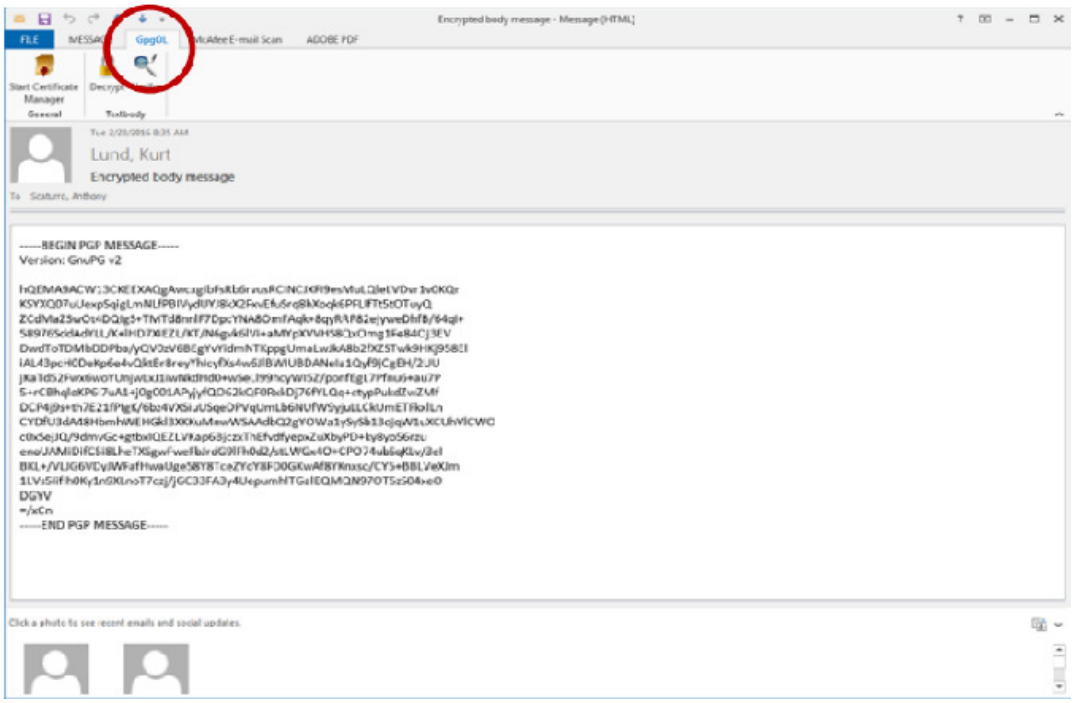


DECRYPTING AN ENCRYPTED E-MAIL THAT HAS BEEN SENT TO YOU:

