# 密码学实验实验报告七

18374480-黄翔

2021 年 5 月 21 日

## 1 实验目的

1. 了解常用的流密码算法, 并对其进行实现
2. 了解常用的伪随机数生成算法, 并对其进行实现

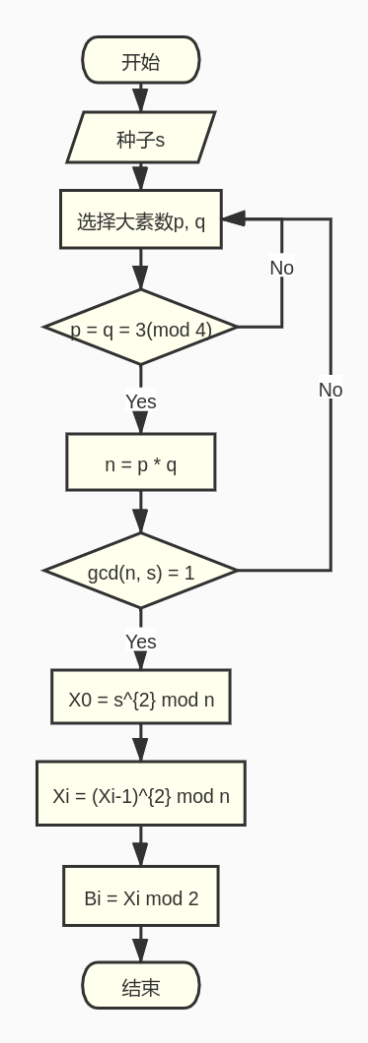
## 2 实验环境

1. python 2.7.18

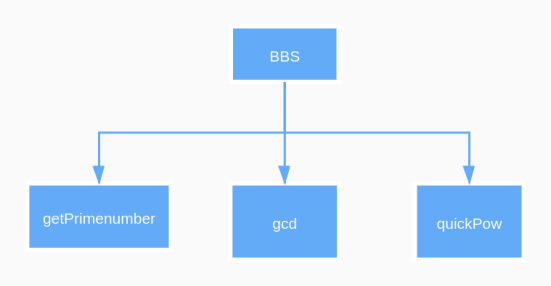
# 3 实验内容

## 3.1 BBS 伪随机数生成算法

### 3.1.1 算法流程图



### 3.1.2 函数调用关系




### 3.1.3 算法伪代码

**Algorithm 1** BBS 伪随机数生成算法

**Input:** 种子 $seed$

**Output:** 比特流 $B_i, i = 1, 2, 3, \cdots$

选择大素数 $p, q$ 满足 $p \equiv q \equiv 4 \mod 4$ 且 $gcd(p \times q, seed) = 1$
$n = p \times q$
$X_0 = seed^2 \mod n$
**for** $i \leftarrow 1$ to $\infty$ **do**
    $X_i = X_{i-1}^2 \mod n$
    **yield** $B_i = X_i \mod 2$
**end for**

### 3.1.4 测试样例及结果截图

选择 $n = 19649$, $seed = 101355$，结果如图

```
[1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0]
```

