# 祖冲之序列密码算法

# 第1部分：算法描述

# **范围**

本部分描述了祖冲之序列密码算法，可用于指导祖冲之算法相关产品的研制、检测和使用。

# **术语和约定**

以下术语和约定适用于本部分。

## 

## **比特 bit**

二进制字符0和1称之为比特。

## 

## **字节 byte**

由8个比特组成的比特串称之为字节。

## 

## **字 word**

由2个以上（包含2个）比特组成的比特串称之为字。

本部分主要使用31比特字和32比特字。

## 

## **字表示 word representation**

本部分字默认采用十进制表示。当字采用其它进制表示时，总是在字的表示之前或之后添加指示符。例如，前缀0x指示该字采用十六进制表示，后缀下角标2指示该字采用二进制表示。

## 

## **高低位顺序 bit ordering**

本部分规定字的最高位总是位于字表示中的最左边，最低位总是位于字表示中的最右边。

# **符号和缩略语**

## **运算符**

+ 算术加法运算

mod 整数取余运算

⨁ 按比特位逐位异或运算

⊞ 模232加法运算

‖ 字符串连接符

**∙H** 取字的最高16比特

**∙L** 取字的最低16比特

<<<*k* 32比特字左循环移*k*位

>>*k* 32比特字右移*k*位

**a**🡪**b** 向量**a**赋值给向量**b**，即按分量逐分量赋值

## **符号**

下列符号适用于本部分：

*s*0,*s*1,*s*2,…,*s*15 线性反馈移位寄存器的16个31比特寄存器单元变量

*X*0,*X*1,*X*2,*X*3 比特重组输出的4个32比特字

*R*1, *R*2 非线性函数F的2个32比特记忆单元变量

*W* 非线性函数F输出的32比特字

Z 算法每拍输出的32比特密钥字

*k* 初始种子密钥

*iv* 初始向量

*D* 用于算法初始化的字符串常量

## **缩略语**

下列缩略语适用于本部分：

ZUC 祖冲之序列密码算法或祖冲之算法

LFSR 线性反馈移位寄存器

BR 比特重组

F 非线性函数

# **算法描述**

## **算法整体结构**

祖冲之算法逻辑上分为上中下三层，见图1。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。

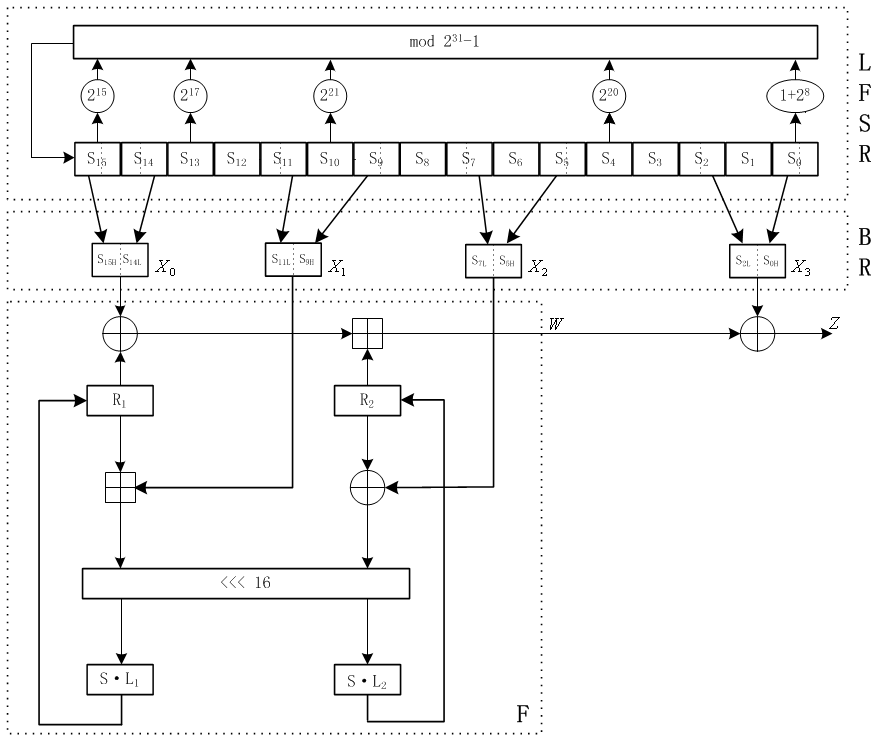


图 1 祖冲之算法结构图

## **线性反馈移位寄存器LFSR**

### **4.2.1 概述**

LFSR包括16个31比特寄存器单元变量*s*0, *s*1, …, *s*15。

LFSR的运行模式有2种：初始化模式和工作模式。

### **4.2.2 初始化模式**

在初始化模式下，LFSR接收一个31比特字*u*。*u*是由非线性函数F的32比特输出*W*通过舍弃最低位比特得到，即*u*=*W* >> 1。在初始化模式下，LFSR计算过程如下：