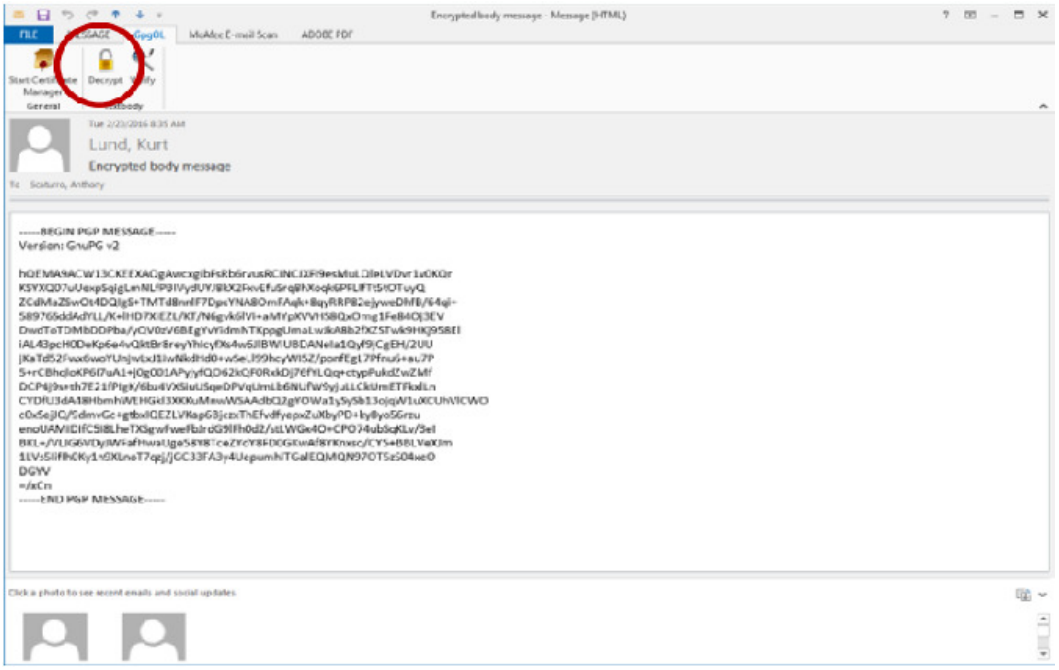
1. Open the e-mail message



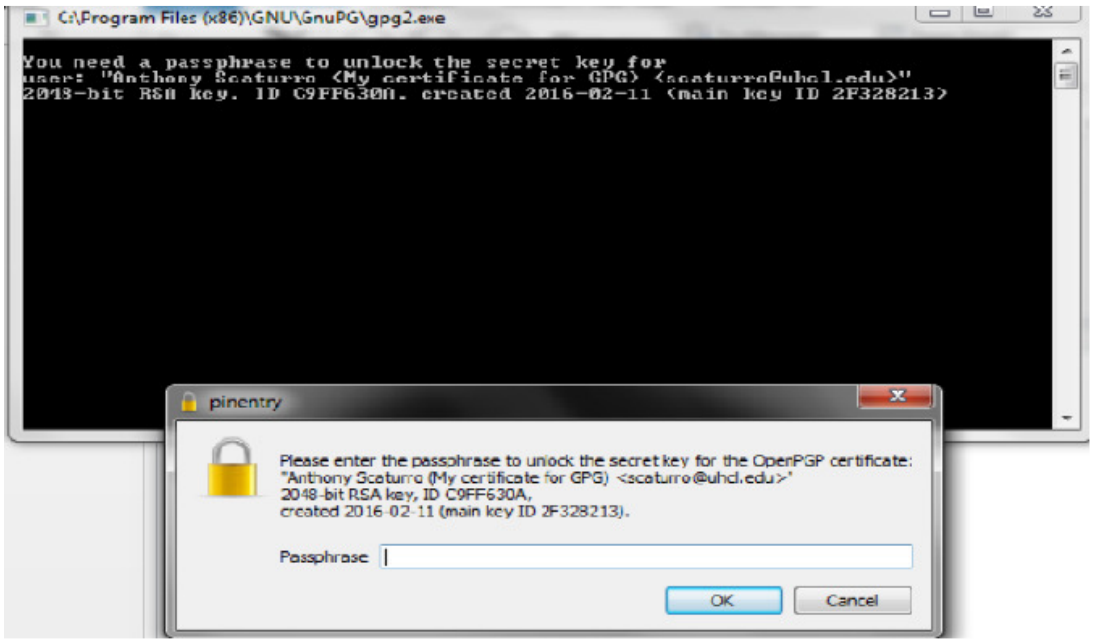
2. Select the GpgOL tab



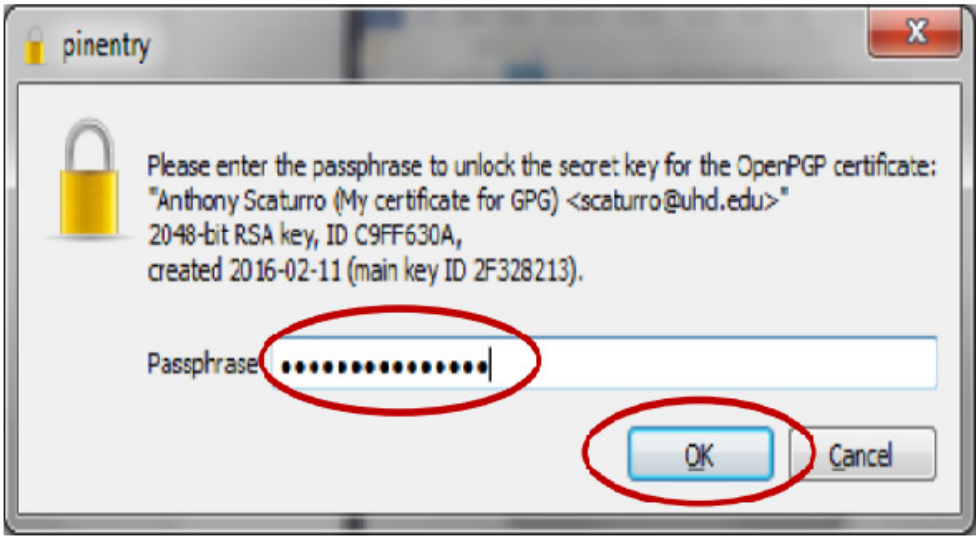
3. Click the "Decrypt" button



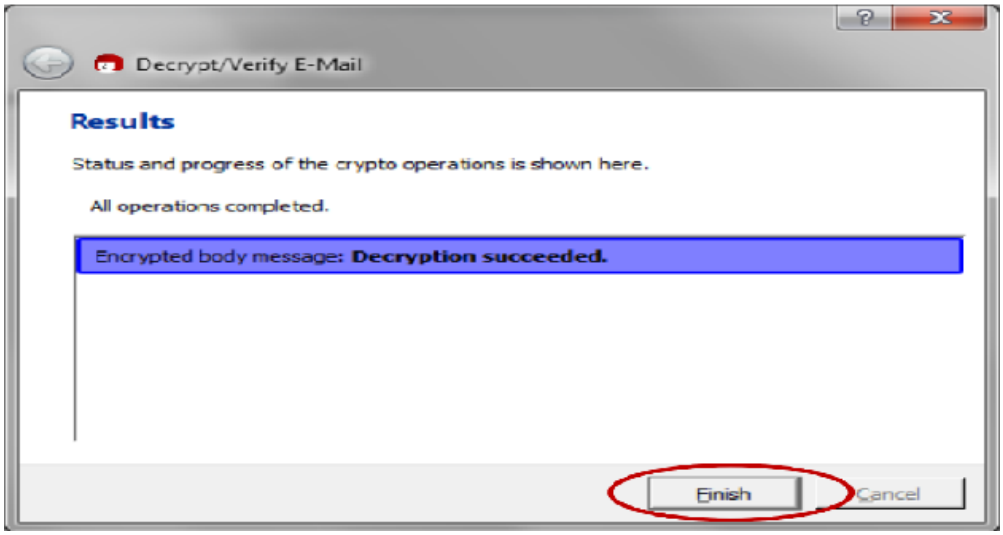
4. A command window will open along with a window that asks for the Passphrase to your private key that will be used to decrypt the incoming message.



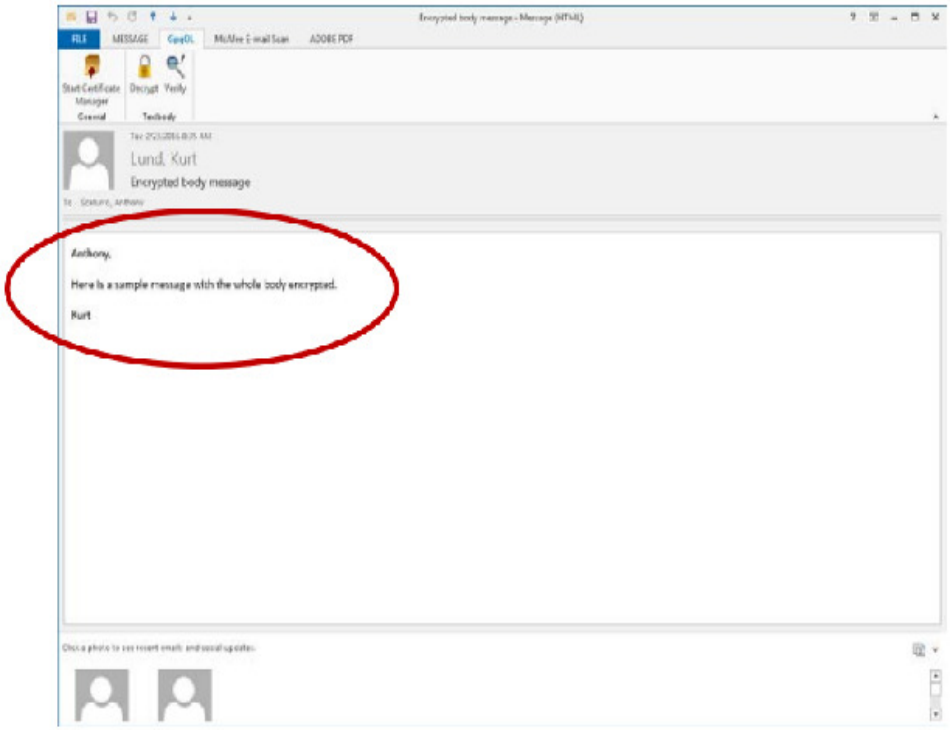
5. Enter your passphrase and click the "OK" button



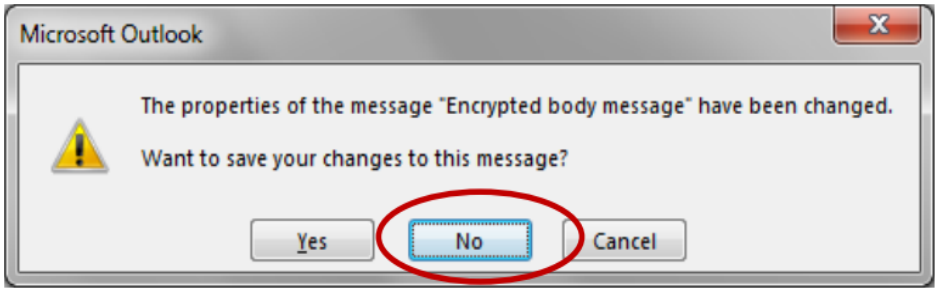
6. The results window will tell you if the decryption succeeded. Click the “Finish” button top close the window



7. Your unencrypted e-mail message body will be displayed.



8. When you close the e-mail you will be asked if you want to save the e-mail message in its unencrypted form. For maximum security, click the "No" button. This will keep the message encrypted within the e-mail system and will require you to enter your passphrase each time you reopen the e-mail message



RESULT:

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

EX. NO: 05

WORKING WITH KF SENSOR TOOL FOR CREATING AND MONITORING HONEYPOT

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's 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.