### 3.1.5 总结

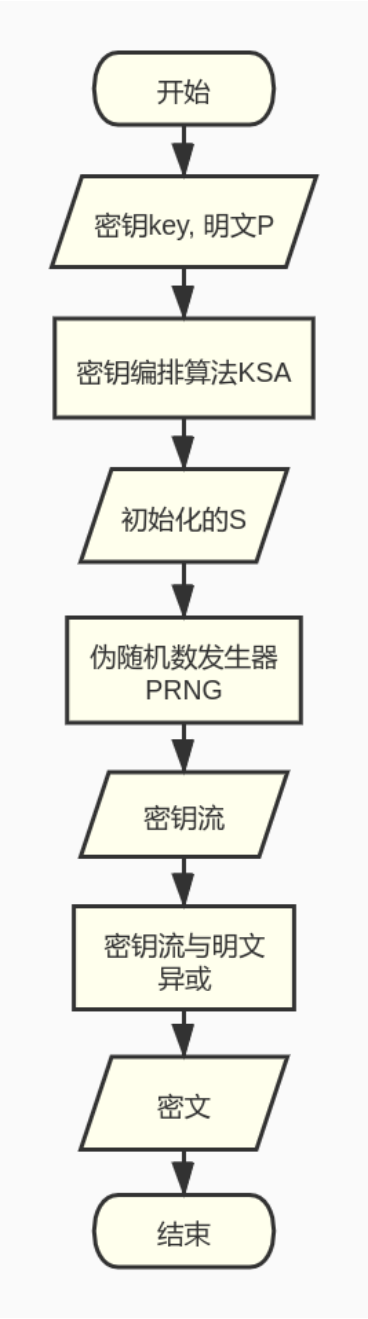
**BBS 安全性** BBS 的安全性是基于对 $n$ 的因子分解的困难性上的。当素数选择合适，$n$ 足够大，即使得到 $X_n$ 的低 $O(loglogM)$ 位，要得到 $X_n$ 至少与求解模 $n$ 的二次剩余一样困难

**$p, q$ 的选择** $p, q$ 应该满足 $p \equiv q \equiv 3 \mod 4$，这保证每个二次剩余都有一个同样有二次剩余的平方根。若为进一步提高安全性，则 $p, q$ 应该选择为

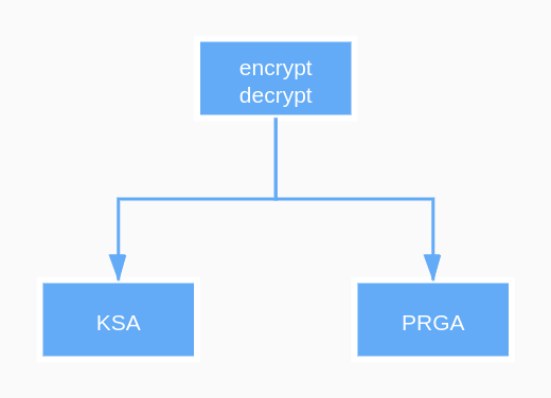
Sophie Germain 素数（$2 \times p + 1$ 也是素数），且 $gcd(\frac{p-3}{2}, \frac{q-3}{2})$ 尽量小，这时循环长度最大

## 3.2 RC4 流密码

### 3.2.1 算法流程图



### 3.2.2 函数调用关系




### 3.2.3 算法伪代码

**Algorithm 2** RC4 加密

**Input:** 明文 $P$，密钥 $key$

**Output:** 密文 $C$

**function** $KSA(key)$
    $keyLen \leftarrow len(key)$
    **for** $i \leftarrow 0$ to 255 **do**
        $S$.append($i$)
    **end for**
    $j = 0$
    **for** $i \leftarrow$ 0to 255 **do**
        $j = (j + S[i] + ord(key[i \mod keyLen])) \mod 256$
        $swap(S[i], S[j])$
    **end for**
    **return** $S$
**end function**
**function** $PRGA(S)$
    $i \leftarrow 0, j \leftarrow 0$
    **for** $i \leftarrow 0$ to $len(P)$ **do**
        $i \leftarrow (i + 1) \mod 256$
        $j \leftarrow (j + S[i]) \mod 256$
        $swap(S[i], S[j])$

19: $K$.append($S[(S[i]+S[j]) \mod 256]$)
20: **end for**
21: **return** $K$
22: **end function**
23: $S \leftarrow KSA(key)$
24: $K \leftarrow PRGA(S)$
25: $C \leftarrow K \oplus P$
26: **return** $C$