LFSRWithInitialisationMode(*u*)

{

1. *v* = 215 *s*15 +217 *s*13 + 221 *s*10 + 220 *s*4 + (1 + 28)*s*0 mod (231-1)；
2. *s*16=(*v*+*u*) mod (231-1)；
3. 如果*s*16=0，则置*s*16=231-1；
4. (*s*1, *s*2, …, *s*15, *s*16) → (*s*0, *s*1, …, *s*14, *s*15)。

}

### **4.2.3 工作模式**

在工作模式下，LFSR不接收任何输入。其计算过程如下：

LFSRWithWorkMode()

{

1. *s*16 = 215 *s*15 +217 *s*13 + 221 *s*10 + 220 *s*4 + (1 + 28)*s*0 mod (231-1)；
2. 如果*s*16=0，则置*s*16=231-1；
3. (*s*1, *s*2, …, *s*15, *s*16) → (*s*0, *s*1, …, *s*14, *s*15)。

}

## **比特重组BR**

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。BR的具体计算过程如下：

BitReconstruction()

{

1. *X*0 = *s*15H‖*s*14L；
2. *X*1 = *s*11L‖*s*9H；
3. *X*2 = *s*7L‖*s*5H；
4. *X*3 = *s*2L‖*s*0H。

}

## **非线性函数F**

F包含2个32比特记忆单元变量*R*1和*R*2。

F的输入为3个32比特字*X*0、*X*1、*X*2，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 ⊕ *R*1) **⊞** *R*2；
2. *W*1 = *R*1 **⊞** *X*1；
3. *W*2 = *R*2 ⊕ *X*2；
4. *R*1 = *S*(*L*1(*W*1L‖*W*2H))；
5. *R*2 = *S*(*L*2(*W*2L‖*W*1H))。

}

其中*S*为32比特的*S*盒变换，定义在附录A中给出；*L*1和*L*2为32比特线性变换，定义如下：

*L*1(*X*) = *X* ⊕ (*X* <<< 2) ⊕ (*X* <<< 10) ⊕ (*X* <<< 18) ⊕ (*X* <<< 24)，

*L*2(*X*) = *X* ⊕ (*X* <<< 8) ⊕ (*X* <<< 14) ⊕ (*X* <<< 22) ⊕ (*X* <<< 30)。

## **密钥装入**

密钥装入过程将128比特的初始密钥*k*和128比特的初始向量*iv*扩展为16个31比特字作为LFSR寄存器单元变量*s*0, *s*1, …, *s*15的初始状态。设*k*和*iv*分别为