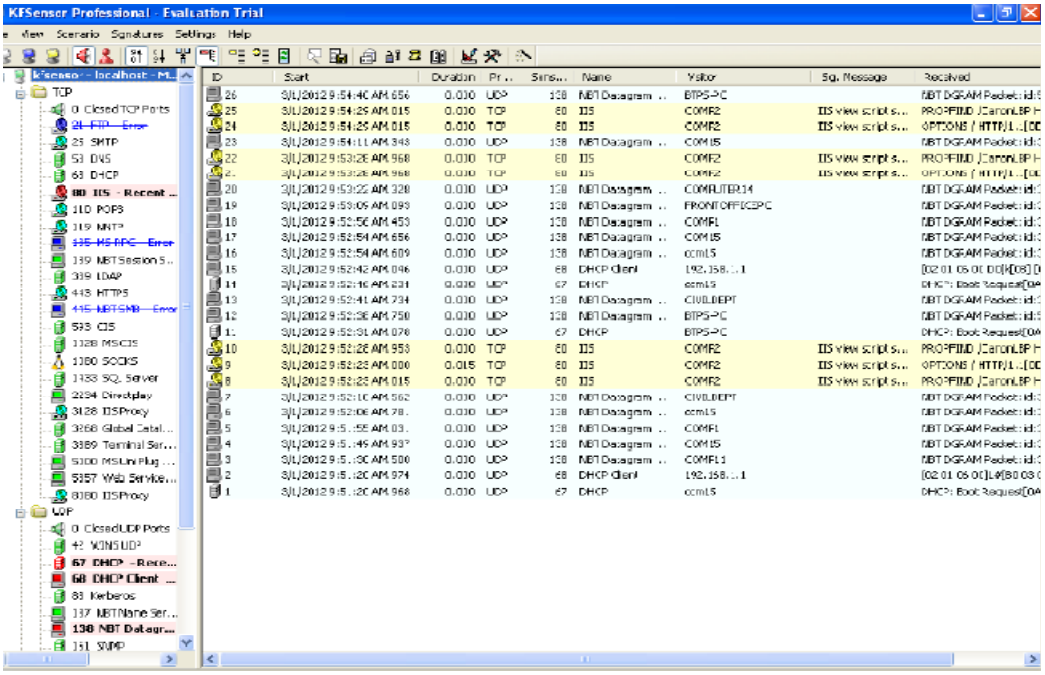
STEP-6: "Send the email and Send from email", enter the ID and Click Next.

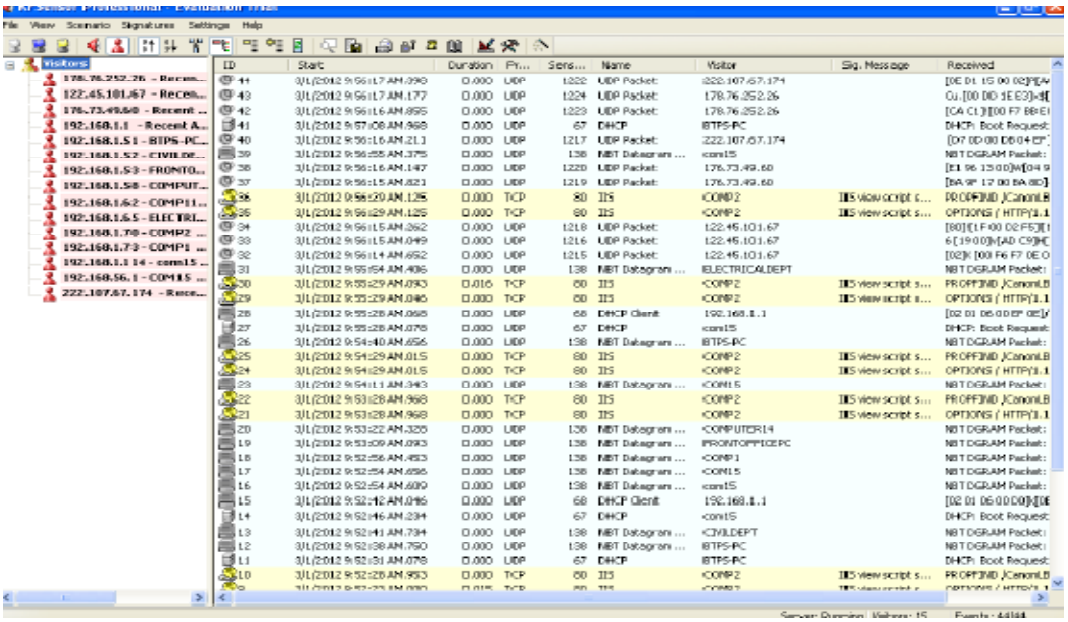
STEP-7: Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next.

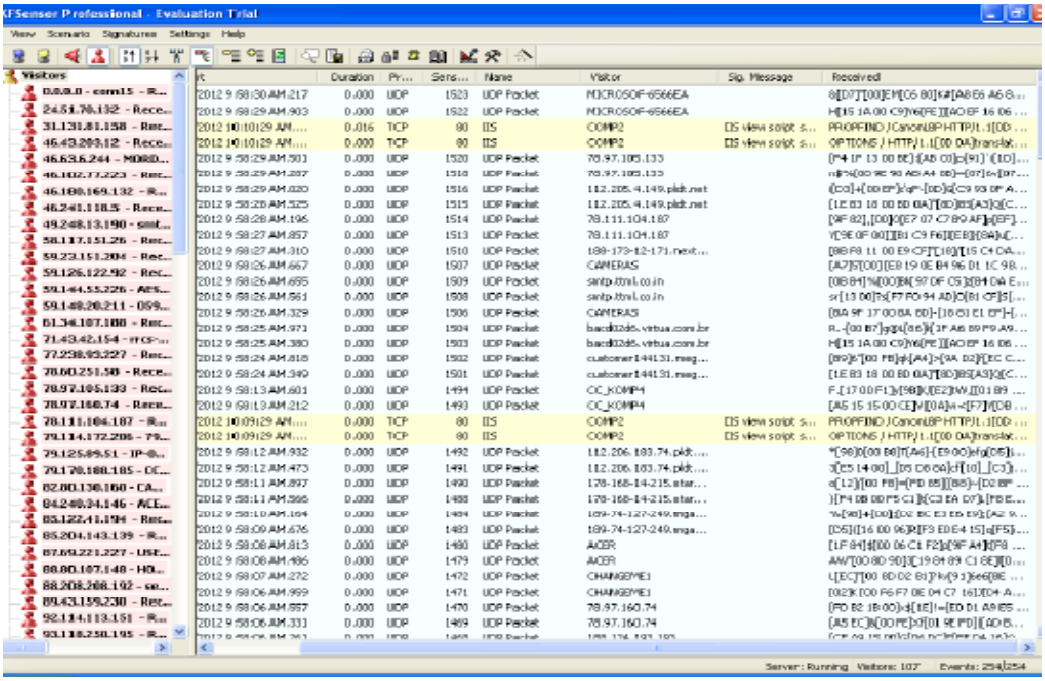
STEP-8: Select Install as System service and Click Next.

