# **PRACTICAL -7**

**Solution :** (S-DES encryption)

public class SDES {

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

static int[] l = new int[5];

static int[] r = new int[5];

static int round = 1;

static int[] plaintext = new int[8];

static String[] massage;

public static void PC1(int[] keymat) {

int[] pc1 = { 2, 4, 1, 6, 3, 9, 0, 8, 7, 5 };

int[] shifted = new int[10];

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

shifted[i] = keymat[pc1[i]];

}

for (int i = 0; i < 5; i++) {

l[i] = shifted[i];

r[i] = shifted[i + 5];

}

}

public static void shift() {

if (round == 1) {

int[] temp = { l[0], r[0] };

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

l[i] = l[i + 1];

r[i] = r[i + 1];

}

l[4] = temp[0];

r[4] = temp[1];

} else if (round == 2) {

int[] temp = { l[0], l[1], r[0], r[1] };

for (int i = 0; i < l.length - 2; i++) {

l[i] = l[i + 2];

r[i] = r[i + 2];

}

l[3] = temp[0];

l[4] = temp[1];

r[3] = temp[2];

r[4] = temp[3];

}

}

public static int[] PC8() {

shift();

int[] genratedKey = { r[0], l[2], r[1], l[3], r[2], l[4], r[4], r[3] };

round++;

return genratedKey;

}

public static void ptToBinaryWithPermutation1(String str) {

PC1(key);

round = 1;

int[] permute = { 1, 5, 2, 0, 3, 7, 4, 6 };

for (int i = 0; i < 8; i++) {

plaintext[i] = (int) str.charAt(permute[i]) - 48;

}

}

public static void generateBinaryFromPlainText(String pt) {

String[] binarypt = new String[pt.length()];

int j = 0;

for (int i : pt.toCharArray()) {

String temp = Integer.toBinaryString(i);

if (temp.length() == 7)

temp = "0" + temp;

else if (temp.length() == 6)

temp = "00" + temp;

binarypt[j] = temp;

j++;

}

massage = binarypt;

}

public static int[] functionImplementation(int[] a, int[] ls) {

// for S0 WE WILL USE TECHNIC LIKE HILL CIPHER AND FOR S1 REVERSE OF IT

int[] extended = { a[3], a[0], a[1], a[2], a[1], a[2], a[3], a[0] };

int[] tempkey = PC8();

String x, y;

for (int i = 0; i < tempkey.length; i++) {

extended[i] = extended[i] ^ tempkey[i];

}

x = Integer.toBinaryString((((2 \* (extended[0]) + extended[1]) + (2 \* extended[2] + extended[3])) % 4));

y = Integer.toBinaryString((((2 \* (extended[4]) + extended[5]) + (2 \* extended[6] + extended[7]) + 1) % 4));

if (x.length() < 2)

x = "0" + x;

if (y.length() < 2)

y = "0" + y;

char[] newtemp = (x + y).toCharArray();

int[] rs = { (int) newtemp[1] - 48, (int) newtemp[3] - 48, (int) newtemp[2] - 48, (int) newtemp[0] - 48 };

for (int j = 0; j < 4; j++) {

rs[j] = rs[j] ^ ls[j];

}

return rs;

}

public static String[] Encrypt(String plt) {

generateBinaryFromPlainText(plt);

String[] encrypt = new String[plt.length()];

for (int i = 0; i < massage.length; i++) {

int[] ls = new int[4];

int[] rs = new int[4];

encrypt[i] = "";

ptToBinaryWithPermutation1(massage[i]);

for (int j = 0; j < plaintext.length / 2; j++) {

ls[j] = plaintext[j];

rs[j] = plaintext[j + 4];

}

int k = 10;

long ans = 0;

rs = functionImplementation(rs, ls);

int[] temp = ls;

ls = rs;

rs = temp;

rs = functionImplementation(rs, ls);

int[] bits = { ls[3], ls[0], ls[2], rs[0], rs[2], ls[1], rs[3], rs[1] };

for (int j = 0; j < 8; j++) {

ans = (ans \* k) + bits[j];

}

for (int j = 0; j < encrypt.length - (ans + "").length(); j++) {

encrypt[i] = "0" + encrypt[i];

}

encrypt[i] = encrypt[i] + ans + "";

// System.out.println(ans);

}

return encrypt;

}

public static int convertBinaryToDecimal(long num) {

int decimalNumber = 0, i = 0;

long remainder;

while (num != 0) {

remainder = num % 10;

num /= 10;

decimalNumber += remainder \* Math.pow(2, i);

++i;

}

return decimalNumber;

}

public static void main(String[] args) {

String[] x = Encrypt("Abcd");

System.out.println("Plain Text -- Abcd");

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

for (int a : key) {

System.out.print(a + " ");

}

System.out.println("\nCipher Text -- \n");

for (int i = 0; i < x.length; i++) {

System.out.println(("Abcd".toCharArray())[i] + " --> " + x[i]);

}

}

}