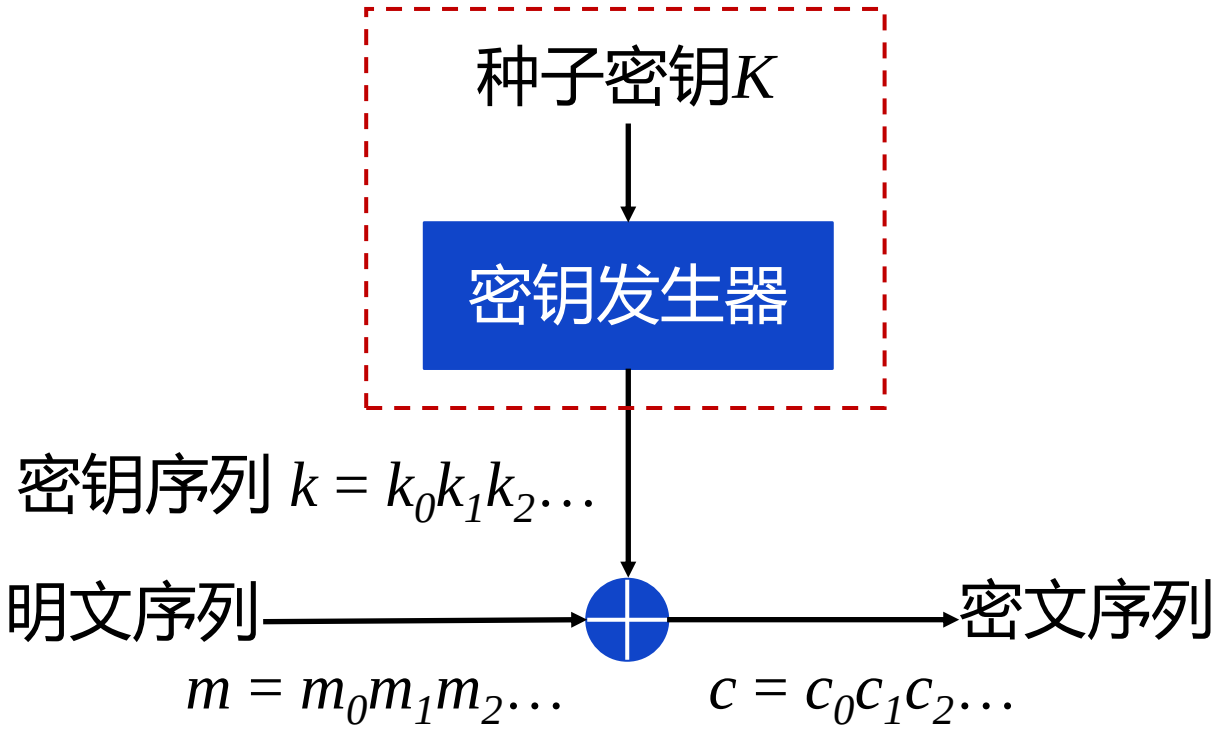
密 码 学

第三章 序列密码算法

网络空间安全学院
朱 丹
zhudan@nwpu.edu.cn

知识回顾—序列密码-原理

- 将一串较短的种子密钥K通过密钥流发生器扩展成足够长的伪随机密钥流$k = k_0k_1k_2\ldots$
- 加密变换：$c_i = m_i \oplus k_i$
- 解密变换：$m_i = c_i \oplus k_i$



- 序列密码最关键的环节在于密钥流的生成
- 加解密变换很简洁

知识回顾—序列密码的分类

Types of Stream Ciphers

同步序列密码（Synchronous Stream Cipher）

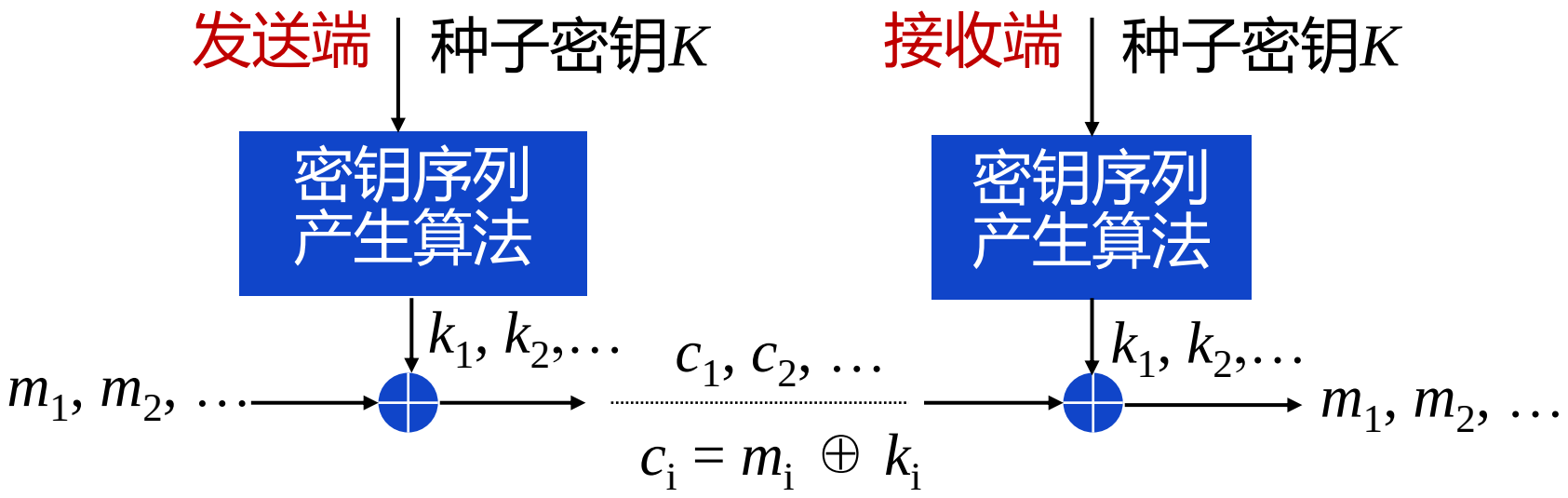
密钥序列产生算法与明文（密文）无关，通信双方必须保持精确的同步，失步将导致无法解密

自同步序列密码（Self- Synchronous Stream Cipher）

密钥序列产生算法与明文（密文）相关，加解密出错会造成错误的有界传播

分为同步序列密码和自同步序列密码2类

知识回顾—同步序列密码



设密文失步 $c = c_1, c_3, c_4, \ldots c_{n-1}, c_n$ (c_2 丢失)