**Algorithm 3** RC4 解密

**Input:** 密文 $C$，密钥 $key$
**Output:** 明文 $P$
1: 与加密相同

### 3.2.4 测试样例及结果截图


### 3.2.5 总结

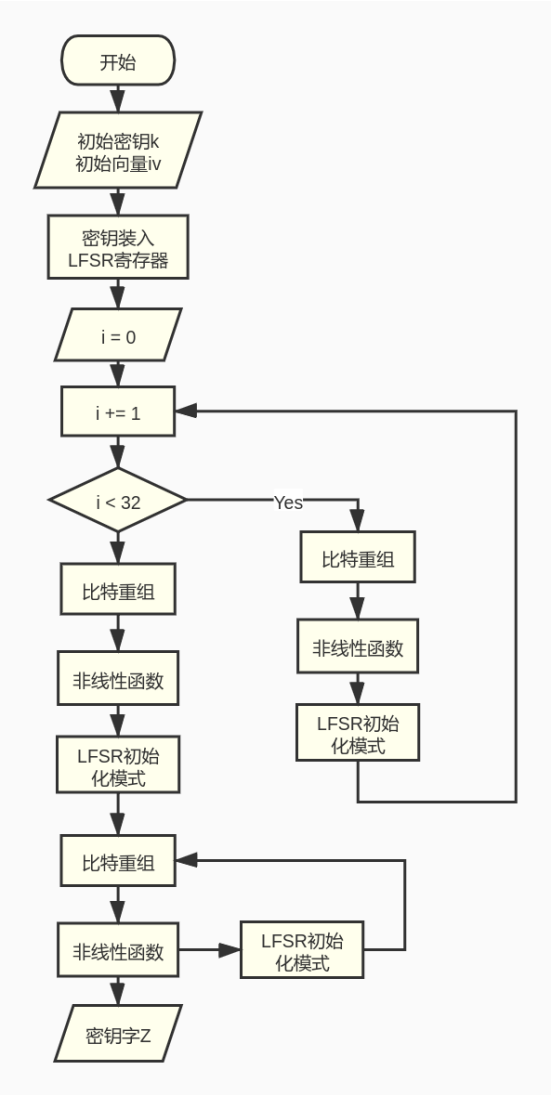
**初始 key 的获得** 实际运用中，初始 key 一般由真随机数发生器 TRNG 生成

**RC4 算法实用性** RC4 是流密码，所以它比普通的分组密码更具延展性。如果不与 MAC 消息认证码一起使用，则容易受到位翻转攻击。

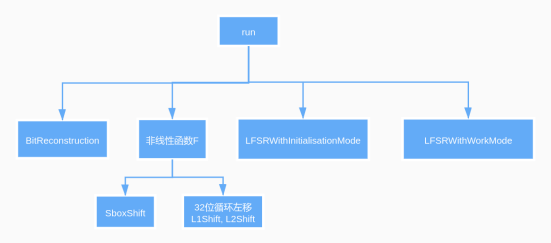
**其它** RC4 流密码算法密钥流的生成与明文独立，属于同步流密码。

## 3.3 祖冲之序列密码算法

### 3.3.1 算法流程图



### 3.3.2 函数调用关系




### 3.3.3 算法伪代码

**Algorithm 4** ZUC_128

**Input:** 密钥 $k$(128 bits), 初始向量 $iv$(128 bits)

**Output:** 密钥字 $Z_i$, $i = 1, 2, \cdots$

**function** LFSRWithInitialisationMode($u$)

$v \leftarrow 2^{15} \times s_{15} + 2^{17} \times s_{13} + 2^{21} \times s_{10} + 2^{20} \times s_4 + (1 + 2^8) \times s_0 \mod (2^{31} - 1)$

$s_{16} \leftarrow (v + u) \mod (2^{31} - 1)$

**if** $s_{16amssymb}$ is 0 **then**

$s_{16} \leftarrow 2^{31} - 1$

**end if**

$(s_0, s_1, \cdots, s_{15}) \leftarrow (s_1, s_2, \cdots, s_{16})$

**end function**

**function** LFSRWithWorkMode( )

$s_{16} \leftarrow 2^{15} \times s_{15} + 2^{17} \times s_{13} + 2^{21} \times s_{10} + 2^{20} \times s_4 + (1 + 2^8) \times s_0 \mod (2^{31} - 1)$

