

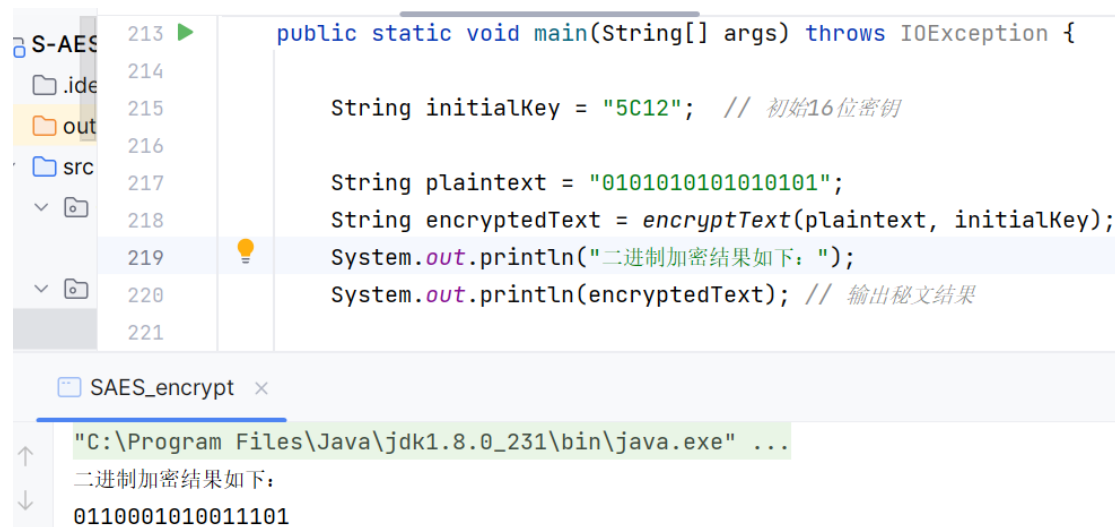
S-AES 加密解密程序测试结果

3.1 第 1 关：基本测试

根据 S-AES 算法编写和调试程序，提供 GUI 解密支持用户交互。输入可以是 16bits 的二进制型数据和 4 位 16 进制的密钥，输出是 16bit 的密文。

考虑到密钥较长，输入费时，为方便用户使用，我们将输入密钥设置为十六进制格式，这样用户只需输入 4 位即可开始加解密操作。

如下是密钥 key=5C12，即二进制 0101 1100 0001 0010，明文为 0101010101010101 的加密结果

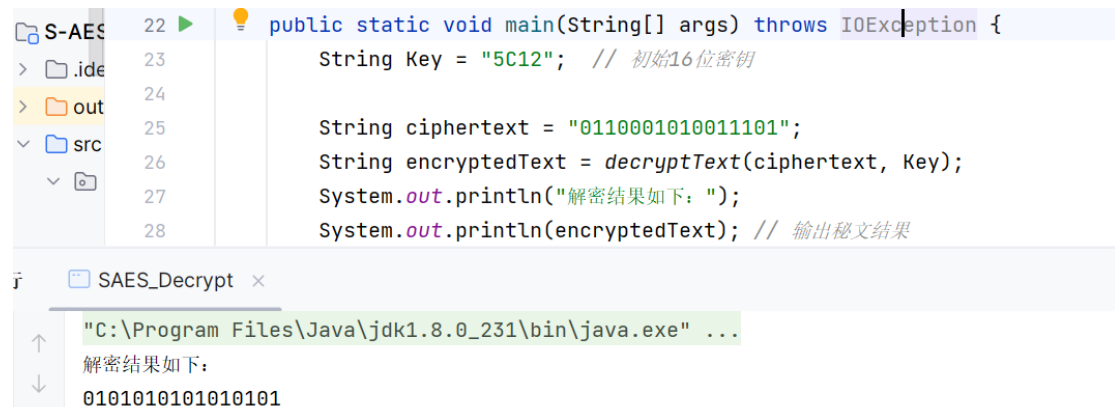


```
public static void main(String[] args) throws IOException {  
  
    String initialKey = "5C12"; // 初始16位密钥  
  
    String plaintext = "0101010101010101";  
    String encryptedText = encryptText(plaintext, initialKey);  
    System.out.println("二进制加密结果如下: ");  
    System.out.println(encryptedText); // 输出秘文结果  
  
}
```