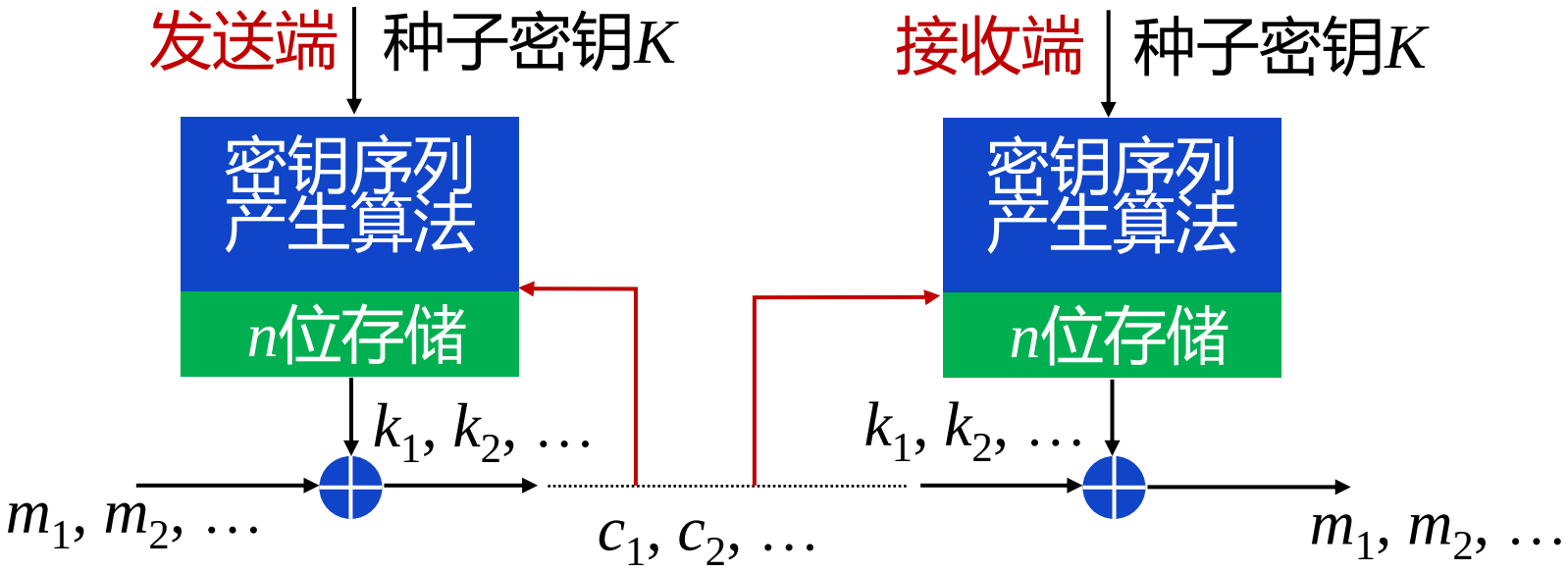
$\oplus\ k = k_1, k_2, k_3, \ldots k_{n-2}, k_{n-1}$ (密钥同步)

$m = m_1, \times, \times, \ldots \times, \times$ (m_1 后明文全错)

失步与位翻转错误是两个不同的概念

- 同步的概念：明文、密钥和密文字符的顺序准确对应　同步的特点：密钥生成算法与明文（密文）无关
- 错误是指传输过程中值发生了变化，如密文比特的值发生的翻转

知识回顾—同步序列密码



c_i的错误将影响n位，错误有界传播

- 此类密码有自己的同步机制
- 密钥序列的产生与明文（密钥）有关

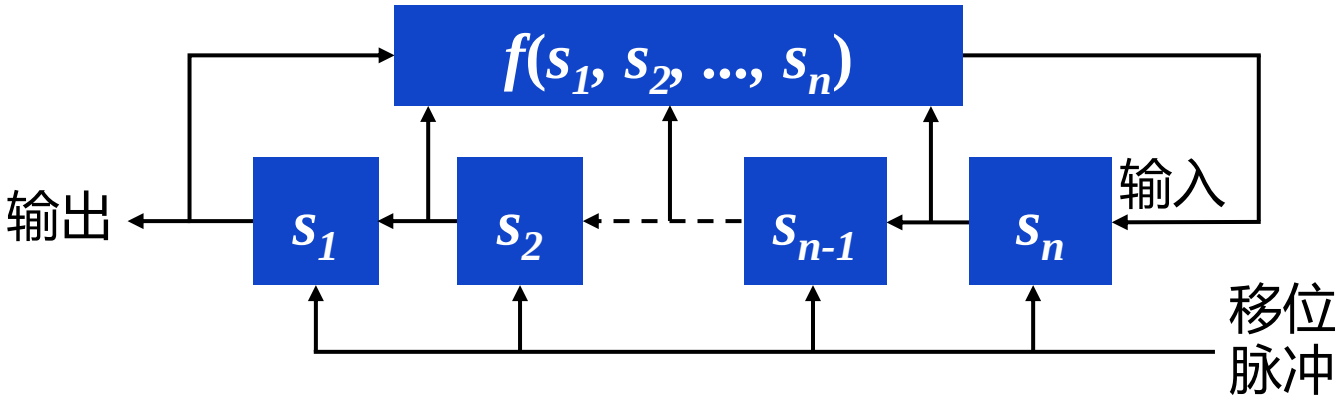
知识回顾—线性反馈移位寄存器

- 设$f(s_1, s_2, \ldots, s_n)$为线性函数，则可写成

$$f(s_1, s_2, \ldots, s_n) = g_n s_1 + g_{n-1} s_2 + \ldots + g_1 s_n$$

其中，g_1，g_2，…，g_n称为反馈系数

- 在$GF(2)$的情况下，式中的+即为$\oplus$，反馈系数$g_i \in GF(2)$

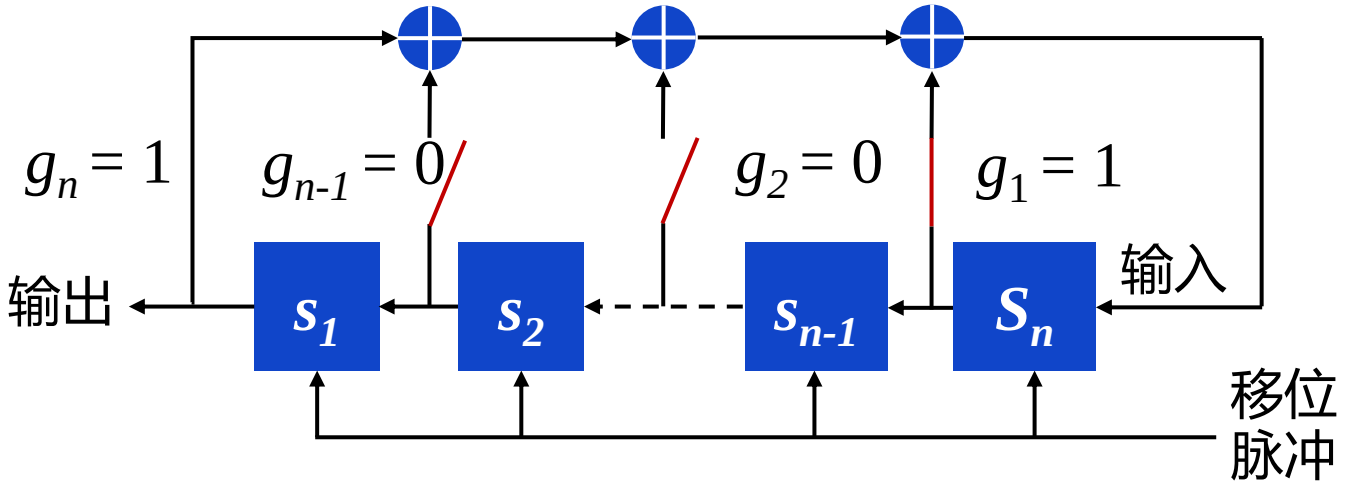


反馈系数用于确定哪些环节参加反馈

知识回顾—线性反馈移位寄存器

- 若$g_i = 0$, 则表示式中的$g_i s_i$项不存在，即s_i不连接。同理$g_i = 1$表示s_i连接
- g_i相当于一个选择开关
- 连接多项式，其中 $g_n = 1$