**if** $s_{16}$ is 0 **then**

$s_{16} \leftarrow 2^{31} - 1$

**end if**

$(s_0, s_1, \cdots, s_{15}) \leftarrow (s_1, s_2, \cdots, s_{16})$

**end function**

**function** BitReconstruction( )

$X_0 \leftarrow s_{15H} || s_{14L}$

$X_1 \leftarrow s_{11L} || s_{9H}$

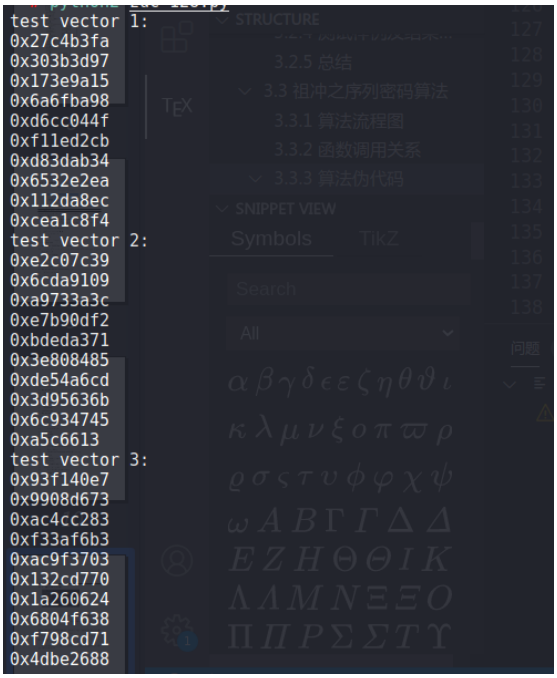
$X_2 \leftarrow s_{7L} || s_{5H}$

$X_3 \leftarrow s_{2L}||s_{0H}$
**return** $X \leftarrow [X_0, X_1, X_2, X_3]$
**end function**
**function** F($X$)
$W \leftarrow (X_0 \oplus R_1) \boxplus R_2$
$W_1 \leftarrow R_1 \boxplus X_1$
$W_2 \leftarrow R_2 \oplus X_2$
$R_1 \leftarrow S(L_1(W_{1L}||W_{2H}))$
$R_2 \leftarrow S(L_2(W_{2L}||W_{1H}))$
**return** $W$
**end function**
**function** KEYPACK($k, iv$)
$k$ is $k_0||k_1||\cdots||k_{15}$
$iv$ is $iv_0||iv_1||\cdots||iv_{15}$
$D$ is $d_0||d_1||\cdots||d_{15}$
**for** $i \leftarrow 0$ to 15 **do**
$s_i \leftarrow k_i||d_i||iv_i$
**end for**
**end function**
**function** RUN($k, iv$)
keyPack($k, iv$)
**for** $i \leftarrow 0$ to 31 **do**
$X \leftarrow$ BitReconstruction()
$W \leftarrow$ F($X$)
LFSRWithInitialisationMode($W >> 1$)
**end for**
$X \leftarrow$ BitReconstruction()
$W \leftarrow$ F($X$)
LFSRWithWorkMode()
**for** $i \leftarrow$ 1to $\infty$ **do**
$X \leftarrow$ BitReconstruction()
**yield** $X[3] \oplus$F($X$)
LFSRWithWorkMode()

53: **end for**

54: **end function**

### 3.3.4 测试样例及结果截图




### 3.3.5 总结

**特性** ZUC_128 流密码算法密钥流产生与明文独立，属于同步流密码。

**密钥生成器**

1. 驱动部分：LFSR，控制密钥发生器的状态序列，并为非线性组合序列提供统计特性好的序列
2. 非线性组合部分：比特重组，非线性函数 F，通过一系列复杂变换，将输入序列组合成密码学特性好的序列

# 4 总结

通过本次实验, 我了解常用的流密码算法以及常用的伪随机数生成算法, 并对其进行实现。进一步了解了随机数的随机性以及独立性。以及流密码的 LSPF 线性移位寄存器序列产生周期长、线性复杂度高、统计特性好的初始乱源的作用

# 5 思考题

使得初始置换后 S 的顺序不变的密钥属于弱密钥, 即 $i \equiv (j+S[i]+T[i]) \mod 256$