Name:-Amrit Ranjan

Reg No:-21BCE1847

Cryptography And Network Security

Assignment 2

Faculty Name:-Dr Tapabarta Roy

DES



Simplified Des Instruction

Ques:-

Take a simplified version of DES which uses a key and a plaintext both of 8 bits. The plaintext undergoes an encryption function given below. Plaintext P is divided into two equal parts L (4 leftmost bits) and R (4 rightmost bits). Both of them are shifted cyclically towards left by one bit. Next both of them are put inside a four bits sbox [0, f, 1, e, 2, d, 3, c, 4, b, 5, a, 6, 9, 7, 8]. Then the remaining 8 bit output is given the following permutation [0, 2, 4, 6, 1, 3, 5, 7]. Finally, The 8 bit round key is XORed with the output. This output is put inside the same above algorithm again and after the completion of the second round what remains will be called the output. The key scheduling is the same as the round function for the encryption except for the fact that it has no add round key round. Write down a code for the above encryption and the corresponding decryption choosing your own plaintext and key pairs and show that the decrypted version matches with the plaintext.

Code:-In Java

/\*package whatever //do not write package name here \*/

import java.io.\*;

public class DES {

    // int key[]= {0,0,1,0,0,1,0,1,1,1};

    int key[] = {

        1, 0, 1, 0, 0, 0, 0, 0, 1, 0

    }; // extra example for checking purpose

    int P10[] = { 3, 5, 2, 7, 4, 10, 1, 9, 8, 6 };

    int P8[] = { 6, 3, 7, 4, 8, 5, 10, 9 };

    int key1[] = new int[8];

    int key2[] = new int[8];

    int[] IP = { 2, 6, 3, 1, 4, 8, 5, 7 };

    int[] EP = { 4, 1, 2, 3, 2, 3, 4, 1 };

    int[] P4 = { 2, 4, 3, 1 };

    int[] IP\_inv = { 4, 1, 3, 5, 7, 2, 8, 6 };

    int[][] S0 = { { 1, 0, 3, 2 },

                { 3, 2, 1, 0 },

                { 0, 2, 1, 3 },

                { 3, 1, 3, 2 } };

    int[][] S1 = { { 0, 1, 2, 3 },

                { 2, 0, 1, 3 },

                { 3, 0, 1, 0 },

                { 2, 1, 0, 3 } };

    // this function basically generates the key(key1 and

    //key2)  using P10 and P8 with (1 and 2)left shifts

    void key\_generation()

    {

        int key\_[] = new int[10];

        for (int i = 0; i < 10; i++) {

            key\_[i] = key[P10[i] - 1];

        }

        int Ls[] = new int[5];

        int Rs[] = new int[5];

        for (int i = 0; i < 5; i++) {

            Ls[i] = key\_[i];

            Rs[i] = key\_[i + 5];

        }

        int[] Ls\_1 = shift(Ls, 1);

        int[] Rs\_1 = shift(Rs, 1);

        for (int i = 0; i < 5; i++) {

            key\_[i] = Ls\_1[i];

            key\_[i + 5] = Rs\_1[i];

        }

        for (int i = 0; i < 8; i++) {

            key1[i] = key\_[P8[i] - 1];

        }

        int[] Ls\_2 = shift(Ls, 2);

        int[] Rs\_2 = shift(Rs, 2);

        for (int i = 0; i < 5; i++) {

            key\_[i] = Ls\_2[i];

            key\_[i + 5] = Rs\_2[i];

        }

        for (int i = 0; i < 8; i++) {

            key2[i] = key\_[P8[i] - 1];

        }

        System.out.println("Your Key-1 :");

        for (int i = 0; i < 8; i++)

            System.out.print(key1[i] + " ");

        System.out.println();

        System.out.println("Your Key-2 :");

        for (int i = 0; i < 8; i++)

            System.out.print(key2[i] + " ");

    }

    // this function is use full for shifting(circular) the

    //array n position towards left

    int[] shift(int[] ar, int n)

    {

        while (n > 0) {

            int temp = ar[0];

            for (int i = 0; i < ar.length - 1; i++) {

                ar[i] = ar[i + 1];

            }

            ar[ar.length - 1] = temp;

            n--;

        }

        return ar;

    }

    // this is main encryption function takes plain text as

    //input  uses another functions and returns the array of

    //cipher text

    int[] encryption(int[] plaintext)

    {

        int[] arr = new int[8];

        for (int i = 0; i < 8; i++) {

            arr[i] = plaintext[IP[i] - 1];

        }

        int[] arr1 = function\_(arr, key1);

        int[] after\_swap = swap(arr1, arr1.length / 2);

        int[] arr2 = function\_(after\_swap, key2);

        int[] ciphertext = new int[8];

        for (int i = 0; i < 8; i++) {

            ciphertext[i] = arr2[IP\_inv[i] - 1];