$$g(x) = g_n x^n + g_{n-1} x^{n-1} + \ldots + g_1 x + 1$$



反馈系数用于确定哪些环节参加反馈

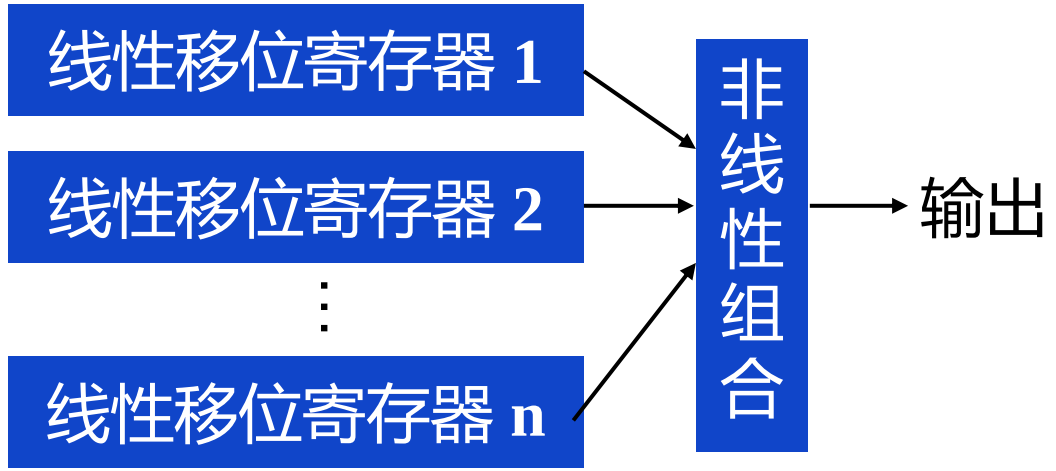
知识回顾—线性反馈移位寄存器

- n级线性反馈移位寄存器最多有2^n个不同的状态
- 若其初始状态为零，则其后续状态恒为零。若其初始状态不为零，则其后续状态也不为零
- 因此，n级线性反馈移位寄存器的状态周期 $\leq 2^n - 1$，其输出序列的周期 $\leq 2^n - 1$
- 选择合适的本原多项式（对应于反馈系数）可使线性反馈移位寄存器的输出序列周期达到最大值$2^n - 1$
- 称此时的输出序列为最大长度线性反馈移位寄存器输出序列，简称为m序列

LFSR的性质：什么时候能够达到最大周期，最大序列长度是多少，m序列的概念

知识回顾—非线性反馈移位寄存器

- 线性移位寄存器序列密码在已知明文攻击下是可破译的，可破译的根本原因在于线性移位寄存器序列是线性的，这一事实促使人们向非线性领域探索
- 目前研究得比较充分的方法包括：
  - 非线性移位寄存器序列
  - 对线性移位寄存器序列进行非线性组合
  - 利用非线性分组码产生非线性序列



- 非线性反馈移位寄存器的构造方法
- 关键点：反馈函数是非线性函数

章节安排

Outline

A5算法

RC4算法

ZUC算法

章节安排

Outline

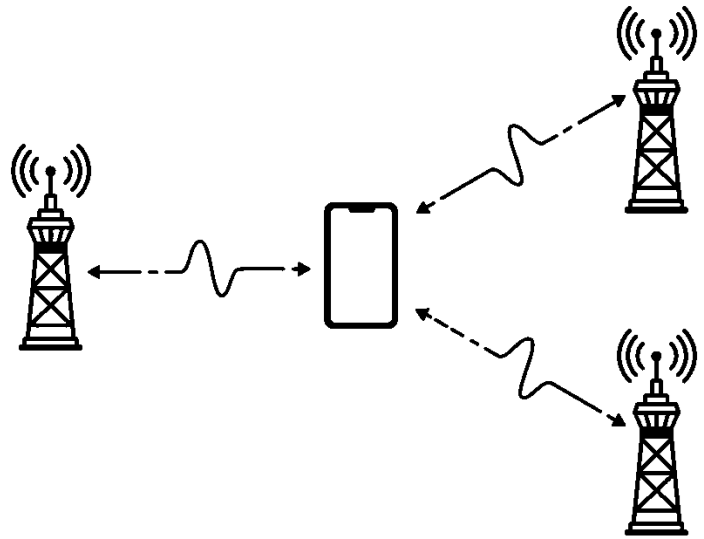
A5算法

RC4算法

ZUC算法

3.7 A5序列密码算法

- A5是欧洲移动通信GSM标准中的一个加密算法，主要用于手机到基站之间的链路语音加密
- A5由法国人设计，最初是保密的，后被公布到网络上
- A5算法尚未完全公开，不同文献介绍略有不同
  主要差别在连接多项式上



序列密码适用于移动和网络通信，具有较高的性能和较好的实时性

3.7 A5序列密码算法

- A5由3个较短的线性反馈移位寄存器LFSR1，LFSR2和LFSR3组成
- 3个线性反馈移位寄存器的位数分别是19，22和23位
- A5算法尚未完全公开，不同文献介绍略有不同
- 连接多项式：

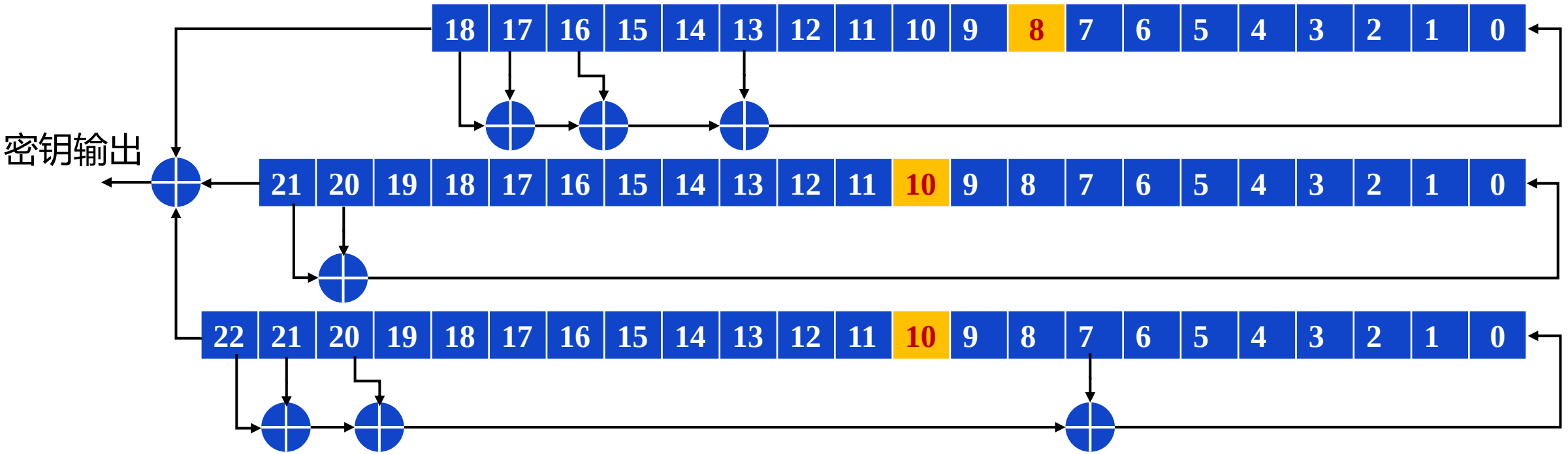
$$g1(x) = x^{19} + x^{18} + x^{17} + x^{14} + 1$$

$$g2(x) = x^{22} + x^{21} + 1$$

$$g3(x) = x^{23} + x^{22} + x^{21} + x^{8} + 1$$

3.7 A5序列密码算法

- A5算法的种子密钥为64位，作为三个线性反馈移位寄存器的初始状态
- 每个时钟周期产生一个比特的密钥，多周期形成密钥流
- 每个时钟周期产生一个比特的密钥，多周期形成密钥流（时钟脉冲控制移位）