STEP-9: Click finish.

SCREENSHOTS:







RESULT:

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

EX. NO: 06

INSTALLATION OF ROOTKITS

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's the equivalent ofAdministrator 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 Autostart, Enable, Disable, System, Boot.

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

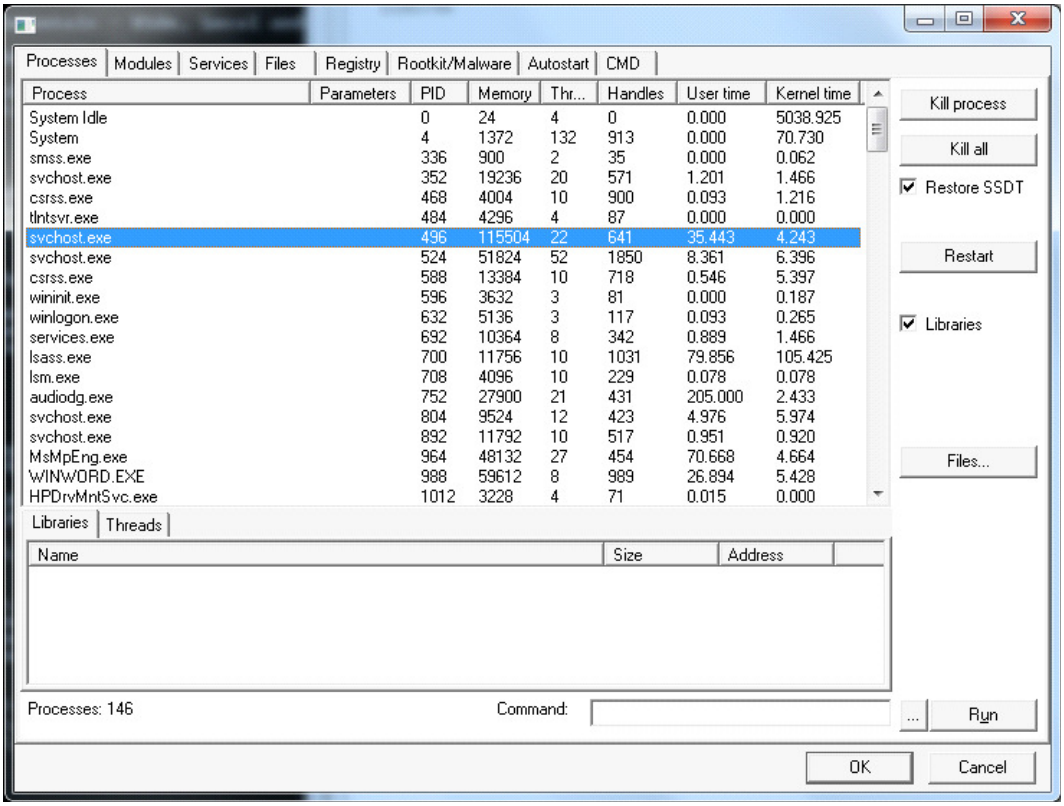
STEP-7: Registry displays Hkey_Current_user and Hkey_Local_Machine.

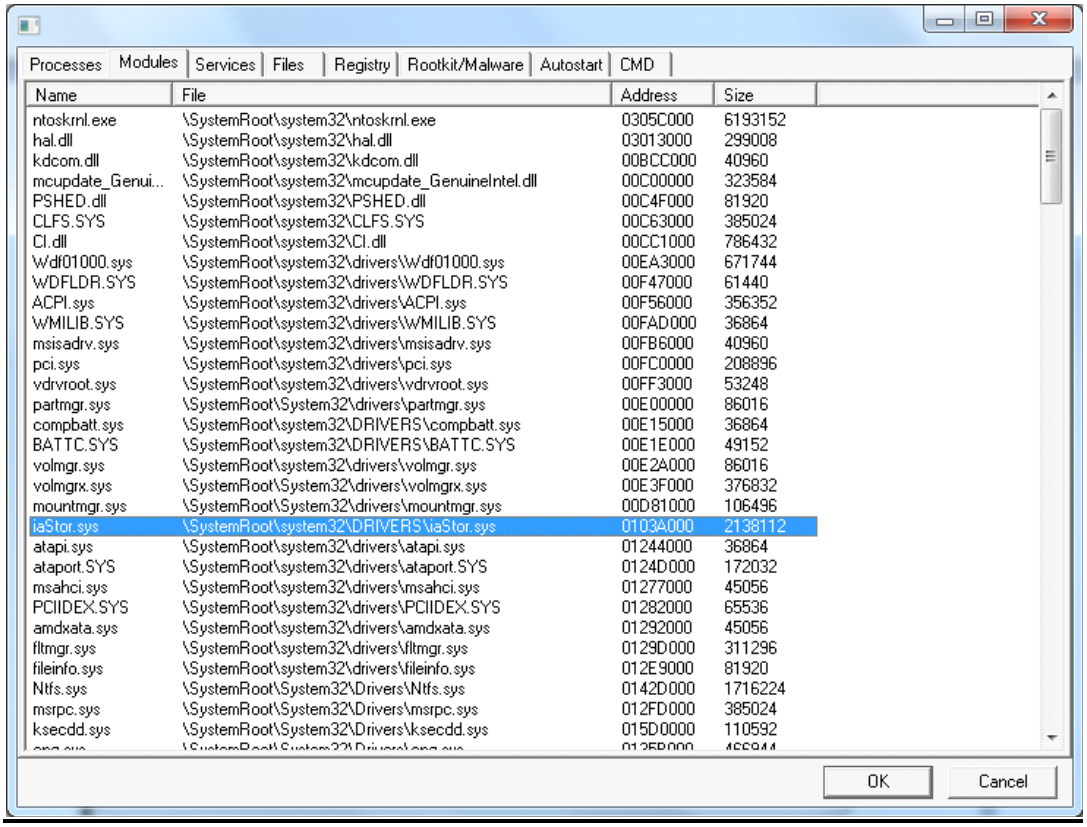
STEP-8: Rootkits / Malwares scans the local drives selected.

