|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 03** | **Simplified DES** | | |
| Date of Exercise | 20 – 01 - 2015 | Date of Output Verification | 03 – 02 - 2015 |

**Question**

For a message size of 8-bits and a key size of 10-bits, write a java program to execute a simplified DES algorithm for two rounds of iteration for given values:

Straight P box table = 3 5 2 7 4 10 1 9 8 6

Compression table P-box = 6 3 7 4 8 5 10 9

Straight P box permutation table = 2 4 3 1

Initial Permutation table = 2 6 3 1 4 8 5 7

Final permutation table = 4 1 3 5 7 2 8 6

Expansion P-box = 4 1 2 3 2 3 4 1

S0:

1 0 3 2

3 2 1 0

0 2 1 3

3 1 3 2

S1:

0 1 2 3

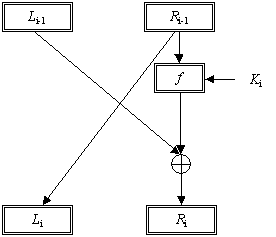
2 0 1 3

3 0 1 0

2 1 0 3

**Procedure**

The block of 12 bits is written in the form  *L*0*R*0, where *L*0 consists of the first 6 bits and *R*0 consists of the last 6 bits. The *i*th round of the algorithm transforms an input *L*i-1*R*i-1 to the output *L*i*R*i using an 8-bit *K*i derived from *K*.

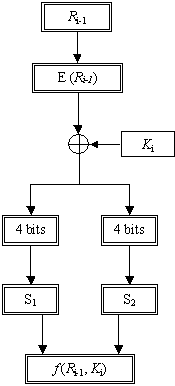
*One Round of a Feistel System*

The output for the *i*th round is found as follows.

*L*i = *R*i-1 and *R*i = *L*i-1 ⊕ *f* (*R*i-1, *K*i)

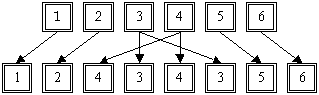
This operation is performed for a certain number of rounds, say *n*, and produces *L*n*R*n. The ciphertext will be *R*n*L*n. Encryption and decryption are done the same way except the keys are selected in the reverse order. The keys for encryption will be *K1*, *K*2 …… *K*n and for decryption will be *K*n, *K*n-1 …… *K1.*

**Function *f*(*R*i-1,*K*i)**: - The function *f*(*R*i-1,*K*i), depicted in the Figure below, is described in following steps.



*The Function* f (*R*i-1, *Ki*)

1. The 6-bits are expanded using the following expansion function. The expansion function takes 6-bit input and produces an 8-bit output. This output is the input for the two S-boxes



*The Expansion Function, E(Ri-1)*

1. The 8-bit output from the previous step is Exclusive-ORed with the key *K*i
2. The 8-bit output is divided into two blocks. The first block consists of the first 4 bits and the last four bits make the second block. The first block is the input for the first S-box (S1) and the second block is the input for the second S-box (S2).
3. The S-boxes take 4 bits as input and produce 3bits of output. The first bit of the input is used to select the row from the S-box, 0 for the first row and 1 for the second row. The last 3 bits are used to select the column.

Example: Let the output from the expander function be 11010010.

So 1101 will be the input for the S1 box and 0010 will be the input for the S2 box. The output from the S1 box will be 111, the first of the input is 1 so select the second row and 101 will select the 6th column. Similarly the output from the S2 box will be 110.

1. The output from the S-boxes is combined to form a single block of 6 bits. These 6 bits will be the output of the function *f*(*R*i-1,*K*i).

In our example we have the S1 output 111 and S2 output 110. So the output for the function *f*(*R*i-1,*K*i) will be 111110, the S1 output followed by the S2 output.

The S1 and S2 boxes are shown below.

**S1-Box**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 101 | 010 | 001 | 110 | 011 | 100 | 111 | 000 |
| 001 | 100 | 110 | 010 | 000 | 111 | 101 | 011 |

**S2-Box**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 000 | 110 | 101 | 111 | 001 | 011 | 010 |
| 101 | 011 | 000 | 111 | 110 | 010 | 001 | 100 |

**Program**

import java.util.Scanner;

public class NetworkSecurityLabExp3SimplifiedDES {

public static String iptext = "00101000", key = "1100011110", k1 = "", k2 = "", mantext = "", fkval = "", optext = "";

public static int s0[][] = {{1, 0, 3, 2}, {3, 2, 1, 0}, {0, 2, 1, 3}, {3, 1, 3, 2}}, s1[][] = {{0, 1, 2, 3}, {2, 0, 1, 3}, {3, 0, 1, 0}, {2, 1, 0, 3}};z

public static Scanner in = new Scanner(System.in);

public static void main(String[] args) {

boolean r = true;

int choice;

// System.out.println("Simplified DES - UR12CS135");

setkey();

do {

System.out.println("\n\n---------Main Menu---------\n1.Change Key \n2.Encode \n3.Decode \n4.Exit");

choice = in.nextInt();

in.nextLine();

switch (choice) {

case 1:

setkey();

break;

case 2:

encode();

break;

case 3:

decode();

break;

case 4:

r = false;

break;

default:

