



NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE

(NAAC Accredited)



(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)

Pampady, Thiruvilwamala (PO), Thrissur (DT), Kerala 680 588

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



LAB MANUAL



08CS 6272 CYBER SECURITY LAB

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- ◆ Established in: 2002
- ◆ Course offered: B. Tech in Computer Science and Engineering
M.Tech in Computer Science and Engineering
M.Tech in Cyber Security
- ◆ Approved by AICTE New Delhi and Accredited by NAAC
- ◆ Certified by ISO 9001:2015
- ◆ Affiliated to the A P J Abdul Kalam Technological University.

DEPARTMENT VISSION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.

PEO2: Graduates will be able to Analyse, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.

PEO3: Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.

PEO4: Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Team work and leadership qualities.

PROGRAM OUTCOMES (POS)

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 modeling 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.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOME

CO 1	To implement various encryption techniques
CO 2	Develop cryptosystems using various public key and encryption methods.
CO 3	Construct codes for various authentication schemes
CO 4	To implement various hashing techniques
CO 5	Develop signature scheme using digital signature method
CO 6	Demonstrate various network security system using open source tools

CO VS PO'S AND PSO'S MAPPING

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO 1	3		3									
CO 2	3	3	3									
CO 3		3										
CO 4												
CO 5		3	2	2								
CO 6				2		3						

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

	PSO1	PSO2	PSO3
CO1	2	3	
CO2	2		
CO3		2	
CO4		2	
CO5		2	
CO6	2		2

PREPARATION FOR THE LABORATORY SESSION
GENERAL INSTRUCTIONS TO STUDENTS

1. Read carefully and understand the description of the experiment in the lab manual. You may go to the lab at an earlier date to look at the experimental facility and understand it better. Consult the appropriate references to be completely familiar with the concepts and hardware.
2. Make sure that your observation for previous week experiment is evaluated by the faculty member and you have transferred all the contents to your record before entering to the lab/workshop.
3. At the beginning of the class, if the faculty or the instructor finds that a student is not adequately prepared, they will be marked as absent and not be allowed to perform the experiment.
4. Bring necessary material needed (writing materials, record etc)
5. Please actively participate in class and don't hesitate to ask questions. Please utilize the teaching assistants fully. To encourage you to be prepared and to read the lab manual before coming to the laboratory, unannounced questions may be asked at any time during the lab.

6. Carelessness in personal conduct or in handling equipment may result in serious injury to the individual or the equipment.
7. Students must follow the proper dress code inside the laboratory. Long hair should be tied back.
8. Maintain silence, order and discipline inside the lab. Don't use cell phones inside the laboratory.
9. Any injury no matter how small must be reported to the instructor immediately.
10. Check with faculty members one week before the experiment to make sure that you have the handout for that experiment and all the apparatus.

AFTER THE LABORATORY SESSION

1. Clean up your work area.
2. Check with the technician before you leave.
3. Make sure you understand what kind of report is to be prepared and due submission of record is next lab class.

MAKE-UPS AND LATE WORK

Students must participate in all laboratory exercises as scheduled. They must obtain permission from the faculty member for absence, which would be granted only under justifiable circumstances. In such an event, a student must make arrangements for a make-up laboratory, which will be scheduled when the time is available after completing one cycle. Late submission will be awarded less mark for record and internals and zero in worst cases.

LABORATORY POLICIES

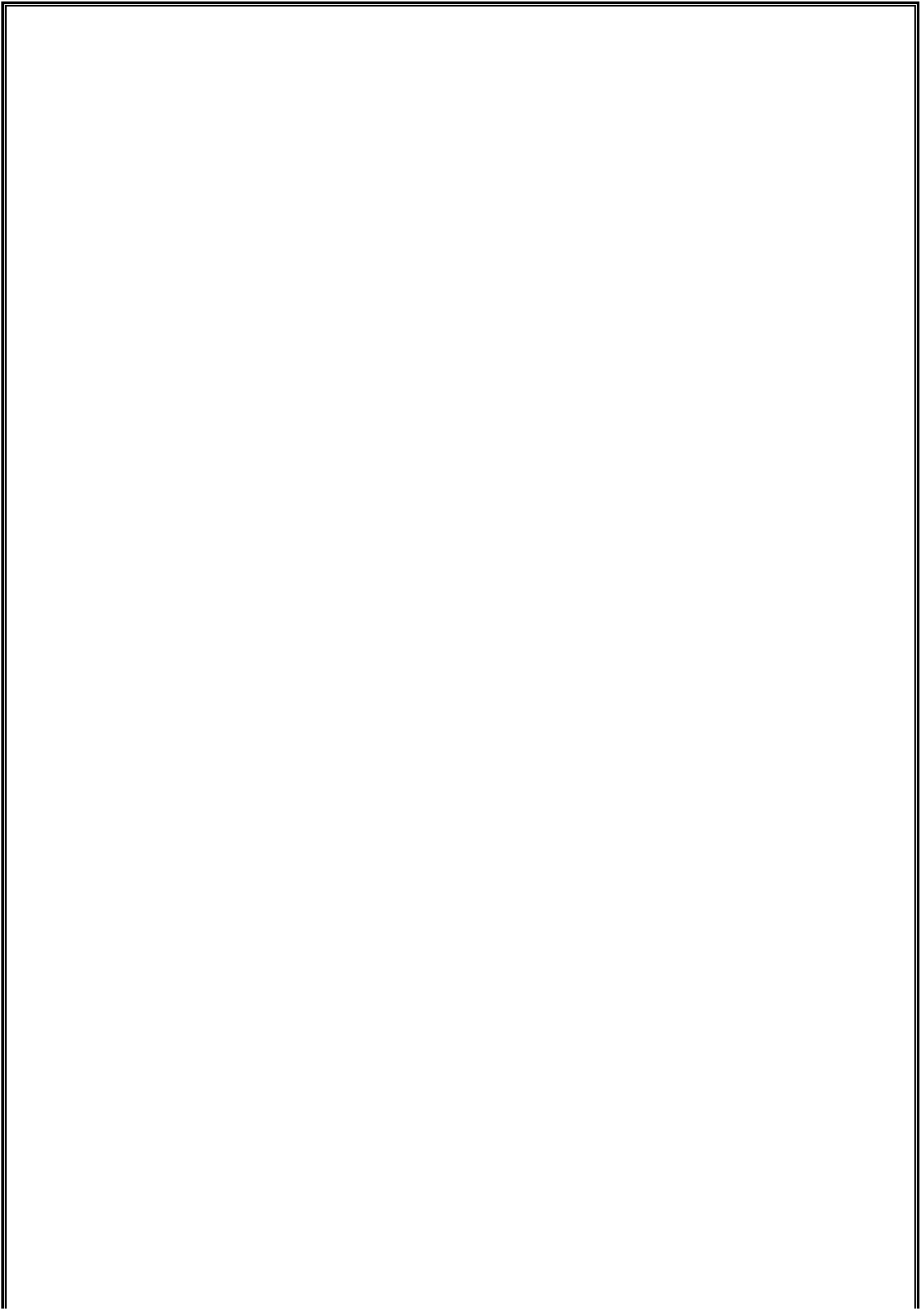
1. Food, beverages & mobile phones are not allowed in the laboratory at any time.
2. Do not sit or place anything on instrument benches.
3. Organizing laboratory experiments requires the help of laboratory technicians and staff. Be punctual.

SYLLABUS

COURSE NO: 08CS6272 COURSE TITLE: CYBER SECURITY LABORATORY

CREDITS: 1

1. Implementation of Substitution and Transposition ciphers
2. Implementation of Data Encryption Standard
3. Implementation of International Data Encryption Algorithm
4. Implementation of Advanced Encryption Standard
5. Implementation of RSA Algorithm
6. Implementation of Diffie-Hellman Key Exchange
7. Implementation of Message Authentication Codes
8. Implementation of Hash functions
9. Implementation of Digital Signature Standard
10. Hiding of confidential information within Image
11. Implementation in FOSS based security mechanisms'



INDEX

S.No.	Name of the Program	Date	Staff Signature	Remarks
1(A)	Caesar Cipher			
1(B)	Playfair Cipher			
1(C)	Hill Cipher			
1(D)	Vigenere Cipher			
1(E)	Rail fence – row & Column Transformation			
2	Data Encryption Standard(DES)			
3	International Data Encryption Algorithm (IDEA)			
4	AES Algorithm			
5	RSA Algorithm			
6	Diffie-Hellman Algorithm			
7	MD5			
8	SHA-1			
9	Implement the Signature Scheme for Digital Signature Standard			
10	Implementation of FOSS based security mechanisms			

EX. NO: 1(A)