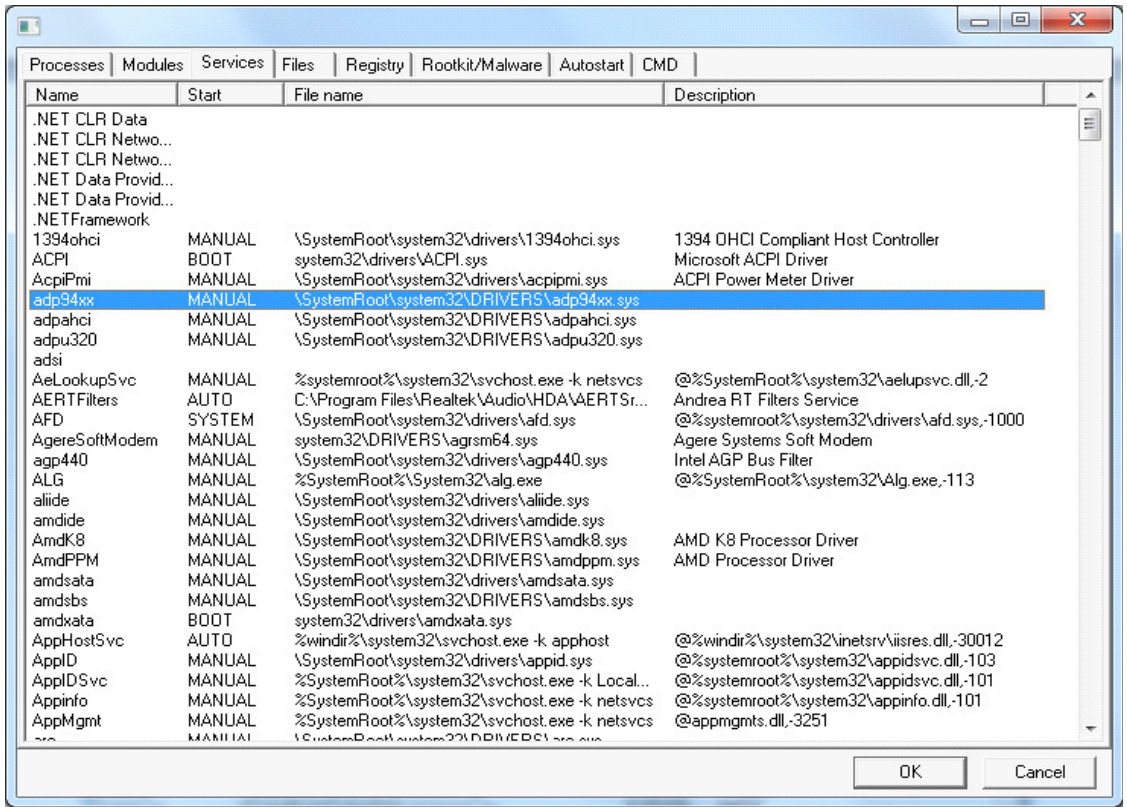
STEP-9: Autostart displays the registry base Autostart applications.

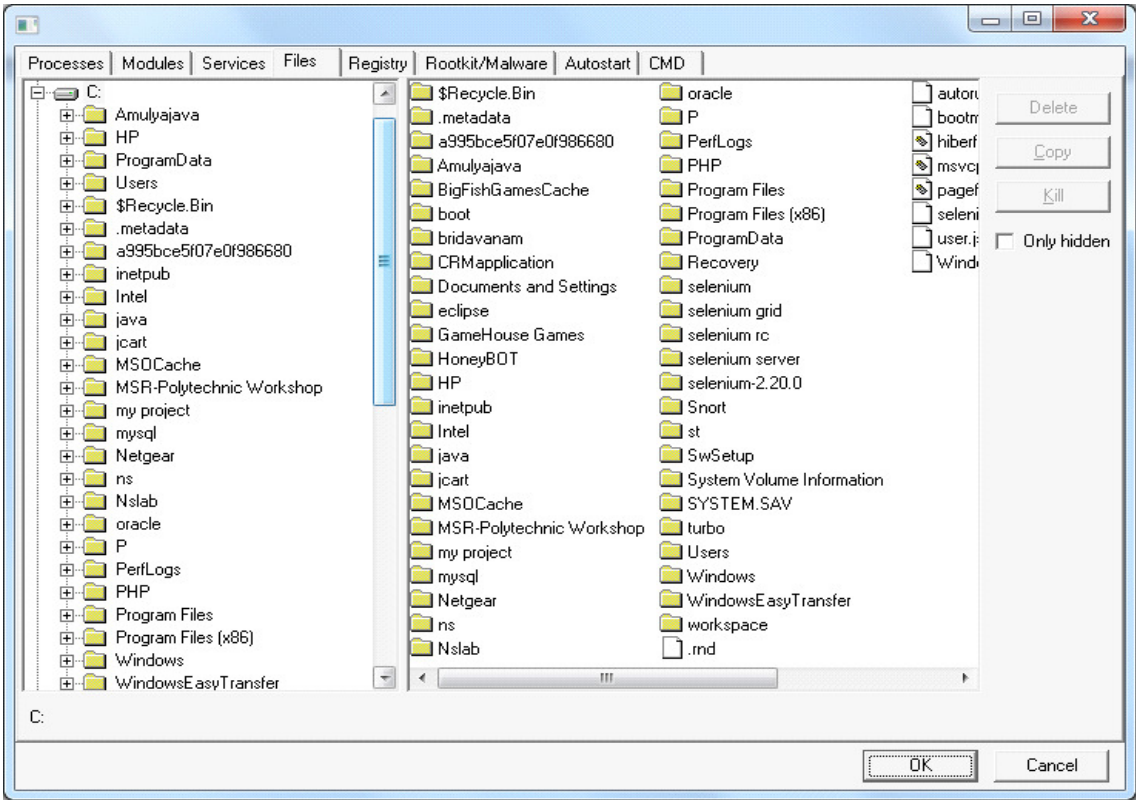
STEP-10:CMD allows the user to interact with command line utilities or Registry

SCREENSHOTS:




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
fltmgr.sys \SystemRoot\system32\drivers\fltmgr.sys 0129D000 311296
fileinfo.sys \SystemRoot\system32\drivers\fileinfo.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



RESULT:

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

EX. NO: 07

WORKING WITH NET STUMBLER TO PERFORM WIRELESS AUDIT ON A ROUTER

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:

- ✓ Verify that your network is set up the way you intended
- ✓ Find locations with poor coverage in your WLAN.
- ✓ Detect other networks that may be causing interference on your network
- ✓ Detect unauthorized "rogue" access points in your workplace
- ✓ Help aim directional antennas for long-haul WLAN links.
- ✓ 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.

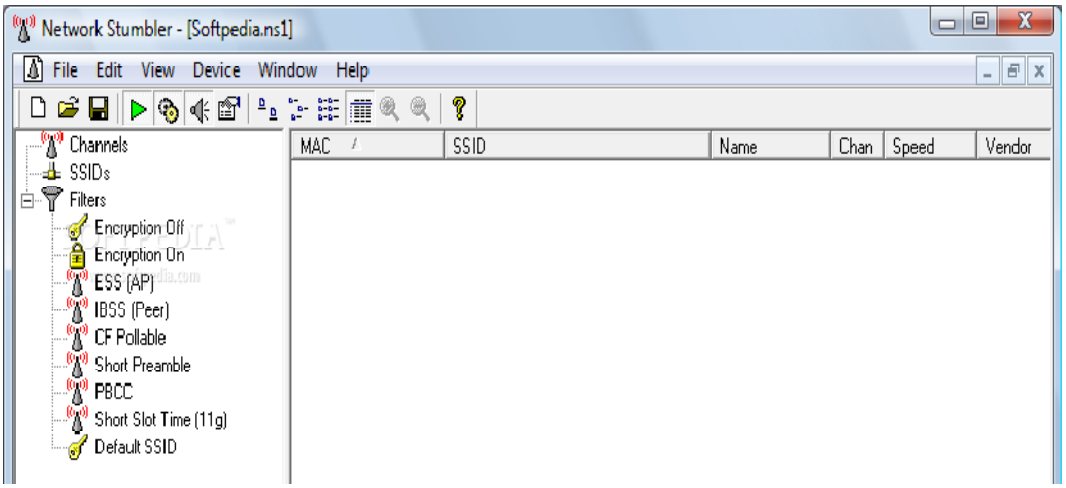
STEP-6: MAC assigned to Wireless Access Point is displayed on right hand pane.