- 密钥长度为64比特，长度较短，安全性不足
- 线性移位寄存器输出的线性组合

3.7 A5序列密码算法

- 三个线性移位寄存器的时钟脉冲由时钟控制电路提供
- 时钟控制信号分别来自LFSR1，LFSR2和LFSR3的第8，10和10位
- 若LFSR1(8) = LFSR2(10) = LFSR3(10)，则它们都移一位
- 否则控制位相同的（表决确定）两个寄存器移位，不同的那个不移位
- 每个时钟周期至少有两个寄存器移位
- 不规则动作阶段：动作100次（不输出）、114次（输出），重复

重点掌握时钟脉冲产生机制，每个LFSR各选一个Tap通过表决来确定哪两个移位

引入了更多的不确定性和非线性，安全性得到提升

3.7 A5序列密码算法

- A5一个显然的特点是三个移位寄存器的长度都太短
- 另一个安全问题是，即使种子密钥不同，也可能产生相同密钥序列
- A5算法对于目前的计算能力来说是不安全的（密钥太短）

A5面临的安全性问题，最主要的是64比特的密钥太短

章节安排

Outline

A5算法

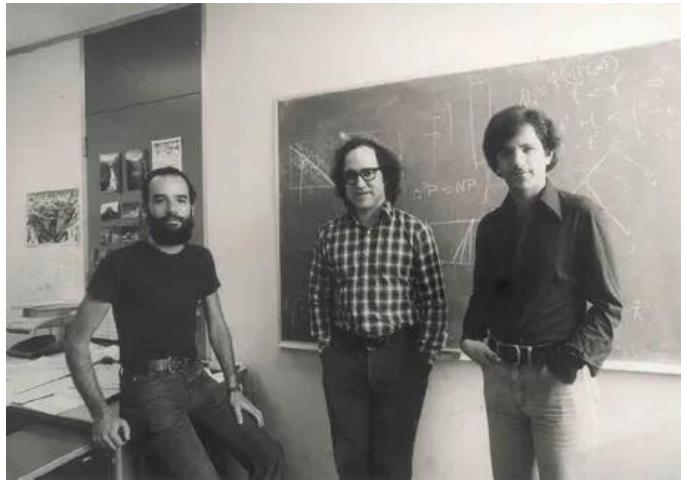
RC4算法

ZUC算法

3.8 RC4序列密码算法

- RC4 (Rivest Cipher 4) 密码是美国RSA数据安全公司设计的一种序列密码，由Ron Rivest在1987年提出了。RSA公司将其收集在加密软件BSAFE中
- 最初并没有公布RC4的算法。通过对软件进行逆向分析得到了算法（1994）
- RSA公司于1997年公布了RC4密码算法

(Left to Right: Ron Rivest, Adi Shamir, Len Adleman)



- RC4密码的设计者
- 目前OpenSSL库中还有RC4算法选项

3.8 RC4序列密码算法

- 密钥40位的RC4密码，通过Internet 32小时可攻破
- RC4密码与基于移位寄存器的序列密码不同是一种基于非线性数据表变换的序列密码
- 它以一个足够大的数据表为基础，对表进行非线性变换，产生非线性的密钥序列

3.8 RC4序列密码算法

- 在用RC4加解密之前，应当首先对S表初始化
  - 对*S*表进行线性填充，即令$S[0] = 0, S[1] = 1, S[2] = 2, \ldots, S[255] = 255$
- 用种子密钥填充另一个256字节的*T*表$T[0], T[1], \ldots, T[255]$，如果密钥的长度*n*小于*T*表的长度，则依次重复填充，直至将*T*表填满

