K.S.R. COLLEGE OF ENGINEERING (Autonomous): TIRUCHENGODE - 637 215

Vision of the Institution

- We envision to achieve status as an excellent educational institution in the global knowledge hub, making self-learners, experts, ethical and responsible engineers, technologists, scientists, managers, administrators and entrepreneurs who will significantly contribute to research and environment friendly sustainable growth of the nation and the world.

Mission of the Institution

- To inculcate in the students self-learning abilities that enable them to become competitive and considerate engineers, technologists, scientists, managers, administrators and entrepreneurs by diligently imparting the best of education, nurturing environmental and social needs.
- To foster and maintain a mutually beneficial partnership with global industries and Institutions through knowledge sharing, collaborative research and innovation.

DEPARTMENTOF COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

- To create ever green professionals for software industry, academicians for knowledge cultivation and researchers for contemporary society modernization.

Mission of the Department

- To produce proficient design, code and system engineers for software development.
- To keep updated contemporary technology and fore coming challenges for welfare of the society.

Programme Educational Objectives (PEOs)

- Figure out, formulate, analyze typical problems and develop effective solutions by imparting the idea and principles of science, mathematics, engineering fundamentals and computing.
- Competent professionally and successful in their chosen career through life-long learning.
- Excel individually or as member of a team in carrying out projects and exhibit social needs and follow professional ethics.

Programme Outcomes (POs) –COMPUTER SCIENCE AND ENIGNEERING

PO1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resource, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environmental and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader diverse teams, and in multidisciplinary settings.
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadcast context of technological change.
PSO1	**Technical competency:** Develop and Implement computer solutions that accomplish goals to the industry, government or research by exploring new technologies.
PSO2	**Professional awareness:** Grow intellectually and professionally in the chosen field.

Course Objectives:

- Learn to implement the algorithms DES, RSA, MD5,SHA-1.
- Learn to use network security tools like GnuPG, KF sensor, NetStrumbler.
- Know about Ethical Hacking to strengthen the security.

Course Outcomes: On Completion of this Course, the student will be able to

CO1 - Implement the cipher techniques.
CO2 - Develop the various security algorithms.
CO3 - Implement the digital signature scheme.
CO4 - Use different open source tools for network security and analysis.
CO5 - Act as ethical hacker to strengthen a network.

Course articulation matrix:

16CS722- Network Security Laboratory														
CO	**Course Outcomes**	**Programme Outcomes**												
		PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PS O1**	**PS O2**
C406.1	Implement the cipher techniques.	3	-	2	3	-	-	2	-	-	1	2	2	3	3
C406.2	Develop the various security algorithms.	2	3	3	2	-	-	-	-	-	1	2	-	3	3
C406.3	Implement the digital signature scheme.	3	-	3	2	3		-	-	-	1	2	-	3	3
C406.4	Use different open source tools for network security and analysis.	3	-	3	2	3	3	-	2	-	1	2	-	3	3
C406.5	Act as ethical hacker to strengthen a network.	2	-	3	2	3	-	-	-	3	2	2	3	3	3
Maximum Appeared value		3	**3**	3	2	**3**	**3**	**2**	2	3	1	2	3	3	3

		L	T	P	C
16CS722	**NETWORK SECURITY LABORATORY**	0	0	3	2

Prerequisite: Basic knowledge on computer networks(16CS515)

Objectives:

- Learn to implement the algorithms DES, RSA, MD5,SHA-1.
- Learn to use network security tools like GnuPG, KF sensor, NetStrumbler.
- Know about Ethical Hacking to strengthen thesecurity.

List of Experiments:

1. Implement the following Substitution and TranspositionTechniques:
   a) PlayfairCipher
   b) HillCipher
   c) VigenereCipher
   d) Rail fence - row & ColumnTransformation
2. Implementthefollowingcryptographicalgorithms:
   a) DES
   b) RSA Algorithm
   c) Diffiee-Hellman
   d) MD5
   e) SHA-1
3. Implement the SIGNATURE SCHEME - Digital SignatureStandard
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 (KFSensor)
6. Installation of root kits and study about the variety ofoptions
7. Perform wireless audit on an access point or a router and decrypt WEP and WPA.(NetStumbler)
8. Demonstrate intrusion detection system (ids) using any tool (snort or any others/w)
9. Case studies: Ethical Hacking - Minimum of any FIVE Activities to Maximum of any TEN Activities suggested in Michael T. Simpson, Ethical Hacking and Network defense, Course Technology, IndiaEdition,2010.

Total : 45 Periods

Course Outcomes: On Completion of this course, the student should will be able to

CO1 - Implement the ciphertechniques.
CO2 - Develop the various securityalgorithms.
CO3 - Implementthedigitalsignaturescheme.
CO4 - Use different open source tools for network security andanalysis.
CO5 - Act as ethical hacker to strengthen anetwork.

9		Case studies: ethical hacking	123		
		Average			

EX.NO:01(a)	IMPLEMENTATION OF PLAYFAIR CIPHER
DATE:	

Aim:

To write a program to encrypt a plain text and decrypt a cipher text using Playfair Cipher substitution technique.

Algorithm:

Step-1: To generate the key table, first fill the spaces in the table with the letters of the keyword, then fill the n remaining spaces with the rest of the letters of the alphabet in order.

Step-2: The key can be written in the top rows of the table, from left to right, or in some other pattern, such as a spiral beginning in the upper-left-hand corner and ending in the centre.

Step-3: The keyword together with the conventions for filling in the 5 by 5 table constitutes the cipher key. To encrypt a message, one would break the message into diagrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. Then apply the following 4 rules, to each pair of letters in the plaintext:.

Step-4: If both letters are the same (or only one letter is left), add an "X" after the first letter. Encrypt the new pair and continue. Some variants of Playfair use "Q" instead of "X", but any letter, itself uncommon as a repeated pair, will do.

Step-5: If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively.

Step-6: If the letters appear on the same column of your table, replace them with the letters immediately below respectively.

Step-7: 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. The order is important – the first letter of the encrypted pair is the one that lies on the same row as the first letter of the plaintext pair.

Step-8: 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. The order is important – the first letter of the encrypted pair is the one that lies on the same row as the first letter of the plaintext pair.

Program:

```
import java.awt.Point;
import java.util.*;
class Play
{
private static char[][] charTable;
private static Point[] positions;
private static String prepareText(String s, boolean chgJtoI)
{
s = s.toUpperCase().replaceAll("[^A-Z]", "");
return chgJtoI ? s.replace("J", "I") : s.replace("Q", "");
}
private static void createTbl(String key, boolean chgJtoI)
{
charTable = new char[5][5];
positions = new Point[26];
String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ", chgJtoI);
int len = s.length();
for (int i = 0, k = 0; i < len; i++)
{
char c = s.charAt(i);
if (positions[c - 'A'] == null)
{
charTable[k / 5][k % 5] = c;
positions[c - 'A'] = new Point(k % 5, k / 5);
k++;
}}}
private static String codec(StringBuilder txt, int dir)
{
int len = txt.length();
for (int i = 0; i < len; i += 2)
{
```

```
char a = txt.charAt(i);
char b = txt.charAt(i + 1);
int row1 = positions[a - 'A'].y;
int row2 = positions[b - 'A'].y;
int col1 = positions[a - 'A'].x;
int col2 = positions[b - 'A'].x;
if (row1 == row2)
{
col1 = (col1 + dir) % 5;
col2 = (col2 + dir) % 5;
}
else if (col1 == col2)
{
row1 = (row1 + dir) % 5;
row2 = (row2 + dir) % 5;
}
else
{
int tmp = col1;
col1 = col2;
col2 = tmp;
}
txt.setCharAt(i, charTable[row1][col1]);
txt.setCharAt(i + 1, charTable[row2][col2]);
}
return txt.toString();
}
private static String encode(String s)
{
StringBuilder sb = new StringBuilder(s);
for (int i = 0; i < sb.length(); i += 2)
{
if (i == sb.length() - 1)
```

```
{
sb.append(sb.length() % 2 == 1 ? 'X' : "");
}
else if (sb.charAt(i) == sb.charAt(i + 1))
{
sb.insert(i + 1, 'X');
}}
return codec(sb, 1);
}
private static String decode(String s)
{
return codec(new StringBuilder(s), 4);
}
public static void main (String[] args) throws java.lang.Exception
{
String key = "mysecretkey";
String txt = "CRYPTOLABS"; /* make sure string length is even */
/* change J to I */
boolean chgJtoI = true;
createTbl(key, chgJtoI);
String enc = encode(prepareText(txt, chgJtoI));
System.out.println("simulation of Playfair Cipher");
System.out.println("input message : " + txt);
System.out.println("encoded message : " + enc);
System.out.println("decoded message : " + decode(enc));
}}
```

Result:

Thus the java program to implement Playfair Cipher was executed and the output was verified.

EX.NO:01(b)	IMPLEMENTATION OF HILL CIPHER
DATE:	

Aim:

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

Algorithm:

Step-1: The Hill cipher is a substitution cipher invented by Lester S. Hill in 1929.

Step-2: Each letter is represented by a number modulo 26. To encrypt a message, each block of n letters is multiplied by an invertible n × n matrix, again modulus 26.

Step-3: To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo26).

Step-4: The cipher can, be adapted to an alphabet with any number of letters.

Step-5: All arithmetic just needs to be done modulo the number of letters instead of modulo 26.

Program:

```
import java.util.*;
class hill
{
/* 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");
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);
}
}
```

Result:

Thus the java program to implement Hill Cipher was executed and the output was verified.

EX.NO:01(c)	IMPLEMENTATION OF VIGENERE CIPHER
DATE:	

Aim:

To write a program for encryption and decryption using Vigenere Cipher substitution technique

Step-1: The Vigenere Cipher is a method of encrypting alphabetic text by using a series of different Caesar ciphers based on the letters of a keyword.

Step-2: It is a simple form of polyalphabetic substitution.

Step-3: To encrypt, a table of alphabets can be used, termed a Vigenere square, or Vigenere 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.

Step-4: At different points in the encryption process, the cipher uses a different alphabet from one of the rows used.

Step-5: The alphabet at each point depends on a repeating keyword.

Program:

```
import java.io.*;
importjava.util.Scanner;
public class VigenereCipher
{
public static String encrypt(String text, final String key)
{
String res = "";
text = text.toUpperCase();
for (int i = 0, j = 0; i <text.length(); i++) {
char c = text.charAt(i);
if (c < 'A' || c > 'Z')
continue;
res += (char) ((c + key.charAt(j) - 2 * 'A') % 26 + 'A');
```

```
j = ++j % key.length(); }
return res;
}
public static String decrypt(String text, final String key)
{
String res = "";
text = text.toUpperCase();
for (int i = 0, j = 0; i <text.length(); i++)
{
char c = text.charAt(i);
if (c < 'A' || c > 'Z')
continue;
res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');
j = ++j % key.length();
} return res; }
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
System.out.println("Enter the line : ");
String message = s.nextLine();
System.out.println("Enter the key");
String key = s.nextLine();
String encryptedMsg = encrypt(message, key);
System.out.println("String: " + message);
System.out.println("Encrypted message: " + encryptedMsg);
System.out.println("Decrypted message: " + decrypt(encryptedMsg, key));
}
}
```

Result:

Thus the java program to implement Vigenere Cipher was executed and the output was verified.

EX.NO:01(d)	IMPLEMENTATION OF RAIL FENCE – ROW & COLUMN TRANSFORMATION TECHNIQUE
DATE:	

Aim:

To write a program for encryption and decryption using Rail-Fence row and column transposition technique.

Algorithm:

Step-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.

Step-2: When we reach the top rail, the message is written downwards again until the whole plaintext is written out.

Step-3: The message is then read off in rows.