IMPLEMENTATION OF CAESAR CIPHER

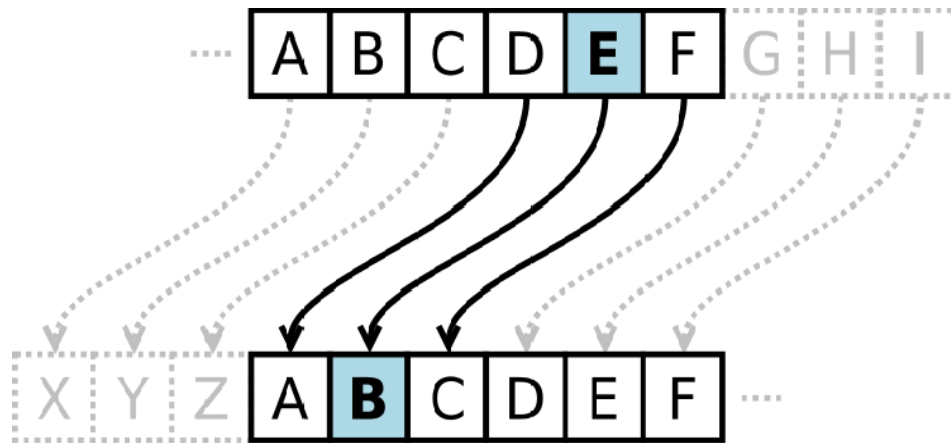
AIM:

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)

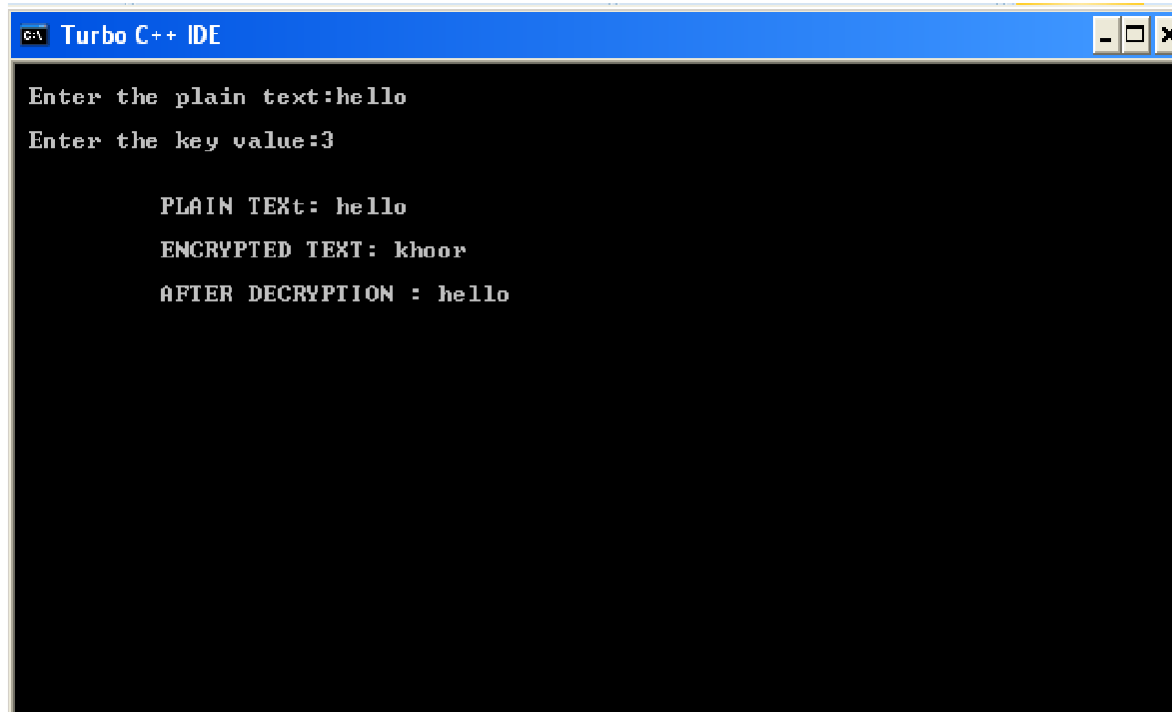
```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
void main ()
```

```

{
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:



```
g++ Turbo C++ IDE
Enter the plain text:hello
Enter the key value:3

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

VIVA QUESTIONS:

1. Crack the following plaintext TRVJRI TZGYVIJ RIV HLZKV VRJP KFTIRTB
 2. What encryption key was used?
 3. Make your own cipher text using the Caesar cipher.
 4. Can you crack other people's ciphertexts?
 5. What key do we need to make "CAESAR" become "MKOCKB"?
 6. What key do we need to make "CIPHER" become "SYFXUH"?
 7. Use the Caesar cipher to encrypt your first name
- How can we find the decryption key from the encryption key?

RESULT:

Thus the implementation of Caesar cipher had been executed successfully.

EX. NO: 1(B)

IMPLEMENTATION OF PLAYFAIR CIPHER

AIM:

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

DESCRIPTION:

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

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

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

EXAMPLE:

D. Playfair Cipher

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

Grouped text: CR YP TO IS TO XO EA SY

Ciphertext: AQ TV YB NI YB YF CB OZ

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

ALGORITHM:

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

STEP-2: Read the keyword from the user.

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

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

STEP-5: Display the obtained cipher text.

PROGRAM: (Playfair Cipher)

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#define MX 5
void playfair(char ch1,char ch2, char key[MX][MX])
{
    int i,j,w,x,y,z;
    FILE *out;
    if((out=fopen("cipher.txt","a+"))==NULL)
    {
        printf("File 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],keyst[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(keyst);
    printf("\nEnter the plain text:");
    gets(str);
    n=strlen(keyst);
    //convert the characters to uppertext
    for (i=0; i<n; i++)
    {
        if(keyst[i]=='j')keyst[i]='i';
        else if(keyst[i]=='J')keyst[i]='I';
        keyst[i] = toupper(keyst[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(keyst[k]==alpa[i])
            break;
            else if(alpa[i]=='J')
            break;
        }
        if(k==n)
        {
            keyminus[j]=alpa[i];j++;
        }
    }
}

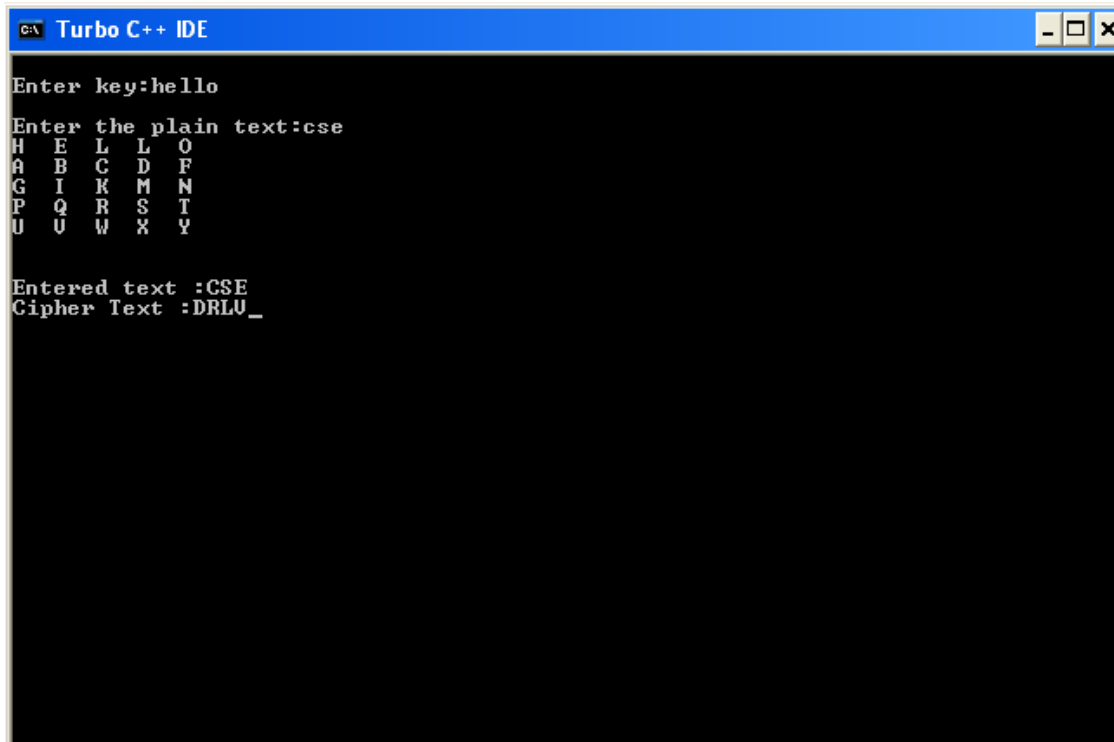
```

```

//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:



```
C:\ Turbo C++ IDE
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 U W X Y

Entered text :CSE
Cipher Text :DRLU_
```

VIVA QUESTIONS :

1. What is difference between a monoalphabetic and a polyalphabetic cipher?
2. What are stream cipher and block cipher and how are they different?
3. How many possible keys does the playfair cipher have?
4. How to find the keyword of playfair cipher, given the plain text and cipher text?
5. Construct a table for the Playfair Cipher with the keyword EFFECTIVENESS?

RESULT:

Thus the Playfair cipher substitution technique had been implemented successfully.