SAES_encrypt x

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
二进制加密结果如下:
0110001010011101

对此加密结果使用相同密钥解密得：



```
public static void main(String[] args) throws IOException {  
  
    String Key = "5C12"; // 初始16位密钥  
  
    String ciphertext = "0110001010011101";  
    String encryptedText = decryptText(ciphertext, Key);  
    System.out.println("解密结果如下: ");  
    System.out.println(encryptedText); // 输出秘文结果  
  
}
```

SAES_Decrypt x

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
解密结果如下:
0101010101010101

解密结果与加密使用的明文一致

3.2 第 2 关：交叉测试

考虑到是"算法标准"，所有人在编写程序的时候需要使用相同算法流程和转换单元(替换盒、列混淆矩阵等)，以保证算法和程序在异构的系统或平台上都可以正常运行。

本算法已与宋选存朱佩苓组、杨家祥组进行共同测验通过。





经过交叉测试，可以看出我们的算法实现是正确的。

3.3 第3关：扩展功能

考虑到向实用性扩展，加密算法的数据输入可以是 ASCII 编码字符串 (分组为 2 Bytes)，对应地输出也可以是 ASCII 字符串 (很可能是乱码)。

如下是密钥 `key=5C12`，即二进制 0101 1100 0001 0010，明文为 `we are lsyt from cqu` 的加密结果

```

213 public static void main(String[] args) throws IOException {
214
215     String Key = "5C12"; // 初始16位密钥
216     String plaintextASCII = "we are lsyt from cqu"; // 输入String类型
217     String encryptedTextASCII = encryptASCII(plaintextASCII, Key);
218     System.out.println("加密结果如下: ");
219     System.out.println(encryptedTextASCII);

```

SAES_encrypt x

```

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
加密结果如下:
0Q$SJ!tE k0!%U0

```

将加密结果输入解密文件中作为密文输入，得到解密结果与加密明文一致

```
21
22 public static void main(String[] args) throws IOException {
23     String Key = "5C12"; // 初始16位密钥
24     String encryptedTextASCII = "0Q$$J!t£\u0003kø;%Ûø\u000B\u0004@J\u0091";
25     String decryptedTextASCII = decryptASCII(encryptedTextASCII, Key);
26     System.out.println("解密结果如下: ");
27     System.out.println(decryptedTextASCII); // 输出解密结果

```

SAES_Decrypt x

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...

解密结果如下:
we are lsyt from cqu

3.4 第4关：多重加密

3.4.1 双重加密

将 S-AES 算法通过双重加密进行扩展，分组长度仍然是 16 bits，但密钥长度为 32 bits。

输入 8 位 16 进制密钥 5C125C12，及明文 0101010101010101，进行双重加密

```
String key = "5C125C12";
String plaintext = "0101010101010101";

// 调用doubleEncrypt方法测试双重加密
String doubleEncryptedText = doubleEncrypt(plaintext, key);
System.out.println("双重加密结果如下: ");
System.out.println(doubleEncryptedText);
// 调用doubleDecrypt方法测试双重解密

```

MultipleEncryptandDecrypt x

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...

