

密码学

第七讲 祖冲之密码

王后珍

武汉大学国家网络安全学院

空天信息安全与可信计算教育部重点实验室





目录

- 第一讲 信息安全概论
- 第二讲 密码学的基本概念
- 第三讲 数据加密标准 (DES)
- 第四讲 高级数据加密标准 (AES)
- 第五讲 中国商用密码SMS4与分组密码应用技术
- 第六讲 序列密码基础
- 第七讲 祖冲之密码**
- 第八讲 中国商用密码HASH函数SM3
- 第九讲 复习





目录

- 第十讲 公钥密码基础
- 第十一讲 中国商用公钥密码SM2加密算法
- 第十二讲 数字签名基础
- 第十三讲 中国商用公钥密码SM2签名算法
- 第十四讲 密码协议
- 第十五讲 认证
- 第十六讲 密钥管理：对称密码密钥管理
- 第十七讲 密钥管理：公钥密码密钥管理
- 第十八讲 复习

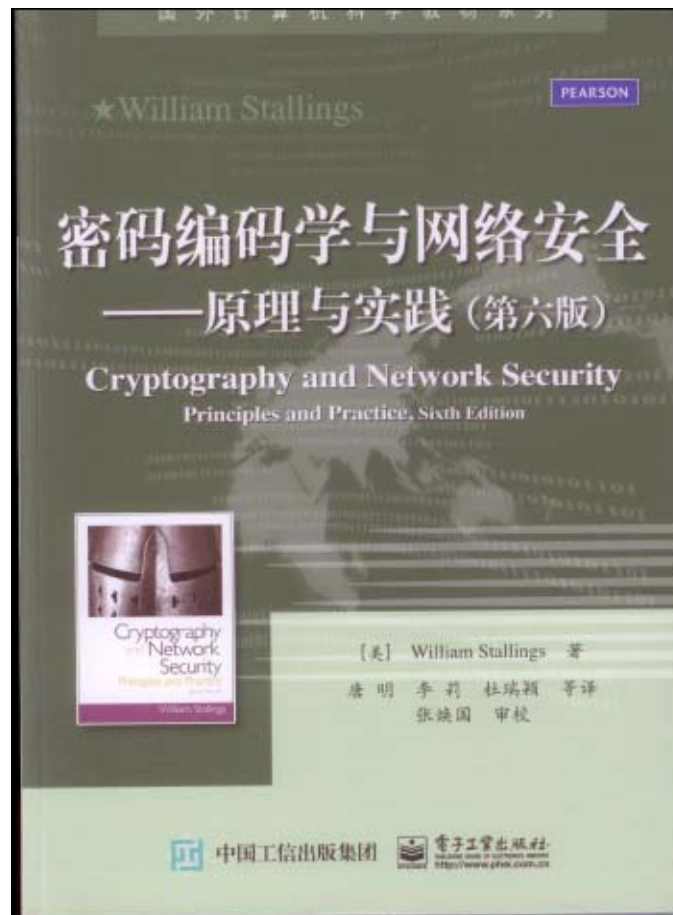


教材与主要参考书

教材



参考书



武汉大学



本讲内容

- 一、祖冲之密码概况
- 二、祖冲之密码算法（ZUC）
- 三、基于ZUC的机密性算法128-EEA3
- 四、基于ZUC的完整性算法128-EIA3
- 五、ZUC的安全性
- 六、RC4序列密码





一、祖冲之密码概况

- 2011年9月19-21日，我国设计的祖冲之密码算法（ZUC）被批准成为新一代宽带无线移动通信系统（LTE）国际标准，即4G的国际标准。
- 我国推荐的4G密码标准包括：
 - 祖冲之密码算法（ZUC）
 - 基于ZUC的机密性算法128-EEA3，用于数据保密
 - 基于ZUC的完整性算法128-EIA3，用于数据完整性保护
- 祖冲之密码
 - 面向32位字的序列密码算法

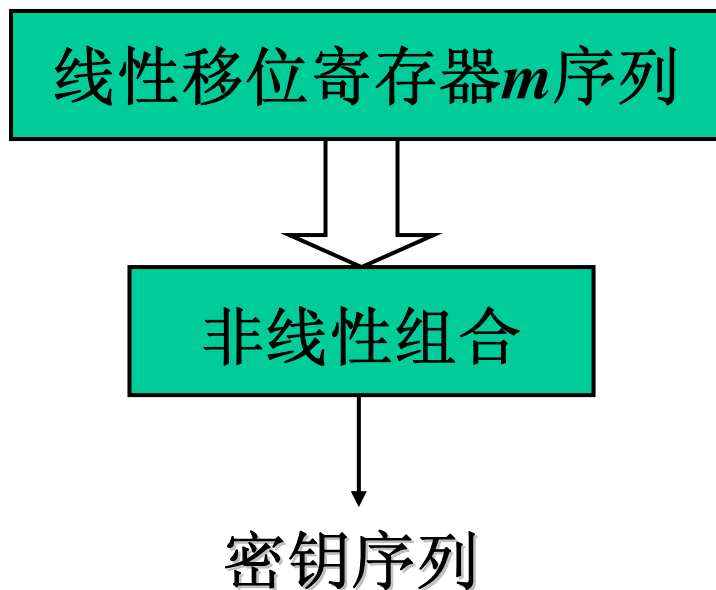




二、祖冲之密码算法 (ZUC)