Program:

```
import java.io.*;
importjava.util.Scanner;
classRailFenceBasic
{
int depth;
String Encryption(String plainText,int depth)throws Exception
{
int r=depth,len=plainText.length();
int c=len/depth;
char mat[][]=new char[r][c];
int k=0;
String cipherText="";
for(int i=0;i<c;i++)
{
for(int j=0;j<r;j++)
{
if(k<len)
mat[j][i]=plainText.charAt(k++);
```

```
else
mat[j][i]='X';
}
}
for(int i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
cipherText+=mat[i][j];
}
}
returncipherText;
}
String Decryption(String cipherText,int depth)throws Exception
{
int r=depth,len=cipherText.length();
int c=len/depth;
char mat[][]=new char[r][c];
int k=0;
String plainText="";
for(int i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
mat[i][j]=cipherText.charAt(k++);
}
}
for(int i=0;i<c;i++)
{
for(int j=0;j<r;j++)
{
plainText+=mat[j][i];
}
}
```

```
eturnplainText;
}
}
public class RailfenceCipher
{
public static void main(String[] args)throws Exception
{
RailFenceBasicrf=new RailFenceBasic();
Scanner scn=new Scanner(System.in);
int depth;
String plainText,cipherText,decryptedText;
System.out.println("Enter plain text:");
plainText=scn.nextLine();
System.out.println("Enter depth for Encryption:");
depth=scn.nextInt();
cipherText=rf.Encryption(plainText,depth);
System.out.println("\nEncrypted text is: "+cipherText);
decryptedText=rf.Decryption(cipherText, depth);
System.out.println("\nDecrypted text is: "+decryptedText);
}
}
```

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the java program to implement Rail-Fence row and column transformation was executed and the output was verified.

EX.NO:02(a)	IMPLEMENTATION OF DATA ENCRYPTION STANDARD (DES)
DATE:	

Aim:

To write a program to implement Data Encryption Standard (DES) for encryption and decryption.

Algorithm:

Step-1: The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

Step-2: DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64- bit.

Step-3: Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

Step-4: General Structure of DES is depicted in the following illustration.

Program:

```
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); //to get the key
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"); //generates the key
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();
}
}
```

Result:

Thus the java program to implement Data Encryption Standard (DES) was executed and the output was verified.

EX.NO:02(b)	IMPLEMENTATION OF RSA ALGORITHM
DATE:	

Aim:

To write a program to implement RSA algorithm for encryption and decryption.

Algorithm:

Step-1: Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key.

Step-2: The process followed in the generation of keys is described below −

Step-3: Generate the RSA modulus (n)

- Select two large primes, p and q.
- Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.

Step-4: Find Derived Number (e)

- Number e must be greater than 1 and less than (p − 1)(q − 1).
- There must be no common factor for e and (p − 1)(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.

Step-5: Form the public key

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

Step-6: Generate the private key

- Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
- Number d is the inverse of e modulo (p - 1)(q – 1). This means that d is the number less

than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1).

Step-7: This relationship is written mathematically as follows ed = 1 mod (p − 1)(q − 1)

Step-8: The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

Program:

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA
{
private BigInteger p;
private BigInteger q;
private BigInteger N;
private BigInteger phi;
private BigInteger e;
private BigInteger d;
private int bitlength = 1024;
private Random r;
public RSA() {
r = new Random();
p = BigInteger.probablePrime(bitlength, r);
q = BigInteger.probablePrime(bitlength, r);
N = p.multiply(q);
phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
e = BigInteger.probablePrime(bitlength / 2, r);
while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
{
e.add(BigInteger.ONE);
}
d = e.modInverse(phi);
}
public RSA(BigInteger e, BigInteger d, BigInteger N){
this.e = e;
this.d = d;
this.N = N;
}
```

```
public static void main(String[] args) throws IOException
{
RSA rsa = new RSA();
DataInputStream in = new DataInputStream(System.in);
String teststring;
System.out.println("Enter the plain text:");
teststring = in.readLine();
System.out.println("Encrypting String: " + teststring);
System.out.println("String in Bytes: " + bytesToString(teststring.getBytes()));
byte[] encrypted = rsa.encrypt(teststring.getBytes());
byte[] decrypted = rsa.decrypt(encrypted);
System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
System.out.println("Decrypted String: " + new String(decrypted));
}private static String bytesToString(byte[] encrypted)
{
String test = "";
for (byte b : encrypted)
{
test += Byte.toString(b);
}
return test; }
public byte[] encrypt(byte[] message)
{
return (new BigInteger(message)).modPow(e, N).toByteArray();
}
public byte[] decrypt(byte[] message)
{
return (new BigInteger(message)).modPow(d,
N).toByteArray();
}}
```

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the java program to implement RSA algorithm was executed and the output was verified.

EX.NO:02(c)	IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM
DATE:	

Aim:

To write a program to implement Diffie Hellman Key Exchange Algorithm for Encryption and Decryption.

Algorithm:

Global Public Elements:

Let q be a prime number and α is a primitive root of q.

Step-1: User A Key Generation:

Select private XA where XA < q

Calculate public YA where YA = αXA mod q

Step-2: User B Key Generation:

Select private XB where XB < q

Calculate public YB where YB = αXB mod q

Step-3: Calculation of Secret Key by User A

K = (YB) XA mod q

Step-4: Calculation of Secret Key by User B:

K = (YA) XB mod q

Program:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigInteger;
public class DeffieHellman {
public static void main(String[]args)throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter prime number:");
BigInteger p=new BigInteger(br.readLine());
```

```
System.out.print("Enter primitive root of "+p+":");
BigInteger g=new BigInteger(br.readLine());
System.out.println("Enter value for x less than "+p+":");
BigInteger x=new BigInteger(br.readLine());
BigInteger R1=g.modPow(x,p);
System.out.println("R1="+R1);
System.out.print("Enter value for y less than "+p+":");
BigInteger y=new BigInteger(br.readLine());
BigInteger R2=g.modPow(y,p);
System.out.println("R2="+R2);
BigInteger k1=R2.modPow(x,p);
System.out.println("Key calculated at Alice's side:"+k1);
BigInteger k2=R1.modPow(y,p);
System.out.println("Key calculated at Bob's side:"+k2);
System.out.println("deffie hellman secret key Encryption has Taken");
}
}
```

Result:

Thus the java program to implement Diffie-Hellman algorithm was executed and the output was verified.

EX.NO:02(d)	IMPLEMENTATION OF MESSAGE DIGEST ALGORITHM (MD5)
DATE:	

Aim:

To write a program to implement Message Digest Algorithm(MD5).

Algorithm:

Step-1: The MD5 message-digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32-digit hexadecimal number.

Step-2: MD5 has been utilized in a wide variety of cryptographic applications and is also commonly used to verify data integrity.

Program:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
public class Md5 {
public static String getMD5(String input) {
try{
MessageDigest md = MessageDigest.getInstance("MD5");
byte[] messageDigest = md.digest(input.getBytes());
BigInteger number = new BigInteger(1, messageDigest);
String hashtext = number.toString(16);
while(hashtext.length() < 32) {
hashtext = "0"+ hashtext;
}
return hashtext;
}
catch(NoSuchAlgorithmException e)
{
throw new RuntimeException(e);
```

```
}
}
public static void main(String[] args) throws NoSuchAlgorithmException
{
Scanner s=new Scanner(System.in);
System.out.println("\nEnter the Input String: ");
String str=s.nextLine();
System.out.println("\nMessage Digest(32 bit long): "+getMD5(str));
}
}
```

Result:

Thus the java program to implement Message Digest Algorithm (MD5) was executed and the output was verified.

EX.NO:02(e)	IMPLEMENTATION OF SECURE HASH ALGORITHM (SHA-1)
DATE:	

Aim:

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

Algorithm:

Secure Hash Algorithm-1 (SHA-1):

Step 1: Append Padding Bits….

Message is "padded" with a 1 and as many 0's as necessary to bring the message length to 64 bits less than an even multiple of 512.

Step 2: Append Length....

64 bits are appended to the end of the padded message. These bits hold the binary format of 64 bits indicating the length of the original message.

Step 3: Prepare Processing Functions….

SHA1 requires 80 processing functions defined as:

f(t;B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19)

f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)

f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t<=59)

f(t;B,C,D) = B XOR C XOR D (60 <= t <= 79)

Step 4: Prepare Processing Constants....

SHA1 requires 80 processing constant words defined as:

K(t) = 0x5A827999 (0 <= t <= 19)

K(t) = 0x6ED9EBA1 (20 <= t <= 39)

K(t) = 0x8F1BBCDC (40 <= t <= 59)

K(t) = 0xCA62C1D6 (60 <= t <= 79)

Step 5: Initialize Buffers….

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

H0 = 0x67452301

H1 = 0xEFCDAB89

H2 = 0x98BADCFE

H3 = 0x10325476

H4 = 0xC3D2E1F0

Step 6: Processing Message in 512-bit blocks (L blocks in totalmessage)….

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

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

H0, H1, H2, H3, H4, H5: 5 Word buffers with initial values

Step 7: Pseudo Code….

- For loop on k = 1 to L

(W(0),W(1),...,W(15)) = M[k] */* Divide M[k] into 16 words */* For t = 16 to 79 do:

- W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)) <<< 1 A = H0, B = H1, C

= H2, D = H3, E = H4

For t = 0 to 79 do:

- TEMP = A<<<5 + f(t;B,C,D) + E + W(t) + K(t) E = D, D = C,

C = B<<<30, B = A, A = TEMP End of for loop

H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E End of for loop

Step 8: Output:

H0, H1, H2, H3, H4, H5: Word buffers with final message digest

Program:

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
public class Sha1
 {
public static void main(String[] args) throws NoSuchAlgorithmException
{
Scanner s=new Scanner(System.in);
System.out.println("\nEnter the Input String: ");
String str=s.nextLine();
System.out.println("\nMessage Digest(40 bits long): "+sha1(str));
}
static String sha1(String input) throws NoSuchAlgorithmException
{
MessageDigest mDigest = MessageDigest.getInstance("SHA1");
byte[] result = mDigest.digest(input.getBytes());
StringBuilder sb = new StringBuilder();
for(int i = 0; i < result.length; i++)
```

```
{
sb.append(Integer.toString((result[i] & 0xff) + 0x100, 16).substring(1));
}
return sb.toString();
}
}
```

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the java program to implement Secure Hash Algorithm (SHA – 1) was executed and the output was verified.

EX.NO:03	IMPLEMENTING SIGNATURE SCHEME – DIGITAL SIGNATURE STANDARD (DSS)
DATE:	

Aim:

To write a program to implement the signature scheme using Digital Signature Standard (DSS).

Algorithm:

Step 1- Start the program.

Step 2- Open the NetBeans IDE and create a java program.

Step 3- Compile and execute the program.

Step 4- Pass the signature to the function to validate the signature.

Step 5- Apply the digital signature scheme to verify and validate the given signature.

Step 6- Display the signature and print “true” if the given signature is valid or display “false” if the signature is not valid.

Step 7- Stop the program.

