
LABORATORY MANUAL

Cryptography & Network Security

BCSP 802

B. TECH.
(IV YEAR – EVEN SEM)
(2022-23)



COLLEGE OF ENGINEERING ROORKEE

(Affiliated to Veer Madho Singh Bhandari Uttarakhand Technical University, Dehradun)

Approved by AICTE & MHRD, Govt of India, Accredited by NAAC

Roorkee-Haridwar Road (NH-58),
Post Box No. 27, Vardhman Puram,
Roorkee- 247667 District. Haridwar (Uttarakhand)

www.coer.ac.in



COLLEGE OF ENGINEERING ROORKEE

Vision

"To impart education in Engineering with training, skill upgradation and research in futuristic technologies and niche areas."

Mission

M1: To develop the professionals having basic and advanced competencies so that they can serve the Society & Industry, and face the global challenges.

M2: To impart education based on latest knowledge, with analytical and experimental skills, through advanced methods of training, research and strong Institute-Industry interface.

M3: To help create innovative and entrepreneurial professional.

M4: To inculcate strong moral values for a disciplined professional growth.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision

To promote innovation-centric education and perform research in Computer Science and Engineering in pace with industrial development.

Mission

M1: To provide a learning environment that helps students to enhance problem solving skills at par with global standards.

M2: To establish Industry-Institute Interaction to make students ready for the industrial environment.

M3: To provide exposure to students to the latest tools and technologies in the area of computer hardware and software.

M4: To promote research-based projects/activities in the emerging areas.

M5: To foster the science of creativity and educating ownership for sustainable & scalable ventures.

M6: To pass on the requisite moral characteristics and infuse discipline amongst the students to make them attain consistent elevation in their professional life.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1: To equip the students with skills and latest updated so that they can work and contribute to the continuously changing landscape of IT Industry.

PEO2: To provide research-oriented education with knowledge of state-of-art analytical and experimental tools to enable students to pursue higher studies in institutions of repute in India and abroad

PEO3: To inculcate culture of professionalism, ethical conduct, team work with good communication skills to enable the students to be successful in their career and enable them to launch start-ups in their chosen field.

PROGRAMME SPECIFIC OUTCOME (PSOs)

PSO1: Students will have the ability to apply software engineering principles to design, build, test, and deliver solutions for Software Industry

PSO2: The students will be able to use programming, database, networking and web development concepts for developing solutions for real-life problems.

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision

To produce professional graduates trained in the latest tools and technologies of Information Technology, and to build a strong teaching & research environment that tracks and responds to the challenges of modern times

Mission

1. To impart quality engineering education that enables the students become competent IT professionals by providing professional technical education and training.
2. To imbibe experiential learning blended with critical thinking and strong Industry-connect for the students and ignite them to pursue Research in emerging areas.
3. To establish Start-up Eco-system within the department to produce entrepreneurs.
4. To create disciplined graduates with strong moral values who may continuously thrive for higher echelons in the personal and professional life.

DEPARTMENT OF INFORMATION TECHNOLOGY

Program Educational Objectives (PEOs)

PEO No.	Program Educational Objectives Statements
PEO1	To equip the students with skills and latest updates so that they can work and contribute to the continuously changing landscape of IT Industry.
PEO2	To provide good academic and research-oriented environment with knowledge of latest IT tools to enable students to pursue higher studies in institutions of repute in India and abroad.
PEO3	To inculcate culture of professionalism, ethical conduct, team work with good communication skills to enable the students to be successful in their career and enable them to launch start-ups in their chosen field.

Program Specific Outcomes (PSOs)

PSO1	The ability to understand, analyse and develop programming skills with the latest tools and technology in computing and apply standard practices, strategies in software development to deliver quality products and/or to pursue entrepreneur.
PSO2	To induce continuous learning aptitude and an inclination for lifelong learning; and intent to act as good citizen by inculcating in them moral values & ethics.

Program Outcomes

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. 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.
11. 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.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
 - Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
 - Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.
 - Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high-end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Sub Code	BCSP 802
Sub Name	Cryptography

Course Outcomes	
CO1	To be able to identify common network security vulnerabilities /attacks; explain the foundations of Cryptography and network security
CO2	To be able to evaluate the risks and threats to networked computers
CO3	To be able to demonstrate detailed knowledge of the role of encryption to protect data
CO4	To be able to analyse security issues arising from the use of certain types of technologies
CO5	To be able to identify the appropriate procedures required to secure networks; identify the appropriate procedures required for system security testing and procedures of Backup and Recovery

CO-PO Matrix												
Course outcomes	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	1	2	1	1						1	2
CO2	2	1	1		2						1	2
CO3	1	2		1	1						1	2
CO4	1	2	2	1	1						1	2
CO5	2	1	2	1	2						1	2
Average	1.4	1.4	1.75	1	1.4						1	2

COS	PSO1	PSO2
CO1	2	2
CO2	1	2
CO3	2	2
CO4	1	1
CO5	2	2
Average	1.6	1.8

Study and Evaluation Scheme

Course Code	Course Name	Teaching Scheme			Credits Assigned			
		Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total
BCSP 802	Cryptography & Network Security	--	02 (50 Marks)	--	--	01	--	01

EX. NO: 1(A)

IMPLEMENTATION OF CAESAR CIPHER

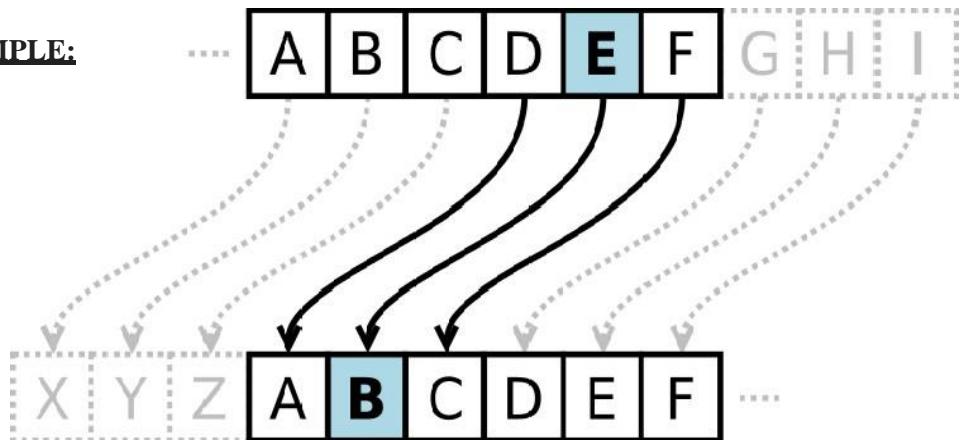
AIM:

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

DESCRIPTION:

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

EXAMPLE:



ALGORITHM:

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

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

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

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

STEP-5: Display the cipher text obtained above.

PROGRAM: (Caesar Cipher)

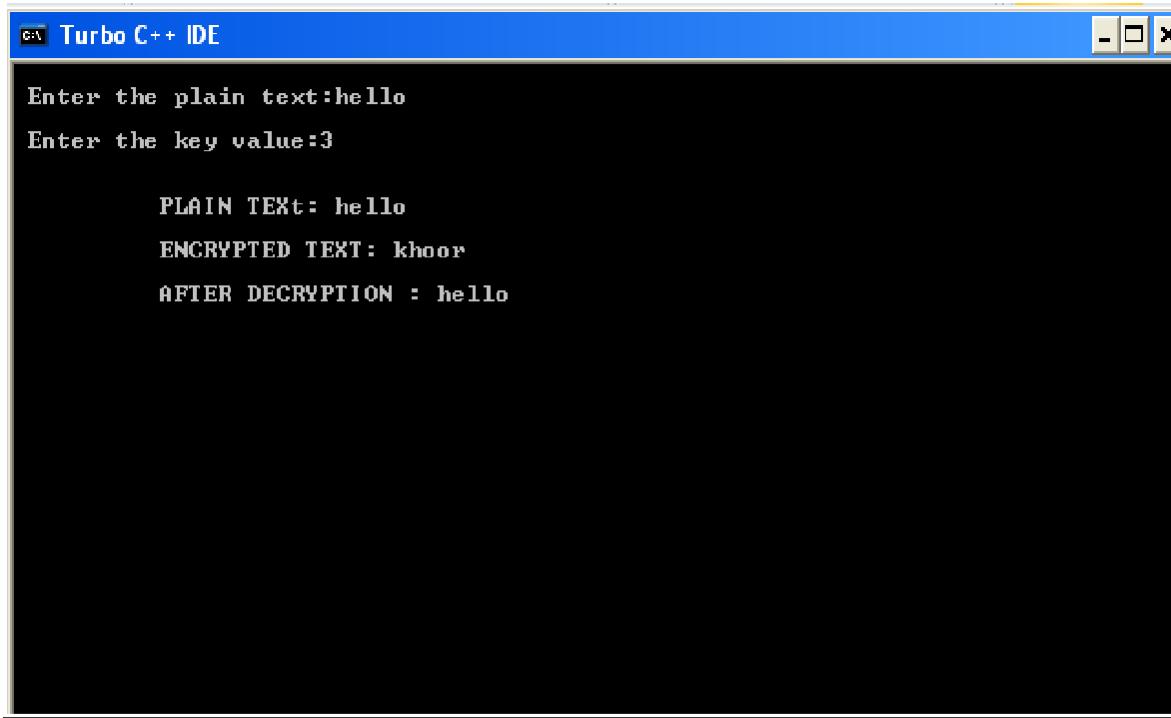
```
#include <stdio.h>
#include <string.h>
#include<conio.h>
#include <ctype.h>
void main ()
```

```

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

```

OUTPUT:



The screenshot shows a window titled "Turbo C++ IDE". Inside the window, the following text is displayed:

```
Enter the plain text:hello
Enter the key value:3

PLAIN TEXT: hello
ENCRYPTED TEXT: khoor
AFTER DECRYPTION : hello
```

RESULT:

Thus, the implementation of Caesar cipher had been executed successfully.

EX. NO: 1(B)**IMPLEMENTATION OF PLAYFAIR CIPHER****AIM:**

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

DESCRIPTION:

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

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

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

EXAMPLE:**D. Playfair Cipher**

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

Grouped text: CR YP TO IS TO XO EA SY

Ciphertext: AQ TV YB NI YB YF CB OZ

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

ALGORITHM:

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

STEP-2: Read the keyword from the user.

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

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

STEP-5: Display the obtained cipher text.

PROGRAM: (Playfair Cipher)

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

```

w=(w+1)%5;y=(y+1)%5;
printf("%c%c",key[w][x],key[y][z]);
fprintf(out, "%c%c",key[w][x],key[y][z]);
}
else
{
    printf("%c%c",key[w][z],key[y][x]);
    fprintf(out, "%c%c",key[w][z],key[y][x]);
}

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

    for(i=0;i<26;i++)
    {
        for(k=0;k<n;k++)
        {
            if(keystr[k]==alpa[i])
            break;
            else if(alpa[i]=='J')
            break;
        }
        if(k==n)
        {
            keyminus[j]=alpa[i];j++;
        }
    }
}

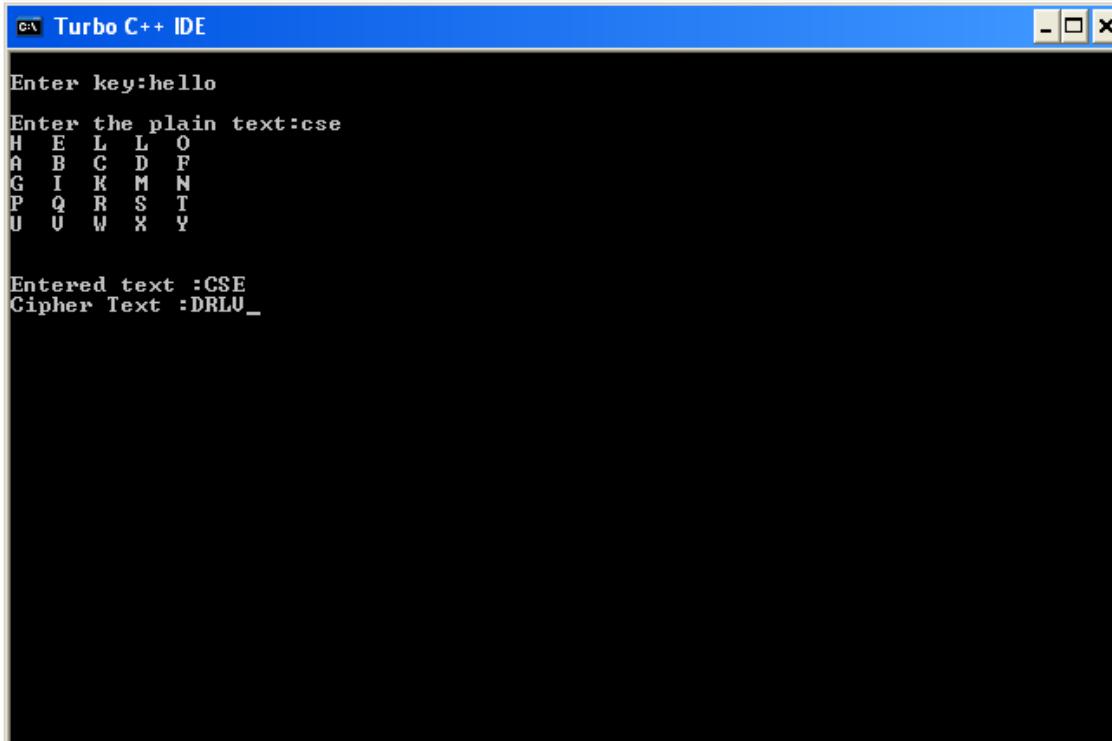
```

```

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

```

OUTPUT:



The screenshot shows a window titled "Turbo C++ IDE". Inside, the terminal output is as follows:

```
Enter key:hello
Enter the plain text:cse
H E L L O
A B C D F
G I K M N
P Q R S T
U V W X Y

Entered text :CSE
Cipher Text :DRLU_
```

RESULT:

Thus the Playfair cipher substitution technique had been implemented successfully.

EX. NO: 1(C)

IMPLEMENTATION OF HILL CIPHER

AIM:

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

DESCRIPTION:

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

EXAMPLE:

$$\begin{bmatrix} 2 & 4 & 5 \\ 9 & 2 & 1 \\ 3 & 17 & 7 \end{bmatrix} \begin{bmatrix} 0 \\ 10 \\ 19 \end{bmatrix} = \begin{bmatrix} 171 \\ 57 \\ 456 \end{bmatrix} \pmod{26} = \begin{bmatrix} 15 \\ 5 \\ 14 \end{bmatrix} = \text{'PFO'}$$

ALGORITHM:

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

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

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

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

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

PROGRAM: (Hill Cipher)

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

```

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

```

OUTPUT:

```

Turbo C++ IDE
Enter plain text
ACT
@ 2 19
Encrypted Cipher Text : P O H
Decrypted Cipher Text : A C T

```

RESULT:

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

EX. NO: 1(D)**IMPLEMENTATION OF VIGENERE CIPHER****AIM:**

To implement the Vigenere Cipher substitution technique using C program.

DESCRIPTION:

To encrypt, a table of alphabets can be used, termed a **tabula recta**, Vigenère square, or Vigenère table. It consists of the alphabet written out 26 times in different rows, each

alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar ciphers. At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

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

EXAMPLE:

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

ALGORITHM:

- STEP-1:** Arrange the alphabets in row and column of a 26*26 matrix.
- STEP-2:** Circulate the alphabets in each row to position left such that the first letter is attached to last.
- STEP-3:** Repeat this process for all 26 rows and construct the final key matrix.
- STEP-4:** The keyword and the plain text is read from the user.
- STEP-5:** The characters in the keyword are repeated sequentially so as to match with that of the plain text.
- STEP-6:** Pick the first letter of the plain text and that of the keyword as the row indices and column indices respectively.
- STEP-7:** The junction character where these two meet forms the cipher character.
- STEP-8:** Repeat the above steps to generate the entire cipher text.

PROGRAM: (Vigenere Cipher)

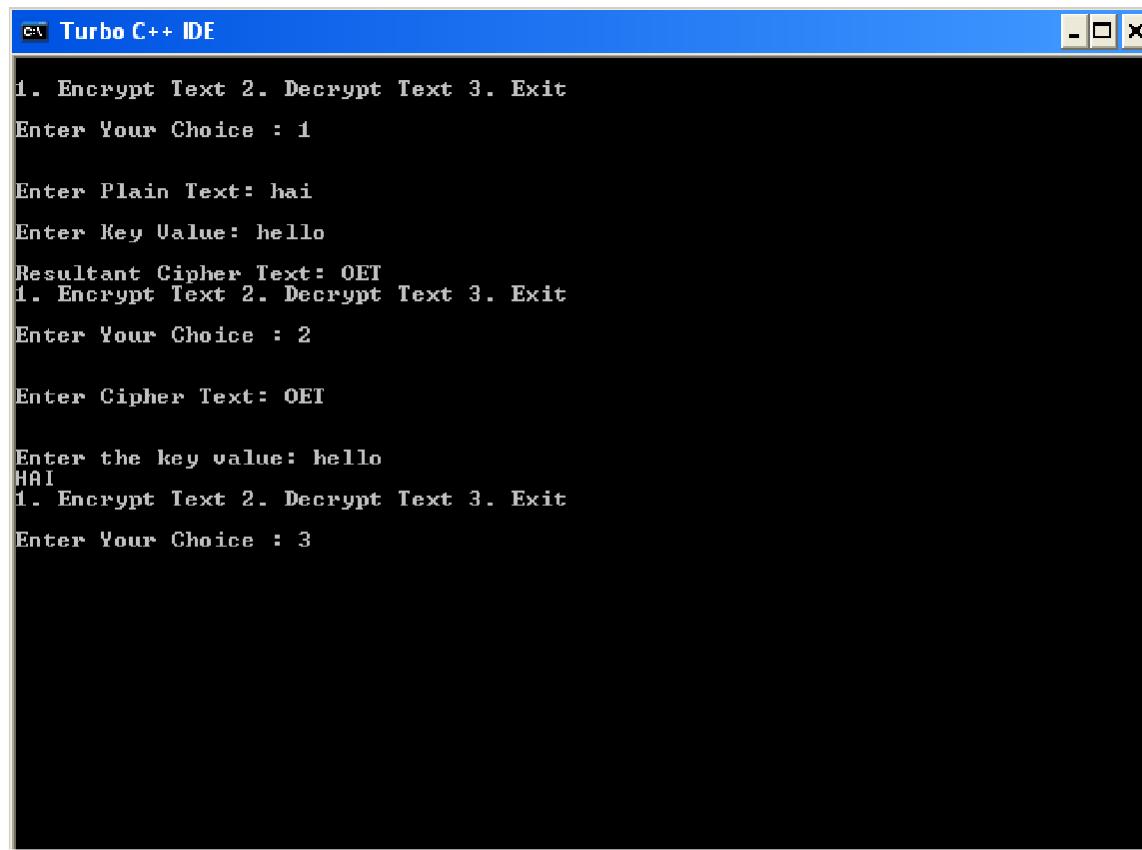
```
#include <stdio.h>
#include<conio.h>
#include <ctype.h>
#include <string.h>
void encipher();
void decipher();
void main()
{
    int choice;
    clrscr();
    while(1)
    {
        printf("\n1. Encrypt Text");
        printf("\t2. Decrypt Text");
        printf("\t3. Exit");
        printf("\n\nEnter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

```

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

```

OUTPUT:



The screenshot shows a terminal window titled "Turbo C++ IDE". The window contains the following text output:

```
1. Encrypt Text 2. Decrypt Text 3. Exit
Enter Your Choice : 1

Enter Plain Text: hai
Enter Key Value: hello
Resultant Cipher Text: OET
1. Encrypt Text 2. Decrypt Text 3. Exit
Enter Your Choice : 2

