ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

B.E (INFORMATION TECHNOLOGY)

VII – SEMESTER

ITCP 706 – NETWORK SECURITY LABORATORY

Name : ________________________________

Reg. No. : ________________________________

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

B.E (INFORMATION TECHNOLOGY)

VII-SEMESTER

ITCP 706 – NETWORK SECURITY LABORATORY

Certified that this is the bonafide record of the work done by Mr./ Ms.____

Reg.No.______________________________________of VII-Semester of B.E (Information Technology) in Network Security Laboratory during the odd semester of the academic year 2023– 2024.

Place: Annamalai Nagar
Date :   /   / 2023

Staff In charge

Internal Examiner **External Examiner**

COURSE OBJECTIVES

The student should be made to:

- Get exposure to the different cipher techniques
- Learn to generate digital signature
- Learn to use network security tools.
- Study Intrusion Detection System
- Understand security threats in wireless network

LIST OF EXERCISES

1. Implement the following substitution & transposition techniques:
    a. Caesar Cipher
    b. Playfair Cipher
    c. Hill Cipher
    d. Vigenere Cipher
    e. Rail fence-row & Column Transformation
2. Implement the following algorithms
    a. DES
    b. RSA Algorithm
    c. Diffie-Hellman Algorithm
    d. MD5
    e. SHA–1
3. Implement the SIGNATURE SCHEME-Digital Signature Standard
4. Demonstrate how to provide secure data storage, secure data transmission andfor 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 others/w)

COURSE OUTCOMES:

At the end of this course, the students will be able to

1. Implement the Cryptographic algorithms
2. Apply message Authentication Codes and digital Signature Techniques
3. Detect threats in Wireless network

Mapping of Course Outcomes (COs) with Programme Outcomes (POs) and Program Specific Outcomes(PSOs)															
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	3	1	2	2	1	-	-	2	2	2	3	3	2	2
CO2	3	3	1	2	2	1	-	-	2	2	2	3	3	2	2
CO3	3	3	1	2	2	1	-	-	2	2	2	3	3	2	2

DEPARTMENT OF INFORMATION TECHNOLOGY
INSTITUTIONAL – VISION & MISSION

VISION:

Providing world class quality education with strong ethical values to nurture and develop outstanding professionals fit for globally competitive environment.

MISSION:

- Provide quality technical education with a sound footing on basic engineering principles, technical and managerial skills, and innovative research capabilities.
- Transform the students into outstanding professionals and technocrats with strong ethical values capable of creating, developing and managing global engineering enterprises.
- Develop a Global Knowledge Hub, striving continuously in pursuit of excellence in Education, Research, Entrepreneurship and Technological services to the Industry and Society.
- Inculcate the importance and methodology of life-long learning to move forward with updated knowledge to face the challenges of tomorrow.

DEPARTMENTAL – VISION & MISSION

VISION:

To produce globally competent, quality technocrats, to inculcate values of leadership and research qualities and to play a vital role in the socio – economic progress of the nation.

MISSION:

- To partner with the University community to understand the information technology needs of faculty, staff and students.
- To develop dynamic IT professionals with globally competitive learning experience by providing high class education.
- To involve graduates in understanding need based Research activities and disseminate the knowledge to develop entrepreneur skills.

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

Do's

- ✓ Students must present a valid ID card before entering the computer lab.
- ✓ Remove your shoes / footwears and foot socks before you enter the lab.
- ✓ Every user must make logbook entry while entering the lab and also at the time of exit from the lab.
- ✓ Only use your assigned computer and workstation.
- ✓ If any problem arises, please bring the same to the notice of lab in-charge.
- ✓ Before leaving the lab, users must close all the programs and shutdown the computer purpose.
- ✓ Internet facility is only for educational/study purpose.
- ✓ Check for your personal belongings before you leave the lab.

Don'ts

- ✘ Students are not allowed to use personal pen drives, CDs, DVDs etc., in a lab.
- ✘ Do not hit the keys on the computer too hard.
- ✘ Don't damage, remove, or disconnect any labels, parts, cables or equipment.
- ✘ Do not change the settings on the computer.
- ✘ Users are strictly prohibited from modifying or deleting any important files and install any software or settings in the computer.
- ✘ Personal files are not to be stored on the local drive C. All lab computers are set up to remove any data stored or any programs installed by users.

Violation of any of the above rules may result in disciplinary action and the loss of lab privileges

CONTENTS:

Ex.No : 1 Date :	**Encryption and Decryption Using Ceaser Cipher**

AIM:

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

ALGORITHMS:

1. In Ceaser Cipher each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.
2. For example, with a **left shift of 3**, **D** would be replaced by **A**, **E** would become **B**, and so on.
3. The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, **A = 0, B = 1, Z = 25.**
4. Encryption of a letter x by a shift n can be described mathematically as, ***En(x) = (x + n) mod26***
5. Decryption is performed similarly,

   Dn (x)=(x - n) mod26

PROGRAM:

```
def encypt_func(txt, s):
    result = ""
# transverse the plain txt
    for i in range(len(txt)):
        char = txt[i]
        # encypt_func uppercase characters in plain txt

        if (char.isupper()):
            result += chr((ord(char) + s - 64) % 26 + 65)
        # encypt_func lowercase characters in plain txt
        else:
            result += chr((ord(char) + s - 96) % 26 + 97)
    return result
# check the above function
txt = "CEASER CIPHER EXAMPLE"
s = 4
print("Plain txt : " + txt)
```

```
    print("Shift pattern : " + str(s))
    print("Cipher: " + encypt_func(txt, s))
```

OUTPUT:

```
Simulating Caesar Cipher
----------------------------------

Plain txt : CEASER CIPHER EXAMPLE
Shift pattern : 4
Cipher: HJFXJWsHNUMJWsJCFRUQJ
```

RESULT:

Thus the program for ceaser cipher encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 2 Date :	Playfair Cipher

AIM:

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

ALGORITHM:

1. To encrypt a message, one would break the message into diagrams (groups of2 letters)
2. For example, "HelloWorld" becomes "HE LL OW OR LD".
3. These diagrams will be substituted using the key table.
4. Since encryption requires pairs of letters, messages with an odd number of characters usually append an uncommon letter, such as "X", to complete the final diagram.
5. The two letters of the diagram are considered opposite corners of a rectanglein the key table. To perform the substitution, apply the following 4 rules, inorder, to each pair of letters in the plaintext:

PROGRAM:

```
key=input("Enter  key")
key=key.replace(" ", "")
key=key.upper()
def matrix(x,y,initial):
    return [[initial for i in range(x)] for j in range(y)]

result=list()
for c in key: #storing key
    if c not in result:
        if c=='J':
            result.append('I')
        else:
            result.append(c)
flag=0
for i in range(65,91): #storing other character
    if chr(i) not in result:
        if i==73 and chr(74) not in result:
            result.append("I")
            flag=1
```

```
        elif flag==0 and i==73 or i==74:
            pass
        else:
            result.append(chr(i))
k=0
my_matrix=matrix(5,5,0) #initialize matrix
for i in range(0,5): #making matrix
    for j in range(0,5):
        my_matrix[i][j]=result[k]
        k+=1

def locindex(c): #get location of each character
    loc=list()
    if c=='J':
        c='I'
    for i ,j in enumerate(my_matrix):
        for k,l in enumerate(j):
            if c==l:
                loc.append(i)
                loc.append(k)
                return loc

def encrypt(): #Encryption
    msg=str(input("ENTER MSG:"))
    msg=msg.upper()
    msg=msg.replace(" ", "")
    i=0
    for s in range(0,len(msg)+1,2):
        if s<len(msg)-1:
            if msg[s]==msg[s+1]:
                msg=msg[:s+1]+'X'+msg[s+1:]
    if len(msg)%2!=0:
        msg=msg[:]+'X'
    print("CIPHER TEXT:",end=' ')
    while i<len(msg):
        loc=list()
        loc=locindex(msg[i])
    loc1=list()
        loc1=locindex(msg[i+1])
        if loc[1]==loc1[1]:
```

```
print("{}{}".format(my_matrix[(loc[0]+1)%5][loc[1]],my_matrix[(loc1[0]+1)%5][l
oc1[1]]),end=' ')
        elif loc[0]==loc1[0]:

print("{}{}".format(my_matrix[loc[0]][(loc[1]+1)%5],my_matrix[loc1[0]][(loc1[1]
+1)%5]),end=' ')
        else:
print("{}{}".format(my_matrix[loc[0]][loc1[1]],my_matrix[loc1[0]][loc[1]]),end='
')
        i=i+2

def decrypt(): #decryption
    msg=str(input("ENTER CIPHER TEXT:"))
    msg=msg.upper()
    msg=msg.replace(" ", "")
    print("PLAIN TEXT:",end=' ')
    i=0
    while i<len(msg):
        loc=list()
        loc=locindex(msg[i])
        loc1=list()
        loc1=locindex(msg[i+1])
        if loc[1]==loc1[1]:
            print("{}{}".format(my_matrix[(loc[0]-1)%5][loc[1]],my_matrix[(loc1[0]-
1)%5][loc1[1]]),end=' ')
        elif loc[0]==loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1]-
1)%5],my_matrix[loc1[0]][(loc1[1]-1)%5]),end=' ')
        else:

print("{}{}".format(my_matrix[loc[0]][loc1[1]],my_matrix[loc1[0]][loc[1]]),end='
')
        i=i+2
while(1):
    choice=int(input("\n 1.Encryption \n 2.Decryption: \n 3.EXIT"))
    if choice==1:
        encrypt()
    elif choice==2:
        decrypt()
    elif choice==3:
```

```
        exit()
    else:
        print("Choose correct choice")
```

OUTPUT:

```
Simulating Playfair Cipher
-----------------------------------

Original: Hide the gold in...the TREESTUMP!!!
Encoded: BM OD ZB XD NA BE KU DM UI XM MO UV IF
Decoded: HI DE TH EG OL DI NT HE TR EX ES TU MP
```

RESULT:

Thus the program for playfair cipher encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 3 Date:	HILL CIPHER

AIM:

To implement a program to encrypt and decrypt using the Hill cipher substitutiontechnique

ALGORITHM:

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

PROGRAM:

```
 keyMatrix = [[0] * 3 for i in range(3)]
messageVector = [[0] for i in range(3)]
cipherMatrix = [[0] for i in range(3)]
def getKeyMatrix(key):
 k=0
 for i in range(3):
   for j in range(3):
      keyMatrix[i][j] = ord(key[k]) % 65
 k += 1
def encrypt(messageVector):
 for i in range(3):
   for j in range(1):
   cipherMatrix[i][j] = 0
   for x in range(3):
     cipherMatrix[i][j] += (keyMatrix[i][x] *
   messageVector[x][j])
     cipherMatrix[i][j] = cipherMatrix[i][j] % 26
def HillCipher(message, key):
getKeyMatrix(key)
 for i in range(3):
```

```
        messageVector[i][0] = ord(message[i]) % 65
    encrypt(messageVector)
    CipherText = []
    for i in range(3):
        CipherText.append(chr(cipherMatrix[i][0] + 65))
    print("Ciphertext: ", "".join(CipherText))
def main():
    message = "ACT"
    key = "GYBNQKURP"
    HillCipher(message, key)
if__name__== "__main__":
    main()
```

OUTPUT:

Simulating Hill Cipher

Input Message : SecurityLaboratory
Padded Message :
SECURITYLABORATORY Encrypted
Message : EACSDKLCAEFQDUKSXU
Decrypted Message :
SECURITYLABORATORY

RESULT:

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

Ex. No : 4 Date:	VIGENERE CIPHER

AIM:

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

ALGORITHM:

1. The Vigenere cipher is a method of encrypting alphabetic text by using aseries of different Caesar ciphers based on the letters of a keyword.
2. It is a simple form of *polyalphabetic* substitution.
3. To encrypt, a table of alphabets can be used, termed a Vigenere square, orVigenere table.
4. 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.
5. At different points in the encryption process, the cipher uses a differentalphabet from one of the rows used.
6. The alphabet at each point depends on a repeating keyword.

PROGRAM:

```
def generateKey(string, key):
key = list(key)
if len(string) == len(key):
return(key)
else:
for i in range(len(string) -len(key)):
key.append(key[i % len(key)])
return("" . join(key))

def encryption(string, key):
encrypt_text = []
for i in range(len(string)):
x = (ord(string[i]) +ord(key[i])) % 26
x += ord('A')
encrypt_text.append(chr(x))
return("" . join(encrypt_text))
def decryption(encrypt_text, key):
```

```
orig_text = []
for i in range(len(encrypt_text)):
x = (ord(encrypt_text[i]) -ord(key[i]) + 26) % 26
x += ord('A')
orig_text.append(chr(x))
return("" . join(orig_text))
if__name__== "__main__":
string = input("Enter the message: ")
keyword = input("Enter the keyword: ")
key = generateKey(string, keyword)
encrypt_text = encryption(string,key)
print("Encrypted message:", encrypt_text)
print("Decrypted message:", decryption(encrypt_text, key))
```

OUTPUT:

Simulating Vigenere Cipher

Enter the message:
CODESPEEDYEnter the
keyword: TIME
Encrypted message:
BCVORDWOCMDecrypted
message: CODESPEEDY

Beware the Jabberwock, my son! The jaws that bite, the claws that catch!
WMCEEIKLGRPIFVMEUGXQPWQVIOIAVEYXUEKFKBTALVXTGAF
XYEVKPAGY
BEWARETHEJABBERWOCKMYSONTHEJAWSTHATBITETHECLAWS
TH ATCATCH

RESULT:

Thus the program for vigenere cipher encryption and decryption algorithmhas been implemented and the output verified successfully.

Ex. No : 5 Date:	RAIL FENCE CIPHER TRANSPOSITION TECHNIQUE

AIM:

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

ALGORITHM:

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

PROGRAM:

```
from math import ceil


def encrypt(text):
    encrypted_text = [text[i] for i in range(0, len(text), 2)] + [text[i] for i in
range(1, len(text), 2)]
    return "".join(encrypted_text)


def decrypt(text):
    if len(text) % 2 != 0:
        text += '?'
    i, j = 0, ceil(len(text)/2)
    decrypted_text = ''
    for k in range(0, ceil(len(text)/2)):
        decrypted_text += text[i] + text[j]
        i += 1
        j += 1
    return decrypted_text.strip("?")


if__name__== '__main__':
    plaintext = input("Enter text: ")
    cipher_text = encrypt(plaintext)
```

```
print("Encrypted text: " + cipher_text)
print("Decrypted text: " + decrypt(cipher_text))
```

OUTPUT:

Simulating Vigenere Cipher

Enter the message:
CODESPEEDYEnter the
keyword: TIME
Encrypted message:
BCVORDWOCMDecrypted
message: CODESPEEDY

Beware the Jabberwock, my son! The jaws that bite, the claws that catch!
WMCEEIKLGRPIFVMEUGXQPWQVIOIAVEYXUEKFKBTALVXTGAF
XYEVKPAGY
BEWARETHEJABBERWOCKMYSONTHEJAWSTHATBITETHECLAWS
TH ATCATCH

RESULT:

Thus the program for vigenere cipher encryption and decryption algorithmhas been implemented and the output verified successfully.