EX. NO: 1(C)

IMPLEMENTATION OF HILL CIPHER

AIM:

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

DESCRIPTION:

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

EXAMPLE:

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

ALGORITHM:

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

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

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

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

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

PROGRAM: (Hill Cipher)


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

```

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

```

OUTPUT:



The screenshot shows a Turbo C++ IDE window with the following output:

```

Turbo C++ IDE
Enter plain text
ACT
0 2 19
Encrypted Cipher Text : P O H
Decrypted Cipher Text : A C T

```

VIVA QUESTIONS

1. Name the aspects to be considered of information security.
2. What is meant by deciphering?
3. What are the two different uses of public key cryptography related to key distribution?
4. How many bit keys are used in S-DES algorithm?
5. What are the parameters are include the certificate request message?
6. What is S/MIME?
7. What is the purpose of dual signature?
8. What are the two common techniques used to protect a password file?
9. What is the size of the key for substitution block cipher?
10. Define Stream Cipher.

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

ALGORITHM:

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

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

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

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

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

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

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

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

PROGRAM: (Vigenere Cipher)

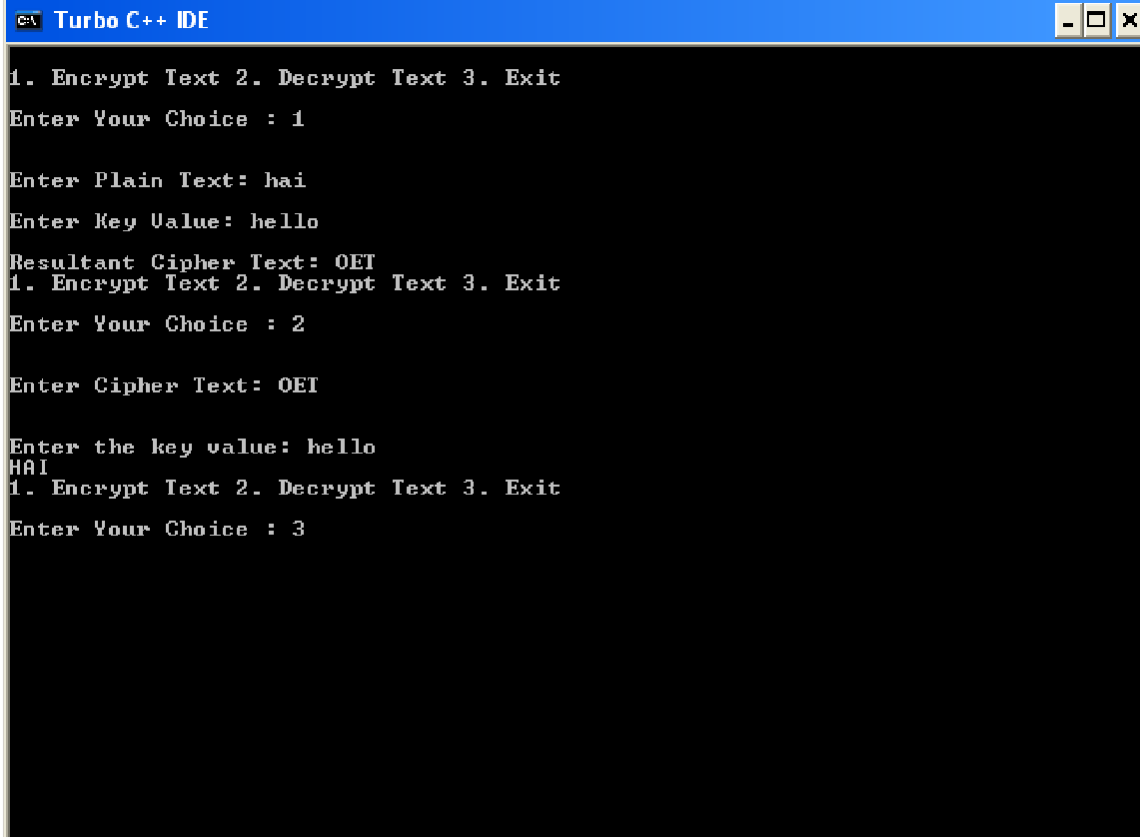
```
#include <stdio.h>
#include<conio.h>
#include <ctype.h>
#include <string.h>
void encipher ();
void decipher ();
void main ()
{
    int choice;
    clrscr();
    while(1)
    {
        printf("\n1. Encrypt Text");
        printf("\t2. Decrypt Text");
        printf("\t3. Exit");
        printf("\n\nEnter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

```

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

```

OUTPUT:



```
cs Turbo C++ IDE
1. Encrypt Text 2. Decrypt Text 3. Exit
Enter Your Choice : 1
Enter Plain Text: hai
Enter Key Ualue: 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
```

VIVA QUESTIONS (PRELAB and POSTLAB):

1. What is convert channel?
2. Give the principle advantages of elliptical curve cryptography.
5. How secure is DES?
6. What is the necessity of firewalls?
7. Between symmetric Vs Public Key cryptography, which method is more convenient?
10. What is Kerberos?

RESULT:

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.

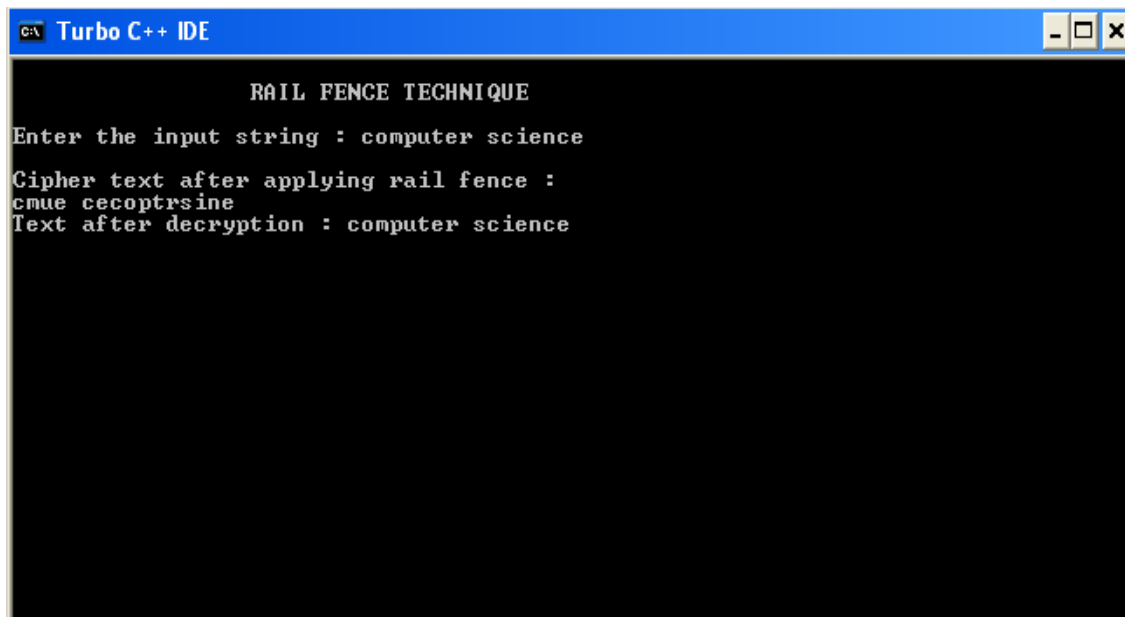
PROGRAM: (Rail Fence)

```
#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:



```
C:\ Turbo C++ IDE
                                RAIL FENCE TECHNIQUE
Enter the input string : computer science
Cipher text after applying rail fence :
cmue cecoptsrine
Text after decryption : computer science
```

VIVA QUESTIONS

1. Where do you apply PGP?
2. List out the basic tasks in Public Key Encryption in key distribution.
3. Give an example for Simple Hash Function.
4. List out the two methods of operations in Authentication Header (AH) and Encapsulating Security Payload (ESP).
5. Enumerate the functions provided by S/MIME.
6. List out the two ways in which password can be protected.
7. Which attack is related to integrity?
8. Which public key cryptosystem can be used for digital signature?
9. Expand: S/MIME.
10. What is the use of trusted system?

RESULT:

The rail fence algorithm had been executed successfully.

