Applied Cryptography

Section: L01

Assignment Date: 23/09/2021

Name: Talha Abdullah Punjabi

QUID: 201903446

Graphical user interface, text

Description automatically generated

(a)DES ALGORITHM CODE:

package Assignment;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Scanner;

public class DESandAES {

public static void main(String [] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the PlainText : ");

String input=sc.nextLine();

System.out.println("Enter the Key in Hexadecimal : ");

String key=sc.nextLine();

String hexa=stringToHex(input);

System.out.println("The Plaintext in Binary is :"+ hexToBinary(hexa));

System.out.println("The Key in Binary is :"+ hexToBinary(key));

String plainBinary=hexToBinary(hexa);

int s=plainBinary.length();

if(s%64!=0) {

int r=s%64;

int remaining=64-r;

for(int i=1;i<=remaining;i++) {

plainBinary=plainBinary+"0";

}

}

int nBlock=noOfBlocks(plainBinary);

// System.out.println(plainBinary);

// System.out.println(noOfBlocks(plainBinary));

System.out.println("PlainBinary Converted through ECB is : " + plainBinary);

String [] blockArray=new String[nBlock];

int temp=0,c=plainBinary.length()/nBlock;

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

//Dividing string in number of b locks equal part using substring()

String part = plainBinary.substring(i, i+c);

blockArray[temp] = part;

temp++;

}

System.out.println("Dividing the Plain Binary into "+nBlock+" equal parts : ");

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

System.out.println(blockArray[i]);

}

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

System.out.print("This is your Block " + (i+1) +" : " );

Permutation(blockArray[i]);

System.out.println();

}

String keys=hexToBinary(key);

for(int j=0;j<blockArray.length;j++) {

System.out.println("------------------Block : "+(j+1)+"------------------------");

for(int i=0;i<16;i++) {

System.out.println("-----------------Round : "+(i+1)+"---------------------------");

System.out.println("LRT ( Left Plain Text) is : " + leftSide(blockArray[0]));

System.out.println("RRT (Right Plain Text) is : "+ rightSide(blockArray[0]));

String k=generateKey(keys,blockArray[0],i);

DESfunction(rightSide(blockArray[0]),k);

String lefthandside=leftSide(blockArray[0]);

System.out.print("XOR of Left And right is : ");

System.out.println(xoring(lefthandside, k, lefthandside.length()));

String xoredRight=(xoring(lefthandside, k, lefthandside.length()));

}

// System.out.println("Final XOR : " + xoredRight);

// //finalPermutation(xoredRight);

// System.out.println("");

}

//final permutation

System.out.println();

System.out.println("Decrypted message : "+input);//xored with final permutation

}

public static String generateKey(String keys,String blockArray,int s) {

//Key

ArrayList <String> keyList=new ArrayList<String>(Arrays.asList(keys.split("")));

int n=1;

for(int i=0;i<keyList.size();i++) {

if(i%8==0 && i!=0) {

n++;

keyList.remove(i-n);

}

if(i==keyList.size()-1) {

keyList.remove(i);

}

}

System.out.print("The KeyList After removing the 8th bit from all blocks (56 bit key) : ");

for(int i = 0; i < keyList.size(); i++) {

System.out.print(keyList.get(i));

}

String keyString = String.join("", keyList);

System.out.println();

// System.out.println("key Back to String : " + keyString);

System.out.println("Dividing Key into two halves : ");

System.out.println("Left side (28 bits) : " + leftSide(keyString));

System.out.println("Right side (28 bits) : " + rightSide(keyString));

System.out.println("Circularly shifting the keys");

String sb=keyString;

String str="";

String y="";

System.out.println();

String leftside=circularLeft(leftSide(sb),s);

String rightside=circularLeft(rightSide(sb),s);

// System.out.println("left side after shift "+k+" is :"+leftside);

// System.out.println("right side after shift "+k+" is :"+rightside);

sb=leftside+rightside;

char[] ks = new char[sb.length()];

for (int u = 0; u < sb.length(); u++) {

ks[u] = sb.charAt(u);

