介绍

1. SAES算法的背景和概述

S-AES 加密算法使用一个16位明文分组和一个16位密钥作为输人生成一个16位密文分组作为输出。S-AES解密算法用一个16位密文分组和相同的密钥作为输入，生成原始的16位明文分组作为输出。加密算法使用4个不同的函数或变换:密钥加（Ax)、半字节代替(NS)、行移位（SR)和列混淆(MC)。解密就是加密的逆，其中的三个函数都有一个对应的逆函数:逆半字节代替(INS)、逆行位移(ISR)和逆列混淆(IMC)

算法设计

1. SAES算法的设计原则和目标

完成加密和解密的算法过程，本小组使用的是java语言，并完成五关的过关测试，分别为基本测试、交叉测试、扩展功能、多重加密和工作模式。设计过程中要保持代码的规范，具体为变量命名规范，代码的注释量和函数式的编码。

2.算法细节

A.轮函数的设计

public static int addRoundKey(int data, int roundKey) {

return data ^ roundKey;

}//实现异或操作

1. 密钥扩展算法

public static int[] keyExpansion(int key) {

int[] roundKeys = new int[4]; // 包含6个轮的轮密钥

int w0 = (key >>> 8) & 0xFF; // 第一个8位字

int w1 = key & 0xFF; // 第二个8位字

// 轮常数

int RCON1 = 0x80; // RCON(1)

// 计算 w2

int w2 = w1 ^ (RCON1 ^ SubNib(RotNib(w1)));

// 计算 w3

int w3 = w2 ^ w1;

// 存储轮密钥

roundKeys[0] = w0;

roundKeys[1] = w1;

roundKeys[2] = w2;

roundKeys[3] = w3;

return roundKeys;

}

public static int SubNib(int x1) {

String i1 = Integer.toBinaryString(x1 & 0xFF);

while (i1.length() < 8) {

i1 = "0" + i1;

}

String p1 = i1.substring(0, 2);

String p2 = i1.substring(2, 4);

int z1 = sBox[Integer.parseInt(p1, 2)][Integer.parseInt(p2, 2)];

String p3 = i1.substring(4, 6);

String p4 = i1.substring(6, 8);

int z2 = sBox[Integer.parseInt(p3, 2)][Integer.parseInt(p4, 2)];

x1 = (int)((z1 << 4) | z2);

// 实现字节替代操作（SubNib）

// 请根据S盒或其他字节替代表进行替代

return x1;

}

1. S盒和逆S盒的设计

static int sBox[][] = { {9,4,10,11},{13,1,8,5},{6,2,0,3},{12,14,15,7} }; // 16字节的S盒

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

public static int subBytes(int data) {

String x1 = Integer.toBinaryString(data & 0xFFFF);

while (x1.length() < 16) {

x1 = "0" + x1;

}

String y1 = x1.substring(0, 2);

String y2 = x1.substring(2, 4);

int z1 = sBox[Integer.parseInt(y1, 2)][Integer.parseInt(y2, 2)];

String y3 = x1.substring(4, 6);

String y4 = x1.substring(6, 8);

int z2 = sBox[Integer.parseInt(y3, 2)][Integer.parseInt(y4, 2)];

String y5 = x1.substring(8, 10);

String y6 = x1.substring(10, 12);

int z3 = sBox[Integer.parseInt(y5, 2)][Integer.parseInt(y6, 2)];

String y7 = x1.substring(12, 14);

String y8 = x1.substring(14, 16);

int z4 = sBox[Integer.parseInt(y7, 2)][Integer.parseInt(y8, 2)];

data = (int)((z1 << 12) | (z2 << 8) | (z3 << 4) | z4);

// 使用S盒执行字节替代

return data;//返回的为整数类型

}

public static int inverseSubBytes(int data) {

String ix1 = Integer.toBinaryString(data & 0xFFFF);

while (ix1.length() < 16) {

ix1 = "0" + ix1;

}

String iy1 = ix1.substring(0, 2);