        }

        return ciphertext;

    }

    // decimal to binary string 0-3

    String binary\_(int val)

    {

        if (val == 0)

            return "00";

        else if (val == 1)

            return "01";

        else if (val == 2)

            return "10";

        else

            return "11";

    }

    // this function is doing core things like expansion

    // then xor with desired key then S0 and S1

    //substitution   P4 permutation and again xor    we have used

    //this function 2 times(key-1 and key-2) during

    //encryption and     2 times(key-2 and key-1) during

    //decryption

    int[] function\_(int[] ar, int[] key\_)

    {

        int[] l = new int[4];

        int[] r = new int[4];

        for (int i = 0; i < 4; i++) {

            l[i] = ar[i];

            r[i] = ar[i + 4];

        }

        int[] ep = new int[8];

        for (int i = 0; i < 8; i++) {

            ep[i] = r[EP[i] - 1];

        }

        for (int i = 0; i < 8; i++) {

            ar[i] = key\_[i] ^ ep[i];

        }

        int[] l\_1 = new int[4];

        int[] r\_1 = new int[4];

        for (int i = 0; i < 4; i++) {

            l\_1[i] = ar[i];

            r\_1[i] = ar[i + 4];

        }

        int row, col, val;

        row = Integer.parseInt("" + l\_1[0] + l\_1[3], 2);

        col = Integer.parseInt("" + l\_1[1] + l\_1[2], 2);

        val = S0[row][col];

        String str\_l = binary\_(val);

        row = Integer.parseInt("" + r\_1[0] + r\_1[3], 2);

        col = Integer.parseInt("" + r\_1[1] + r\_1[2], 2);

        val = S1[row][col];

        String str\_r = binary\_(val);

        int[] r\_ = new int[4];

        for (int i = 0; i < 2; i++) {

            char c1 = str\_l.charAt(i);

            char c2 = str\_r.charAt(i);

            r\_[i] = Character.getNumericValue(c1);

            r\_[i + 2] = Character.getNumericValue(c2);

        }

        int[] r\_p4 = new int[4];

        for (int i = 0; i < 4; i++) {

            r\_p4[i] = r\_[P4[i] - 1];

        }

        for (int i = 0; i < 4; i++) {

            l[i] = l[i] ^ r\_p4[i];

        }

        int[] output = new int[8];

        for (int i = 0; i < 4; i++) {

            output[i] = l[i];

            output[i + 4] = r[i];

        }

        return output;

    }

    // this function swaps the nibble of size n(4)

    int[] swap(int[] array, int n)

    {

        int[] l = new int[n];

        int[] r = new int[n];

        for (int i = 0; i < n; i++) {

            l[i] = array[i];

            r[i] = array[i + n];

        }

        int[] output = new int[2 \* n];

        for (int i = 0; i < n; i++) {

            output[i] = r[i];

            output[i + n] = l[i];

        }

        return output;

    }

    // this is main decryption function

    // here we have used all previously defined function

    // it takes cipher text as input and returns the array

    //of     decrypted text

    int[] decryption(int[] ar)

    {

        int[] arr = new int[8];

        for (int i = 0; i < 8; i++) {

            arr[i] = ar[IP[i] - 1];

        }

        int[] arr1 = function\_(arr, key2);

        int[] after\_swap = swap(arr1, arr1.length / 2);

        int[] arr2 = function\_(after\_swap, key1);

        int[] decrypted = new int[8];

        for (int i = 0; i < 8; i++) {

            decrypted[i] = arr2[IP\_inv[i] - 1];

        }

        return decrypted;

    }

    public static void main(String[] args)

    {

        DES obj = new DES();

        obj.key\_generation(); // call to key generation

                            // function

        // int []plaintext= {1,0,1,0,0,1,0,1};

        int[] plaintext = {

            1, 0, 0, 1, 0, 1, 1, 1

        }; // extra example for checking purpose

        System.out.println();

        System.out.println("Your plain Text is :");

        for (int i = 0; i < 8; i++) // printing the

                                    // plaintext

            System.out.print(plaintext[i] + " ");

        int[] ciphertext = obj.encryption(plaintext);

        System.out.println();

        System.out.println(

            "Your cipher Text is :"); // printing the cipher

                                    // text

        for (int i = 0; i < 8; i++)

            System.out.print(ciphertext[i] + " ");

        int[] decrypted = obj.decryption(ciphertext);

        System.out.println();

        System.out.println(

            "Your decrypted Text is :"); // printing the

                                        // decrypted text

        for (int i = 0; i < 8; i++)

            System.out.print(decrypted[i] + " ");

    }

}

OUTPUT:-