System.out.println("Invalid Option!, Try Again...");

break;

}

} while (r == true);

}

public static void setkey() {

System.out.println("\n--------Set Key--------");

System.out.println("Enter Key (Note: Key should be 10 bit)");

key = in.next();

genkey();

}

public static void encode() {

System.out.println("\n---------Encode--------\nEnter Plain Text");

iptext = in.next();

rollout(k1, k2);

System.out.println("Cipher Text: " + optext);

}

public static void decode() {

System.out.println("\n---------Decode--------\nEnter Cipher Text");

iptext = in.next();

rollout(k2, k1);

System.out.println("Plain Text: " + optext);

}

public static void rollout(String a, String b) {

System.out.println("\n---Manipulating----");

mantext = ip(iptext);

System.out.println(mantext);

fk(mantext, a);

sw();

fk(fkval, b);

optext = ipinverse(fkval);

}

public static void genkey() {

System.out.println("\n--Generating Keys--");

String p = "", l = "", r = "", t = "";

p = p10(key);

System.out.println(p);

l = p.substring(0, 5);

r = p.substring(5, 10);

System.out.println(l + " " + r);

l = leftshift(l);

r = leftshift(r);

t = l + r;

System.out.println(l + " " + r);

System.out.println(t);

k1 = p8(t);

System.out.println("Key 1: " + k1);

l = leftshift(l);

r = leftshift(r);

l = leftshift(l);

r = leftshift(r);

t = l + r;

k2 = p8(t);

System.out.println("Key 2: " + k2);

}

public static void fk(String p, String a) {

String l = "", r = "", la = "", ra = "", t = "", row = "", col = "";

int rowval, colval, lval, rval;

la = p.substring(0, 4);

ra = p.substring(4, 8);

t = ep(ra);

System.out.println(t);

t = xor(t, a);

System.out.println(t);

l = t.substring(0, 4);

r = t.substring(4, 8);

row = l.charAt(0) + "" + l.charAt(3);

col = l.charAt(1) + "" + l.charAt(2);

lval = s0[getdecimalval(row)][getdecimalval(col)];

l = getbinaryval(lval);

row = r.charAt(0) + "" + r.charAt(3);

col = r.charAt(1) + "" + r.charAt(2);

rval = s1[getdecimalval(row)][getdecimalval(col)];

r = getbinaryval(rval);

t = l + "" + r;

System.out.println(t);

t = p4(t);

System.out.println(t);

la = xor(t, la);

fkval = la + "" + ra;

System.out.println(fkval);

}

public static void sw() {

System.out.println("\*Swap");

fkval = fkval.substring(4, 8) + "" + fkval.substring(0, 4);

}

public static String ep(String a) {

System.out.println("\*Expansion Permutation");

String b = "";

b = a.charAt(3) + "" + a.charAt(0) + "" + a.charAt(1) + "" + a.charAt(2) + "" + a.charAt(1) + "" + a.charAt(2) + "" + a.charAt(3) + "" + a.charAt(0);

return b;

}

public static String p10(String a) {

System.out.println("\*p10");

String b = "";

b = a.charAt(2) + "" + a.charAt(4) + "" + a.charAt(1) + "" + a.charAt(7) + "" + a.charAt(3) + "" + a.charAt(9) + "" + a.charAt(0) + "" + a.charAt(8) + "" + a.charAt(7) + "" + a.charAt(5);

return b;

}

public static String p8(String a) {

System.out.println("\*p8");

String b = "";

b = a.charAt(5) + "" + a.charAt(2) + "" + a.charAt(6) + "" + a.charAt(3) + "" + a.charAt(7) + "" + a.charAt(4) + "" + a.charAt(9) + "" + a.charAt(8);

return b;

}

public static String p4(String a) {

String b = "";

System.out.println("\*p4");

b = a.charAt(1) + "" + a.charAt(3) + "" + a.charAt(2) + "" + a.charAt(0);

return b;

}

public static String ip(String a) {

System.out.println("\*Initial Permutation");

String b = "";

b = a.charAt(1) + "" + a.charAt(5) + "" + a.charAt(2) + "" + a.charAt(0) + "" + a.charAt(3) + "" + a.charAt(7) + "" + a.charAt(4) + "" + a.charAt(6);

return b;

}

public static String ipinverse(String a) {

System.out.println("\*Inverse IP");

String b = "";

b = a.charAt(3) + "" + a.charAt(0) + "" + a.charAt(2) + "" + a.charAt(4) + "" + a.charAt(6) + "" + a.charAt(1) + "" + a.charAt(7) + "" + a.charAt(5);

return b;

}

public static String leftshift(String a) {

String val = "";

for (int i = 1; i < a.length(); i++) {

val += a.charAt(i) + "";

}

val += a.charAt(0) + "";

return val;

}

public static String xor(String a, String b) {

String val = "";

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

val += (a.charAt(i) ^ b.charAt(i)) + "";

}

return val;