EX. NO: 2**IMPLEMENTATION OF DES****AIM:**

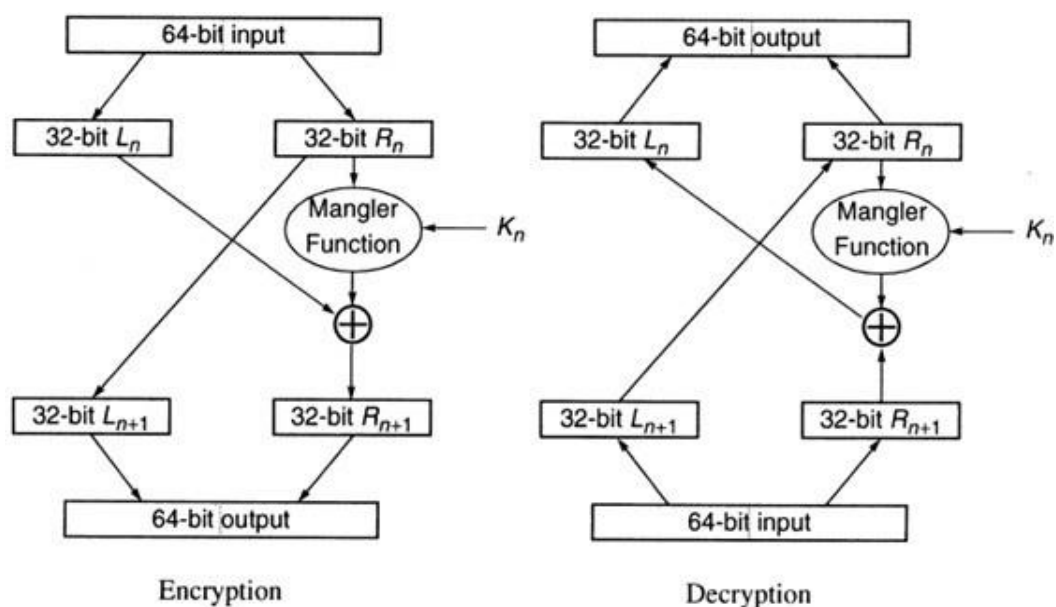
To write a program to implement Data Encryption Standard (DES)

DESCRIPTION:

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

The main parts of the algorithm are as follows:

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

EXAMPLE:

ALGORITHM:

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

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

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

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

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

PROGRAM:

DES.java

```
import 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[] iblete = inputMessage.getBytes ();
    byte[] ebyte=encrypt (raw, iblete);
    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();
}
}

```


EX. NO: 3

IMPLEMENTATION OF IDEA

AIM:

To write a program to implement International Data Encryption Algorithm (IDEA)

DESCRIPTION:

The Simplified **International Data Encryption Algorithm (IDEA)** is a **symmetric key block cipher** that:

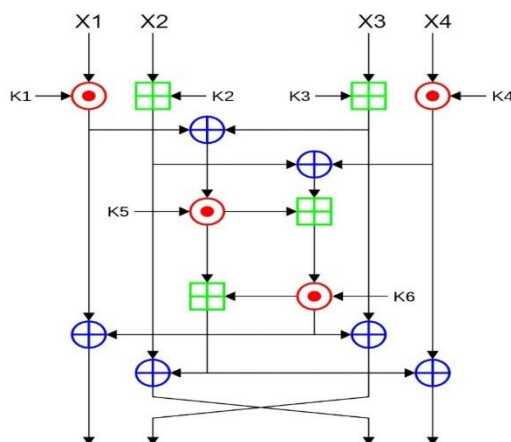
- uses a fixed-length plaintext of **16 bits** and
- encrypts them in **4 chunks of 4 bits** each
- to produce **16 bits ciphertext**.
- The length of the key used is **32 bits**.
- The key is also divided into 8 blocks of 4 bits each.

This algorithm involves a series of 4 identical complete rounds and 1 half-round. Each complete round involves a series of 14 steps that includes operations like:

- Bitwise XOR
- Addition modulo
- Multiplication modulo

After 4 complete rounds, the final “half-round” consists of only the first 4 out of the 14 steps previously used in the full rounds. To perform these rounds, each binary notation must be converted to its equivalent decimal notation, perform the operation and the result obtained should be converted back to the binary representation for the final result of that particular step.

International Data Encryption Algorithm(IDEA)



Where,



= Modular Addition



= Modular Multiplication



= Bitwise XOR



PROGRAM:

```
import java.io.*;

public class IdeaCipher extends BlockCipher
{

    // Constructor, string key.
    public IdeaCipher( String keyStr )
        {
            super( 16, 8 );
            setKey( keyStr );
        }

    // Constructor, byte-array key.
    public IdeaCipher( byte[] key )
        {
            super( 16, 8 );
            setKey( key );
        }

    // Key routines.

    private int[] encryptKeys = new int[52];
    private int[] decryptKeys = new int[52];

    /// Set the key.
    public void setKey( byte[] key )
        {
            int k1, k2, j;
            int t1, t2, t3;

            // Encryption keys. The first 8 key values come from the 16
            // user-supplied key bytes.
            for ( k1 = 0; k1 < 8; ++k1 )
                encryptKeys[k1] =
                    ( ( key[2 * k1] & 0xff ) << 8 ) | ( key[ 2 * k1 + 1] & 0xff );
            for ( ; k1 < 52; ++k1 )
                encryptKeys[k1] =
                    ( ( encryptKeys[k1 - 8] << 9 ) |
                    ( encryptKeys[k1 - 7] >>> 7 ) ) & 0xffff;
            k1 = 0;
            k2 = 51;
            t1 = mulinv( encryptKeys[k1++] );
            t2 = -encryptKeys[k1++];
```

```

        t3 = -encryptKeys[k1++];
        decryptKeys[k2--] = mulinv( encryptKeys[k1++] );
        decryptKeys[k2--] = t3;
        decryptKeys[k2--] = t2;
        decryptKeys[k2--] = t1;
        for ( j = 1; j < 8; ++j )
            {
                t1 = encryptKeys[k1++];
                decryptKeys[k2--] = encryptKeys[k1++];
                decryptKeys[k2--] = t1;
                t1 = mulinv( encryptKeys[k1++] );
                t2 = -encryptKeys[k1++];
                t3 = -encryptKeys[k1++];
                decryptKeys[k2--] = mulinv( encryptKeys[k1++] );
                decryptKeys[k2--] = t2;
                decryptKeys[k2--] = t3;
                decryptKeys[k2--] = t1;
            }
        t1 = encryptKeys[k1++];
        decryptKeys[k2--] = encryptKeys[k1++];
        decryptKeys[k2--] = t1;
        t1 = mulinv( encryptKeys[k1++] );
        t2 = -encryptKeys[k1++];
        t3 = -encryptKeys[k1++];
        decryptKeys[k2--] = mulinv( encryptKeys[k1++] );
        decryptKeys[k2--] = t3;
        decryptKeys[k2--] = t2;
        decryptKeys[k2--] = t1;
    }

    private int[] tempShorts = new int[4];
    public void encrypt( byte[] clearText, int clearOff, byte[] cipherText, int
cipherOff )
    {
        squashBytesToShorts( clearText, clearOff, tempShorts, 0, 4 );
        idea( tempShorts, tempShorts, encryptKeys );
        spreadShortsToBytes( tempShorts, 0, cipherText, cipherOff, 4 );
    }

    public void decrypt( byte[] cipherText, int cipherOff, byte[] clearText, int
clearOff )
    {
        squashBytesToShorts( cipherText, cipherOff, tempShorts, 0, 4 );
        idea( tempShorts, tempShorts, decryptKeys );
        spreadShortsToBytes( tempShorts, 0, clearText, clearOff, 4 );
    }.

```

```

private void idea( int[] inShorts, int[] outShorts, int[] keys )
{
    int x1, x2, x3, x4, k, t1, t2;

    x1 = inShorts[0];
    x2 = inShorts[1];
    x3 = inShorts[2];
    x4 = inShorts[3];
    k = 0;
    for ( int round = 0; round < 8; ++round )
        {
            x1 = mul ( x1 & 0xffff, keys[k++] );
            x2 = x2 + keys[k++];
            x3 = x3 + keys[k++];
            x4 = mul ( x4 & 0xffff, keys[k++] );
            t2 = x1 ^ x3;
            t2 = mul ( t2 & 0xffff, keys[k++] );
            t1 = t2 + ( x2 ^ x4 );
            t1 = mul ( t1 & 0xffff, keys[k++] );
            t2 = t1 + t2;
            x1 ^= t1;
            x4 ^= t2;
            t2 ^= x2;
            x2 = x3 ^ t1;
            x3 = t2;
        }
    outShorts[0] = mul ( x1 & 0xffff, keys[k++] ) & 0xffff;
    outShorts[1] = ( x3 + keys[k++] ) & 0xffff;
    outShorts[2] = ( x2 + keys[k++] ) & 0xffff;
    outShorts[3] = mul ( x4 & 0xffff, keys[k++] ) & 0xffff;
}

private static int mul( int a, int b )
{
    int ab = a * b;
    if ( ab != 0 )
        {
            int lo = ab & 0xffff;
            int hi = ab >>> 16;
            return ( ( lo - hi ) + ( lo < hi ? 1 : 0 ) ) & 0xffff;
        }
    if ( a != 0 )
        return ( 1 - a ) & 0xffff;
    return ( 1 - b ) & 0xffff;
}