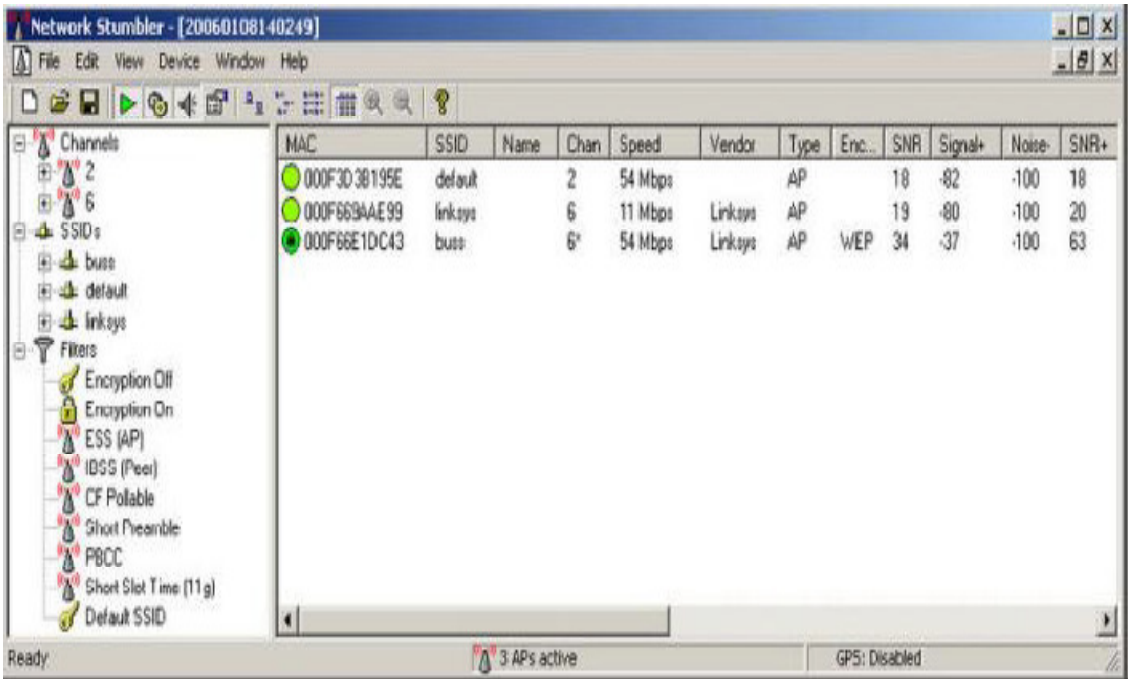
STEP-7: The next column displays the Access points Service Set Identifier[SSID] which is useful to crack the password.

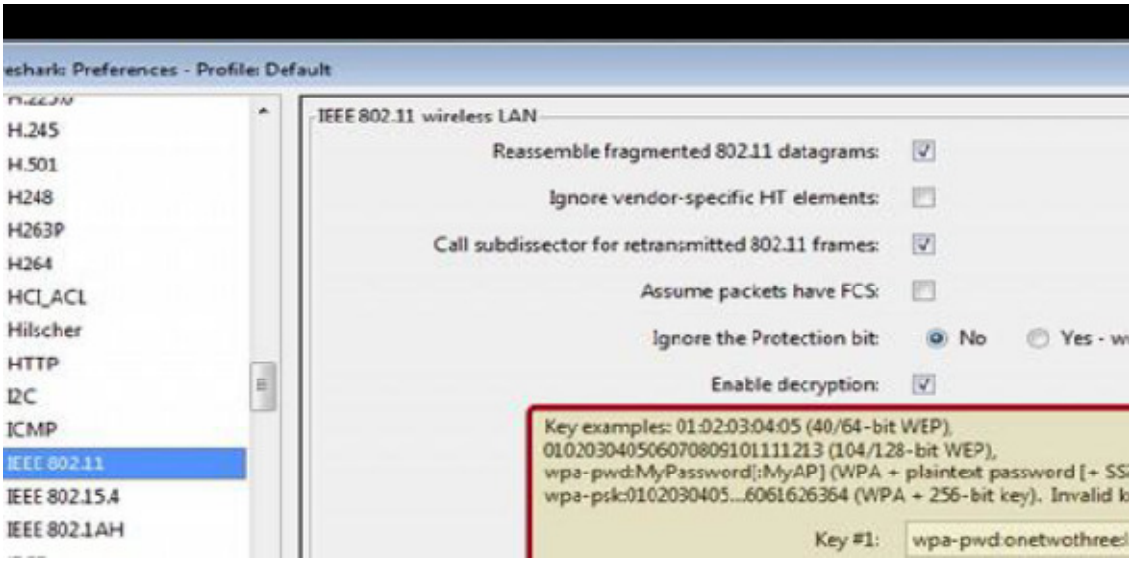
STEP-8: To decrypt use WireShark tool by selecting Edit → preferences → IEEE 802.11.

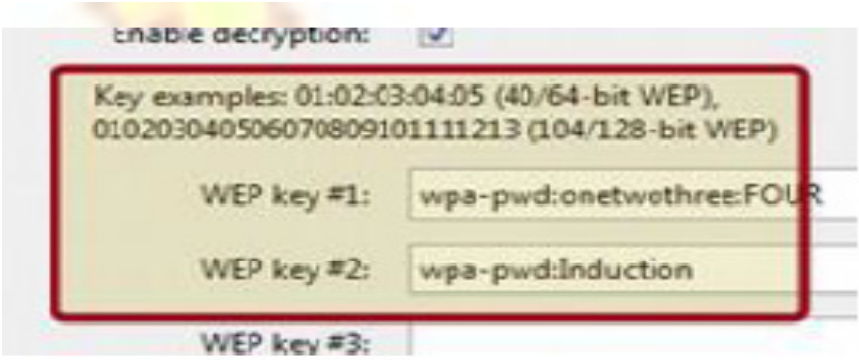
STEP-9: Enter the WEP keys as a string of hexadecimal numbers as A1B2C3D4E5.