*S*表

0	1	2	3	4	5	…	253	254	255

*T*表

```
FOR i = 0 to 255 DO
    T[i] = K[i % n]
END FOR
```

- S表初始化是无参量的线性（顺序）填充
- 密钥循环填充，密码长度少于256字节时，按顺序重复填充

初始化排列S

- $j = 0$
- 对$i = 0$ 到255重复以下操作：

  $j = (j + S[i] + T[i]) \bmod 256$ // 根据i计算得到新的j

  交换$S[i]$和$S[j]$

$j = 0$, 对$i = 0$ 到255执行：$j = (j + S[i] + T[i]) \bmod 256$

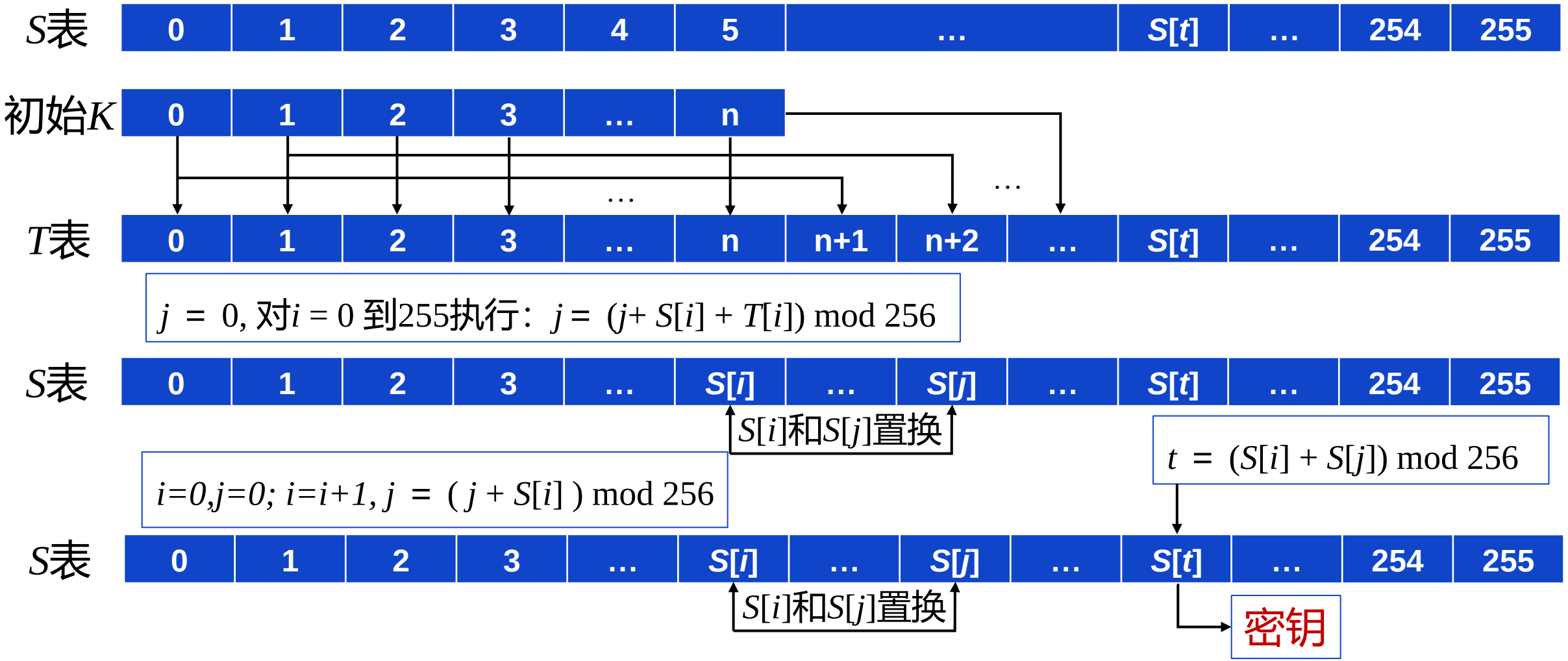
- 通过置换（换位）来实现置乱
- 回顾第二章置换的概念

3.8 RC4序列密码算法

- **产生密钥流**
  - $i = 0,\ j = 0$
  - $i = i + 1 \bmod 256$ // 不断循环
  - $j = j + S[i] \bmod 256$
  - 交换$S[i]$和$S[j]$
- RC4的输出函数定义如下：
  - $t = S[i] + S[j] \bmod 256$
  - $k = S[t]$
  - 每次循环产生1字节密钥

通过置换（换位）来实现置乱，下标的计算中存在非线性环节

3.8 RC4序列密码算法



- 掌握S表和T表的初始化填充、S表的初始置乱
- 掌握密钥选出规则

3.8 RC4序列密码算法

- RC4存在弱密钥使得其在初始置换后S的顺序不变，即$i = j$
- RC4加密是采用的XOR，一旦子密钥序列出现重复，密文就有可能被破解

  （参见Applied Cryptography一书的1.4节Simple XOR）
- 存在部分弱密钥，使得子密钥序列在不到100万字节内就发生了完全的重复
- 密钥较短的RC4实现容易受到黑客攻击，但目前没有密码分析方法对密钥长度达到128位的RC4有效
- 2001年，研究者利用WEP协议的脆弱性破译了RC4算法

  CVE-2013-2566：SSL/TLS RC4 信息泄露漏洞

  CVE-2015-2808：RC4加密漏洞

章节安排

Outline

A5算法

RC4算法

ZUC算法

3.9.1 ZUC序列密码算法

- 祖冲之序列密码算法（ZUC）是我国自主设计的密码算法
- 被批准为新一代宽带无线通信系统（LTE）国际标准，即4G国际标准
- ZUC算法由信息安全国家重点实验室等单位研制
- ZUC算法包括机密性算法128-EEA3和完整性算法128-EIA3，分别用于4G移动通信中数据机密性和完整性保护
- ZUC算法的本质是一个密钥序列产生算法
- ZUC算法在结构上属于对一个LFSR进行非线性组合的逻辑结构

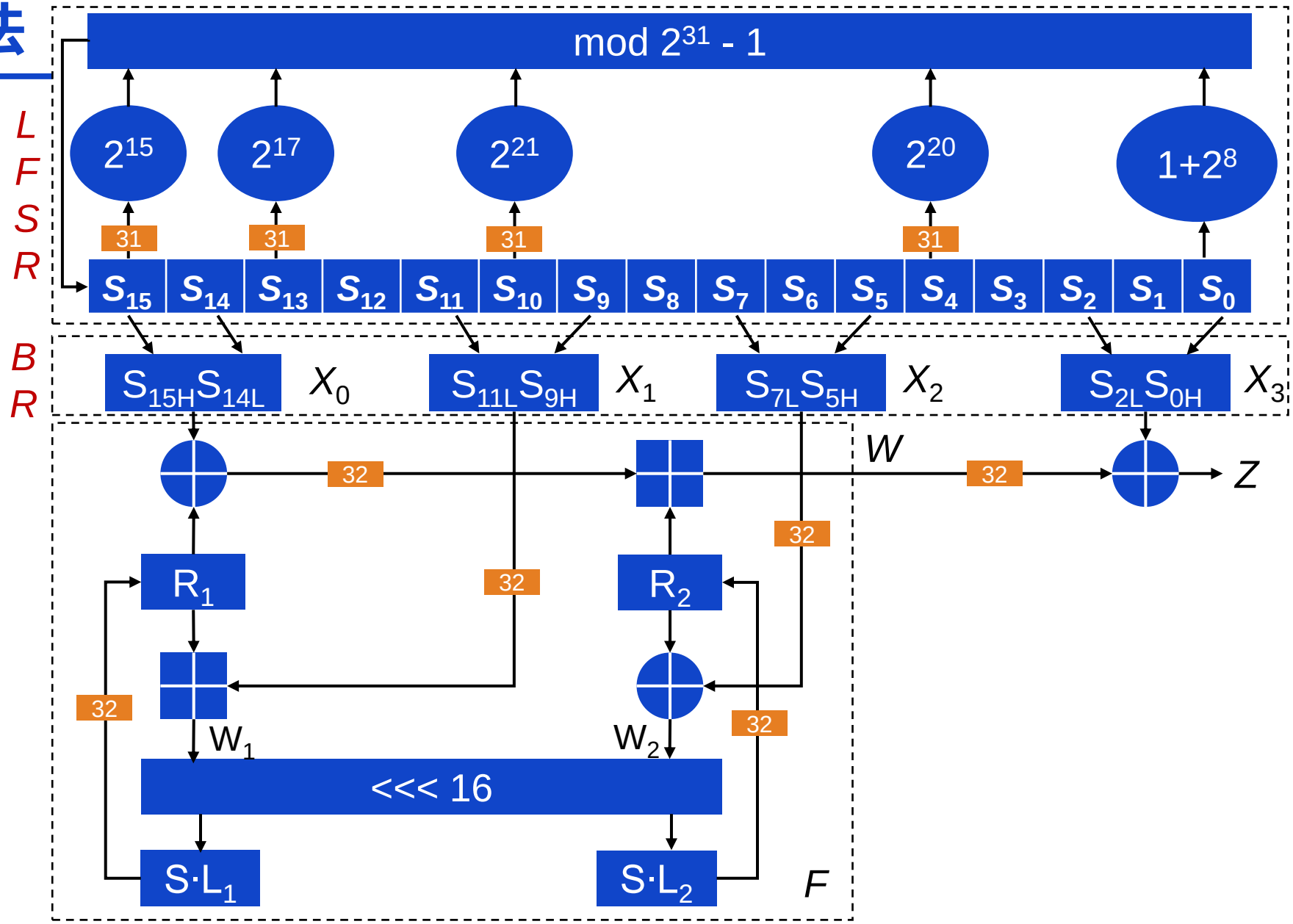
- ZUC是国密标准，也已经成为国际标准
- ZUC算法的本质还是密码序列产生算法

