

**VELALAR COLLEGE OF ENGINEERING AND TECHNOLOGY
THINDAL (POST), ERODE -638 012, TAMILNADU
(AUTONOMOUS)**

(Approved by AICTE, Affiliated to Anna University, Accredited by NAAC with Grade “A”)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(Accredited by NBA)**

REGULATION 2018

Lab Manual

18CSL71- NETWORK SECURITY LABORATORY



Prepared by

Approved by

INDEX

S.No	Particulars	Page No.
1	Dos and Don't	3
2	Vision, Mission, PEOs, POs/ PSOs	4
3	Syllabus	7
List of Experiments		
Exp. No	Name of the Program	Page No.
1(A)	Caesar Cipher	8
1(B)	Playfair Cipher	11
1(C)	Hill Cipher	19
2(A)	Data Encryption Standard(DES)	22
2(B)	RSA Algorithm	26
2(C)	Diffiee-Hellman Algorithm	30
2(D)	MD5	34
2(E)	SHA-1	36
3	Implement the Signature Scheme forDigital Signature Standard	40
4	Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures (GnuPG)	43
5	Setup a honey pot and monitor the honey pot on network (KF Sensor)	63
6	Installation of rootkits and study aboutthe variety of options	66
7	Perform wireless audit on an access point or a router and decrypt WEP and WPA (Net Stumbler)	71
8	Demonstrate intrusion detectionsystem (ids) using snort tool	75

DOs AND DON'Ts IN THE LABORATORY

DO's

- Must maintain silence
- All students should enter the lab with formal dress
- All students must carry observation book to the lab
- While leaving the lab, shutdown the system and arrange your chairs properly
- Keep your voice low when speaking to others in lab
- Do ask the staff for assistance if you need help
- Use computer as much as you need for academic purpose
- Report system malfunctioning to appropriate staff

DONT's

- Don't misuse the systems
- Don't use Network Resources for playing games, chat, etc.,
- Don't eat or drink while working in computer
- Don't attempt to bypass the computer security system
- Don't delete, examine, copy or modify others files
- Don't open an external device without scanning them for Computer viruses
- Don't install any programs without getting permission from the lab in-charge
- Don't attempt to repair, open, tamper or interfere with, any of the equipment in the laboratory
- Avoid stepping on electrical wires or any other computer cables.

Vision of the Institute

To provide a wide range of Academic and Research Programmes and strive to develop a Centre of Excellence for Learning.

Mission of the Institute

M1: To impart essential knowledge to the students through quality education, training and research activities.

M2: To inculcate the students with Societal Awareness, Professional Ethics and Leadership Qualities.

M3: To mould the students as high quality Engineers, Technocrats, Scientists, Administrators and Entrepreneurs.

Vision of the Department

To produce competent professionals for a globalized technological society by preparing them to become part of the industry and research organizations in the field of science and technology and orient them towards serving the society.

Mission of the Department

- Strengthening the core competence in Computer Science and Engineering through analytical learning and Industry – Institute interaction.
- Providing exposure to latest tools and technologies in the core area to design solutions and conduct investigations.
- Promoting innovative research based projects / activities in the emerging areas of technology convergence.
- Inculcating professional behavior, strong ethical values, leadership abilities, entrepreneurship skills and commitment to lifelong learning.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates after few years of graduation will:

1. **Preparation:** Develop professional skills for a successful professional career as an engineer, scientist, technocrat, administrator or an entrepreneur.
2. **Core Competence:** Design and implement practical systems consisting of software and/or hardware components through the key principles and practices of computation, mathematics and basic principles of engineering.
3. **Multidisciplinary:** Design and apply new ideas and technologies as the field evolves, to solve real world problems in related interdisciplinary areas.

4. **Professional Environment:** Demonstrate professional attitude and ethics, effective communication, team work and managerial skills for societal, environmental and global context.
5. **Learning Environment:** Engage in continuing education and training and acquire professional competence through lifelong learning.

PROGRAM OUTCOMES

Graduates will be able to

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 analyse 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 for complex problems.

PO5: **Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling 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: **Environment 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 in 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 lifelong learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO1. Apply a variety of programming skills to solve business problems

PSO2. Apply design principles in the construction of computing systems and expert systems of varying complexity.

18CSL71 NETWORK SECURITY LABORATORY L T P C 0 0 3 1

(Common to B.E/B.Tech.- CS& IT Programmes)

Preamble: This laboratory course is intended to expose the students to different cipher techniques and to implement algorithms like DES, RSA, MD5, SHA-1 using Java programming language/Python programming language.

Course Outcomes: Upon completion of the course, students will be able to:

1. Implement and test symmetric and asymmetric cipher techniques using Java/Python.
2. Implement digital signature standard and verify the authentication process using Java/Python.
3. Implement and test digital signature standard using GnuPG.
4. Implement and test honey pot using KF Sensor.
5. Implement and test IDS and WEP, WPA using snort and Net stumbler respectively.

LIST OF EXPERIMENTS

1. Implement the following Substitution & Transposition Techniques
 - a) Caesar Cipher
 - b) Playfair Cipher
 - c) Hill Cipher
2. Implement the following algorithms
 - a) DES
 - b) RSA Algorithm
 - c) Diffie-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)

EX. NO: 1(A)

IMPLEMENTATION OF CAESAR CIPHER

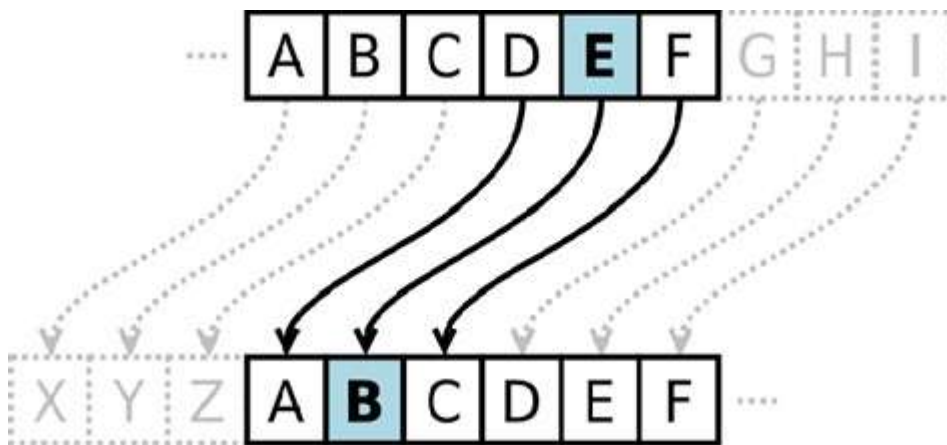
AIM:

To implement the simple substitution technique named Caesar cipher using java 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)

//A Java Program to illustrate Caesar Cipher Technique
class CaesarCipher

```
{
    // Encrypts text using a shift of s
    public static StringBuffer encrypt(String text, int s)
    {
        StringBuffer result= new StringBuffer();

        for (int i=0; i<text.length(); i++)
        {
            if (Character.isUpperCase(text.charAt(i)))
            {
                char ch = (char)(((int)text.charAt(i) +
                    s - 65) % 26 + 65);
                result.append(ch);
            }
            else
            {
                char ch = (char)(((int)text.charAt(i) +
                    s - 97) % 26 + 97);
                result.append(ch);
            }
        }
        return result;
    }

    // Driver code
    public static void main(String[] args)
    {
        String text = "ATTACKATONCE";
        int s = 4;
        System.out.println("Text : " + text);
        System.out.println("Shift : " + s);
        System.out.println("Cipher: " + encrypt(text, s));
    }
}
```

OUTPUT:

Text : NARUTO

Shift : 4

Cipher: REVYXS

RESULT:

Thus the implementation of Caesar cipher had been executed successfully.

EX. NO: 1(B)

IMPLEMENTATION OF PLAYFAIR CIPHER

AIM:

To write a java 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)

// Java Program to Encode a Message Using Playfair Cipher

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

class Playfair {
    String key;
    String plainText;
    char[][] matrix = new char[5][5];

    public Playfair(String key, String plainText)
    {
        // convert all the characters to lowercase
        this.key = key.toLowerCase();

        this.plainText = plainText.toLowerCase();
    }

    // function to remove duplicate characters from the key
    public void cleanPlayFairKey()
    {
        LinkedHashSet<Character> set
            = new LinkedHashSet<Character>();

        String newKey = "";

        for (int i = 0; i < key.length(); i++)
            set.add(key.charAt(i));

        Iterator<Character> it = set.iterator();

        while (it.hasNext())
            newKey += (Character)it.next();
    }
}
```

```

    key = newKey;
}

// function to generate playfair cipher key table
public void generateCipherKey()
{
    Set<Character> set = new HashSet<Character>();

    for (int i = 0; i < key.length(); i++)
    {
        if (key.charAt(i) == 'j')
            continue;
        set.add(key.charAt(i));
    }

    // remove repeated characters from the cipher key
    String tempKey = new String(key);

    for (int i = 0; i < 26; i++)
    {
        char ch = (char)(i + 97);
        if (ch == 'j')
            continue;

        if (!set.contains(ch))
            tempKey += ch;
    }

    // create cipher key table
    for (int i = 0, idx = 0; i < 5; i++)
        for (int j = 0; j < 5; j++)
            matrix[i][j] = tempKey.charAt(idx++);

    System.out.println("Playfair Cipher Key Matrix:");

    for (int i = 0; i < 5; i++)
        System.out.println(Arrays.toString(matrix[i]));
}

// function to preprocess plaintext
public String formatPlainText()
{
    String message = "";
    int len = plainText.length();

    for (int i = 0; i < len; i++)
    {
        // if plaintext contains the character 'j',
        // replace it with 'i'
        if (plainText.charAt(i) == 'j')
            message += 'i';
        else

```

```

        message += plainText.charAt(i);
    }

    // if two consecutive characters are same, then
    // insert character 'x' in between them
    for (int i = 0; i < message.length(); i += 2)
    {
        if (message.charAt(i) == message.charAt(i + 1))
            message = message.substring(0, i + 1) + 'x'
                + message.substring(i + 1);
    }

    // make the plaintext of even length
    if (len % 2 == 1)
        message += 'x'; // dummy character

    return message;
}

// function to group every two characters
public String[] formPairs(String message)
{
    int len = message.length();
    String[] pairs = new String[len / 2];

    for (int i = 0, cnt = 0; i < len / 2; i++)
        pairs[i] = message.substring(cnt, cnt += 2);

    return pairs;
}

// function to get position of character in key table
public int[] getCharPos(char ch)
{
    int[] keyPos = new int[2];

    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (matrix[i][j] == ch)
            {
                keyPos[0] = i;
                keyPos[1] = j;
                break;
            }
        }
    }
    return keyPos;
}

```

```

public String encryptMessage()
{
    String message = formatPlainText();
    String[] msgPairs = formPairs(message);
    String encText = "";

    for (int i = 0; i < msgPairs.length; i++)
    {
        char ch1 = msgPairs[i].charAt(0);
        char ch2 = msgPairs[i].charAt(1);
        int[] ch1Pos = getCharPos(ch1);
        int[] ch2Pos = getCharPos(ch2);

        // if both the characters are in the same row
        if (ch1Pos[0] == ch2Pos[0]) {
            ch1Pos[1] = (ch1Pos[1] + 1) % 5;
            ch2Pos[1] = (ch2Pos[1] + 1) % 5;
        }

        // if both the characters are in the same column
        else if (ch1Pos[1] == ch2Pos[1])
        {
            ch1Pos[0] = (ch1Pos[0] + 1) % 5;
            ch2Pos[0] = (ch2Pos[0] + 1) % 5;
        }

        // if both the characters are in different rows
        // and columns
        else {
            int temp = ch1Pos[1];
            ch1Pos[1] = ch2Pos[1];
            ch2Pos[1] = temp;
        }

        // get the corresponding cipher characters from
        // the key matrix
        encText = encText + matrix[ch1Pos[0]][ch1Pos[1]]
            + matrix[ch2Pos[0]][ch2Pos[1]];
    }

    return encText;
}

public class GFG {
    public static void main(String[] args)
    {
        System.out.println("Example-1\n");

        String key1 = "Problem";
        String plainText1 = "Playfair";
    }
}

```

```

    System.out.println("Key: " + key1);
    System.out.println("PlainText: " + plainText1);

    Playfair pfc1 = new Playfair(key1, plainText1);
    pfc1.cleanPlayFairKey();
    pfc1.generateCipherKey();

    String encText1 = pfc1.encryptMessage();
    System.out.println("Cipher Text is: " + encText1);

    System.out.println("\nExample-2\n");

    String key2 = "Problem";
    String plainText2 = "Hello";

    System.out.println("Key: " + key2);
    System.out.println("PlainText: " + plainText2);

    Playfair pfc2 = new Playfair(key2, plainText2);
    pfc2.cleanPlayFairKey();
    pfc2.generateCipherKey();

