**Lab Practical #10:**

1. Implementation of Block Cipher mode of Operation - Output Feedback mode (OFM).

**Program:**

import java.util.Scanner;

public class OFM {

    public static void main(String[] args) {

        String cipherTextString = "";

        String decTextString = "";

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Plain Text (Length : 8) :");

        String plainTexString = sc.nextLine();

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

        String keyString = sc.nextLine();

        int initalVector = 2536;

        char[] pt = plainTexString.toCharArray();

        //Encryption ::

        for (int i = 0; i < pt.length; i=i+4) {

            String plainTextSubString = plainTexString.substring(i, i+4);

            if(i==0){

                int xOROfInitialVectorWithKey = (initalVector) ^ Integer.parseInt(keyString);

               initalVector = xOROfInitialVectorWithKey;

                cipherTextString += Integer.parseInt(plainTextSubString) ^ xOROfInitialVectorWithKey;

            }

            else{

                int xOROfInitialVectorWithKey =(initalVector) ^ Integer.parseInt(keyString);

                initalVector = xOROfInitialVectorWithKey;

                cipherTextString += Integer.parseInt(plainTextSubString) ^ xOROfInitialVectorWithKey;

            }

        }

        System.out.println("Encrypted String :: "+cipherTextString);

        char[] ctArray  = cipherTextString.toCharArray();

        //Decryption ::

        for (int i = 0; i < ctArray.length; i=i+4) {

            String ciphetTextSubString = cipherTextString.substring(i, i+4);

            if(i==0){

                int xOROfInitialVectorWithKey = (initalVector) ^ Integer.parseInt(keyString);

                initalVector = xOROfInitialVectorWithKey;

                decTextString += Integer.parseInt(ciphetTextSubString) ^ xOROfInitialVectorWithKey;

            }

            else{

                int xOROfInitialVectorWithKey =(initalVector) ^ Integer.parseInt(keyString);

                initalVector = xOROfInitialVectorWithKey;

                decTextString += Integer.parseInt(ciphetTextSubString) ^ xOROfInitialVectorWithKey;

            }

        }

        System.out.println("Decrypted String :: "+decTextString);

    }

}

            if(i==0){

                int xOROfInitialVectorWithKey = (initalVector) ^ Integer.parseInt(keyString);

                initalVector = xOROfInitialVectorWithKey;

                decTextString += Integer.parseInt(ciphetTextSubString) ^ xOROfInitialVectorWithKey;

            }

            else{

                int xOROfInitialVectorWithKey =(initalVector) ^ Integer.parseInt(keyString);

                initalVector = xOROfInitialVectorWithKey;

                decTextString += Integer.parseInt(ciphetTextSubString) ^ xOROfInitialVectorWithKey;

            }

        }

        System.out.println("Decrypted String :: "+decTextString);

    }

}

**Output:**

Enter Plain Text (Length : 8) :

12345678

Enter Key :

1234

Encrypted String :: 25368134

Decrypted String :: 12345678