String iy2 = ix1.substring(2, 4);

int iz1 = nsBox[Integer.parseInt(iy1, 2)][Integer.parseInt(iy2, 2)];

String iy3 = ix1.substring(4, 6);

String iy4 = ix1.substring(6, 8);

int iz2 = nsBox[Integer.parseInt(iy3, 2)][Integer.parseInt(iy4, 2)];

String iy5 = ix1.substring(8, 10);

String iy6 = ix1.substring(10, 12);

int iz3 = nsBox[Integer.parseInt(iy5, 2)][Integer.parseInt(iy6, 2)];

String iy7 = ix1.substring(12, 14);

String iy8 = ix1.substring(14, 16);

int iz4 = nsBox[Integer.parseInt(iy7, 2)][Integer.parseInt(iy8, 2)];

data = (int)((iz1 << 12) | (iz2 << 8) | (iz3 << 4) | iz4);

return data;//返回的为整数类型

// 使用逆S盒执行字节替代

}

1. 字循环移位操作

public static int RotNib(int data) {

// 实现字循环移位操作（RotNib）

int rotated = ((data << 4) & 0xFF) | ((data >>> 4) & 0x0F);

return rotated;

}

1. 列混淆操作

public static int mixColumns(int data) {

// 执行列混淆操作

int[] columnData = new int[4];

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

columnData[i] = (data >> (i \* 4)) & 0x0F;

}

int[] mixedColumnData = new int[4];

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

mixedColumnData[i] = mixColumn(columnData, i);

}

int resultData = 0;

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

resultData |= (mixedColumnData[i] << (i \* 4));

}

return (int) resultData;

}

private static int mixColumn(int[] columnData, int columnIndex) {

int result = 0;

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

result ^= multiplyGF2\_4(mixBox[i], columnData[(columnIndex + i) % 4]);

}

return result;

}

private static int multiplyGF2\_4(int a, int b) {

int result = 0;

while (b > 0) {

if ((b & 1) == 1) {

result ^= a;

}

a <<= 1;

if ((a & 0x10) == 0x10) {

a ^= 0x13;

}

b >>= 1;

}

return result;

}

1. 逆列混淆操作

public static int inverseMixColumns(int data) {

// 执行逆列混淆操作

int[] columnData = new int[4];

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

columnData[i] = (data >> (i \* 4)) & 0x0F;

}

int[] inverseMixedColumnData = new int[4];

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

inverseMixedColumnData[i] = inverseMixColumn(columnData, i);

}

int resultData = 0;

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

resultData |= (inverseMixedColumnData[i] << (i \* 4));

}

return (int) resultData;

}

private static int inverseMixColumn(int[] columnData, int columnIndex) {

int result = 0;

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

result ^= multiplyGF2\_4(nmixBox[i], columnData[(columnIndex + i) % 4]);

}

return result;

}

1. 行移位操作和逆行移位操作

public static int shiftRows(int data) {

int rowValue = ((data & 0xFF00) >> 8) | ((data & 0x00FF) << 8); // 行移位操作

return (int) rowValue;

// 执行行移位操作

}

public static int inverseShiftRows(int data) {

// 执行逆行移位操作

int rowValue = ((data & 0x00FF) << 8) | ((data & 0xFF00) >> 8); // 逆行移位操作

return (int) rowValue;

}

1. 加密流程

 public void actionPerformed(ActionEvent e) {  
               // 实现A-DES加密的代码  
   if (input3.getText().trim().equals("")){  
      message3.setText("<html><font color='red'>请输入明文!</font> <html>");  
  
   }

else if (keyfield3.getText().trim().equals(""))

{  
   message3.setText("<html><font color='red'>请输入密钥1!</font> <html>");}  
  else if (keyfield33.getText().trim().equals("")) {  
      message3.setText("<html><font color='red'>请输入密钥2!</font> <html>");}

else {  
      String plaintext = input3.getText();  
      String key1 = keyfield3.getText();  
      String key2=keyfield33.getText();  
      String key3=keyfield333.getText();  
if(keyfield333.getText().trim().equals("")) {  
     String ciphertext = SAES.tripleEncrypt(plaintext, key1, key2);  
      encry\_output3.setText(ciphertext);  
                   }

else if (keyfield333.getText().length()==16)

