28.08.2020

Ex No: 3

IMPLEMENTATION OF DES

AIM:

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

DESCRIPTION:

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

The main parts of the algorithm are as follows:

Fractioning of the text into 64-bit
blocks Initial permutation of blocks

Breakdown of the blocks into two parts: left and right, named L and
R Permutation and substitution steps repeated 16 times

Re-joining of the left and right parts then inverse initial permutation

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 originalsecond 32-bit sequence forms the first part.

STEP-5: Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the sameprocess 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:

```
Encryption:

After initial permutation: 14A7D67818CA18AD
After splitting: L0=14A7D678 R0=18CA18AD

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

Cipher Text: C0B7A8D05F3A829C
```

RESULT:

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