Enter Cipher Text: OET
Enter the key value: hello
HAI
1. Encrypt Text 2. Decrypt Text 3. Exit
Enter Your Choice : 3
```

RESULT:

Thus the Vigenere Cipher substitution technique had been implemented successfully.

EX. NO: 1(E)

IMPLEMENTATION OF RAIL FENCE – ROW & COLUMN

TRANSFORMATION TECHNIQUE

AIM:

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

DESCRIPTION:

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

EXAMPLE:

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

yields the cipher

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

ALGORITHM:

STEP-1: Read the Plain text.

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

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

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

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

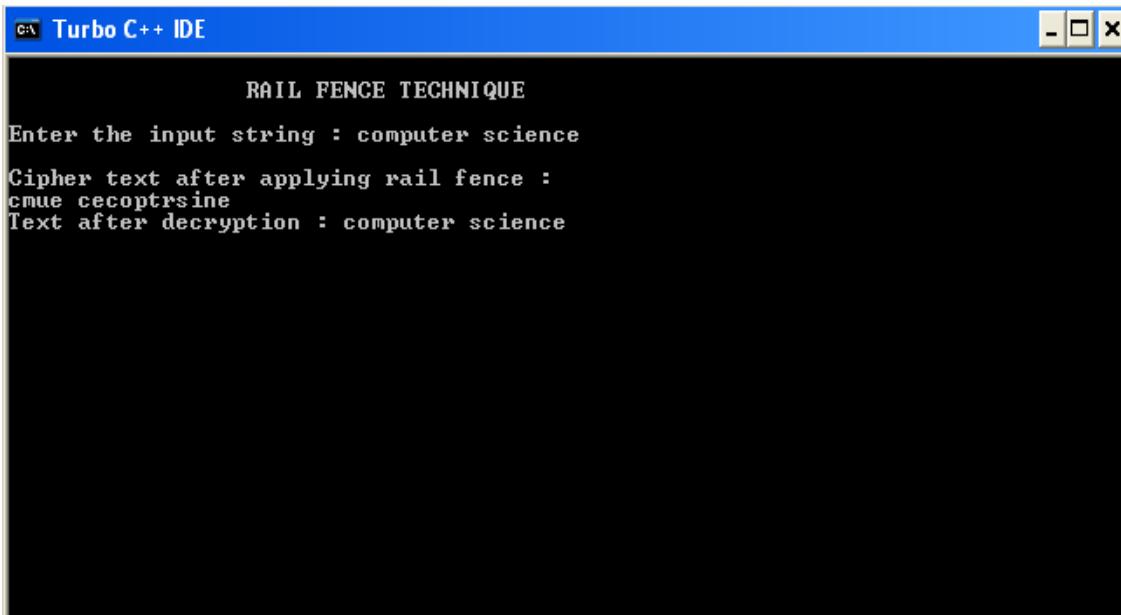
PROGRAM: (Rail Fence)

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

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

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

OUTPUT:



The screenshot shows a window titled "Turbo C++ IDE" with a blue header bar. The main window has a black background and contains white text. The text reads:
RAIL FENCE TECHNIQUE
Enter the input string : computer science
Cipher text after applying rail fence :
cmue cecoptrsine
Text after decryption : computer science

RESULT:

Thus the rail fence algorithm had been executed successfully.

AIM:

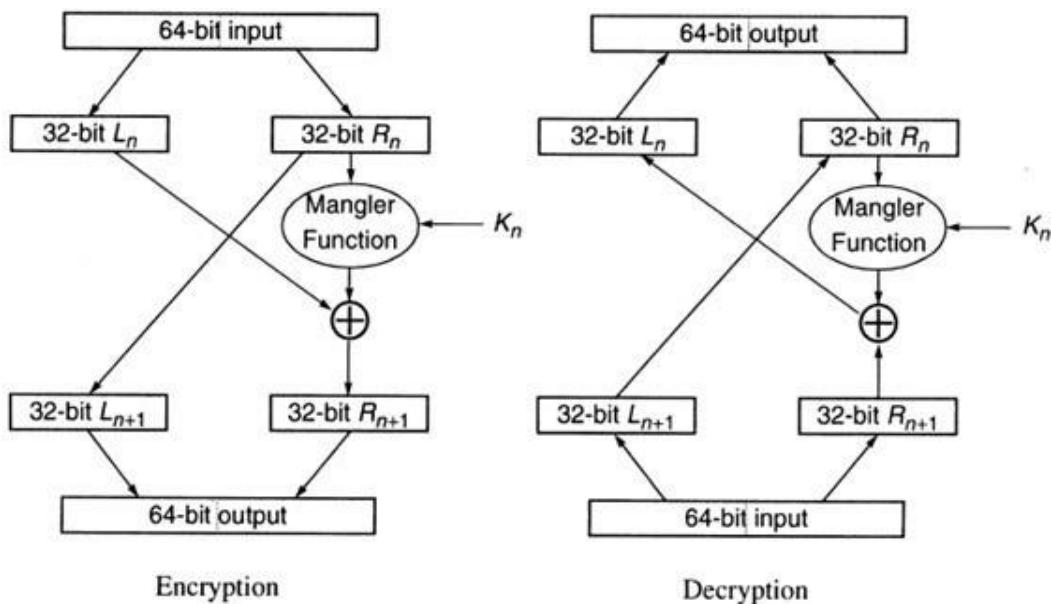
To write a C program to implement Data Encryption Standard (DES) using C Language.

DESCRIPTION:

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

The main parts of the algorithm are as follows:

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

EXAMPLE:

ALGORITHM:

- STEP-1:** Read the 64-bit plain text.
- STEP-2:** Split it into two 32-bit blocks and store it in two different arrays.
- STEP-3:** Perform XOR operation between these two arrays.
- STEP-4:** The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.
- STEP-5:** Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same process for the remaining plain text characters.

PROGRAM:

DES.java

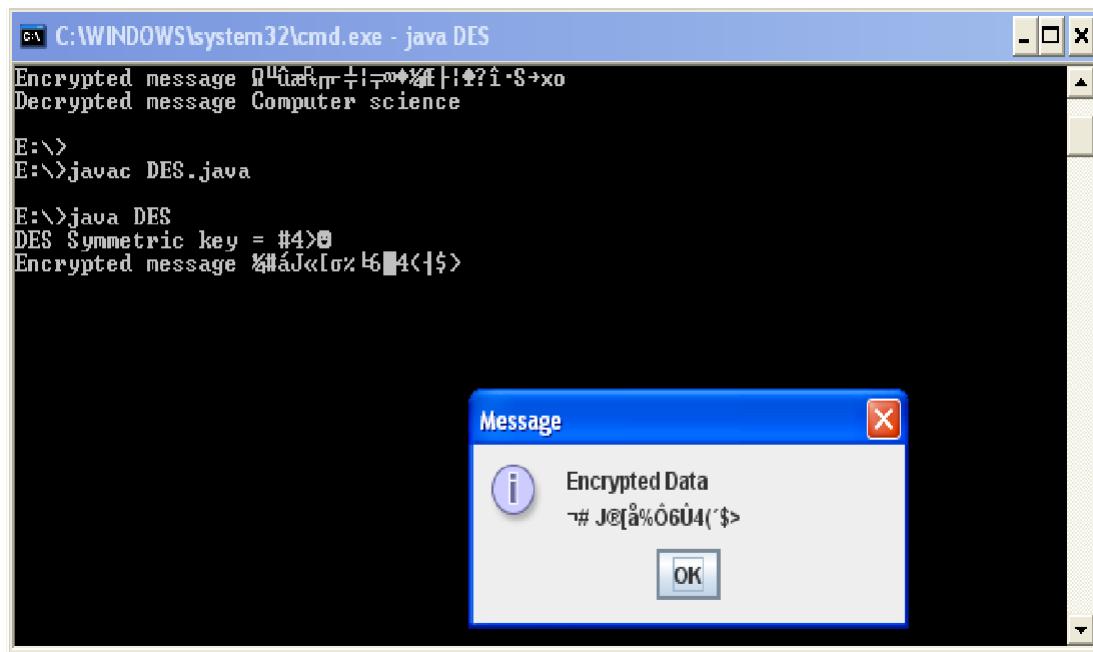
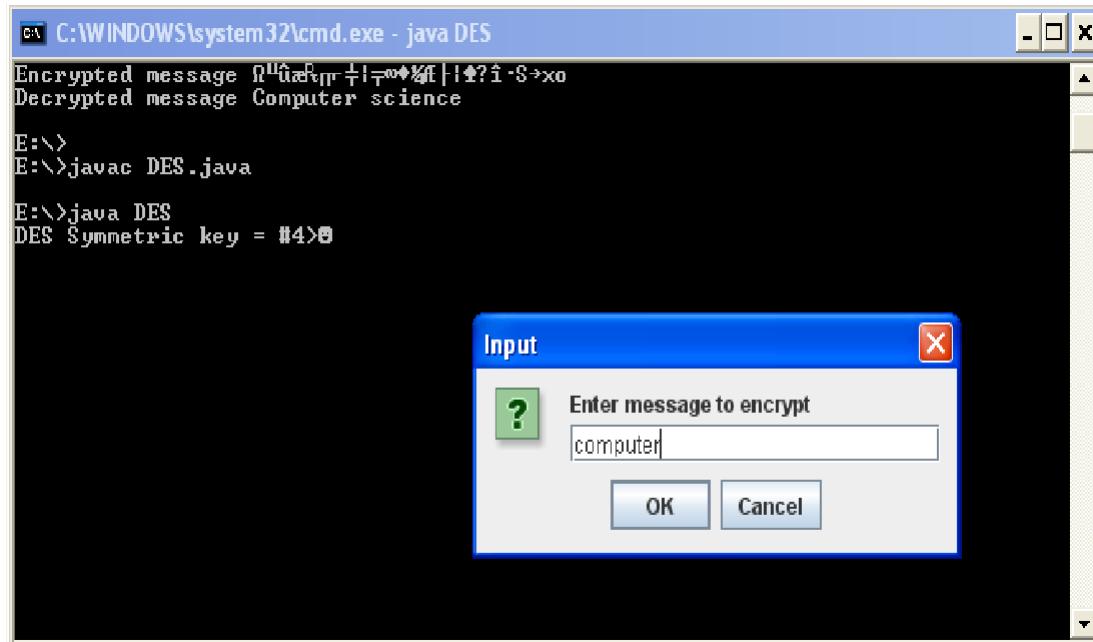
```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
class DES {
    byte[] skey = new byte[1000];
    String skeyString;
    static byte[]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

```

void generateSymmetricKey() {
    try {
        Random r = new Random();
        int num = r.nextInt(10000);
        String knum = String.valueOf(num);
        byte[] knumb = knum.getBytes();
        skey=getRawKey(knumb);
        skeyString = new String(skey);
        System.out.println("DES Symmetric key = "+skeyString);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}
private static byte[] getRawKey(byte[] seed) throws Exception
{
    KeyGenerator kgen = KeyGenerator.getInstance("DES");
    SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
    sr.setSeed(seed);
    kgen.init(56, sr);
    SecretKey skey = kgen.generateKey();
    raw = skey.getEncoded();
    return raw;
}
private static byte[] encrypt(byte[] raw, byte[] clear) throws
Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(raw,
    "DES");
    Cipher cipher = Cipher.getInstance("DES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
    byte[] encrypted = cipher.doFinal(clear);
    return encrypted;
}
private static byte[] decrypt(byte[] raw, byte[] encrypted)
throws Exception
{
    SecretKeySpec skeySpec = new SecretKeySpec(raw,
    "DES");
    Cipher cipher = Cipher.getInstance("DES");
    cipher.init(Cipher.DECRYPT_MODE, skeySpec);
    byte[] decrypted = cipher.doFinal(encrypted);
    return decrypted;
}
public static void main(String args[]) {
    DES des = new DES();
}
}

```

OUTPUT:



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

Message

Decrypted Data
computer

OK

RESULT:

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

AIM:

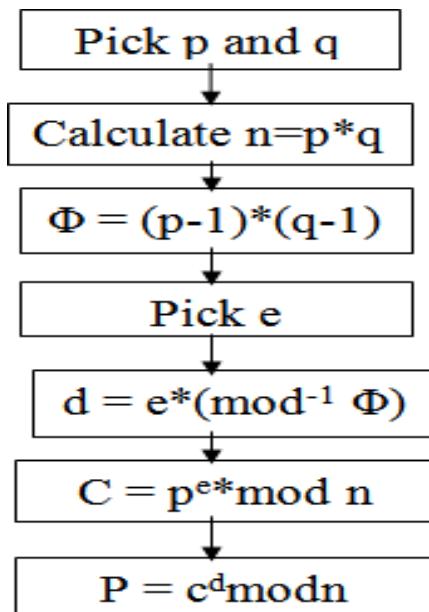
To write a C program to implement the RSA encryption algorithm.

DESCRIPTION:

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of them can be given to everyone. A basic principle behind RSA is the observation that it is practical to find three every 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 $e^{-1} \pmod{z}$.
- STEP-6:** The cipher text is computed as $\text{message}^e \pmod{n}$.
- STEP-7:** Decryption is done as $\text{cipher}^d \pmod{n}$.

PROGRAM: (RSA)

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

```

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

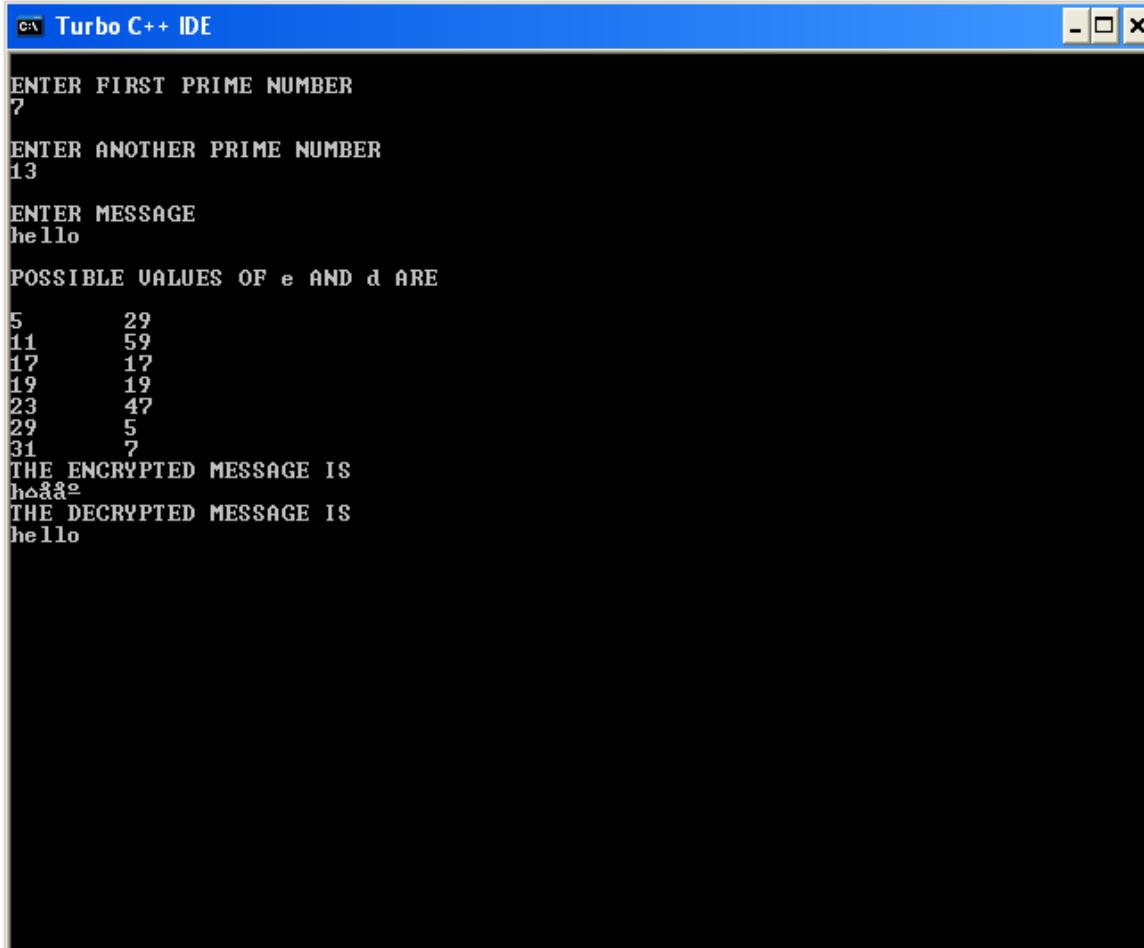
```

```

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

```

OUTPUT:



The screenshot shows a window titled "Turbo C++ IDE". Inside, the program's output is displayed in white text on a black background. The output is as follows:

```
ENTER FIRST PRIME NUMBER
7
ENTER ANOTHER PRIME NUMBER
13
ENTER MESSAGE
hello
POSSIBLE VALUES OF e AND d ARE
5      29
11     59
17     17
19     19
23     47
29     5
31     7
THE ENCRYPTED MESSAGE IS
h@#@#
THE DECRYPTED MESSAGE IS
hello
```

RESULT:

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

EX. NO: 2(C)

IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

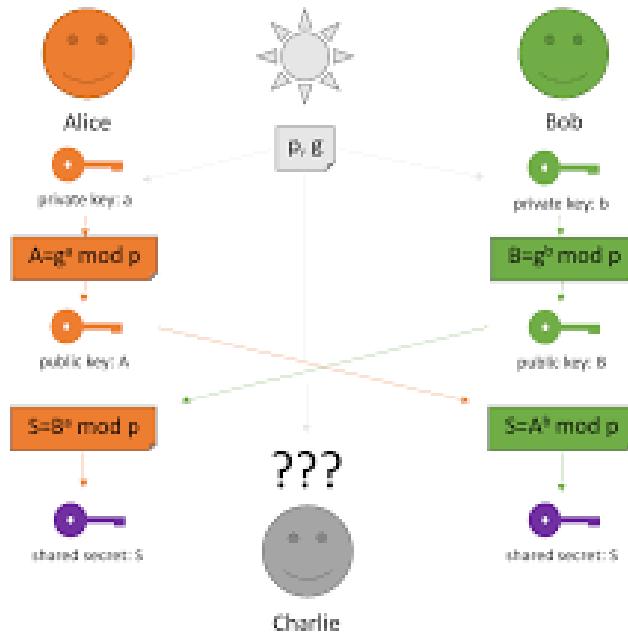
AIM:

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

DESCRIPTION:

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

EXAMPLE:



ALGORITHM:

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

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

STEP-3: Alice computes his secret key A as $g^a \text{ mod } p$.

STEP-4: Then Alice sends A to Bob.

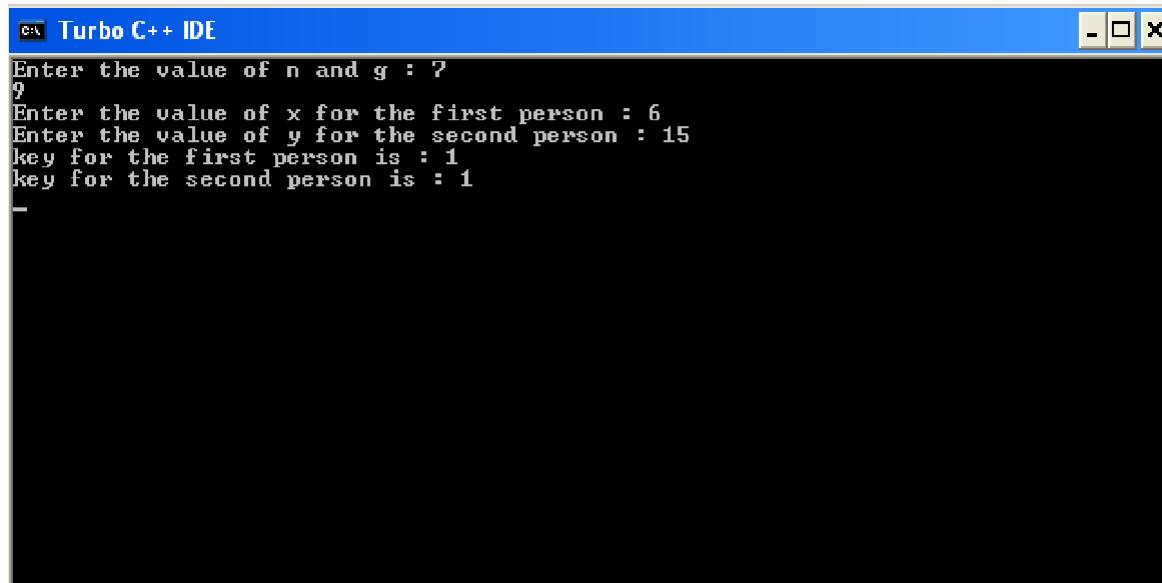
STEP-5: Similarly Bob also selects a public key b and computes his secret key as B and sends the same back to Alice.