3.9.1 ZUC序列密码算法

模2加

mod 2^{32}加

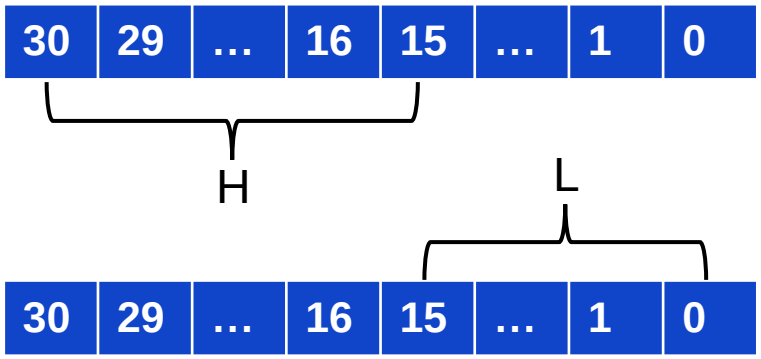
<<< 循环左移

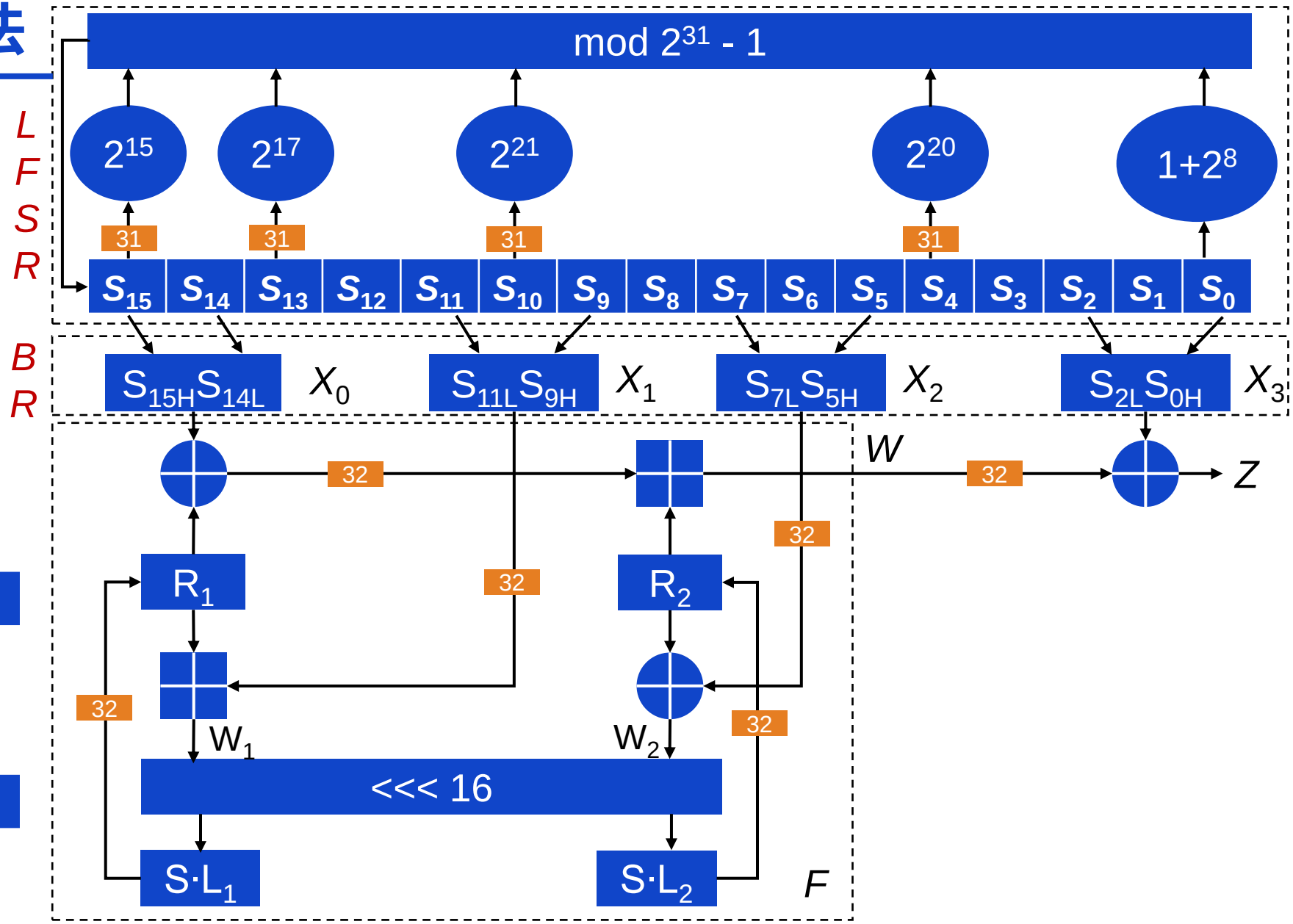


- 掌握ZUC的基本运算类型
- 注意：有限域发生了变化，寄存器的宽度也发生了变化（31比特）

3.9.1 ZUC序列密码算法

- 三层结构
  - LFSR
  - 比特重组
  - 非线性F函数





掌握三层结构和字节重组的选出规则

3.9.1 ZUC序列密码算法

- LFSR以有限域$GF(2^{31}-1)$上的16次（n = 16）本原多项式为连接多项式

$$g(x) = x^{16} - 2^{15}x^{15} - 2^{17}x^{13} - 2^{21}x^{10} - 2^{20}x^{4} - (2^{8}+1)$$

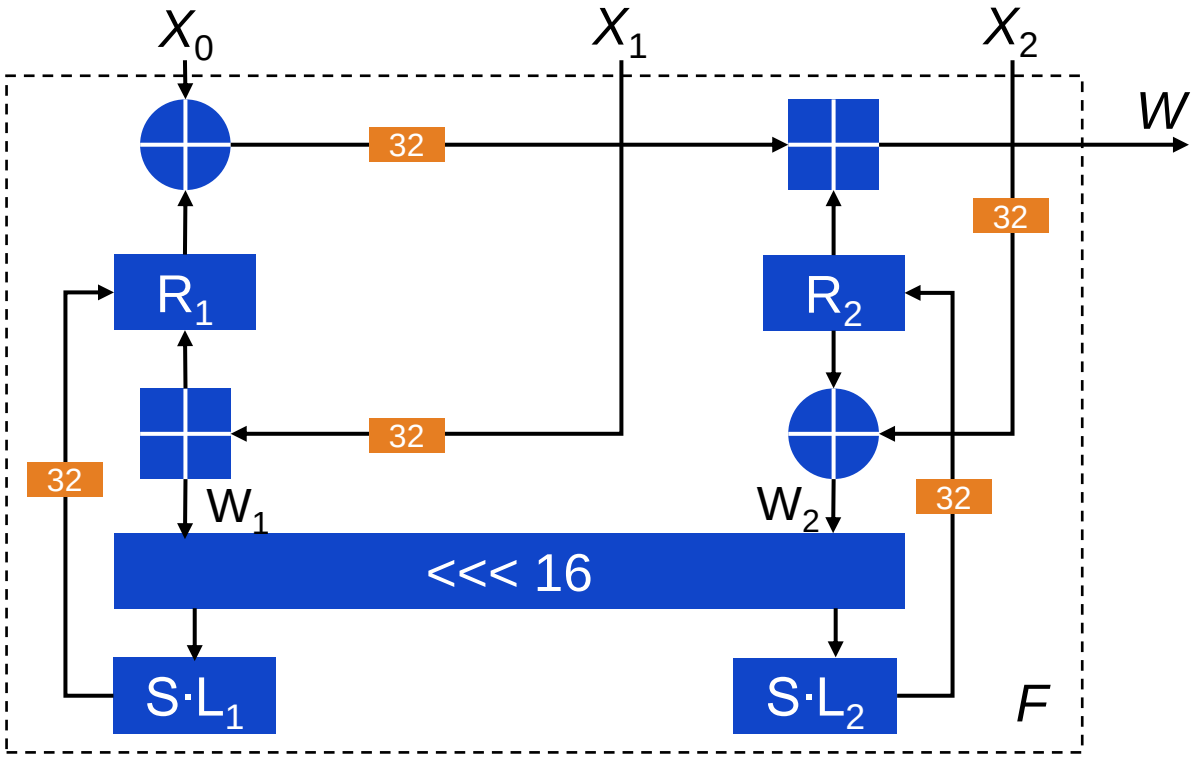
- LFSR的输出是$GF(2^{31}-1)$的m序列
- 序列周期为$(2^{31}-1)^{16}-1$
- LFSR状态长度为16个31比特，S_0至S_{15}均为31比特，在$1 \sim 2^{31}-1$之间取值
- 中间层从16*31个比特中抽取128比特，形成4个32位的字$X_0 \sim X_3$