● ZUC算法结构

- 本质上是一个密钥序列产生算法
- 对 $GF(2^{31}-1)$ 上的线性移位寄存器 m 序列进行非线性组合
- 结构上, 属于对一个LFSR进行非线性组合前馈电路



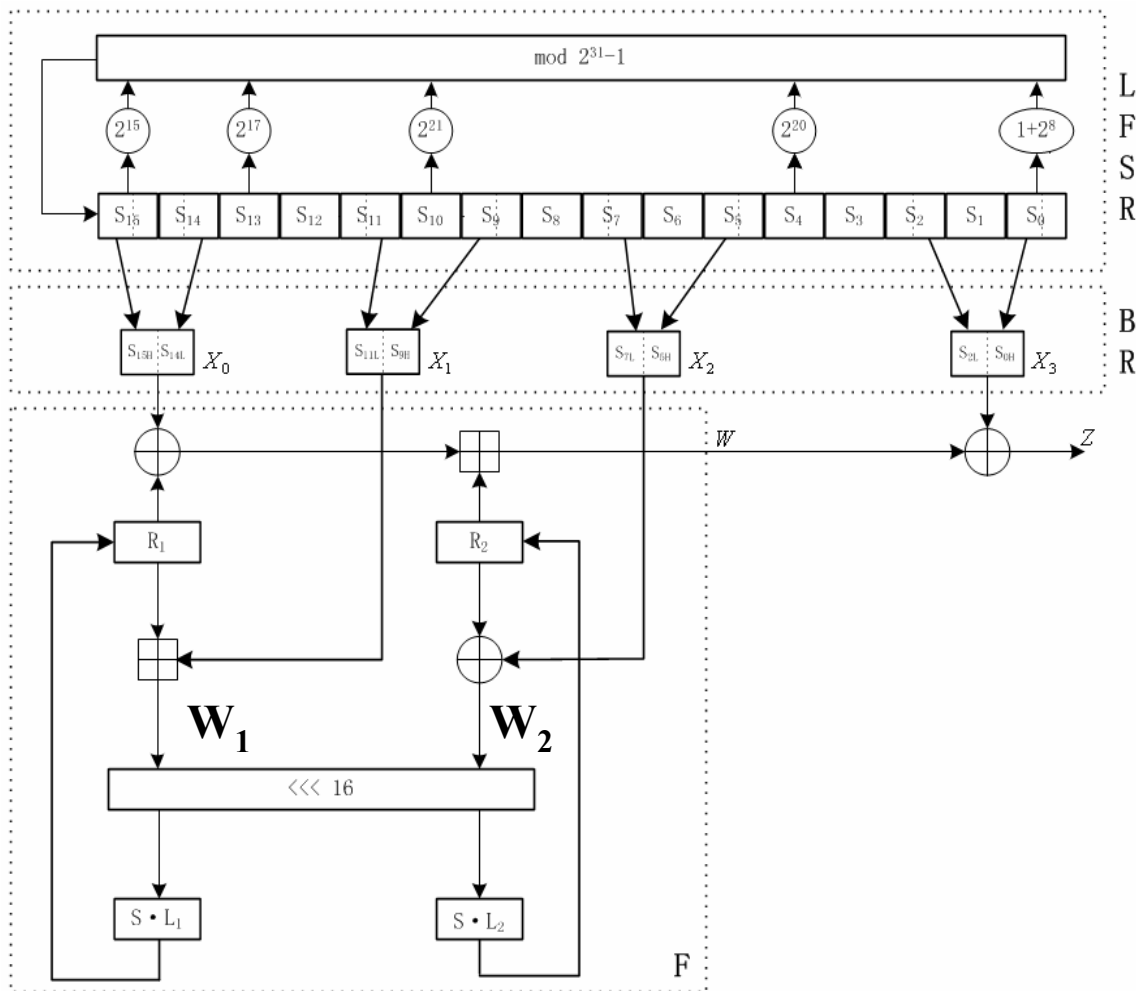
二、祖冲之密码算法 (ZUC)

● 3层结构

■ 上层是16级线性反馈移位寄存器 (LFSR)

■ 中层是比特重组 (BR)

■ 下层是非线性函数 F





二、祖冲之密码算法（ZUC）

● 上层：LFSR

- 采用 $GF(2^{31}-1)$ 上的16次本原多项式为LFSR的连接多项式：

$$\begin{aligned} p(x) &= x^{16} - 2^{15}x^{15} - 2^{17}x^{13} - 2^{21}x^{10} - 2^{20}x^4 - (2^8 + 1) \\ &= x^{16} - \{2^{15}x^{15} + 2^{17}x^{13} + 2^{21}x^{10} + 2^{20}x^4 + (2^8 + 1)\} \end{aligned}$$

