

EX. NO: 4

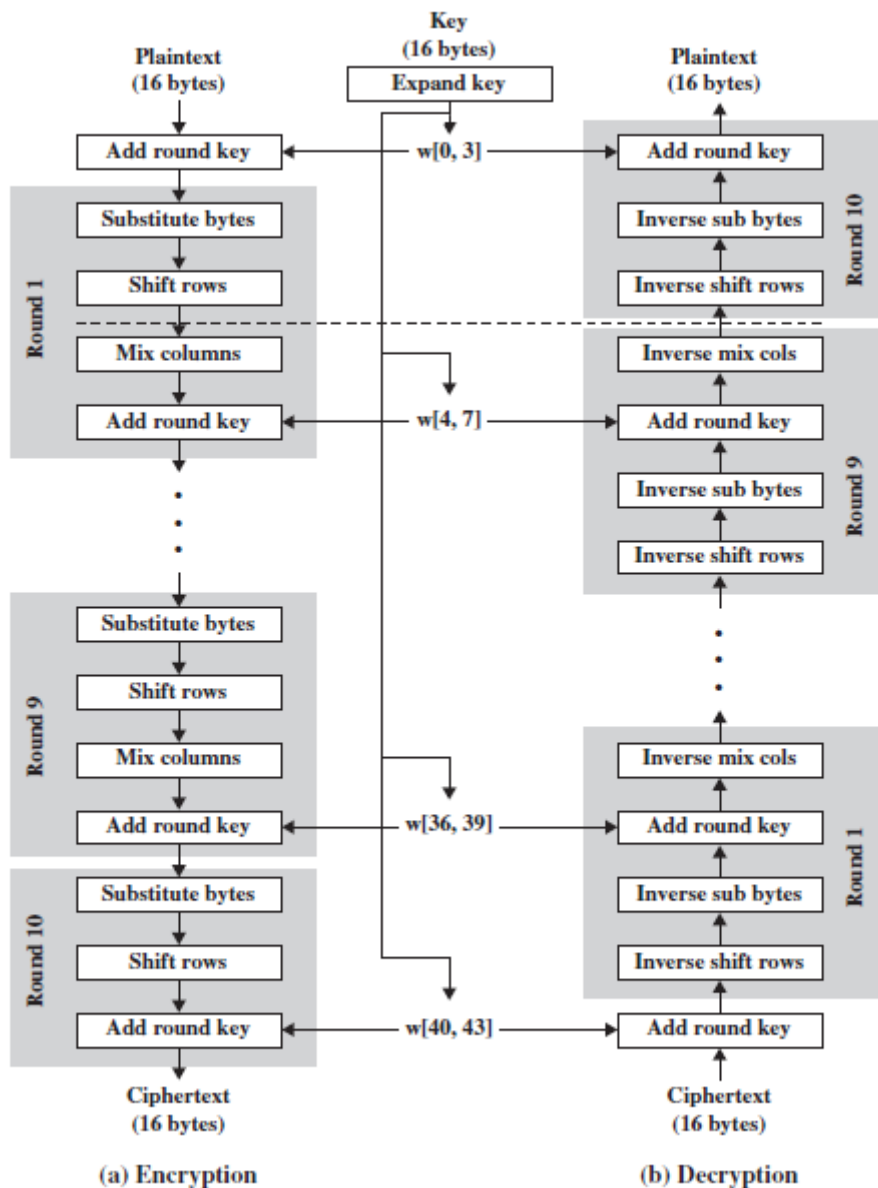
IMPLEMENTATION OF AES

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 a 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 = "ABCDEFGHJKLMNOP";
    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,
            aesEncryptionAlgorithm); 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: 2**IMPLEMENTATION OF DES****AIM:**