Ex. No : 6 Date:	**IMPLEMENTATION OF DES**

AIM:

To implement a program for encryption and decryption using DES Algorithm

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:

```
# Hexadecimal to binary conversion
def hex2bin(s):
 mp = {'0' : "0000",
          '1' : "0001",
          '2' : "0010",
          '3' : "0011",
          '4' : "0100",
          '5' : "0101",
          '6' : "0110",
          '7' : "0111",
          '8' : "1000",
          '9' : "1001",
          'A' : "1010",
          'B' : "1011",
          'C' : "1100",
          'D' : "1101",
```

```
            'E' : "1110",
            'F' : "1111" }
    bin = ""
    for i in range(len(s)):
            bin = bin + mp[s[i]]
    return bin

# Binary to hexadecimal conversion
def bin2hex(s):
    mp = {"0000" : '0',
            "0001" : '1',
            "0010" : '2',
            "0011" : '3',
            "0100" : '4',
            "0101" : '5',
            "0110" : '6',
            "0111" : '7',
            "1000" : '8',
            "1001" : '9',
            "1010" : 'A',
            "1011" : 'B',
            "1100" : 'C',
            "1101" : 'D',
            "1110" : 'E',
            "1111" : 'F' }
    hex = ""
    for i in range(0,len(s),4):
            ch = ""
            ch = ch + s[i]
            ch = ch + s[i + 1]
            ch = ch + s[i + 2]
            ch = ch + s[i + 3]
            hex = hex + mp[ch]

    return hex

# Binary to decimal conversion
def bin2dec(binary):

    binary1 = binary
    decimal, i, n = 0, 0, 0
    while(binary != 0):
            dec = binary % 10
```

```
            decimal = decimal + dec * pow(2, i)
            binary = binary//10
            i += 1
    return decimal

# Decimal to binary conversion
def dec2bin(num):
    res = bin(num).replace("0b", "")
    if(len(res)%4 != 0):
            div = len(res) / 4
            div = int(div)
            counter =(4 * (div + 1)) - len(res)
            for i in range(0, counter):
                    res = '0' + res
    return res

# Permute function to rearrange the bits
def permute(k, arr, n):
    permutation = ""
    for i in range(0, n):
            permutation = permutation + k[arr[i] - 1]
    return permutation

# shifting the bits towards left by nth shifts
def shift_left(k, nth_shifts):
    s = ""
    for i in range(nth_shifts):
            for j in range(1,len(k)):
                    s = s + k[j]
            s = s + k[0]
            k = s
            s = ""
    return k

# calculating xow of two strings of binary number a and b
def xor(a, b):
    ans = ""
    for i in range(len(a)):
            if a[i] == b[i]:
                    ans = ans + "0"
            else:
                    ans = ans + "1"
    return ans
```

```
# Table of Position of 64 bits at initial level: Initial Permutation Table
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]

# Expansion D-box Table
exp_d = [32, 1 , 2 , 3 , 4 , 5 , 4 , 5,
         6 , 7 , 8 , 9 , 8 , 9 , 10, 11,
         12, 13, 12, 13, 14, 15, 16, 17,
         16, 17, 18, 19, 20, 21, 20, 21,
         22, 23, 24, 25, 24, 25, 26, 27,
         28, 29, 28, 29, 30, 31, 32, 1 ]

# Straight Permutation Table
per = [ 16, 7, 20, 21,
        29, 12, 28, 17,
        1, 15, 23, 26,
        5, 18, 31, 10,
        2, 8, 24, 14,
        32, 27, 3, 9,
        19, 13, 30, 6,
        22, 11, 4, 25 ]

# S-box Table
sbox = [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
         [ 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
         [ 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
         [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 ]],

         [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
            [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
            [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
         [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 ]],

         [ [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
         [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
         [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
```

```
            [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 ]],

        [ [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
        [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
        [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
            [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14] ],

        [ [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
        [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
            [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
        [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 ]],

        [ [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
        [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
            [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
            [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13] ],

        [ [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
        [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
            [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
            [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12] ],

        [ [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
            [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
            [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
            [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11] ] ]

# Final Permutation Table
final_perm = [ 40, 8, 48, 16, 56, 24, 64, 32,
               39, 7, 47, 15, 55, 23, 63, 31,
               38, 6, 46, 14, 54, 22, 62, 30,
               37, 5, 45, 13, 53, 21, 61, 29,
               36, 4, 44, 12, 52, 20, 60, 28,
               35, 3, 43, 11, 51, 19, 59, 27,
               34, 2, 42, 10, 50, 18, 58, 26,
               33, 1, 41, 9, 49, 17, 57, 25 ]

def encrypt(pt, rkb, rk):
 pt = hex2bin(pt)

 # Initial Permutation
 pt = permute(pt, initial_perm, 64)
 print("After initial permutation", bin2hex(pt))
```

```
    # Splitting
    left = pt[0:32]
    right = pt[32:64]
    for i in range(0, 16):
        # Expansion D-box: Expanding the 32 bits data into 48 bits
        right_expanded = permute(right, exp_d, 48)

        # XOR RoundKey[i] and right_expanded
        xor_x = xor(right_expanded, rkb[i])

        # S-boxex: substituting the value from s-box table by calculating
row and column
        sbox_str = ""
        for j in range(0, 8):
            row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
            col = bin2dec(int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] +
xor_x[j * 6 + 3] + xor_x[j * 6 + 4]))
            val = sbox[j][row][col]
            sbox_str = sbox_str + dec2bin(val)

        # Straight D-box: After substituting rearranging the bits
        sbox_str = permute(sbox_str, per, 32)

        # XOR left and sbox_str
        result = xor(left, sbox_str)
        left = result

        # Swapper
        if(i != 15):
            left, right = right, left
        print("Round ", i + 1, " ", bin2hex(left), " ", bin2hex(right), " ",
rk[i])

    # Combination
    combine = left + right

    # Final permutation: final rearranging of bits to get cipher text
    cipher_text = permute(combine, final_perm, 64)
    return cipher_text

pt = "123456ABCD132536"
key = "AABB09182736CCDD"
```

```
# Key generation
# --hex to binary
key = hex2bin(key)

# --parity bit drop table
keyp = [57, 49, 41, 33, 25, 17, 9,
        1, 58, 50, 42, 34, 26, 18,
        10, 2, 59, 51, 43, 35, 27,
        19, 11, 3, 60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
        7, 62, 54, 46, 38, 30, 22,
        14, 6, 61, 53, 45, 37, 29,
        21, 13, 5, 28, 20, 12, 4 ]

# getting 56 bit key from 64 bit using the parity bits
key = permute(key, keyp, 56)

# Number of bit shifts
shift_table = [1, 1, 2, 2,
               2, 2, 2, 2,
               1, 2, 2, 2,
               2, 2, 2, 1 ]

# Key- Compression Table : Compression of key from 56 bits to 48 bits
key_comp = [14, 17, 11, 24, 1, 5,
            3, 28, 15, 6, 21, 10,
            23, 19, 12, 4, 26, 8,
            16, 7, 27, 20, 13, 2,
            41, 52, 31, 37, 47, 55,
            30, 40, 51, 45, 33, 48,
            44, 49, 39, 56, 34, 53,
            46, 42, 50, 36, 29, 32 ]

# Splitting
left = key[0:28] # rkb for RoundKeys in binary
right = key[28:56] # rk for RoundKeys in hexadecimal
rkb = []
rk = []
for i in range(0, 16):
 # Shifting the bits by nth shifts by checking from shift table
 left = shift_left(left, shift_table[i])
 right = shift_left(right, shift_table[i])
```

```
        # Combination of left and right string
        combine_str = left + right

        # Compression of key from 56 to 48 bits
        round_key = permute(combine_str, key_comp, 48)

        rkb.append(round_key)
        rk.append(bin2hex(round_key))

print("Encryption")
cipher_text = bin2hex(encrypt(pt, rkb, rk))
print("Cipher Text : ",cipher_text)

print("Decryption")
rkb_rev = rkb[::-1]
rk_rev = rk[::-1]
text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))
print("Plain Text : ",text)
```

OUTPUT:

```
Simulating DES Algorithm
_________________________-
  Encryption
After  initial permutation  14A7D67818CA18AD
Round  1   18CA18AD   5A78E394 194CD072DE8C
Round  2   5A78E394   4A1210F6   4568581ABCCE
Round  3   4A1210F6   B8089591   06EDA4ACF5B5
Round  4   B8089591   236779C2   DA2D032B6EE3
Round  5   236779C2   A15A4B87   69A629FEC913
Round  6   A15A4B87   2E8F9C65   C1948E87475E
Round  7   2E8F9C65   A9FC20A3   708AD2DDB3C0
Round  8   A9FC20A3   308BEE97   34F822F0C66D
Round  9   308BEE97   10AF9D37   84BB4473DCCC
Round  10   10AF9D37   6CA6CB20   02765708B5BF
Round  11   6CA6CB20   FF3C485F   6D5560AF7CA5
Round  12   FF3C485F   22A5963B   C2C1E96A4BF3
Round  13   22A5963B  387CCDAA 99C31397C91F
Round  14   387CCDAA   BD2DD2AB   251B8BC717D0
Round  15   BD2DD2AB   CF26B472   3330C5D9A36D
Round  16   19BA9212   CF26B472   181C5D75C66D
Cipher Text : C0B7A8D05F3A829C
```

Decryption
After initial permutation 19BA9212CF26B472
Round 1 CF26B472 BD2DD2AB 181C5D75C66D
Round 2 BD2DD2AB 387CCDAA 3330C5D9A36D
Round 3 387CCDAA 22A5963B 251B8BC717D0
Round 4 22A5963B FF3C485F 99C31397C91F
Round 5 FF3C485F 6CA6CB20 C2C1E96A4BF3
Round 6 6CA6CB20 10AF9D37 6D5560AF7CA5
Round 7 10AF9D37 308BEE97 02765708B5BF
Round 8 308BEE97 A9FC20A3 84BB4473DCCC
Round 9 A9FC20A3 2E8F9C65 34F822F0C66D
Round 10 2E8F9C65 A15A4B87 708AD2DDB3C0
Round 11 A15A4B87 236779C2 C1948E87475E
Round 12 236779C2 B8089591 69A629FEC913
Round 13 B8089591 4A1210F6 DA2D032B6EE3
Round 14 4A1210F6 5A78E394 06EDA4ACF5B5
Round 15 5A78E394 18CA18AD 4568581ABCCE
Round 16 14A7D678 18CA18AD 194CD072DE8C
Plain Text : 123456ABCD132536

RESULT:

Thus the program for DES encryption and decryption algorithmhas been implemented and the output verified successfully.

Ex. No : 7 Date:	**IMPLEMENTATION OF RSA**

AIM:

To implement a program for encryption and decryption using DES Algorithm

ALGORITHM:

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

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

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

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

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

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

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

PROGRAM:

```
import math

print("RSA ENCRYPTOR/DECRYPTOR")
print("*****************************************************")

#Input Prime Numbers
print("PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:")
p = int(input("Enter a prime number for p: "))
q = int(input("Enter a prime number for q: "))
print("*****************************************************")

#Check if Input's are Prime
'''THIS FUNCTION AND THE CODE IMMEDIATELY BELOW THE
FUNCTION CHECKS WHETHER THE INPUTS ARE PRIME OR NOT.'''
def prime_check(a):
    if(a==2):
        return True
    elif((a<2) or ((a%2)==0)):
        return False
    elif(a>2):
        for i in range(2,a):
```

```
        if not(a%i):
            return false
    return True

check_p = prime_check(p)
check_q = prime_check(q)
while(((check_p==False)or(check_q==False))):
    p = int(input("Enter a prime number for p: "))
    q = int(input("Enter a prime number for q: "))
    check_p = prime_check(p)
    check_q = prime_check(q)

#RSA Modulus
'''CALCULATION OF RSA MODULUS 'n'.'''
n = p * q
print("RSA Modulus(n) is:",n)

#Eulers Toitent
'''CALCULATION OF EULERS TOITENT 'r'.'''
r= (p-1)*(q-1)
print("Eulers Toitent(r) is:",r)
print("*****************************************************")

#GCD
'''CALCULATION OF GCD FOR 'e' CALCULATION.'''
def egcd(e,r):
    while(r!=0):
        e,r=r,e%r
    return e

#Euclid's Algorithm
def eugcd(e,r):
    for i in range(1,r):
        while(e!=0):
            a,b=r//e,r%e
            if(b!=0):
                print("%d = %d*(%d) + %d"%(r,a,e,b))
            r=e
            e=b

#Extended Euclidean Algorithm
def eea(a,b):
```

```
    if(a%b==0):
        return(b,0,1)
    else:
        gcd,s,t = eea(b,a%b)
        s = s-((a//b) * t)
        print("%d = %d*(%d) + (%d)*(%d)"%(gcd,a,t,s,b))
        return(gcd,t,s)

#Multiplicative Inverse
def mult_inv(e,r):
    gcd,s,_=eea(e,r)
    if(gcd!=1):
        return None
    else:
        if(s<0):
            print("s=%d. Since %d is less than 0, s = s(modr), i.e.,
s=%d."%(s,s,s%r))
        elif(s>0):
            print("s=%d."%(s))
        return s%r

#e Value Calculation
'''FINDS THE HIGHEST POSSIBLE VALUE OF 'e' BETWEEN 1 and 1000
THAT MAKES (e,r) COPRIME.'''
for i in range(1,1000):
    if(egcd(i,r)==1):
        e=i
print("The value of e is:",e)
print("*****************************************************")

#d, Private and Public Keys
'''CALCULATION OF 'd', PRIVATE KEY, AND PUBLIC KEY.'''
print("EUCLID'S ALGORITHM:")
eugcd(e,r)
print("END OF THE STEPS USED TO ACHIEVE EUCLID'S
ALGORITHM.")
print("*****************************************************")
print("EUCLID'S EXTENDED ALGORITHM:")
d = mult_inv(e,r)
print("END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.")
print("The value of d is:",d)
print("*****************************************************")
```

```
public = (e,n)
private = (d,n)
print("Private Key is:",private)
print("Public Key is:",public)
print("****************************************************")

#Encryption
'''ENCRYPTION ALGORITHM.'''
def encrypt(pub_key,n_text):
    e,n=pub_key
    x=[]
    m=0
    for i in n_text:
        if(i.isupper()):
            m = ord(i)-65
            c=(m**e)%n
            x.append(c)
        elif(i.islower()):
            m=  ord(i)-97
            c=(m**e)%n
            x.append(c)
        elif(i.isspace()):
            spc=400
            x.append(400)
    return x


#Decryption
'''DECRYPTION ALGORITHM'''
def decrypt(priv_key,c_text):
    d,n=priv_key
    txt=c_text.split(',')
    x=''
    m=0
    for i in txt:
        if(i=='400'):
            x+=' '
        else:
            m=(int(i)**d)%n
            m+=65
            c=chr(m)
            x+=c
```

```
    return x

#Message
message = input("What would you like encrypted or decrypted?(Separate
numbers with ',' for decryption):")
print("Your message is:",message)

#Choose Encrypt or Decrypt and Print
choose = input("Type '1' for encryption and '2' for decrytion.")
if(choose=='1'):
    enc_msg=encrypt(public,message)
    print("Your encrypted message is:",enc_msg)
    print("Thank you for using the RSA Encryptor. Goodbye!")
elif(choose=='2'):
    print("Your decrypted message is:",decrypt(private,message))
    print("Thank you for using the RSA Encryptor. Goodbye!")
else:
    print("You entered the wrong option.")
    print("Thank you for using the RSA Encryptor. Goodbye!")
```