SCREENSHOTS:



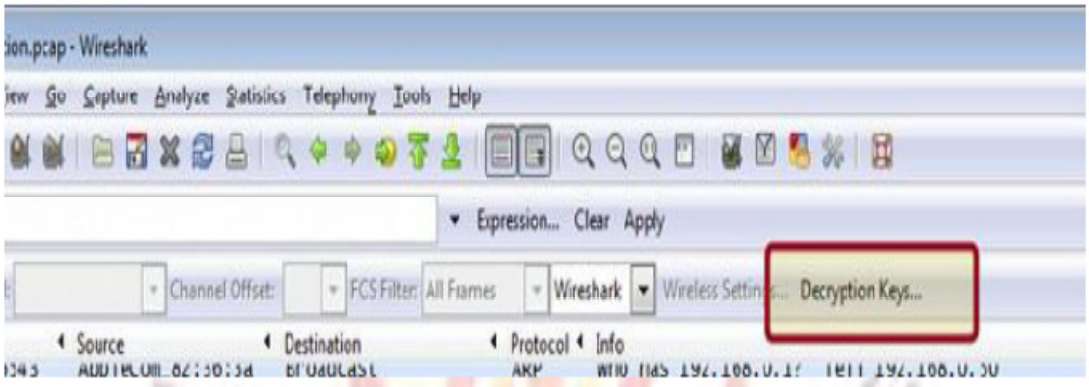






Adding Keys: Wireless Toolbar

- 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.
- If the toolbar isn't visible, you can show it by selecting View →Wireless Toolbar.
- 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:

RESULT:

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

EX. NO: 08

WORKING WITH SNORT TOOL TO DEMONSTRATE INTRUSION DETECTION SYSTEM

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:

- ✓ Signature-based intrusion detection systems
- ✓ 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/IPtraffic sniffers and analyzers. Through protocolanalysis and content searching and matching, Snort detects attack methods, including denial of service, buffer overflow, CGI attacks, stealthport scans, and SMB probes. When suspicious behavior is detected, Snort sends a real-time alert to syslog, a separate 'alerts' file, or to apop-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:

1. Sniffer mode
2. Packet Logger mode
3. 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 cnfigure 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.

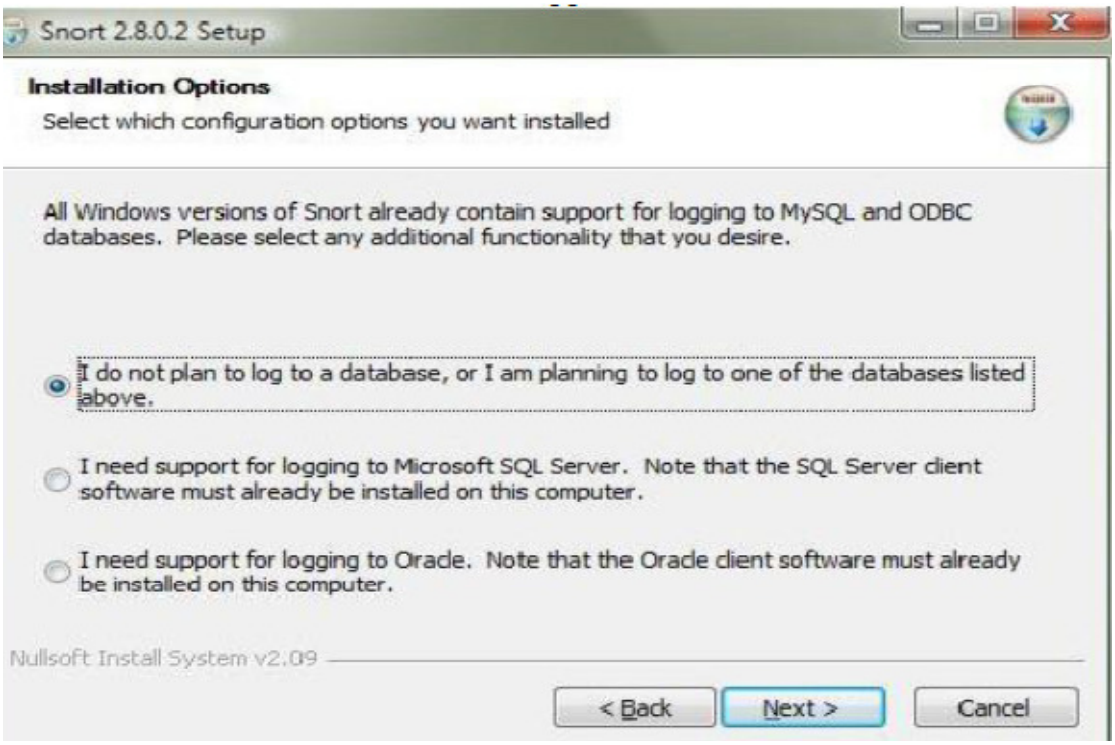
STEP-10: Skip the WinPcap driver installation.

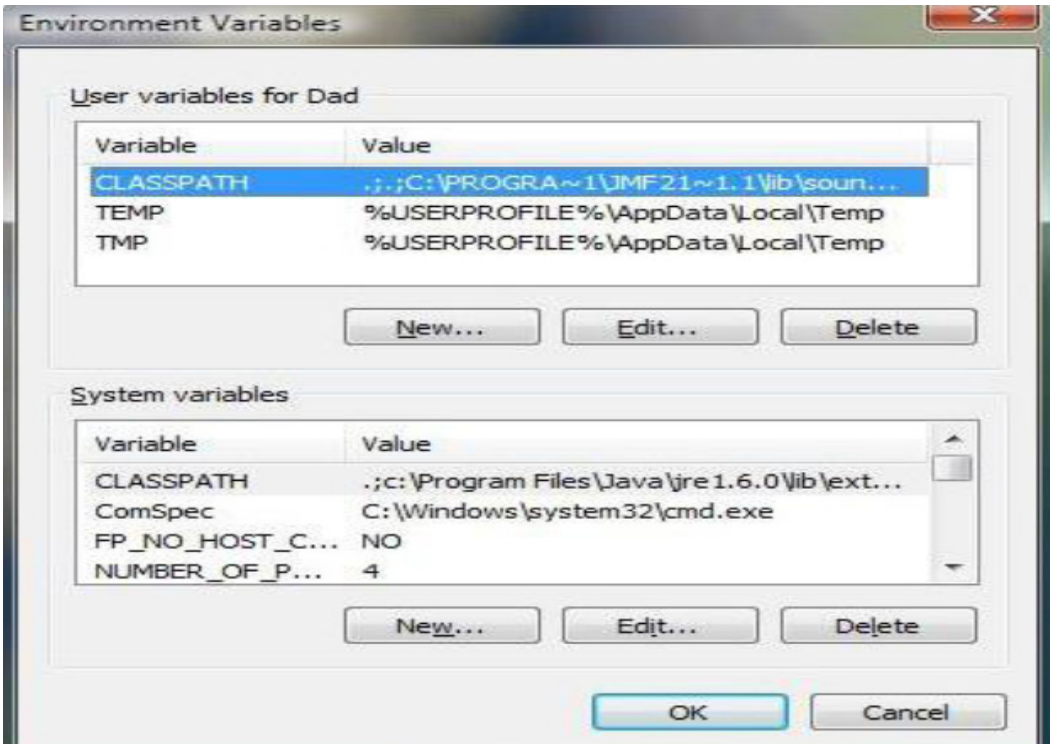
STEP-11: Add the path variable in windows environment variable by selecting new classpath.