- 注意： $2^{31}-1=2147483647$ 是素数， $GF(2^{31}-1)$ 是素域。

LFSR的寄存器元素是31比特的。

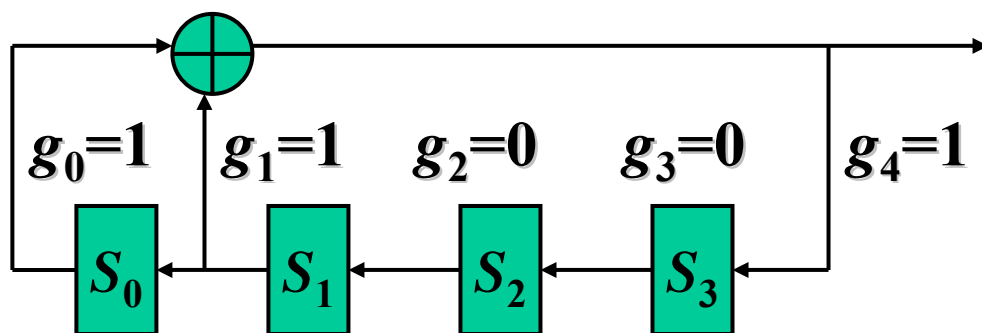
- LFSR的输出为 $GF(2^{31}-1)$ 上的 m 序列，具有良好的随机性，作为ZUC密码的驱动源。
- LFSR的输出作为中层比特重组（BR）的输入。



三、线性移位寄存器序列密码

● 回忆： $GF(2)$ 上的线性移位寄存器

■ $g(x)=x^4+x+1$



■ 工作过程:

◆ $g_0 S_0 \oplus g_1 S_1 \rightarrow v$, v 是临时工作变量

◆ $S_1 \rightarrow S_0, S_2 \rightarrow S_1, S_3 \rightarrow S_2,$

◆ $v \rightarrow S_3$





三、线性移位寄存器序列密码

● 祖冲之密码： $GF(2^{31}-1)$ 上的线性移位寄存器

$$\begin{aligned} \blacksquare g(x) &= x^{16} - 2^{15}x^{15} - 2^{17}x^{13} - 2^{21}x^{10} - 2^{20}x^4 - (2^8 + 1) \\ &= x^{16} - \{2^{15}x^{15} + 2^{17}x^{13} + 2^{21}x^{10} + 2^{20}x^4 + (2^8 + 1)\} \end{aligned}$$

$$\blacksquare g_0 = 2^8 + 1, \quad g_4 = 2^{20}, \quad g_{10} = 2^{21}, \quad g_{13} = 2^{17}, \quad g_{15} = 2^{15}$$

■ 工作过程：

◆ $2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)s_0 \bmod (2^{31}-1) \rightarrow v$, v 是临时工作变量

◆ $v \rightarrow S_{16}$ 注意： S_{16} 不在移位寄存器中, 是工作变量！

◆ $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15})$ 。 移位一次。





二、祖冲之密码算法 (ZUC)

● 上层: LFSR

- 两种工作模式: 初始化模式, 工作模式。
- 在初始化模式下, LFSR接收一个31比特字 u 。
- u 是由F输出的32位字 W , 舍弃最低位得到的。
- 初始化模式下LFSR运算如下:

LFSRWithInitialisationMode(u)

- ① $v = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1 + 2^8)s_0 \bmod (2^{31}-1);$
- ② $s_{16} = (v + u) \bmod (2^{31}-1);$
- ③ 如果 $s_{16} = 0$, 则置 $s_{16} = 2^{31}-1$;
- ④ $(s_1, s_2, \dots, s_{15}, s_{16}) \longrightarrow (s_0, s_1, \dots, s_{14}, s_{15})$ 。 移位一次。





二、祖冲之密码算法 (ZUC)

● 上层: LFSR

■ 在工作模式下, LFSR不接收任何输入, 输出 m 序列。

■ 工作模式计算过程如下:

LFSRWithWorkMode()

① $s_{16} = 2^{15}s_{15} + 2^{17}s_{13} + 2^{21}s_{10} + 2^{20}s_4 + (1+2^8)s_0 \bmod (2^{31}-1);$

② 如果 $s_{16}=0$, 则置 $s_{16}=2^{31}-1$;

③ $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15})$ 。移位一次。

■ 初始化模式和工作模式的差异:

◆ 初始化时需要引入由非线性函数F输出W通过舍弃最低位比特得到的u, 而工作模式不需要。

◆ 目的: 引入非线性函数F的输出, 使LFSR的状态随机化。



二、祖冲之密码算法（ZUC）

● 中层：比特重组BR

- 比特重组在LFSR和F之间，起到数据组合和关联的作用
- 比特重组从LFSR的寄存器单元中抽取128比特组成4个32比特字 X_0 、 X_1 、 X_2 、 X_3 。
- 把字 X_0 、 X_1 、 X_2 、 X_3 ，送给非线性函数F。
- BR是一种选择置换，属于线性变换，提供一定的扩散。
- 计算过程如下：

BitReconstruction()

- ① $X_0 = s_{15}H \parallel s_{14}L$ ；其中符号 \parallel 表示两个字符首尾拼接。
- ② $X_1 = s_{11}L \parallel s_9H$ ；其中H表示高16位，L表示低16位。
- ③ $X_2 = s_7L \parallel s_5H$ ；
- ④ $X_3 = s_2L \parallel s_0H$ 。