{  
  String ciphertext=SAES.tripleEncrypt(plaintext,key1,key2,key3);  
  encry\_output3.setText( ciphertext);  
                   }  
 else {  
     message3.setText("<html><font color='red'>请输入正确的密钥3!</font> <html>");  
                   }  
               }  
           }

1. 三重加密的实现

public void actionPerformed(ActionEvent e) {

//三重加密的SAES分为两种

String ciphertext = input3.getText();

String key1 = keyfield3.getText();

String key2=keyfield33.getText();

String key3=keyfield333.getText();

if(key3.trim().equals("")) {

String plaintext = SAES.tripleDecode(ciphertext, key1, key2);

decry\_output3.setText( plaintext );

}

else {

String plaintext = SAES.tripleDecode(ciphertext, key1, key2,key3);

decry\_output3.setText(plaintext );

}

}

});

}

J.相遇攻击

public static void main(String[] args) {

tripleEncryption ui = new tripleEncryption();

}

k.ui设计

public class tripleEncryption extends JFrame {

JLabel lb13 = new JLabel();

JLabel lb23 = new JLabel();

JLabel lb33 = new JLabel();

JLabel lb43 = new JLabel();

JLabel lb53 = new JLabel("密钥2");

JLabel lb533 = new JLabel("密钥3");

JLabel message3 = new JLabel();

JTextField input3 = new JTextField();

JTextField keyfield3 = new JTextField();

JTextField keyfield33 = new JTextField();

JTextField keyfield333 = new JTextField();

JTextField encry\_output3 = new JTextField();

JTextField decry\_output3 = new JTextField();

JButton encry\_btn3 = new JButton("Encrypt");

JButton decry\_btn3 = new JButton("Decrypt");

JSeparator line13 = new JSeparator(JSeparator.HORIZONTAL);

public tripleEncryption() {

setVisible(true);

maininit();

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

}

public void maininit() {

this.setTitle("S-AES");

this.setSize(800, 600);

// 设置常驻屏幕中央

Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();

int screenWidth = (int) screen.getWidth();

int screenHeight = (int) screen.getHeight();

int frameWidth = this.getWidth();

int frameHeight = this.getHeight();

int x = (screenWidth - frameWidth) / 2;

int y = (screenHeight - frameHeight) / 2;

this.setLocation(x, y);

getContentPane().setBackground(new Color(173, 216, 230));

this.setLayout(null); // 使用绝对布局

encry\_btn3.setBackground(Color.blue);

encry\_btn3.setForeground(Color.white);

decry\_btn3.setBackground(Color.red);

decry\_btn3.setForeground(Color.white);

lb13.setText("Enter the plaintext or the ciphertext:");

lb23.setText("Enter binary key 1:");

lb53.setText("Enter binary key 2:");

lb533.setText("Enter binary key 3:");

lb33.setText("Encryption result:");

lb43.setText("Decryption result:");

// 组件布局

lb13.setBounds(50, 40, 250, 50);

lb23.setBounds(50, 100, 200, 50);

lb53.setBounds(50, 150, 200, 50);

lb533.setBounds(50, 200, 200, 50);

lb33.setBounds(50, 350, 100, 50);

lb43.setBounds(50, 400, 100, 50);

message3.setBounds(50, 480, 100, 50);

input3.setBounds(300, 40, 250, 40);

keyfield3.setBounds(300, 100, 250, 40);

keyfield33.setBounds(300, 150, 250, 40);

keyfield333.setBounds(300, 200, 250, 40);

encry\_output3.setBounds(300, 350, 250, 40);

decry\_output3.setBounds(300, 400, 250, 40);

encry\_btn3.setBounds(50, 270, 140, 30);

decry\_btn3.setBounds(200, 270, 140, 30);

line13.setBounds(0, 310, 800, 2);

