|  |  |
| --- | --- |
| **计算机网络安全与管理** | |
| RC4的实现 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
|  | |
|  | |
|  | |
| **姓名** | 易俊泉 |
| **班级** | 软件92班 |
| **学号** | 2194411245 |
| **电话** | 18813517223 |
| **Email** | 1302190004@stu.xjtu.edu.cn |
| **日期** | 2022-6-4 |

# RC4的实现

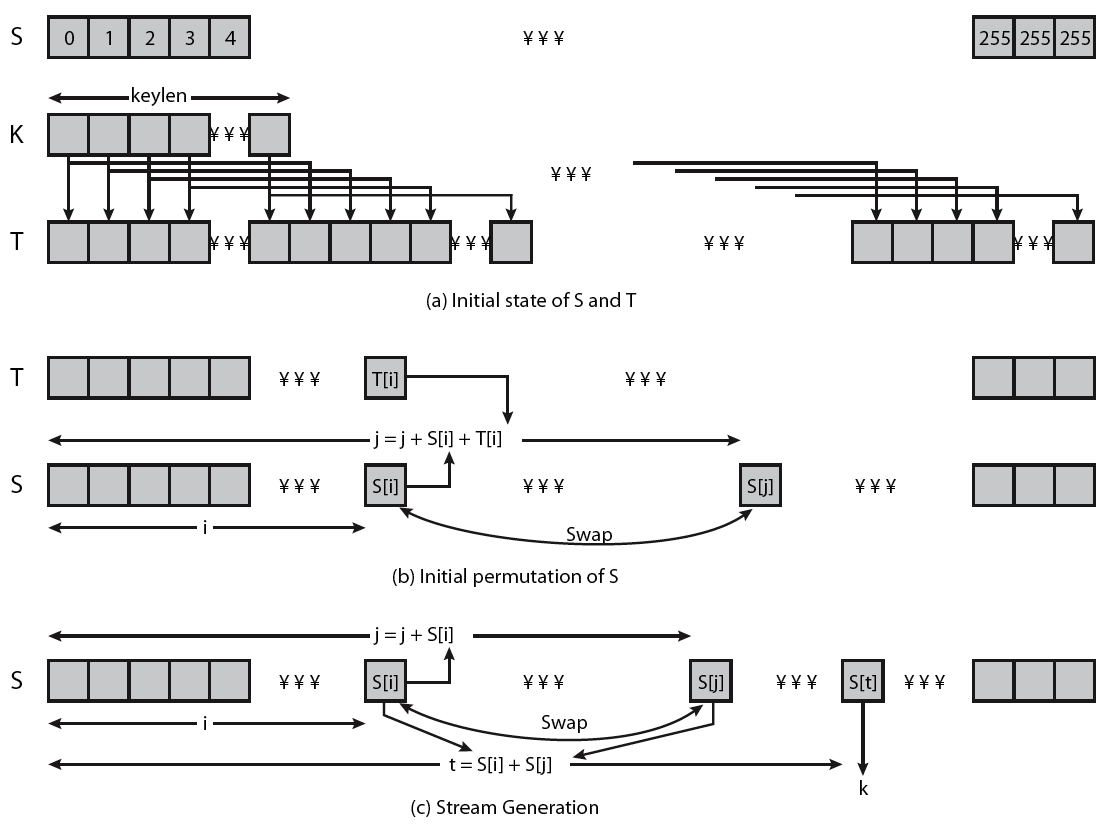
### 实验要求

1. 编码实现RC4，简单分析代码
2. 加解密个人信息，包含姓名学号，等相关信息

### 实验原理[[1]](#footnote-1)

RC4是RonRivest为RSA公司在1987年设计的一种流密码。它是一个可变密钥长度、面向字节操作的流密码。该算法以随机置换作为基础。分析显示该密码的周期很可能大于。每输出1字节的结果仅需要8~16条机器操作指令，软件实现的该密码运行很快。RC4应用很广，例如，它用于作为EEE802.l1无线局域网标准一部分的WEP(WiredEquivalentPrivacy)协议和新的WiFi受保护访问协议(WPA)中。作为可选，它也被用于SecureShell(SSH)和Kerberos中。在当时RC4作为RSA公司的商业机密并没有公开。直到1994年9月，RC4算法才通过Cypherpunks匿名邮件列表匿名地公开于Internet上。