二、祖冲之密码算法 (ZUC)

● 下层：非线性函数F

■ 非线性函数F是一个把96比特压缩为32比特的一个非线性压缩函数。

■ 非线性函数F内部包含2个32比特存储单元 R_1 和 R_2

■ F的输入为来自比特重组的3个32比特字 X_0 、 X_1 、 X_2

■ F的输出为一个32比特字W

■ F的计算过程如下：

$F(X_0, X_1, X_2)$

① $W = (X_0 \oplus R_1) \boxplus R_2$ ；其中符号 \boxplus 表示mod 2^{32} 加法。

② $W_1 = R_1 \boxplus X_1$ ；

③ $W_2 = R_2 \oplus X_2$ ；

④ $R_1 = S(L_1(W_{1L} \parallel W_{2H}))$ ；

⑤ $R_2 = S(L_2(W_{2L} \parallel W_{1H}))$ 。





二、祖冲之密码算法 (ZUC)

● 下层：非线性函数F

■ F的核心部件：

◆ S盒：非线性部件，为ZUC提供混淆

◆ L变换：线性变换，为ZUC提供扩散

■ F使用了两个8×8的S盒：

◆ S_0 和 S_1

◆ F的处理单位是32位字，故用两个S盒处理四个字节：

$$S=(S_0, S_1, S_0, S_1)。$$

◆ 设32位字 $A=A_0 \parallel A_1 \parallel A_2 \parallel A_3$ ，则

$$S(A)=S_0(A_0) \parallel S_1(A_1) \parallel S_0(A_2) \parallel S_1(A_3)$$



二、祖冲之密码算法 (ZUC)

S_0

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	3E	72	5B	47	CA	E0	00	33	04	D1	54	98	09	B9	6D	CB
1	7B	1B	F9	32	AF	9D	6A	A5	B8	2D	FC	1D	08	53	03	90
2	4D	4E	84	99	E4	CE	D9	91	DD	B6	85	48	8B	29	6E	AC
3	CD	C1	F8	1E	73	43	69	C6	B5	BD	FD	39	63	20	D4	38
4	76	7D	B2	A7	CF	ED	57	C5	F3	2C	BB	14	21	06	55	9B
5	E3	EF	5E	31	4F	7F	5A	A4	0D	82	51	49	5F	BA	58	1C
6	4A	16	D5	17	A8	92	24	1F	8C	FF	D8	AE	2E	01	D3	AD
7	3B	4B	DA	46	EB	C9	DE	9A	8F	87	D7	3A	80	6F	2F	C8
8	B1	B4	37	F7	0A	22	13	28	7C	CC	3C	89	C7	C3	96	56
9	07	BF	7E	F0	0B	2B	97	52	35	41	79	61	A6	4C	10	FE
A	BC	26	95	88	8A	B0	A3	FB	C0	18	94	F2	E1	E5	E9	5D
B	D0	DC	11	66	64	5C	EC	59	42	75	12	F5	74	9C	AA	23
C	0E	86	AB	BE	2A	02	E7	67	E6	44	A2	6C	C2	93	9F	F1
D	F6	FA	36	D2	50	68	9E	62	71	15	3D	D6	40	C4	E2	0F
E	8E	83	77	6B	25	05	3F	0C	30	EA	70	B7	A1	E8	A9	65
F	8D	27	1A	DB	81	B3	A0	F4	45	7A	19	DF	EE	78	34	60



二、祖冲之密码算法 (ZUC)

S_1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	55	C2	63	71	3B	C8	47	86	9F	3C	DA	5B	29	AA	FD	77
1	8C	C5	94	0C	A6	1A	13	00	E3	A8	16	72	40	F9	F8	42
2	44	26	68	96	81	D9	45	3E	10	76	C6	A7	8B	39	43	E1
3	3A	B5	56	2A	C0	6D	B3	05	22	66	BF	DC	0B	FA	62	48
4	DD	20	11	06	36	C9	C1	CF	F6	27	52	BB	69	F5	D4	87
5	7F	84	4C	D2	9C	57	A4	BC	4F	9A	DF	FE	D6	8D	7A	EB
6	2B	53	D8	5C	A1	14	17	FB	23	D5	7D	30	67	73	08	09
7	EE	B7	70	3F	61	B2	19	8E	4E	E5	4B	93	8F	5D	DB	A9
8	AD	F1	AE	2E	CB	0D	FC	F4	2D	46	6E	1D	97	E8	D1	E9
9	4D	37	A5	75	5E	83	9E	AB	82	9D	B9	1C	E0	CD	49	89
A	01	B6	BD	58	24	A2	5F	38	78	99	15	90	50	B8	95	E4
B	D0	91	C7	CE	ED	0F	B4	6F	A0	CC	F0	02	4A	79	C3	DE
C	A3	EF	EA	51	E6	6B	18	EC	1B	2C	80	F7	74	E7	FF	21
D	5A	6A	54	1E	41	31	92	35	C4	33	07	0A	BA	7E	0E	34
E	88	B1	98	7C	F3	3D	60	6C	7B	CA	D3	1F	32	65	04	28
F	64	BE	85	9B	2F	59	8A	D7	B0	25	AC	AF	12	03	E2	F2