```

```

private static int mulinv( int x )
{
    int t0, t1, q, y;
    if ( x <= 1 )
        return x;          // 0 and 1 are self-inverse
    t0 = 1;
    t1 = 0x10001 / x;      // since x >= 2, this fits into 16 bits
    y = ( 0x10001 % x ) & 0xffff;
    for (;;)
    {
        if ( y == 1 )
            return ( 1 - t1 ) & 0xffff;
        q = x / y;
        x = x % y;
        t0 = ( t0 + q * t1 ) & 0xffff;
        if ( x == 1 )
            return t0;
        q = y / x;
        y = y % x;
        t1 = ( t1 + q * t0 ) & 0xffff;
    }
}

/// Test routine.
public static void main( String[] args )
{
    // Check that mul and mulinv are working for all 16-bit numbers.
    for ( int a = 0; a < 65536; ++a )
    {
        int b = mulinv( a );
        int c = mul( a, b );
        if ( c != 1 )
            System.err.println( "mul/mulinv flaw: " + a + " * " + b + " = " + c );
    }
}
}

```

RESULT:

International Data encryption algorithm had been implemented successfully

EX. NO: 4

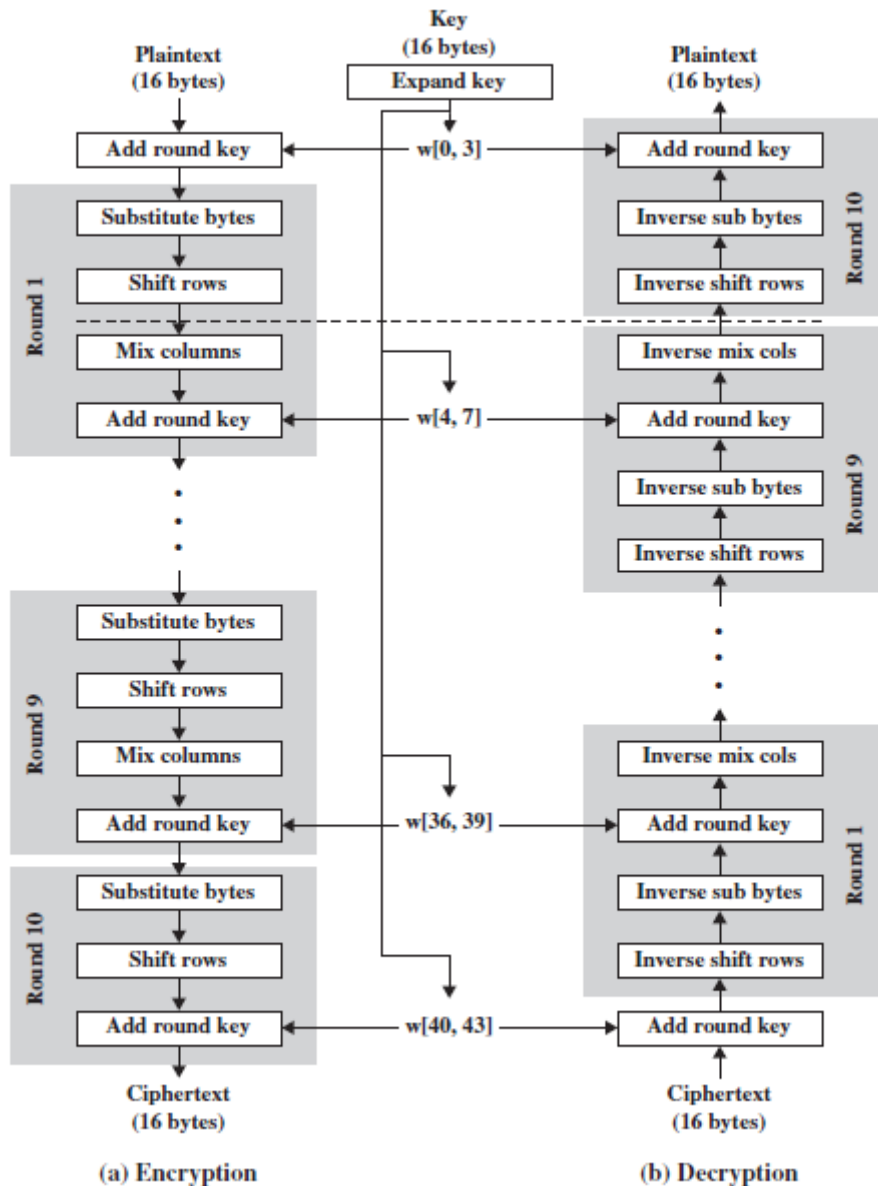
IMPLEMENTATION OF AES

AIM:

To write a program to implement Advanced Encryption Standard (AES)

DESCRIPTION:

The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.



The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a 4×4 square matrix of bytes. This block is copied into the **State** array, which is modified at each stage of encryption or decryption. After the final stage, **State** is copied to an output matrix. Similarly, the key is depicted as a square matrix of bytes. This key is then expanded

into an array of key schedule words. Each word is four bytes, and the total key schedule is 44 words for the 128-bit key. Note that the ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the **in** matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the **w** matrix.

The cipher consists of N rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key. The first $N - 1$ rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The final round contains only three transformations, and there is an initial single transformation (AddRoundKey) before the first round, which can be considered Round 0. Each transformation takes one or more $4 * 4$ matrices.

PROGRAM:

```
package com.includehelp.stringsample;

import java.util.Base64; import java.util.Scanner; import
javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec; import
javax.crypto.spec.SecretKeySpec;

/**
 * Program to Encrypt/Decrypt String Using AES 128 bit Encryption
Algorithm
 */
public class EncryptDecryptString
{
    private static final String encryptionKey = "ABCDEFGHijklmnop";
    private static final String characterEncoding = "UTF-8";
    private static final String cipherTransformation=
"AES/CBC/PKCS5PADDING"; private static final String
aesEncryptionAlgorithem = "AES";
/**
 * Method for Encrypt Plain String Data
 * @param plainText
 * @return encryptedText
 */
    public static String encrypt(String plainText)
    {
        String encryptedText = ""; try
        {
            Cipher cipher = Cipher.getInstance(cipherTransformation);
            byte[] key= encryptionKey.getBytes(characterEncoding);
            SecretKeySpec secretKey = new SecretKeySpec(key,
aesEncryptionAlgorithem); IvParameterSpec ivparameterspec
= new IvParameterSpec(key);
            cipher.init(Cipher.ENCRYPT_MODE, secretKey,
ivparameterspec);
byte[] cipherText = cipher.doFinal(plainText.getBytes("UTF8"));
Base64.Encoder encoder = Base64.getEncoder();
```

```

encryptedText = encoder.encodeToString(cipherText);
} catch (Exception E)
{
System.err.println("Encrypt Exception : "+E.getMessage());
}
return encryptedText;
}

    public static String decrypt(String encryptedText)
    {
        String decryptedText = ""; try
        {
            Cipher cipher =
                Cipher.getInstance(cipherTransformation); byte[] key =
                encryptionKey.getBytes(characterEncoding);
            SecretKeySpec secretKey = new SecretKeySpec(key,
                aesEncryptionAlgorithem); IvParameterSpec
                ivparameterspec = new IvParameterSpec(key);
            cipher.init(Cipher.DECRYPT_MODE, secretKey,
                ivparameterspec); Base64.Decoder decoder =
                Base64.getDecoder();
            byte[] cipherText =
                decoder.decode(encryptedText.getBytes("UTF8"));
            decryptedText = new String(cipher.doFinal(cipherText),
                "UTF-8");
        } catch (Exception E)
        {
            System.err.println("decrypt Exception :
                "+E.getMessage());
        }
        return decryptedText;
    }

    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in); System.out.println("Enter
        String : "); String plainString = sc.nextLine();

        String encryptStr = encrypt(plainString); String decryptStr =
        decrypt(encryptStr);

        System.out.println("Plain String : "+plainString);
        System.out.println("Encrypt String : "+encryptStr);
        System.out.println("Decrypt String : "+decryptStr);
    }
}

```

OUTPUT:

```

Enter String : Hello World
Plain String : Hello World
Encrypt String : IMfL/ifkuvkZwG/v2bn6Bw==
Decrypt String : Hello World

```


VIVA QUESTIONS (PRELAB and POSTLAB):

1. AES follows

a) Hash Algorithm b) Caesars Cipher c) Feistel Cipher Structure d) SP Networks

2. The AES Algorithm Cipher System consists of _____ rounds (iterations) each with a round key

a) 12 b) 18 c) 9 d) 16

3. The AES algorithm has a key length of

a) 128 Bits b) 32 Bits c) 64 Bits d) 16 Bits

4. In the AES algorithm, although the key size is 64 bits only 48bits are used for the encryption procedure, the rest are parity bits.

a) True b) False

5. In the AES algorithm the round key is _____ bit and the Round Input is _____ bits.

a) 48, 32 b) 64,32 c) 56, 24 d) 32, 32

6. In the AES algorithm the Round Input is 32 bits, which is expanded to 48 bits via _____

a) Scaling of the existing bits b) Duplication of the existing bits
c) Addition of zeros d) Addition of ones

7. The Initial Permutation table/matrix is of size

a) 16×8 b) 12×8 c) 8×8 d) 4×8

8. The number of unique substitution boxes in AES after the 48 bit XOR operation are a) 8 b) 4 c) 6 d) 12

9. In the AES algorithm the 64 bit key input is shortened to 56 bits by ignoring every 4th bit.

a) True b) False

RESULT:

The program to implement AES encryption technique was developed and executed successfully.