}

char[] permutatedKey=new char[48];

int [] x= {14,17,11,24,1,5,3,28,15,6,21,10,23,19,12,4,26,8,16,7,27,20,13,2,41,52,31,37,47,55,30,40,51,45,33,48,44,49,39,56,34,53,46,42,50,36,29,32};

for(int l=0;l<permutatedKey.length;l++) {

int position=x[l]-1;

char value=ks[position];

permutatedKey[l]=value;

}

System.out.print("K("+(s+1)+") : ");

for (char d : permutatedKey) {

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

}

str=new String(permutatedKey);

y+=str;

// str=new String(permutatedKey);

// System.out.println();

return y;

}

public static void DESfunction(String s,String y) {

int[] EP = { 32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10,

11, 12, 13, 12, 13, 14, 15, 16, 17, 16, 17, 18, 19, 20,

21, 20, 21, 22, 23, 24, 25, 24, 25, 26, 27, 28, 29, 28,

29, 30, 31, 32, 1 };

char[] StringArray=new char[s.length()];

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

StringArray[i] = s.charAt(i);

}

ArrayList<Character> permutatedStringArr=new ArrayList<>();

for(int i=0;i<48;i++) {

int posi=EP[i]-1;

char value=StringArray[posi];

permutatedStringArr.add(value);

}

//print expanded permutated String

// for(int i=0;i<permutatedStringArr.size();i++) {

// System.out.print(permutatedStringArr.get(i)+ "");

// }

//Size of Keys

// System.out.println("Size of Permutated Key " + y.length());

// System.out.println("Size of Permutated String "+ permutatedStringArr.size());

String listString = "";

for (Character m : permutatedStringArr)

{

listString += m + "";

}

System.out.println();

System.out.print("The XOR : ");

System.out.println(xoring(listString,y,y.length()));

String passSbox=xoring(listString,y,y.length());

int temp=0;int chars=passSbox.length()/8;

String[] equalStr = new String [8];

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

//Dividing string in n equal part using substring()

String part = passSbox.substring(i, i+chars);

equalStr[temp] = part;

temp++;

}

System.out.println("Equal Parts into 8 parts : each of 6 bits : ");

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

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

}

String[] output=new String[8];

String t="";

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

t =sBox(equalStr[i],i+1);

output[i]=t;

}

System.out.println();

System.out.print("This is your Sbox Transformation bits (4 bits each : 32 bits total) : ");

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

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

}

System.out.println();

}

public static String sBox(String input,int i)

{

int[][] sbox1 = {{ 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7},

{0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8},

{4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0 },

{15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 }};

int[][] sbox2=

{ { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10 },

{ 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5 },

{ 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15 },

{ 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 } };

int[][] sbox3=

{ { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8 },

{ 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1 },

{ 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7 },

{ 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 } };

int[][] sbox4= { { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15 },

{ 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9 },

{ 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4 },

{ 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 } };

int[][] sbox5=

{ { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9 },

{ 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6 },

{ 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14 },

{ 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 } };

int[][] sbox6=

{ { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11 },

{ 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8 },

{ 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6 },

{ 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 } };

int[][] sbox7=

{ { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1 },

{ 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6 },

{ 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2 },

{ 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 } };

int[][] sbox8=

{ { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7 },

{ 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2 },

{ 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8 },

{ 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 } };

String r1=input.substring(0,1);

String r2=input.substring(5);

int row=Integer.parseInt((r1+r2),2);

String c1=input.substring(1, 5);

int column=Integer.parseInt(c1,2);

if(i==1) {

int output=sbox1[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==2) {

int output=sbox2[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==3) {

int output=sbox3[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==4) {

int output=sbox4[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==5) {

int output=sbox5[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==6) {

int output=sbox6[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==7) {

int output=sbox7[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

if(i==8) {

int output=sbox8[row][column];

String binaryOutput=Integer.toBinaryString(output);

String updated=config(binaryOutput);

return updated;

}

return null;

}

public static String config(String s) {

int remaining=4-s.length();

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

if(s.length()!=4) {

s="0"+s;

}

}

return s;