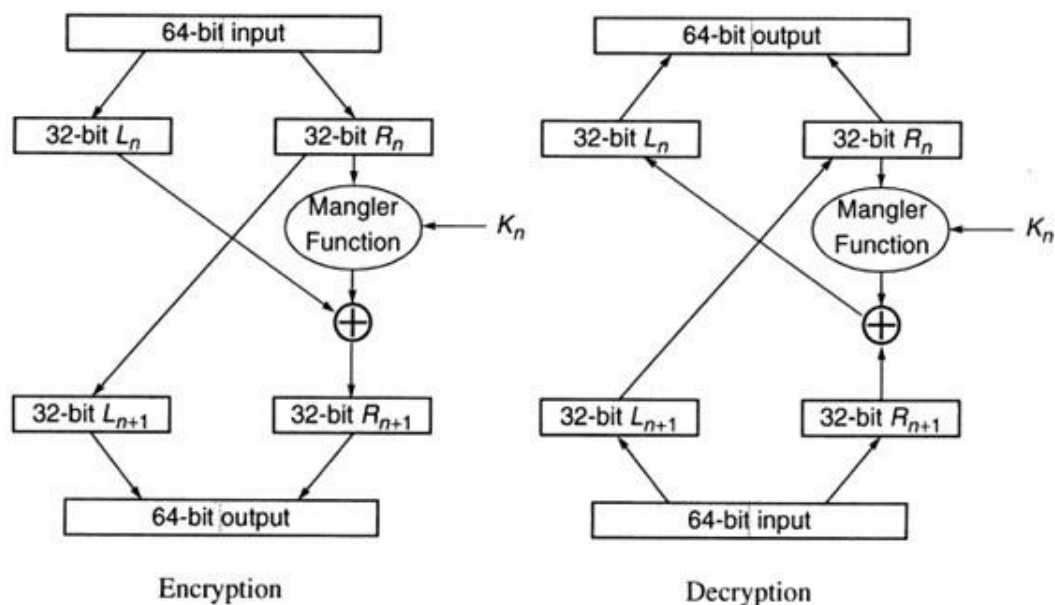
To write a program to implement Data Encryption Standard (DES)

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

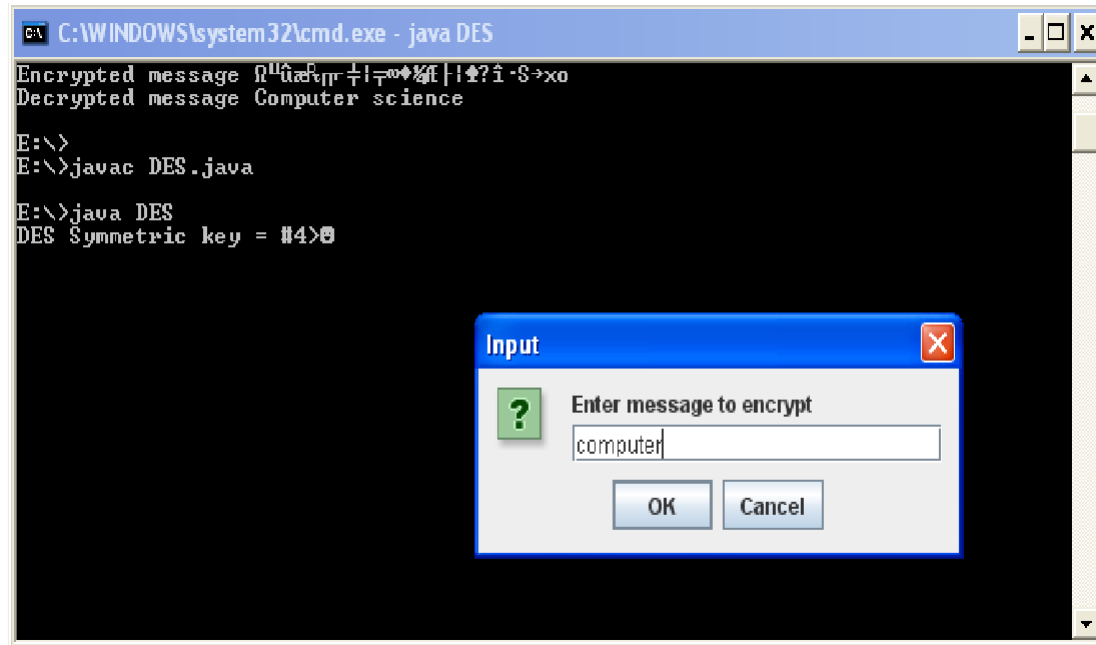
```
import javax.swing.*;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random ;
class DES {
    byte[] skey = new byte[1000];
    String skeyString;
    static byte[] raw;
    String inputMessage, encryptedData, decryptedMessage;
public DES()
{
    try
    {
        generateSymmetricKey();
        inputMessage=JOptionPane.showInputDialog(null,"Enter
        message to encrypt");
        byte[] 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 R^u^æR^p÷!7^oo^%f|!±?î·S→xo
Decrypted message Computer science

E:\>
E:\>javac DES.java

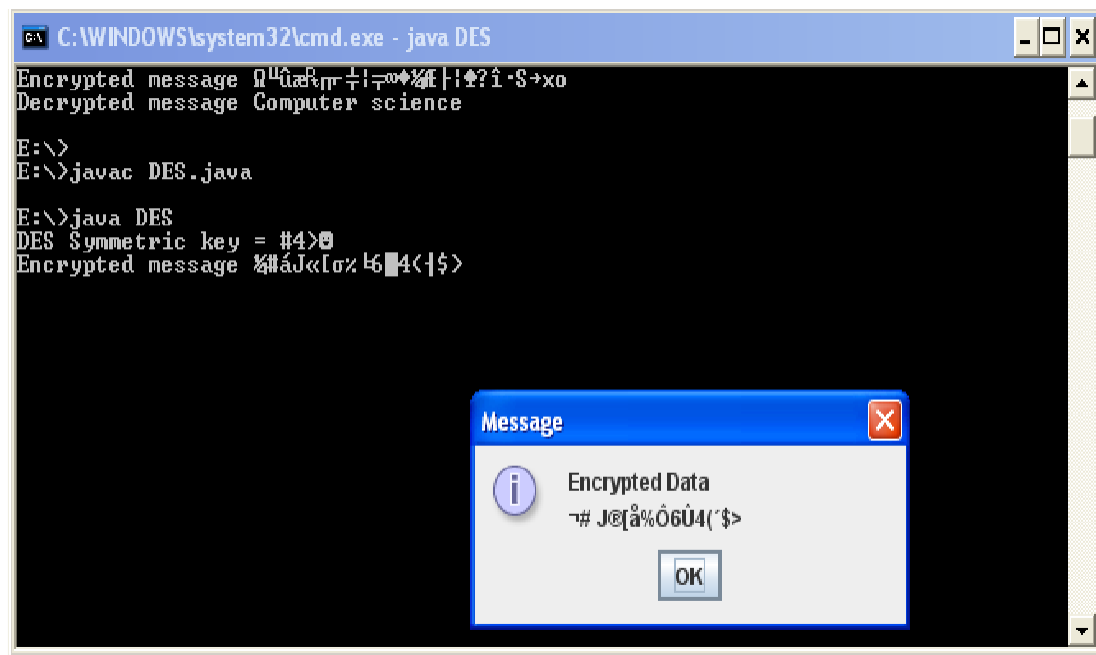
E:\>java DES
DES Symmetric key = #4>0
```