OUTPUT:

```
Simulating RSA Algorithm
_________________________-

PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:
Enter a prime number for p: 3
Enter a prime number for q: 5
*****************************************************
RSA Modulus(n) is: 15
Eulers Toitent(r) is: 8
*****************************************************
The value of e is: 999
*****************************************************
EUCLID'S ALGORITHM:
8 = 0*(999) + 8
999 = 124*(8) + 7
8 = 1*(7) + 1
END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.
*****************************************************
```

EUCLID'S EXTENDED ALGORITHM:
1 = 8*(1) + (-1)*(7)
1 = 999*(-1) + (125)*(8)
s=-1. Since -1 is less than 0, s = s(modr), i.e., s=7.
END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.
The value of d is: 7
****
Private Key is: (7, 15)
Public Key is: (999, 15)
****
What would you like encrypted or decrypted?(Separate numbers with ',' for decryption):13,4,11,11,14
Your message is: 13,4,11,11,14
Type '1' for encryption and '2' for decrytion.2
Your decrypted message is: HELLO
Thank you for using the RSA Encryptor. Goodbye!

RESULT:

Thus the program for RSA encryption and decryption algorithmhas been implemented and the output verified successfully.

Ex. No : 8 Date:	**IMPLEMENTATION OF DIFFIE-HELLMAN**

AIM:

To implement a program for encryption and decryption using Diffie-Hellman Algorithm

ALGORITHM:

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

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

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

STEP-4: Then Alice sends A to Bob.

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

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

PROGRAM:

```
from secrets import SystemRandom

# pseudo random number generator
prng = SystemRandom()

# g and p values agreed by both Alice and Bob.
g = prng.randint(1, 100)
p = prng.randint(1, 100)
 print(f"Agreed g value:{g}, agreed modulo:{p}\n")
 # Alice's private random number chose between 1 and 100
A = prng.randint(1, 100)
print(f"Alice's random number is {A}.\n")
# Bob's private random number chose between 1 and 100
B = prng.randint(1, 100)
print(f"Bob's random number is {B}.\n")

# public value of Alice's to be sent over to Bob.
```

```
a = g**A % p
print(f"Alice's calculated public value is {a}, which will be sent to Bob
publicly.\n")

# public value of Bob's to be sent over to Alice.
b = g**B % p
print(f"Bob's calculated public value is {b}, which will be sent over to Alice
publicly.\n")

# Alice uses Bob's public value and her private value to compute the secret key.
secret_key1 = b**A % p
 # Bob uses Alice's public value and his private value to compute the secret key.
secret_key2 = a**B % p
 if secret_key1 == secret_key2:
    print("Secret key has been successfully derived! See below...\n")
    print(f"Bob uses Alice's public value which is {a}, and his own private value
which is {B}, "
          f"the secret key is {secret_key2}.\n")
    print(f"Alice uses Bob's public value which is {b}, and her own private
random value which is {A},"
          f"the secret key is {secret_key1}.")
else:
    print("Alice and Bob have different secret keys, which is wrong! Try again!")
```

OUTPUT:

Simulating Diffie-Hellman Algorithm

___________________________-

Alice's random number is 65.
Bob's random number is 79.
Alice's calculated public value is 16, which will be sent to Bob publicly.
Bob's calculated public value is 16, which will be sent over to Alice publicly.
Secret key has been successfully derived! See below...
Bob uses Alice's public value which is 16, and his own private value which is 79, the secret key is 41.
Alice uses Bob's public value which is 16, and her own private random value which is 65, the secret key is 41.

RESULT:

Thus the program for Diffie-Hellman encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 9 Date:	IMPLEMENTATION OF MD5

AIM:

To implement a program for encryption and decryption using MD5 Algorithm

ALGORITHM:

STEP-1: Read the 128-bit plaintext

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:

```
import hashlib
result = hashlib.md5(b"Hello MD5").hexdigest()
print(result)
result = hashlib.md5("Hello MD5".encode("utf-8")).hexdigest()
print(result)
m = hashlib.md5(b"Hello MD5")
print(m.name)
print(m.digest_size) # 16 bytes (128 bits)
print(m.digest())    # bytes
print(m.hexdigest()) # bytes in hex representation
```

OUTPUT:

Simulating MD5 Algorithm

______________________________-

Terminal
e5dadf6524624f79c3127e247f04b548
e5dadf6524624f79c3127e247f04b548
md5
16
b'\xe5\xda\xdfe$bOy\xc3\x12~$\x7f\x04\xb5H'
e5dadf6524624f79c3127e247f04b548

RESULT:

Thus the program for MD5 encryption and decryption algorithm has been implemented and the output verified successfully.

Ex. No : 10 Date:	IMPLEMENTATION OF SHA-1

AIM:

To implement a program for encryption and decryption using SHA-1 Algorithm

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:

```
import hashlib

# initializing string
str = "SHA"

# encoding SHA using encode()
# then sending to SHA256()
result = hashlib.sha256(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA256 is : ")
print(result.hexdigest())

print ("\r")
```

```
# initializing string
str = " SHA "

# encoding SHA using encode()
# then sending to SHA384()
result = hashlib.sha384(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA384 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = " SHA"

# encoding SHA using encode()
# then sending to SHA224()
result = hashlib.sha224(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA224 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = " SHA"

# encoding SHA using encode()
# then sending to SHA512()
result = hashlib.sha512(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = " SHA"
```

```
# encoding SHA using encode()
# then sending to SHA1()
result = hashlib.sha1(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```

OUTPUT:

Simulating SHA Algorithm

-

The hexadecimal equivalent of SHA256 is :

f6071725e7ddeb434fb6b32b8ec4a2b14dd7db0d785347b2fb48f9975126178f

The hexadecimal equivalent of SHA384 is :

d1e67b8819b009ec7929933b6fc1928dd64b5df31bcde6381b9d3f90488d25324
0490460c0a5a1a873da8236c12ef9b3

The hexadecimal equivalent of SHA224 is :

173994f309f727ca939bb185086cd7b36e66141c9e52ba0bdcfd145d

The hexadecimal equivalent of SHA512 is :

0d8fb9370a5bf7b892be4865cdf8b658a82209624e33ed71cae353b0df254a75d
b63d1baa35ad99f26f1b399c31f3c666a7fc67ecef3bdcdb7d60e8ada90b722

The hexadecimal equivalent of SHA1 is :

4175a37afd561152fb60c305d4fa6026b7e79856

RESULT:

Thus the program for SHA1 encryption and decryption algorithm hasbeen implemented and the output verified successfully.

Ex. No : 11 Date:	IMPLEMENTATION OF DIGITAL SIGNATURE STANDARD

AIM:

To implement a program for 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:

```
# Function to find gcd
# of two numbers
def euclid(m, n):
    if n == 0:
        return m
    else:
        r = m % n
        return euclid(n, r)

# Program to find
# Multiplicative inverse
def exteuclid(a, b):
    r1 = a
    r2 = b
```

```
    s1 = int(1)
    s2 = int(0)
    t1 = int(0)
    t2 = int(1)
       while r2 > 0:
       q = r1//r2
        r = r1-q * r2
        r1 = r2
        r2 = r
        s = s1-q * s2
        s1 = s2
        s2 = s
        t = t1-q * t2
        t1 = t2
        t2 = t

    if t1 < 0:
        t1 = t1 % a

    return (r1, t1)

# Enter two large prime
# numbers p and q
p = 823
q = 953
n = p * q
Pn = (p-1)*(q-1)

# Generate encryption key
# in range 1<e<Pn
key = []
for i in range(2, Pn):
    gcd = euclid(Pn, i)
    if gcd == 1:
        key.append(i)
# Select an encryption key
# from the above list
e = int(313)
# Obtain inverse of
# encryption key in Z_Pn
r, d = exteuclid(Pn, e)
if r == 1:
    d = int(d)
```

```
    print("decryption key is: ", d)
else:
    print("Multiplicative inverse for\
    the given encryption key does not \
    exist. Choose a different encryption key ")
# Enter the message to be sent
M = 19070
# Signature is created by Alice
S = (M**d) % n
# Alice sends M and S both to Bob
# Bob generates message M1 using the
# signature S, Alice's public key e
# and product n.
M1 = (S**e) % n
 # If M = M1 only then Bob accepts
# the message sent by Alice.
 if M == M1:
    print("As M = M1, Accept the\
    message sent by Alice")
else:
    print("As M not equal to M1,\
    Do not accept the message\
    sent by Alice ")
```

OUTPUT:

```
decryption key is:  160009
As M = M1, Accept the message sent by Alice
```

RESULT:

Thus the program for Digital signature standard has been implemented and the output verified successfully.

Ex. No : 12 **Date:**	**SECURE DATA, STORAGE, SECURE DATA TRANSMISSION AND FOR CREATING DIGITAL SIGNATURES**

AIM:

Demonstrate how to provide secure data storage, secure data transmission and for creating digital signatures

INTRODUCTION:

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

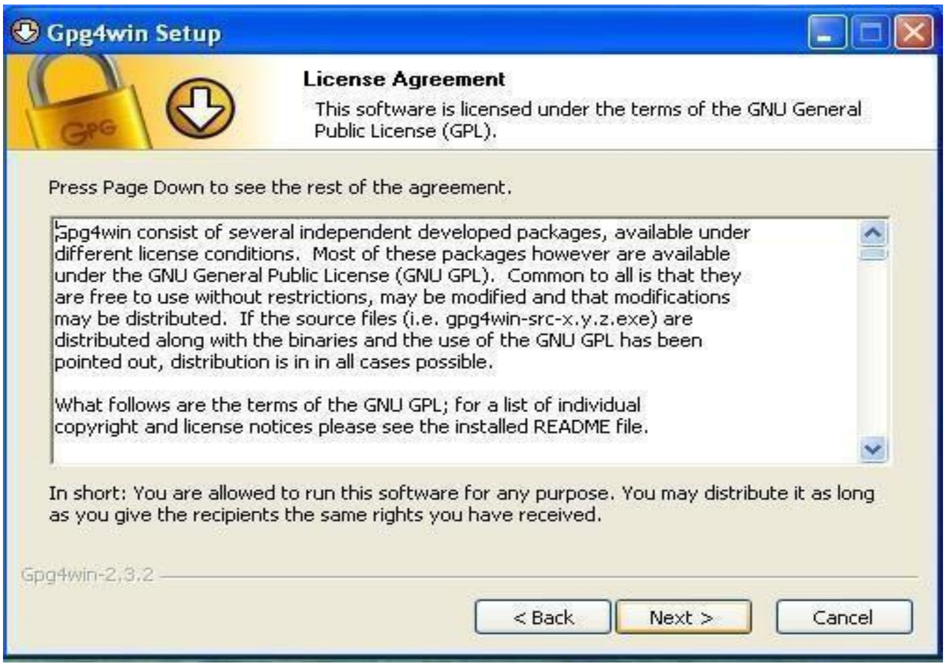
INSTALLING THE SOFTWARE:

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

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

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

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



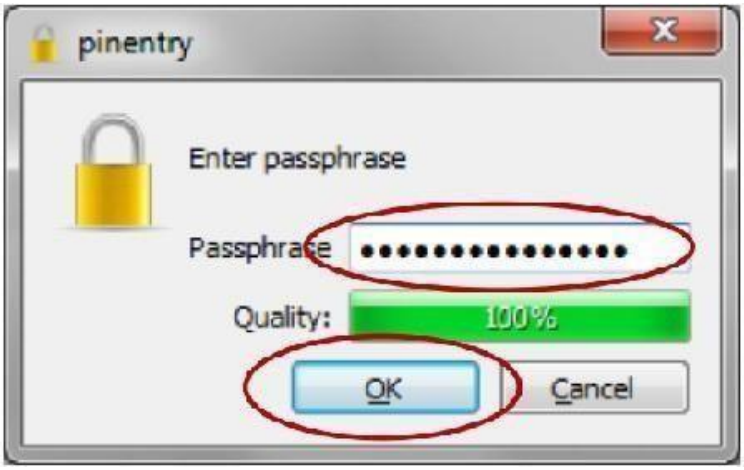
5. Set the check box values as specified below, then click the “Next” button

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

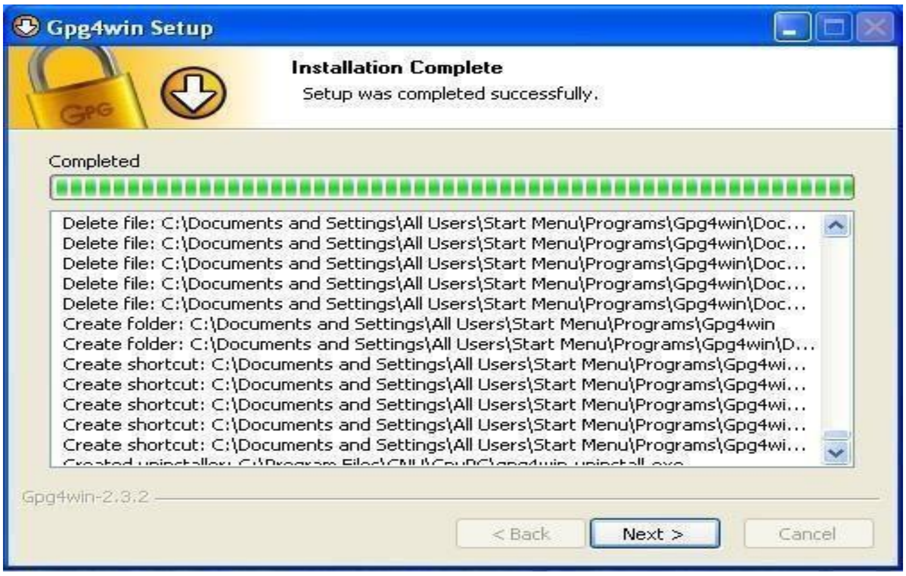
7. Specify where you want shortcuts to the software placed, then click the “Next” button.

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

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

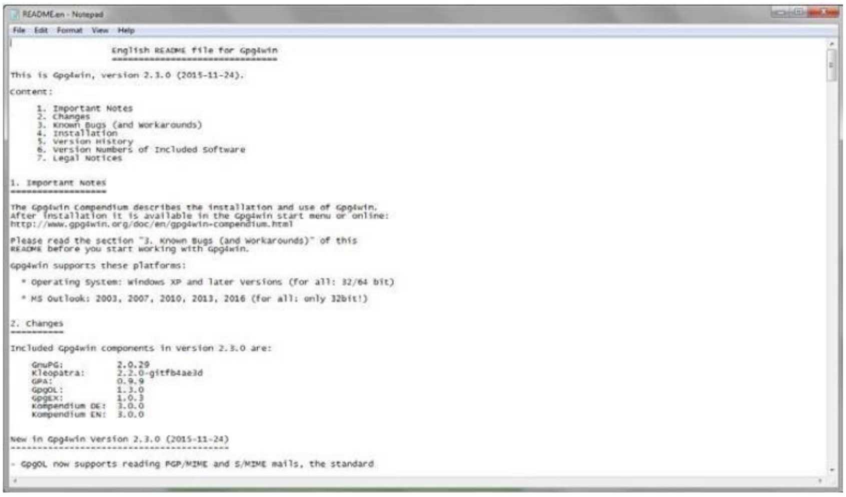


10. The installation process will tell you when it is complete. Click the “Next” button



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

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

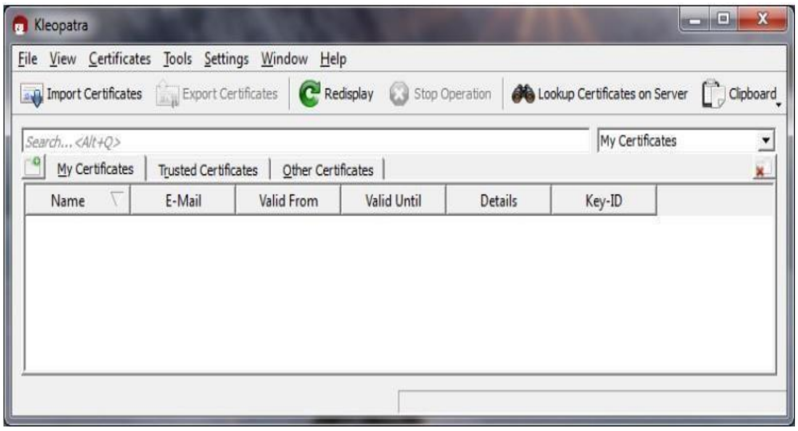


CREATING YOUR PUBLIC AND PRIVATE KEYS

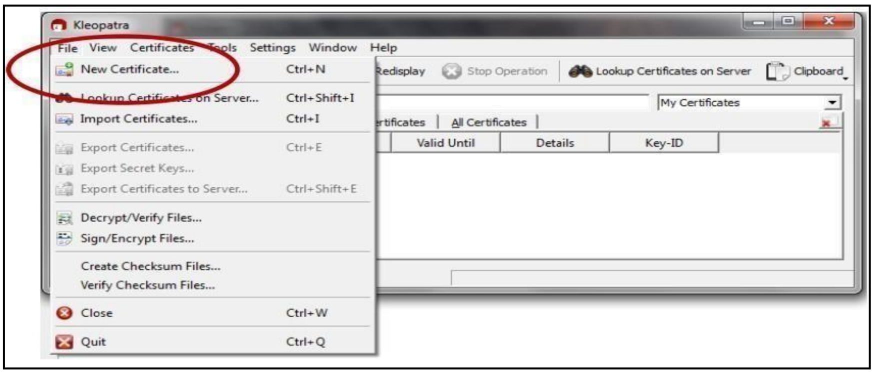
GPG encryption and decryption is based upon the keys of the person who will be receiving the encrypted file or message. Any individual who wants to send the person an encrypted file or message must possess the recipient’s public key certificate to encrypt the message. The recipient must have the associated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

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

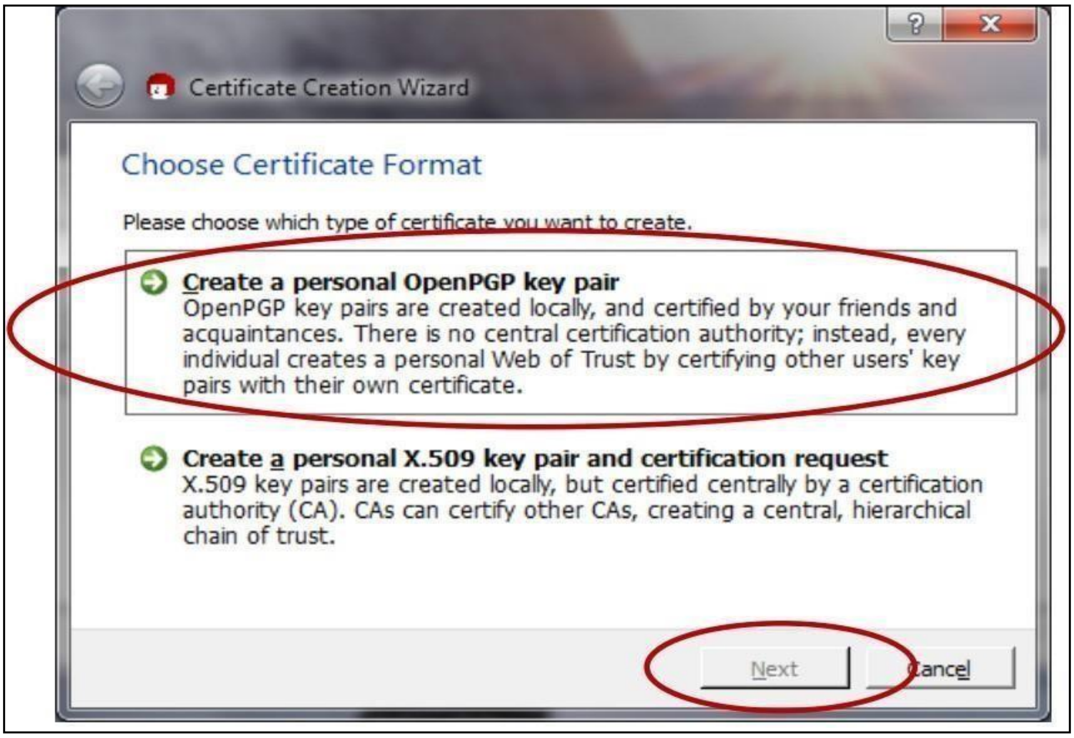
2. The following screen will be displayed



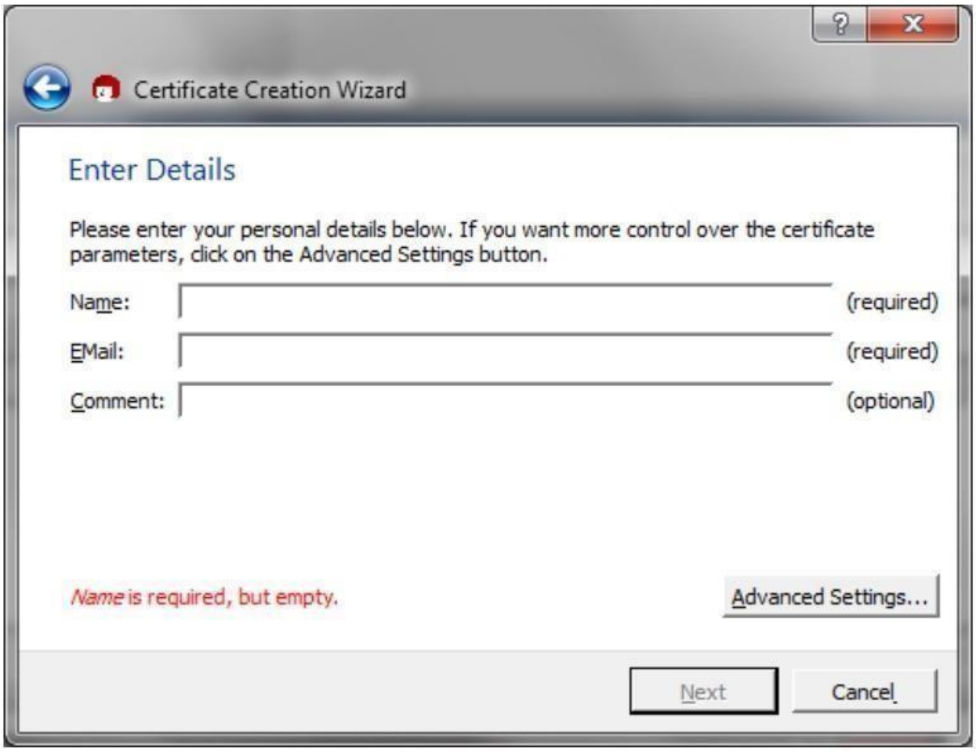
3. From the “File” dropdown, click on the “New Certificate” option



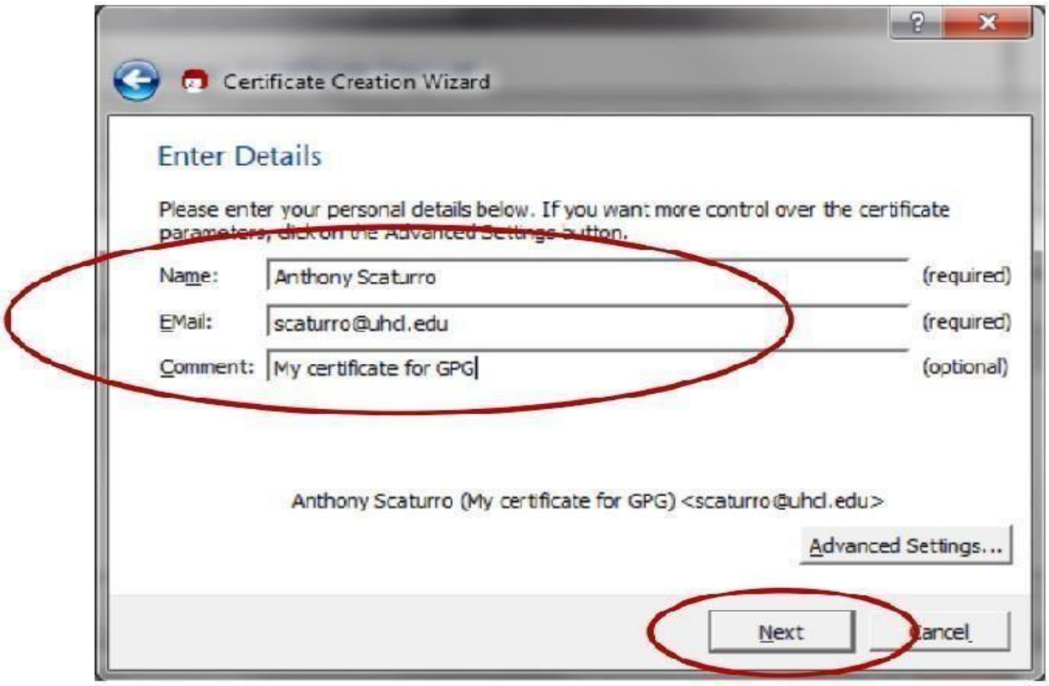
4. The following screen will be displayed. Click on “Create a personal OpenGPG key pair” and the “Next” button



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



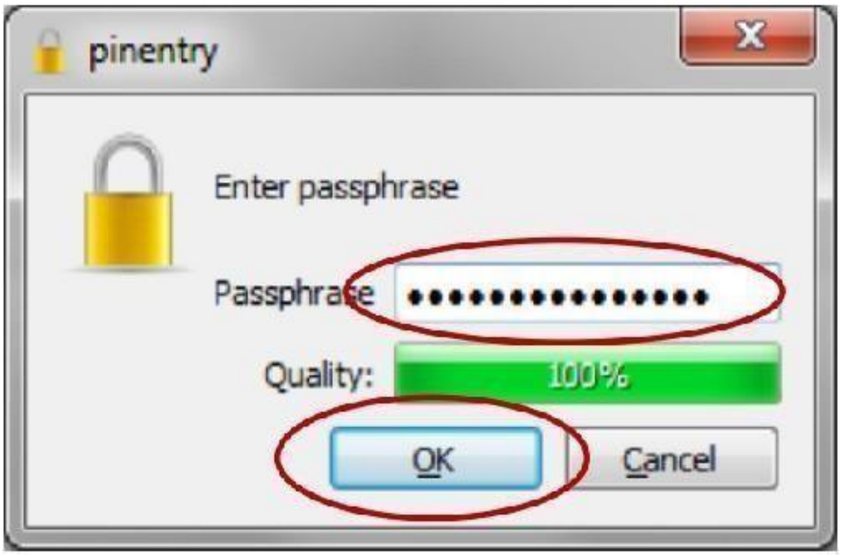
6. Enter your name and e-mail address. You may also enter an optional comment. Then, click the "Next" button



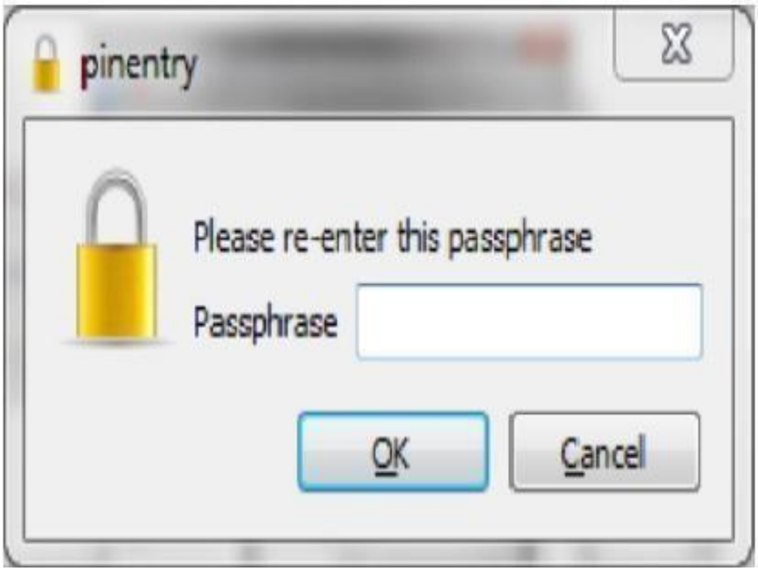
7. Review your entered values. If OK, click the “Create Key” button

8. You will be asked to enter a passphrase

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



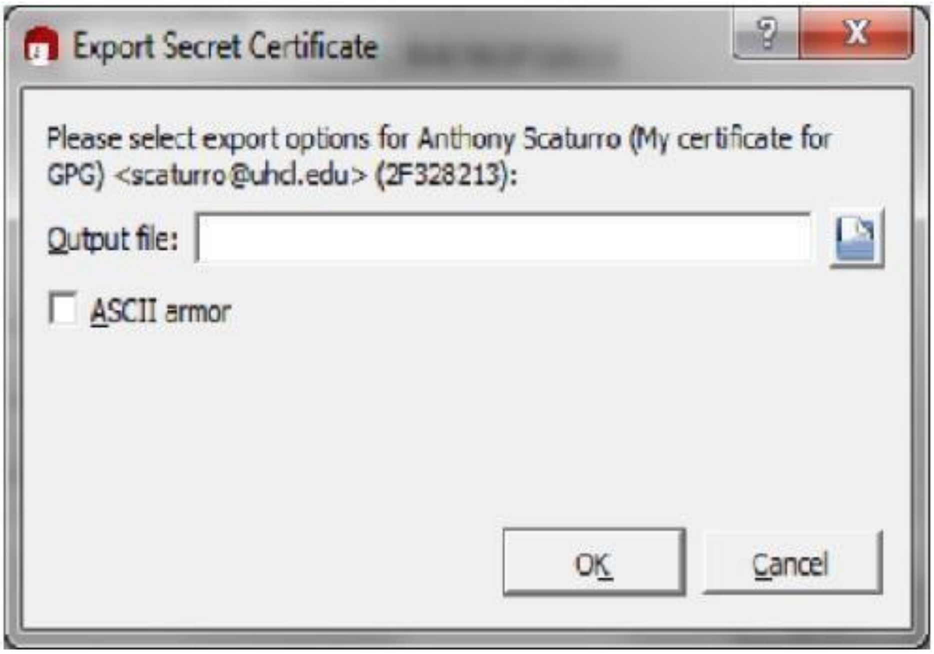