*k*0‖*k*1‖……‖*k*15

和

*iv*0‖*iv* 1‖……‖*iv* 15，

其中*k*i和*iv*i均为8比特字节，0≤*i*≤15。密钥装入过程如下：

1. D为240比特的常量，可按如下方式分成16个15比特的子串：

*D* =*d*0‖*d*1‖……‖*d*15，

其中：

*d*0 = 1000100110101112，

*d*1= 0100110101111002，

*d*2 = 1100010011010112，

*d*3 = 0010011010111102，

*d*4= 1010111100010012，

*d*5 = 0110101111000102，

*d*6 = 1110001001101012，

*d*7 = 0001001101011112，

*d*8 = 1001101011110002，

*d*9 = 0101111000100112，

*d*10 = 1101011110001002，

*d*11= 0011010111100012，

*d*12 = 1011110001001102，

*d*13 = 0111100010011012，

*d*14 = 1111000100110102，

*d*15 = 1000111101011002。

1. 对0≤i≤15，有 *s*i = *k*i‖*d*i‖*iv*i。

## **算法运行**

### **4.6.1 初始化阶段**

首先把128比特的初始密钥*k*和128比特的初始向量*iv*按照4.5节密钥装入方法装入到LFSR的寄存器单元变量*s*0, *s*1, …, *s*15中，作为LFSR的初态，并置32比特记忆单元变量*R*1和*R*2为全0。然后执行下述操作：

重复执行下述过程32次：

1. BitReconstruction()；
2. *W*= *F*(*X*0, *X*1, *X*2)；
3. LFSRWithInitialisationMode (*W* >> 1)。

### **4.6.2 工作阶段**

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)；
3. LFSRWithWorkMode()。

然后进入密钥输出阶段。在密钥输出阶段，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 (X0, X1, X2) ⊕ X3；
3. LFSRWithWorkMode()。

（规范性附录）

**S盒**

32比特S盒*S*由4个小的8х8的S盒并置而成，即*S*=(*S*0, *S*1, *S*2, *S*3)，其中*S*0=*S*2，*S*1=*S*3。*S*0和*S*1的定义分别见表1和表2。设*S*0(或*S*1)的8比特输入为*x*。将*x*视作两个16进制数的连接，即*x*=*h*||*l*，则表1 (或表2)中第*h*行和第*l*列交叉的元素即为*S*0(或*S*1)的输出*S*0(*x*)(或*S*1(*x*))。

设S盒*S*的32比特输入X和32比特输出Y分别为：

X = *x*0 ‖ *x*1 ‖ *x*2 ‖ *x*3，

Y = *y*0 ‖ *y*1 ‖ *y*2 ‖ *y*3，

其中*x*i和*y*i均为8比特字节，*i* = 0, 1, 2, 3。则有*y*i = *Si*(*x*i), *i* = 0, 1, 2, 3。

表1 ***S*0盒**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

表2 ***S*1盒**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

注：*S*0盒和*S*1盒数据均为十六进制表示。

（资料性附录）

**模231-1乘法和模231-1加法的实现**

## **B.1 模231-1乘法**

两个31比特字模231-1乘法可以快速实现。特别地，当其中一个字具有较低的汉明重量时，可以通过31比特的循环移位运算和模231-1加法运算实现。例如，计算*ab* mod(231-1)，其中*b*=2*i*+2*j*+2*k*。则

*ab* mod(231-1) = (*a* <<<31 *i*) + (*a* <<<31 *j*) +(*a* <<<31 *k*) mod(231-1)，

其中<<<31表示31比特左循环移位运算。

## **B.2 模231-1加法**

在32位处理平台上，两个31比特字*a*和*b*模231-1加法运算*c* =*a* + *b* mod(231-1)可以通过下面的两步计算实现：

1. *c* = *a* + *b*；
2. *c* = (*c* & 0x7FFFFFFF) + (*c* >> 31)。

（资料性附录）

**算法计算实例**

## **C.1 测试向量1（全0）**

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:

z1: 27bede74

z2: 018082da

初始化：

线性反馈移位寄存器初态:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | S0+i | S1+i | S2+i | S3+i | S4+i | S5+i | S6+i | S7+i |
| 0 | 0044d700 | 0026bc00 | 00626b00 | 00135e00 | 00578900 | 0035e200 | 00713500 | 0009af00 |
| 8 | 004d7800 | 002f1300 | 006bc400 | 001af100 | 005e2600 | 003c4d00 | 00789a00 | 0047ac00 |
| t | X0 | X1 | X2 | X3 | R1 | R2 | W | S15 |
| 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 |
| 3 | e8c92a0f | 9a0000bc | c400009a | e2000026 | 29c272f3 | 8cac7f5d | 141698fb | 3f5644ba |
| 4 | 7eacf744 | ac000078 | f100005e | 350000af | 2c85a655 | 24259cb0 | e41b0514 | 006a144c |
| 5 | 00d444ba | cb1b00f1 | 260000d7 | af00006b | cbfbc5c0 | 44c10b3a | 50777f9f | 07038b9b |
| 6 | 0e07144c | 2a0f008f | 4d000035 | 780000e2 | e083c8d3 | 7abf7679 | 0abddcc6 | 69b90e2b |
| 7 | d3728b9b | f7448ac7 | 9a0000bc | 13000013 | 147e14f4 | b669e72d | aeb0b9c1 | 62a913ea |
| 8 | c5520e2b | 44ba50ca | ac000078 | c400009a | 982834a0 | f095d694 | 8796020c | 7b591cc0 |
| 9 | f6b213ea | 144ce8c9 | cb1b00f1 | f100005e | e14727d6 | d0225869 | 5f2ffdde | 70e21147 |

初始化后线性反馈移位寄存器状态:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | S0+i | S1+i | S2+i | S3+i | S4+i | S5+i | S6+i | S7+i |
| 0 | 7ce15b8b | 747ca0c4 | 6259dd0b | 47a94c2b | 3a89c82e | 32b433fc | 231ea13f | 31711e42 |
| 8 | 4ccce955 | 3fb6071e | 161d3512 | 7114b136 | 5154d452 | 78c69a74 | 4f26ba6b | 3e1b8d6a |

有限状态机内部状态:

R1 = 14cfd44c

R2 = 8c6de800

密钥流：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | X0 | X1 | X2 | X3 | R1 | R2 | z | S15 |
| 0 | 7c37ba6b | b1367f6c | 1e426568 | dd0bf9c2 | 3512bf50 | a0920453 | 286dafe5 | 7f08e141 |
| 1 | fe118d6a | d4522c3a | e955463d | 4c2be8f9 | c7ee7f13 | 0c0fa817 | 27bede74 | 3d383d04 |
| 2 | 7a70e141 | 9a74e229 | 071e62e2 | c82ec4b3 | dde63da7 | b9dd6a41 | 018082da | 13d6d780 |

## **C.2 测试向量 2（全1）**

输入:

密钥k: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

初始向量iv: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

输出:

z1: 0657cfa0

z2: 7096398b

初始化：

线性反馈移位寄存器初态:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | S0+i | S1+i | S2+i | S3+i | S4+i | S5+i | S6+i | S7+i |
| 0 | 7fc4d7ff | 7fa6bcff | 7fe26bff | 7f935eff | 7fd789ff | 7fb5e2ff | 7ff135ff | 7f89afff |
| 8 | 7fcd78ff | 7faf13ff | 7febc4ff | 7f9af1ff | 7fde26ff | 7fbc4dff | 7ff89aff | 7fc7acff |
| t | X0 | X1 | X2 | X3 | R1 | R2 | W | S15 |
| 0 | ff8f9aff | f1ffff5e | afffff6b | 6bffff89 | b51c2110 | 30a3629a | ff8f9aff | 76e49a1a |
| 1 | edc9acff | 26ffffd7 | 78ffffe2 | 5effff4d | a75b6f4b | 1a079628 | 8978f089 | 5e2d8983 |
| 2 | bc5b9a1a | 4dffff35 | 13ffff13 | 89ffffc4 | 9810b315 | 99296735 | 35088b79 | 5b9484b8 |
| 3 | b7298983 | 9affffbc | c4ffff9a | e2ffff26 | 4c5bd8eb | 2d577790 | c862a1cb | 2db5c755 |
| 4 | 5b6b84b8 | acffff78 | f1ffff5e | 35ffffaf | a13dcb66 | 21d0939f | 4487d3e3 | 60579232 |
| 5 | c0afc755 | 9a1afff1 | 26ffffd7 | afffff6b | cc5ce260 | 0c50a8e2 | 83629fd2 | 29d4e960 |
| 6 | 53a99232 | 8983ff8f | 4dffff35 | 78ffffe2 | dada0730 | b516b128 | ac461934 | 5e02d9e5 |
| 7 | bc05e960 | 84b8edc9 | 9affffbc | 13ffff13 | 2bbe53a4 | 12a8a16e | 1bf69f78 | 7904dddc |
| 8 | f209d9e5 | c755bc5b | acffff78 | c4ffff9a | 4a90d661 | d9c744b4 | ec602baf | 0c3c9016 |
| 9 | 1879dddc | 9232b729 | 9a1afff1 | f1ffff5e | 76bc13d7 | a49ea404 | 2cb05071 | 0b9d257b |