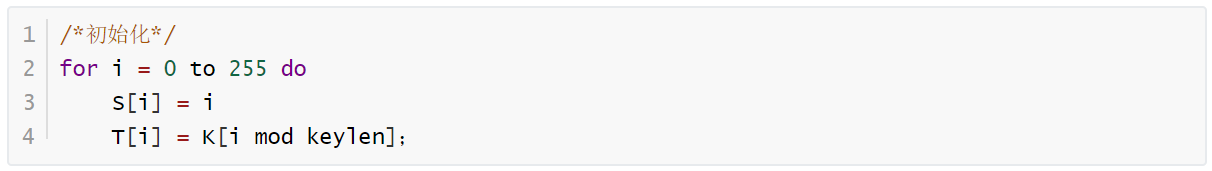
RC4算法非常简单，易于描述：用1~ 256个字节(8~2048位)的可变长度密钥初始化一个256个字节的状态向量S，S的元素记为S [0]，S [1]，…，S [255]，从始至终置换后的S包含从0~255所有的8位数。对于加密和解密，字节k是从S的255个元素中按一种系统化的方式选出的一个元素生成的。每生成一个k值，S中的元素个体就被重新置换一次。



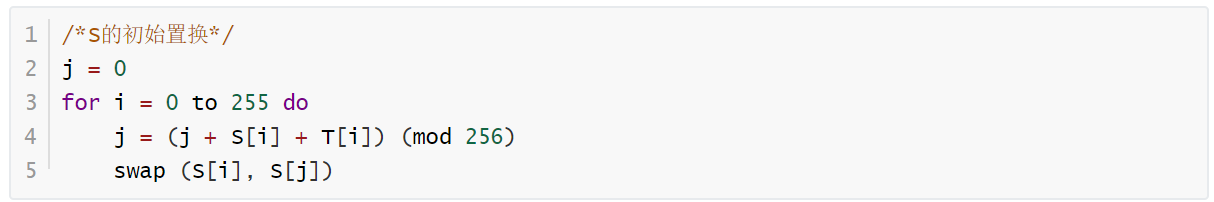
图片 1 算法逻辑结构

#### 初始化*S*

开始时，S中元素的值按升序被置为从0~255，即S[0]，S[1]，…，S[255]。同时建立一个临时向量T。如果密钥K的长度为256字节，则将K赋给T。否则若密钥长度为keylen个字节(keylen<256),则将K的值赋给T的前keylen个元素，并循环重复用K的值赋给T剩下的元素，直到T的所有元素都被赋值。这些预操作可被概括为如下：



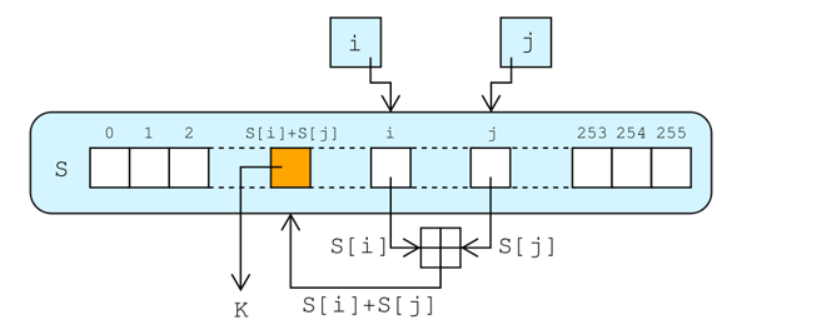
然后用T产生S的初始置换，从S[0]到S[255],对每个S[i]，根据由T[i]确定的方案，将S[i]可置换为S中的另一字节。



因为对S的操作仅是交换，所以唯一的改变就是置换。S仍然包含所有值为0~255的元素。

#### 密钥流的产生

向量S一旦完成初始化，输入密钥就不再被使用。密钥流的生成过程是，从S[0]到S[255]，对每个S[i]，根据S的当前配置，将S[i]与S中的另一字节置换。当S[255]完成置换后，操作继续重复从S[0]开始。