二、祖冲之密码算法 (ZUC)

● 下层：非线性函数F

■ F使用了两个L变换：

◆ L_1 和 L_2

◆ L_1 和 L_2 是32位字的线性变换

■ L_1 和 L_2 的定义如下：

◆ $L_1(X) = X \oplus (X \lll 2) \oplus (X \lll 10) \oplus (X \lll 18) \oplus (X \lll 24)$ 注意： L_1 与SM4中的L变换相同！

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

◆ 其中X为32位字；

◆ 符号 $a \lll n$ 表示把 a 循环左移 n 位。





二、祖冲之密码算法（ZUC）

● ZUC的密钥装入

- 密钥装入将128位的初始密钥KEY和128位的初始向量IV扩展为16个31比特字，作为LFSR寄存器 s_0, s_1, \dots, s_{15} 的初始状态。
 - ◆ $KEY = k_0 \parallel k_1 \parallel \dots \parallel k_{15}$
 - ◆ $IV = iv_0 \parallel iv_1 \parallel \dots \parallel iv_{15}$
 - ◆ 其中， k_i 和 iv_i 均为8比特字节， $0 \leq i \leq 15$
 - ◆ 对 $0 \leq i \leq 15$ ，置 $s_i = k_i \parallel d_i \parallel iv_i$ ， s_i 是31位字。
 - ◆ 其中 d_i 是15位的常量。





二、祖冲之密码算法 (ZUC)

● ZUC的密钥装入

■ 使用的16个15位常量 d_i

$$d_0 = 100010011010111$$

$$d_1 = 010011010111100$$

$$d_2 = 110001001101011$$

$$d_3 = 001001101011110$$

$$d_4 = 101011110001001$$

$$d_5 = 011010111100010$$

$$d_6 = 111000100110101$$

$$d_7 = 000100110101111$$

$$d_8 = 100110101111000$$

$$d_9 = 010111100010011$$

$$d_{10} = 110101111000100$$

$$d_{11} = 001101011110001$$

$$d_{12} = 101111000100110$$

$$d_{13} = 011110001001101$$

$$d_{14} = 111100010011010$$

$$d_{15} = 100011110101100$$





二、祖冲之密码算法 (ZUC)

● ZUC算法运行

■ 初始化阶段:

1. 密钥装入设置LFSR的初始状态: s_0, s_1, \dots, s_{15}
2. 置非线性函数 F 的存储单元 R_1 和 R_2 为全0。
3. 重复执行下述过程32次:
 - ① BitReconstruction(); / *产生 X_0, X_1, X_2, X_3 /
 - ② $W = F(X_0, X_1, X_2)$; / * W 舍弃最低位成为 u 送给LFSR /
 - ③ LFSRWithInitialisationMode (u). / 初始化LFSR /

■ 重复32次是为了使LFSR随机化!





二、祖冲之密码算法 (ZUC)

● ZUC算法运行

■ 工作阶段:

1. 首先执行下列过程一次, 并将 F 的输出 W 舍弃:

■ ① BitReconstruction();

■ ② $F(X_0, X_1, X_2)$; / 将 F 的输出 W 舍弃 /

■ ③ LFSRWithWorkMode()。

2. 密钥输出阶段, 每运行一个节拍, 执行下列过程一次, 并输出一个32比特的密钥字 Z :

① BitReconstruction();