10. You will be asked to re-enter the passphrase



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

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

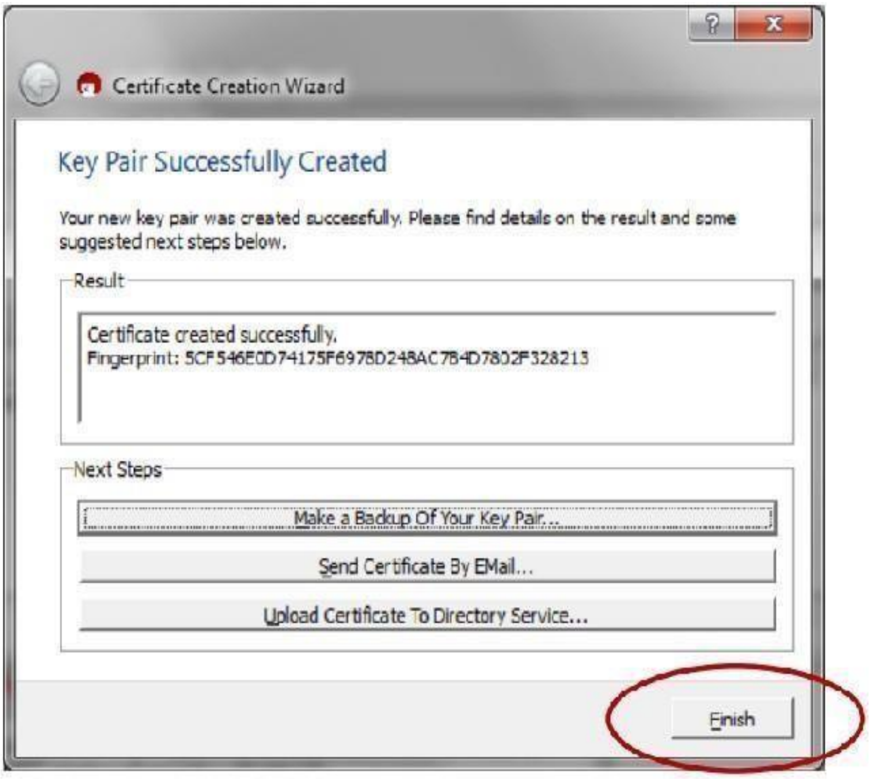
13. If you choose to backup your key pair, you will be presented with the following screen:



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

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

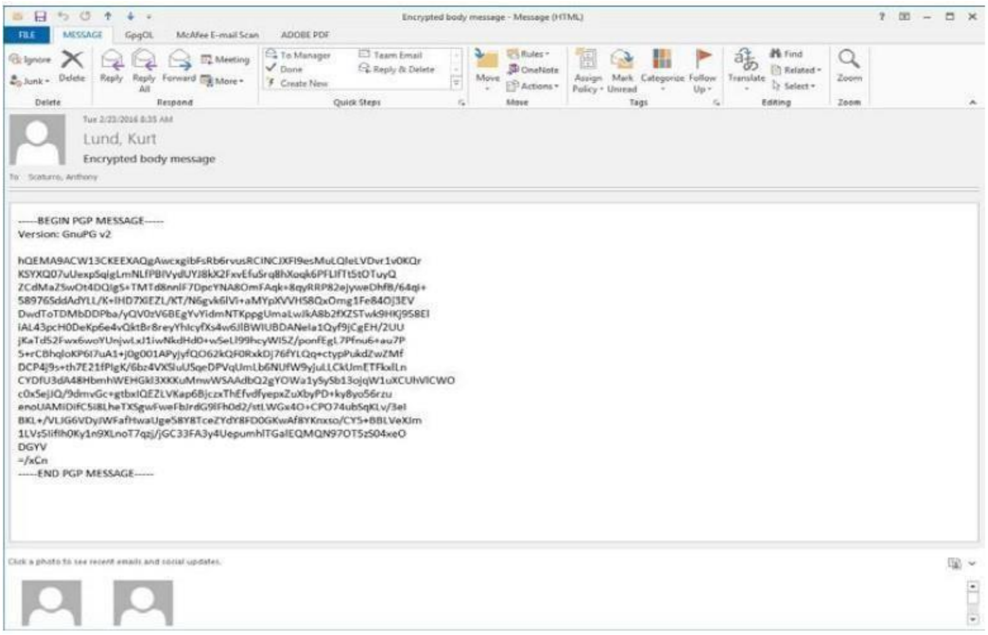
16. You will be returned to the "Key Pair Successfully Created" screen. Click the "Finish" button.



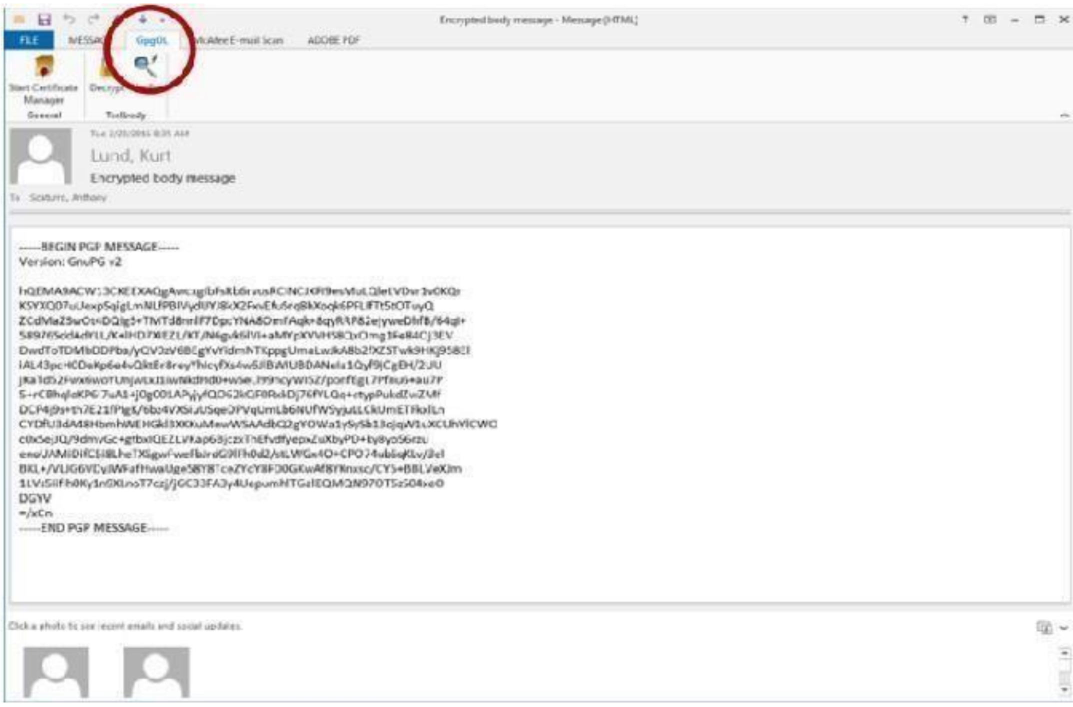
17. Before the program closes, you will need to confirm that you want to close the program by clicking on the "Quit Kleopatra" button

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

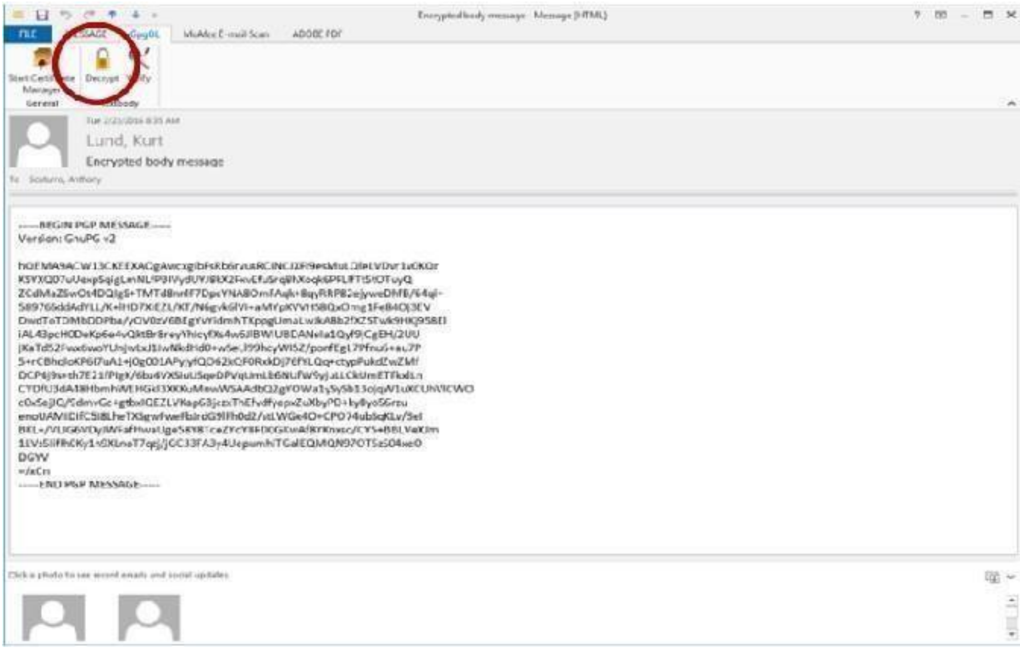
1. Open the e-mail message



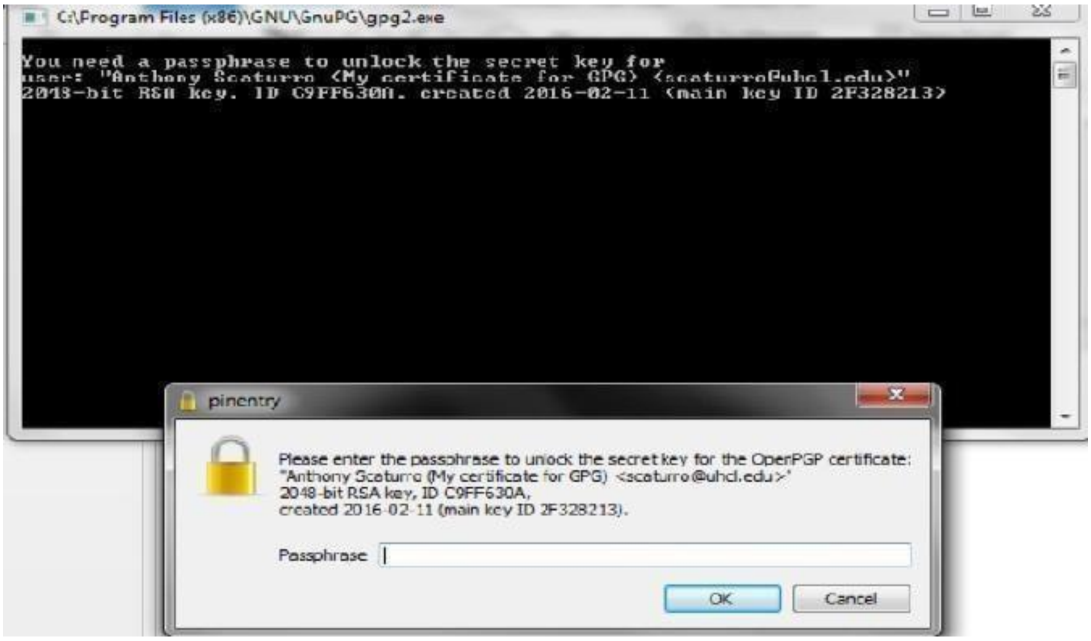
2. Select the GpgOL tab



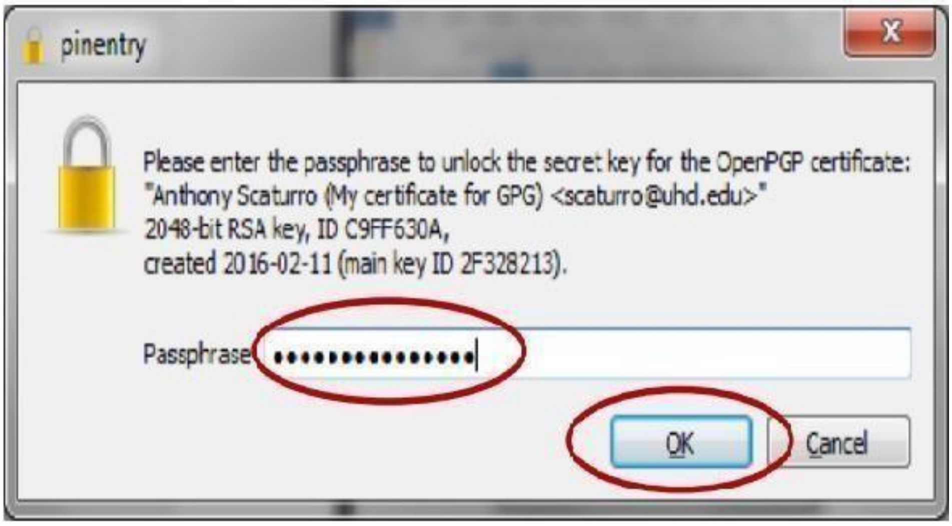
3. Click the “Decrypt” button



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

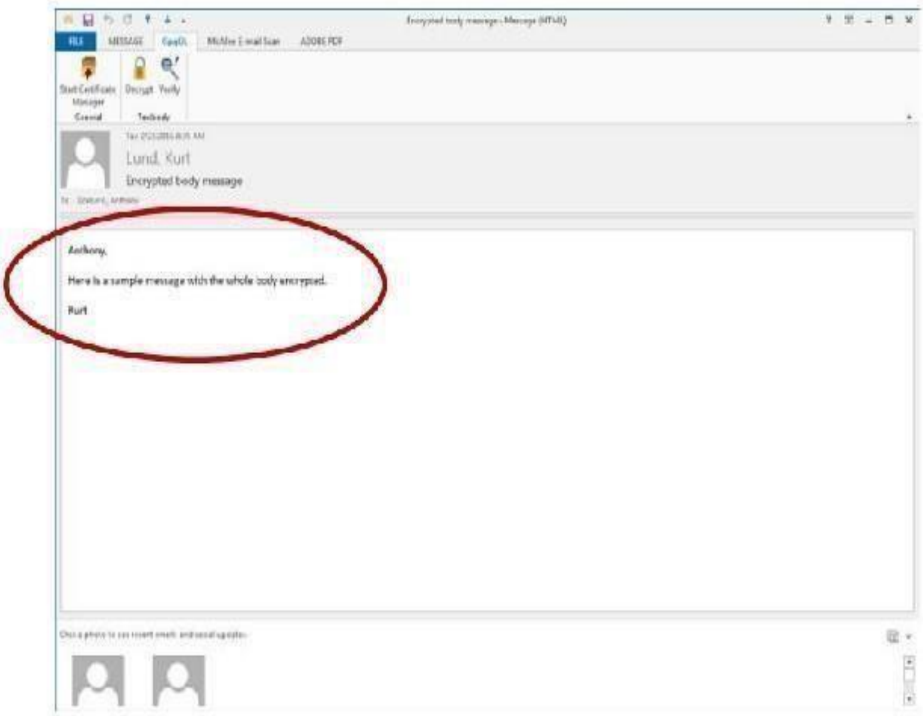


5. Enter your passphrase and click the “OK” button



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

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



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

RESULT:

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

Ex. No : 13 Date:	**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:

HON EY 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 KFSensoraccurately simulates the way Microsoft's web server (IIS) responds to both valid and invalid requests. As well as being able to host a website it also handles complexities such as range requests and client side cache negotiations. This makes it extremely difficult for an attacker to fingerprint, or identify KFSensor as a honeypot.

PROCEDURE:

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

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

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

boot.

STEP-4: Click Next to setup wizard.

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

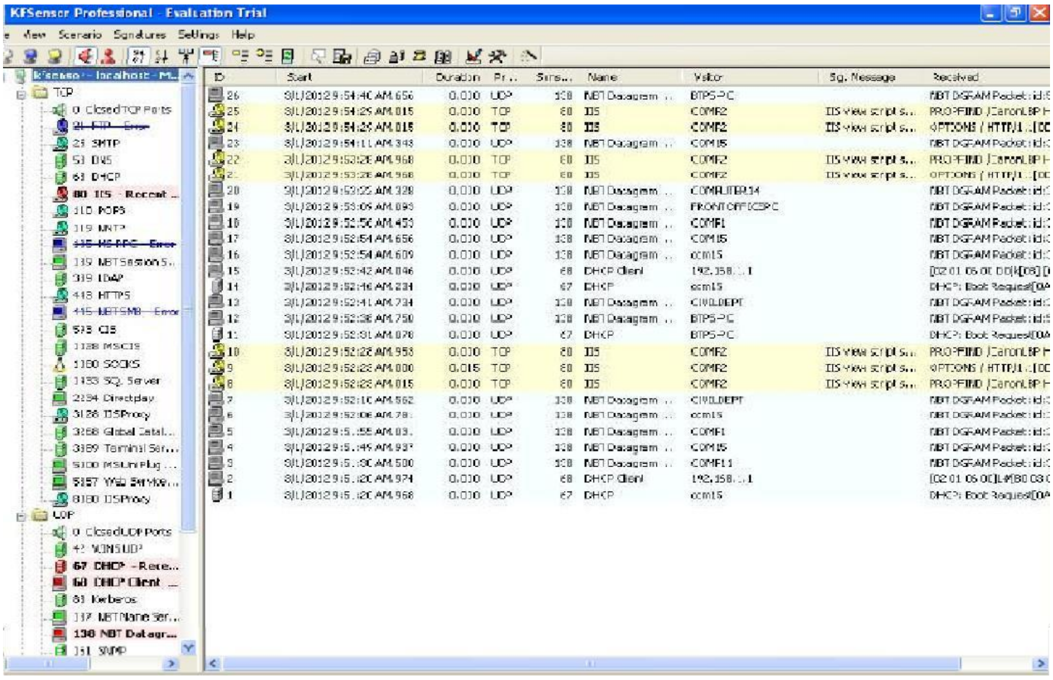
STEP-6: "Send the email and Send from email", enter the ID and Click Next.

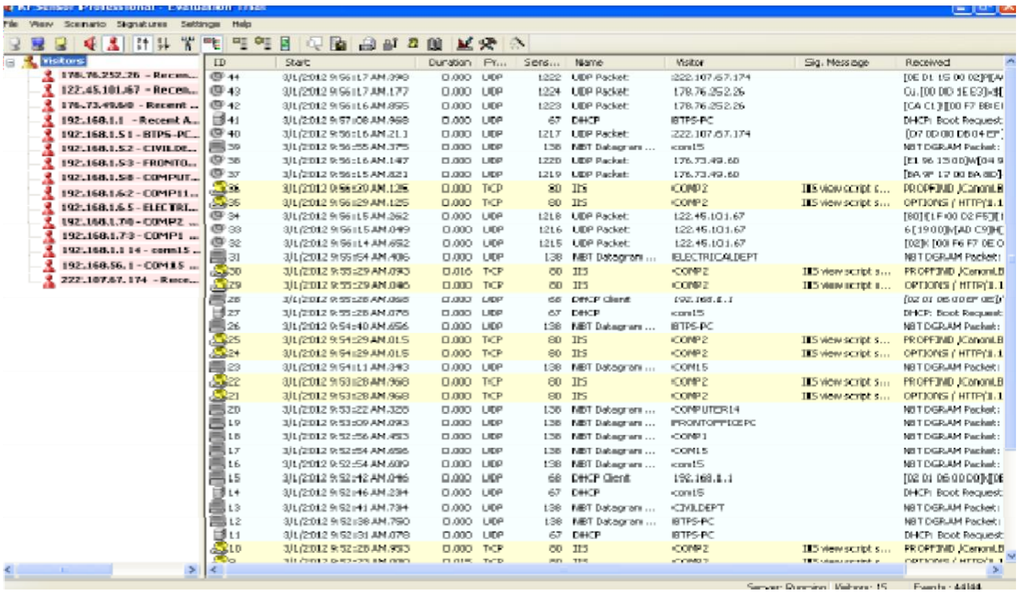
STEP-7: Select the options such as Denial of Service[DOS], Port Activity, Proxy Emulsion, Network Port Analyzer, Click Next.

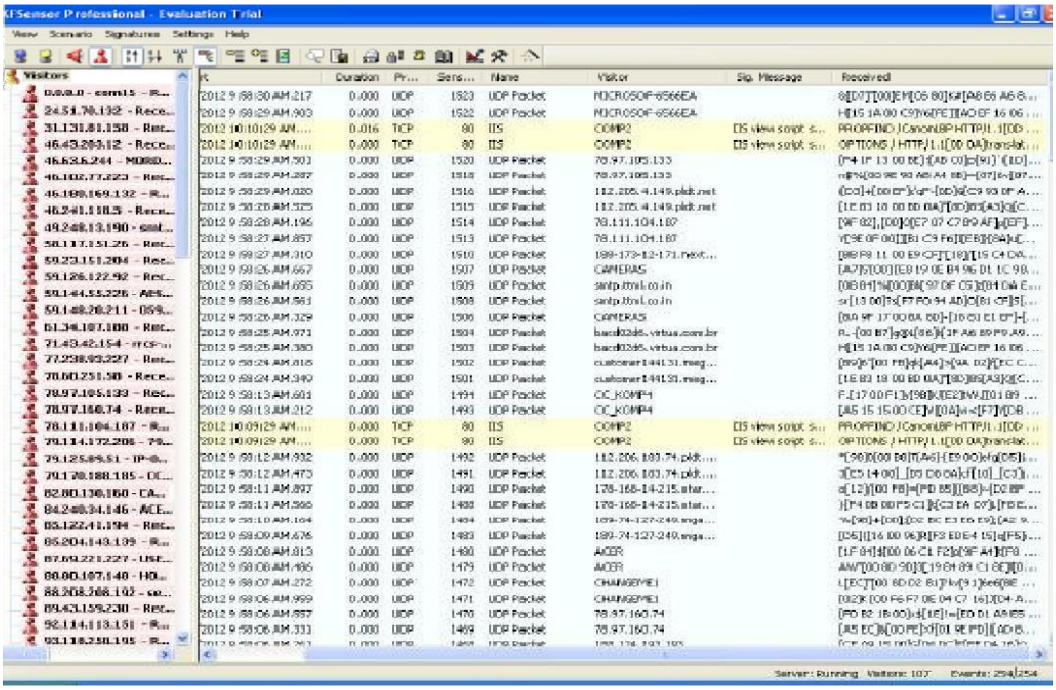
STEP-8: Select Install as System service and Click Next.

STEP-9: Click finish.

SCREENSHOTS:







RESULT:

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

Ex. No : 14 Date:	INSTALLATION OF ROCKETS

AIM:

Rootkit is a stealth type of malicious software designed to hide the existence of certain process from normal methods of detection and enables continued privileged access to a computer.