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)

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

OUTPUT:



The screenshot shows a window titled "Turbo C++ IDE". Inside, the terminal output is displayed. It prompts for values of n and g (both 7), x (6), and y (15). It then calculates and prints the keys for both persons, both of which are 1.

```
Enter the value of n and g : 7
7
Enter the value of x for the first person : 6
Enter the value of y for the second person : 15
key for the first person is : 1
key for the second person is : 1
```

RESULT:

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

EX. NO: 2(D)

IMPLEMENTATION OF MD5

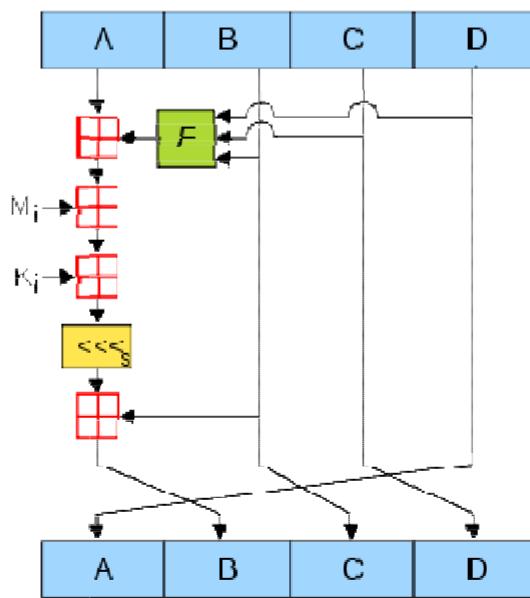
AIM:

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

DESCRIPTION:

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

EXAMPLE:



ALGORITHM:

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

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

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

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

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

PROGRAM:(MD5)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include<conio.h>
typedef union uwb
{
    unsigned w;
    unsigned char b[4];
} MD5union;

typedef unsigned DigestArray[4];
unsigned func0( unsigned abcd[] ){
    return ( abcd[1] & abcd[2] ) | (~abcd[1] & abcd[3]);}
unsigned func1( unsigned abcd[] ){
    return ( abcd[3] & abcd[1] ) | (~abcd[3] & abcd[2]);}
unsigned func2( unsigned abcd[] ){
    return abcd[1] ^ abcd[2] ^ abcd[3];}
unsigned func3( unsigned abcd[] ){
    return abcd[2] ^ (abcd[1] | ~ abcd[3]);}
typedef unsigned (*DgstFctn)(unsigned a[]);
unsigned *calctable( unsigned *k)
{
    double s, pwr;
    int i;
    pwr = pow( 2, 32 );
    for (i=0; i<64; i++)
    {
        s = fabs(sin(1+i));
        k[i] = (unsigned) ( s * pwr );
    }
    return k;
}
unsigned rol( unsigned r, short N )
{
    unsigned mask1 = (1<<N) -1;
    return ((r>>(32-N)) & mask1) | ((r<<N) & ~mask1);
}
```

```

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

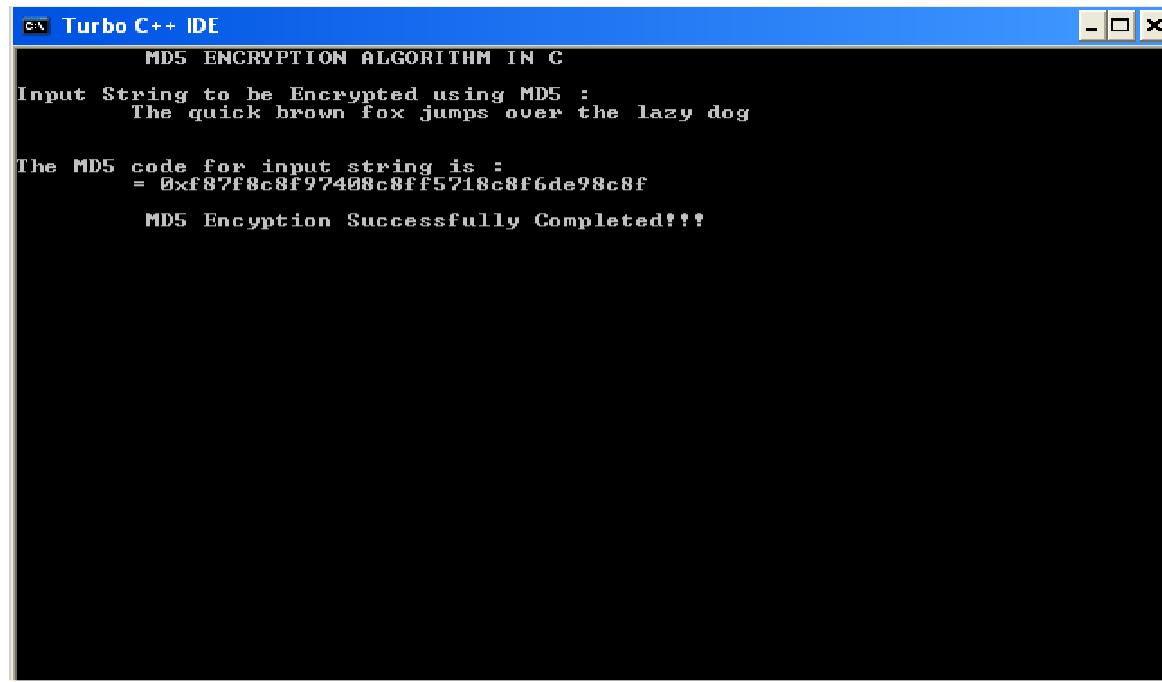
```

```

        for(q=0;q<4;q++) abcd[q] = h[q];
        for (p = 0; p<4; p++)
        {
            fctn = ff[p];
            rotn = rots[p];
            m = M[p]; o= O[p];
            for (q=0; q<16; q++)
            {
                g = (m*q + o) % 16;
                f = abcd[1] + rol( abcd[0]+ fctn(abcd)+k[q+16*p]
                + mm.w[g], rotn[q%4]);
                abcd[0] = abcd[3];
                abcd[3] = abcd[2];
                abcd[2] = abcd[1];
                abcd[1] = f;
            }
        }
        for (p=0; p<4; p++)
        h[p] += abcd[p];
        os += 64;
    }
    return h;
}

void main()
{
    int j,k;
    const char *msg = "The quick brown fox jumps over
    the lazy dog";
    unsigned *d = md5(msg, strlen(msg));
    MD5union u;
    clrscr();
    printf("\t MD5 ENCRYPTION ALGORITHM IN C \n\n");
    printf("Input String to be Encrypted using MD5 :
    \n\t%s",msg);
    printf("\n\nThe MD5 code for input string is: \n");
    printf("\t= 0x");
    for (j=0;j<4; j++){
    u.w = d[j];
    for (k=0;k<4;k++) printf("%02x",u.b[k]);
    }
    printf("\n");
    printf("\n\t MD5 Encryption Successfully
Completed!!!\n\n");
    getch();
    system("pause");
    getch();}
```

OUTPUT:



The screenshot shows a window titled "Turbo C++ IDE" with a blue header bar. The main area is black and contains white text. The text reads:

```
MD5 ENCRYPTION ALGORITHM IN C
Input String to be Encrypted using MD5 :
The quick brown fox jumps over the lazy dog

The MD5 code for input string is :
= 0xf87f8c8f97408c8ff5718c8f6de98c8f

MD5 Encryption Successfully Completed!!!
```

RESULT:

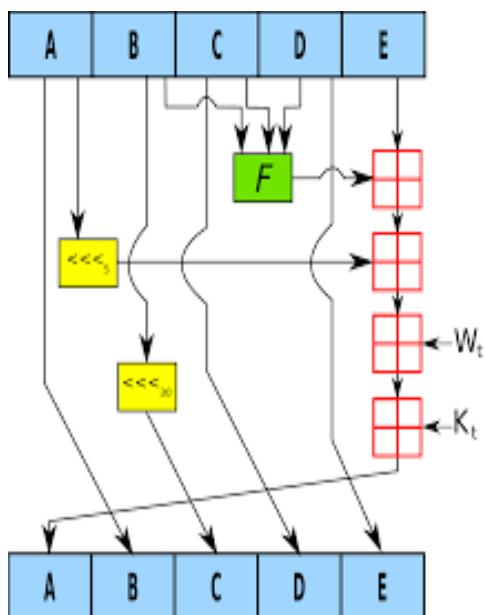
Thus the implementation of MD5 hashing algorithm had been implemented successfully using C.

AIM:

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

DESCRIPTION:

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function. SHA-1 produces a 160-bit hash value known as a message digest. The way this algorithm works is that for a message of size $< 2^{64}$ bits it computes a 160-bit condensed output called a message digest. The SHA-1 algorithm is designed so that it is practically infeasible to find two input messages that hash to the same output message. A hash function such as SHA-1 is used to calculate an alphanumeric string that serves as the cryptographic representation of a file or a piece of data. This is called a digest and can serve as a digital signature. It is supposed to be unique and non-reversible.

EXAMPLE:**ALGORITHM:**

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

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

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

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

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

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

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

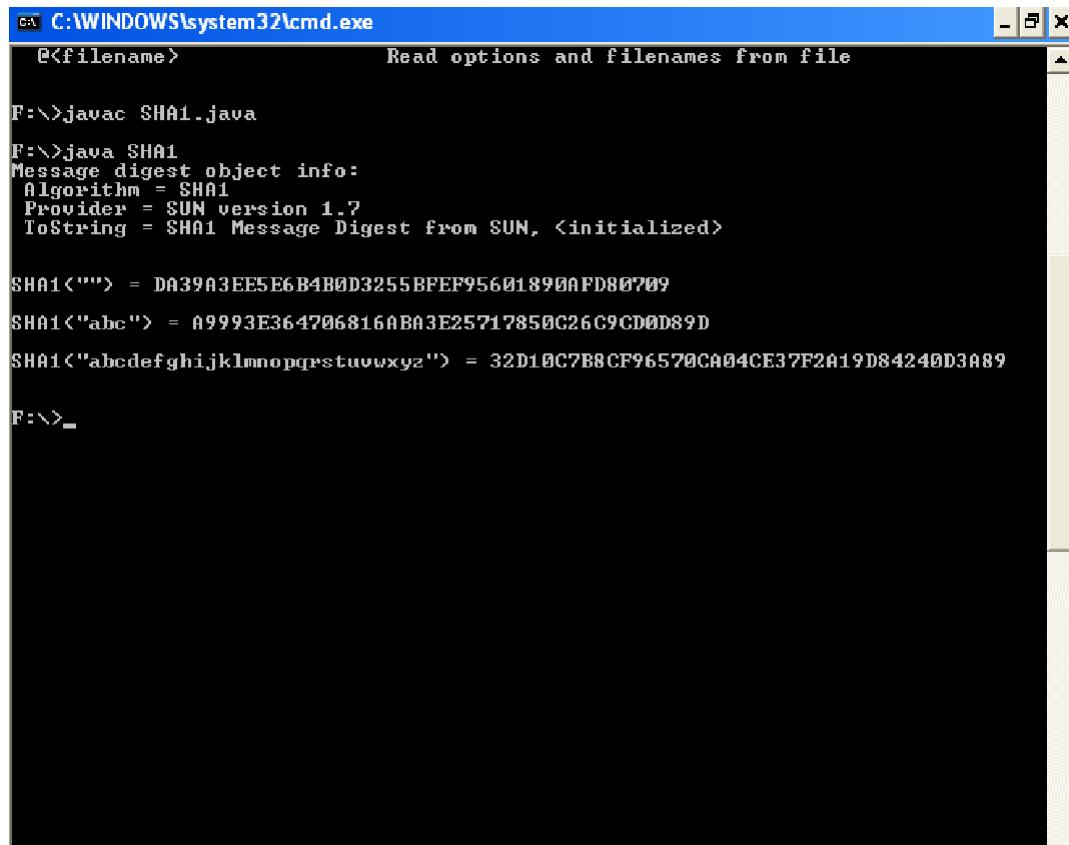
STEP-8: The blocks C and D are taken as the block D and E for the final output.

PROGRAM: (Secure Hash Algorithm)

```
import java.security.*;
public class SHA1 {
    public static void main(String[] a) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA1");
            System.out.println("Message digest object info: ");
            System.out.println(" Algorithm = " +md.getAlgorithm());
            System.out.println(" Provider = " +md.getProvider());
            System.out.println(" ToString = " +md.toString());
            String input = "";
            md.update(input.getBytes());
            byte[] output = md.digest();
            System.out.println();
            System.out.println("SHA1(\""+input+"\") = "
+bytesToHex(output));
            input = "abc";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println();
            System.out.println("SHA1(\""+input+"\") = "
+bytesToHex(output));
            input = "abcdefghijklmnopqrstuvwxyz";
            md.update(input.getBytes());
            output = md.digest();
            System.out.println();
            System.out.println("SHA1(\"" +input+"\") = "
+bytesToHex(output));
            System.out.println("");
        } catch (Exception e) {
            System.out.println("Exception: " +e);
        }
    }
    public static String bytesToHex(byte[] b)
    {
        char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6',
'7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'};
        StringBuffer buf = new StringBuffer();
        for (int j=0; j<b.length; j++) {
```

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

OUTPUT:



The screenshot shows a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The window title bar also displays the text '@<filename> Read options and filenames from file'. The command line shows the user navigating to a directory (F:\), compiling a Java source file ('javac SHA1.java'), and running the resulting class ('java SHA1'). The output of the program is displayed, showing the SHA-1 digest for three different inputs: an empty string, the string 'abc', and a longer string consisting of all lowercase letters from 'a' to 'z'. The output is as follows:

```
F:>javac SHA1.java
F:>java SHA1
Message digest object info:
Algorithm = SHA1
Provider = SUN version 1.7
ToString = SHA1 Message Digest from SUN, <initialized>

SHA1<""> = DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
SHA1<"abc"> = A9993E364706816ABA3E25717850C26C9CD0D89D
SHA1<"abcdefghijklmnopqrstuvwxyz"> = 32D10C7B8CF96570CA04CE37F2A19D84240D3A89

F:>
```

RESULT:

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

IMPLEMENTATION OF DIGITAL SIGNATURE STANDARD

AIM:

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

ALGORITHM:

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

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

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

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

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

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

PROGRAM: (Digital Signature Standard)

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

```

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

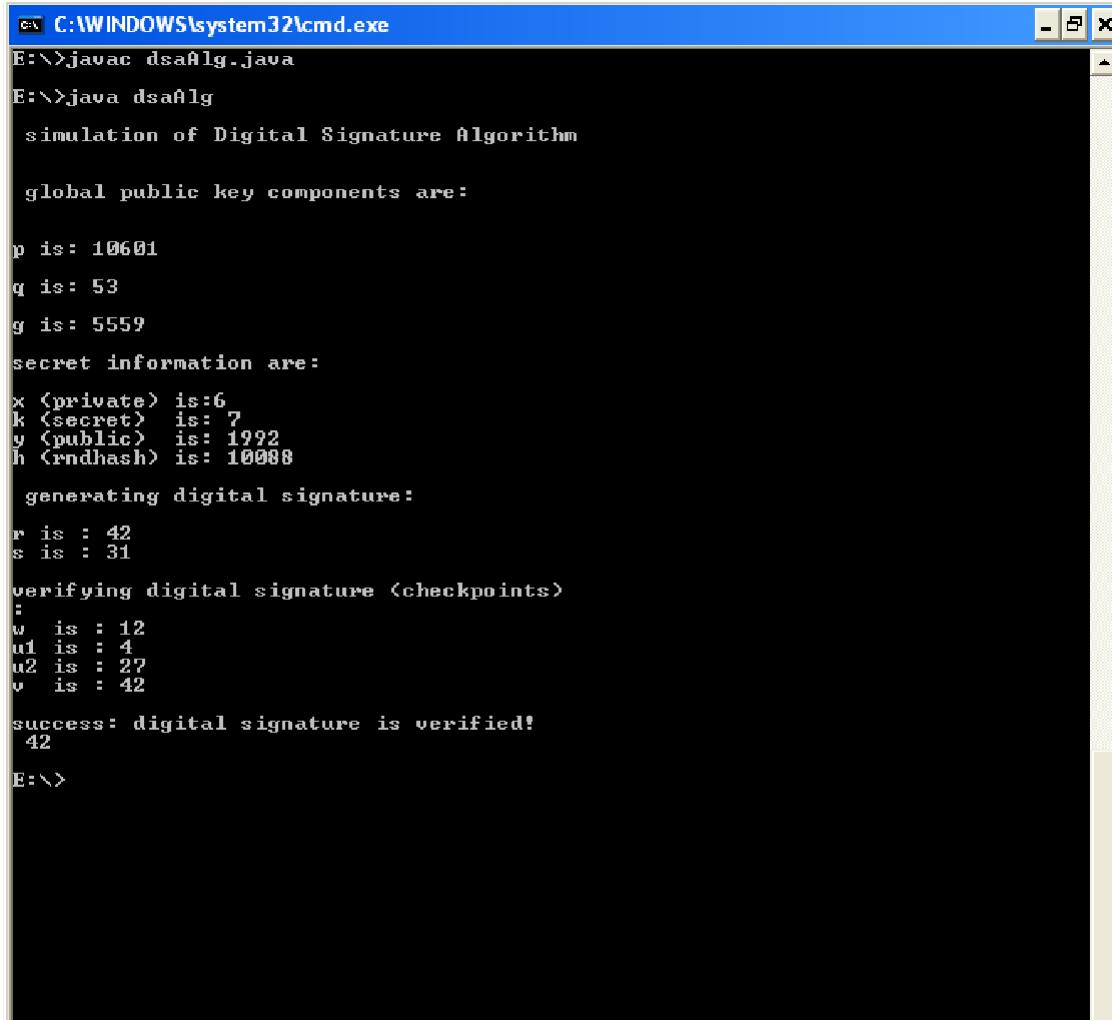
```

```

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

```

OUTPUT:



The screenshot shows a Windows Command Prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The command 'java dsaAlg' is run, which outputs the following text:

```

E:\>javac dsaAlg.java
E:\>java dsaAlg
simulation of Digital Signature Algorithm

global public key components are:

p is: 10601
q is: 53
g is: 5559

secret information are:

x <private> is:6
k <secret> is: 7
y <public> is: 1992
h <rndhash> is: 10088

generating digital signature:

r is : 42
s is : 31

verifying digital signature (checkpoints)
:
u  is : 12
u1 is : 4
u2 is : 27
v  is : 42

success: digital signature is verified!
42
E:\>

```

RESULT:

Thus, the simple Code Optimization techniques had been implemented successfully.

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:



[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](#)
[Compendium \(en\) 3.0.0](#)

EX. NO: 04 Calculate the message digest of a text using the SHA-1 algorithm.

AIM:

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

ALGORITHM:

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

PROGRAM:

```
sha1.java
import java.security.*;

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

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

        for (byte aB : b) {

            buf.append(hexDigit[(aB >> 4) & 0x0f]);
            buf.append(hexDigit[aB & 0x0f]);
        }
    }
}
```