有限域发生的变化$GF(2^{31}-1)$，要理解序列周期是如何计算得到的

3.9.1 ZUC序列密码算法

- F函数是ZUC密码算法中唯一的非线性部件，对算法安全性至关重要
  - 模2加
  - mod 2^{32}
  - 循环左移16位
  - S盒置换：4个S盒（8*8）

    $S = (S_0, S_1, S_0, S_1)$
  - L_1和L_2



$$L_1(X) = X \oplus (X<<<2) \oplus (X<<<10) \oplus (X<<<18) \oplus (X<<<24)$$

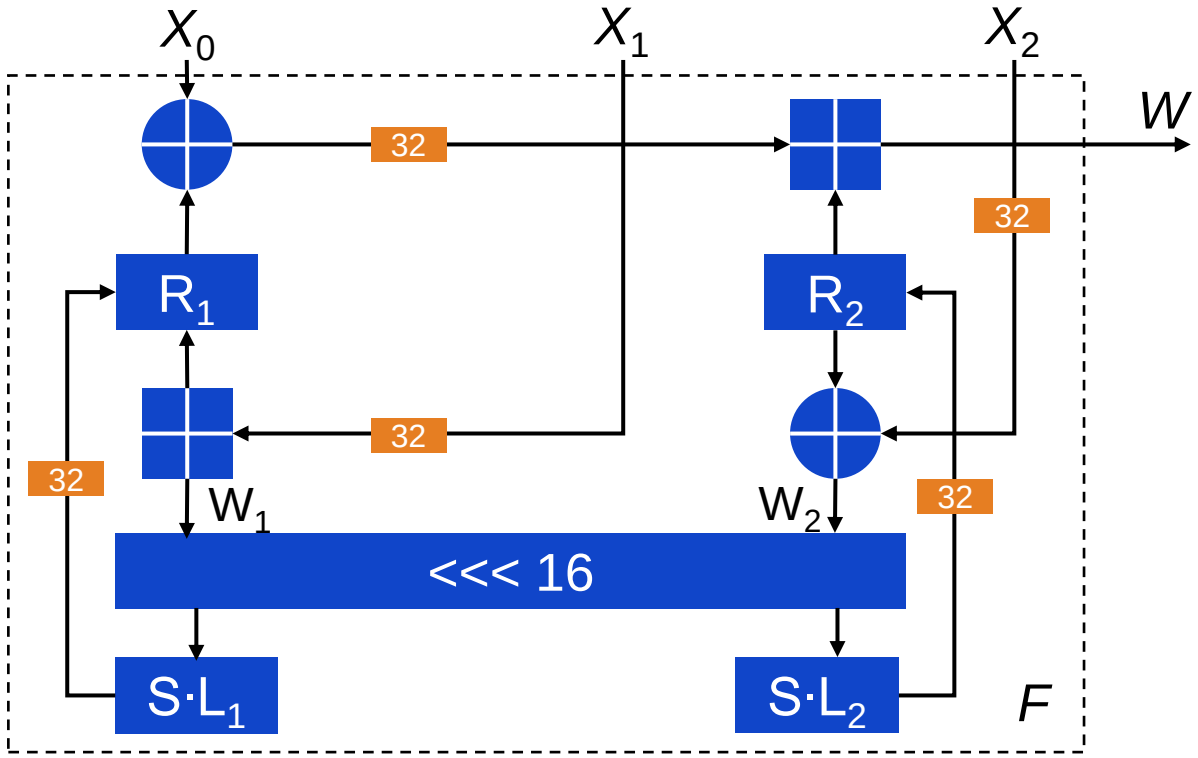
$$L_2(X) = X \oplus (X<<<8) \oplus (X<<<14) \oplus (X<<<22) \oplus (X<<<30)$$

理解F函数对应ZUC安全性的重要性，F函数是非线性部件，其中包含非线性S盒替代

ZUC使用的S盒规模；此外，ZUC使用了两种不同的S盒子，S_0和S_1

3.9.1 ZUC序列密码算法

- F函数是ZUC密码算法中唯一的非线性部件，对算法安全性至关重要



输入：

密钥 k：00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

初始向量 iv：00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

输出：

z_1：27bede74

z_2：018082da

初始化：

线性反馈移位寄存器初态：

i	S_{0+i}	S_{1+i}	S_{2+i}	S_{3+i}	S_{4+i}	S_{5+i}	S_{6+i}	S_{7+i}
0	0044d700	0026bc00	00626b00	00135e00	00578900	0035e200	00713500	0009af00
8	004d7800	002f1300	006bc400	001af100	005e2600	003c4d00	00789a00	0047ac00
t	X_0	X_1	X_2	X_3	R_1	R_2	W	S_{15}
0	008f9a00	f100005e	af00006b	6b000089	67822141	62a3a55f	008f9a00	4563cb1b
1	8ac7ac00	260000d7	780000e2	5e00004d	474a2e7e	119e94bb	4fe932a0	28652a0f
2	50cacb1b	4d000035	13000013	890000c4	c29687a5	e9b6eb51	291f7a20	7464f744

- 理解F函数对应ZUC安全性的重要性，F函数是非线性部件，其中包含非线性S盒替代
- 通过例子理解字节重组是如何进行的

3.9.1 ZUC序列密码算法

- ZUC算法运行流程
  - S1：将128位初始密钥KEY和128位初始向量IV装载到LFSR的S_0至S_{15}
  - S2：置F函数中的两个32比特寄存器R_1和R_2为0
  - S3：以初始化模式运行32次
  - S4：以工作模式运行一次，并将输出W舍弃
  - S5：进入密钥产生阶段，每时钟节拍产生32位密钥字Z