    String encText2 = pfc2.encryptMessage();
    System.out.println("Cipher Text is: " + encText2);
}
}
}}
```


OUTPUT:

Example-1

Key:diamond

PlainText: biramingham

Playfair Cipher Key Matrix:

[d, i, a, m,o]

[n, b, c, e, f]

[g, h, k, l, p]

[q, r, s, t, u]

[v, w, x, y, z]

Cipher Text is: hbtidbhkmo

Example-2

Key: genere

PlainText: jazz

Playfair Cipher Key Matrix:

[g, e, n, r, a]

[b, c, d, f, h]

[i, k, l, m, o]

[p, q, s, t, u]

[v, w, x, y, z]

Cipher Text is:ogvv

RESULT:

Thus the Playfair cipher substitution technique had been implemented successfully.

EX. NO: 1(C)

IMPLEMENTATION OF HILL CIPHER

AIM:

To write a java 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)

```
// Java code to implement Hill Cipher
public class GFG
{
    // Following function generates the
    // key matrix for the key string
    static void getKeyMatrix(String key, int keyMatrix[][])
    {
        int k = 0;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
```

```

        keyMatrix[i][j] = (key.charAt(k)) % 65;
        k++;
    }
}

// Following function encrypts the message
static void encrypt(int cipherMatrix[],
    int keyMatrix[],
    int messageVector[])
{
    int x, i, j;
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 1; j++)
        {
            cipherMatrix[i][j] = 0;

            for (x = 0; x < 3; x++)
            {
                cipherMatrix[i][j] +=
                    keyMatrix[i][x] * messageVector[x][j];
            }

            cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
        }
    }
}

// Function to implement Hill Cipher
static void HillCipher(String message, String key)
{
    // Get key matrix from the key string
    int [][]keyMatrix = new int[3][3];
    getKeyMatrix(key, keyMatrix);

    int [][]messageVector = new int[3][1];

    // Generate vector for the message
    for (int i = 0; i < 3; i++)
        messageVector[i][0] = (message.charAt(i)) % 65;

    int [][]cipherMatrix = new int[3][1];

    // Following function generates
    // the encrypted vector
    encrypt(cipherMatrix, keyMatrix, messageVector);

    String CipherText="";

    // Generate the encrypted text from
    // the encrypted vector
    for (int i = 0; i < 3; i++)

```

```
CipherText += (char)(cipherMatrix[i][0] + 65);

// Finally print the ciphertext
System.out.print(" Ciphertext:" + CipherText);
}

// Driver code
public static void main(String[] args)
{
    // Get the message to be encrypted
    String message = "HILLCIPHERALGORITHM";

    // Get the key
    String key = "NWKC";

    HillCipher(message, key);
}
}
```

OUTPUT:

Ciphertext:HIVCUKLIKWIWWKHELWOW

RESULT:

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

AIM:

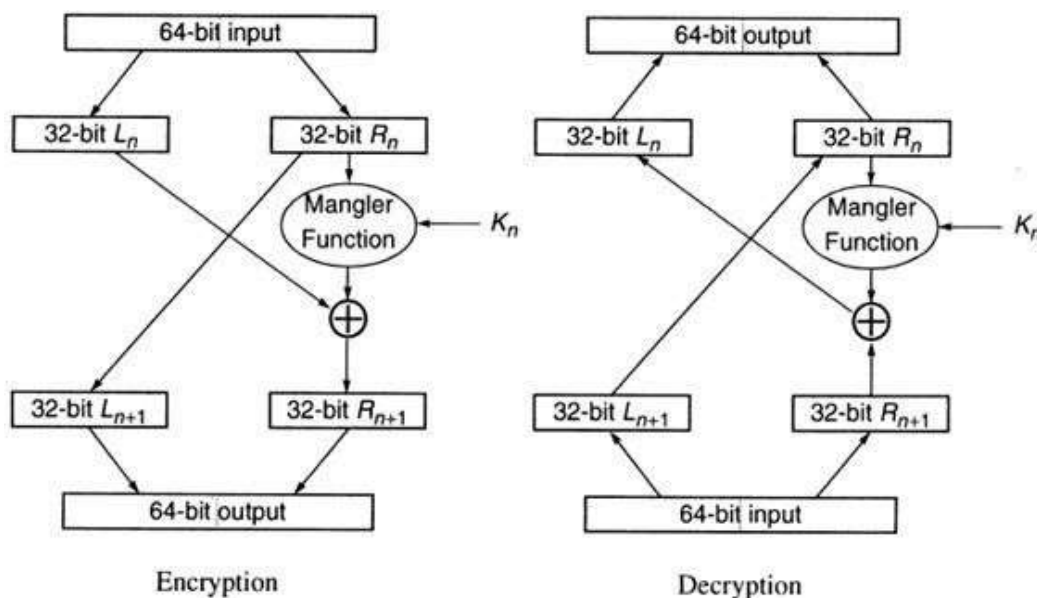
To write a java program to implement Data Encryption Standard (DES).

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 java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class DesProgram
{
    //creating an instance of the Cipher class for encryption
    private static Cipher encrypt;
    //creating an instance of the Cipher class for decryption
    private static Cipher decrypt;
    //initializing vector
    private static final byte[] initialization_vector = { 22, 33, 11,
        44, 55, 99, 66, 77 };
    //main() method
    public static void main(String[] args)
    {
        //path of the file that we want to encrypt
        String textFile = "C:/Users/Anubhav/Desktop/DemoData.txt";
        //path of the encrypted file that we get as output
        String encryptedData = "C:/Users/Anubhav/Desktop/encrypteddata.txt";
        //path of the decrypted file that we get as output
        String decryptedData = "C:/Users/Anubhav/Desktop/decrypteddata.txt";
        try
        {
            //generating keys by using the KeyGenerator class
            SecretKey scrtkey = KeyGenerator.getInstance("DES").generateKey();
```

```

AlgorithmParameterSpec aps = new
IvParameterSpec(initialization_vector);
//setting encryption mode
encrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);
//setting decryption mode
decrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);
//calling encrypt() method to encrypt the file
encryption(new FileInputStream(textFile), new
FileOutputStream(encryptedData));
//calling decrypt() method to decrypt the file
decryption(new FileInputStream(encryptedData), new
FileOutputStream(decryptedData));
//prints the statment if the program runs successfully
System.out.println("The encrypted and decrypted files have been
created successfully.");
}
//catching multiple exceptions by using the | (or) operator in a
single catch block
catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException | InvalidAlgorithmParameterException |
IOException e)
{
//prints the message (if any) related to exceptions
e.printStackTrace();
}
}
//method for encryption
private static void encryption(InputStream input, OutputStream
output)
throws IOException
{
output = new CipherOutputStream(output, encrypt);
//calling the writeBytes() method to write the encrypted bytes to
the file
writeBytes(input, output);
}
//method for decryption
private static void decryption(InputStream input, OutputStream
output)
throws IOException
{
input = new CipherInputStream(input, decrypt);
//calling the writeBytes() method to write the decrypted bytes to
the file
writeBytes(input, output);
}
//method for writting bytes to the files
private static void writeBytes(InputStream input, OutputStream
output)
throws IOException
{
byte[] writeBuffer = new byte[512];
int readBytes = 0;
while ((readBytes = input.read(writeBuffer)) >= 0)
{

```



```
output.write(writeBuffer, 0, readBytes);  
}  
//closing the output stream  
output.close();  
//closing the input stream  
input.close();  
}  
}
```

OUTPUT:

Enter message to encrypt :

The habit of persistence is the habit of victory.

Decrypted message :

L#@\$@kjs#@\$ask&^jdvf(*&kjbaj&^%\$sads%\$kjbskjb

RESULT:

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

AIM:

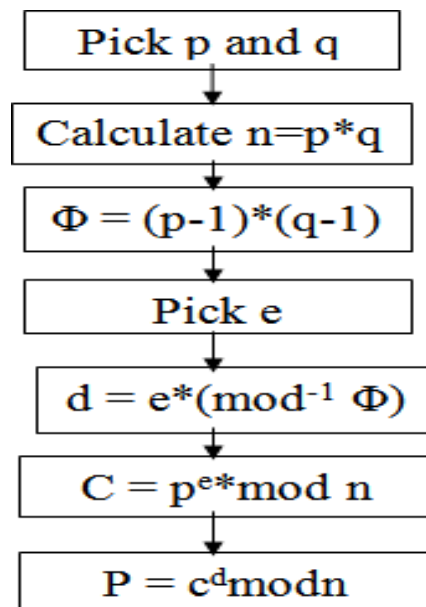
To write a java program to implement the RSA encryption algorithm.

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 $e * \text{mod}^{-1}(z)$.

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

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

PROGRAM: (RSA)

// Java Program to Implement the RSA Algorithm

```
import java.math.*;
```

```
import java.util.*;
```

```
public class RSA {
```

```
    public static void main(String args[])
```

```
    {
```

```
        int p, q, n, z, d = 0, e, i;
```

```
        // The number to be encrypted and decrypted
```

```
        int msg = 14;
```

```
        double c;
```

```
        BigInteger msgback;
```

```
        // 1st prime number p
```

```
        p = 3;
```

```
        // 2nd prime number q
```

```
        q = 11;
```

```
        n = p * q;
```

```
        z = (p - 1) * (q - 1);
```

```
        System.out.println("the value of z = " + z);
```

```
        for (e = 2; e < z; e++) {
```

```

        // e is for public key exponent
        if (gcd(e, z) == 1) {
            break;
        }
    }
    System.out.println("the value of e = " + e);
    for (i = 0; i <= 9; i++) {
        int x = 1 + (i * z);

        // d is for private key exponent
        if (x % e == 0) {
            d = x / e;
            break;
        }
    }
    System.out.println("the value of d = " + d);
    c = (Math.pow(msg, e)) % n;
    System.out.println("Encrypted message is : " + c);

    // converting int value of n to BigInteger
    BigInteger N = BigInteger.valueOf(n);

    // converting float value of c to BigInteger
    BigInteger C = BigDecimal.valueOf(c).toBigInteger();
    msgback = (C.pow(d)).mod(N);
    System.out.println("Decrypted message is : "
        + msgback);
}

static int gcd(int e, int z)
{
    if (e == 0)
        return z;
    else
        return gcd(z % e, e);
}

```

}

OUTPUT:

the value of $z = 20$

the value of $e = 3$

the value of $d = 7$

Encrypted message is : 5.0

Decrypted message is : 14

RESULT:

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

EX. NO: 2(C)

IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

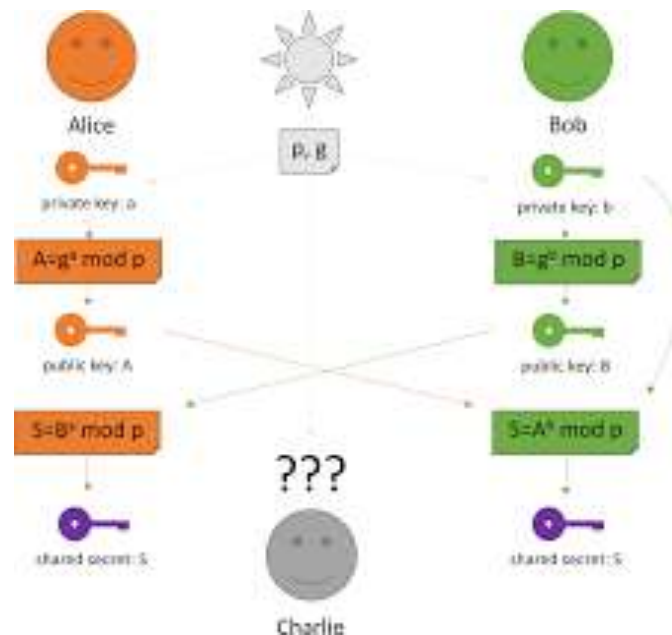
AIM:

To implement the Diffie-Hellman Key Exchange algorithm using java 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 $g^a \text{ mod } p$.