Program:

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.Signature;
import sun.misc.BASE64Encoder;
public class DigitalSignature
{
public static void main(String[] args) throws Exception
{
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
kpg.initialize(1024);
KeyPair keyPair = kpg.genKeyPair();
byte[] data = "test".getBytes("UTF8");
Signature sig = Signature.getInstance("SHAWithDSA");
sig.initSign(keyPair.getPrivate());
```

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

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the java program to implement signature scheme using Digital Signature Standard (DSS) was executed and the output was verified.

EX.NO:04	DIGITAL SIGNATURE USING GnuPG
DATE:	

Aim:

To demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures using GnuPG tool.

Procedure:

- Download Gpg4win from the Internet and install.
- The response to the question of whether you want to install the program is [*Yes*].

The installation assistant will start and ask you for the language to be used with the installation process:

- Confirm your language selection with [*OK*].

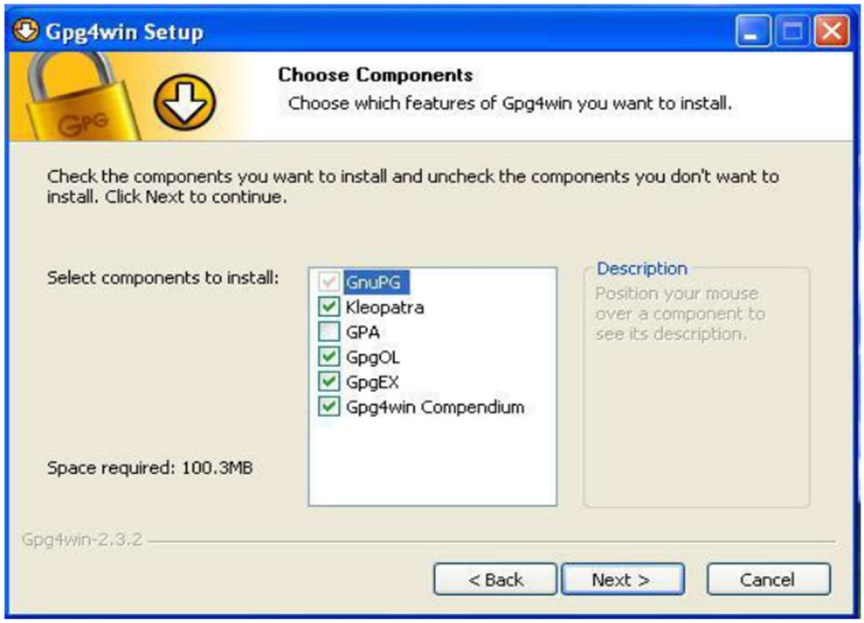
Afterwards you will see this welcome dialog:

- Close all programs that are running on your computer and click on [*Next*].

The next page displays the licensing agreement - it is only important if you wish to modify or forward Gpg4win. If you only want to use the software, you can do this right away - without reading the license.

Click on [*Next*].

- On the page that contains the selection of components you can decide which programs you want to install .A default selection has already been made for you. You can also install individual components at a later time .Moving your mouse cursor over a component will display a brief description. Another useful feature is the display of required hard drive space for all selected components.



Click on [*Next*].

· The system will suggest a folder for the installation,

e.g.: C:\Program Files\GNU\GnuPG

· You can accept the suggestion or select a different folder for installing Gpg4win.

Then click on [*Next*].

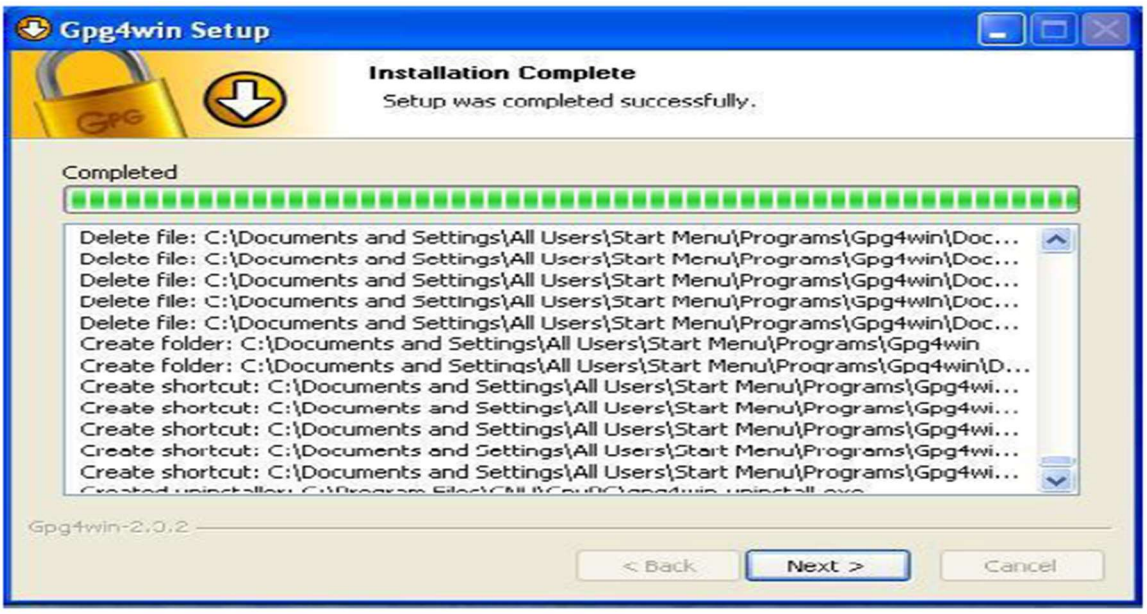
- Now you can decide which links should be insalled - the system will automatically create a link with the start menu.You can change this link later on using the Windows dashboard settings.

Then click on [*Next*].

- If you have selected the default setting - link with start menu - you can define the name of this start menu on the next page or simply accept the name.

Then click on [*Install*].

- During the installation process that follows, you will see a progress bar and information on which file is currently being installed. You can press [*Show details*] at any time to show the installation log.



Once you have completed the installation, please click on [*Next*].

- The last page of the installation process is shown once the installation has been successfully completed:

- You have the option of displaying the README file, which contains important information on the Gpg4win version you have just installed. If you do not wish to view this file, deactivate this option.

Then click on [*Finish*].

In some cases you may have to restart Windows. In this case, you will see the following page:

- Now you can decide whether Windows should be restarted immediately or

manually at a later time.

Click on [*Finish*].

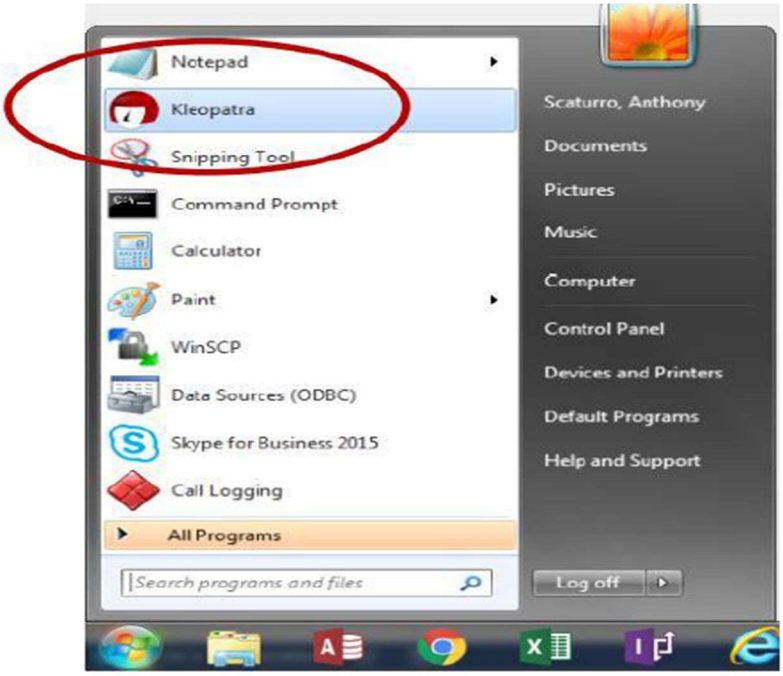
- Please read the README file which contains up-to-date information on the Gpg4win version that has just been installed. You can find this file e.g. via the start menu:

And that's it!

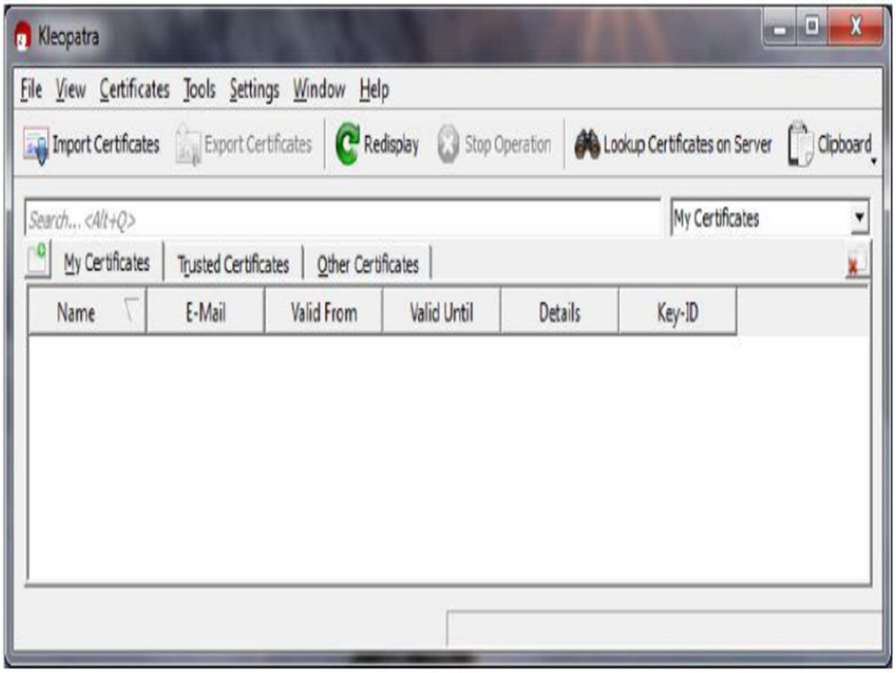
Start -> Programs -> Gpg4win -> Documentation -> Gpg4win README

You have successfully installed Gpg4win and are ready to work with the program.

Open Kleopatra using the Windows start menu:



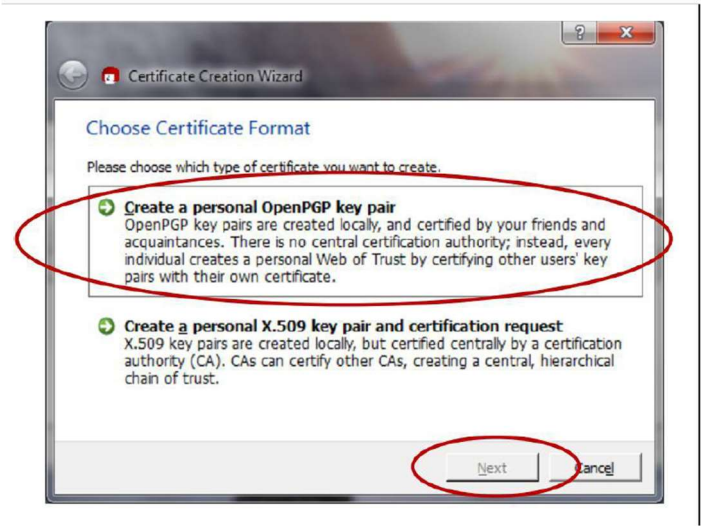
You will see the main Kleopatra screen - the certificate administration:



At the beginning, this overview will be empty, since you have not created or imported any certificates yet.

Click on *File -> New Certificate.*

In the following dialog you select the format for the certificate. You can choose from the following: OpenPGP (PGP/MIME) or X.509 (S/MIME).

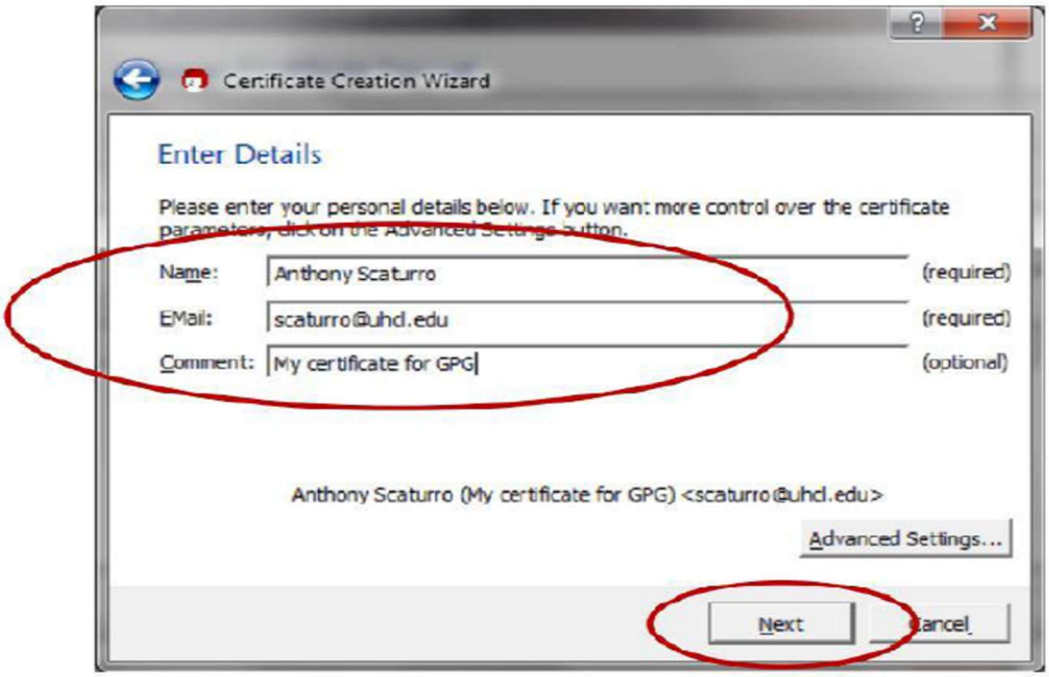


Creating an OpenPGP certificate:

In the certificate option dialog, click on [*Create personal OpenPGP key pair*].