```
        return buf.toString();
    }
}
```

OUTPUT:

Message digest object info:

Algorithm=SHA1

Provider=SUN version 12

ToString=SHA1 Message Digest from SUN, <initialized>

SHA1("")=DA39A3EE5E6B4B0D3255BFEF95601890AFD80709

SHA1("abc")=A9993E364706816ABA3E25717850C26C9CD0D89D

SHA1("abcdefghijklmnopqrstuvwxyz")=32D10C7B8CF96570CA04CE37F2A19D84240D3A89

RESULT:

thus, the *Secure Hash Algorithm (SHA-1)* has been implemented and the output has been verified successfully.

EX. NO: 05

**DEMONSTRATE HOW TO PROVIDE 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 we'll be using is GPG4Win

INSTALLING THE SOFTWARE:

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

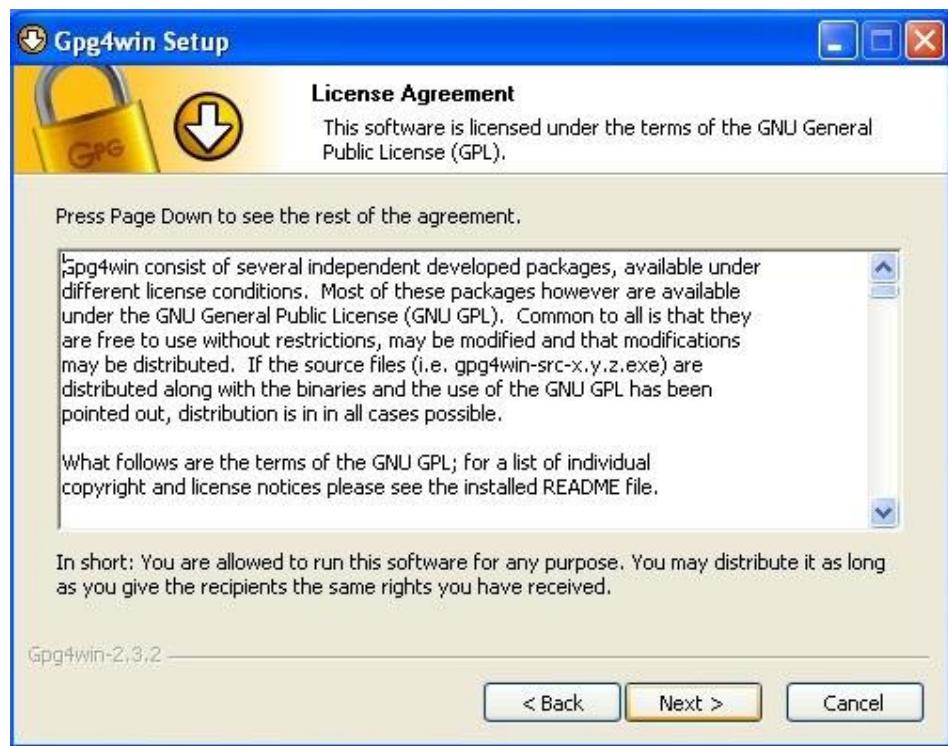
2. On the following screen, click the “Download Gpg4win” button.



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



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



- 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.



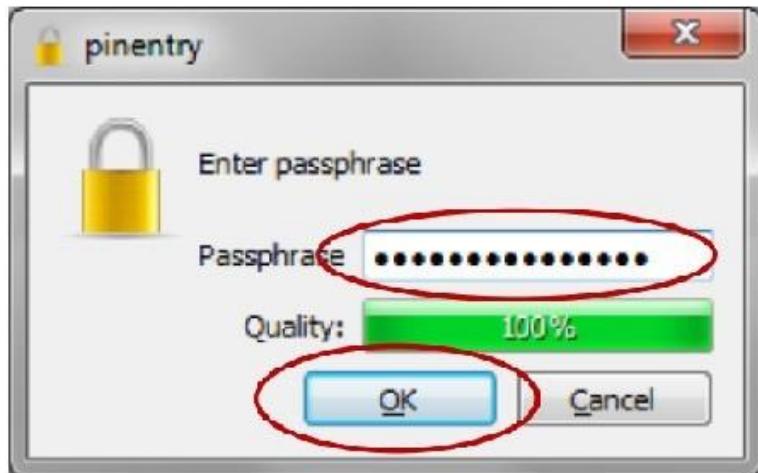
7. Specify where you want shortcuts to the software placed, then click the “Next” button.



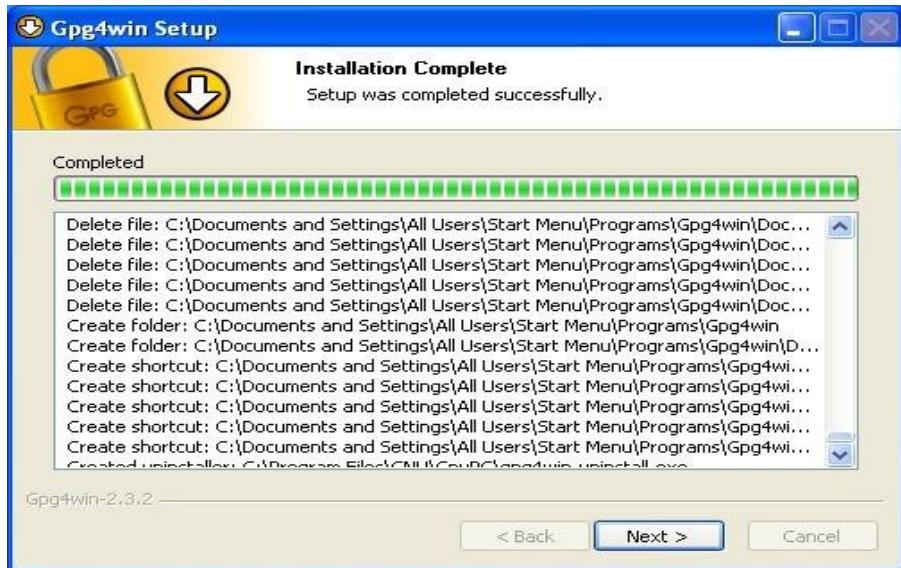
8. If you selected to have a GPG shortcut in your Start Menu, specify the folder in which it will be placed. The default “Gpg4win” is OK. Click the “Install” button to continue



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



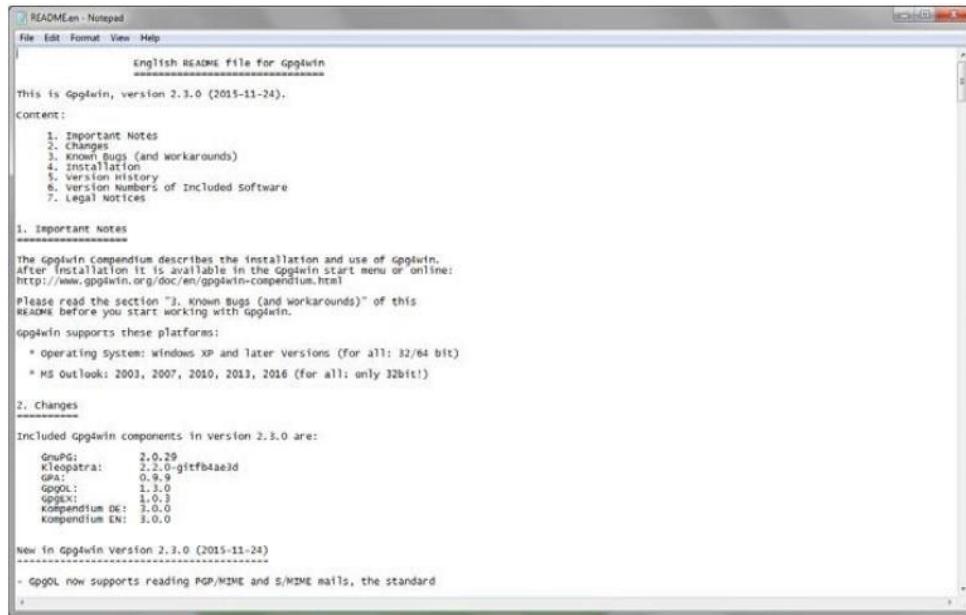
10. The installation process will tell you when it is complete. Click the “Next” button



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.



The screenshot shows a Windows Notepad window titled "READMEen - Notepad". The content is the English README file for Gpg4win version 2.3.0 (2015-11-24). The file starts with a header indicating it is an English README file for Gpg4win. It contains sections for "Content", "Important Notes", "Changes", and "Included Gpg4win components in version 2.3.0". It also includes a note about the "Kompendium DE" and "Kompendium EN" files. At the bottom, there is a section titled "New in Gpg4win Version 2.3.0 (2015-11-24)" which lists a single bullet point: "- GpgOL now supports reading PGP/MIME and S/MIME mails, the standard."

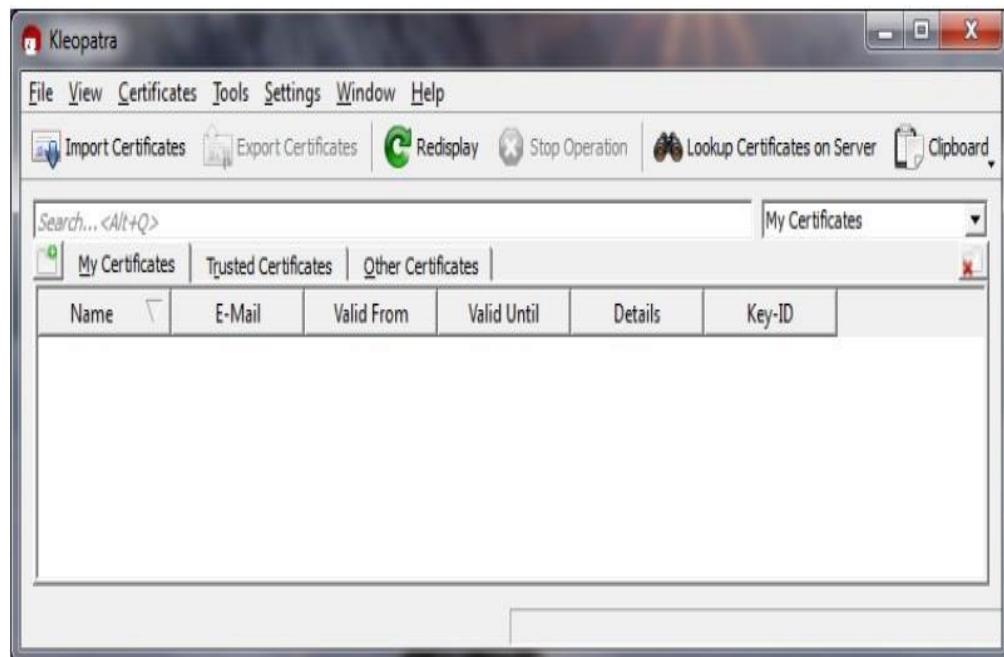
CREATING YOUR PUBLIC AND PRIVATE KEYS

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:

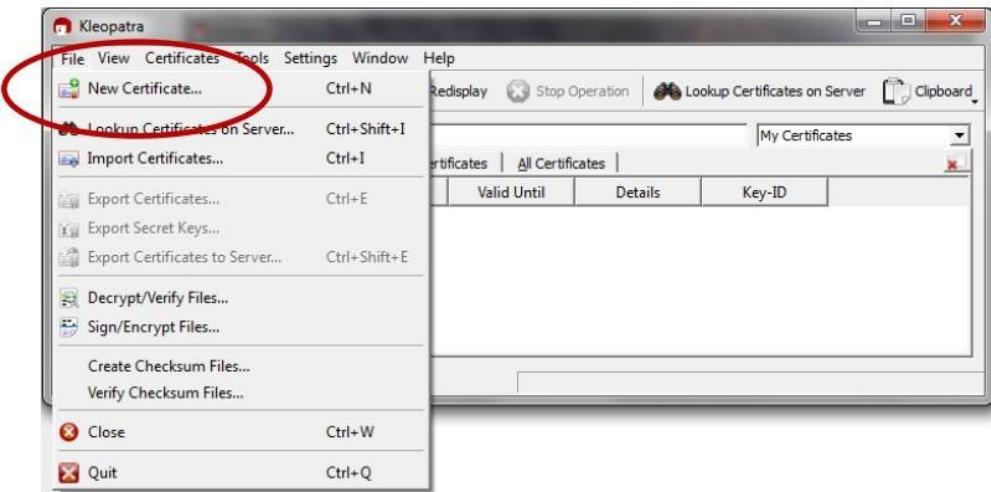
1. From your start bar, select the “Kleopatra” icon to start the Kleopatra certificate management software



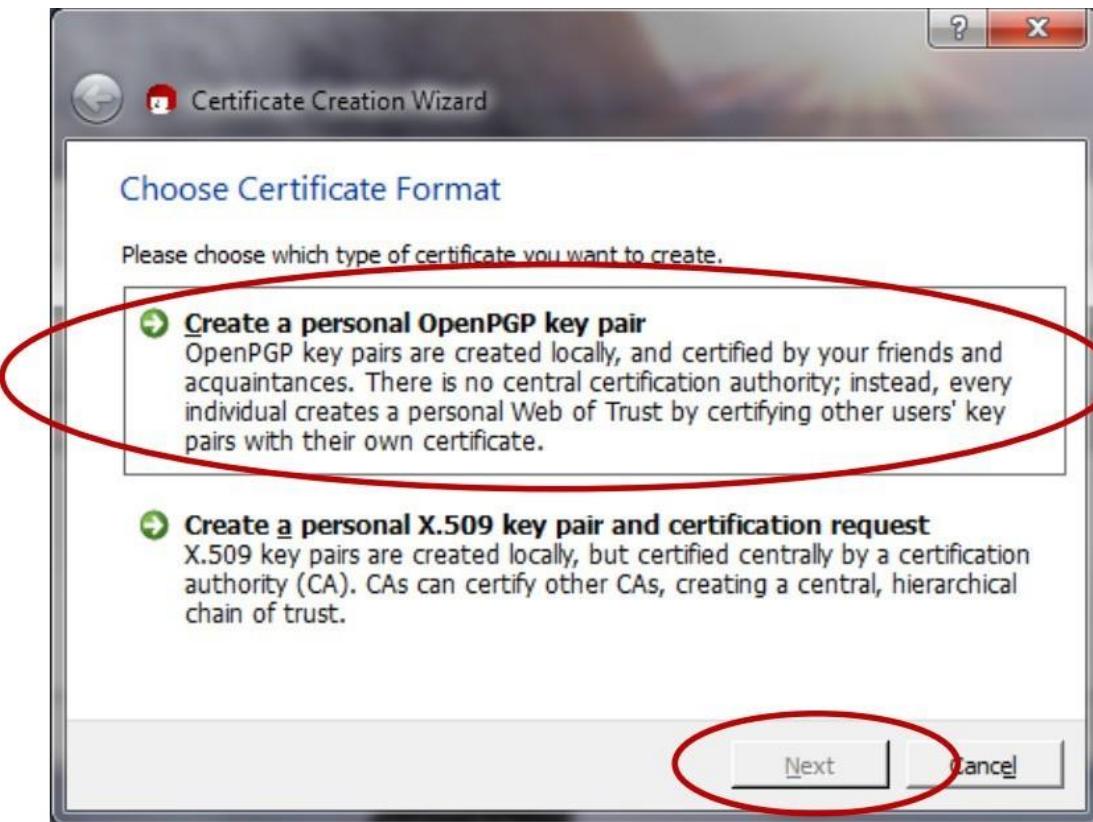
2. The following screen will be displayed



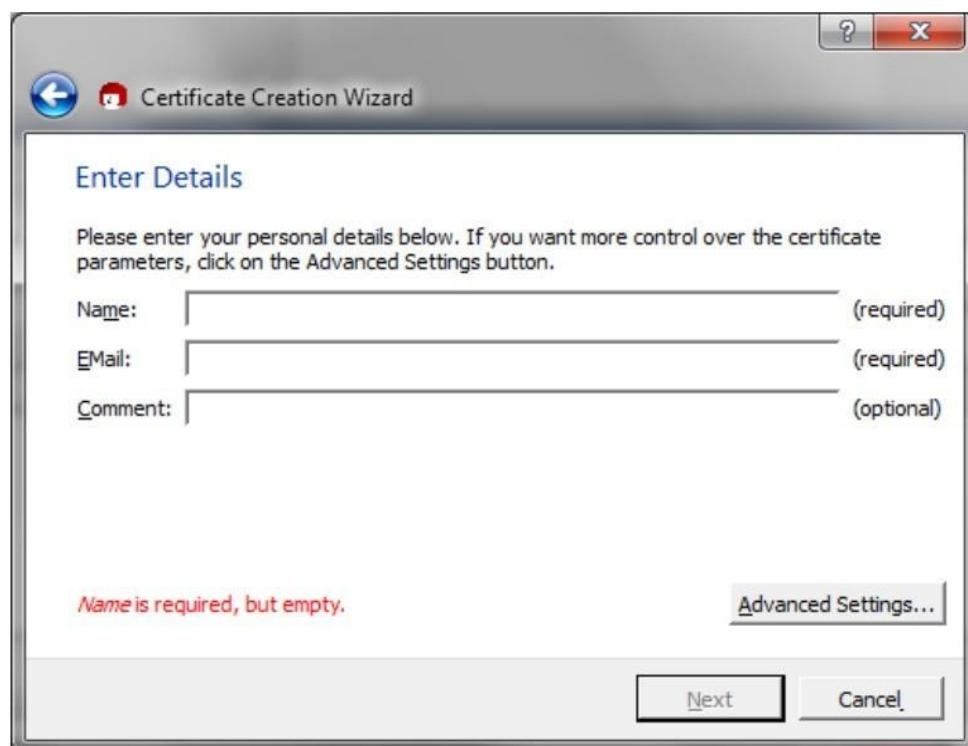
3. From the “File” dropdown, click on the “New Certificate” option



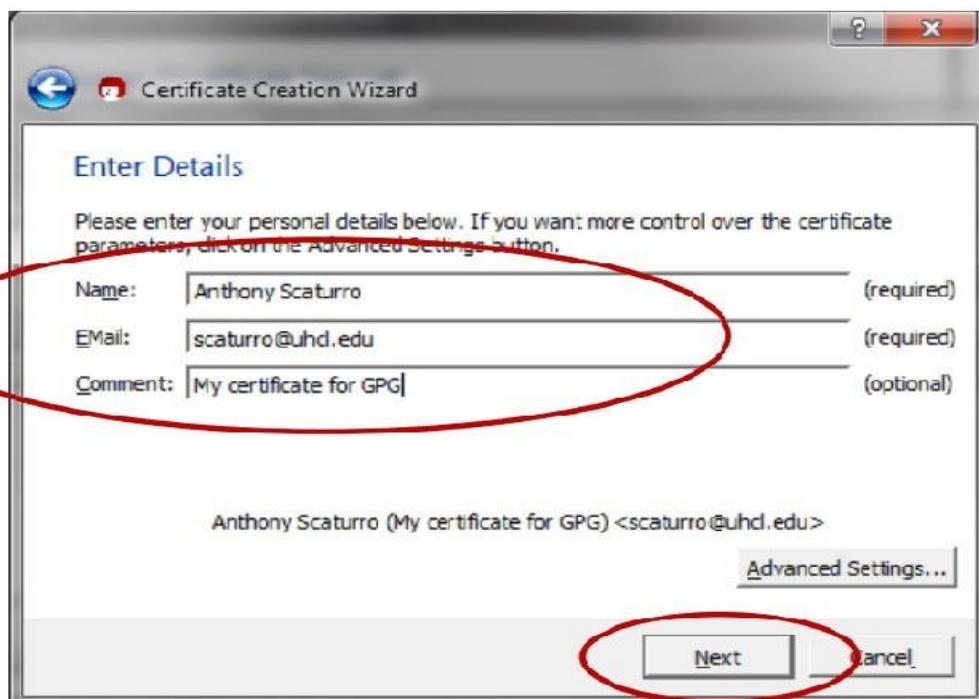
4. The following screen will be displayed. Click on “Create a personal OpenPGP key pair” and the “Next” button



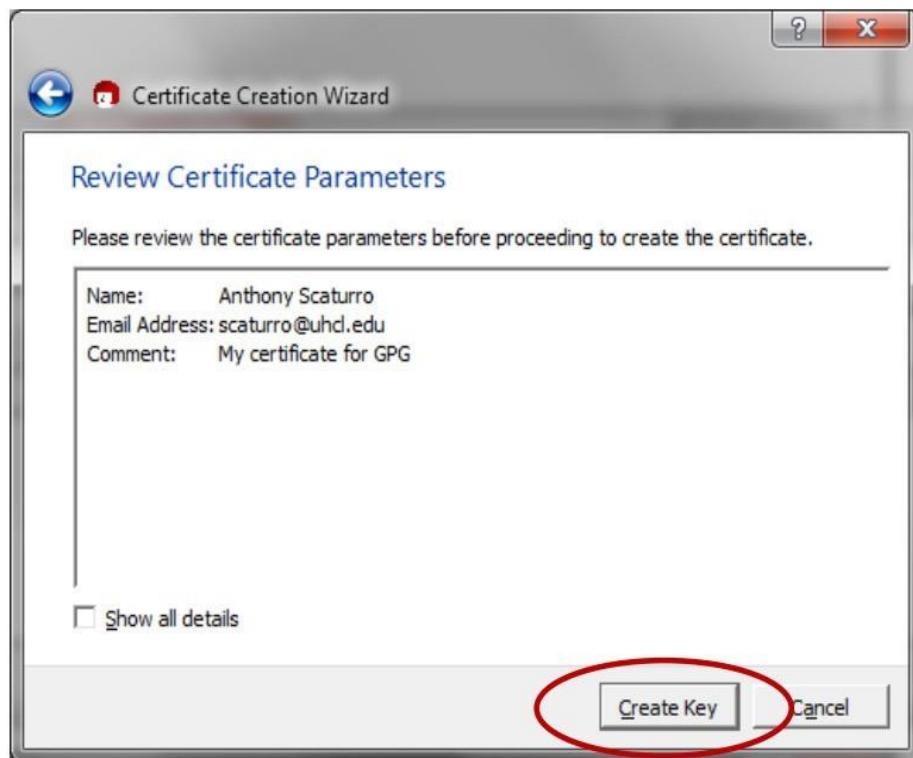
5. The Certificate Creation Wizard will start and display the following:



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



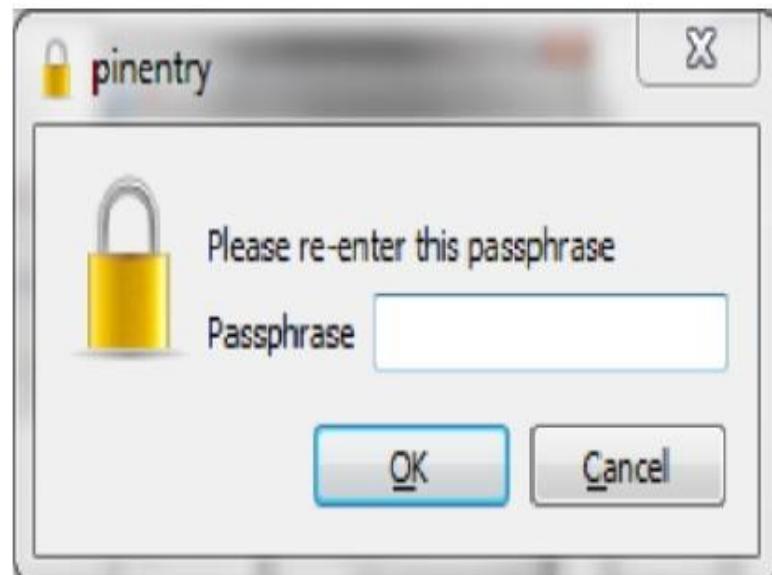
8. You will be asked to enter a passphrase



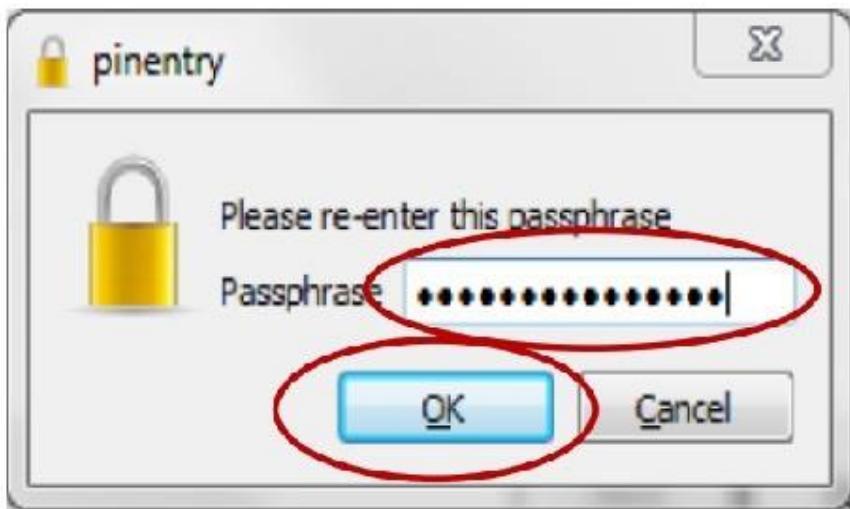
9. The passphrase should follow strong password standards. After you've entered your passphrase, click the “OK” button.



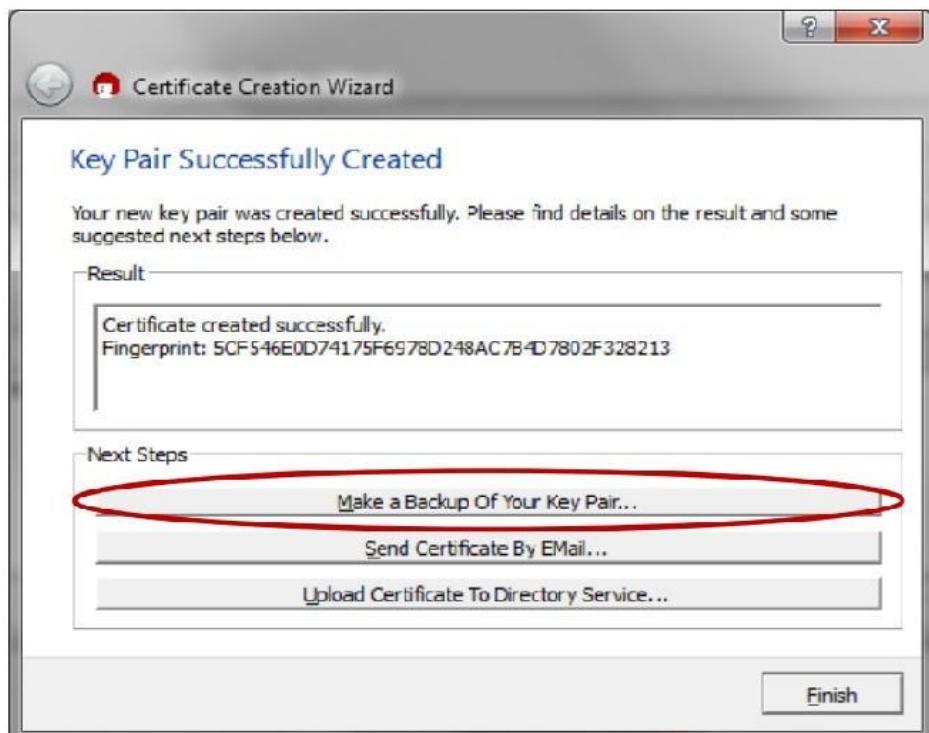
10. You will be asked to re-enter the passphrase



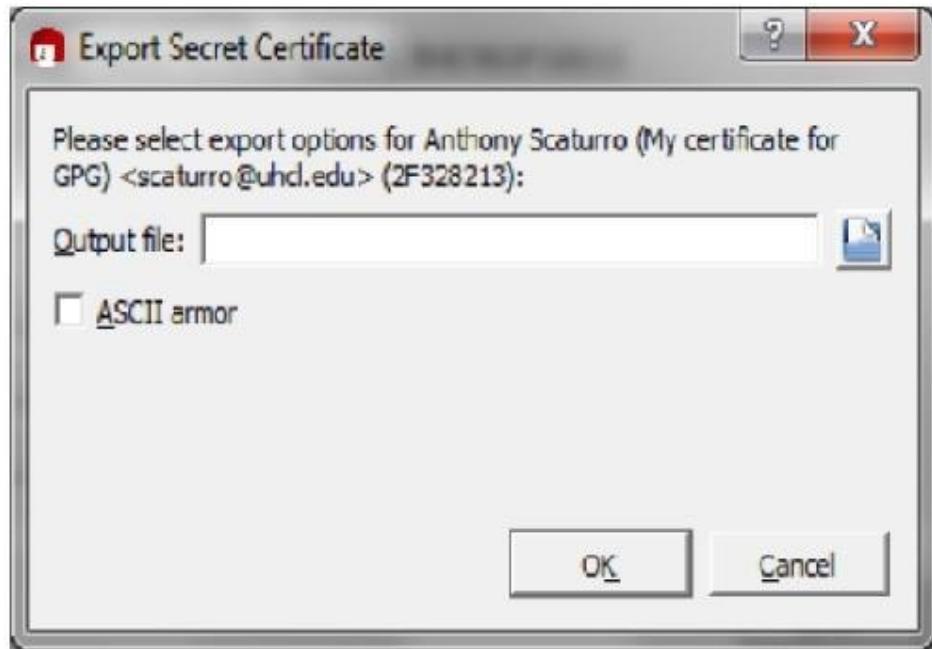
11. Re-enter the passphrase value. Then click the “OK” button. If the passphrases match, the certificate will be created.



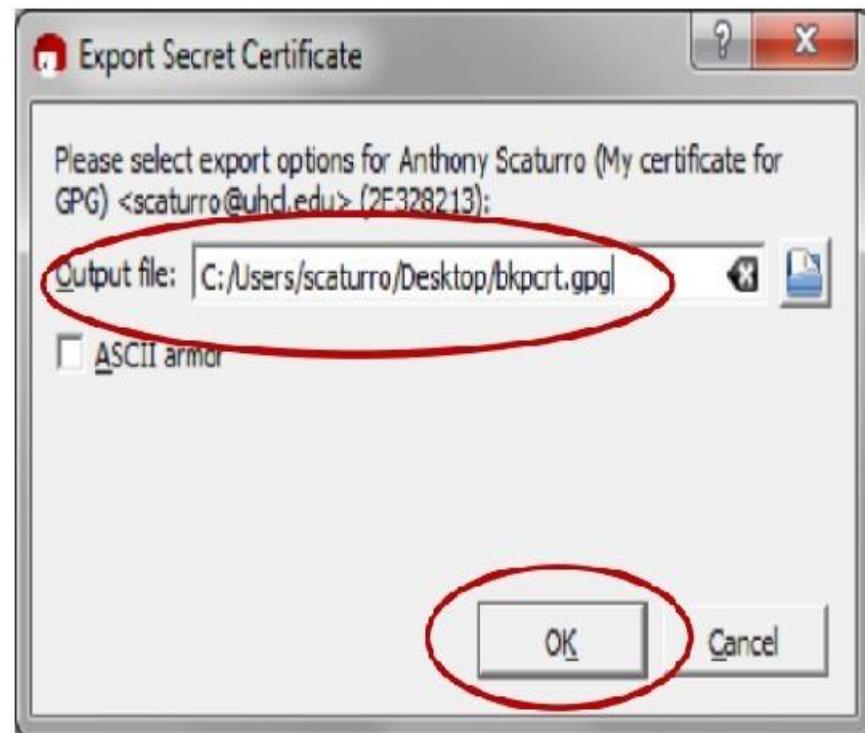
12. Once the certificate is created, the following screen will be displayed. You can save a backup of your public and private keys by clicking the “Make a backup Of Your Key Pair” button. This backup can be used to copy certificates onto other authorized computers.



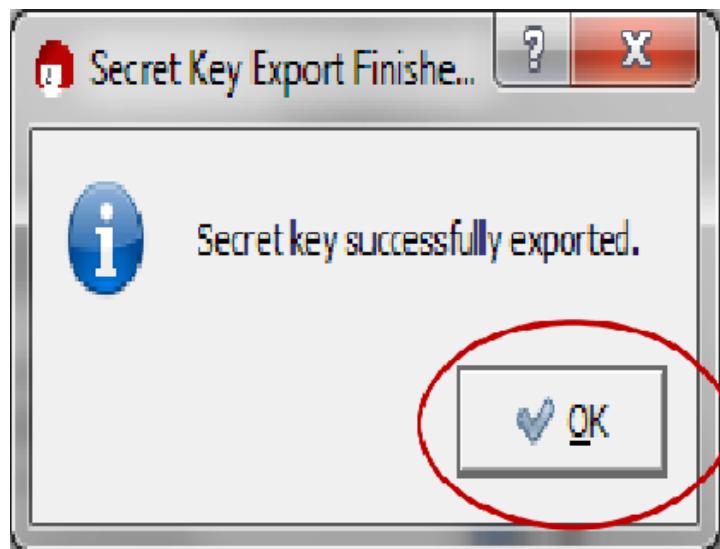
13. If you choose to backup your key pair, you will be presented with the following screen:



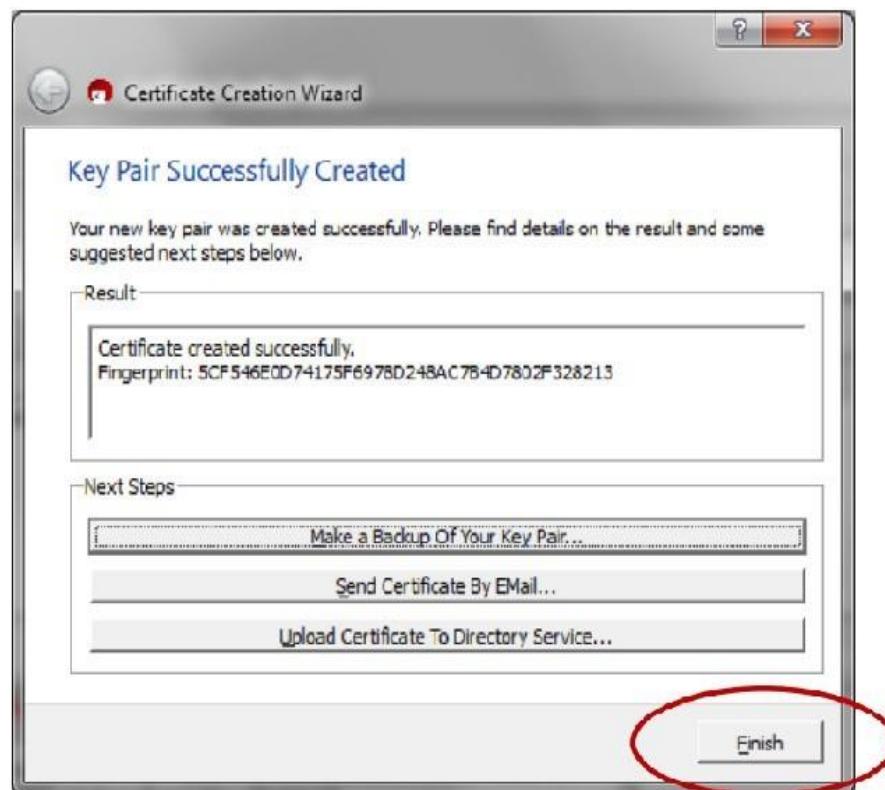
14. Specify the folder and name the file. Then click the “OK” button.



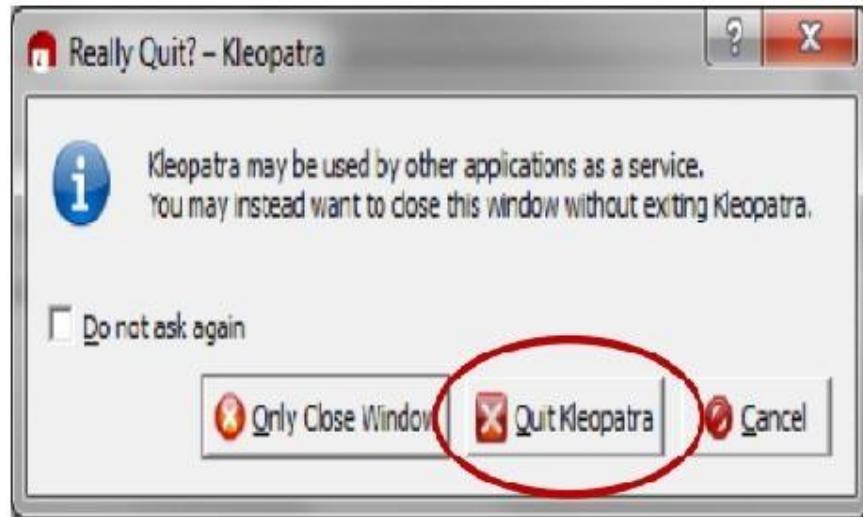
15. After the key is exported, the following will be displayed. Click the “OK” button.



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



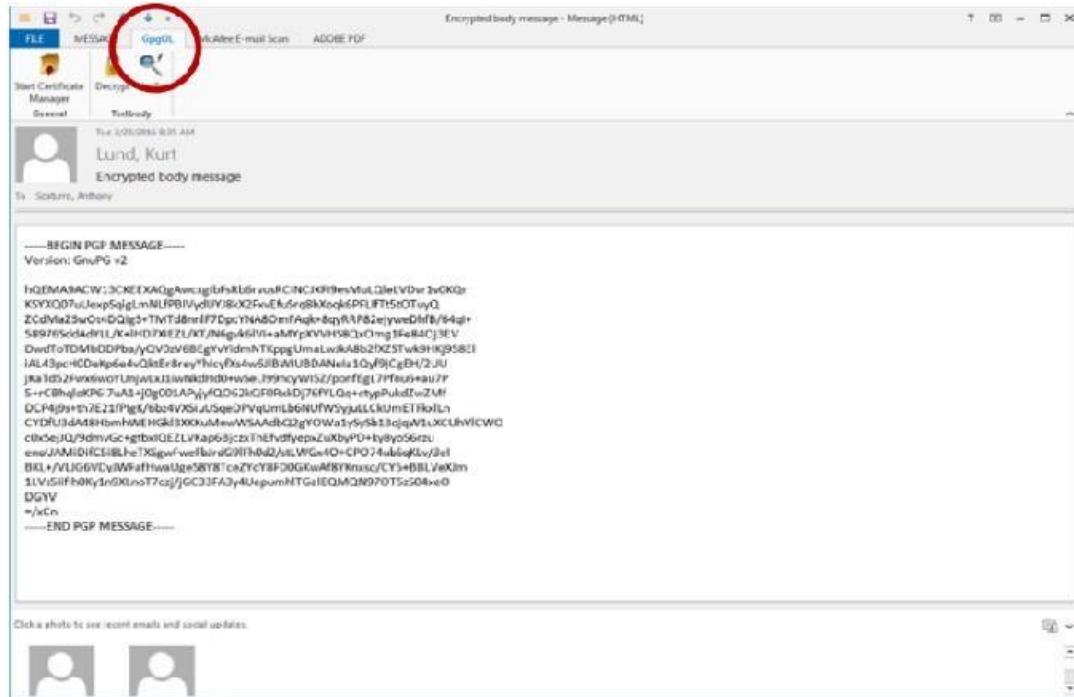
17. Before the program closes, you will need to confirm that you want to close the program by clicking on the “Quit Kleopatra” button



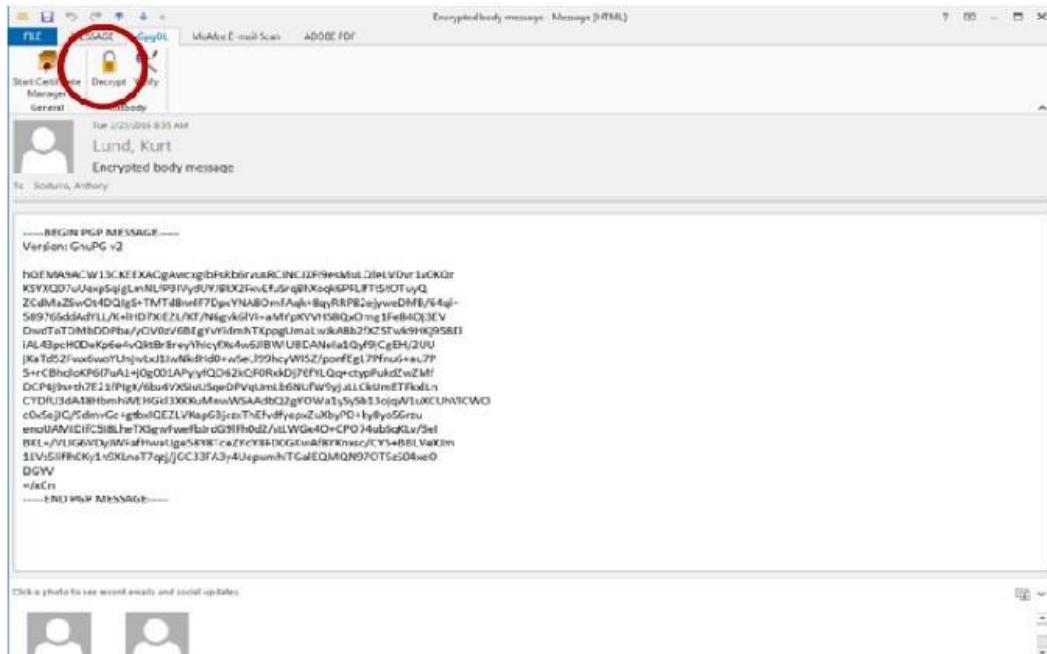
DECRYPTING AN ENCRYPTED E-MAIL THAT HAS BEEN SENT TO YOU:

1. Open the e-mail message

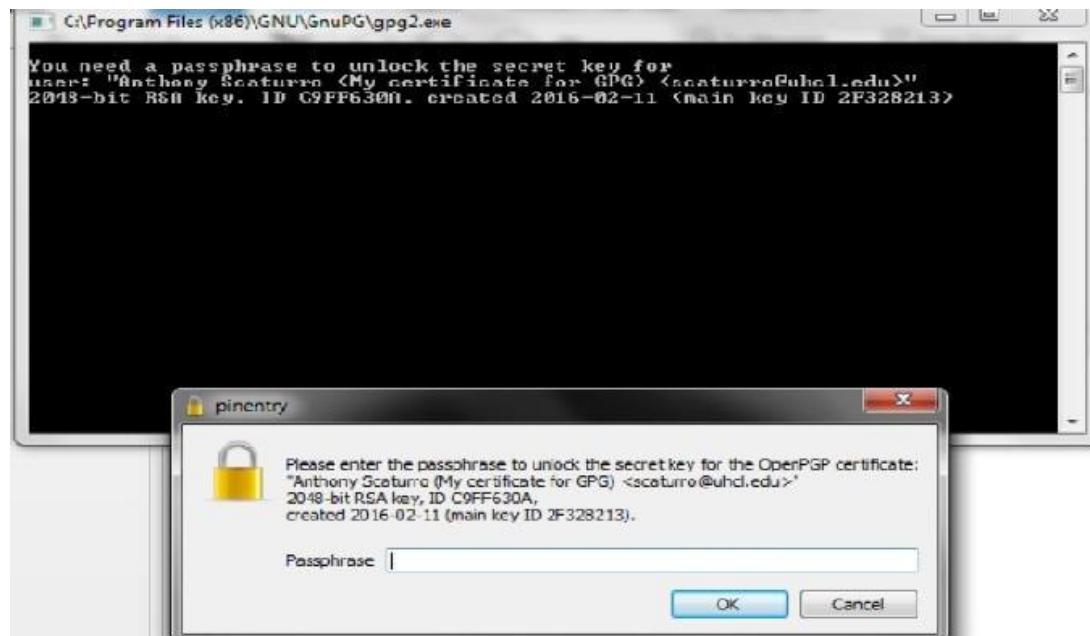
2. Select the GpgOL tab



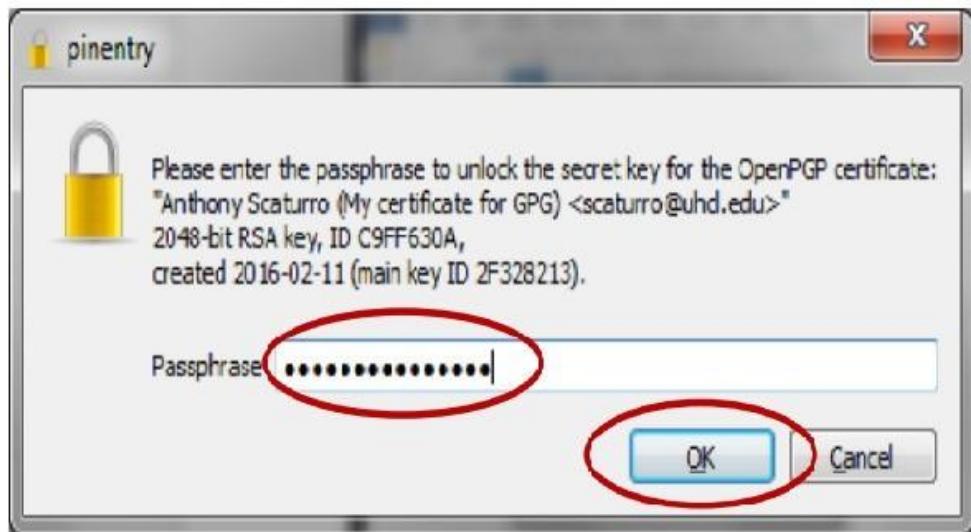
3. Click the “Decrypt” button



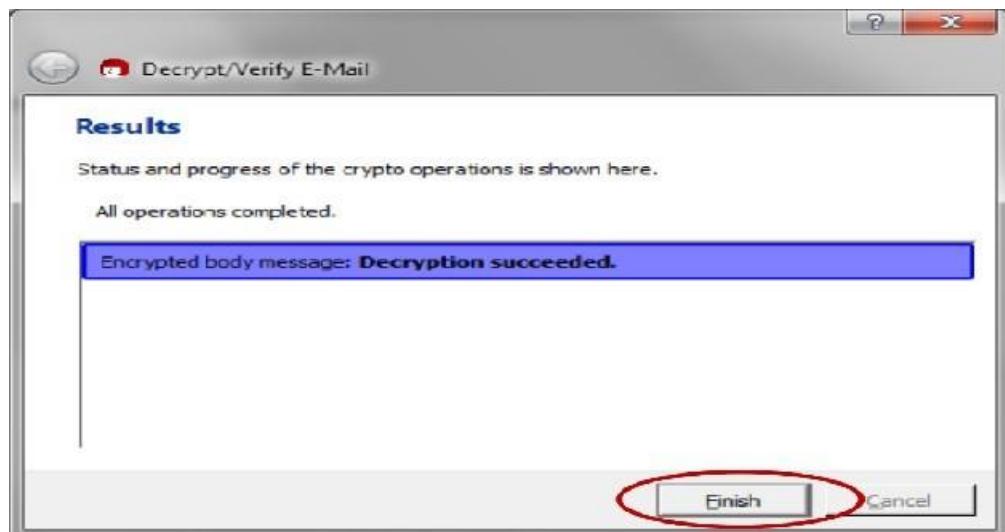
4. A command window will open along with a window that asks for the Passphrase to your private key that will be used to decrypt the incoming message.



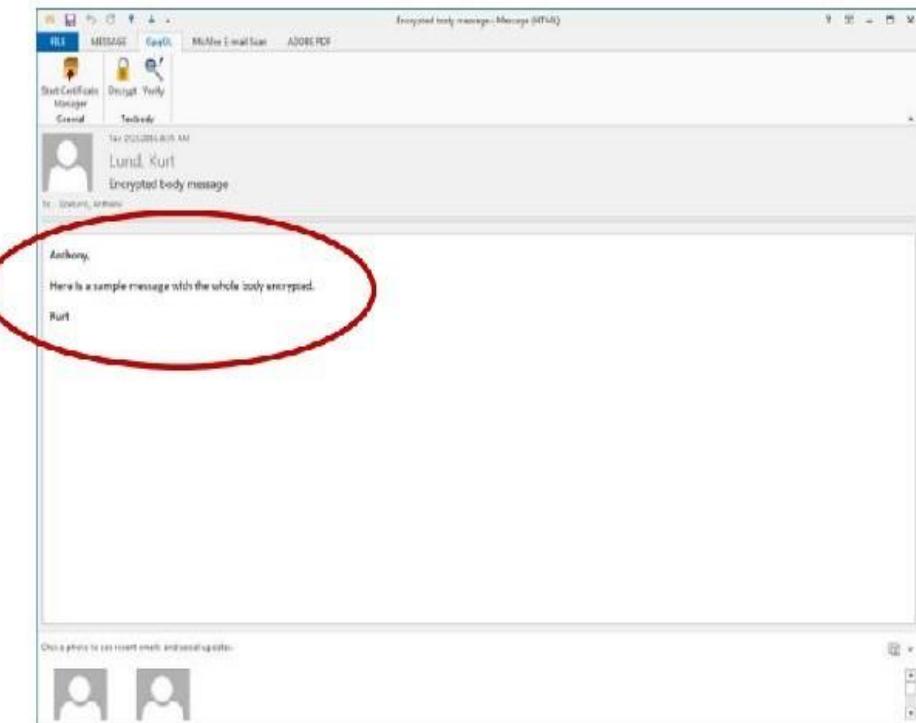
5. Enter your passphrase and click the “OK” button



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: 06

Setup a honey pot and monitor the honeypot on network (KF Sensor).

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 windowsboot.

STEP-4: Click Next to setup wizard.

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

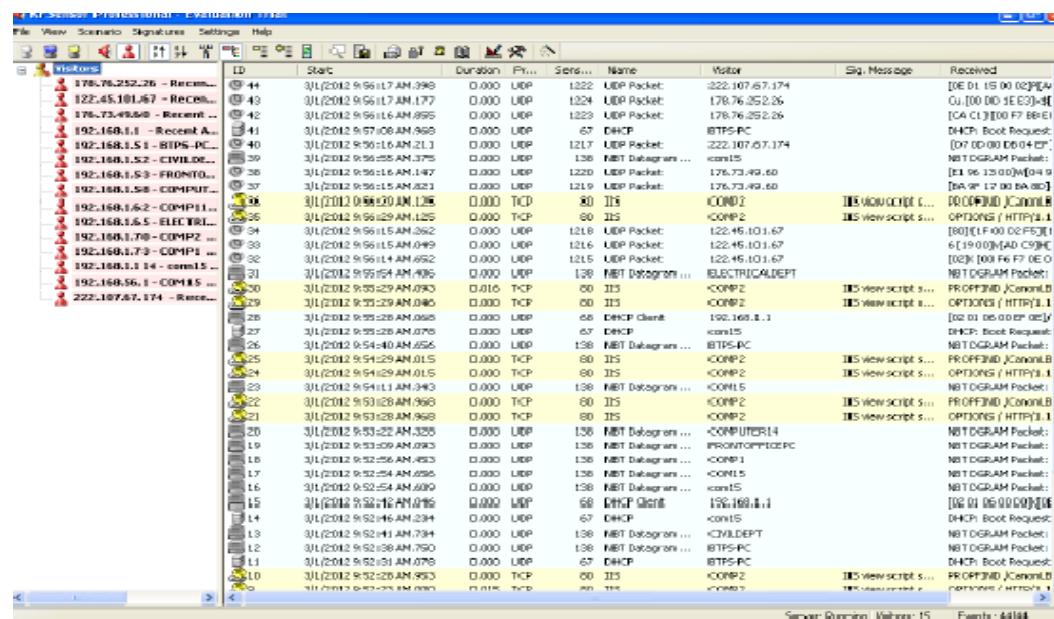
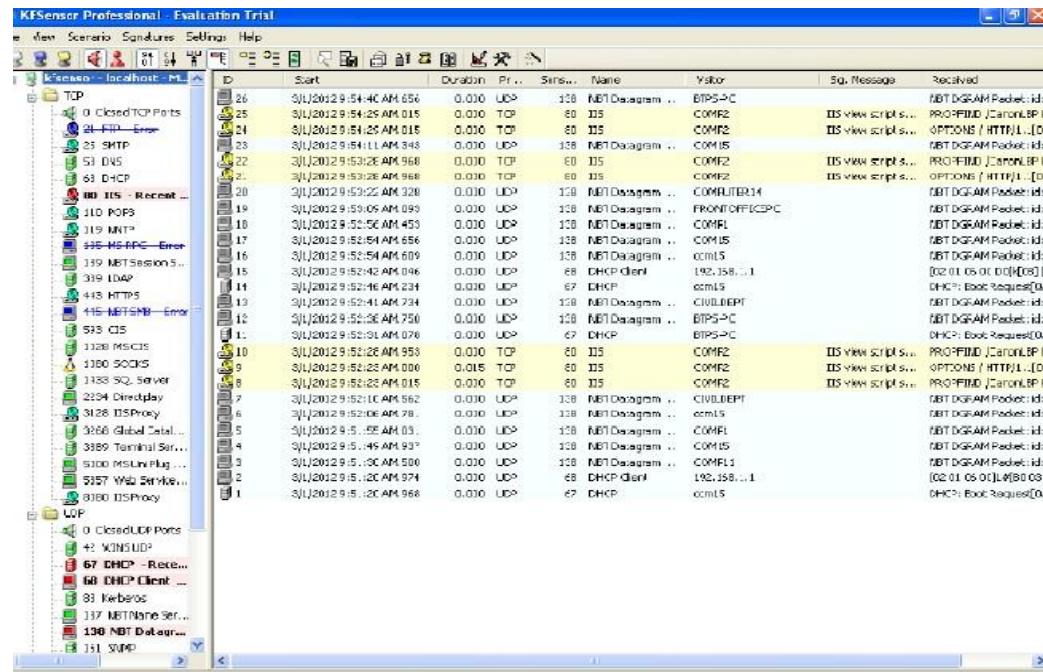
STEP-6: "Send the email and Send from email", enter the ID and Click Next.

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:



Server: RunTime: 10 hours: 15 Events: 44144

Visitors	Duration	Pr...	Sens...	Name	Visitor	Sig. Message	Received
0.0.0.0 - conn15 - Rec...	2012-9-09 09:29:45.903	0.000	UDP	1523	UDP Packet	MICROSOFT-65666EA	01D7100E[06 80]#A8E A6 8...