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)

```
// This program calculates the Key for two persons
// using the Diffie-Hellman Key exchange algorithm
public class GFG{
```

```
// Power function to return value of  $a^b \bmod P$ 
private static long power(long a, long b, long p)
{
    if (b == 1)
        return a;
    else
        return (((long)Math.pow(a, b)) % p);
}
```

```
// Driver code
public static void main(String[] args)
{
    long P, G, x, a, y, b, ka, kb;

    // Both the persons will be agreed upon the
    // public keys G and P

    // A prime number P is taken
    P = 33;
    System.out.println("The value of P:" + P);

    // A primitive root for P, G is taken
    G = 8;
    System.out.println("The value of G:" + G);
```

```
// Alice will choose the private key a
// a is the chosen private key
a = 3;
System.out.println("The private key a for Alice:" + a);

// Gets the generated key
x = power(G, a, P);

// Bob will choose the private key b
// b is the chosen private key
b = 3;
System.out.println("The private key b for Bob:" + b);

// Gets the generated key
y = power(G, b, P);

// Generating the secret key after the exchange
// of keys
ka = power(y, a, P); // Secret key for Alice
kb = power(x, b, P); // Secret key for Bob

System.out.println("Secret key for the Alice is:" + ka);
System.out.println("Secret key for the Bob is:" + kb);
}
}
```


OUTPUT:

The value of P:33

The value of G:8

The private key a for Alice:3

The private key b for Bob:2

Secret key for the Alice is:25

Secret key for the Bob is:25

RESULT:

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

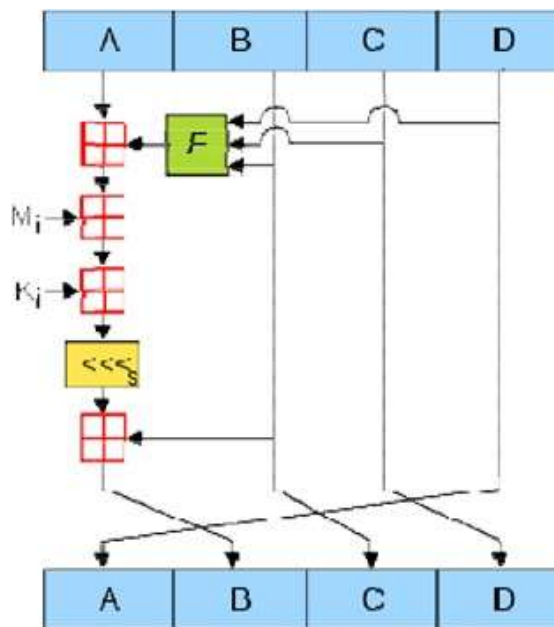
EX. NO: 2(D)

IMPLEMENTATION OF MD5**AIM:**

To write a java 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)

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

// Java program to calculate MD5 hash value
public class MD5 {
    public static String getMd5(String input)
    {
        try {

            // Static getInstance method is called with hashing MD5
            MessageDigest md = MessageDigest.getInstance("MD5");

            // digest() method is called to calculate message digest
            // of an input digest() return array of byte
            byte[] messageDigest = md.digest(input.getBytes());

            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
    }
}
```

```
// For specifying wrong message digest algorithms
catch (NoSuchAlgorithmException e) {
    throw new RuntimeException(e);
}

// Driver code
public static void main(String args[]) throws NoSuchAlgorithmException
{
    String s = "hello world";
    System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));
}
}
```

OUTPUT:

Your HashCode Generated by MD5 is: e39b9c178b2c9be4e99b141d956c6ff6

RESULT:

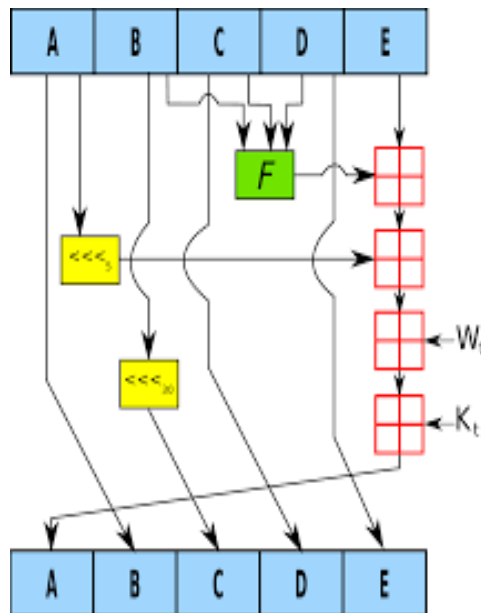
Thus the implementation of MD5 hashing algorithm had been implemented successfully using java.

AIM:

To implement the SHA-I hashing technique using java 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.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class GFG {
    public static String encryptThisString(String input)
    {
        try {
            // getInstance() method is called with algorithm
SHA-1
            MessageDigest md = MessageDigest.getInstance("SHA-
1");

            // digest() method is called
            // to calculate message digest of the input string
            // returned as array of byte
            byte[] messageDigest = md.digest(input.getBytes());

            // Convert byte array into signum representation
            BigInteger no = new BigInteger(1, messageDigest);

            // Convert message digest into hex value
            String hashtext = no.toString(16);

            // Add preceding 0s to make it 32 bit
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }

            // return the HashText
            return hashtext;
        }

        // For specifying wrong message digest algorithms
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    // Driver code
    public static void main(String args[]) throws

    NoSuchAlgorithmException
    {

        System.out.println("HashCode Generated by SHA-1 for: ");
    }
}
```

```
        String s1 = "Network";
        System.out.println("\n" + s1 + " : " +
encryptThisString(s1));

        String s2 = "hello world";
        System.out.println("\n" + s2 + " : " +
encryptThisString(s2));
    }
}
```

OUTPUT:

HashCode Generated by SHA-1 for:

Network : addf120b430021c36c232c99ef8d926aea2acd6b

hello world : 2aae6c35c94fcfb415dbe95f408b9ce91ee846ed

RESULT:

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

EX. NO: 3

IMPLEMENTATION OF DIGITAL SIGNATURE STANDARD

AIM:

To write a Java 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)

```
package java_cryptography;

// Imports
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.util.Scanner;

import javax.xml.bind.DatatypeConverter;

public class Digital_Signature_GeeksforGeeks {

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

    // Function to implement Digital signature
    // using SHA256 and RSA algorithm
    // by passing private key.
    public static byte[] Create_Digital_Signature(
```

```

        byte[] input,
        PrivateKey Key)
        throws Exception
    {
        Signature signature
            = Signature.getInstance(
                SIGNING_ALGORITHM);
        signature.initSign(Key);
        signature.update(input);
        return signature.sign();
    }

    // Generating the asymmetric key pair
    // using SecureRandom class
    // functions and RSA algorithm.
    public static KeyPair Generate_RSA_KeyPair()
        throws Exception
    {
        SecureRandom secureRandom
            = new SecureRandom();
        KeyPairGenerator keyPairGenerator
            = KeyPairGenerator
                .getInstance(RSA);
        keyPairGenerator
            .initialize(
                2048, secureRandom);
        return keyPairGenerator
            .generateKeyPair();
    }

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

    // Driver Code
    public static void main(String args[])
        throws Exception
    {
        String input
            = "GEEKSFORGEEKS IS A"
            + " COMPUTER SCIENCE PORTAL";
        KeyPair keyPair
            = Generate_RSA_KeyPair();
    }

```

```

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

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

        System.out.println(
            "Verification: "
            + Verify_Digital_Signature(
                input.getBytes(),
                signature, keyPair.getPublic()));
    }
}

```

OUTPUT:

Signature Value :

24920335AE7782EEV75E18C1C76651384FDE30178DBE806A67EA4C884D52BF15A35CB8D1F

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

Download

Gpg4win 2.3.3 (Released: 2016-08-18)

You can download the full version (including the Gpg4win compendium) of Gpg4win 2.3.3 here:

Gpg4win 2.3.3

Size: 26 MB

↓

OpenPGP signature (for gpg4win-2.3.3.exe)

SHA1 checksum (for gpg4win-2.3.3.exe): 67e13c4f90ff6a70ad57bd31af64a238c9315308

Changelog

Gpg4win 2.3.3 contains:

- GnuPG 2.0.30
- Kleopatra 2.2.0-gitfb4ae3d
- GPA 0.9.9
- GpgOL 1.4.0
- GpgEX 1.0.4
- Kompendium (de) 3.0.0
- Kompendium (en) 3.0.0

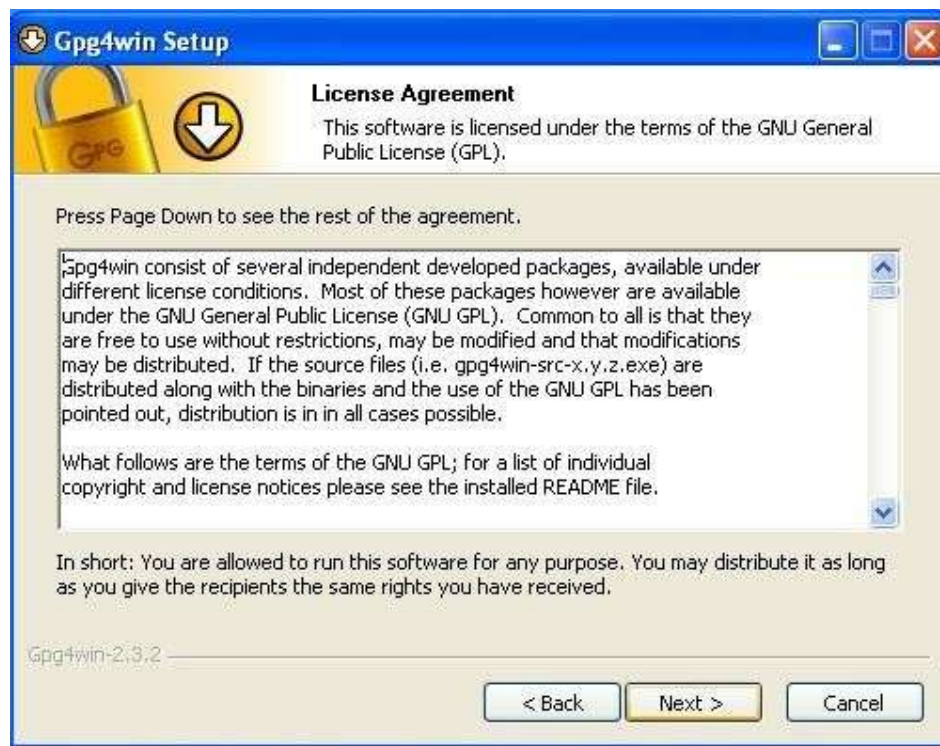
2. On the following screen, click the “Download Gpg4win” button.



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



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



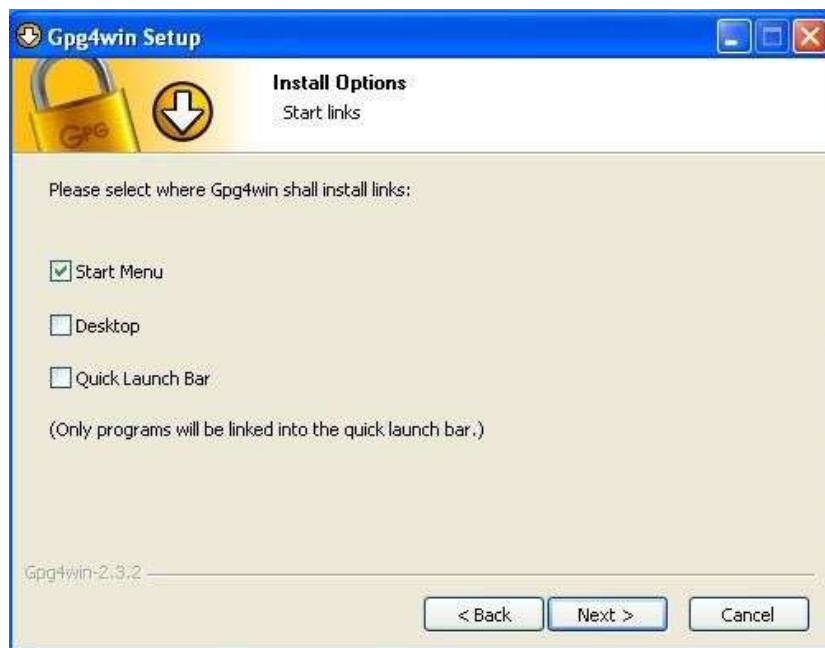
5. Set the check box values as specified below, then click the “Next” button



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



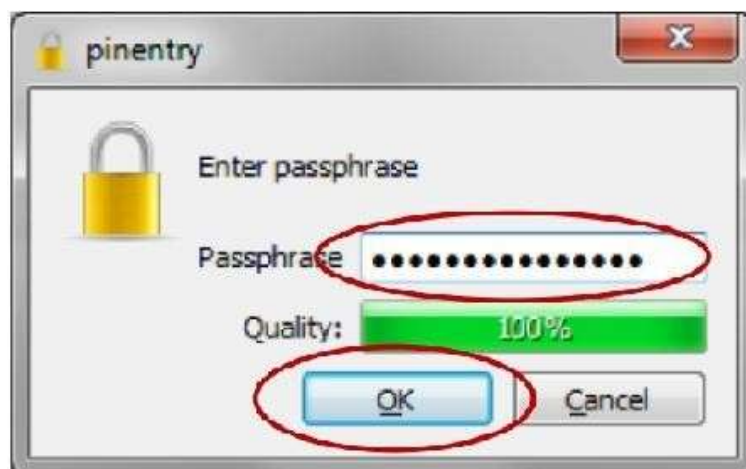
- Specify where you want shortcuts to the software placed, then click the “Next” button.