初始化后线性反馈移位寄存器状态:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | S0+i | S1+i | S2+i | S3+i | S4+i | S5+i | S6+i | S7+i |
| 0 | 09a339ad | 1291d190 | 25554227 | 36c09187 | 0697773b | 443cf9cd | 6a4cd899 | 49e34bd0 |
| 8 | 56130b14 | 20e8f24c | 7a5b1dcc | 0c3cc2d1 | 1cc082c8 | 7f5904a2 | 55b61ce8 | 1fe46106 |

有限状态机内部状态:

R1 = b8017bd5

R2 = 9ce2de5c

密钥流：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | X0 | X1 | X2 | X3 | R1 | R2 | z | S15 |
| 0 | 3fc81ce8 | c2d141d1 | 4bd08879 | 42271346 | aa131b11 | 09d7706c | 668b56df | 13f56dbf |
| 1 | 27ea6106 | 82c8f4b6 | 0b14d499 | 91872523 | 251e7804 | caac5d66 | 0657cfa0 | 0c0fe353 |
| 2 | 181f6dbf | 04a21879 | f24c93c6 | 773b4aaa | d94e9228 | 91d88fba | 7096398b | 10f1eecf |

## **C.3 测试向量3（随机）**

输入:

密钥k: 3d 4c 4b e9 6a 82 fd ae b5 8f 64 1d b1 7b 45 5b

初始向量iv: 84 31 9a a8 de 69 15 ca 1f 6b da 6b fb d8 c7 66

输出:

z1: 14f1c272

z2: 3279c419

初始化：

线性反馈移位寄存器初态:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | S0+i | S1+i | S2+i | S3+i | S4+i | S5+i | S6+i | S7+i |
| 0 | 1ec4d784 | 2626bc31 | 25e26b9a | 74935ea8 | 355789de | 4135e269 | 7ef13515 | 5709afca |
| 8 | 5acd781f | 47af136b | 326bc4da | 0e9af16b | 58de26fb | 3dbc4dd8 | 22f89ac7 | 2dc7ac66 |
| t | X0 | X1 | X2 | X3 | R1 | R2 | W | S15 |
| 0 | 5b8f9ac7 | f16b8f5e | afca826b | 6b9a3d89 | 9c62829f | 5df00831 | 5b8f9ac7 | 3c7b93c0 |
| 1 | 78f7ac66 | 26fb64d7 | 781ffde2 | 5ea84c4d | 3d533f3a | 80ff1faf | 4285372a | 41901ee9 |
| 2 | 832093c0 | 4dd81d35 | 136bae13 | 89de4bc4 | 2ca57e9d | d1db72f9 | 3f72cca9 | 411efa99 |
| 3 | 823d1ee9 | 9ac7b1bc | c4dab59a | e269e926 | 0e8dc40f | 60921a4f | 8073d36d | 24b3f49f |
| 4 | 4967fa99 | ac667b78 | f16b8f5e | 35156aaf | 16c81467 | da8e7d8a | a87c58e5 | 74265785 |
| 5 | e84cf49f | 93c045f1 | 26fb64d7 | afca826b | 50c9eaa4 | 3c3b2dfd | d9135e82 | 481c5b9d |
| 6 | 90385785 | 1ee95b8f | 4dd81d35 | 781ffde2 | 59857b80 | be0fbdc1 | fd2ceb1e | 4b7f87ed |
| 7 | 96ff5b9d | fa9978f7 | 9ac7b1bc | 136bae13 | 9528f8ea | bcc7f7eb | 8d89ddde | 0e633ce7 |
| 8 | 1cc687ed | f49f8320 | ac667b78 | c4dab59a | c59d2932 | e1098a64 | 46b676f2 | 643ae5a6 |
| 9 | c8753ce7 | 5785823d | 93c045f1 | f16b8f5e | 755ebae8 | 3f9e6e86 | eef1a039 | 625ac5d7 |