INTRODUCTION:

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

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

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

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

PROCEDURE:

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

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

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

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

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

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

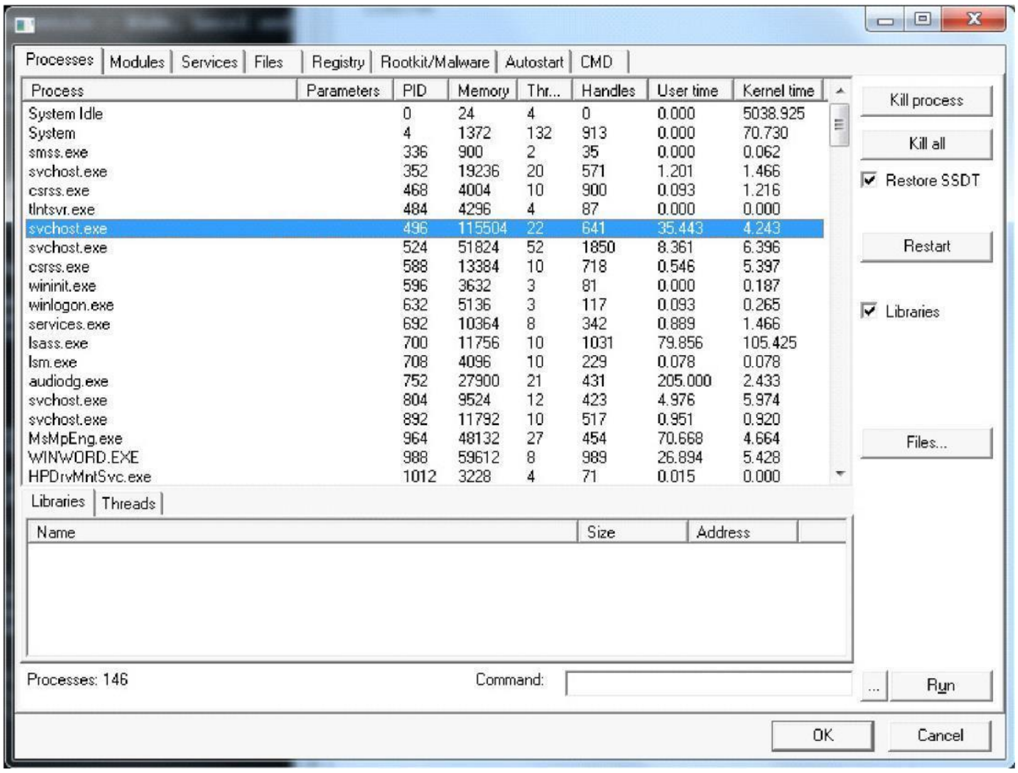
STEP-7: Registry displays Hkey_Current_user and Hkey_Local_Machine.

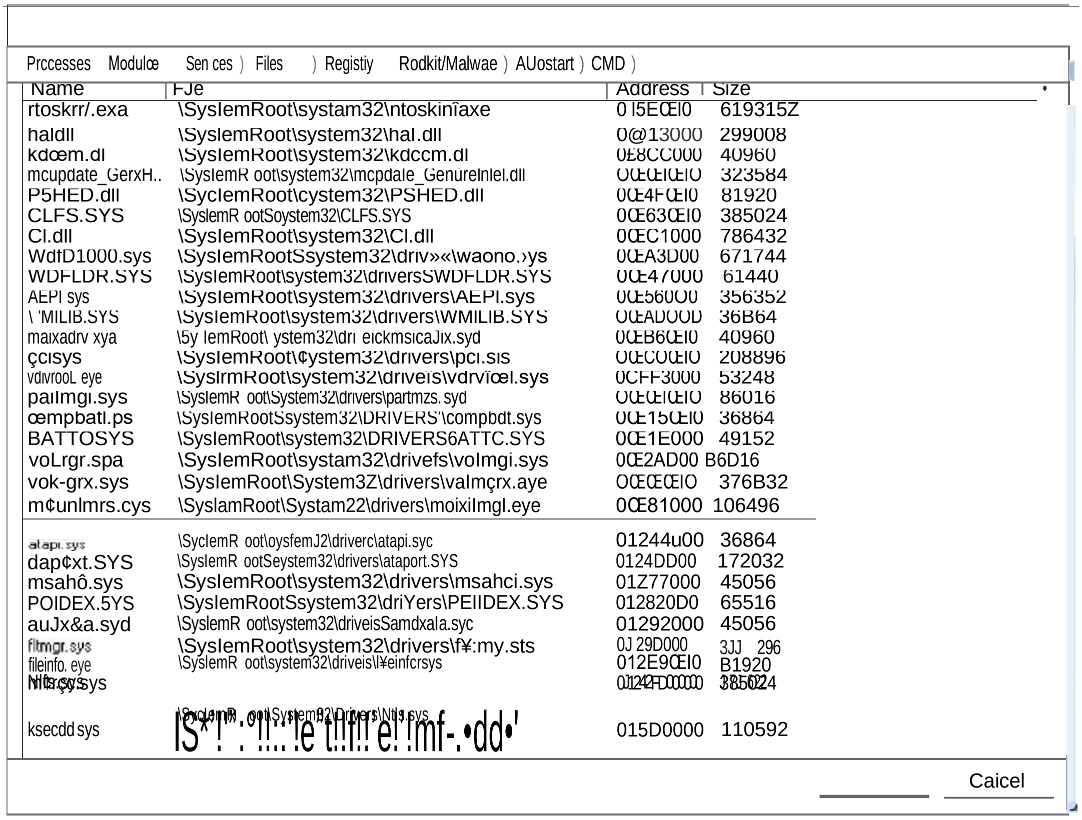
STEP-8: Rootkits / Malwares scans the local drives selected.

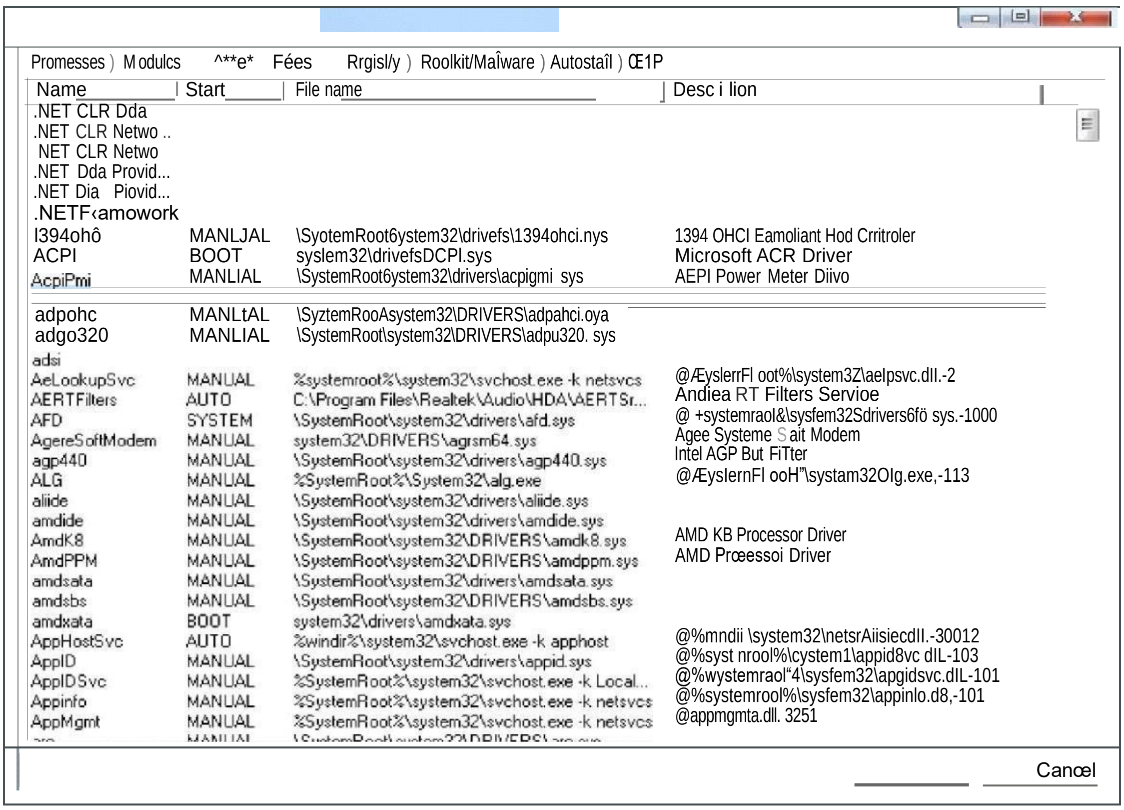
STEP-9: Autostart displays the registry base Autostart applications.

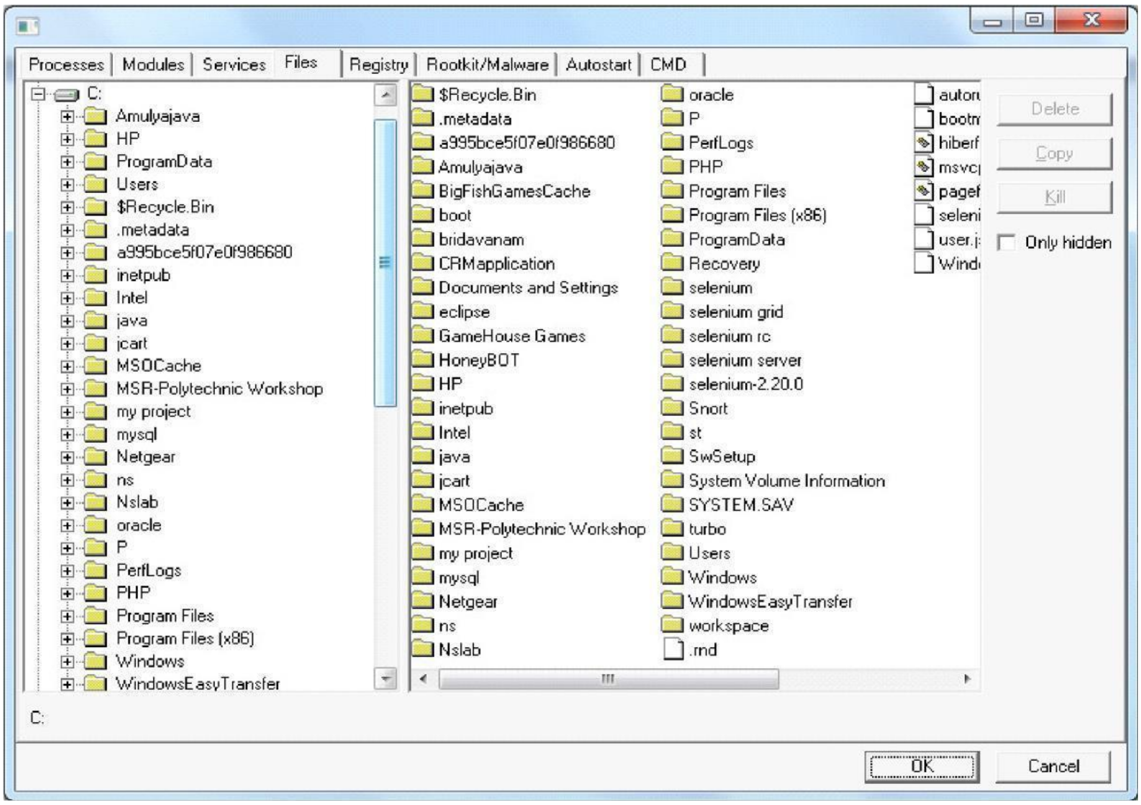
STEP-10:CMD allows the user to interact with command line utilities or Registry

SCREENSHOTS:




Processes Modules Services Files Registry Rootkit/Malware Autostart CMD
Name File Address Size
ntoskrnl.exe \SystemRoot\system32\ntoskrnl.exe 619315Z
hal.dll \SystemRoot\system32\hal.dll 299008
kdcom.dll \SystemRoot\system32\kdcom.dll 40960
mcupdate_GenuineIntel.dll \SystemRoot\system32\mcupdate_GenuineIntel.dll 323584
PSHED.dll \SystemRoot\system32\PSHED.dll 81920
CLFS.SYS \SystemRoot\system32\CLFS.SYS 385024
CI.dll \SystemRoot\system32\CI.dll 786432
Wdf01000.sys 671744
WDFLDR.SYS \SystemRoot\system32\drivers\WDFLDR.SYS 61440
ACPI.sys \SystemRoot\system32\drivers\ACPI.sys 356352
WMILIB.SYS \SystemRoot\system32\drivers\WMILIB.SYS 36864
40960
pci.sys \SystemRoot\system32\drivers\pci.sys 208896
53248
partmgr.sys \SystemRoot\system32\drivers\partmgr.sys 86016
compbatt.sys \SystemRoot\system32\DRIVERS\compbatt.sys 36864
BATTC.SYS \SystemRoot\system32\DRIVERS\BATTC.SYS 49152
volmgr.sys \SystemRoot\system32\drivers\volmgr.sys
376B32
106496
atapi.sys \SystemRoot\system32\drivers\atapi.sys 01244000 36864
ataport.SYS \SystemRoot\system32\drivers\ataport.SYS 0124DD00 172032
msahci.sys \SystemRoot\system32\drivers\msahci.sys 01277000 45056
PCIIDEX.SYS \SystemRoot\system32\drivers\PCIIDEX.SYS 65516
amdxata.sys \SystemRoot\system32\drivers\amdxata.sys 01292000 45056
fltmgr.sys \SystemRoot\system32\drivers\fltmgr.sys
fileinfo.sys \SystemRoot\system32\drivers\fileinfo.sys
Ntfs.sys \SystemRoot\System32\Drivers\Ntfs.sys
ksecdd.sys 015D0000 110592
Cancel


Processes Modules Services Files Registry Rootkit/Malware Autostart CMD
Name Start File name Description
.NET CLR Data
.NET CLR Netwo ..
.NET CLR Netwo
.NET Data Provid...
.NET Data Provid...
.NETFramework