每个寄存器都是31位的，密钥和初始化向量不够填充，还有其它参数

R1和R2初始状态置零

3.9.1 ZUC序列密码算法

- S1：将128位初始密钥*KEY*和128位初始向量*IV*装载到LFSR的S_0至S_{15}
  - 128位初始密钥KEY分为16个字节 KEY = $k_0 \| k_1 \| \ldots \| k_{15}$
  - 128位初始向量IV分为16个字节IV = $iv_0 \| iv_1 \| \ldots \| iv_{15}$
  - 240比特的常量D(P106)，分为16个15比特的常量D = $d_0 \| d_1 \| \ldots \| d_{15}$

$$S_i = k_i \| d_i \| iv_i$$

- 每个寄存器都是31位的，密钥和初始化向量不够填充，还有其它参数
- 还有240比特产于密钥初始化填充，每个寄存器占15比特

3.9.1 ZUC序列密码算法

- S2：置F函数中的两个32比特寄存器R_1和R_2为0
- S3：以初始化模式运行32次
  - BitRecon()
  - $W = F(X_0, X_1, X_2)$
  - LFSRWithInitMode() // 教材P108

初始化模式：参见教材

3.9.1 ZUC序列密码算法

- S4：以工作模式运行一次，并将输出W舍弃
  - BitRecon()
  - $W = F(X_0, X_1, X_2)$
  - LFSRWithWorkMode() // 教材P108

工作模式和初始化模式的差别在于初始化模式会接收一个31比特子u

工作模式：参见教材

3.9.2 ZUC序列密码的机密性算法128-EEA3

输入参数表

输入参数	比特长度	备注
COUNT	32	计数器
BEARER	5	承载层标识
DIRECTION	1	传输方向标志
CK	128	机密性密钥
LENGTH	32	明文消息流比特长度
IBS	LENGTH	输入比特流

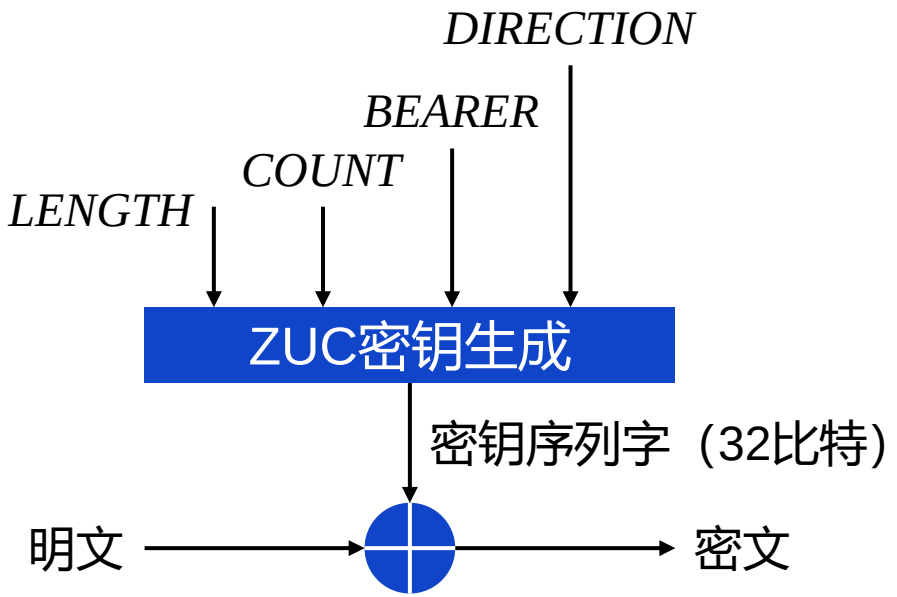
输出参数表

输入参数	比特长度	备注
OBS	LENGTH	输出比特流

- ZUC工作时的一些其它参数，用于初始化向量生成
- LENGTH是比特流的长度

3.9.2 ZUC序列密码的机密性算法128-EEA3

- 祖冲之序列密码的机密性算法128-EEA3算法结构



- ZUC工作时的一些其它参数，用于初始化向量生成
- 最主要的还是密钥序列的生成

3.9.2 ZUC序列密码的机密性算法128-EEA3

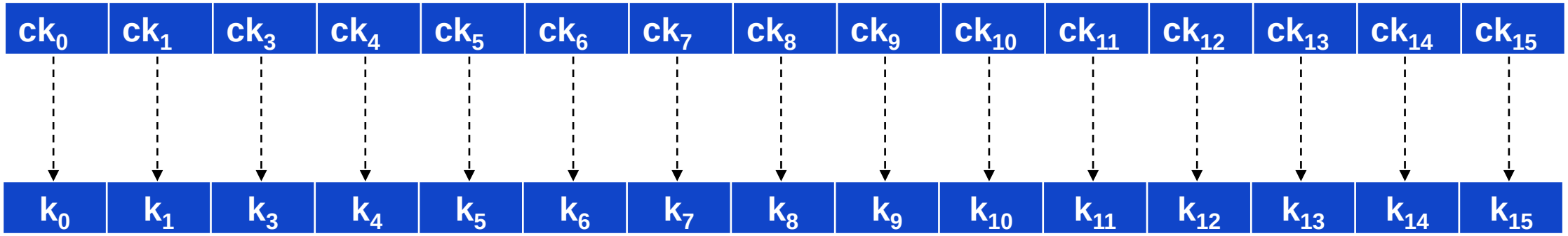
- 128位机密性密钥CK是事先获得的，分为16个字节

$$CK = ck_0 \| ck_1 \| \ldots \| ck_{15}$$

- ZUC算法的初始密钥KEY

$$KEY = k_0 \| k_1 \| \ldots \| k_{15}$$

- 最简单的方法：



3.9.2 ZUC序列密码的机密性算法128-EEA3

- IV分为16个字节

$$\mathrm{IV} = iv_0 \,||\, iv_1 \,||\, \ldots \,||\, iv_{15}$$

- 通过如下规则产生：

$$iv_0 = COUNT[0],\ iv_1 = COUNT[1]$$

$$iv_2 = COUNT[2],\ iv_3 = COUNT[3]$$

$$iv_4 = BEARER \,||\, DIRECTION \,||\, 00$$

$$iv_5 = iv_6 = iv_7 = 00000000$$

$$iv_{j+8} = iv_j\ ,\ j = 8, 9\ \ldots\ 15$$

3.9.2 ZUC序列密码的机密性算法128-EEA3

- 加解密过程
  - 最核心的部分是密钥字流的产生（ZUC算法）
  - ZUC算法每个时钟节拍产生32比特的密钥
  - 将明文/密文比特流以32比特为单位进行划分
  - 明文/密文字和密钥字进行模2加（异或）即实现加/解密

最核心的部分仍然是密钥生成

3.9.3 ZUC序列密码的安全性

- 能够抵御多种已知攻击
  - 弱密钥攻击
  - Guess-and-Determine攻击
  - Binary Decision Tree攻击
  - 线性区分攻击
  - 代数攻击
  - 选择初始向量攻击等
- 主要安全威胁在于侧信道攻击，如DPA攻击

课后阅读

After-class Reading

- 流密码和RC4，https://www.cnblogs.com/block2016/p/5601925.html
- 密码行业标准化技术委员会，http://www.gmbz.org.cn/main/bzlb.html
- GMSSL，http://gmssl.org/docs/docindex.html
- OpenSSL，https://www.openssl.org
- 作业：编程实现ZUC算法

可以从密码行业标准化技术委员会查看ZUC的算法标准


感谢聆听!
THANK YOU FOR YOUR ATTENTION!