Now enter your e-mail address and your name in the following window. Name and e-mail

address will be made publicly visible later. You also have the option of adding a comment for the key pair. Usually this field stays empty, but if you are creating a key for test purposes, you should enter "test" so you do not forget it is a test key. This comment becomes part of your login name, and will become public just like your name and e-mail address.



The Advanced settings are only be required in exceptional cases.

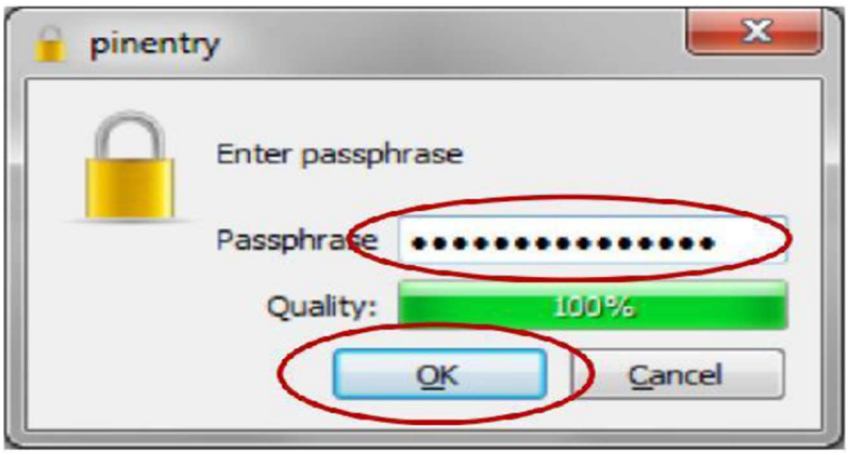
Click on [*Next*].

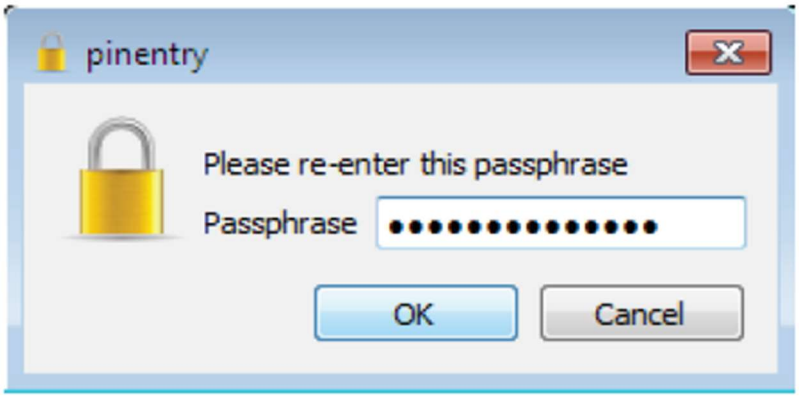
You will see a list of all of the main entries and settings for review purposes. If you are interested in the (default) expert settings, you can view these via the *All details* option.

If everything is correct, click on [*Create key*].

Now to the most important part: entering your passphrase!

To create a key pair, you must enter your personal passphrase:





To make sure that you did not make any typing errors, the system will prompt you to enter your passphrase twice. Always confirm your entry with [*OK*].

Now your OpenPGP key pair is being created:

As soon as the key pair creation has been successful, you will see the following dialog:

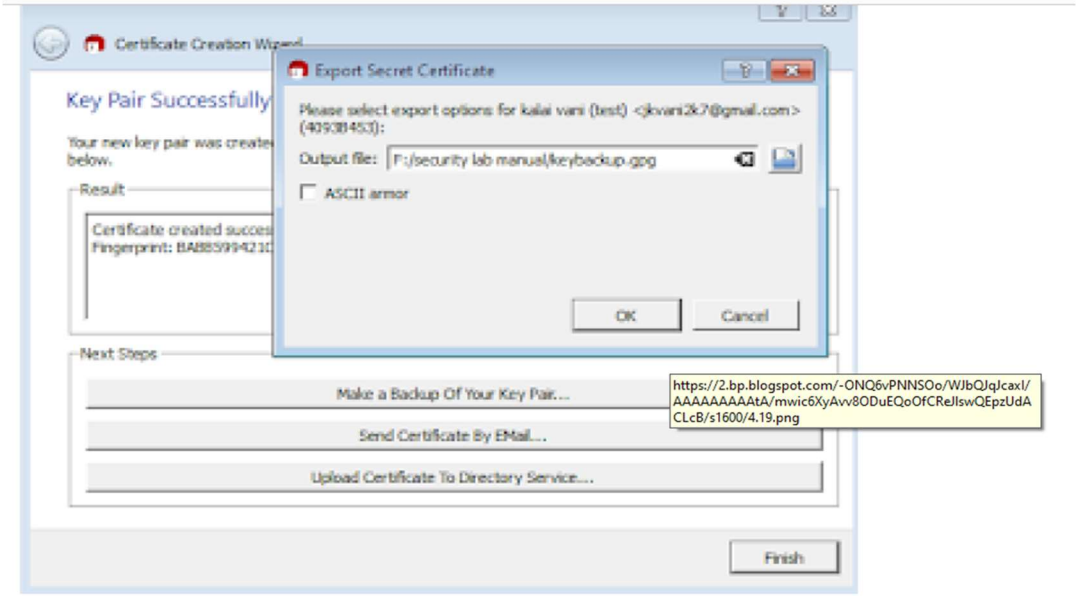
The 40-digit "fingerprint" of your newly generated OpenPGP certificate is displayed in the results text field. This fingerprint is unique anywhere in the world, i.e. no other person will have a certificate with the same fingerprint. Actually, even at 8 digits it would already be quite unlikely that the same sequence would occur twice anywhere in world. For this reason, it is often only the last 8 digits of a fingerprint which are used or shown, and which are described as the key ID. This fingerprint identifies the identity of the certificate as well as the fingerprint of a person.

However, you do not need to remember or write down the fingerprint. You can also display it later in Cleopatra's certificate details.

Next, you can activate one or more of the following three buttons:

Creating a backup copy of your (private) certificate...

Enter the path under which your full certificate (which contains your new keypair,hence the private *and* public key) should be exported:



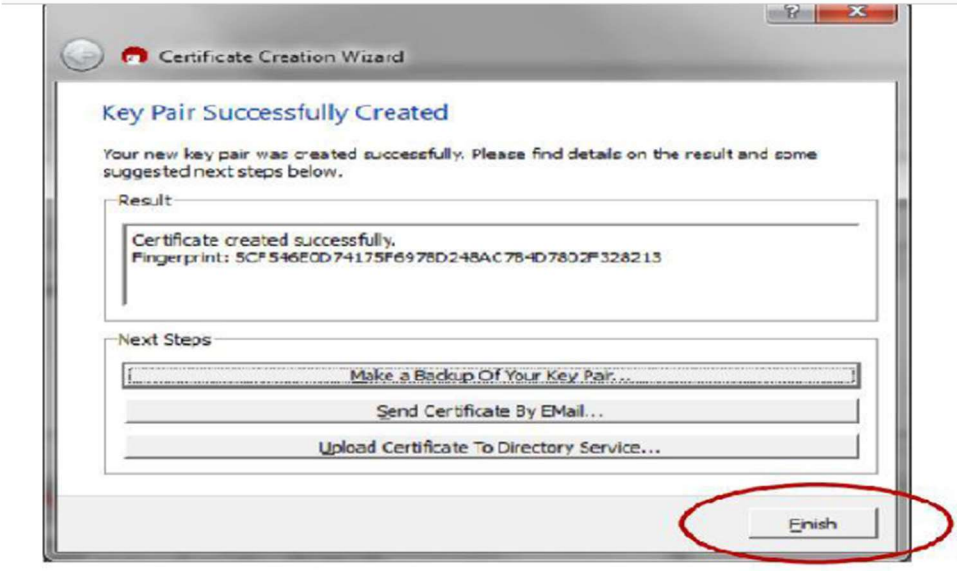
Sending a certificate via e-mail ...

Clicking on this button should create a new one e-mail - with your new public certificate in the attachment. Your secret Open PGP key will of course *not* be sent. Enter a recipient e-mail address; you can also add more text to the prepared text for this e-mail.

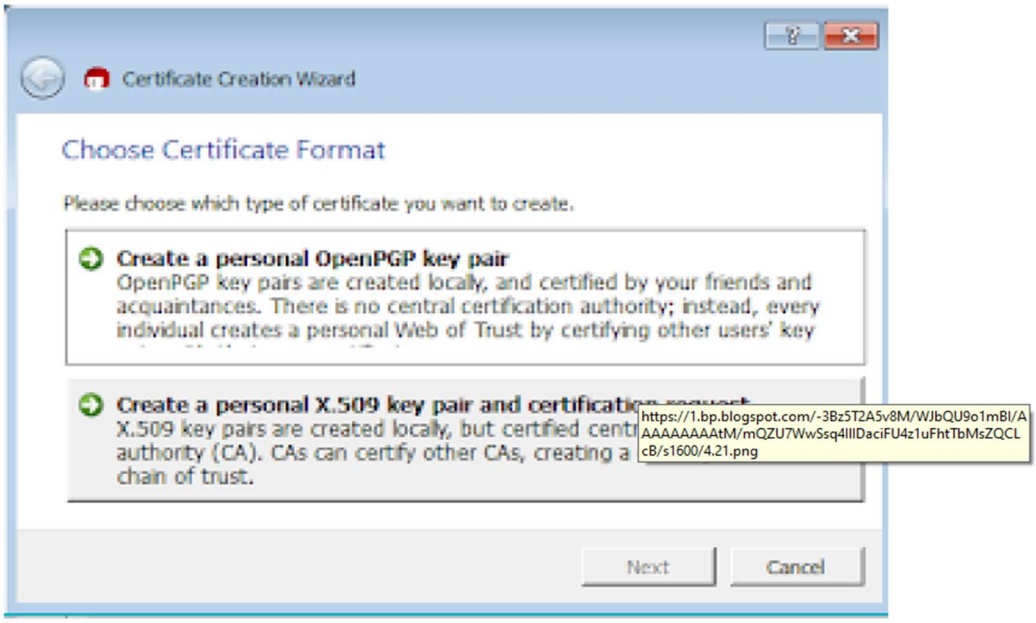
Please note:

Not all e-mail programs support this function. Of course you can also do this manually: If you do not see a new e-mail window, shut down the certificate creation assistant, save your public certificate via *File -> Export certificate* and sent this file via e-mail to the people you are corresponding with.

This completes the creation of your OpenPGP certificate. End the Cleopatra assistant with [*Finish*].



In the home page, click on the button Fileànew certificate and choose the option, [*Create personal X.509 key pair and authentication request*].

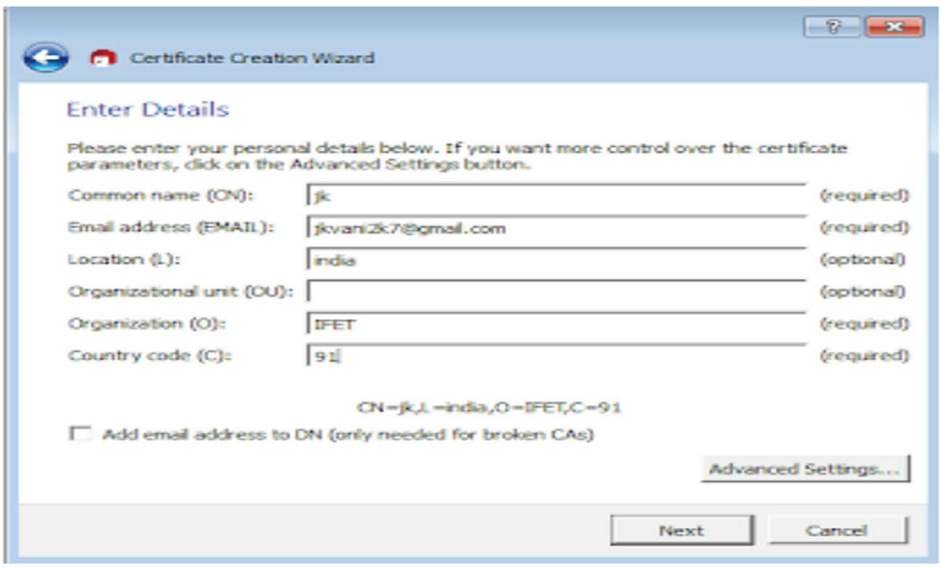


In the following window, enter your name (CN = common name), your e-mail address (EMAIL), organisation(O) and your country code (C). Optionally, you can also add your location (L = Locality) and department (OU =Organizational Unit).

The Advanced settings will only be required in exceptional cases. For details, see the Kleopatra handbook (via *Help -> Kleopatra handbook*).

Click on [*Next*].

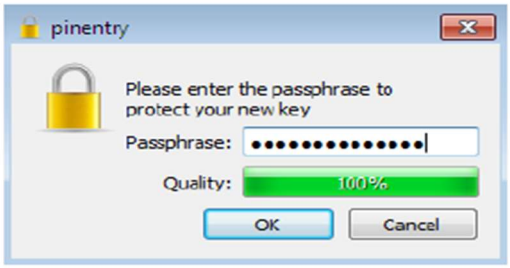
You will see a list of all main entries and settings for review purposes. If you are interested in the (default) expert settings, you can view these via the *All details* option.



Once everything is correct, click on [*Create key*].

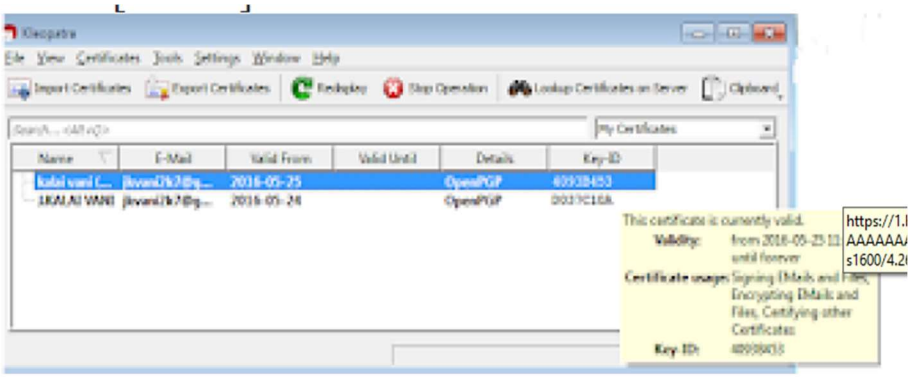
Now to the most important part: Entering your passphrase!

In order to create a key pair, you will be asked to enter your passphrase:



Now your X.509 key pair is being created: As soon as the key pair has been successfully created, you will see the following dialog:

End the Kleopatra assistant with [*Finish*].



Save request in file:

Here, you enter the path under which your X.509 certificate request should be backed up, and confirm your entry. Kleopatra will automatically add the file ending. p10 during the saving process. This file can then be sent to an authentication instance (in short CA for Certificate