24.51.76.132 - Rec...	2012-9-09 09:29:45.903	0.000	UDP	1522	UDP Packet	MICROSOFT-65666EA	HHS 1A.00 C976F6PE[[AO:0F:16 06 ...
31.121.81.158 - Rec...	2012-10-10 02:29 AM...	0.016	TCP	90	IIS	COMP2	DS View script S...
46.43.20.51.2 - Rec...	2012-10-10 02:29 AM...	0.000	TCP	90	IIS	COMP2	PROPFIND /anon/0P-HTTP/1.1[D0 ...
46.63.5.244 - MRBB...	2012-9-09 09:29:45.903	0.000	UDP	1520	UDP Packet	70.97.105.133	DS View script S...
46.102.77.223 - Rec...	2012-9-09 09:29:45.907	0.000	UDP	1518	UDP Packet	70.97.105.133	OPTIONS / HTTP/1.1[D0 0A]trans4...
46.198.169.132 - Rec...	2012-9-09 09:29:45.900	0.000	UDP	1516	UDP Packet	112.205.4.149.pclt.net	HTTP/1.1[D0 0C 9E AB:A4:0B]=[07]6[07...
46.24.0.118.5 - Rec...	2012-9-09 09:29:45.905	0.000	UDP	1515	UDP Packet	112.205.4.149.pclt.net	(C0)=[00]6[04]<[00][C5 09 0F A...
49.248.131.190 - gmt...	2012-9-09 09:29:45.900	0.000	UDP	1514	UDP Packet	70.111.104.187	[LE 0D 10 00 0A][D0]0[D0][A3][C0]...
50.113.151.26 - Rec...	2012-9-09 09:27 AM...	0.000	UDP	1513	UDP Packet	70.111.104.187	[0F 82][D0][E7 07 C7 9D AF][EF]...
50.123.151.204 - Rec...	2012-9-09 09:27 AM...	0.000	UDP	1510	UDP Packet	109.173.82.171.next...	VC9E 0F 00][C9 F6[0E]B[04]A[C...
50.126.122.92 - Rec...	2012-9-09 09:26:45.967	0.000	UDP	1507	UDP Packet	CAMERAS	[0E]F8 11.00 E9-CF[16]15 C4 C4 ...
50.144.55.229 - Rec...	2012-9-09 09:26:45.965	0.000	UDP	1509	UDP Packet	SWD.mnl.0.in	[AU]7[00][EB]19 0E B4 D1 1C 98 ...
50.148.26.211 - 08...	2012-9-09 09:26:45.961	0.000	UDP	1508	UDP Packet	swtp.mnl.0.in	[0E 81]14[00][B9 97 DE CS 204 D4 E...
61.348.107.100 - Rec...	2012-9-09 09:25:45.329	0.000	UDP	1506	UDP Packet	CAMERAS	<[13 00]7[FF]F0-94 AD[C6 1C FF]S[...
71.43.42.154 - ITSP...	2012-9-09 09:25:45.300	0.000	UDP	1504	UDP Packet	broadband-vetus.com.br	[0A 9F 37 00 0A BD]=[16 C1 E1 FF]H...
72.238.93.227 - Rec...	2012-9-09 09:24:45.816	0.000	UDP	1503	UDP Packet	broadband-vetus.com.br	R-[00]B[00]A[00]7[00]A[00]B[00]A[...
76.60.251.50 - Rec...	2012-9-09 09:24:45.840	0.000	UDP	1502	UDP Packet	customer1441.51.msg...	[F1 15 00 00 0A][D0]0[D0][A3][C0]...
78.97.165.133 - Rec...	2012-9-09 09:13 AM...	0.000	UDP	1494	UDP Packet	OC_KOMP4	F-[17 00]F1[198][0E2]W[D1]B9 ...
78.97.186.74 - Rec...	2012-9-09 09:13 AM...	0.000	UDP	1493	UDP Packet	OC_KOMP4	[A5 15 15 00]CEM[01A][E7]7[CB ...
78.11.104.107 - Rec...	2012-10-09 09:29 AM...	0.000	TCP	80	IIS	COMP2	DS view script S...
79.125.89.51 - IP@...	2012-9-09 09:26:45.932	0.000	UDP	1492	UDP Packet	112.206.183.74.pclt...	PROPFIND /anon/0P-HTTP/1.1[D0 ...
79.170.188.185 - DC...	2012-9-09 09:26:45.970	0.000	UDP	1491	UDP Packet	112.206.183.74.pclt...	DS View script S...
82.00.136.160 - CA...	2012-9-09 09:11 AM...	0.000	UDP	1490	UDP Packet	128-166-14-215.static...	OPTIONS / HTTP/1.1[D0 0A]trans4...
84.240.34.146 - AEL...	2012-9-09 09:11 AM...	0.000	UDP	1489	UDP Packet	129-166-14-215.static...	HTTP/1.1[D0 0C 9E AB:A4:0B]=[07]6[07...
85.122.41.134 - Rec...	2012-9-09 09:10 AM...	0.000	UDP	1484	UDP Packet	109.74-127-249.ngn...	(C0)=[00]6[04]<[00][C5 09 0F A...
85.204.143.139 - Rec...	2012-9-09 09:09 AM...	0.000	UDP	1483	UDP Packet	109.74-127-249.ngn...	[C5][16 00 9C]IP9 EDE4-151[F5]...
87.69.221.227 - DSL...	2012-9-09 09:08 AM...	0.000	UDP	1480	UDP Packet	ACER	[F1 81]4[00]6[04]F2[0F A4]F0[00] ...
88.60.107.140 - H0...	2012-9-09 09:08 AM...	0.000	UDP	1479	UDP Packet	ACER	AW/[00]8D 9D][C1 04 01]C1 8E[00]...
88.208.268.192 - ce...	2012-9-09 06:45 AM...	0.000	UDP	1472	UDP Packet	CHANGEME1	SELECT[00] 0D 02 B1[J0]C1 94[00]B...
89.43.159.230 - Rec...	2012-9-09 06:45 AM...	0.000	UDP	1471	UDP Packet	CHANGEME1	[0E 82][00]F6 F7 0E 04 C7 16]D04-A...
92.134.113.151 - Rec...	2012-9-09 06:45 AM...	0.000	UDP	1470	UDP Packet	70.97.160.74	[FD 82 1B 00]4[00][E0 D1 A9]E5 ...
93.138.230.195 - Rec...	2012-9-09 06:45 AM...	0.000	UDP	1469	UDP Packet	70.97.160.74	[A5 1C][00]F0[00]B[00]RE[0D]E[00]B...
		n.nnn	None	1468	UDP Packet	109.74-127-249.ng...	From: 10.10.10.100 To: 10.10.10.100

RESULT:

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

EX. NO: 07

Installation of rootkits and study about the variety of options.

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 system's 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.

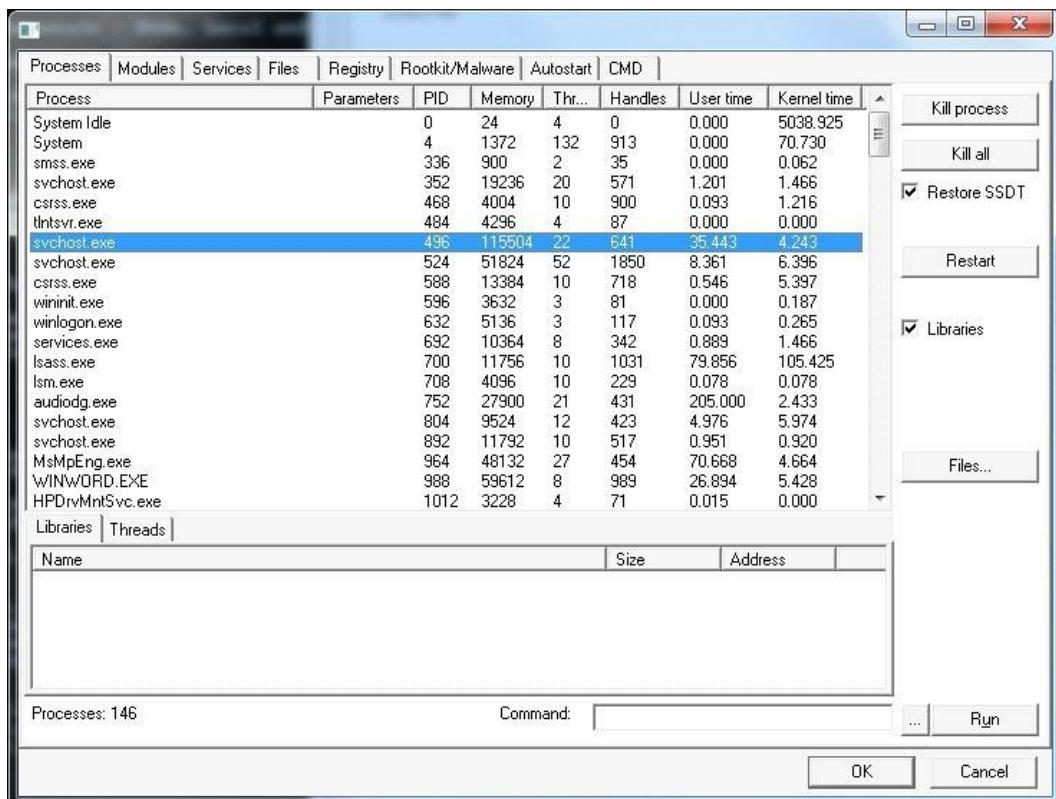
STEP-7: Registry displays Hkey_Current_user and Hkey_Local_Machine.

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

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

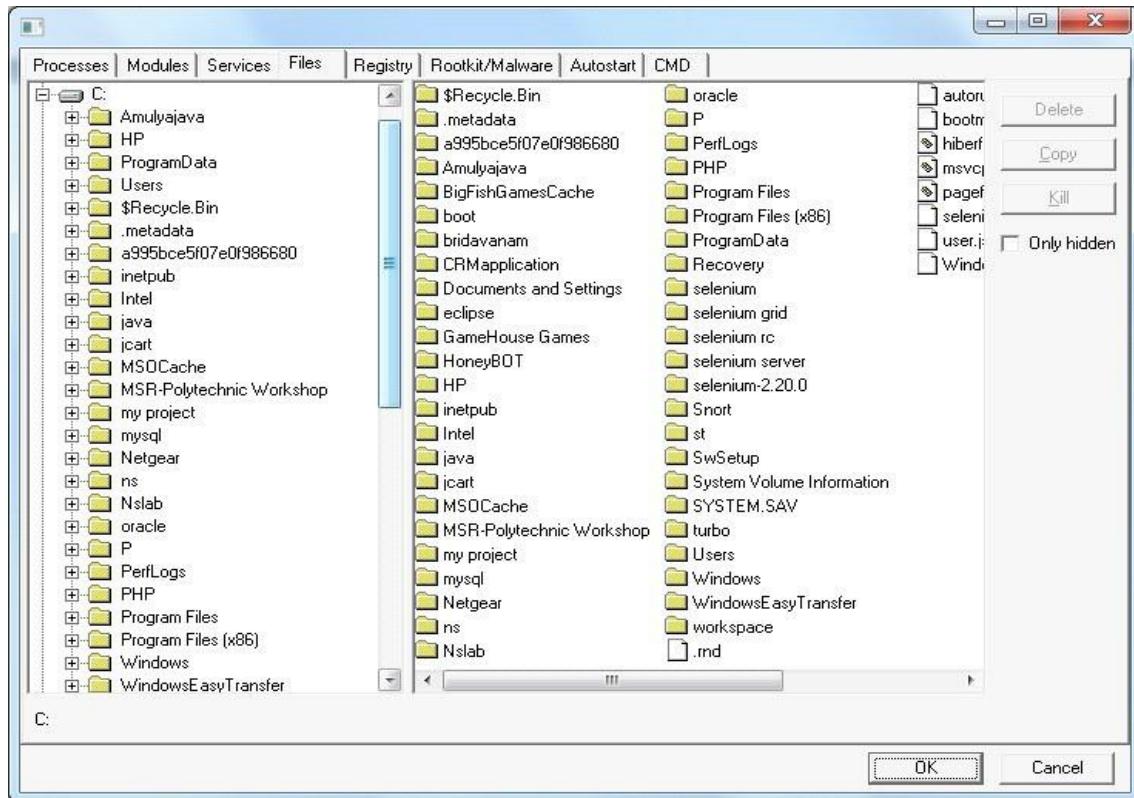
STEP-10: CMD allows the user to interact with command line utilities or Registry

SCREENSHOTS:



Windows Task Manager - Applications					
Name		File		Address	Size
ntoskrnl.exe		\SystemRoot\system32\ntoskrnl.exe		0305C000	6193152
hal.dll		\SystemRoot\system32\hal.dll		03013000	299008
kdcom.dll		\SystemRoot\system32\kdcom.dll		008CC000	40960
mcupdate_Genui..		\SystemRoot\system32\mcupdate_GenuineIntel.dll		00C00000	323584
PSHED.dll		\SystemRoot\system32\PSHED.dll		00C4F000	81920
CLFS.SYS		\SystemRoot\system32\CLFS.SYS		00CB3000	385024
Cl.dll		\SystemRoot\system32\Cl.dll		00CC1000	786432
Wdf01000.sys		\SystemRoot\system32\drivers\Wdf01000.sys		00EA3000	671744
WDFLDR.SYS		\SystemRoot\system32\drivers\WDFLDR.SYS		00F47000	61440
ACPI.sys		\SystemRoot\system32\drivers\ACPI.sys		00F56000	356352
WMILIB.SYS		\SystemRoot\system32\drivers\WMILIB.SYS		00FAD000	36864
msisadvr.sys		\SystemRoot\system32\drivers\msisadvr.sys		00FB6000	40960
pci.sys		\SystemRoot\system32\drivers\pci.sys		00FC0000	208896
vdrvroot.sys		\SystemRoot\system32\drivers\vdrvroot.sys		00FF3000	53248
partmgr.sys		\SystemRoot\system32\drivers\partmgr.sys		00E00000	86016
compatb.sys		\SystemRoot\system32\DRIVERS\compatb.sys		00E15000	36864
BATTC.SYS		\SystemRoot\system32\DRIVERS\BATTC.SYS		00E1E000	49152
volmgr.sys		\SystemRoot\system32\drivers\volmgr.sys		00E2A000	86016