初始化后线性反馈移位寄存器状态:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | S0+i | S1+i | S2+i | S3+i | S4+i | S5+i | S6+i | S7+i |
| 0 | 10da5941 | 5b6acbf6 | 17060ce1 | 35368174 | 5cf4385a | 479943df | 2753bab2 | 73775d6a |
| 8 | 43930a37 | 77b4af31 | 15b2e89f | 24ff6e20 | 740c40b9 | 026a5503 | 194b2a57 | 7a9a1cff |

有限状态机内部状态:

R1 = 860a7dfa

R2 = bf0e0ffc

密钥流：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | X0 | X1 | X2 | X3 | R1 | R2 | z | S15 |
| 0 | f5342a57 | 6e20ef69 | 5d6a8f32 | 0ce121b4 | 129d8b39 | 2d7cdce1 | 3ead461d | 3d4aa9e7 |
| 1 | 7a951cff | 40b92b65 | 0a374ea7 | 8174b6d5 | ab7cf688 | c1598aa6 | 14f1c272 | 71db1828 |
| 2 | e3b6a9e7 | 550349fe | af31e6ee | 385a2e0c | 3cec1a4a | 9053cc0e | 3279c419 | 258937da |

注：上述祖冲之算法计算实例中数据全部采用十六进制表示。

————————————

# **参考文献**

1. ETSI/SAGE TS 35.221. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification.
2. ETSI/SAGE TS 35.222. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification.
3. ETSI/SAGE TS 35.223. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 3: Implementor's Test Data.
4. ETSI/SAGE TR 35.924. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 4: Design and Evaluation Report.

# 祖冲之序列密码算法

# 第2部分：基于祖冲之算法的机密性算法

# 1 范围

本部分描述了基于祖冲之算法的机密性算法。该机密性算法可适用于3GPP LTE通信中的加密和解密。本部分可用于指导基于祖冲之算法的机密性算法的相关产品的研制、检测和使用。

# 2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注明日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GM/T 0001.1 – 2012 祖冲之序列密码算法 第1部分：算法描述

# **术语和约定**

以下术语和约定适用于本部分。

## **比特 bit**

二进制字符0和1称之为比特。

## **字节 byte**

由8个比特组成的比特串称之为字节。

## **字 word**

由2个以上（包含2个）比特组成的比特串称之为字。

本部分主要使用31比特字和32比特字。

## 

## **字表示 word representation**

本部分字默认采用十进制表示。当字采用其它进制表示时，总是在字的表示之前或之后添加指示符。例如，前缀0x指示该字采用十六进制表示，后缀下角标2指示该字采用二进制表示。