Authority). Further below, we will refer you to cacert.org, which is a non-commercial authentication instance (CA) that issues X.509 certificates free of charge.

Sending a request by e-mail:

This creates a new e-mail with the certificate request which has just been created in the attachment. Enter a recipient e-mail address - usually that of your certificate authority in charge; you can also add more text to the prepared text of this e-mail.

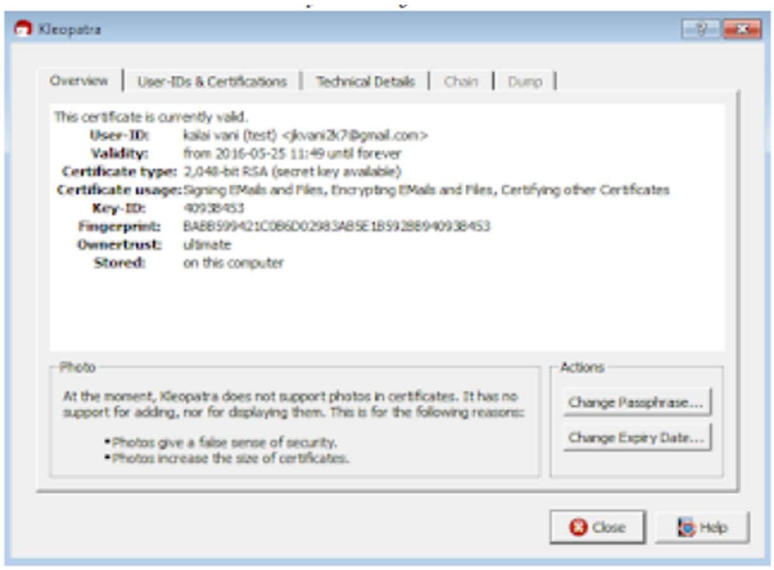
Not all e-mail programs support this function. Of course, you can also do this manually: If you do not see a new e-mail window, save your request in a file (see above) and send it by e-mail to your certificate authority (CA).

Certificate creation process completion:

This completes the creation of your OpenPGP or X.509 key pair. You now have a unique electronic key.

During the course of this compendium, we will always use an OpenPGP certificate for sample purposes - however, all information will also apply accordingly to X509 certificates.

You are now back in the Kleopatra main window. The OpenPGP certificate which was just created can be in the certificate administration under the tab *My certificates*:



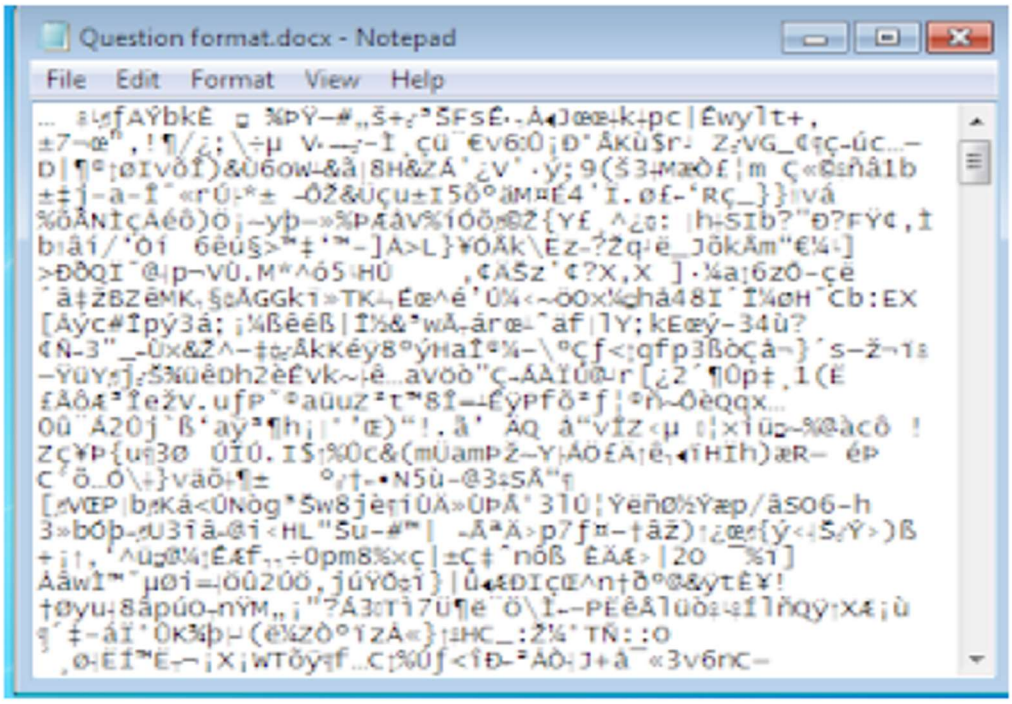
Exporting your public OpenPGP certificate

Select the public certificate to be exported in Kleopatra (by clicking on the corresponding line in the list of certificates) and then click on

File -> Export certificates... in the menu.

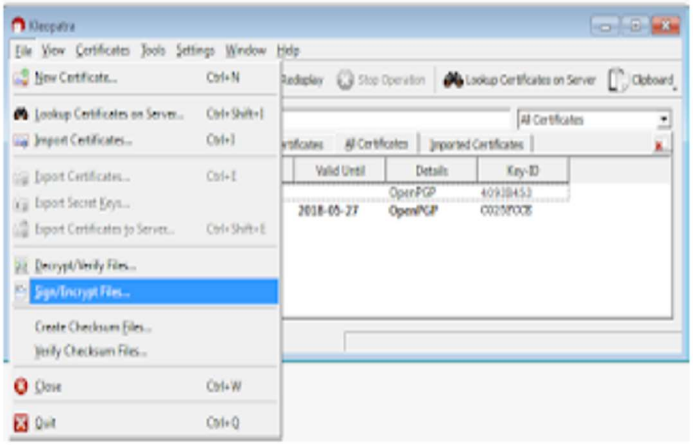
Select a suitable file folder on your PC and save the public certificate with the file type.asc e.g.: main- OpenPGP-Zertifikat.asc. The other file types, which can be selected, .gpg or.pgp,

will save your certificate in binary format. That means that in contrast to an .as file, they cannot be read in the text editor.

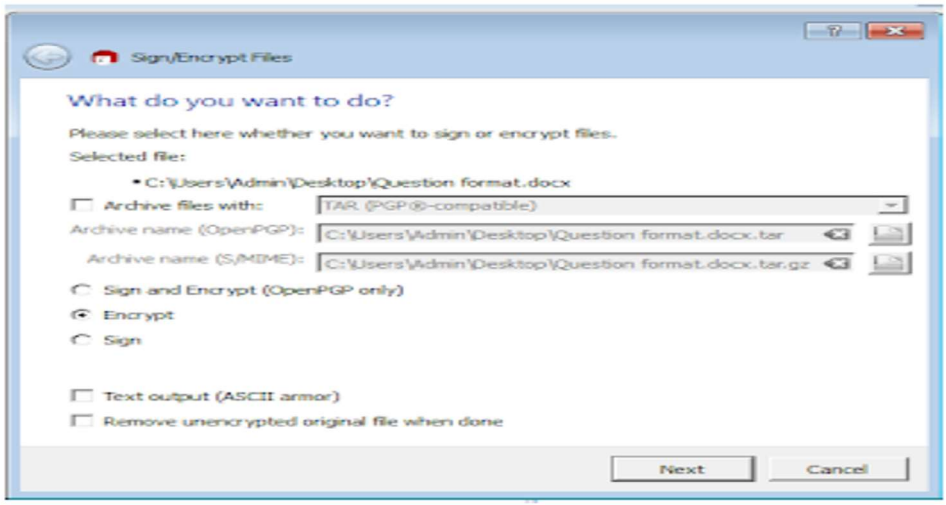


For encrypting files with the certificate created,

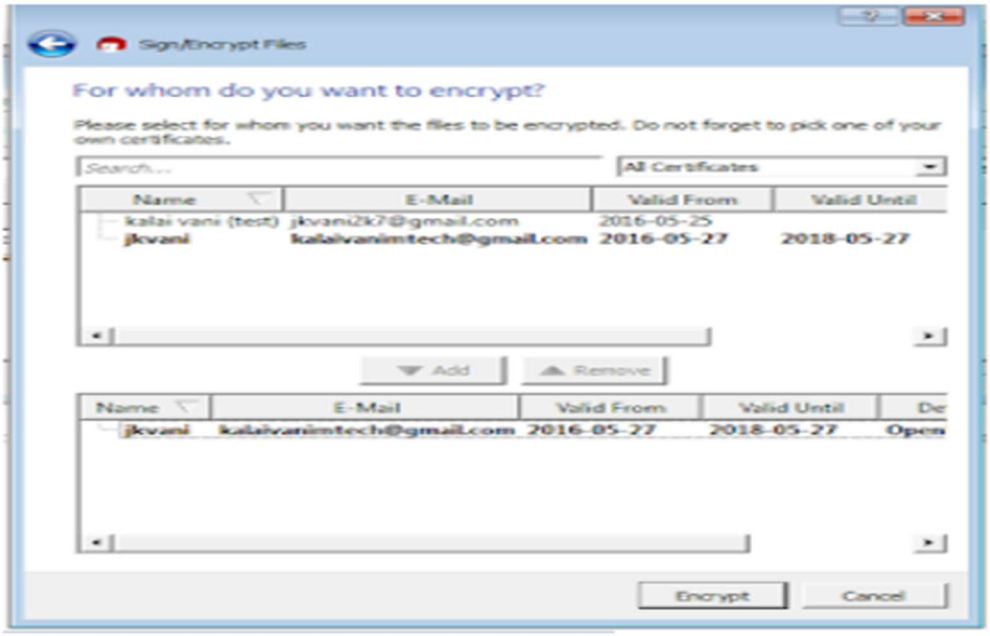
Choose -> file -> sign/encrypt files and upload the file to be encrypted.



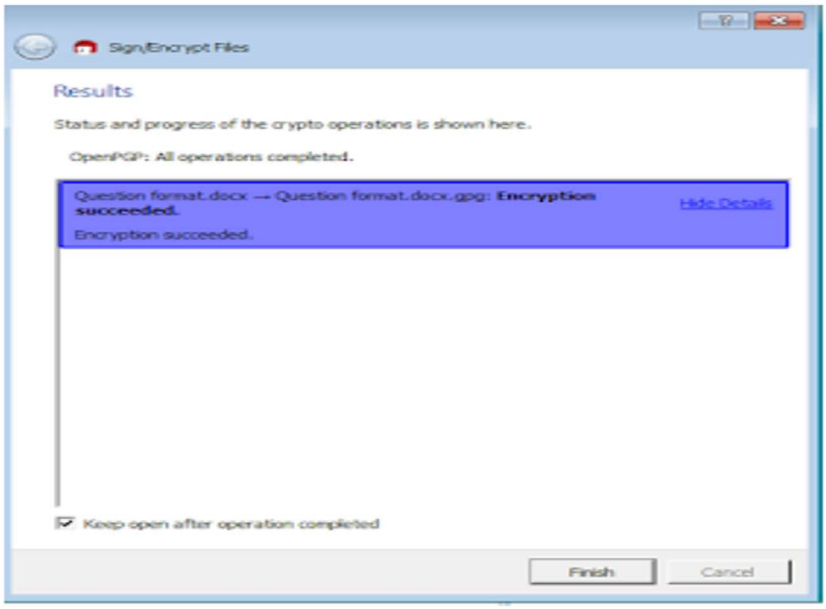
Choose the option encrypt or sign and encrypt from the options and give next.



Select the recipient for whom you want the encrypted files to be sent and click the button add, then click encrypt.

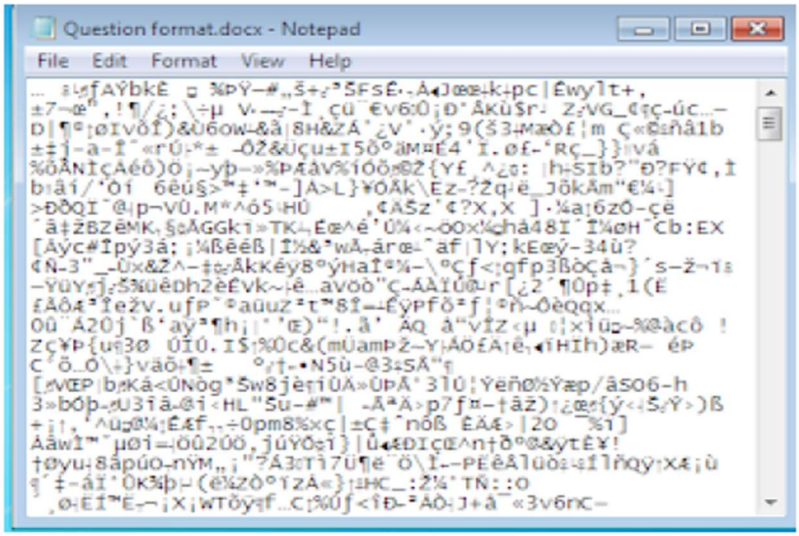


The file chosen for encryption will be encrypted and stored in the same location with the extension. gpg, and will be sent to the recipient.



Click finish

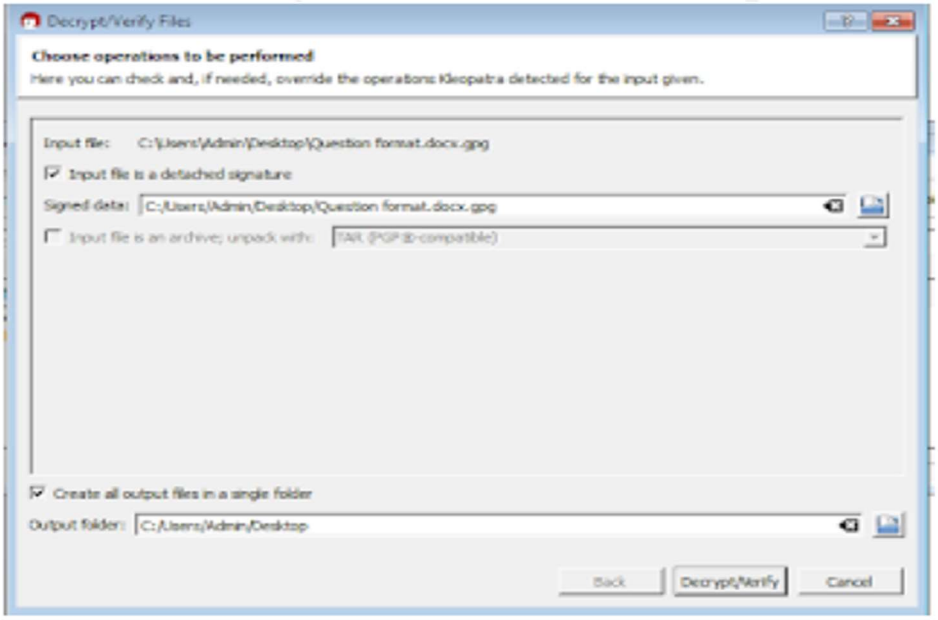
Encrypted file content will be like,



For decryption,

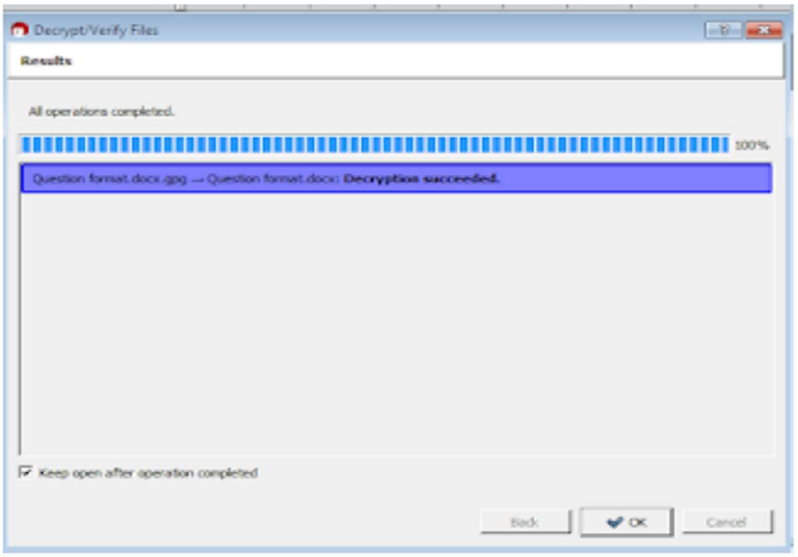
The receiver should have the sender's public key received already through e-mail.

Choose -> file-> decrypt/verify files and choose your certificate, then upload the file to be decrypted



Enter the passphrase once again for authentication..

The file will be decrypted and will be saved in the same location



Click ok.

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus, the demonstration for providing secure data storage, secure data transmission and for creating digital signatures using GnuPG tool was executed and verified successfully.

EX.NO:05	MONITOR THE HONEYPOT ON NETWORK USING KF SENSOR
DATE:	

Aim:

To setup and monitor the Honeypot on network using KF Sensor.

Description:

Honeypot is a computer security mechanism set to detect, deflect, or, in some manner, counteract attempts at unauthorized use of information systems.

KF Sensor is the tool to setup as honeypot. When KF Sensor is running it places a warning icon in the windows system tray in the bottom right of the screen. If there are no alerts then green icon is displayed.

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.

Step 4. The KF Sensor automatically starts during windows boot Click Next to setup wizard.

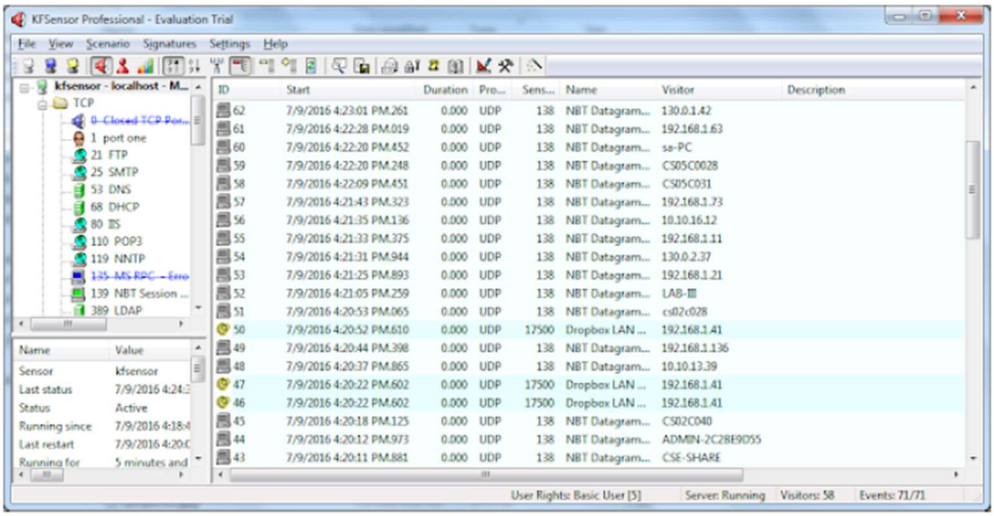
Step 5. Select all port classes to include and Click Next.

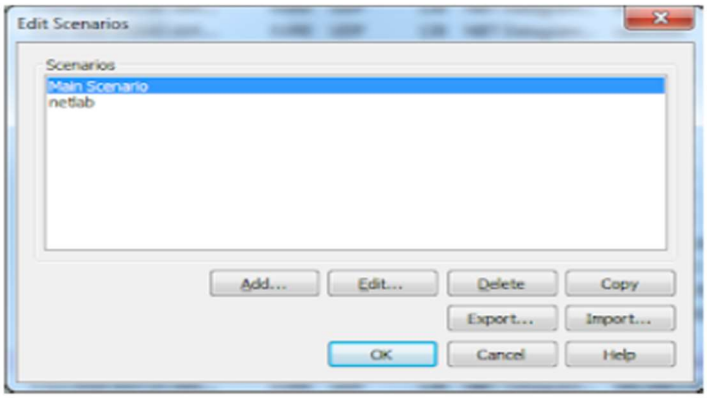
Step 6. Enter the Send to and Send from email ID and Click Next. Select the options such as Denial of Service [DOS],

Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next. Select Install as System service and Click

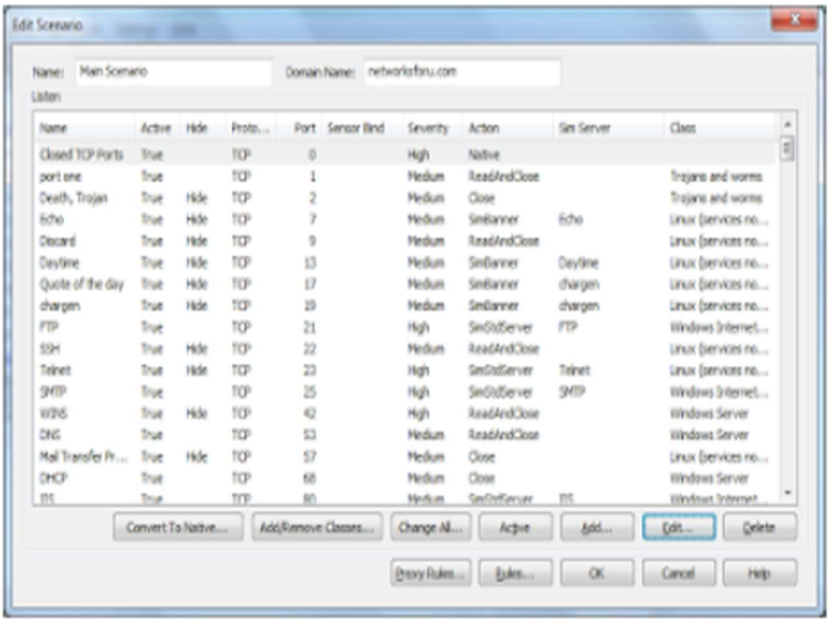
Next. Click finish.

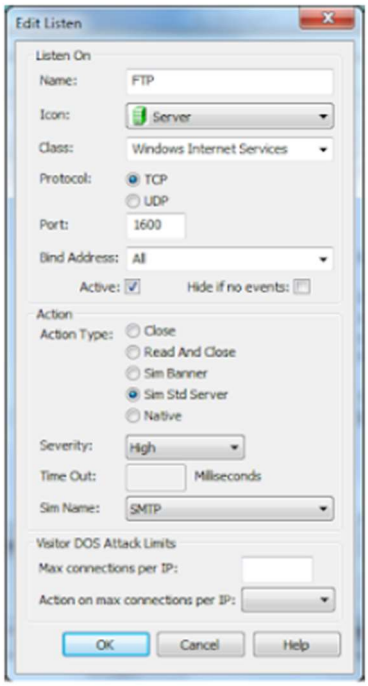