EX. NO: 5

IMPLEMENTATION OF RSA

AIM:

To write a program to implement the RSA encryption algorithm.

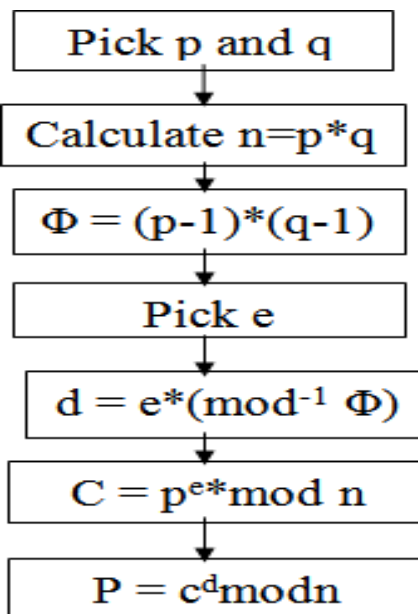
DESCRIPTION:

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

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

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

EXAMPLE:



ALGORITHM:

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

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

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

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

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

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

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

PROGRAM:(RSA)

```
#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);

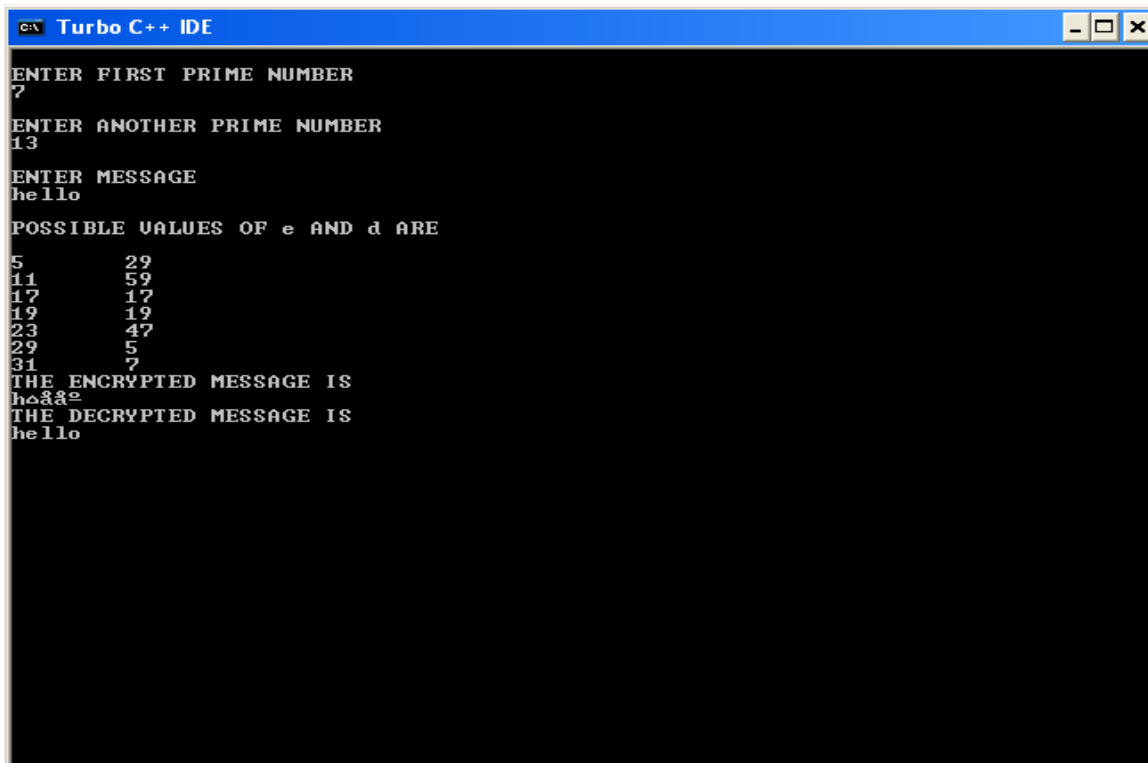
```

```

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:



```
GV Turbo C++ IDE
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
hΔ33e
THE DECRYPTED MESSAGE IS
hello
```

VIVA QUESTIONS

1. What Is RSA?
2. Are Strong Primes Necessary In RSA?
3. Can Users Of RSA Run Out Of Distinct Primes?
4. What Are The Alternatives To RSA?
5. How fast is RSA?
6. What would it take to break RSA?
7. How large a modulus key should be used in RSA?
8. How large should the primes be?
9. How do you know if a number is prime?
10. How is RSA used for authentication in practice?

RESULT:

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

EX. NO: 6

IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

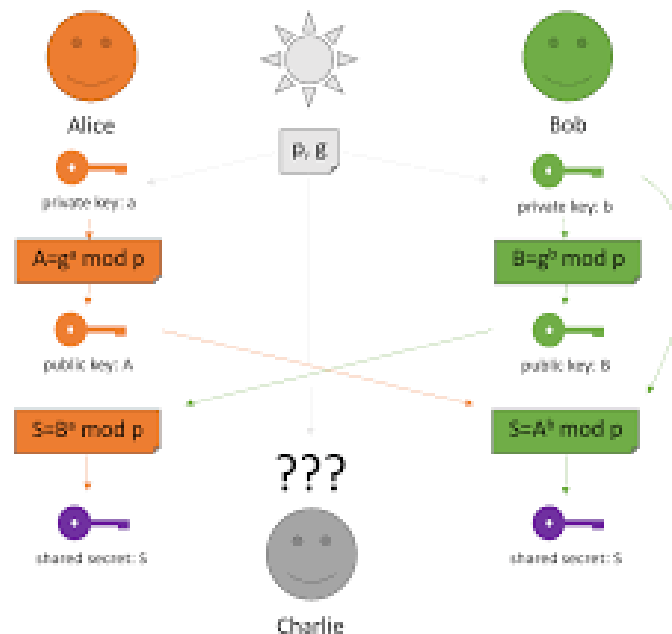
AIM:

To implement the Diffie-Hellman Key Exchange algorithm

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.

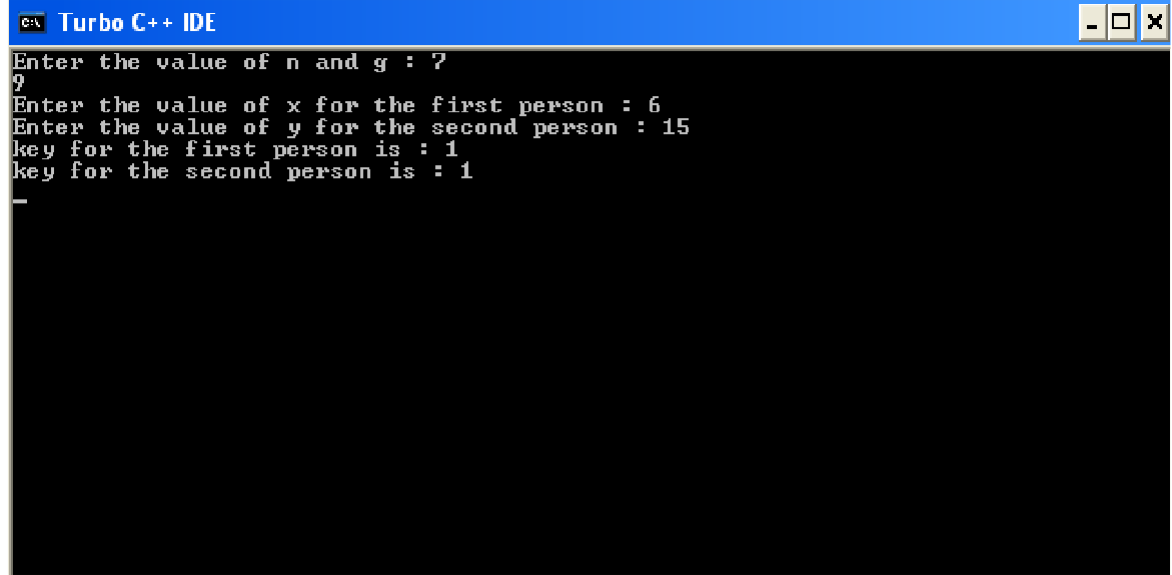
STEP-5: Similarly Bob also selects a public key b and computes his secret key as B and sends the same back to Alice.

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

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:



```
Enter the value of n and g : ?
9
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
-
```

VIVA QUESTIONS

1. What's the difference between Diffie-Hellman and RSA?
2. Does Diffie Hellman guarantee secrecy?
3. Why is RSA preferred over Diffie-Hellman if they are both used to establish shared key?
4. Are there any one way operations that could be used for Diffie-Hellman post quantum?
5. Why is Diffie-Hellman required when RSA is already used for key exchange in TLS?
6. What is Authenticated Diffie-Hellman Key Agreement?
7. How secure is ECDH if the public keys are never shared?
8. Which is better when the secret is leaked, RSA or Diffie-Hellman?
9. What role does RSA play in DH-RSA cipher suite?
10. Why is Diffie-Hellman used alongside public keys?
11. Is Diffie-Hellman key exchange based on one-way function or trapdoor function?

RESULT:

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

EX. NO: 7

IMPLEMENTATION OF MD5

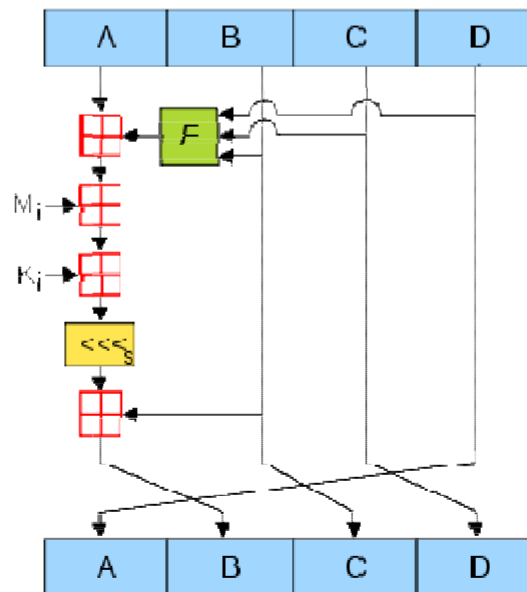
AIM:

To write a program to implement the MD5 hashing technique.

DESCRIPTION:

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

EXAMPLE:



ALGORITHM:

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

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

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

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

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

PROGRAM:(MD5)

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

```

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

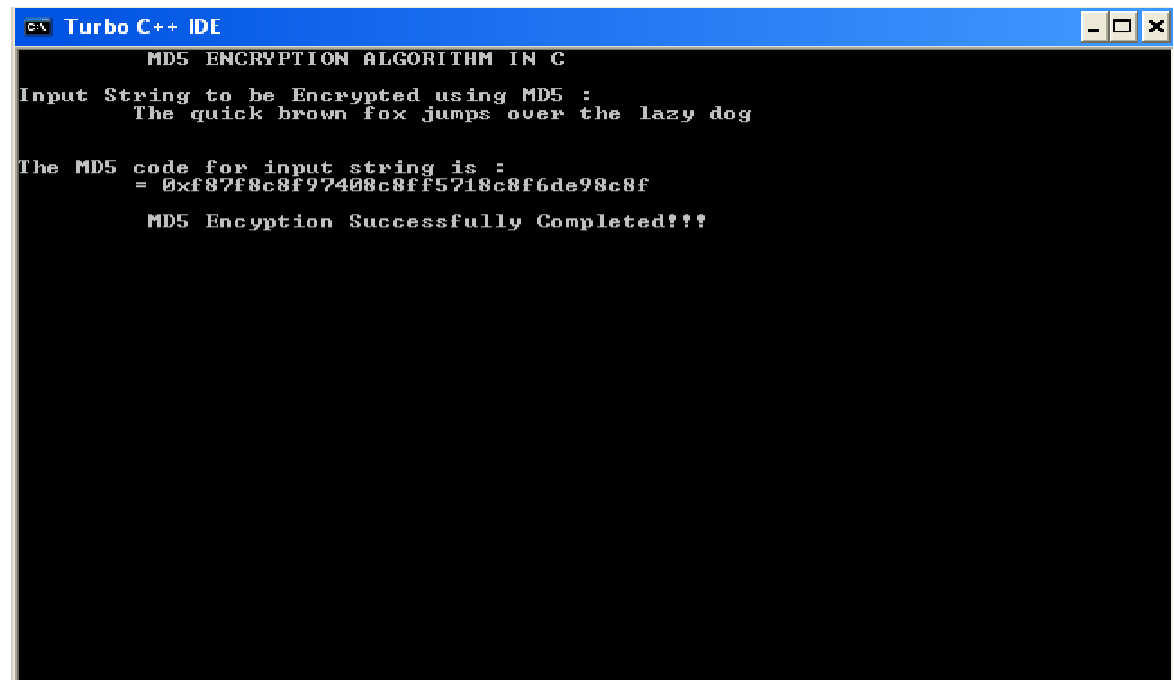
```

```

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

```

OUTPUT:

A screenshot of the Turbo C++ IDE window. The title bar reads "Turbo C++ IDE". The main window contains the following text:

```
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 Encyption Successfully Completed!!!
```

RESULT:

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

EX. NO: 8

IMPLEMENTATION OF SHA-I

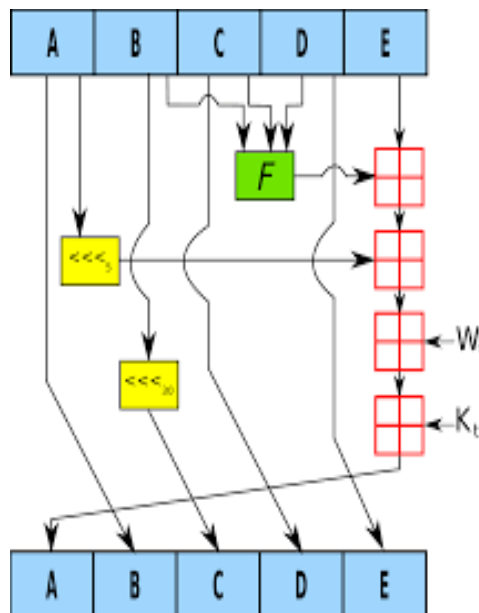
AIM:

To implement the SHA-I hashing technique

DESCRIPTION:

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

EXAMPLE:



ALGORITHM:

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

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

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

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

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

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

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

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

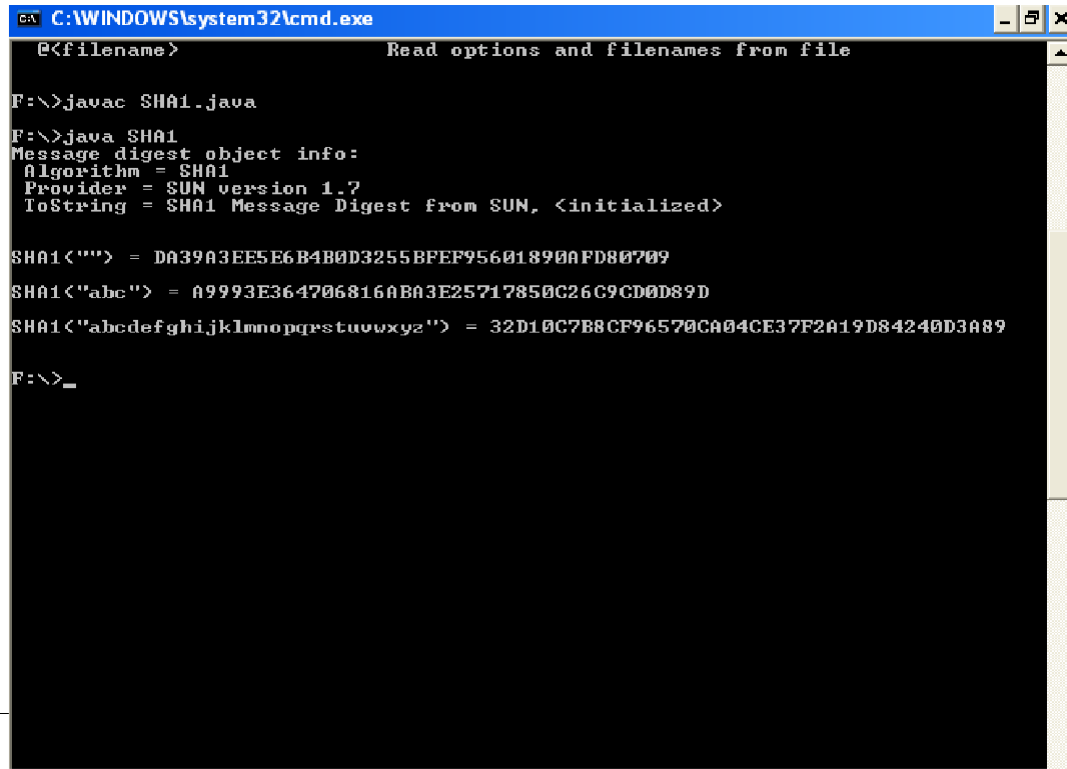
PROGRAM: (Secure Hash Algorithm)

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



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

OUTPUT:



```
C:\WINDOWS\system32\cmd.exe  
@<filename> Read options and filenames from file  
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:\>_
```

VIVA QUESTIONS

1. SHA-1 produces a hash value of how many bits?
2. What is the number of round computation steps in the SHA-256 algorithm?
3. In SHA-512, the message is divided into blocks size of how many bits for the hash computation.
4. What is the maximum length of the message (in bits) that can be taken by SHA-512?
5. What is the size of W (in bits) in the SHA-512 processing of a single 1024-bit block?
6. In the SHA-512 processing of a single 1024-bit block, how the round constants are obtained?
7. What is the maximum length of the message (in bits) that can be taken by SHA-512?

RESULT:

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

EX. NO: 9

IMPLEMENTATION OF DIGITAL SIGNATURE STANDARD

AIM:

To write a program to implement the signature scheme named digital signature standard

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);
    }
}
```