}

public static int getdecimalval(String a) {

int i = 0;

if (a.charAt(0) == '0' && a.charAt(1) == '0') {

i = 0;

} else if (a.charAt(0) == '0' && a.charAt(1) == '1') {

i = 1;

} else if (a.charAt(0) == '1' && a.charAt(1) == '0') {

i = 2;

} else if (a.charAt(0) == '1' && a.charAt(1) == '1') {

i = 3;

}

return i;

}

public static String getbinaryval(int a) {

String i = "";

if (a == 0) {

i = '0' + "" + '0';

} else if (a == 1) {

i = '0' + "" + '1';

} else if (a == 2) {

i = '1' + "" + '0';

} else if (a == 3) {

i = '1' + "" + '1';

}

return i;

}

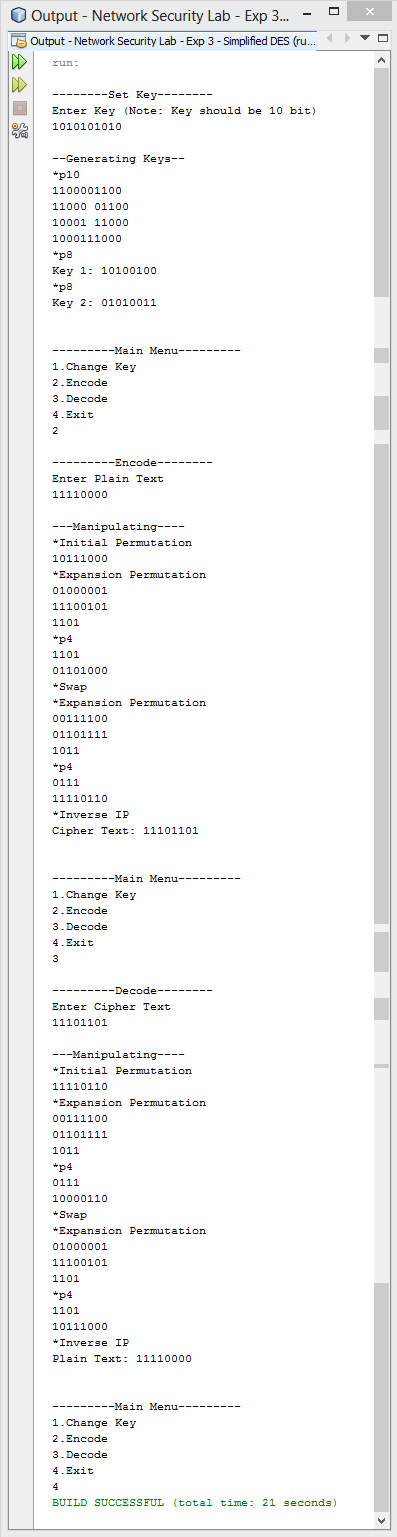
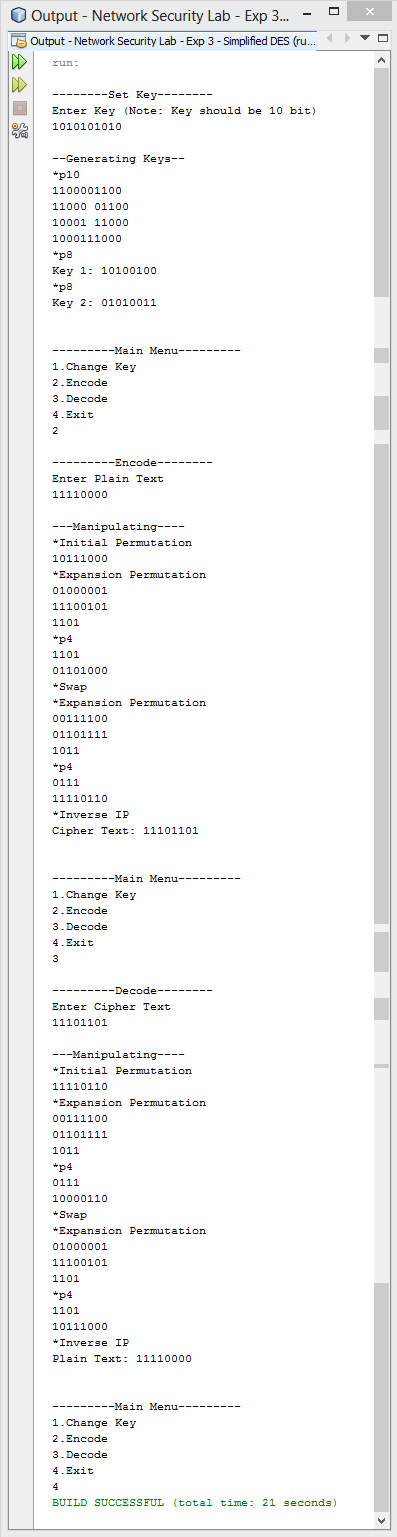
}

**Input**

Key: 1010101010

Plain text: 11110000

**Output**



**Result**

Simplified DES is successfully implementation in java.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Manoj Kumar

Date: 03 –02 - 2015