图片 2 密钥流的产生

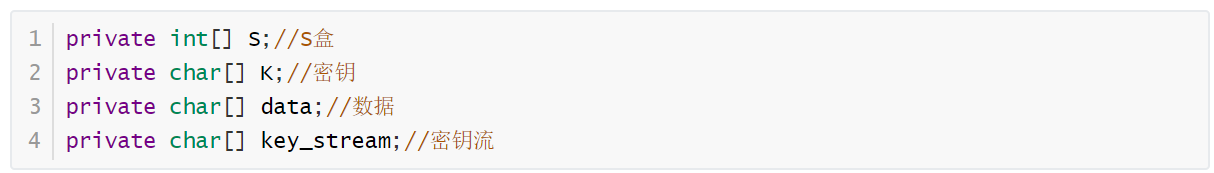


加密中，将k的值与明文的下一字节异或；解密中，将k的值与密文的下一字节异或。

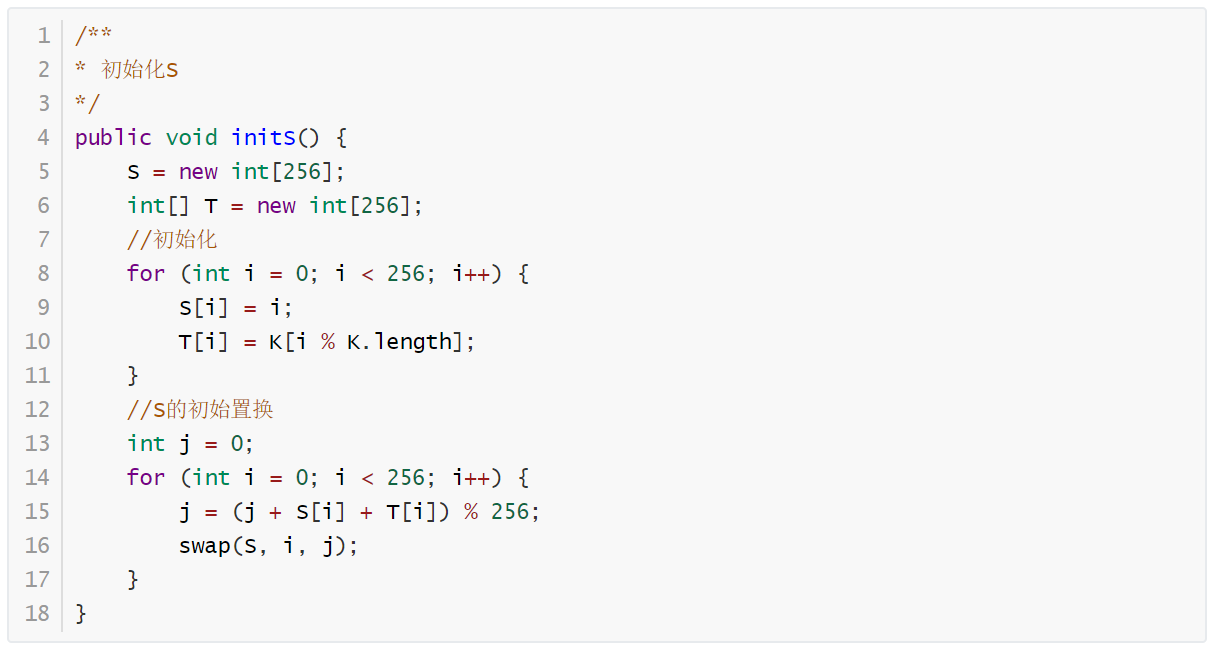
### 代码说明

这里我使用java来实现RC4。

首先定义了几个变量



#### 初始化*S*



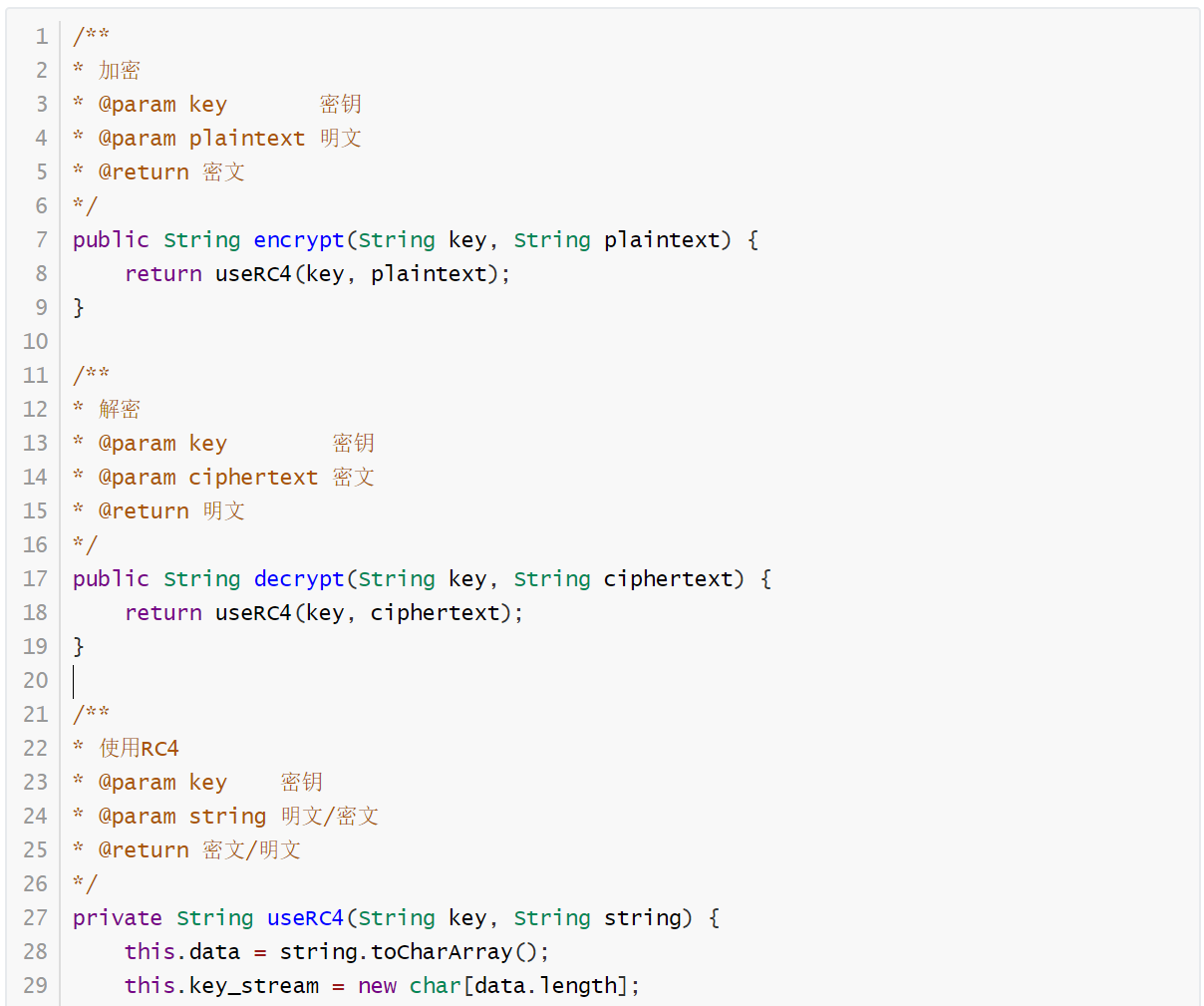
#### 密钥流的产生

根据 S 盒生成与明文长度相同的秘钥流。