② $Z = F(X_0, X_1, X_2) \oplus X_3$; / 产生32位密钥字 Z /

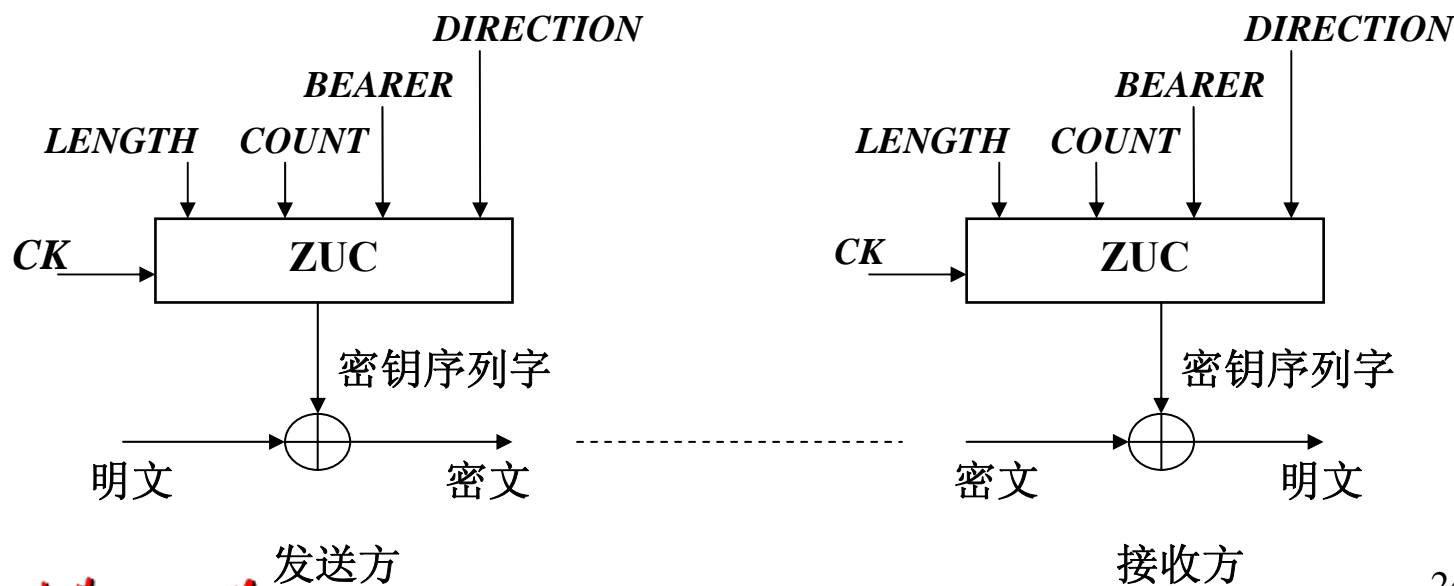
③ LFSRWithWorkMode()。



三、基于ZUC的机密性算法128-EEA3

● 用途

- 主要用于4G移动通信中移动用户设备UE和无线网络控制设备RNC之间的无线链路上通信信令和数据的加解密工作阶段：
- 加解密框图





三、基于ZUC的机密性算法128-EEA3

1. 初始化

- 根据机密性密钥 CK 以及其他输入参数构造祖冲之算法的初始密钥 KEY 和初始向量 IV 。
 - 设128位的机密性密钥为 CK ,
$$CK = CK[0] \parallel CK[1] \parallel CK[2] \parallel \dots \parallel CK[15]$$
 - 设祖冲之算法的128位初始密钥为 KEY ,
$$KEY = KEY[0] \parallel KEY[1] \parallel KEY[2] \parallel \dots \parallel KEY[15]$$
 - 直接令祖冲之算法初始密钥等于机密性密钥。于是有
$$KEY[i] = CK[i], i=0,1,2,\dots,15.$$





三、基于ZUC的机密性算法128-EEA3

1. 初始化

- 根据机密性密钥 CK 以及其他输入参数构造祖冲之算法的**初始密钥** KEY 和**初始向量** IV 。
 - 把32位的通信计数器COUNT表示为4个8位字
 $COUNT = COUNT[0] \parallel COUNT[1] \parallel COUNT[2] \parallel COUNT[3]$
 - 设祖冲之算法的128位的初始向量为 IV ,
 $IV = IV[0] \parallel IV[1] \parallel IV[2] \parallel \dots \parallel IV[15]$
 - $IV[i]$ 的产生
由于 IV 可取任何值, 为简单, 直接用通信参数产生
 - ◆ COUNT
 - ◆ DIRECTION
 - ◆ BEARER





三、基于ZUC的机密性算法128-EEA3

1. 初始化

■ IV[i]产生

$IV[0] = COUNT[0], IV[1] = COUNT[1],$

$IV[2] = COUNT[2], IV[3] = COUNT[3],$

$IV[4] = BEARER \parallel \text{DIRECTION} \parallel 00,$

$IV[5] = IV[6] = IV[7] = 00000000,$

$IV[8] = IV[0], IV[9] = IV[1],$

$IV[10] = IV[2], IV[11] = IV[3],$

$IV[12] = IV[4], IV[13] = IV[5],$

$IV[14] = IV[6], IV[15] = IV[7].$





三、基于ZUC的机密性算法128-EEA3

2. 产生加解密密钥流

- 设要加解密的数据流长度为LENGTH比特，所以祖冲之密码必须产生L个32位字的加解密密钥，

$$L = \lceil \text{LENGTH} / 32 \rceil$$

- 利用初始密钥KEY和初始向量IV，执行祖冲之密码算法便可产生L个32位字的加解密密钥流，表示成比特串

$$k[0], k[1], \dots, k[32 \times L - 1]$$

3. 加解密

- 设输入比特流为：

$$\text{IBS} = \text{IBS}[0] \parallel \text{IBS}[1] \parallel \text{IBS}[2] \parallel \dots \parallel \text{IBS}[\text{LENGTH}-1]$$

- 输出比特流为：

$$\text{OBS} = \text{OBS}[0] \parallel \text{OBS}[1] \parallel \text{OBS}[2] \parallel \dots \parallel \text{OBS}[\text{LENGTH}-1]$$