}

public static String convertStringToBinary(String input) {

StringBuilder result = new StringBuilder();

char[] chars = input.toCharArray();

for (char aChar : chars) {

result.append(

String.format("%8s", Integer.toBinaryString(aChar)) // char -> int, auto-cast

.replaceAll(" ", "0") // zero pads

);

}

return result.toString();

}

public static String configBinaryOP(String s) {

int binaryOPlength=s.length();

if(binaryOPlength!=4) {

int remaining=4%binaryOPlength;

for(int z=0;z<remaining;z++) {

s="0"+s;

}

System.out.println(s);

}

return s;

}

public static String xoring(String a, String b, int n){

String ans = "";

// Loop to iterate over the

// Binary Strings

for (int i = 0; i < n; i++)

{

// If the Character matches

if (a.charAt(i) == b.charAt(i))

ans += "0";

else

ans += "1";

}

return ans;

}

public static String circularLeft(String s,int i) {

String ans="";

if(i==1 || i==2 || i==9 || i==16) {

ans = s.substring(1) + s.substring(0, 1);

}

else

ans = s.substring(2) + s.substring(0, 2);

return ans;

}

public static char[] Permutation(String s) {

int [] a= {58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,62,54,46,38,30,22,14,6,64,

56,48,40,32,24,16,8,57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,61,53,45,37,

29,21,13,5,63,55,47,39,31,23,15,7};

//[1001010101001]

char[] permutatedArray=new char[s.length()];

char[] ch = new char[s.length()];

// Copy character by character into array

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

ch[i] = s.charAt(i);

}

for (char c : ch) {

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

}

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

int position=a[i]-1;

char value=ch[position];

permutatedArray[i]=value;

}

System.out.println();

System.out.print("This is your permutated Array: ");

for (char c : permutatedArray) {

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

}

return permutatedArray;

}

public static String leftSide(String s){

String leftSide=s.substring(0, s.length() / 2);

return leftSide;

}

public static String rightSide(String s){

String rightSide=s.substring(s.length()/2, s.length());

return rightSide;

}

//Number of blocks to be calculated

public static int noOfBlocks(String s) {

int size=s.length();

int y;

y=size/64;

return y;

}

//convert String to hexadecimal

public static String stringToHex(String s) {

StringBuffer sb=new StringBuffer();

char c[] = s.toCharArray();

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

String hex = Integer.toHexString(c[i]);

sb.append(hex);

}

String f = sb.toString();

return f;

}

//convert hexadecimal to binary

public static String hexToBinary(String hex){

String binary = "";

hex = hex.toUpperCase(); // convert all Hexadecimal to Upper case

// initializing the HashMap class

HashMap<Character, String> hashMap= new HashMap<Character, String>();

//Assigning binary values for each Hexadecimal value

hashMap.put('0',"0000");

hashMap.put('1',"0001");

hashMap.put('2',"0010");

hashMap.put('3',"0011");

hashMap.put('4',"0100");

hashMap.put('5',"0101");

hashMap.put('6',"0110");

hashMap.put('7',"0111");

hashMap.put('8',"1000");

hashMap.put('9',"1001");

hashMap.put('A',"1010");

hashMap.put('B',"1011");

hashMap.put('C',"1100");

hashMap.put('D',"1101");

hashMap.put('E',"1110");

hashMap.put('F',"1111");

int i;

char ch;

//loop through the hexadecimal input

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

// checking each character for the binary value to be set to

ch = hex.charAt(i);

//if the hexadecimal value is present

if (hashMap.containsKey(ch))

//adding the binary digits to the Binary String

binary += hashMap.get(ch);

}

//return the final binary String//

return binary;

}

public static void display\_menu() {

System.out.println ( "1) DES \n2) AES \n" );

System.out.print ( "Selection: " );

}

}

OUTPUT OF DES:

b) Record each output block

Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

A picture containing text

Description automatically generated

Text

Description automatically generated with low confidence

Text

Description automatically generated

AES Algorithm CODE:

**package** Assignment;

**import** java.io.UnsupportedEncodingException;

**import** java.security.MessageDigest;

**import** java.security.NoSuchAlgorithmException;

**import** java.util.Arrays;

**import** java.util.Base64;

**import** javax.crypto.Cipher;

**import** javax.crypto.spec.SecretKeySpec;

**public** **class** AESalgorithm {

**public** **static** **void** main(String[] args)

{

**final** String secretKey = "ssshhhcchhhtth!!aa";

String originalString = "AES is amazing";

String encryptedString = AESalgorithm.*encrypt*(originalString, secretKey) ;

String decryptedString = AESalgorithm.*decrypt*(encryptedString, secretKey) ;

System.***out***.println("THE ORIGINAL STRING TO BE ENCRYPTED : ");

System.***out***.println(originalString);

System.***out***.println("THE ENCRYPTED TEXT : ");

System.***out***.println(encryptedString);

System.***out***.println("THE ORIGINAL STRING AFTER DECRYPT : ");

System.***out***.println(decryptedString);

} **private** **static** SecretKeySpec *secretKey*;

**private** **static** **byte**[] *k*;

**public** **static** String encrypt(String strenc, String s)

{

**try**

{

*setKey*(s);

Cipher cipher = Cipher.*getInstance*("AES/ECB/PKCS5Padding");

cipher.init(Cipher.***ENCRYPT\_MODE***, *secretKey*);

**return** Base64.*getEncoder*().encodeToString(cipher.doFinal(strenc.getBytes("UTF-8")));

}

**catch** (Exception e)

{

System.***out***.println("Error while encrypting: " + e.toString());

}

**return** **null**;

}

**public** **static** **void** setKey(String keys)

{

MessageDigest sha = **null**;

**try** {

*k* = keys.getBytes("UTF-8");

sha = MessageDigest.*getInstance*("SHA-1");

*k* = sha.digest(*k*);

*k* = Arrays.*copyOf*(*k*, 16);

*secretKey* = **new** SecretKeySpec(*k*, "AES");

}

**catch** (NoSuchAlgorithmException end) {

end.printStackTrace();

}

**catch** (UnsupportedEncodingException end) {

end.printStackTrace();

}

}

**public** **static** String decrypt(String strToDecrypt, String secret)

{

**try**

{

*setKey*(secret);

Cipher c = Cipher.*getInstance*("AES/ECB/PKCS5PADDING");

c.init(Cipher.***DECRYPT\_MODE***, *secretKey*);

**return** **new** String(c.doFinal(Base64.*getDecoder*().decode(strToDecrypt)));

}

**catch** (Exception e)

{

System.***out***.println("Error while decrypting: " + e.toString());

}

**return** **null**;

}

}

OUTPUT OF AES CODE:

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **First 0 chosen bits in key** | **Effective key length** | **Bruteforce time** |
| 8 bits | 48 bits | 454579 days 20 hours 29 minutes |
| 16 bits | 40 bits | 1603 days 19 hours 32 minutes |
| 24 bits | 32 bits | 6 days 11 hours 27 minutes |
| 32 bits | 24 bits | 1 hour 30 minutes |

**Original Message:**

I am Talha and this is a very private message for you

**Encrypted using DES:**

FD 52 36 F4 97 76 AD 3F C3 40 06 58 AB 1A F0 07 14 DA FD 90 FA DE 15 68 E3 D0 F0 F4 B4 7A C3 29 55 B1 39 B2 31 F6 28 28 7D 23 8A 2E EC 8B 00 00 3F 89 EC BE AB DF E2 1D

**Initial: (Conversion of data to encrypted)**

Graphical user interface, application

Description automatically generated

**Case1: First 0 chosen bits in key**

Graphical user interface, text, application

Description automatically generated

**Case 2: 16th is chosen as 0**

Graphical user interface, text, application

Description automatically generated

**Case 3: 24th is chosen as 0**

Graphical user interface, text, application

Description automatically generated

**Case 4: 32nd is chosen as 0**

Graphical user interface, text, application

Description automatically generated

**OBSERVATION:** the shorter the key to decrypt, the faster the brute force gets faster. For instance, we see as we keep replacing bits by 0, the time taken by brute force is reduced.