#### 加密与解密

由于异或运算的特性，使得加密与解密过程一致。**如果输入的是明文，输出的就是密文；如果输入的是密文，输出的就是明文**。调用过程如下：





### 实验结果

测试数据如下：

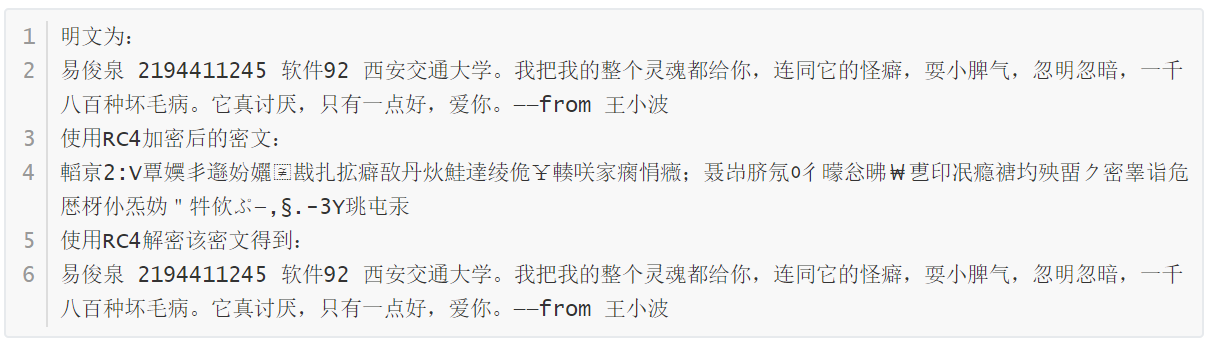


测试代码如下





实验结果如下：



**综上，该实验能成功实现简单的RC4加密解密。**

1. 该内容均参考《密码编码学与网络安全（第七版）》 [↑](#footnote-ref-1)