1394ohci MANUAL \SystemRoot\system32\drivers\1394ohci.sys 1394 OHCI Compliant Host Controller
ACPI BOOT system32\drivers\ACPI.sys Microsoft ACPI Driver
AcpiPmi MANUAL \SystemRoot\system32\drivers\acpipmi.sys ACPI Power Meter Driver
adpahci MANUAL \SystemRoot\system32\DRIVERS\adpahci.sys
adpu320 MANUAL \SystemRoot\system32\DRIVERS\adpu320.sys
adsi
AeLookupSvc MANUAL %systemroot%\system32\svchost.exe -k netsvcs
AERTFilters AUTO C:\Program Files\Realtek\Audio\HDA\AERTSr... Andrea RT Filters Service
AFD SYSTEM \SystemRoot\system32\drivers\afd.sys
AgereSoftModem MANUAL system32\DRIVERS\agrsm64.sys
agp440 MANUAL \SystemRoot\system32\drivers\agp440.sys Intel AGP Bus Filter
ALG MANUAL %SystemRoot%\System32\alg.exe
aliide MANUAL \SystemRoot\system32\drivers\aliide.sys
amdide MANUAL \SystemRoot\system32\drivers\amdide.sys
AmdK8 MANUAL \SystemRoot\system32\DRIVERS\amdk8.sys AMD K8 Processor Driver
AmdPPM MANUAL \SystemRoot\system32\DRIVERS\amdppm.sys AMD Processor Driver
amdsata MANUAL \SystemRoot\system32\drivers\amdsata.sys
amdsbs MANUAL \SystemRoot\system32\DRIVERS\amdsbs.sys
amdxata BOOT system32\drivers\amdxata.sys
AppHostSvc AUTO %windir%\system32\svchost.exe -k apphost
AppID MANUAL \SystemRoot\system32\drivers\appid.sys
AppIDSvc MANUAL %SystemRoot%\system32\svchost.exe -k Local...
Appinfo MANUAL %SystemRoot%\system32\svchost.exe -k netsvcs
AppMgmt MANUAL %SystemRoot%\system32\svchost.exe -k netsvcs @appmgmts.dll,-3251
Cancel



RESULT:

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

Ex. No : 15 Date:	WORKING WITH NET STUMBLER TO PERFORM WIRELES 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.

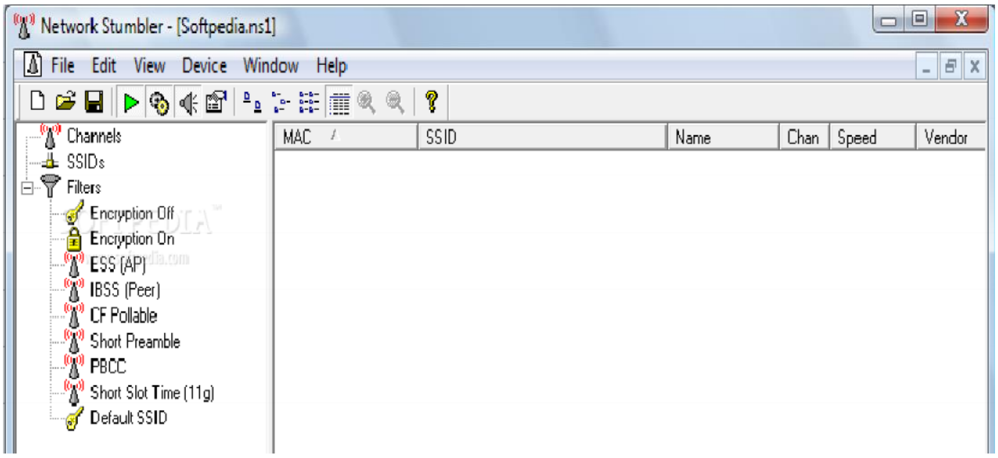
STEP-7: The next column displays the Access points Service Set Identifier[SSID] which is

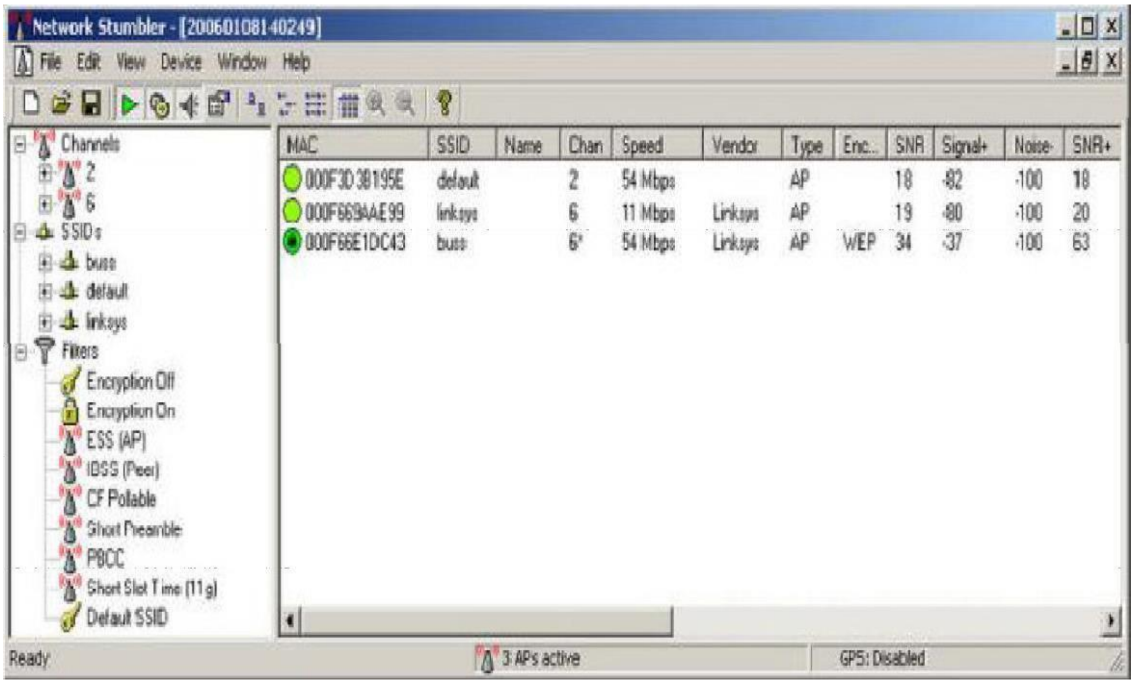
useful to crack the password.

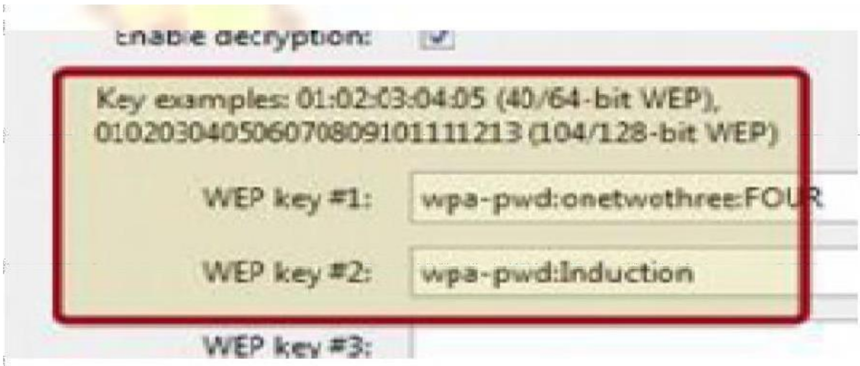
STEP-8: To decrypt use WireShark tool by selecting Edit preferences IEEE 802.11.

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

SCREENSHOTS:



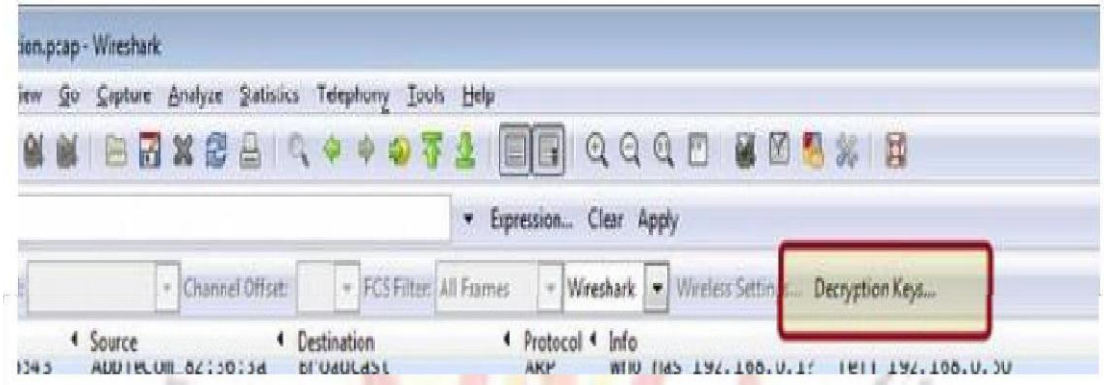




Adding Keys: Wireless Toolbar

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 :16 Date:	WORKING WITH SNORT TOOL TO DEMONSTRATE INTRUSION

WORKING WITH SNORT TOOL TO DEMONSTRATE INTRUSION DETECTION SYSTEM

AIM:

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

INTRODUCTION:

INTRUSION DETECTION SYSTEM :

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

Signature-based intrusion detection systems

Anomaly detection systems.

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

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

SNORT TOOL:

Snort is based on libpcap (for library packet capture), a tool that is widely used in TCP/IPtraffic sniffers and analyzers. Through protocolanalysis and content searching and matching, Snort detects attack methods, including denial of service, buffer overflow, CGI attacks, stealthport scans, and SMB probes. When suspicious behavior is detected, Snort sends a real-time alert to syslog, a separate 'alerts' file, or to apop-up window.

Snort is currently the most popular free network intrusion detection software. The advantages of Snort are numerous. According to the snort web site, “It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflow, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more” (Caswell).

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

SNORT can be configured to run in three modes:

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

1. **Sniffer mode**

   Snort –v Print out the TCP/IP packets header on the screen

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

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

   snort –dev –l c:\log –h ipaddress/24:This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory. snort –l c:\log –b This is binary mode logs everything into a single file.

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

   Snort –d –h ipaddress/24 –l c:\log –c snort.conf This will cnfigure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.

PROCEDURE:

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

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

STEP-3: Packet Logger mode snort –dev –l c:\log [create this directory in the C drive]

and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.

STEP-4: snort –dev –l c:\log –h ipaddress/24 This rule tells snort that you want to print

out the data link and TCP/IP headers as well as application data into the log directory.

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

STEP-6: Network Intrusion Detection System mode snort –d c:\log –h ipaddress/24 –c

snort.conf This is a configuration file that applies rule to each packet to decide it an action based upon the rule type in the file.

STEP-7: snort –d –h ip address/24 –l c:\log –c snort.conf This will configure snort to run

in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.

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

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

STEP-10: Skip the WinPcapdriver installation.

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

classpath.

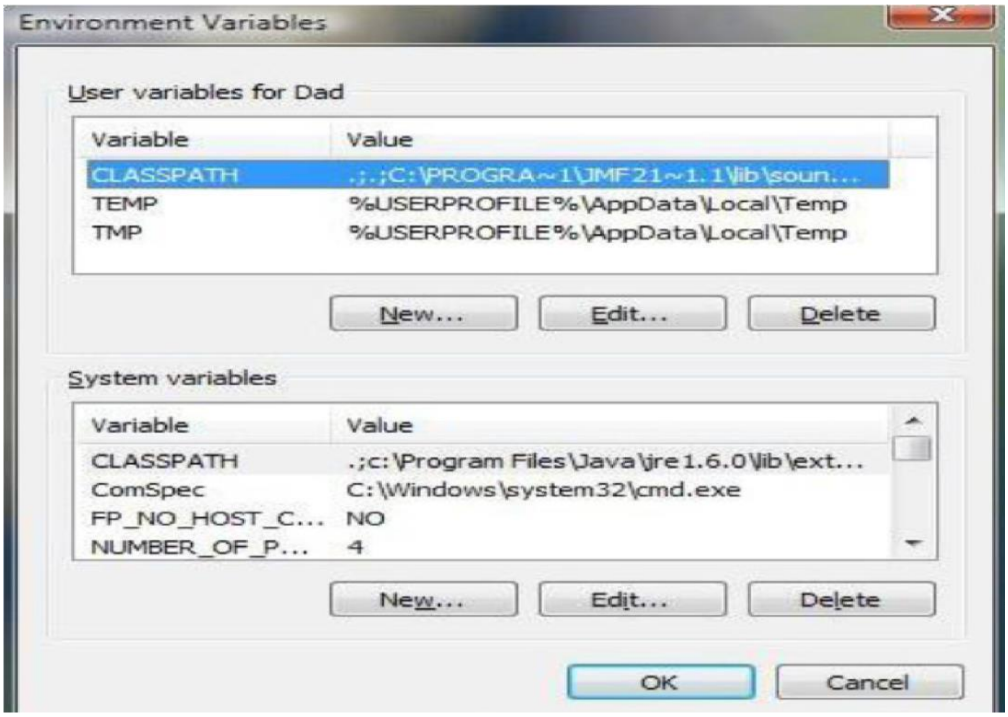
STEP-12: Create a path variable and point it at snort.exe variable name path and variable

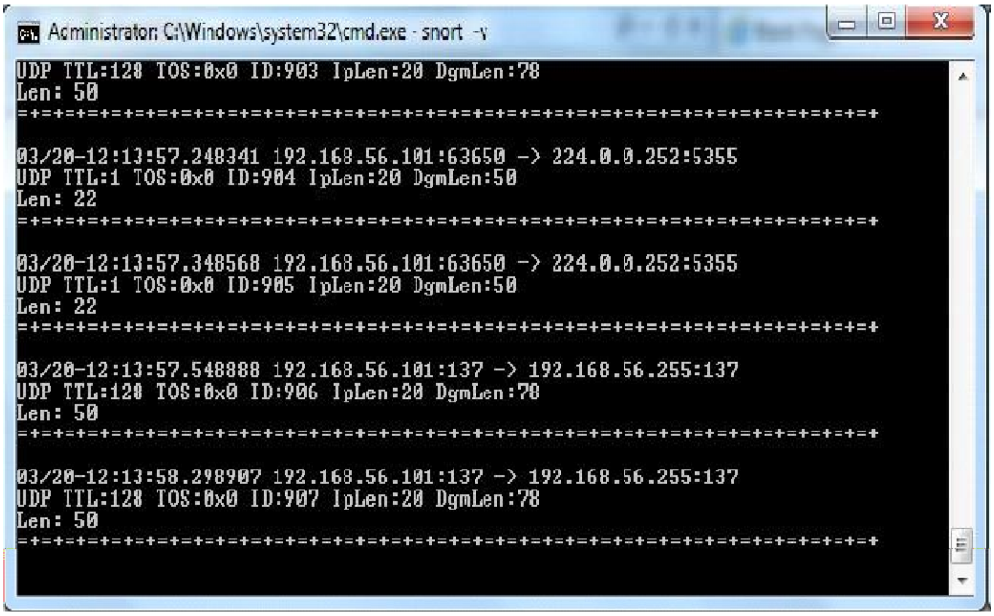
value c:\snort\bin.

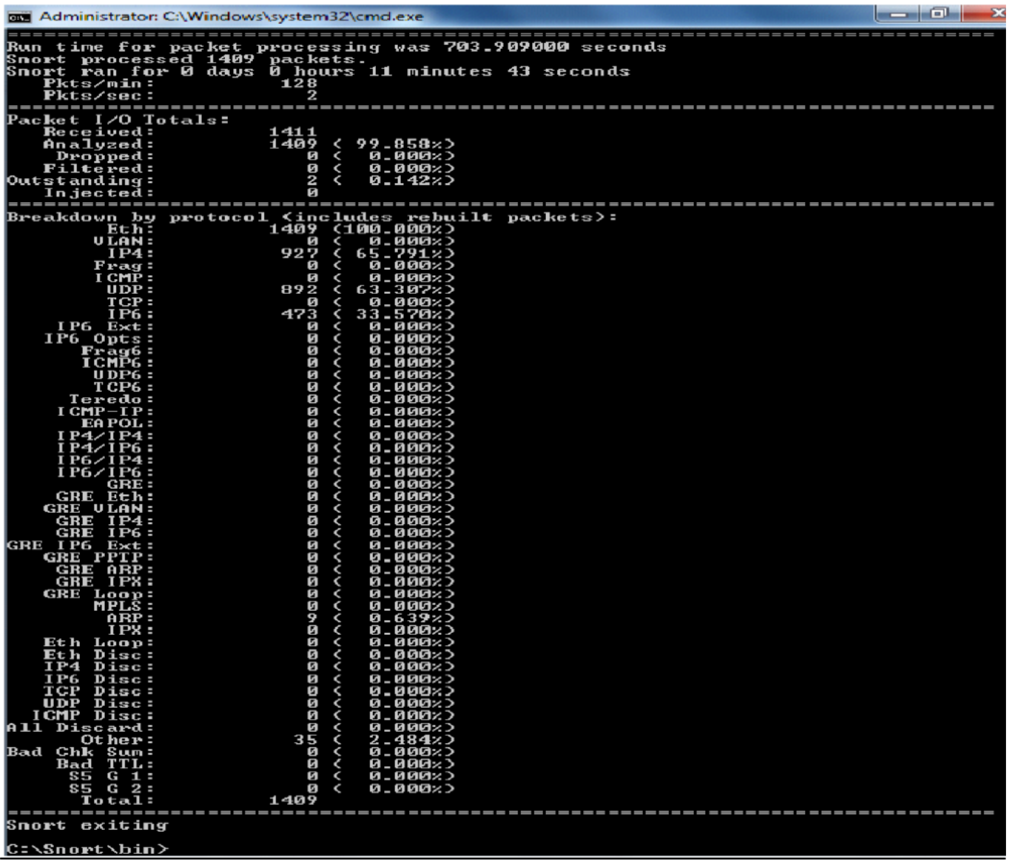
STEP-13: Click OK button and then close all dialog boxes. Open command prompt and type

the following commands:

INSTALLATION PROCESS :







RESULT:

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