Input

Enter message to encrypt

computer

OK Cancel



```
C:\WINDOWS\system32\cmd.exe - java DES
Encrypted message R^u^æR^p÷!7^oo^%f|!±?î·S→xo
Decrypted message Computer science

E:\>
E:\>javac DES.java

E:\>java DES
DES Symmetric key = #4>0
Encrypted message %HâJ«[σ%6■4<|$>
```

Message

Encrypted Data

~# J@[â%Ô6Ô4(' \$>

OK

EX. NO: 6

IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE ALGORITHM

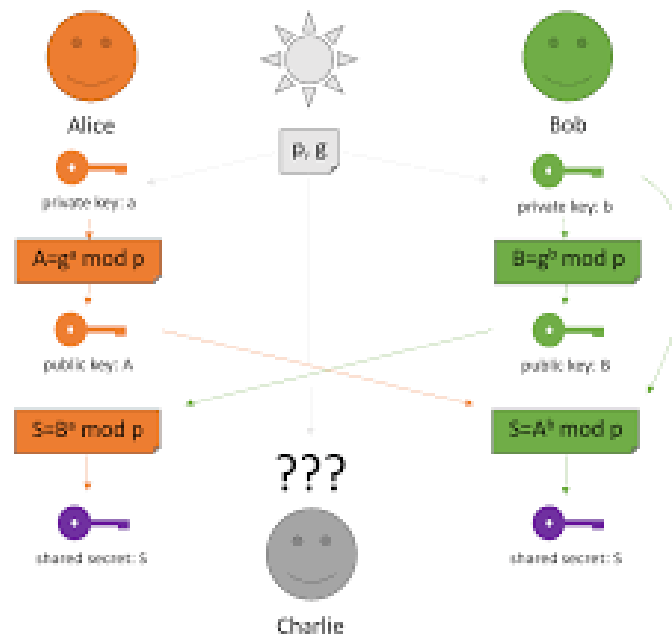
AIM:

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.