Step 7. Select Scenario Edit Scenario and double click Main Scenario.





Step 8. Click Edit tab and edit the following window.





9. The above port will be monitored by the KF.

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the demonstration for setting up and monitoring the Honeypot on the network using KF Sensor tool was executed and verified successfully.

EX.NO:06	INSTALLATION OF ROOTKITS AND STUDY ABOUT THE VARIETY OF OPTIONS
DATE:	

Aim:

To install rootkits and study about the variety of options.

Description:

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.

Rootkits

Rootkit is a malicious software program, used to gain elevated access to a computer while it remains hidden from the owner of the computer and installed security software. Rootkits typically run at a low level and load before the computer's operating system to remain hidden. The rootkit can then divert any OS functions that would reveal its presence and display manipulated results to the user.

Malicious users or software often install a rootkit once they have gained access to a computer, through vulnerabilities in the computer's software or through gaining the password by social engineering, for example. The rootkit allows them continued access to the computer, but it leaves no trace of their activity, as it would if they were logged in through a normal user account. Once installed, the rootkit owner can access the computer at any time to run software, or to control the computer remotely.

Why Root Kits Are Used

- Root kits are used by criminals for a variety of purposes, usually to turn a computer into part of a botnet, which can then, in turn, go on to infect other computers or send spam email messages.
- The rootkit owner can install keyloggers to capture user-entered passwords for online banking and similar activities, or steal the user’s personal details to use for identity fraud. If the rootkit owner uses the computer for criminal acts, such as breaking into other computers, it will appear as if the computer owner is responsible if authorities trace the connection.

How Root Kits Stay Undetected

- Many root kits infect the boot sectors of the computer’s hard disk, allowing them to

load before the computers operating system.

- The rootkit then patches the operating system and changes common functions to hide its existence. For example, the root kit could intercept calls for a list of files in a directory, removing its own file names before showing the results to the user, so it would appear as if the directory is clean. Both anti-virus and security software programs are vulnerable to the effects of a root kit, which runs at a lower level, ensuring the anti-virus software cannot detect or remove it. This leads the anti-virus software into believing the system is clean, when it is actually infected and running malicious software.

Current Rootkit Capabilities:

Root kits Hide processes, Hide files, Hide registry entries, Hide services, Completely bypass personal firewalls, Undetectable by antivirus, Remotely undetectable, Covert channels - undetectable on the network, Defeat cryptographic hash checking, Install silently, All capabilities ever used by viruses or worms

GMER is an application that detects and removes rootkits .

It scans for:

- hidden processes
- hidden threads
- hidden modules
- hidden services
- hidden files
- hidden disk sectors (MBR)
- hidden Alternate Data Streams
- hidden registry keys
- drivers hooking SSDT
- drivers hooking IDT
- drivers hooking IRP calls
- inline hooks

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.

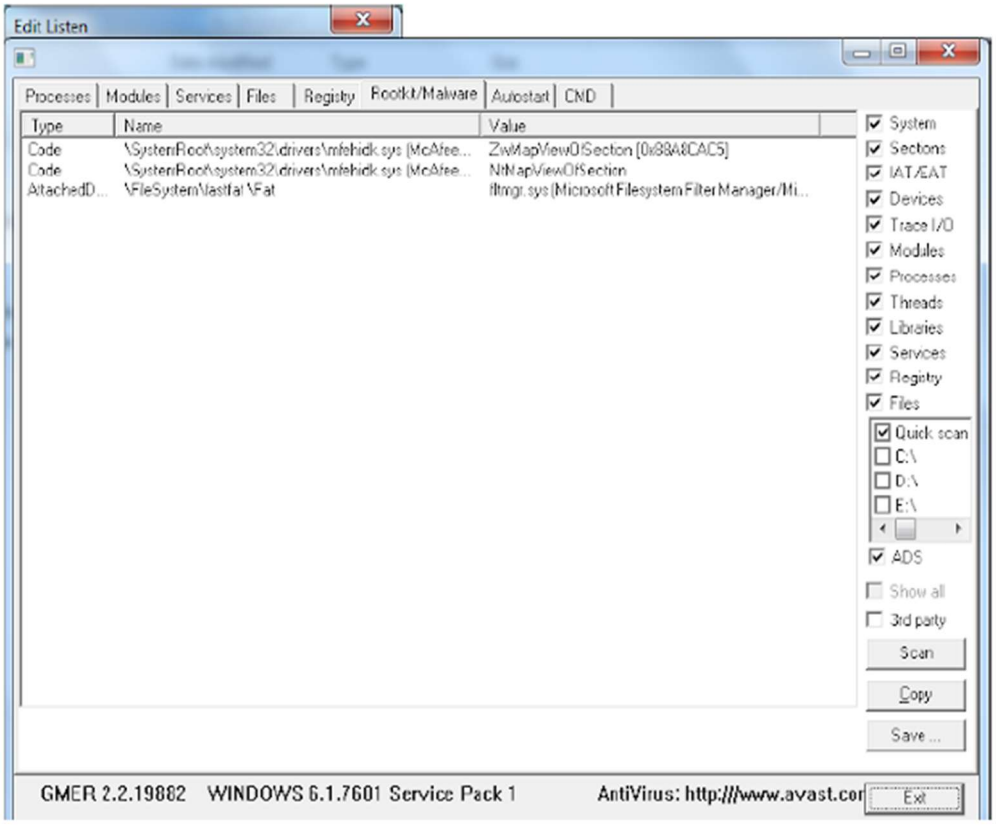
Step 6. Files menu displays full files on Hard-Disk volumes.

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



Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the installation of rootkits and study about the variety of options using rootkit tool was executed and verified successfully.

EX.NO:07	PERFORM WIRELESS AUDIT ON AN ACCESS POINT OR A ROUTER AND DECRYPT WEP AND WPA USING NETSTUMBLER
DATE:	

Aim:

To perform wireless audit on an access point or a router and decrypt WEP and WPA using NetStumbler.

Description:

NetStumbler is one of the best known and most widely used Wireless Hacking Tools which operates on Windows. This tool is able to list Wireless Access Points and display their basic details such as: SSID, channel, speed, MAC address, vendor, and level of encryption. Unlike other tools in this category, this tool also lists the signal, noise, and signal-to-noise ratio (SNR) levels. Additionally, it has GPS support to record access point locations when wardriving.