## 

## **高低位顺序 bit ordering**

本部分规定字的最高位总是位于字表示中的最左边，最低位总是位于字表示中的最右边。

# 4 符号和缩略语

## 4.1 符号

下列符号适用于本部分：

* 按比特位逐位异或运算

*a*‖*b* 字符串连接符

 不小于的最小整数

## 4.2 缩略语

下列缩略语适用于本部分：

CK 基于祖冲之算法的机密性算法密钥

KEY 祖冲之算法的初始密钥

IV 祖冲之算法的初始向量

IBS 输入比特流

OBS 输出比特流

# 5 算法描述

## 5.1 算法输入与输出

本算法的输入参数见表1，输出参数见表2。

**表1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 | 传输方向标识 |
| **CK** | 128 | 机密性密钥 |
| **LENGTH** | 32 | 明文消息流的比特长度 |
| **IBS** | **LENGTH** | 输入比特流 |

**表2输出参数表**

|  |  |  |
| --- | --- | --- |
| **输出参数** | **比特长度** | **备注** |
| **OBS** | **LENGTH** | 输出比特流 |

## 5.2 算法工作流程

### 5.2.1 初始化

本算法的初始化是指根据机密性密钥**CK**以及其它输入参数（参见5.1节表1）构造祖冲之算法的初始密钥**KEY**和初始向量**IV**。

记机密性密钥

**CK = CK**[0]∥**CK**[1]∥**CK**[2]∥**…**∥**CK**[15]

和祖冲之算法的初始密钥

**KEY** = **KEY**[0]∥**KEY**[1]∥**KEY**[2]∥…∥**KEY**[15]，

其中**CK**[i]、**KEY** [i](0≤i≤15)都是8比特的字节。则有

**KEY**[i] = **CK**[i], i=0,1,2,…,15。

记计数器

**COUNT**=**COUNT**[0]∥**COUNT**[1]∥**COUNT**[2]∥**COUNT**[3]

和祖冲之算法的初始向量

**IV** = **IV**[0]∥**IV**[1]∥**IV**[2]∥…∥**IV**[15]，

其中**COUNT**[0]、**COUNT**[1]、**COUNT**[2]、**COUNT**[3]和**IV**[0]、**IV**[1]、…、**IV**[15]都是8比特的字节。则有

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

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

**IV**[4] = **BEARER**∥**DIRECTION**∥002，

**IV**[5] = **IV**[6] = **IV**[7] = 000000002，

**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]。

### 5.2.2 产生密钥流

利用5.2.1节生成的初始密钥**KEY**和初始向量**IV**，祖冲之算法产生L个字的密钥流。将生成的密钥流用比特串表示为**k**[0]，**k**[1]，…，**k**[32\*L-1]，其中**k**[0]为祖冲之算法生成的第一个密钥字的最高位比特，**k**[31]为最低位比特，其它依此类推。为了处理**LENGTH**比特的输入比特流，L的取值为L=⎡**LENGTH**/32⎤。

### 5.2.3 加解密

设长度为**LENGTH**的输入比特流为

**IBS**= **IBS**[0]∥**IBS**[1]∥**IBS**[2]∥…∥**IBS**[**LENGTH**-1]，

对应的输出比特流为

**OBS**= **OBS**[0]∥**OBS**[1]∥**OBS**[2]∥…∥**OBS**[**LENGTH**-1]，

其中**IBS**[i]和**OBS**[i]均为比特，i=0，1，2，……，LENGTH-1。则有

**OBS**[i]= **IBS**[i]⨁**k**[i]，i=0,1,2,…,LENGTH-1。

（资料性附录）

算法计算实例

以下为本算法的计算实例。数据采用16进制表示。

第一组加密实例：

**CK** = 17 3d 14 ba 50 03 73 1d 7a 60 04 94 70 f0 0a 29

**COUNT** = 66035492

**BEARER** = f

**DIRECTION** = 0

**LENGTH** = c1

**IBS**:

6cf65340 735552ab 0c9752fa 6f9025fe 0bd675d9 005875b2 00000000

**OBS**:

a6c85fc6 6afb8533 aafc2518 dfe78494 0ee1e4b0 30238cc8 00000000

第二组加密实例：

**CK** = e5 bd 3e a0 eb 55 ad e8 66 c6 ac 58 bd 54 30 2a

**COUNT** = 56823

**BEARER** = 18

**DIRECTION** = 1

**LENGTH** = 320

**IBS**:

14a8ef69 3d678507 bbe7270a 7f67ff50 06c3525b 9807e467 c4e56000 ba338f5d 42955903 67518222 46c80d3b 38f07f4b e2d8ff58 05f51322 29bde93b bbdcaf38 2bf1ee97 2fbf9977 bada8945 847a2a6c 9ad34a66 7554e04d 1f7fa2c3 3241bd8f 01ba220d