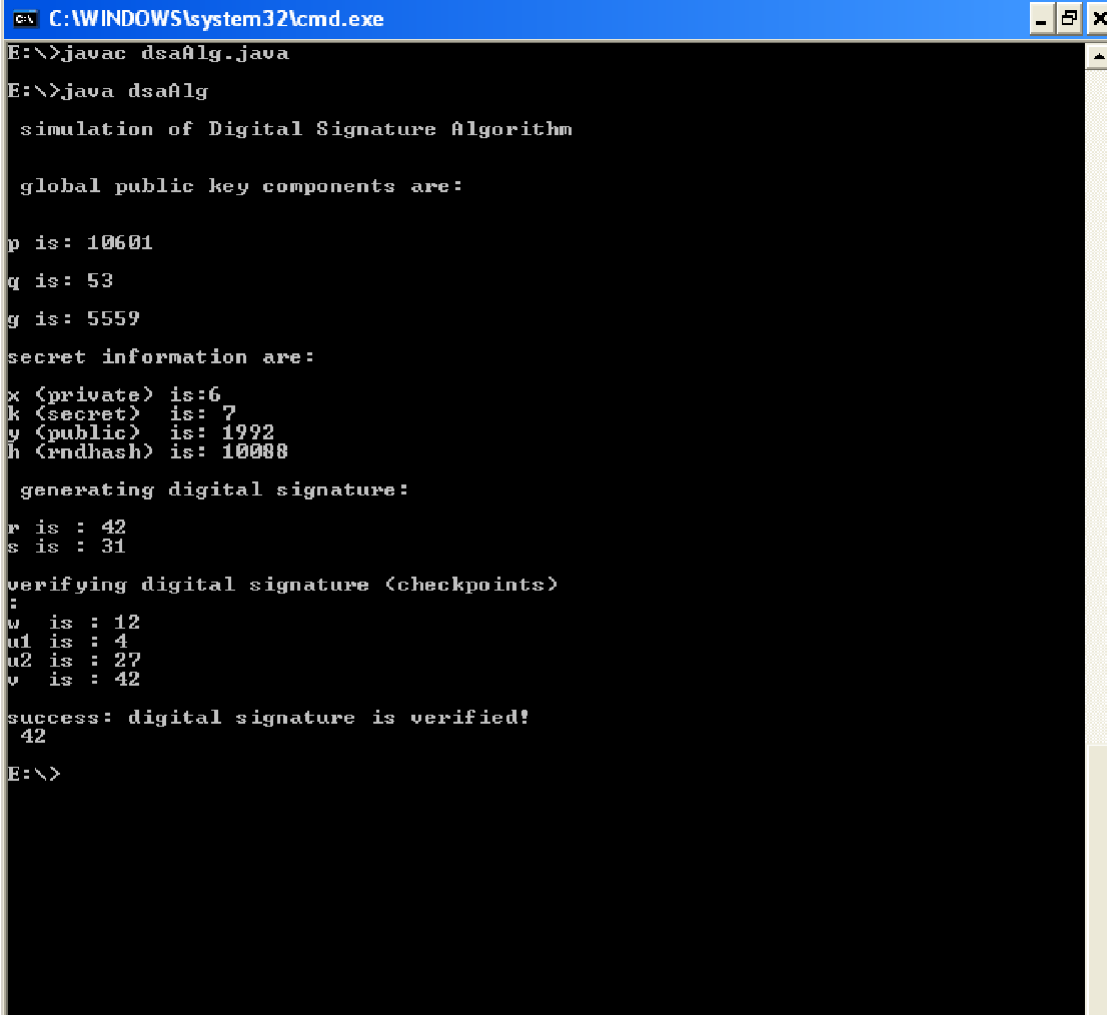
```

    }
    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:



```
C:\WINDOWS\system32\cmd.exe
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)
:
w is : 12
u1 is : 4
u2 is : 27
v is : 42

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

VIVA QUESTIONS

1. What is a digital signature?
2. What does a digital signature look like?
3. What is an electronic document?
4. Does that mean that the authenticity of any electronic document can be verified by a digital signature?
5. What is it like to actually sign an electronic document?
6. Can you actually see the signer's handwritten signature?
7. How do I get a digital signature certificate?
8. What is a certificate? What does it mean to "publish" a certificate?
9. How am I identified as the signer?
10. If my private key is stored on my computer, can't someone sign the documents without my permission by getting access to the computer?
11. Can a digital signature be forged?
12. What are the responsibilities and the liability of a digital signature certificate subscriber?
13. What are the practical uses of a digital signature?

RESULT:

Digital signature method program is implemented and executed successfully.

EX. NO: 10

IMPLEMENTATION OF FOSS BASED SECURITY MECHANISM

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.

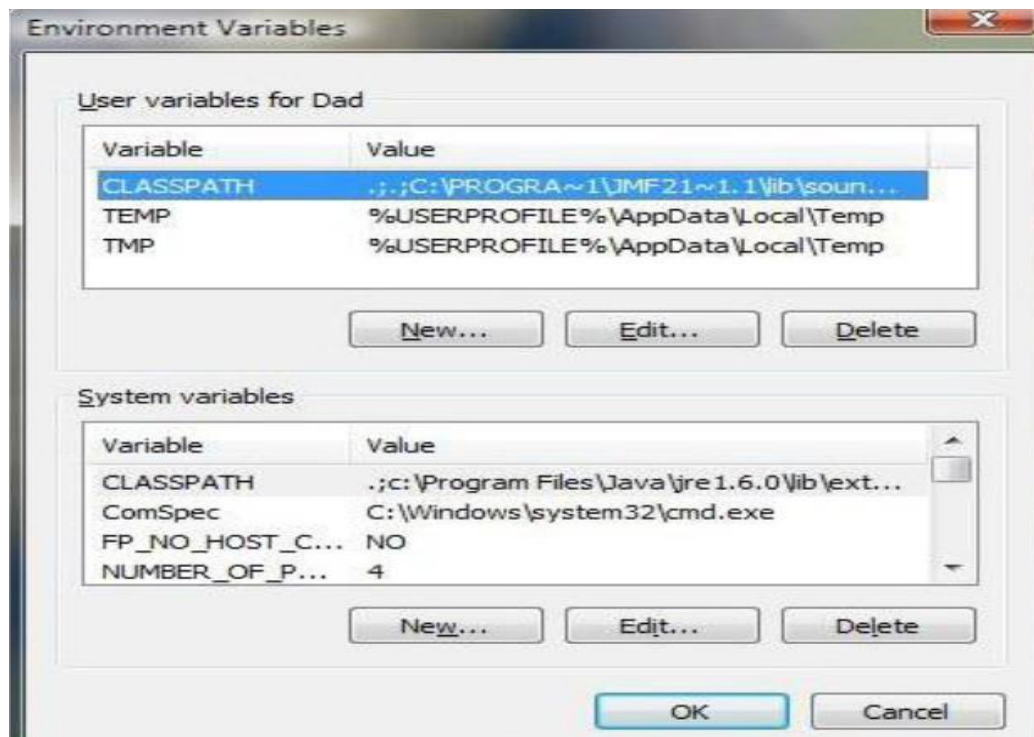
PROCEDURE:

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
Len: 50
=====
```

```
Administrator: C:\Windows\system32\cmd.exe
-----
Run time for packet processing was 703.909000 seconds
Snort processed 1409 packets.
Snort ran for 0 days 0 hours 11 minutes 43 seconds
  Pkts/min:      128
  Pkts/sec:       2
-----
Packet I/O Totals:
  Received:      1411
  Analyzed:      1409 < 99.858%>
  Dropped:       0 < 0.000%>
  Filtered:      0 < 0.000%>
  Outstanding:   2 < 0.142%>
  Injected:      0
-----
Breakdown by protocol (includes rebuilt packets):
  Eth:           1409 <100.000%>
  ULAN:          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 ULAN:      0 < 0.000%>
  GRE IP4:       0 < 0.000%>
  GRE IP6:       0 < 0.000%>
  GRE IP6 Ext:   0 < 0.000%>
  GRE PPTP:      0 < 0.000%>
  GRE ARP:       0 < 0.000%>
  GRE IPX:       0 < 0.000%>
  GRE Loop:      0 < 0.000%>
  MPLS:          0 < 0.000%>
  ARP:           7 < 0.639%>
  IPX:           0 < 0.000%>
  Eth Loop:      0 < 0.000%>
  Eth Disc:      0 < 0.000%>
  IP4 Disc:      0 < 0.000%>
  IP6 Disc:      0 < 0.000%>
  TCP Disc:      0 < 0.000%>
  UDP Disc:      0 < 0.000%>
  ICMP Disc:     0 < 0.000%>
  All Discard:   0 < 0.000%>
  Other:         35 < 2.484%>
  Bad Chk Sum:   0 < 0.000%>
  Bad TTL:       0 < 0.000%>
  S5 G 1:        0 < 0.000%>
  S5 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.