Procedure:

Download and install Netstumbler

Step 1· It is highly recommended that your PC should have wireless network card in order to access wireless router.

Step 2· Now Run Netstumbler in record mode and configure wireless card.

Step 3· 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 4· Lock symbol with GREEN bubble indicates the Access point has encryption enabled.

Step 5· MAC assigned to Wireless Access Point is displayed on right hand pane.

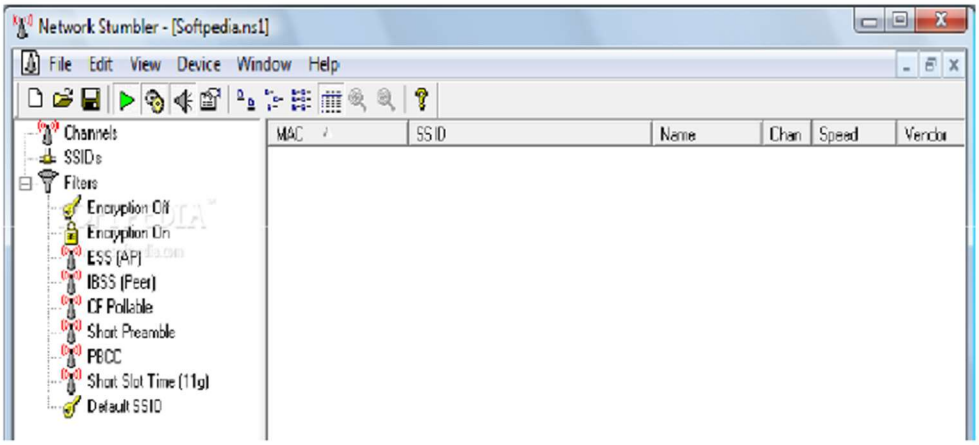
Step 6· The next column displays the Access points Service Set Identifier[SSID] which is useful to crack the password.

Step 7· To decrypt use WireShark tool by selecting EditàpreferencesàIEEE 802.11

Step 8· Enter the WEP keys as a string of hexadecimal numbers as A1B2C3D4E5

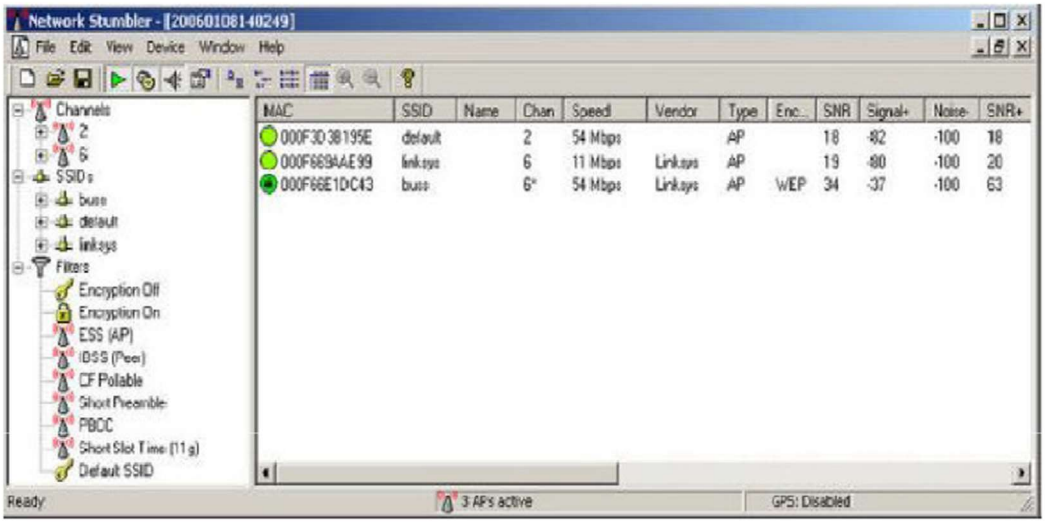
OUTPUT

Home page:



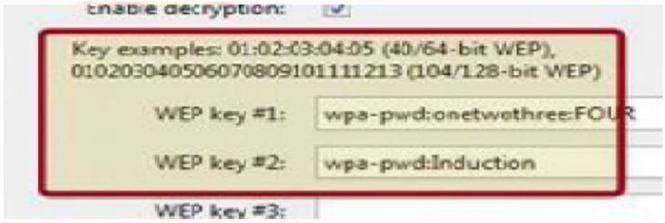
Click on the channels icon. It will display the below window showing the MAC, SSID and other details.

Open the wireshark tool and choose the profile:

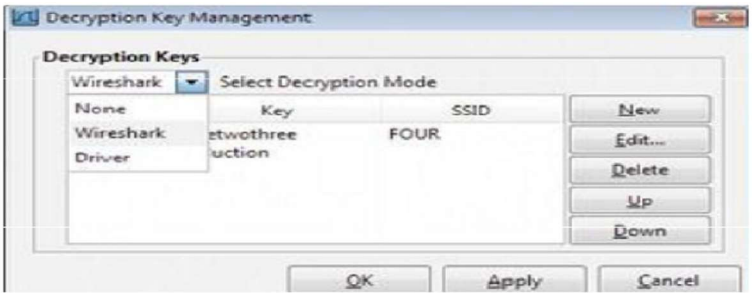


Adding Keys: Wireless Toolbar

If you are using the Windows version of Wireshark and you have an AirPcap adapter you 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:



Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the demonstration for wireless audit on an access point or a router and decrypt WEP and WPA using NetStumbler was executed and verified successfully.

EX.NO:08	INTRUSION DETECTION SYSTEM (IDS) USING SNORT
DATE:	

Aim:

To demonstrate Intrusion Detection System (IDS) using Snort.

Description:

Overview of IDS

With the development of network technologies and applications, network attacks are greatly increasing both in number and severity. As a key technique in network security domain, Intrusion Detection System (IDS) plays vital role of detecting various kinds of attacks and secures the networks. Main purpose of IDS is to find out intrusions among normal audit data and this can be considered as classification problem. Intrusion detection systems (IDS) are an effective security technology, which can detect, prevent and possibly react to the attack. It performs monitoring of target sources of activities, such as audit and network traffic data in computer or network systems, requiring security measures, and employs various techniques for providing security services. With the tremendous growth of network based services and sensitive information on networks, network security is becoming more and more important than ever before.

Intrusion : Attempting to break into or misuse your system. Intruders may be from outside the network or legitimate users of the network. Intrusion can be a physical, system or remote intrusion. Intrusion Detection Systems look for attack signatures, which are specific patterns that usually indicate malicious or suspicious intent.

Snort: Snort is an open source network intrusion prevention system, capable of performing real time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS Fingerprinting attempts, and much more.

Snort has three primary uses: It can be used as a straight packet sniffer like tcp dump, a packet logger (useful for network traffic debugging, etc), or as a full blown network intrusion prevention system.

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

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.

Network Intrusion Detection System mode: snort –d c:\log –h ip address/24 –c nort.conf. This is a configuration file applies rule to each packet to decideit an action based upon the rule type in the file.

Procedure:

Step 1. Go to the web site www.snort.org/start/download

Step 2. Click on download option and support path to save the setup file.

Step 3. Double click on Snort Installation icon to run setup.

Step 4. Accept License agreement and Specify path for installation, then Click on Next.

Step 5. Install snort with or without database support.

Step 6. Skip the WinPcap driver installation

Step 7. Select all the components and Click Next.

Step 8. Install and Close.

Step 9. Add the path variable in windows environment variable by selecting new class path.

Step 10. Create a path variable and point it at snort.exe variable name : path and variable value as c:\snort\bin.