volmgrx.sys		\SystemRoot\system32\drivers\volmgrx.sys		00E3F000	376832
mountmgr.sys		\SystemRoot\system32\drivers\mountmgr.sys		00D81000	106496
iaStor.sys		\SystemRoot\system32\DRIVERS\iaStor.sys		0103A000	2138112
atapi.sys		\SystemRoot\system32\drivers\atapi.sys		01244000	36864
ataport.SYS		\SystemRoot\system32\drivers\ataport.SYS		0124D000	172032
msahci.sys		\SystemRoot\system32\drivers\msahci.sys		01277000	45056
PCIIDEX.SYS		\SystemRoot\system32\drivers\PCIIDEX.SYS		01282000	65536
amdxata.sys		\SystemRoot\system32\drivers\amdxata.sys		01292000	45056
fltmgr.sys		\SystemRoot\system32\drivers\fltmgr.sys		0129D000	311296
fileinfo.sys		\SystemRoot\system32\drivers\fileinfo.sys		012E9000	81920
Ntfs.sys		\SystemRoot\system32\Drivers\Ntfs.sys		0142D000	1716224
msrpc.sys		\SystemRoot\system32\Drivers\msrpc.sys		012FD000	385024
kseaddr.sys		\SystemRoot\system32\Drivers\kseaddr.sys		015D0000	110592
... (truncated)		\SystemRoot\system32\Drivers\... (truncated)		015E9000	466944

Windows Task Manager - Applications					
Name		Start	File name	Description	
.NET CLR Data					
.NET CLR Netwo...					
.NET CLR Netwo...					
.NET Data Provid...					
.NET Data Provid...					
.NETFramework					
1394ohci	MANUAL	SYSTEM	\SystemRoot\system32\drivers\1394ohci.sys	1394 OHCI Compliant Host Controller	
ACPI	BOOT	SYSTEM	\SystemRoot\system32\drivers\ACPI.sys	Microsoft ACPI Driver	
AcpiPmi	MANUAL	SYSTEM	\SystemRoot\system32\drivers\acpipmi.sys	ACPI Power Meter Driver	
adp34xx	MANUAL	SYSTEM	\SystemRoot\system32\DRIVERS\adp34xx.sys		
adpahci	MANUAL	SYSTEM	\SystemRoot\system32\DRIVERS\adpahci.sys		
adpu320	MANUAL	SYSTEM	\SystemRoot\system32\DRIVERS\adpu320.sys		
adsi					
AeLookupSvc	MANUAL	%systemroot%\system32\svchost.exe -k netsvcs		@%SystemRoot%\system32\aelupserv.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\agrm64.sys		Agere Systems Soft Modem	
agg440	MANUAL	system32\DRIVERS\agg440.sys		Intel AGP Bus Filter	
ALG	MANUAL	%SystemRoot%\System32\alg.exe		@%SystemRoot%\System32\Alg.exe,-113	
alide	MANUAL	\SystemRoot\system32\drivers\alide.sys			
amidde	MANUAL	\SystemRoot\system32\drivers\amidde.sys			
AmdK8	MANUAL	\SystemRoot\system32\DRIVERS\amdk8.sys		AMD K8 Processor Driver	
AmdPPM	MANUAL	\SystemRoot\system32\DRIVERS\amddppm.sys		AMD Processor Driver	
amsata	MANUAL	\SystemRoot\system32\drivers\amsata.sys			
amdsbs	MANUAL	\SystemRoot\system32\DRIVERS\amdsbs.sys			
amdxata	BOOT	system32\drivers\amdxata.sys			
AppHostSvc	AUTO	%windir%\system32\svchost.exe -k apphost		@%windir%\system32\inetsrv\iisres.dll,-30012	
ApplD	MANUAL	\SystemRoot\system32\drivers\appld.sys		@%systemroot%\system32\appidsvc.dll,-103	
AppIDSvc	MANUAL	%SystemRoot%\system32\svchost.exe -k Local...		@%systemroot%\system32\appidsvc.dll,-101	
Appinfo	MANUAL	%SystemRoot%\system32\svchost.exe -k netsvcs		@%systemroot%\system32\appinfo.dll,-101	
AppMgmt	MANUAL	%SystemRoot%\system32\svchost.exe -k netsvcs		@appmgmts.dll,-3251	
... (truncated)	MANUAL	\SystemRoot\system32\DRIVERS\... (truncated)			



RESULT:

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

EX. NO: 08

To perform wireless audit on an access point or a router and decrypt WEP and WPA (Net Stumbler).

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.

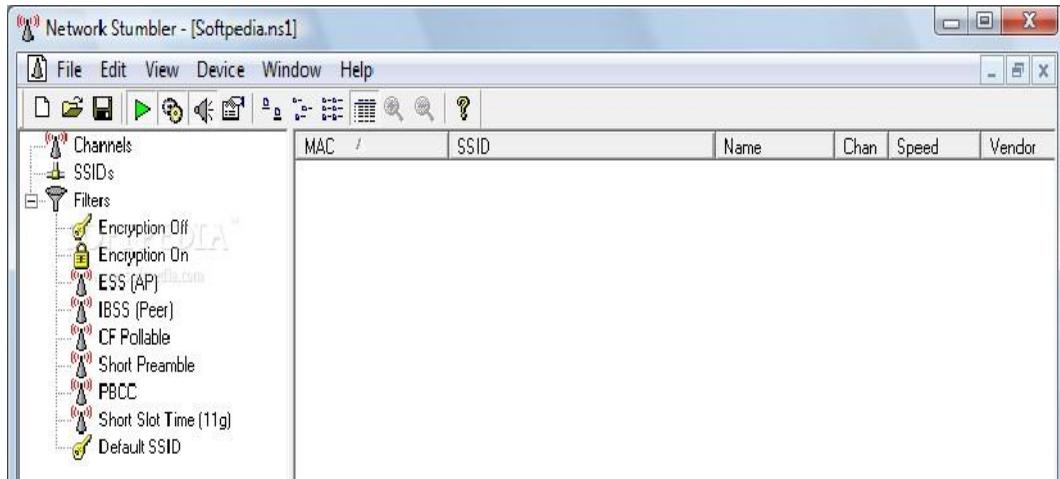
STEP-6: MAC assigned to Wireless Access Point is displayed on right hand pane.

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

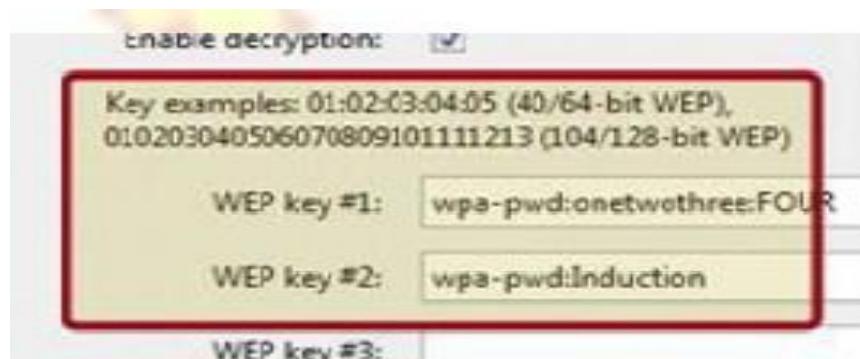
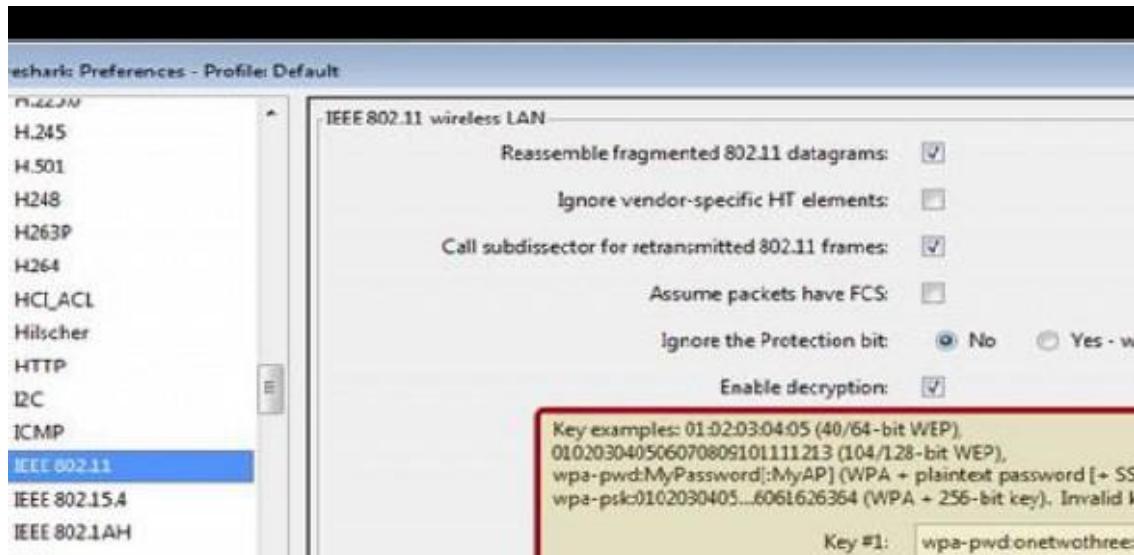
STEP-8: To decrypt use Wireshark tool by selecting Edit □ preferences □ IEEE 802.11.

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

SCREENSHOTS:

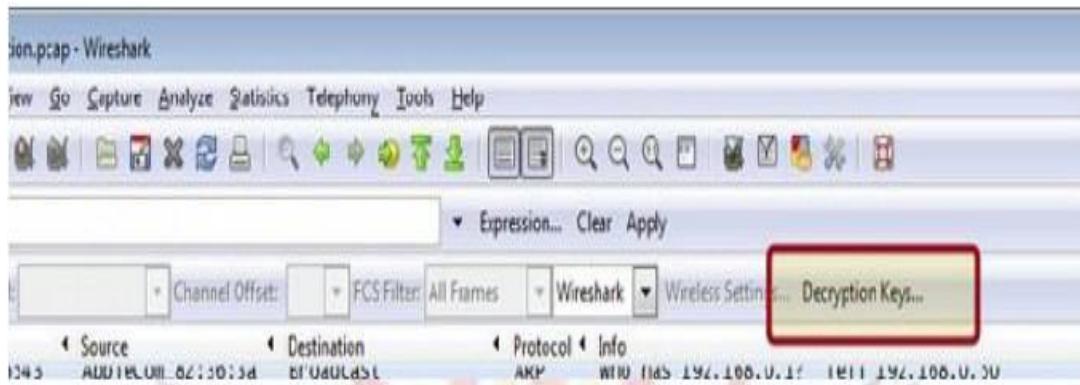


MAC	SSID	Name	Chan	Speed	Vendor	Type	Enc.	SNR	Signal+	Noise	SNR+
000F3D38195E	default	2	54 Mbps			AP	18	-82	-100	18	
000F669AAE99	linksys	6	11 Mbps	Linksys	Linksys	AP	19	-80	-100	20	
000F66E1DC43	buss	6*	54 Mbps	Linksys	Linksys	AP	WEP	34	-37	-100	63



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: 09

**Demonstrate intrusion detection system (ids) using any tool
(snort or any other s/w).**

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

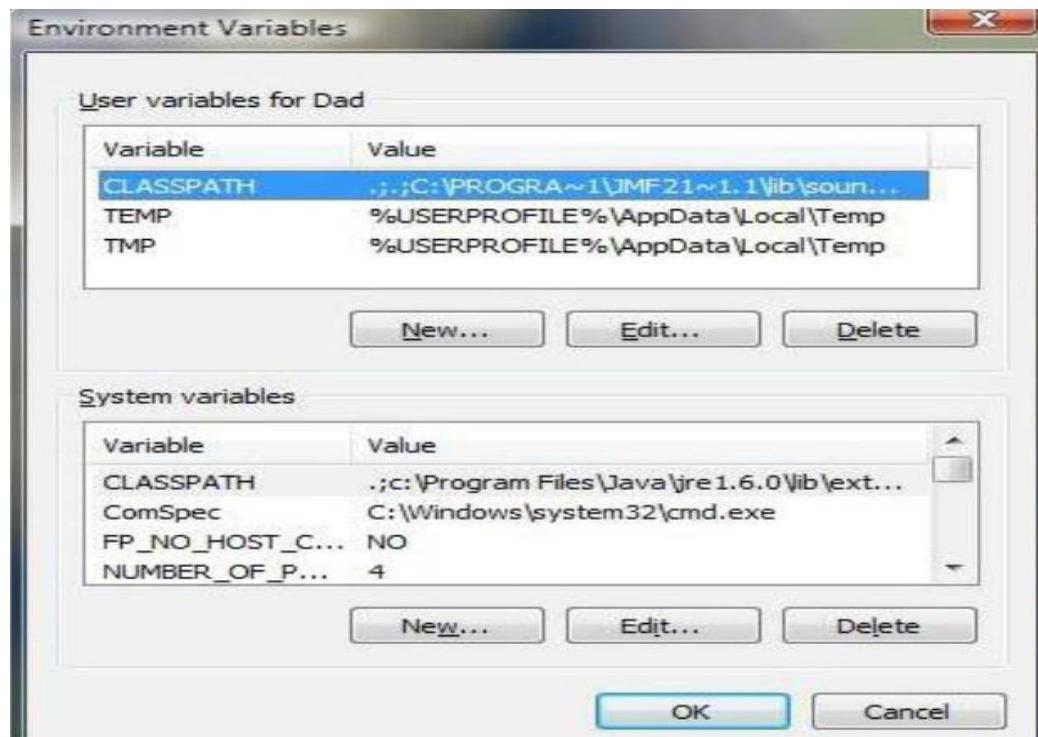
PROCEDURE:

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

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

- STEP-3:** Packet Logger mode snort -dev -l c:\log [create this directory in the C drive]
and snort will automatically know to go into packet logger mode, it collects every packet it sees and places it in log directory.
- STEP-4:** snort -dev -l c:\log -h ipaddress/24 This rule tells snort that you want to print out the data link and TCP/IP headers as well as application data into the log directory.
- STEP-5:** snort -l c:\log -b this binary mode logs everything into a single file.
- STEP-6:** Network Intrusion Detection System mode snort -d c:\log -h ipaddress/24 -c snort.conf This is a configuration file that applies rule to each packet to decide it an action based upon the rule type in the file.
- STEP-7:** snort -d -h ip address/24 -l c:\log -c snort.conf This will configure snort to run in its most basic NIDS form, logging packets that trigger rules specifies in the snort.conf.
- STEP-8:** Download SNORT from snort.org. Install snort with or without database support.
- STEP-9:** Select all the components and Click Next. Install and Close.
- STEP-10:** Skip the WinPcap driver installation.
- STEP-11:** Add the path variable in windows environment variable by selecting new classpath.
- STEP-12:** Create a path variable and point it at snort.exe variable name path and variable value c:\snort\bin.
- STEP-13:** Click OK button and then close all dialog boxes. Open command prompt and type the following commands:

INSTALLATION PROCESS :