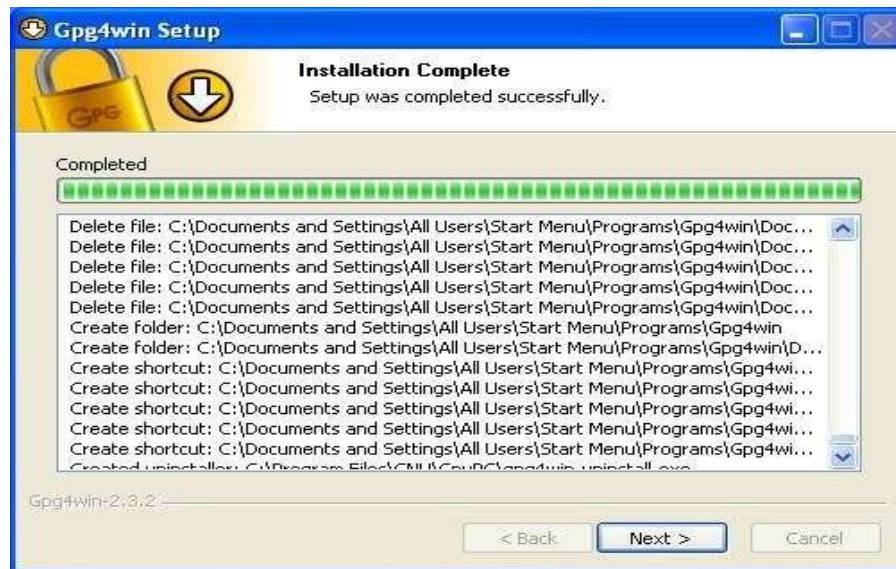
8. If you selected to have a GPG shortcut in your Start Menu, specify the folder in which it will be placed. The default “Gpg4win” is OK. Click the “Install” button to continue



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



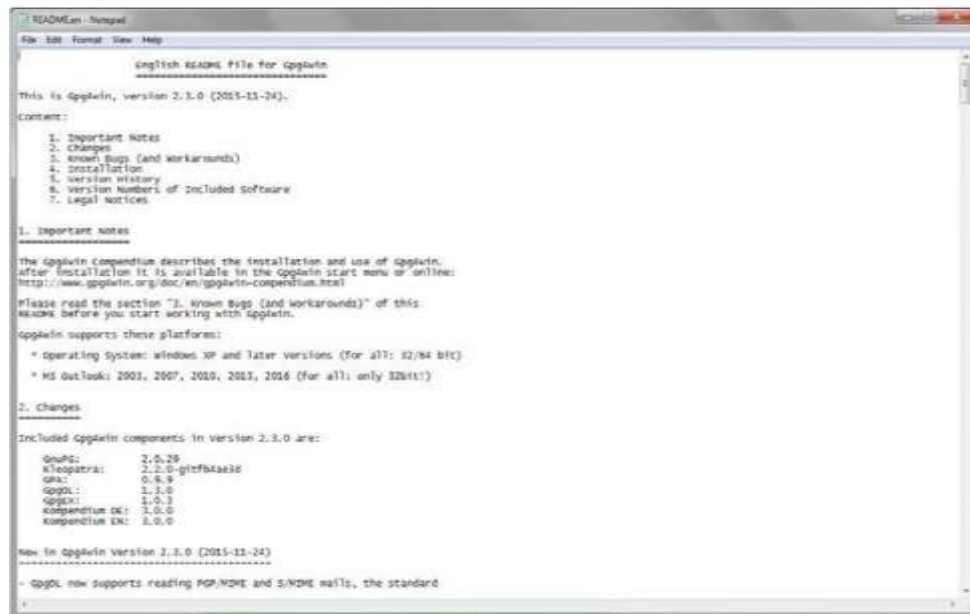
10. The installation process will tell you when it is complete. Click the “Next” button



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.



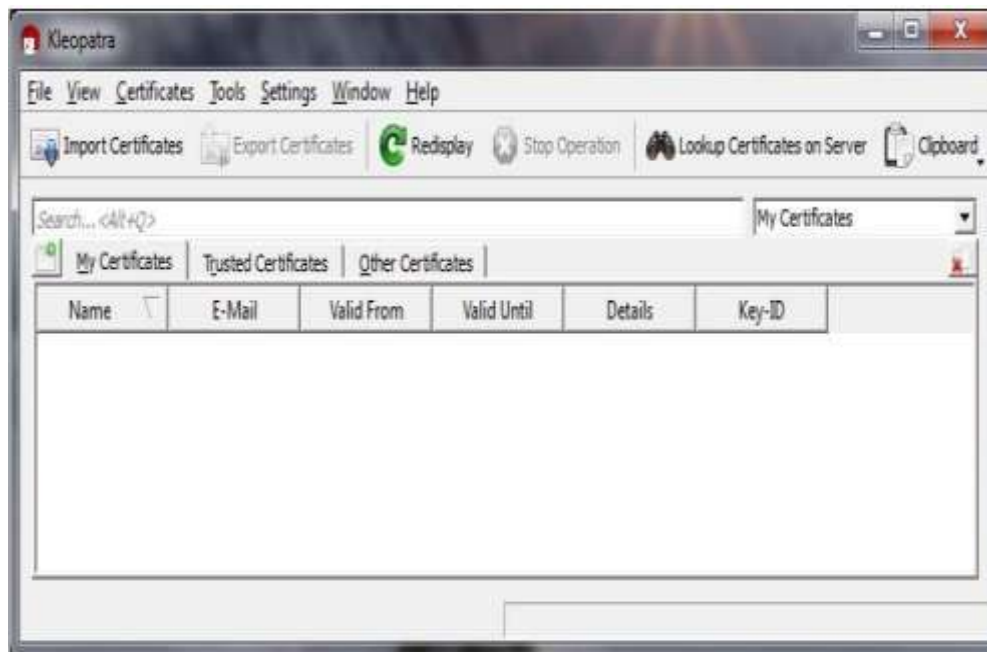
CREATING YOUR PUBLIC AND PRIVATE KEYS

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:

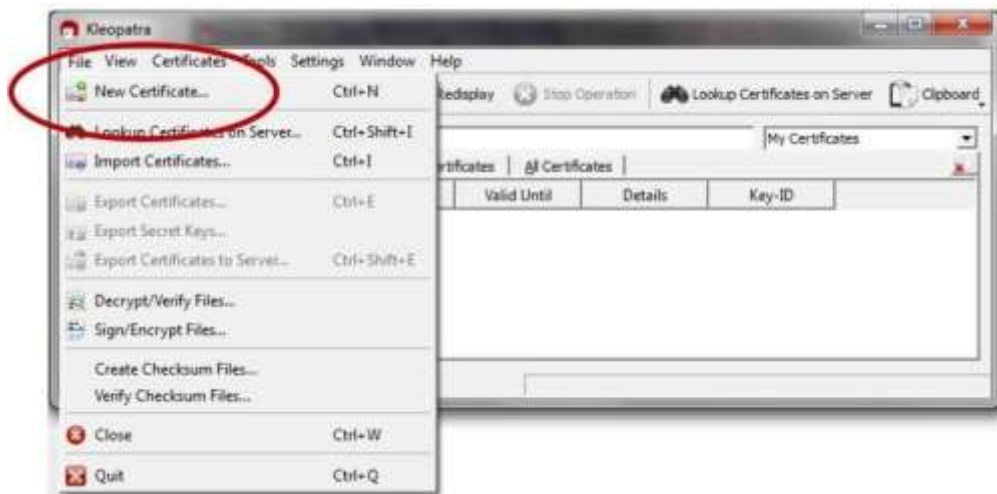
1. From your start bar, select the “Kleopatra” icon to start the Kleopatra certificate management software



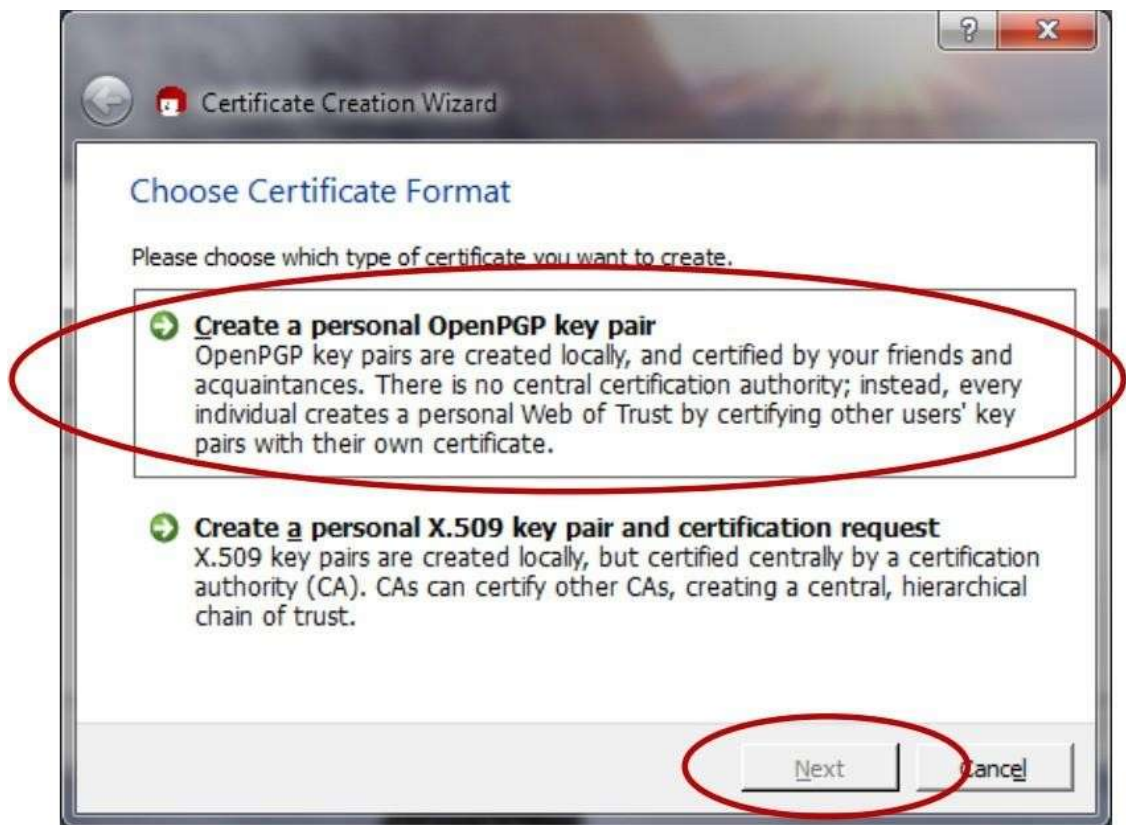
2. The following screen will be displayed



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



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



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



The screenshot shows the 'Certificate Creation Wizard' window with the 'Enter Details' tab selected. The window has a title bar with a question mark and a close button. Below the title bar is a navigation bar with a back arrow and a forward arrow. The main content area contains the following text: 'Please enter your personal details below. If you want more control over the certificate parameters, click on the Advanced Settings button.' Below this text are three input fields: 'Name:' (required), 'Email:' (required), and 'Comment:' (optional). The 'Name' field is empty, and a red error message 'Name is required, but empty.' is displayed below it. The 'Email' field is also empty. The 'Comment' field is empty. At the bottom right, there is an 'Advanced Settings...' button. At the bottom center, there are 'Next' and 'Cancel' buttons.

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



The screenshot shows the 'Certificate Creation Wizard' window with the 'Enter Details' tab selected. The window has a title bar with a question mark and a close button. Below the title bar is a navigation bar with a back arrow and a forward arrow. The main content area contains the following text: 'Please enter your personal details below. If you want more control over the certificate parameters, click on the Advanced Settings button.' Below this text are three input fields: 'Name:' (required), 'Email:' (required), and 'Comment:' (optional). The 'Name' field contains 'Anthony Scaturro', the 'Email' field contains 'scaturro@uhd.edu', and the 'Comment' field contains 'My certificate for GPG'. A red oval highlights the input fields. Below the input fields, the text 'Anthony Scaturro (My certificate for GPG) <scaturro@uhd.edu>' is displayed. At the bottom right, there is an 'Advanced Settings...' button. At the bottom center, there are 'Next' and 'Cancel' buttons. A red oval highlights the 'Next' button.