加解密： $\text{OBS}[i] = \text{IBS}[i] \oplus k[i], i=0,1,2,\dots,\text{LENGTH}-1$

武汉大学

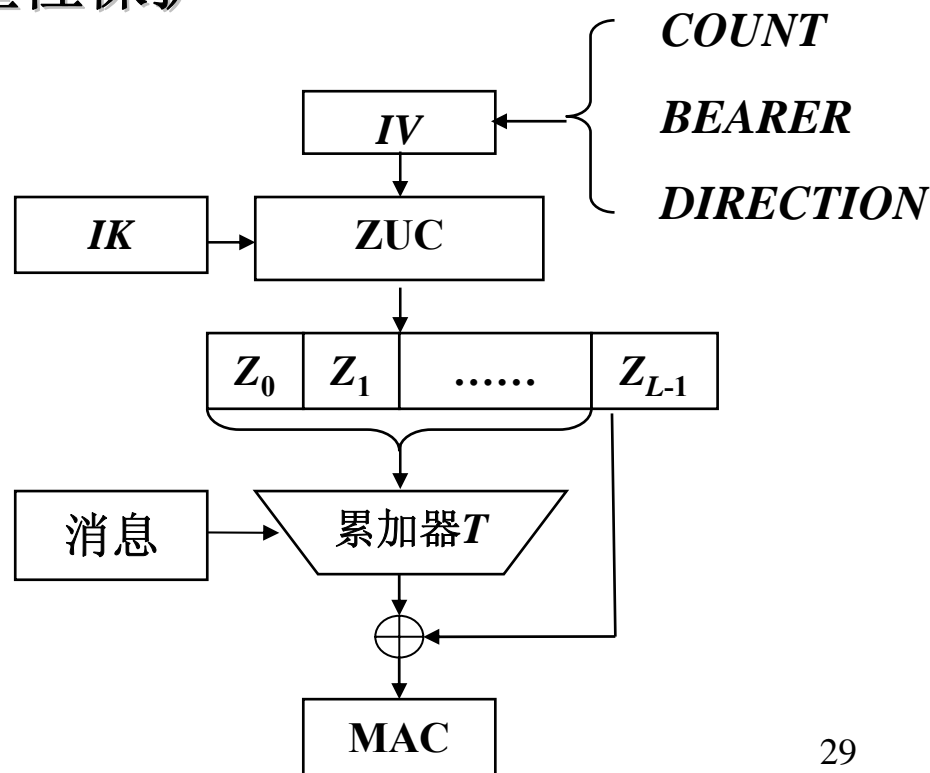


四、基于ZUC的完整性算法128-EIA3

● 用途

- 用户设备UE和无线网络控制设备RNC之间的无线链路上通信信令和数据的完整性保护

● 完整性算法框图





四、基于ZUC的完整性算法128-EIA3

1. 初始化

- 根据完整性密钥 IK 以及其他输入参数构造祖冲之算法的初始密钥 KEY 和初始向量 IV 。

- 构造祖冲之密码初始密钥 KEY

- 设128位的完整性密钥为 IK ,

$$IK = IK[0] \parallel IK[1] \parallel IK[2] \parallel \dots \parallel IK[15]$$

- 设祖冲之算法的128位初始密钥为 KEY ,

$$KEY = KEY[0] \parallel KEY[1] \parallel KEY[2] \parallel \dots \parallel KEY[15]$$

- 直接令祖冲之算法初始密钥等于完整性密钥。于是有

$$KEY[i] = IK[i], i=0,1,2,\dots,15。$$





四、基于ZUC的完整性算法128-EIA3

1. 初始化

● 构造初始向量IV

- 把32位的通信计数器COUNT表示为4个8位字节，

$COUNT = COUNT[0] \parallel COUNT[1] \parallel COUNT[2] \parallel COUNT[3]$

- 设祖冲之算法的128位的初始向量为IV，

$$IV = IV[0] \parallel IV[1] \parallel IV[2] \parallel \dots \parallel IV[15]$$

- IV[i]的产生

由于IV可取任何值，为简单，直接用通信参数产生

- ◆ COUNT
- ◆ DIRECTION
- ◆ BEARER





四、基于ZUC的完整性算法128-EIA3

1. 初始化

■ $IV[i]$ 产生

$IV[0] = COUNT[0], IV[1] = COUNT[1],$
 $IV[2] = COUNT[2], IV[3] = COUNT[3],$
 $IV[4] = BEARER \parallel 000,$
 $IV[5] = IV[6] = IV[7] = 00000000,$
 $IV[8] = IV[0] \oplus (DIRECTION \ll 7),$
 $IV[9] = IV[1],$
 $IV[10] = IV[2], IV[11] = IV[3],$
 $IV[12] = IV[4], IV[13] = IV[5],$
 $IV[14] = IV[6] \oplus (DIRECTION \ll 7),$
 $IV[15] = IV[7].$

■ 其中符号 $A \ll n$ 表示把 A 左移 n 位。





四、基于ZUC的完整性算法128-EIA3

2. 产生完整性密钥字流

- 对长度为LENGTH比特的消息计算消息认证码（MAC），所以祖冲之密码必须产生L个32位字的完整性密钥，
$$L = \lceil \text{LENGTH}/32 \rceil + 2$$
- 利用初始密钥KEY和初始向量IV，执行祖冲之密码算法便可产生L个32位字的完整性密钥流，表示成比特串

$$k[0], k[1], \dots, k[32 \times L - 1]$$

- 为了计算消息认证码（MAC），需要把比特串 $k[0], k[1], k[2], \dots, k[32 \times L - 1]$ ，重新组合成新的 $32 \times (L - 1) + 1$ 个32位新密钥字 K_i 。

$$K_i = k[i] \parallel k[i+1] \parallel \dots \parallel k[i+31]$$

$$i = 0, 1, 2, \dots, 32 \times (L - 1)$$





四、基于ZUC的完整性算法128-EIA3

3. 计算MAC

- 设要计算消息MAC的消息比特序列为

$M = m[0], m[1], \dots, m[\text{LENGTH}-1]$

- 设T为一个32比特的字变量

- **MAC的计算:**

MACComputation()