```
Administrator: C:\Windows\system32\cmd.exe - snort -v
UDP TTL:128 TOS:0x0 ID:903 IpLen:20 DgmLen:78
Len: 50
=====
03/20-12:13:57.248341 192.168.56.101:63650 -> 224.0.0.252:5355
UDP TTL:1 TOS:0x0 ID:904 IpLen:20 DgmLen:50
Len: 22
=====
03/20-12:13:57.348568 192.168.56.101:63650 -> 224.0.0.252:5355
UDP TTL:1 TOS:0x0 ID:905 IpLen:20 DgmLen:50
Len: 22
=====
03/20-12:13:57.548888 192.168.56.101:137 -> 192.168.56.255:137
UDP TTL:128 TOS:0x0 ID:906 IpLen:20 DgmLen:78
Len: 50
=====
03/20-12:13:58.298907 192.168.56.101:137 -> 192.168.56.255:137
UDP TTL:128 TOS:0x0 ID:907 IpLen:20 DgmLen:78
```

```

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
  Plts/min:      128
  Plts/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:              9 < 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%>
SS_G_1:             0 < 0.000%>
SS_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.

Automated Attack and Penetration Tools Exploring N-Stalker, a Vulnerability Assessment Tool

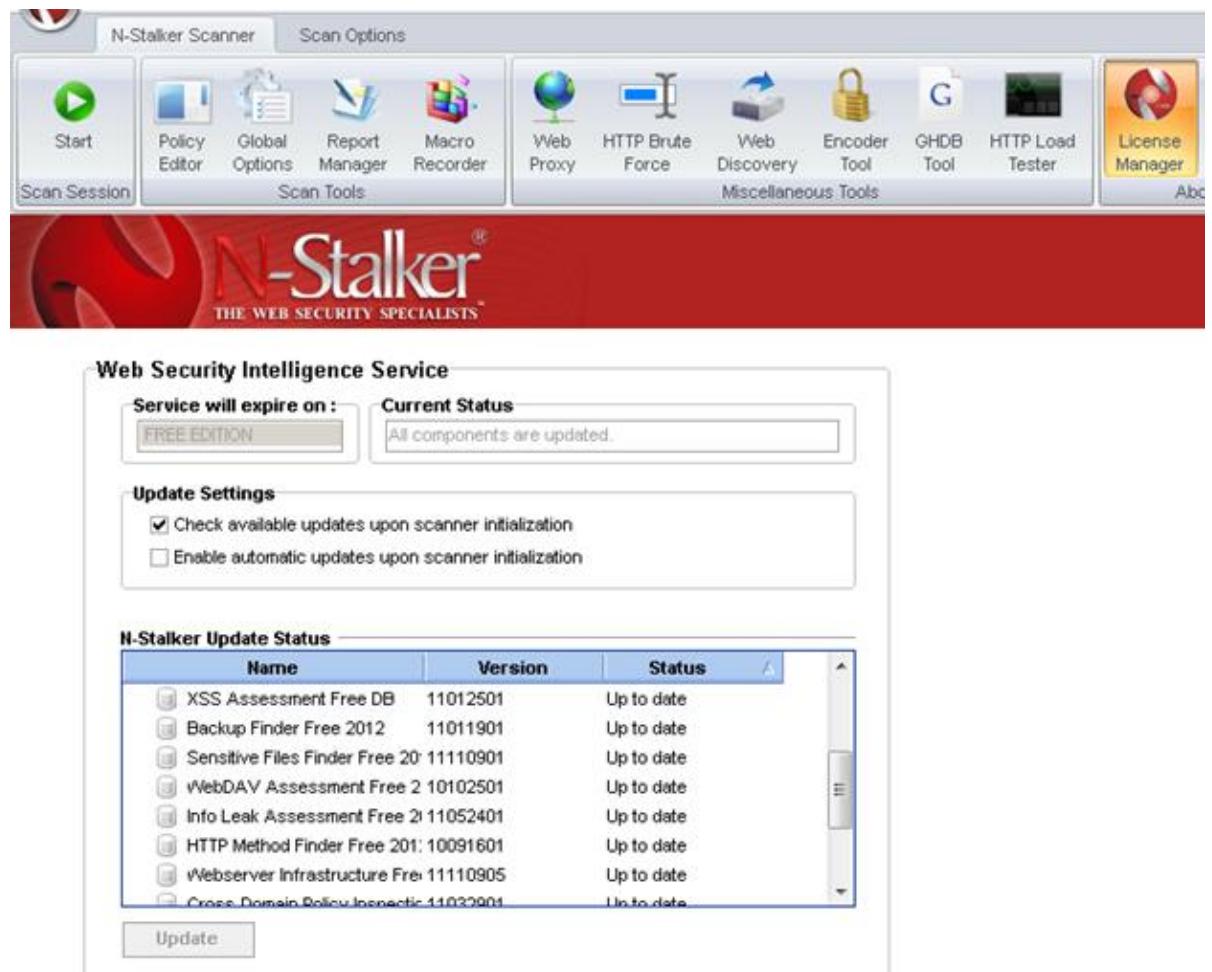
AIM:

To download the N-Stalker Vulnerability Assessment Tool and exploring the features.

EXPLORING N-STALKER:

- N-Stalker Web Application Security Scanner is a Web security assessment tool.
 - It incorporates with a well-known N-Stealth HTTP Security Scanner and 35,000 Web attack signature database.
 - This tool also comes in both free and paid version.
 - Before scanning the target, go to “License Manager” tab, perform the update.
 - Once update, you will note the status as up to date.
 - You need to download and install N-Stalker from www.nstalker.com.
 -
1. Start N-Stalker from a Windows computer. The program is installed under Start ⇔ Programs ⇔ N-Stalker ⇔ N-Stalker Free Edition.
 2. Enter a host address or a range of addresses to scan.
 3. Click Start Scan.
 4. After the scan completes, the N-Stalker Report Manager will prompt
 5. you to select a format for the resulting report as choose Generate HTML.