双重加密结果如下:
11111111100001011

将加密得到密文使用相同密钥进行解密，得到结果与明文一致

```
13 // 调用doubleDecrypt方法测试双重解密
14 String ciphertext = "1111111100001011";
15 String doubleDecryptedText = doubleDecrypt(ciphertext, key);
16 System.out.println("双重解密结果如下: ");
17 System.out.println(doubleDecryptedText); // 输出双重加密的密文

运行 MultipleEncryptandDecrypt x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
双重解密结果如下:
0101010101010101
```

3.4.2 中间相遇攻击

假设你找到了使用相同密钥的明、密文对(一个或多个)，请尝试使用中间相遇攻击的方法找到正确的密钥 Key (K_1+K_2)。

使用双重加密和密钥 key 对明文进行加密操作，得到密文

```
64 // 定义两个密钥和明文
65 String key = "FFBCD36D";
66 String plaintext = "0000000000000000";
67
68 // 调用doubleEncrypt方法测试双重加密
69 String doubleEncryptedText = doubleEncrypt(plaintext, key);
70 System.out.println("双重加密结果如下: ");
71 System.out.println(doubleEncryptedText);
72 // 调用doubleDecrypt方法测试双重解密

MultipleEncryptandDecrypt x SAES_encrypt x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
双重加密结果如下:
0011011011010010
```

对所得明密文对进行中间相遇攻击后，我们找到了很多对 key1 和 key2，并在其中找到了我们加密明文时使用的密钥


```
String key3 = "5C125C125C12";
String plaintext = "0101010101010101";
// 调用tripleEncrypt方法测试三重加密
String tripleEncryptedText = tripleEncrypt(plaintext, key3);
System.out.println("三重加密结果如下: ");
System.out.println(tripleEncryptedText); // 输出三重加密的密文
```

MultipleEncryptandDecrypt x

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...

三重加密结果如下:

0011101001001010

对三重加密得到的密文使用相同密钥进行解密，可得到输入明文

```
String tripleCiphertext="0011101001001010";
String tripleDecryptedText = tripleDecrypt(tripleCiphertext, key3);
System.out.println("三重解密结果如下: ");
System.out.println(tripleDecryptedText); // 输出三重解密的明文
```

MultipleEncryptandDecrypt x

"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...

三重解密结果如下:

0101010101010101

3.5 第 5 关：工作模式

基于 S-AES 算法，使用密码分组链 (CBC) 模式对较长的明文消息进行加密。注意初始向量 (16 bits) 的生成，并需要加解密双方共享。

在 CBC 模式下进行加密，并尝试对密文分组进行替换或修改，然后进行解密，请对比篡改密文前后的解密结果。

使用 4 位 16 进制密钥 5C12，16bits 初始向量以及 64bits 长明文进行 CBC 加密

```
String plaintext = "000000001111111000000001111111000000001111111000000001111111";
String initialVector = "1010101010101010"; // 16位初始向量
String key = "5C12"; // 16-bit key

String encryptedText = CBCModeEncrypt(plaintext, initialVector, key);
System.out.println("加密结果如下: ");
System.out.println(encryptedText); // 输出加密结果
```

```
CBC x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
加密结果如下:
000110111100000000101000001111110011011001000011011000011001110
```

对得到的密文使用相同密钥和初始向量进行解密，可得到输入明文

```
String initialVector = "1010101010101010"; // 16位初始向量
String key = "5C12"; // 16-bit key

String encryptedTextforDecrypt = "000110111100000000101000001111110011011001000011011000011001110";
String decryptedText = CBCModeDecrypt(encryptedTextforDecrypt, initialVector, key);
System.out.println("解密结果如下: ");
System.out.println(decryptedText); // 输出解密结果
}
```

```
CBC x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
解密结果如下:
000000001111111000000001111111000000001111111000000001111111
```

对最后一组密文进行修改，再次使用相同密钥和初始向量解密，结果只有最后一组 16bits 发生改变

```
public static void main(String[] args) {
    String plaintext = "000000001111111000000001111111000000001111111000000001111111";
    String initialVector = "1010101010101010"; // 协商好的16位初始向量
    String key = "5C12"; // 密钥

    String encryptedTextforDecrypt = "000110111100000000101000001111110011011001000011011000011001000";
    String decryptedText = CBCModeDecrypt(encryptedTextforDecrypt, initialVector, key);
    System.out.println("解密结果如下: ");
}
```

```
CBC x
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
解密结果如下:
000000001111111000000001111111000000001111111001000001110101
```