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



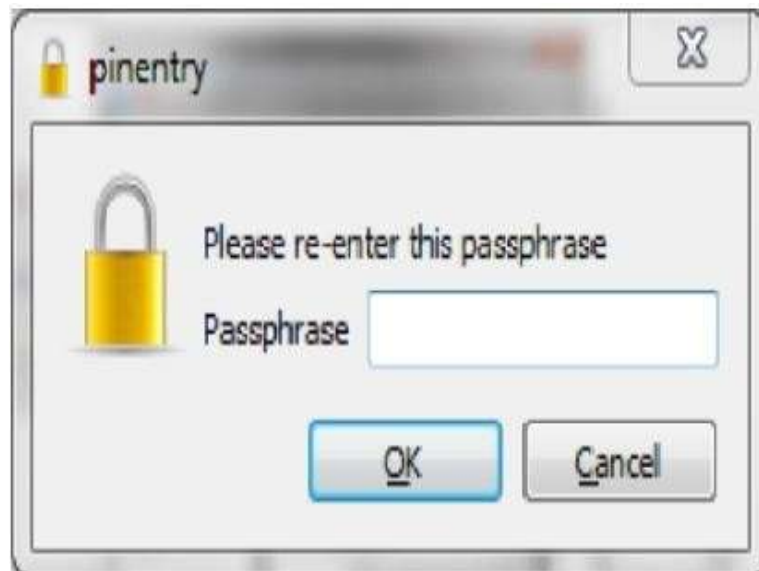
8. You will be asked to enter a passphrase



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



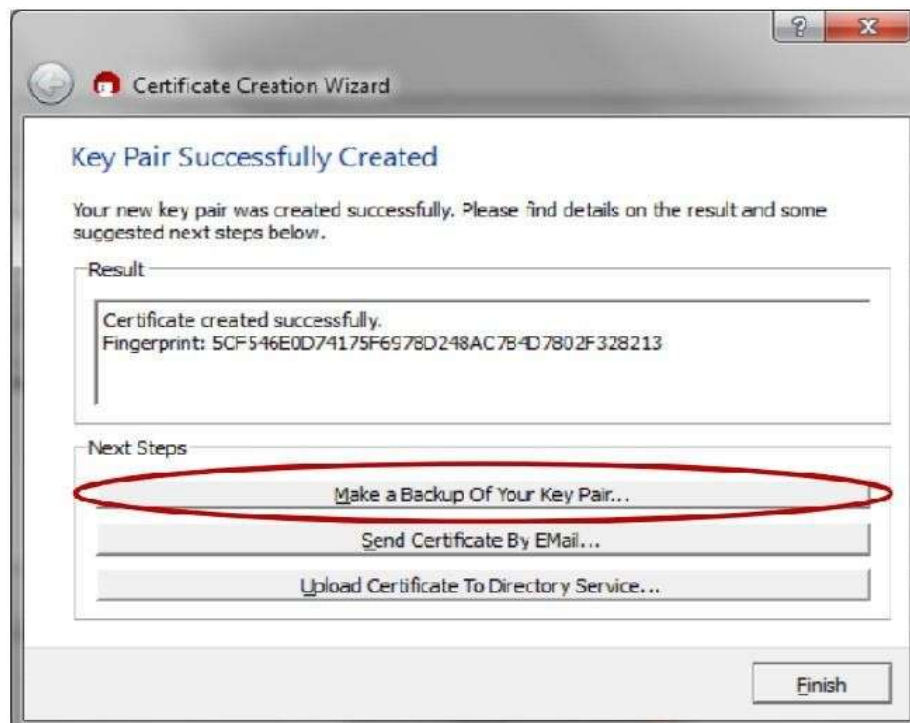
10. You will be asked to re-enter the passphrase



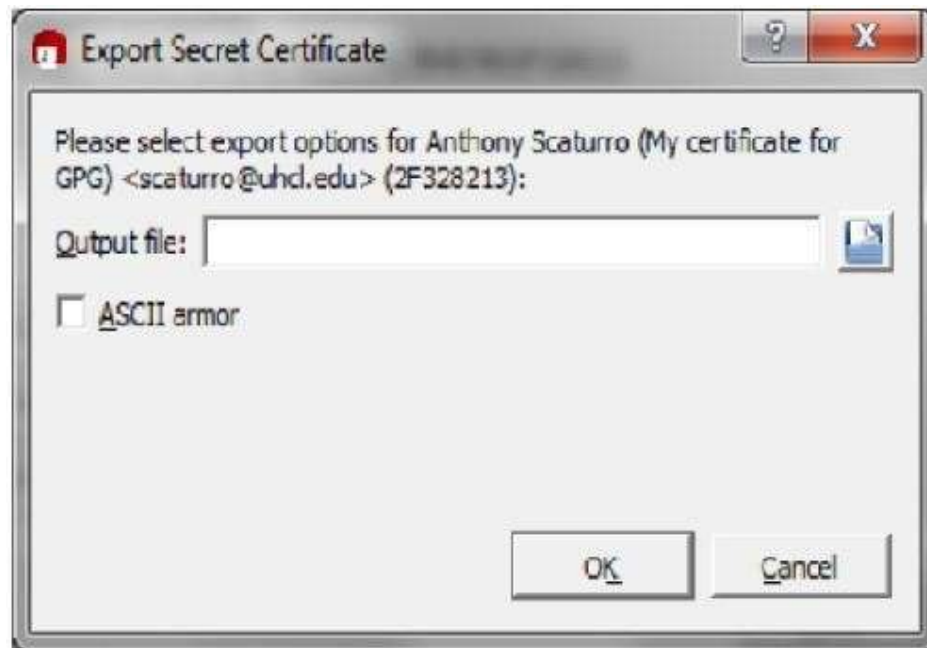
11. Re-enter the passphrase value. Then click the “OK” button. If the passphrases match, the certificate will be created.



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.



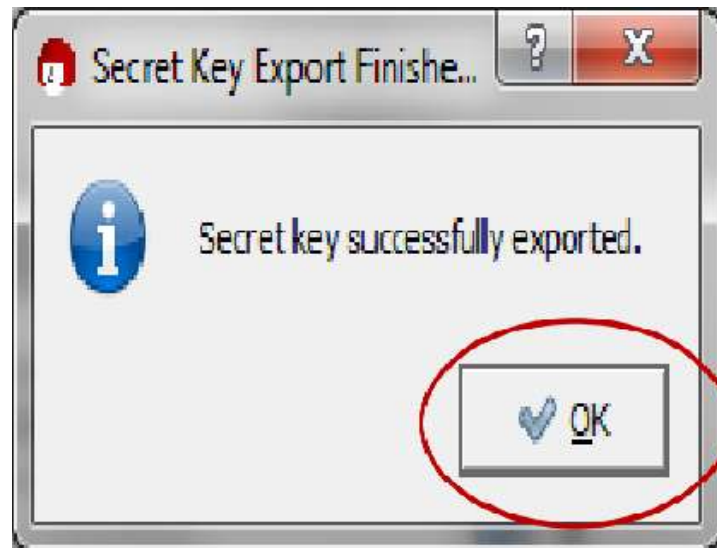
13. If you choose to backup your key pair, you will be presented with the following screen:



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.

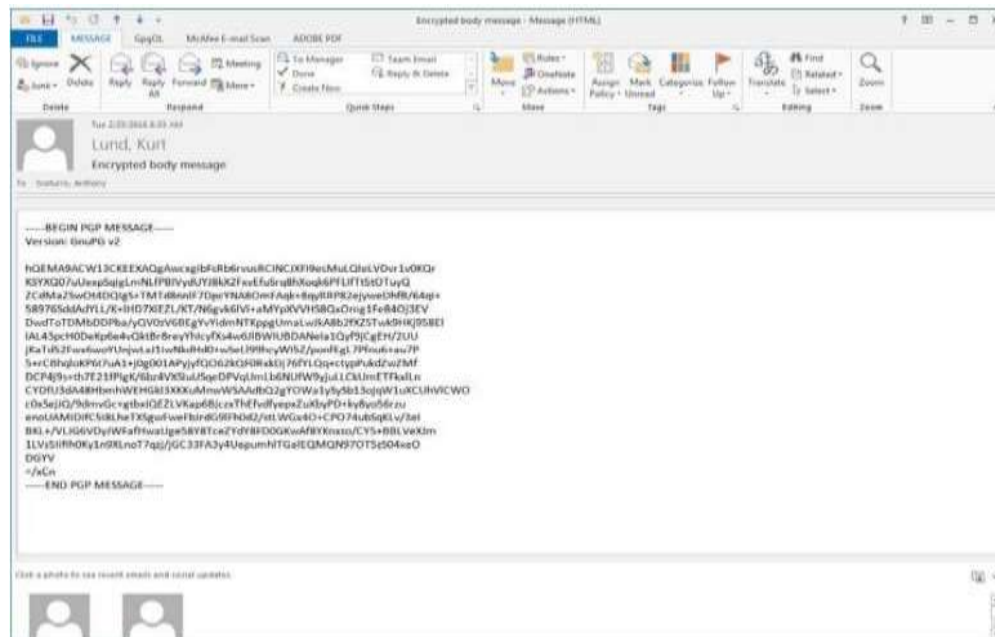


- Before the program closes, you will need to confirm that you want to close the program by clicking on the “Quit Kleopatra” button

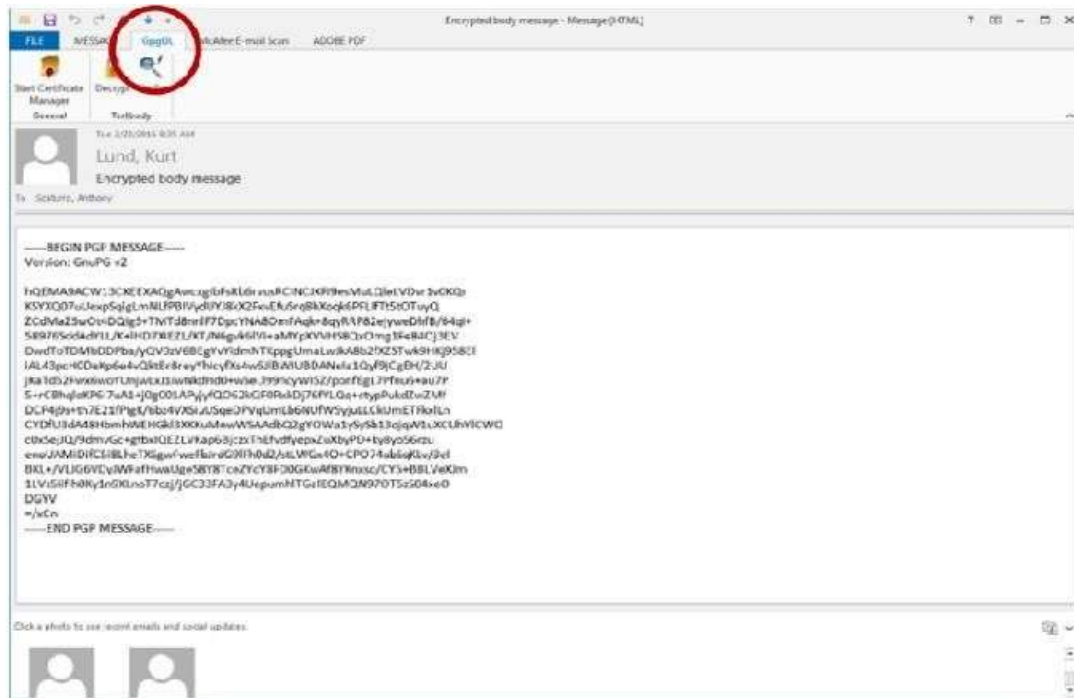


DECRYPTING AN ENCRYPTED E-MAIL THAT HAS BEEN SENT TO YOU:

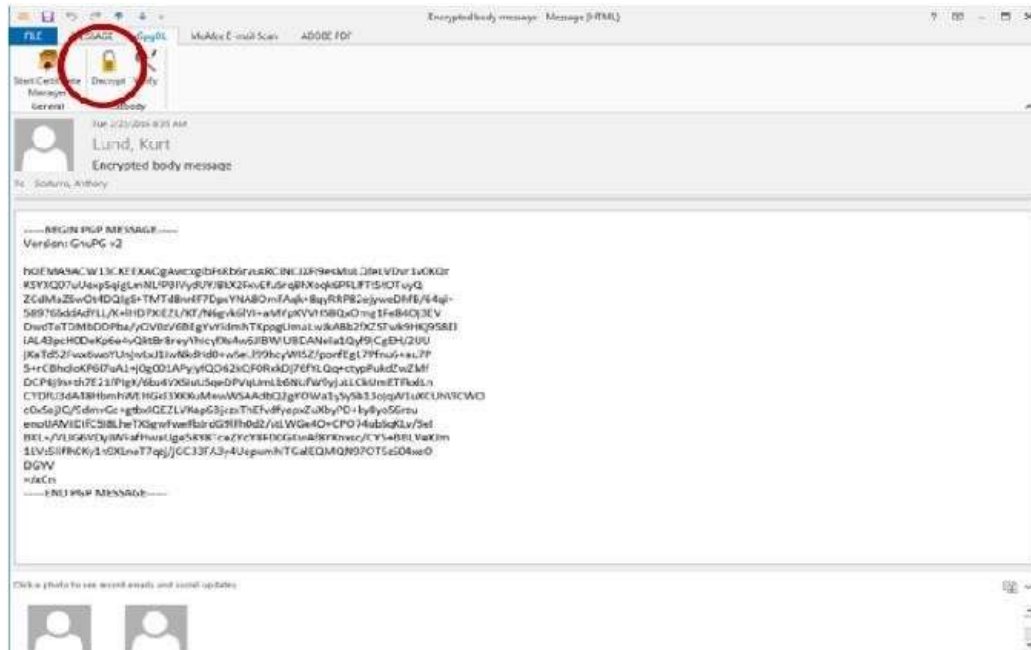
- Open the e-mail message



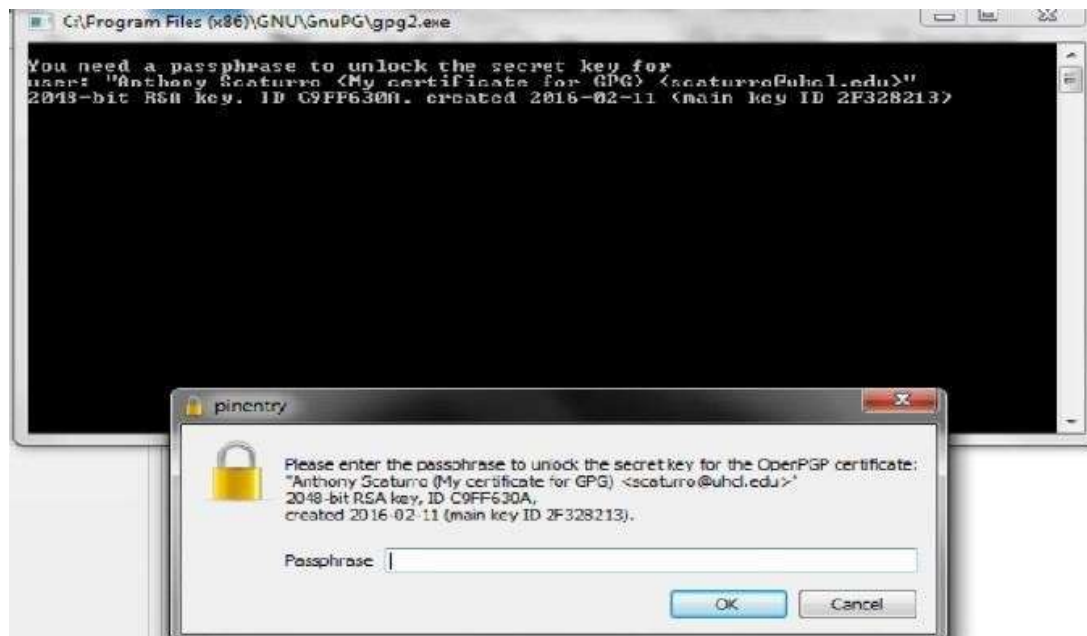
2. Select the GpgOL tab



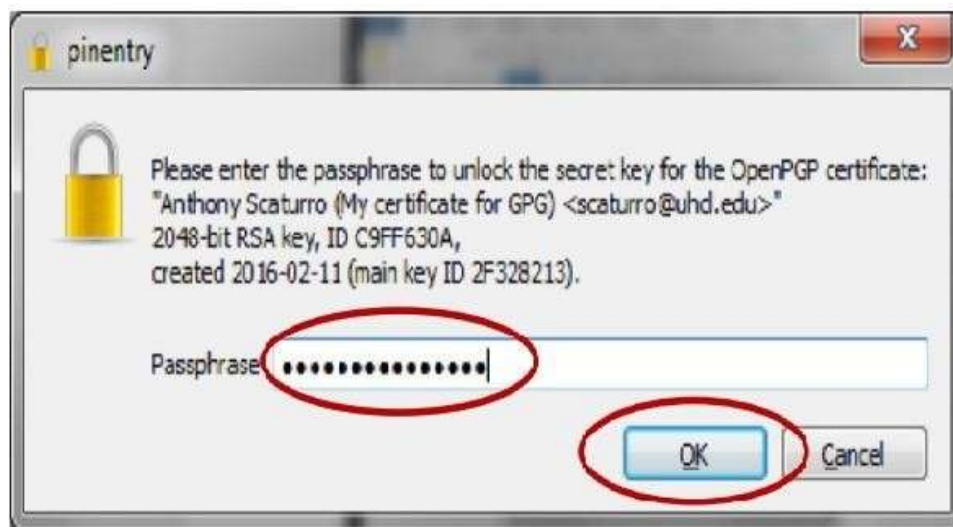
3. Click the "Decrypt" button



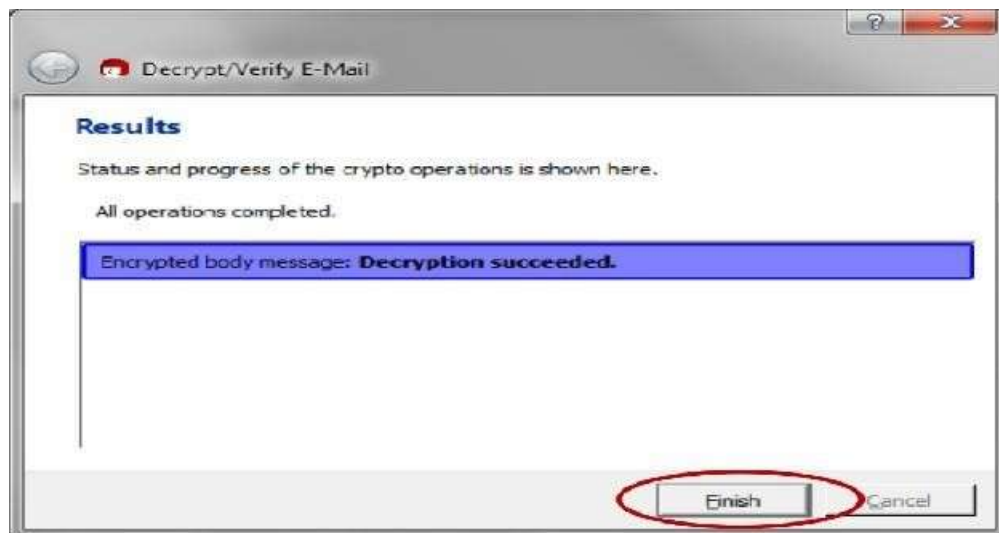
4. A command window will open along with a window that asks for the Passphrase to your private key that will be used to decrypt the incoming message.



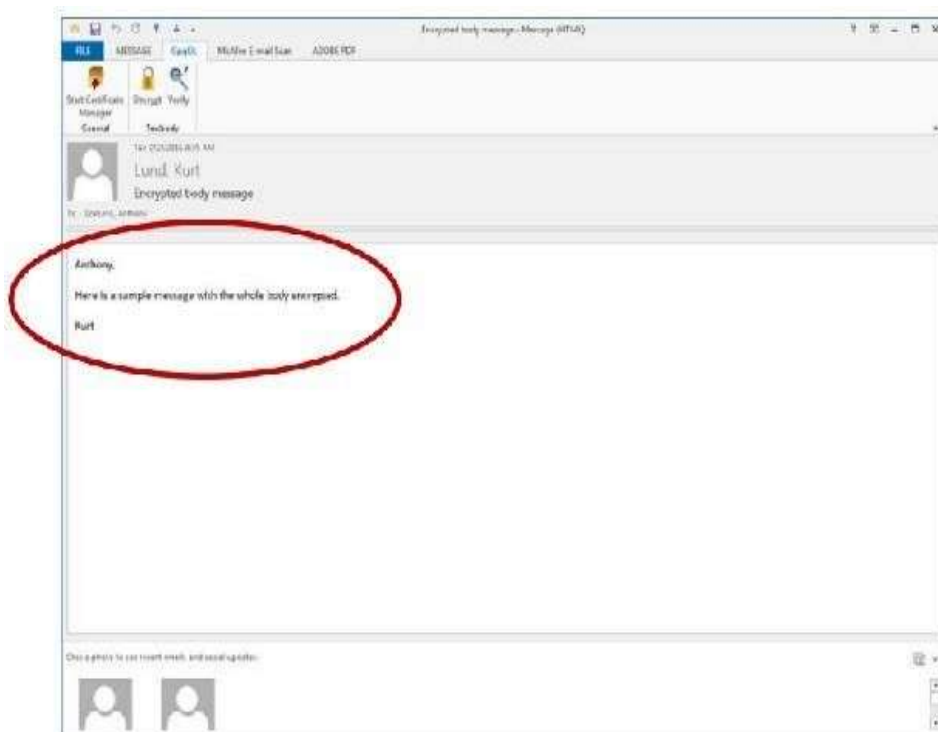
5. Enter your passphrase and click the "OK" button



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

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:

KESensor Professional - Evaluation Trial

File View Scenario Signatures Settings Help

Hosts: localhost

ID	Start	Duration	Pr...	Sens...	Name	Victor	Sig. Message	Received
26	3/1/2012 9:54:46 AM 656	0.000	UDP	128	NET Datagram ...	BTSP-PC		NET Datagram Packet: id:1
25	3/1/2012 9:54:25 AM 515	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
24	3/1/2012 9:54:25 AM 515	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
23	3/1/2012 9:54:11 AM 343	0.000	UDP	138	NET Datagram ...	COMP15		NET Datagram Packet: id:1
22	3/1/2012 9:53:28 AM 968	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
21	3/1/2012 9:53:28 AM 968	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
20	3/1/2012 9:53:22 AM 328	0.000	UDP	138	NET Datagram ...	COMPUTER14		NET Datagram Packet: id:1
19	3/1/2012 9:53:09 AM 593	0.000	UDP	138	NET Datagram ...	FRONT OFFICE PC		NET Datagram Packet: id:1
18	3/1/2012 9:52:56 AM 493	0.000	UDP	138	NET Datagram ...	COMP1		NET Datagram Packet: id:1
17	3/1/2012 9:52:54 AM 656	0.000	UDP	138	NET Datagram ...	COMP15		NET Datagram Packet: id:1
16	3/1/2012 9:52:54 AM 659	0.000	UDP	138	NET Datagram ...	comp15		NET Datagram Packet: id:1
15	3/1/2012 9:52:46 AM 696	0.000	UDP	67	DHCP Client	192.168.1.1		[02 01 06 00 00 00]DHCP
14	3/1/2012 9:52:46 AM 234	0.000	UDP	67	DHCP	comp15		DHCP: Boot Request/DNA
13	3/1/2012 9:52:41 AM 734	0.000	UDP	138	NET Datagram ...	CIVILDEPT		NET Datagram Packet: id:1
12	3/1/2012 9:52:36 AM 756	0.000	UDP	138	NET Datagram ...	BTSP-PC		NET Datagram Packet: id:1
11	3/1/2012 9:52:31 AM 676	0.000	UDP	67	DHCP	BTSP-PC		DHCP: Boot Request/DNA
10	3/1/2012 9:52:28 AM 553	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
9	3/1/2012 9:52:23 AM 696	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
8	3/1/2012 9:52:23 AM 615	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
7	3/1/2012 9:52:11 AM 562	0.000	UDP	138	NET Datagram ...	CIVILDEPT		NET Datagram Packet: id:1
6	3/1/2012 9:52:06 AM 716	0.000	UDP	138	NET Datagram ...	comp15		NET Datagram Packet: id:1
5	3/1/2012 9:51:55 AM 83	0.000	UDP	138	NET Datagram ...	COMP1		NET Datagram Packet: id:1
4	3/1/2012 9:51:49 AM 937	0.000	UDP	138	NET Datagram ...	COMP15		NET Datagram Packet: id:1
3	3/1/2012 9:51:30 AM 586	0.000	UDP	138	NET Datagram ...	COMP13		NET Datagram Packet: id:1
2	3/1/2012 9:51:20 AM 974	0.000	UDP	67	DHCP Client	192.168.1.1		[02 01 06 00 00 00]DHCP
1	3/1/2012 9:51:20 AM 968	0.000	UDP	67	DHCP	comp15		DHCP: Boot Request/DNA

KESensor Professional - Evaluation Trial

File View Scenario Signatures Settings Help

Hosts: 178.76.252.26 - Recent...