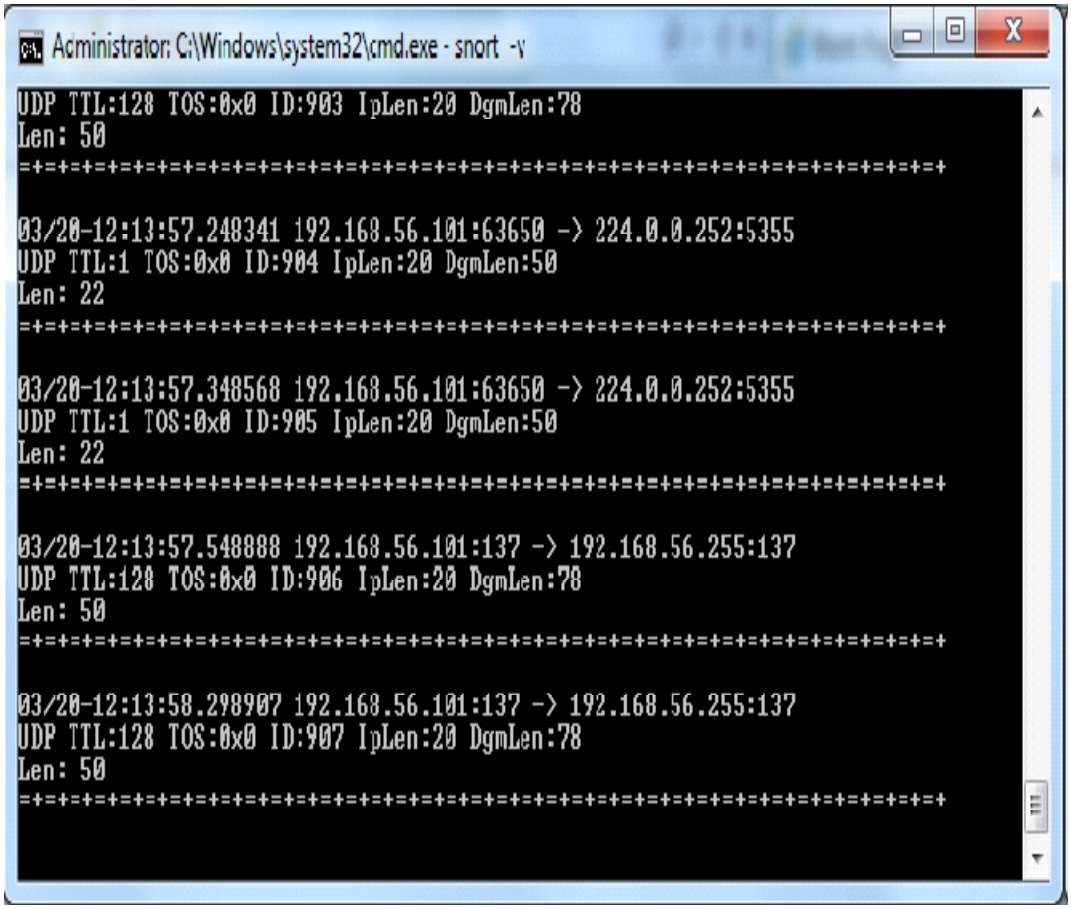
STEP-12: Create a path variable and point it at snort.exe variable name → path and variable value → c:\snort\bin.

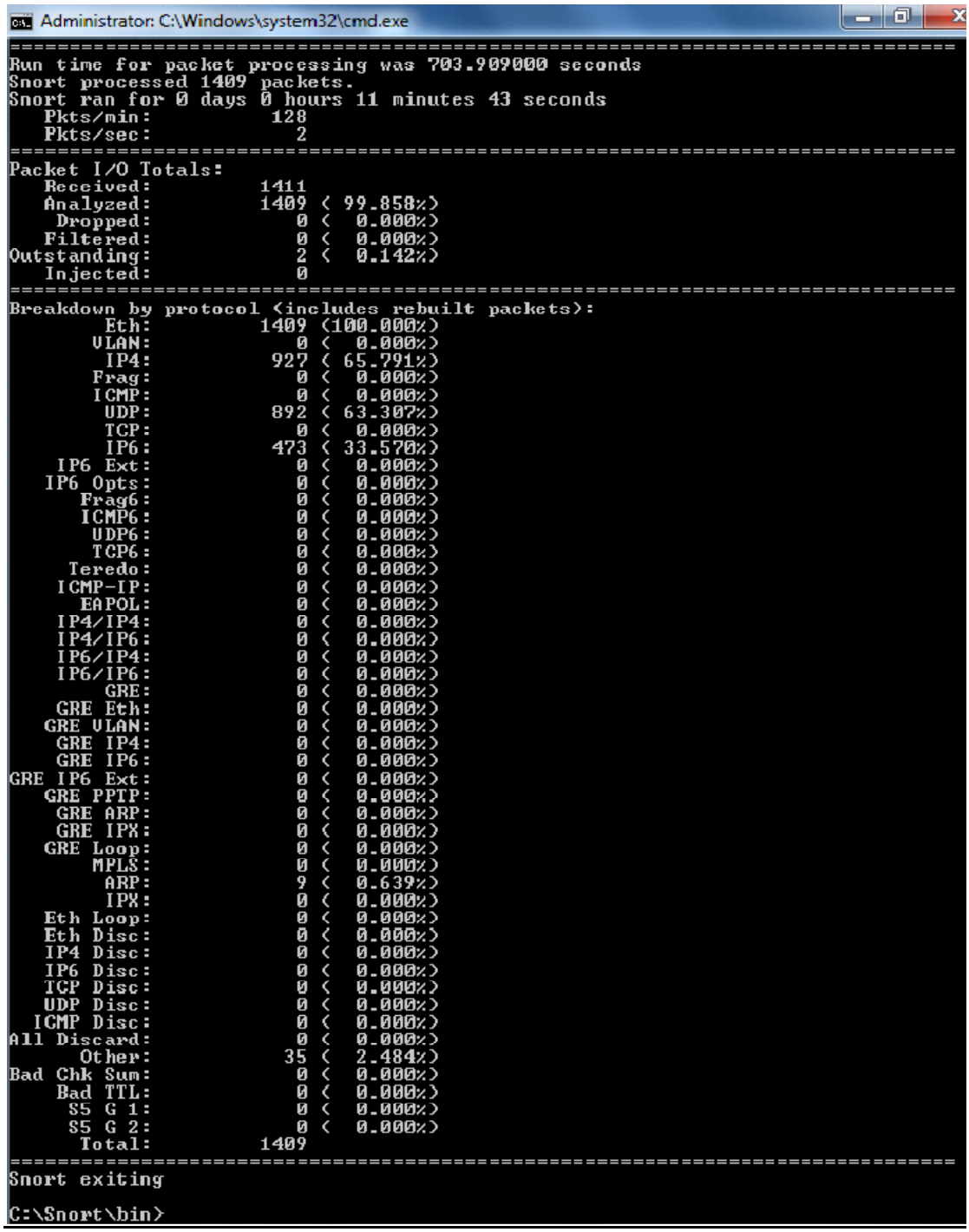
STEP-13: Click OK button and then close all dialog boxes. Open command prompt and type the following commands:

INSTALLATION PROCESS :






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



RESULT:

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