Review the HTML report for vulnerabilities



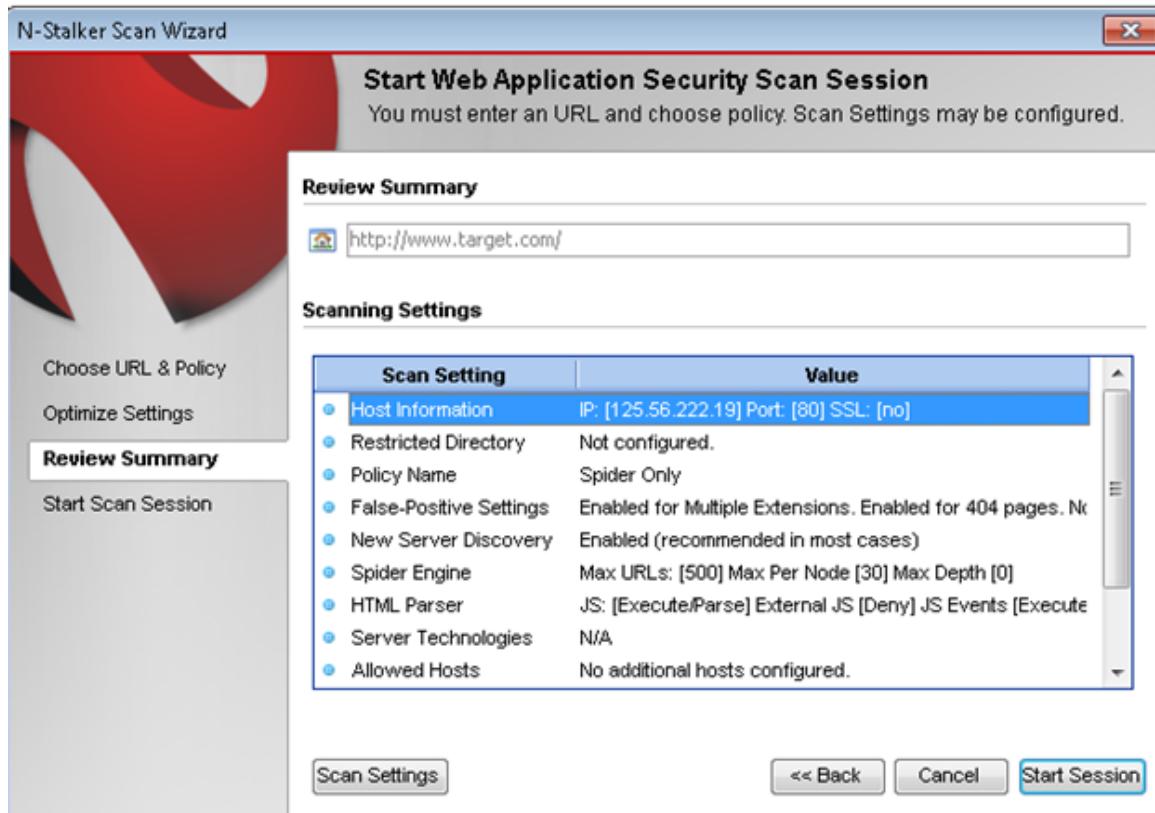
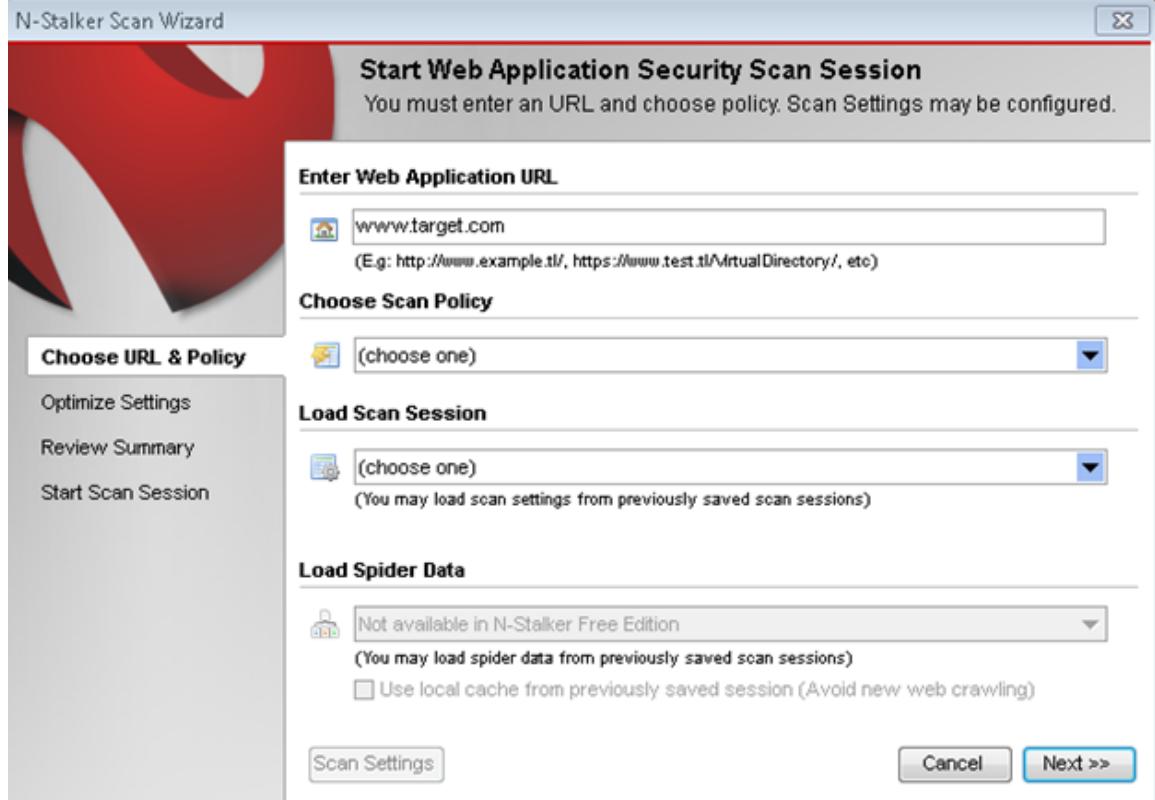
Now goto “Scan Session”, enter the target URL.

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

- Manual test which will crawl the website and will be waiting for manual attacks.
- full xss assessment
- owasp policy
- Web server infrastructure analysis.

Once, the option has been selected, next step is “Optimize settings” which will crawl the whole website for further analysis.

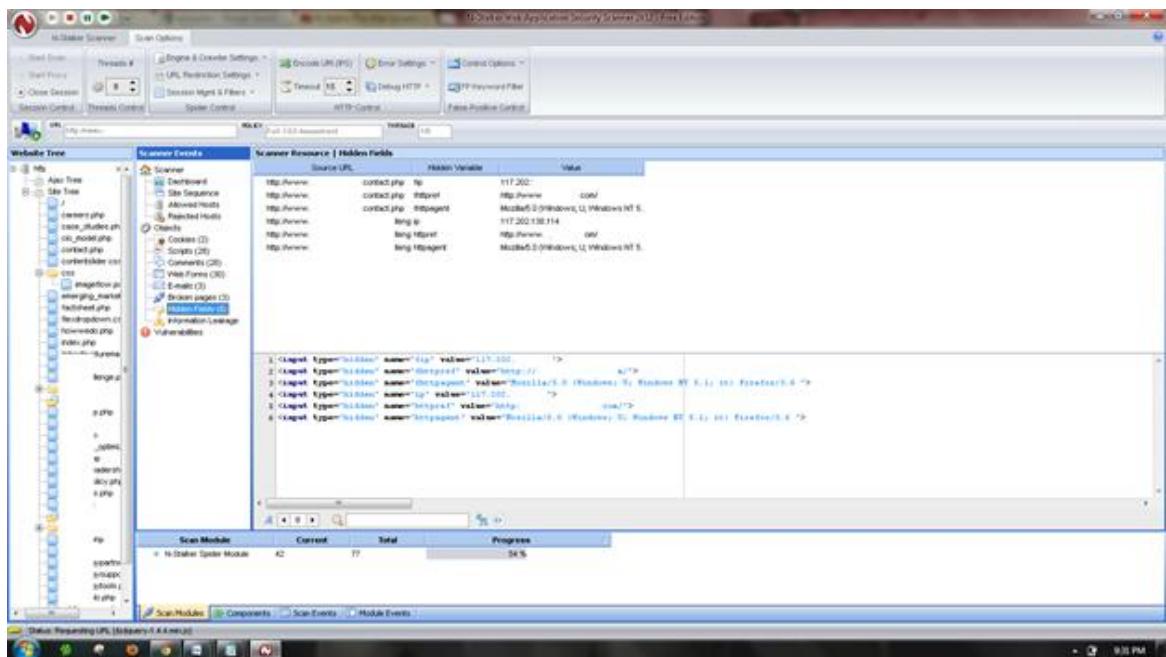
In review option, you can get all the information like host information, technologies used, policy name, etc.



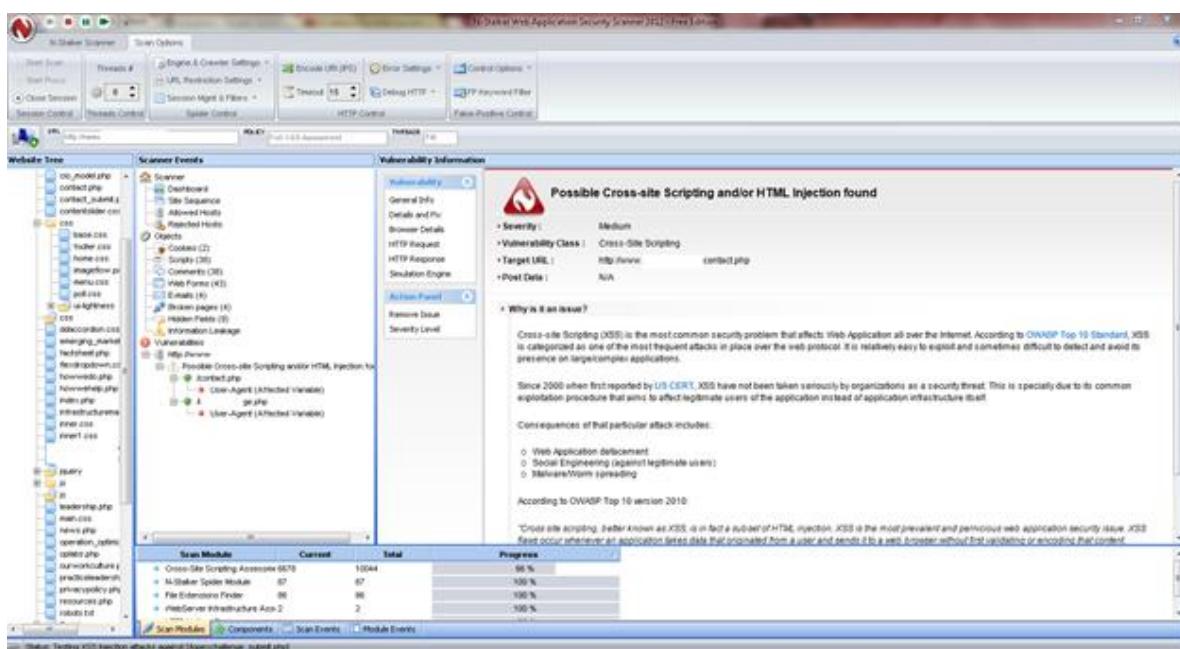
Once done, start the session and start the scan.

The scanner will crawl the whole website and will show the scripts, broken pages, hidden fields, information

leakage, web forms related information which helps to analyze further.



Once the scan is completed, the NStalker scanner will show details like severity level, vulnerability class, why is it an issue, the fix for the issue and the URL which is vulnerable to the particular vulnerability?



RESULT:

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

Experiment 11

Defeating Malware - Building Trojans

AIM:

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

PROCEDURE:

1. Create a simple trojan by using Windows Batch File (*.bat*)
2. Type these below code in notepad and save it as **Trojan.bat**
3. Double click on **Trojan.bat** file.
4. When the trojan code executes, it will open MS-Paint, Notepad, Command Prompt, Explorer, etc., infinitely.
5. Restart the computer to stop the execution of this trojan.

TROJAN:

- In computing, a Trojan horse, or trojan, is any malware which misleads users of its true intent.
- Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or anywhere else.
- Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer.
- Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity.
- **Example:** Ransomware attacks are often carried out using a *trojan*.

CODE:

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

OUTPUT

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

RESULT:

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

Defeating Malware - Rootkit hunter

AIM:

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

ROOTKIT HUNTER:

- rkHunter (Rootkit Hunter) is a Unix-based tool that scans for rootkits, backdoors and possible local exploits.

- It does this by comparing SHA-1 hashes of important files with known good ones in online databases, searching for default directories (of rootkits), wrong permissions, hidden files, suspicious strings in kernel modules, and special tests for Linux and FreeBSD.
- rkHunter is notable due to its inclusion in popular operating systems (Fedora, Debian, etc.)
- The tool has been written in Bourne shell, to allow for portability. It can run on almost all UNIX-derived systems.

GMER ROOTKIT TOOL:

- GMER is a software tool written by a Polish researcher Przemysław Gmerek, for detecting and removing rootkits.
- It runs on Microsoft Windows and has support for Windows NT, 2000, XP, Vista, 7, 8 and 10. With version 2.0.18327 full support for Windows x64 is added.

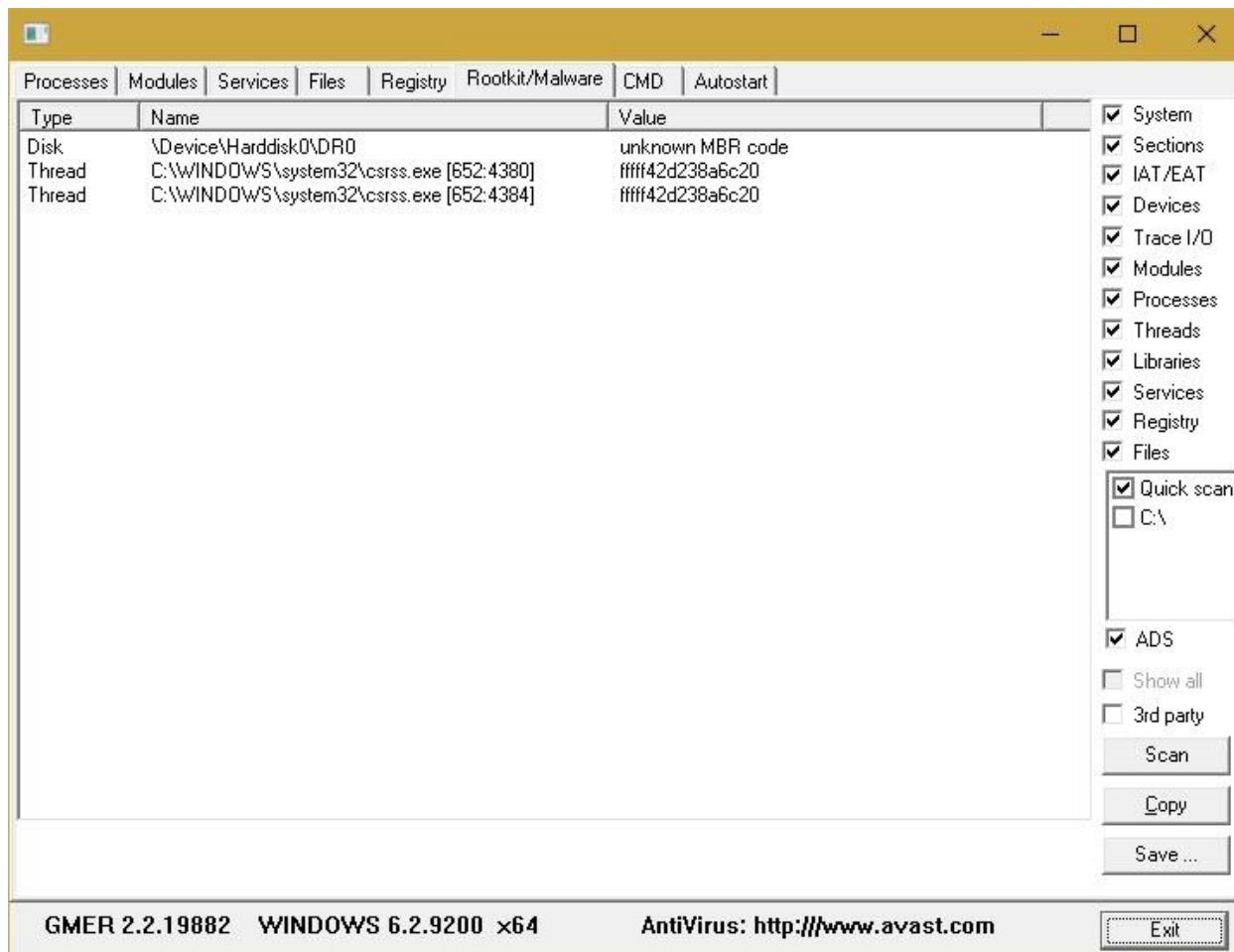
Step 1

The screenshot shows the GMER 2.0.18323 interface running on Windows 6.1.7600 x64. The main window title is "GMER 2.0.18323 - WINDOWS 6.1.7600 x64". The left sidebar includes links for Start, Files, News, Rootkits, FAQ, and Contact. The "Rootkits" section is active, stating "GMER is an application that detects and removes rootkits . It scans for:" followed by a list of items such as hidden processes, threads, modules, services, files, disk sectors, alternate data streams, registry keys, drivers, and hooks. The main pane displays a table titled "RootKit/Malware" with columns "Type", "Name", and "Value". The table lists numerous registry keys under the "Value" column, many of which begin with "ffff80000b9b840\". A small warning dialog box is overlaid on the bottom right of the main window, reading "WARNING !!! GMER has found system modification caused by ROOTKIT activity." with an exclamation mark icon.

Visit GMER's website (see Resources) and download the GMER executable.

Click the "Download EXE" button to download the program with a random file name, as some rootkits will close "gmer.exe" before you can open it.

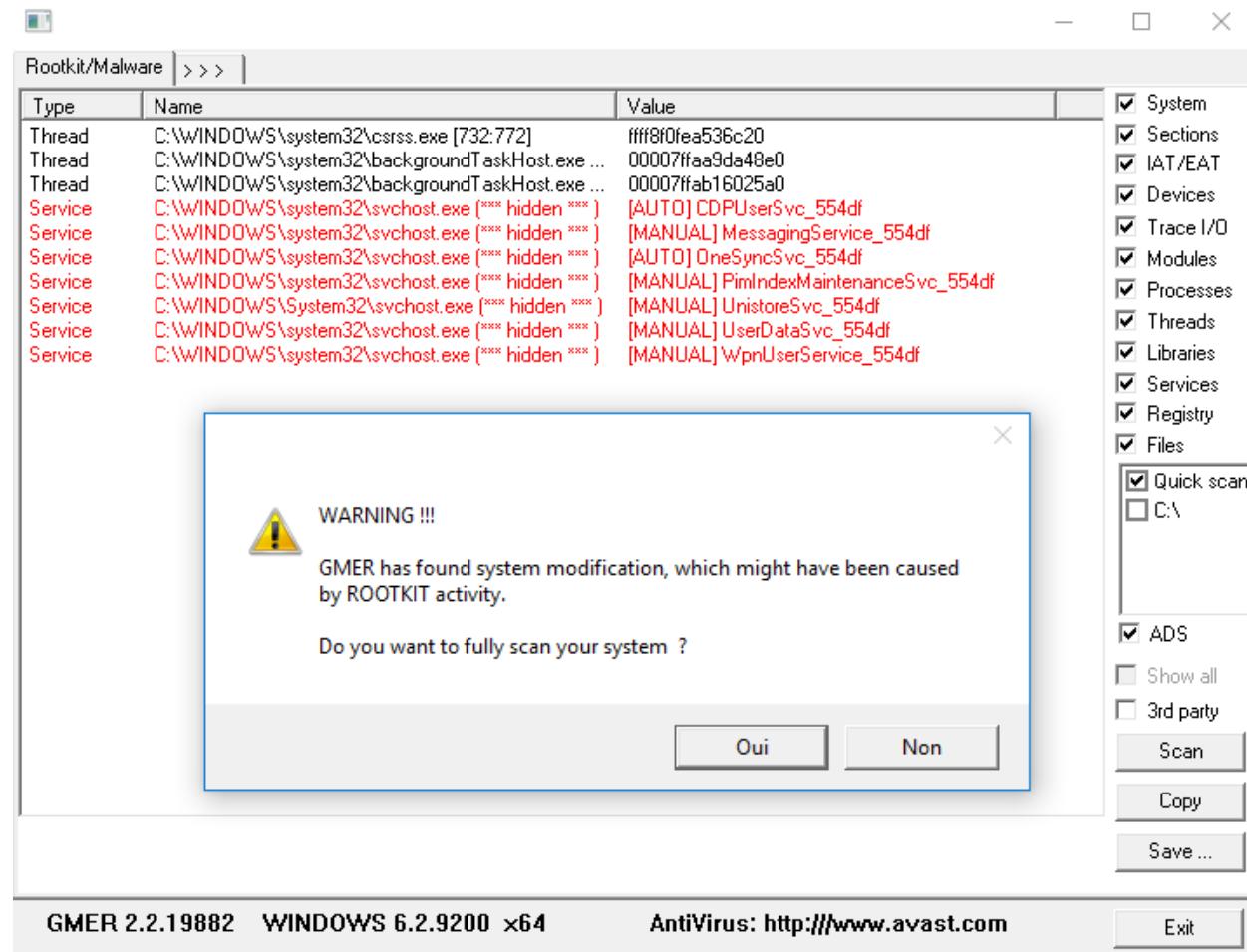
Step 2



Double-click the icon for the program.

Click the "Scan" button in the lower-right corner of the dialog box. Allow the program to scan your entire hard drive.

Step 3



When the program completes its scan, select any program or file listed in red. Right-click it and select "Delete."

If the red item is a service, it may be protected. Right-click the service and select "Disable." Reboot your computer and run the scan again, this time selecting "Delete" when that service is detected.

When your computer is free of Rootkits, close the program and restart your PC.

RESULT:

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