ID	Start	Duration	Pr...	Sens...	Name	Victor	Sig. Message	Received
44	3/1/2012 9:56:17 AM 398	0.000	UDP	1222	UDP Packet:	222.107.67.174		[0E 01 15 00 00 00]JAV
43	3/1/2012 9:56:17 AM 177	0.000	UDP	1224	UDP Packet:	178.76.252.26		[04 00 00 00 00 00]M
42	3/1/2012 9:56:16 AM 895	0.000	UDP	1223	UDP Packet:	178.76.252.26		[CA C1 00 00 FF 80 E]
41	3/1/2012 9:57:08 AM 968	0.000	UDP	67	DHCP	BTSP-PC		DHCP: Boot Request
40	3/1/2012 9:56:16 AM 211	0.000	UDP	1217	UDP Packet:	222.107.67.174		[07 00 00 00 04 00]P
39	3/1/2012 9:56:25 AM 375	0.000	UDP	138	NET Datagram ...	comp15		NET Datagram Packet: id:1
38	3/1/2012 9:56:16 AM 147	0.000	UDP	1220	UDP Packet:	178.76.252.26		[E1 06 15 00 00 00]H
37	3/1/2012 9:56:15 AM 621	0.000	UDP	1219	UDP Packet:	178.76.252.26		[8A 0F 17 00 0A 00]S
36	3/1/2012 9:56:29 AM 125	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
35	3/1/2012 9:56:29 AM 125	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
34	3/1/2012 9:56:15 AM 262	0.000	UDP	1218	UDP Packet:	122.45.101.67		[00 1F 00 00 00 00]C
33	3/1/2012 9:56:15 AM 049	0.000	UDP	1216	UDP Packet:	122.45.101.67		6 [19 00 00 00 C9 00]
32	3/1/2012 9:56:14 AM 652	0.000	UDP	1215	UDP Packet:	122.45.101.67		[02 00 00 F6 FF 0E 0
31	3/1/2012 9:55:54 AM 406	0.000	UDP	138	NET Datagram ...	ELECTRICALDEPT		NET Datagram Packet: id:1
30	3/1/2012 9:55:29 AM 093	0.016	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
29	3/1/2012 9:55:29 AM 046	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
28	3/1/2012 9:55:28 AM 098	0.000	UDP	66	DHCP Client	192.168.1.1		[02 01 06 00 00 00]DHCP
27	3/1/2012 9:55:28 AM 078	0.000	UDP	67	DHCP	comp15		DHCP: Boot Request
26	3/1/2012 9:54:40 AM 656	0.000	UDP	138	NET Datagram ...	BTSP-PC		NET Datagram Packet: id:1
25	3/1/2012 9:54:29 AM 015	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
24	3/1/2012 9:54:29 AM 015	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
23	3/1/2012 9:54:11 AM 343	0.000	UDP	138	NET Datagram ...	COMP15		NET Datagram Packet: id:1
22	3/1/2012 9:53:28 AM 968	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
21	3/1/2012 9:53:28 AM 968	0.000	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE
20	3/1/2012 9:53:22 AM 328	0.000	UDP	138	NET Datagram ...	COMPUTER14		NET Datagram Packet: id:1
19	3/1/2012 9:53:09 AM 593	0.000	UDP	138	NET Datagram ...	FRONT OFFICE PC		NET Datagram Packet: id:1
18	3/1/2012 9:52:56 AM 493	0.000	UDP	138	NET Datagram ...	COMP1		NET Datagram Packet: id:1
17	3/1/2012 9:52:54 AM 656	0.000	UDP	138	NET Datagram ...	COMP15		NET Datagram Packet: id:1
16	3/1/2012 9:52:54 AM 690	0.000	UDP	138	NET Datagram ...	comp15		NET Datagram Packet: id:1
15	3/1/2012 9:52:46 AM 696	0.000	UDP	67	DHCP Client	192.168.1.1		[02 01 06 00 00 00]DHCP
14	3/1/2012 9:52:46 AM 234	0.000	UDP	67	DHCP	comp15		DHCP: Boot Request
13	3/1/2012 9:52:41 AM 734	0.000	UDP	138	NET Datagram ...	CIVILDEPT		NET Datagram Packet: id:1
12	3/1/2012 9:52:36 AM 750	0.000	UDP	138	NET Datagram ...	BTSP-PC		NET Datagram Packet: id:1
11	3/1/2012 9:52:31 AM 676	0.000	UDP	67	DHCP	BTSP-PC		DHCP: Boot Request
10	3/1/2012 9:52:28 AM 553	0.000	TCP	80	335	COMP2	IS view script s...	PROFFEND / JaronLP H
9	3/1/2012 9:52:23 AM 696	0.016	TCP	80	335	COMP2	IS view script s...	OPTIONS / HTTP1.1 DE

Server: Domain - Value: 15. Ports: 44184

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 of Administrator in Windows. The word kit denotes programs that allow someone to obtain root/admin-level access to the computer by executing the programs in the kit — all of which is done without end-user consent or knowledge.

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

Rootkits have two primary functions: remote command/control (back door) and software eavesdropping. Rootkits allow someone, legitimate or otherwise, to administratively control a computer. This means executing files, accessing logs, monitoring user activity, and even changing the computer's configuration. Therefore, in the strictest sense, even versions of VNC are rootkits. This surprises most people, as they consider rootkits to be solely malware, but in of themselves they aren't malicious at all.

The presence of a rootkit on a network was first documented in the early 1990s. At that time, Sun and Linux operating systems were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems, including Windows, and are increasingly difficult to detect on any network.

PROCEDURE:

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

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

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

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

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

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

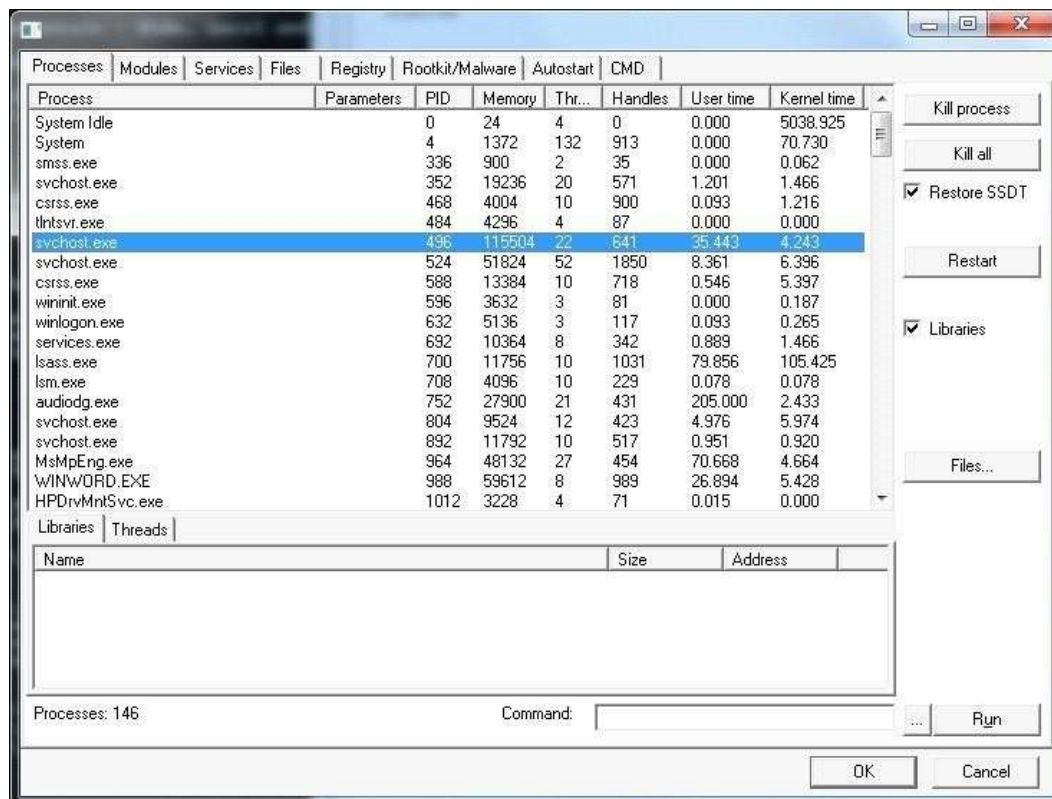
STEP-7: Registry displays Hkey_Current_user and Hkey_Local_Machine.

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

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

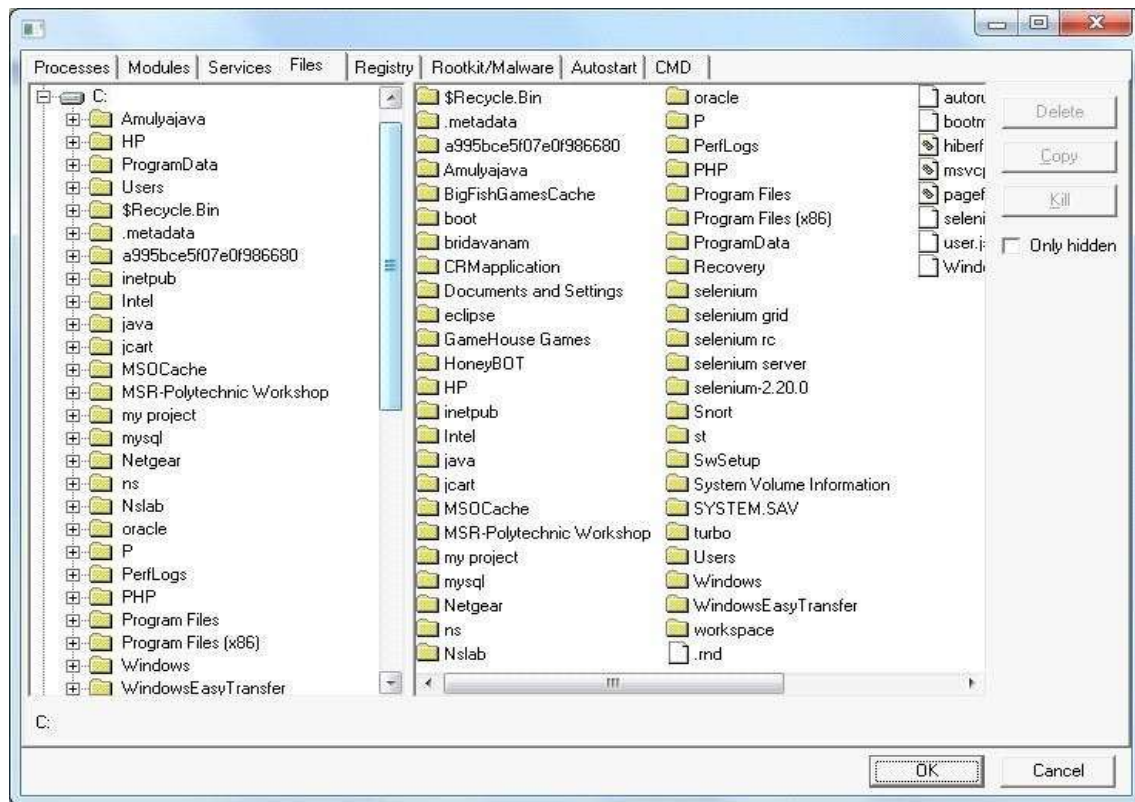
STEP-10: CMD allows the user to interact with command line utilities or Registry

SCREENSHOTS:



Processes Modules Services Files Registry Rootkit/Malware Autostart CMD					
Name	File	Address	Size		
ntoskrnl.exe	\SystemRoot\system32\ntoskrnl.exe	0050C000	6193152		
hal.dll	\SystemRoot\system32\hal.dll	00013000	299008		
kdcom.dll	\SystemRoot\system32\kdcom.dll	000CC000	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		
Wdft000.sys	\SystemRoot\system32\drivers\Wdft000.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		
misadrv.sys	\SystemRoot\system32\drivers\misadrv.sys	00FB6000	40960		
pci.sys	\SystemRoot\system32\drivers\pci.sys	00FC0000	208896		
vdroot.sys	\SystemRoot\system32\drivers\vdroot.sys	00FF3000	53248		
partmgr.sys	\SystemRoot\system32\drivers\partmgr.sys	00E00000	86016		
compbatt.sys	\SystemRoot\system32\DRIVERS\compbatt.sys	00E15000	36864		
BATTIC.SYS	\SystemRoot\system32\DRIVERS\BATTIC.SYS	00E1E000	48152		
volumgr.sys	\SystemRoot\system32\drivers\volumgr.sys	00E2A000	86016		
volumgrx.sys	\SystemRoot\system32\drivers\volumgrx.sys	00E3F000	376832		
mountmgr.sys	\SystemRoot\system32\drivers\mountmgr.sys	00D61000	106496		
iaStor.sys	\SystemRoot\system32\DRIVERS\iaStor.sys	0103A000	2130112		
atap.sys	\SystemRoot\system32\drivers\atap.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		
amdata.sys	\SystemRoot\system32\drivers\amdata.sys	01292000	45056		
ltmgr.sys	\SystemRoot\system32\drivers\ltmgr.sys	0129D000	311296		
fileinfo.sys	\SystemRoot\system32\drivers\fileinfo.sys	012E9000	61920		
Nifs.sys	\SystemRoot\system32\drivers\Nifs.sys	0142D000	1716224		
marpc.sys	\SystemRoot\system32\drivers\marpc.sys	012FD000	385024		
kseodd.sys	\SystemRoot\system32\drivers\kseodd.sys	015D0000	110592		
...		

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\agrsbm64.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	
alide	MANUAL	\SystemRoot\system32\drivers\alide.sys		
amdide	MANUAL	\SystemRoot\system32\drivers\amdide.sys		
AmdK8	MANUAL	\SystemRoot\system32\DRIVERS\amdK8.sys	AMD K8 Processor Driver	
AmdPPM	MANUAL	\SystemRoot\system32\DRIVERS\amdpmp.sys	AMD Processor Driver	
amdsata	MANUAL	\SystemRoot\system32\drivers\amdsata.sys		
amdsbs	MANUAL	\SystemRoot\system32\DRIVERS\amdsbs.sys		
amdxta	BOOT	system32\drivers\amdxta.sys		
AppHostSvc	AUTO	%windir%\system32\svchost.exe -k apphost	@%windir%\system32\inetsrv\iisres.dll,-30012	
AppID	MANUAL	\SystemRoot\system32\drivers\appid.sys	@%systemroot%\system32\appid.sys.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\appidinfo.dll,-101	
AppMgmt	MANUAL	%SystemRoot%\system32\svchost.exe -k netsvcs	@appmgmts.dll,-3251	
...	



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.