// 组件添加到 JFrame

this.add(lb13);

this.add(lb23);

this.add(lb33);

this.add(lb43);

this.add(lb53);

this.add(lb533);

this.add(message3);

this.add(input3);

this.add(keyfield3);

this.add(keyfield33);

this.add(keyfield333);

this.add(encry\_output3);

this.add(decry\_output3);

this.add(decry\_btn3);

this.add(encry\_btn3);

1. CRC

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class mainui extends JFrame {

JLabel lb14 = new JLabel();

JLabel lb24 = new JLabel();

JLabel lb34 = new JLabel();

JLabel lb44 = new JLabel();

JLabel lb54 = new JLabel();

JLabel message4 = new JLabel();

JTextField input4 = new JTextField();

JTextField keyfield4 = new JTextField();

JTextField keyfield44 = new JTextField();

JTextField encry\_output4 = new JTextField();

JTextField decry\_output4 = new JTextField();

JButton encry\_btn4 = new JButton("加密");

JButton decry\_btn4 = new JButton("解密");

public mainui() {

setVisible(true);

mainInit();

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

}

public void mainInit() {

this.setTitle("S-AES");

this.setSize(800, 600);

// Set the frame to be centered on the screen

Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();

int screenWidth = (int) screen.getWidth();

int screenHeight = (int) screen.getHeight();

int frameWidth = this.getWidth();

int frameHeight = this.getHeight();

int x = (screenWidth - frameWidth) / 2;

int y = (screenHeight - frameHeight) / 2;

this.setLocation(x, y);

this.setLayout(null); // Set the layout to null so you can position components manually

getContentPane().setBackground(new Color(173, 216, 230));

lb14.setText("请输入你要加密的明文或要解密的密文：");

lb24.setText("请输入二进制密钥:");

lb54.setText("随机数IV:");

lb34.setText("加密结果：");

lb44.setText("解密结果：");

encry\_btn4.setBackground(Color.red);

encry\_btn4.setForeground(Color.white);

decry\_btn4.setBackground(Color.red);

decry\_btn4.setForeground(Color.white);

lb14.setBounds(50, 40, 250, 50);

lb24.setBounds(50, 100, 200, 50);

lb54.setBounds(50, 150, 200, 50);

lb34.setBounds(50, 300, 100, 50);

lb44.setBounds(50, 350, 100, 50);

message4.setBounds(50, 400, 100, 50);

input4.setBounds(300, 40, 250, 40);

keyfield4.setBounds(300, 100, 250, 40);

keyfield44.setBounds(300, 150, 250, 40);

encry\_output4.setBounds(300, 300, 250, 40);

decry\_output4.setBounds(300, 350, 250, 40);

encry\_btn4.setBounds(80, 250, 70, 30);

decry\_btn4.setBounds(300, 250, 70, 30);

// Add components to the frame

add(lb14);

add(lb24);

add(lb34);

add(lb44);

add(lb54);

add(message4);

add(input4);

add(keyfield4);

add(keyfield44);

add(encry\_output4);

add(decry\_output4);

add(decry\_btn4);

add(encry\_btn4);

encry\_btn4.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (input4.getText().trim().equals("")) {

message4.setText("<html><font color='red'>请输入明文!</font> </html>");

} else if (keyfield4.getText().trim().equals("")) {

message4.setText("<html><font color='red'>请输入密钥1!</font> </html>");

} else if (keyfield44.getText().trim().equals("")) {

message4.setText("<html><font color='red'>请输入IV!</font> </html>");

} else {

String plaintext = input4.getText();

String key1 = keyfield4.getText();

String four = keyfield44.getText();

String ciphertext = SAES.CBC\_Encrypt(plaintext, key1, four);

encry\_output4.setText(ciphertext);

}

}

});

decry\_btn4.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// Implement the CBC-A-DES decryption code

String ciphertext = input4.getText();

String key1 = keyfield4.getText();

String four = keyfield44.getText();

String plaintext = SAES.CBC\_Decode(ciphertext, key1, four);

decry\_output4.setText(plaintext);

}

});

}

public static void main(String[] args) {

mainui ui = new mainui();

}

}