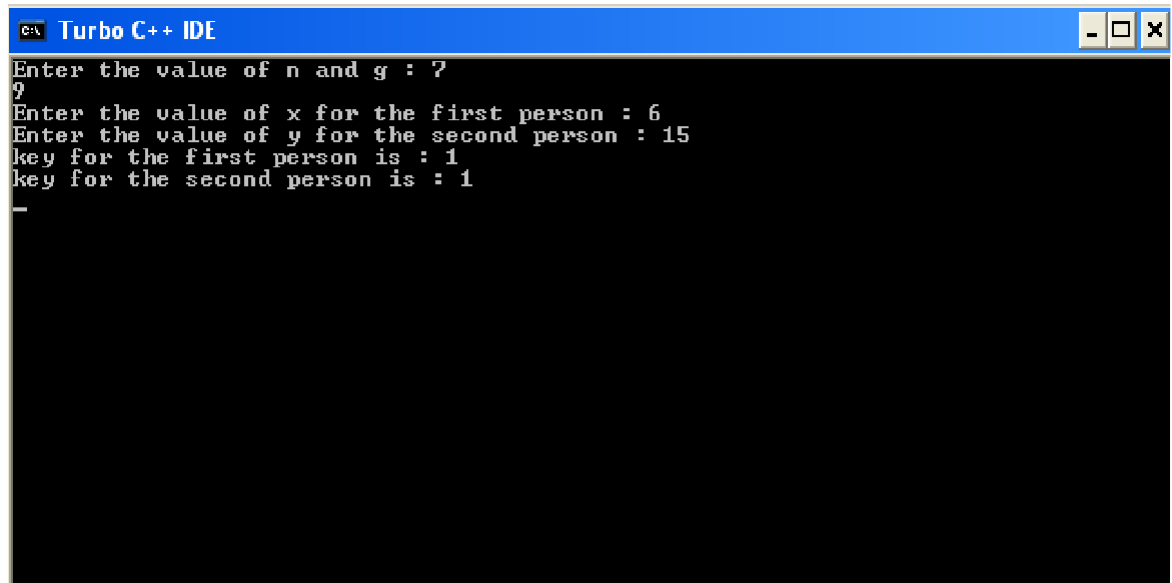
STEP-5: Similarly Bob also selects a public key b and computes his secret key as B and sends the same back to Alice.

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



```
c:\ Turbo C++ IDE
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: 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.

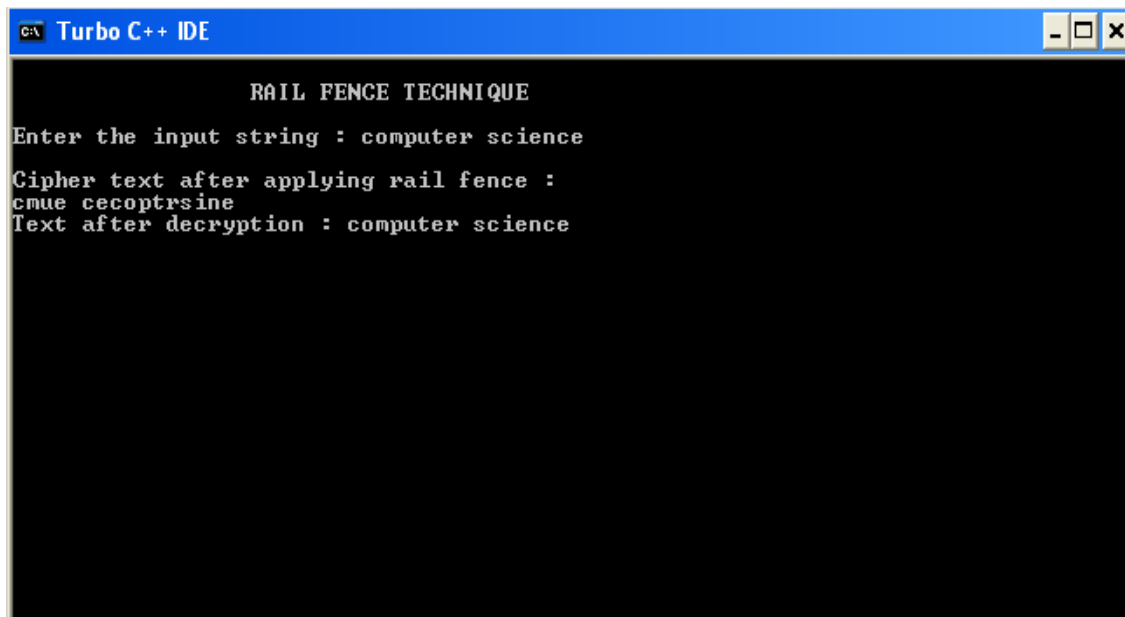
PROGRAM: (Rail Fence)

```
#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:



```
RAIL FENCE TECHNIQUE
Enter the input string : computer science
Cipher text after applying rail fence :
cmue cecoptrsine
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.