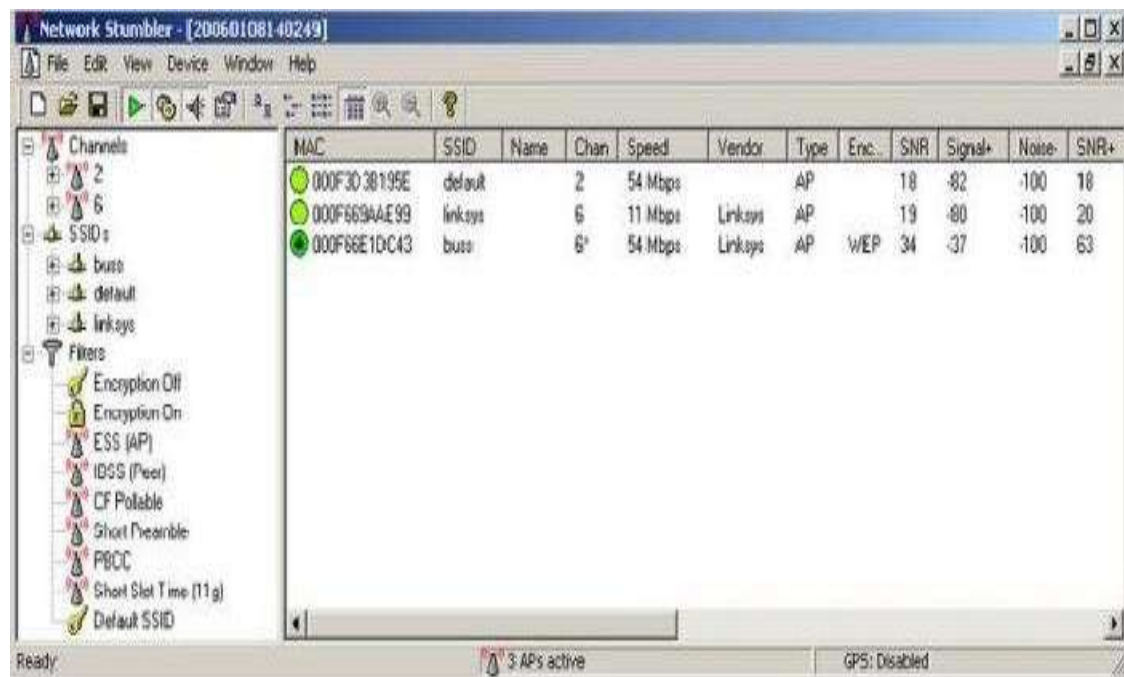
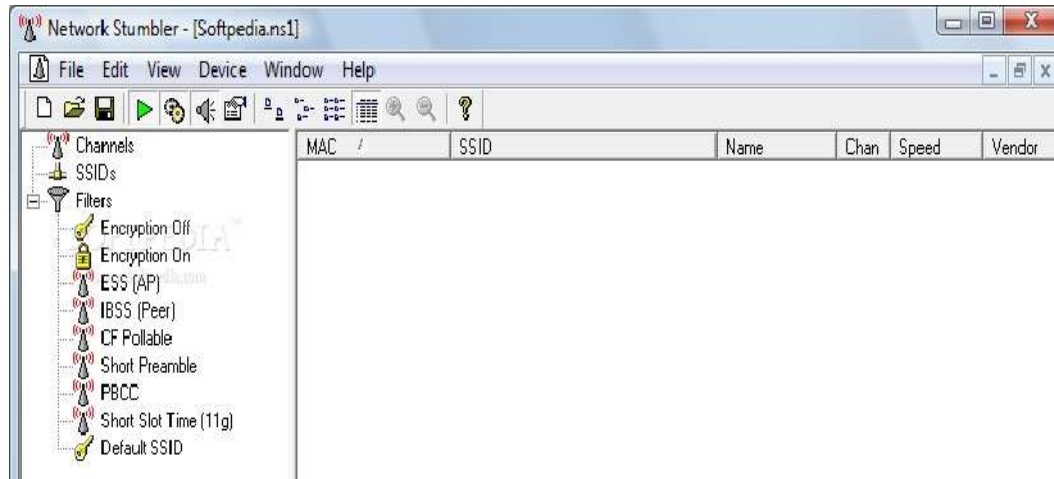
STEP-6: MAC assigned to Wireless Access Point is displayed on right hand pane.

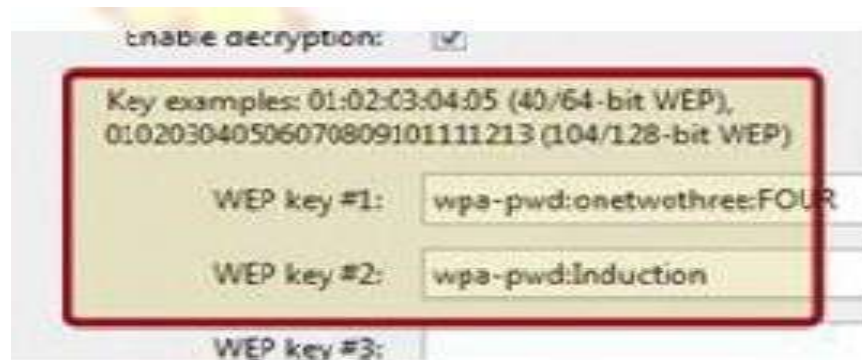
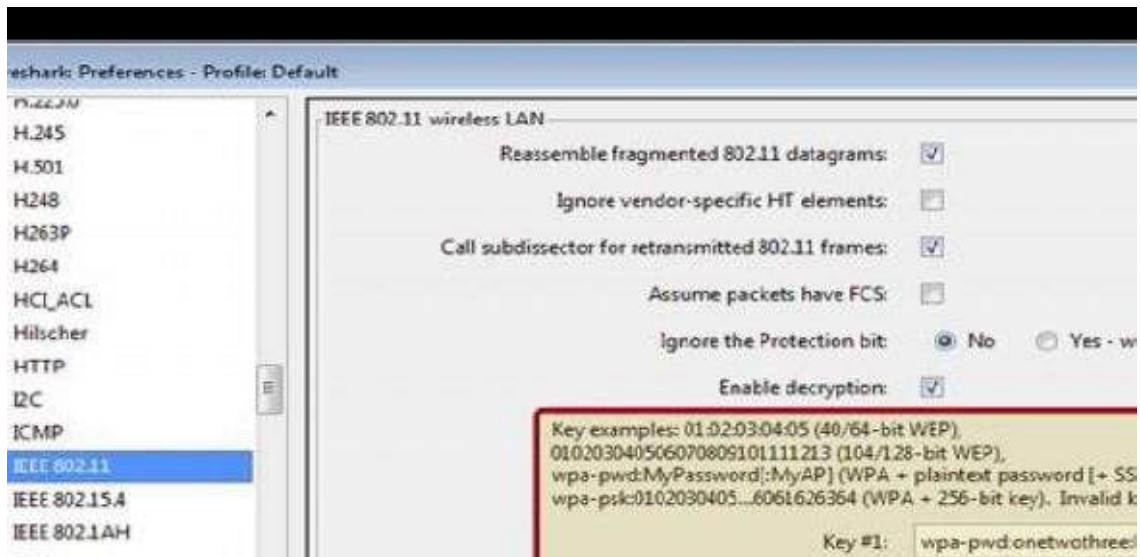
STEP-7: The next column displays the Access points Service Set Identifier[SSID] which is useful to crack the password.

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/IP traffic sniffers and analyzers. Through protocol analysis and content searching and matching, Snort detects attack methods, including denial of service, buffer overflow, CGI attacks, stealth port scans, and SMB probes. When suspicious behavior is detected, Snort sends a real-time alert to syslog, a separate 'alerts' file, or to a pop-up window.

Snort is currently the most popular free network intrusion detection software. The advantages of Snort are numerous. According to the snort web site, "It can perform protocol

analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflow, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more” (Caswell).

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

SNORT can be configured to run in three modes:

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 if an action based upon the rule type in the file.
- ✓ **Snort -d -h ipaddress/24 -l c:\log -c snort.conf** This will configure snort to run in its most basic NIDS form, logging packets that trigger rules specified 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.

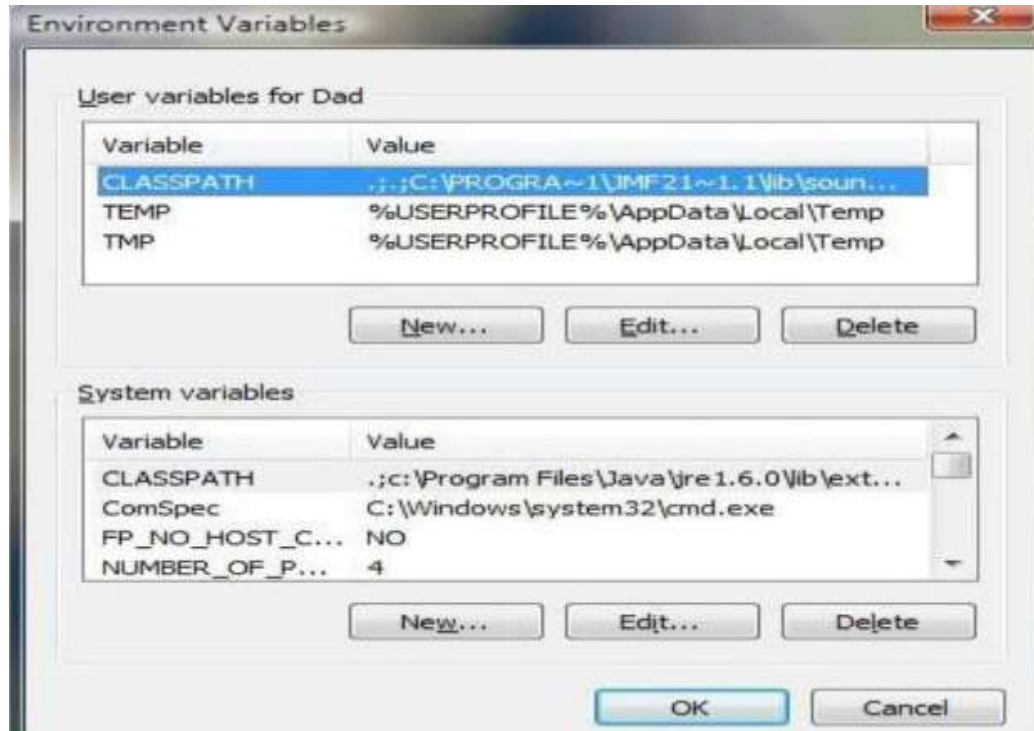
STEP-10: Skip the WinPcap driver installation.

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
=====
```



```
Administrator: C:\Windows\system32\cmd.exe
=====
Run time for packet processing was 703.909000 seconds
Snort processed 1409 packets.
Snort ran for 0 days 0 hours 11 minutes 43 seconds
  Pkts/min:      128
  Pkts/sec:       2
=====
Packet I/O Totals:
  Received:      1411
  Analyzed:      1409 ( 99.858%)
  Dropped:       0 ( 0.000%)
  Filtered:      0 ( 0.000%)
  Outstanding:   2 ( 0.142%)
  Injected:      0
=====
Breakdown by protocol (includes rebuilt packets):
  Eth:           1409 (100.000%)
  VLAN:          0 ( 0.000%)
  IP4:           927 ( 65.791%)
  Frag:          0 ( 0.000%)
  ICMP:          0 ( 0.000%)
  UDP:           892 ( 63.307%)
  TCP:           0 ( 0.000%)
  IP6:           473 ( 33.570%)
  IP6 Ext:       0 ( 0.000%)
  IP6 Opts:      0 ( 0.000%)
  Frag6:         0 ( 0.000%)
  ICMP6:         0 ( 0.000%)
  UDP6:          0 ( 0.000%)
  TCP6:          0 ( 0.000%)
  Teredo:        0 ( 0.000%)
  ICMP-IP:       0 ( 0.000%)
  EAPOL:         0 ( 0.000%)
  IP4/IP4:       0 ( 0.000%)
  IP4/IP6:       0 ( 0.000%)
  IP6/IP4:       0 ( 0.000%)
  IP6/IP6:       0 ( 0.000%)
  GRE:           0 ( 0.000%)
  GRE Eth:       0 ( 0.000%)
  GRE VLAN:      0 ( 0.000%)
  GRE IP4:       0 ( 0.000%)
  GRE IP6:       0 ( 0.000%)
  GRE IP6 Ext:   0 ( 0.000%)
  GRE PPTP:      0 ( 0.000%)
  GRE ARP:       0 ( 0.000%)
  GRE IPX:       0 ( 0.000%)
  GRE Loop:      0 ( 0.000%)
  MPLS:          0 ( 0.000%)
  ARP:           7 ( 0.639%)
  IPX:           0 ( 0.000%)
  Eth Loop:      0 ( 0.000%)
  Eth Disc:      0 ( 0.000%)
  IP4 Disc:      0 ( 0.000%)
  IP6 Disc:      0 ( 0.000%)
  TCP Disc:      0 ( 0.000%)
  UDP Disc:      0 ( 0.000%)
  ICMP Disc:     0 ( 0.000%)
  All Discard:   0 ( 0.000%)
  Other:         35 ( 2.484%)
  Bad Chk Sum:   0 ( 0.000%)
  Bad TTL:       0 ( 0.000%)
  $5 G 1:        0 ( 0.000%)
  $5 G 2:        0 ( 0.000%)
  Total:         1409
=====
Snort exiting
C:\Snort\bin>
```

RESULT:

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