Step 11. Click OK button and then close all dialog boxes.

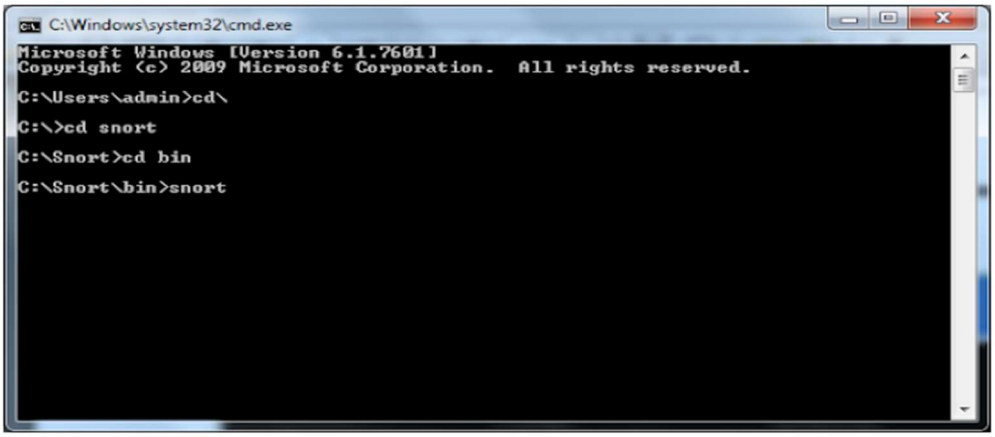
Step 12. Type the following commands:

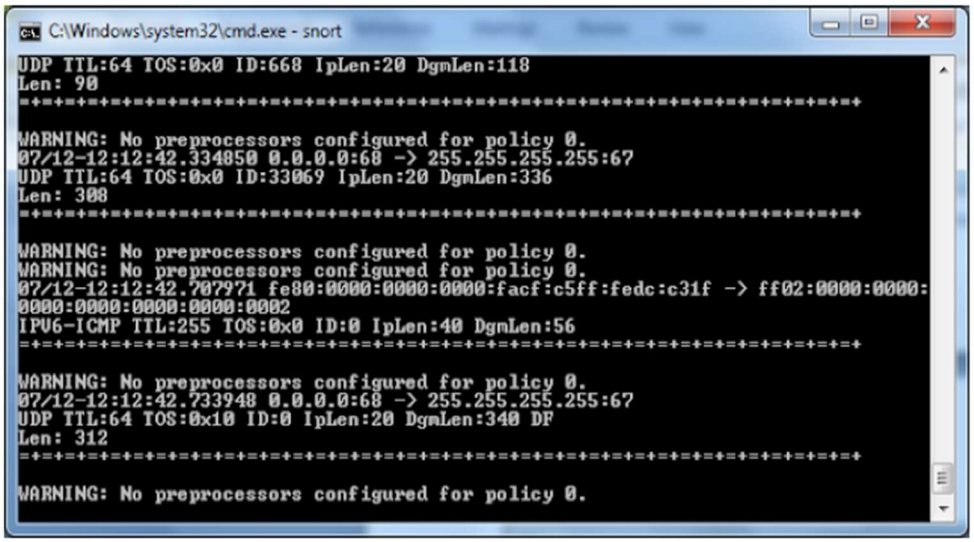
a. Go to command prompt and get into Snort/bin directory and run Snort.exe file.

b. An editor window displays the complete details of packets flowing across the system,the IP Address of packet generator, date &Time, length of Packet, Time to live(TTL)etc at Realtime.

Step 15. By analyzing these details Intruders can be traced at real time.

Step 16. These details can be documents by using a print screen option.





Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus the demonstration of Intrusion Detection System (IDS) using Snort tool was executed and verified successfully.

EX.NO:09	CASE STUDIES: ETHICAL HACKING
DATE:	

Case study 1: Defamatory websites, online attacks, tortious interference

Case Scenario: This client runs a large scale internet marketing business where ALL of their clients research their name on Google. Three individuals decided to establish websites defaming the client. Furthermore, a former ex, high level employee was thought to be spearheading the entire campaign to derail the business in attempt to grow their own, competing business.

Through a variety of methods, CIS Investigators were able to determine the identity and location of the three website owners. CIS then recommended an attorney for this case that filed 3 separate lawsuits. One of the defendants proceeded to claim that they were not responsible and CIS was then able to provide all the necessary counter evidence. The attorney was then able to reach a satisfactory arrangement with each of the 3 website posters.

On the larger case of the ex-employee intentionally derailing the business, CIS investigators assisted attorneys in providing the necessary evidence to ultimately obtain a $3.6 Million judgment.

Case study 2: Tortious Interference and Defamation

Case Scenario: The client was notified by several of their co-insurers that they were receiving emails slandering the client and making claims that they were not operating legitimately. The information had recently cost the insurance company a very large contract worth over a million dollars.

CIS Investigators were able to obtain the original emails and attachments and through email tracing and forensic techniques were able to identify the sender of the email, which turned out to be someone they were in current litigation with. The suspect was trying to get off of his civil case by making the client look bad, and by damaging their reputation globally.

Case study 3: Stalking and extortion

Case Scenario: Client was a key executive for a major retailer and had been contacted by an unknown person that stated he had nude pictures of her that he said he had

obtained by her ex-husband. The suspect then proceeded to provide proof of having possession of those pictures. The suspect began calling the client and her husband at work and on their cell phones threatening them with exposing the pictures. The suspect the escalated the threats to demanding that the client have sex with the suspect while her husband watched at a downtown high rise hotel and they had to bring $10,000 dollars.
The clients contacted CIS to determine who the individual was. Within a day and a half CIS Investigators were able to locate the anonymous suspect, capture his location and IP address and set up a sting operation with the local sheriff's office. The suspect showed up to the hotel and the sheriff's office arrested the individual for stalking and extortion.

Case study 4: International auction websites

Case Scenario: An international auction website came under attack from an unknown source that was stopping their auctions from being able to complete transactions costing them thousands of dollars per day. CIS investigated the case determined it was an inside job, gathered the forensic evidence and turned the case over to the FBI Cyber Unit.

Case study 5: Personal hacking incident

Case Scenario: A client had been hacked by an unknown individual and harassed by the hacker for a period of over a year. CIS investigated the case found the forensic evidence and IP information of the Hackers. We then turned over our evidence to the local police and the clients attorney.

Observation	25	
Record	10	
Total	35	
Signature		

Result:

Thus,The give case studies for ethical hacking has been studied successfully.