- ① 置 $T = 0$;
- ② For($I=0$; $I < \text{LENGTH}$; $I++$)
- ③ If $m[I] = 1$ Then $T = T \oplus K_i$;
- ④ For End
- ⑤ $T = T \oplus K_{\text{LENGTH}}$;
- ⑥ $\text{MAC} = T \oplus K_{32 \times (L-1)}$;






五、祖冲之算法的安全性

- LFSR采用了精心挑选的 $GF(2^{31}-1)$ 上的16次本原多项式，使其输出 m 序列随机性好、周期足够大。
- 比特重组部分，精心选用数据使得重组的数据具有良好的随机性，并且出现重复的概率足够小。
- 非线性函数F中采用了两个存储部件R、两个线性部件L和两个非线性S盒，使得其输出具有良好的非线性、混淆特性和扩散特性。
- 可抵抗常见攻击，因此是安全的。
- 理论与实验都表明ZUC经不起DPA类侧信道的攻击。
- 完整性算法128-EIA3所产生的消息认证码（MAC）只有32位，显然是太短了。





六、RC4序列密码

- RC4序列密码是美国RSA数据安全公司设计的一种序列密码。RSA公司将其收集在加密工具软件BSAFE中。最初并没有公布RC4的算法。人们通过对软件进行逆向分析得到了算法。
- 在这种情况下RSA公司于1997年公布了RC4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。它是一种基于非线性数据表变换的序列密码。
- 它以一个大的数据表为基础，对表进行非线性变换，产生非线性的密钥序列。






六、RC4序列密码

- **RC4使用256个字节的S表和两个指针 (I 和 J) 。**
- S 表的值 S_0, S_1, \dots, S_{255} 是 $0, 1, \dots, 255$ 的一个排列。
- **I 和 J 的初值为0。**
- 我们把RC4算法看成一个有限状态自动机。把 S 表和 I 、 J 指针的具体取值称为RC4的一个状态：

$$T = \langle S_0, S_1, \dots, S_{255}, I, J \rangle$$

- 对状态 T 进行非线性变换，产生出新的状态，并输出密钥序列中一个字节 k 。





六、RC4序列密码

●RC4的下一状态函数定义如下：


- (1) $I=0, J=0;$
- (2) $I=I+1 \bmod 256;$
- (3) $J=J+S[I] \bmod 256;$
- (4) 交换 $S[I]$ 和 $S[J]$ 。

●RC4的输出函数定义如下：

- (1) $h=S[I]+S[J] \bmod 256;$
- (2) $k=S[h]$ 。

■输出 k 就是产生出的密钥字节。





六、RC4序列密码

●在用RC4加解密之前，应当首先对S表初始化：

(1) 对S表进行线性填充，即令

$$S[0]=0, S[1]=1, S[2]=2, \dots, S[255]=255;$$

(2) 用密钥填充另一个256字节的R表R[0],R[1],...,R[255]，如果密钥的长度小于R表的长度，则依次重复填充，直至将R表填满。


(3) $J=0$;

(4) 对于I=0 到255重复以下操作：

① $J = (J + S[I] + R[I]) \bmod 256;$

② 交换S[I]和S[J]。





六、RC4序列密码

- 注意：对S表初始化的过程是对S表进行随机化处理的过程，只有当这一过程完成后，才能计算产生密钥字符，才能进行加解密，否则将是不安全的。
- RC4算法的优点是算法简单，高效，特别适合软件实现。
- RC4是目前应用最广的商用序列密码。





六、RC4序列密码

RC4(PLAINTEXT, K, S, R)

Initialization (K, S, R) /RC4初始化/

For(I=0; I<256; I++) /线性填充S表/

S[I]=I;

For(I=0; I<256; I++) /用种子密钥填充R表/

{ L=I mod N;

R[I]=K[L]; }


J=0;

For(I=0; I<256; I++) /用R表随机化S表/

{ J=J +S[I]+R[I] mod 256;

SWAP(S[I], S[J]); }





六、RC4序列密码

KeyStreamGeneration (I, J, S, R) /产生256个字节的密钥序列, 并加密256个明文字节/

I=0;

J=0;

WHILE (I<256)

{ J= J+S[I] mod 256;

SWAP(S[I], S[J]) ;

H=S[I]+S[J] mod 256;

KEYSTREAM[I]=S[H]; /产生密钥字节/

CIPERTEXT[I] = PLAINTEXT[I] \oplus KEYSTREAM[I] ; /
加密/

I=I+1 ;

}





谢 谢！



武汉大学