**OBS**:

131d43e0 dea1be5c 5a1bfd97 1d852cbf 712d7b4f 57961fea 3208afa8 bca433f4 56ad09c7 417e58bc 69cf8866 d1353f74 865e8078 1d202dfb 3ecff7fc bc3b190f e82a204e d0e350fc 0f6f2613 b2f2bca6 df5a473a 57a4a00d 985ebad8 80d6f238 64a07b01

第三组加密实例：

**CK** = e1 3f ed 21 b4 6e 4e 7e c3 12 53 b2 bb 17 b3 e0

**COUNT** = 2738cdaa

**BEARER** = 1a

**DIRECTION** = 0

**LENGTH** = FB3

**IBS**:

8d74e20d 54894e06 d3cb13cb 3933065e 8674be62 adb1c72b 3a646965 ab63cb7b 7854dfdc 27e84929 f49c64b8 72a490b1 3f957b64 827e71f4 1fbd4269 a42c97f8 24537027 f86e9f4a d82d1df4 51690fdd 98b6d03f 3a0ebe3a 312d6b84 0ba5a182 0b2a2c97 09c090d2 45ed267c f845ae41 fa975d33 33ac3009 fd40eba9 eb5b8857 14b768b6 97138baf 21380eca 49f644d4 8689e421 5760b906 739f0d2b 3f091133 ca15d981 cbe401ba f72d05ac e05cccb2 d297f4ef 6a5f58d9 1246cfa7 7215b892 ab441d52 78452795 ccb7f5d7 9057a1c4 f77f80d4 6db2033c b79bedf8 e60551ce 10c667f6 2a97abaf abbcd677 2018df96 a282ea73 7ce2cb33 1211f60d 5354ce78 f9918d9c 206ca042 c9b62387 dd709604 a50af16d 8d35a890 6be484cf 2e74a928 99403643 53249b27 b4c9ae29 eddfc7da 6418791a 4e7baa06 60fa6451 1f2d685c c3a5ff70 e0d2b742 92e3b8a0 cd6b04b1 c790b8ea d2703708 540dea2f c09c3da7 70f65449 e84d817a 4f551055 e19ab850 18a0028b 71a144d9 6791e9a3 57793350 4eee0060 340c69d2 74e1bf9d 805dcbcc 1a6faa97 6800b6ff 2b671dc4 63652fa8 a33ee509 74c1c21b e01eabb2 16743026 9d72ee51 1c9dde30 797c9a25 d86ce74f 5b961be5 fdfb6807 814039e7 137636bd 1d7fa9e0 9efd2007 505906a5 ac45dfde ed7757bb ee745749 c2963335 0bee0ea6 f409df45 80160000

**OBS**：

94eaa4aa 30a57137 ddf09b97 b25618a2 0a13e2f1 0fa5bf81 61a879cc 2ae797a6 b4cf2d9d f31debb9 905ccfec 97de605d 21c61ab8 531b7f3c 9da5f039 31f8a064 2de48211 f5f52ffe a10f392a 04766998 5da454a2 8f080961 a6c2b62d aa17f33c d60a4971 f48d2d90 9394a55f 48117ace 43d708e6 b77d3dc4 6d8bc017 d4d1abb7 7b7428c0 42b06f2f 99d8d07c 9879d996 00127a31 985f1099 bbd7d6c1 519ede8f 5eeb4a61 0b349ac0 1ea23506 91756bd1 05c974a5 3eddb35d 1d4100b0 12e522ab 41f4c5f2 fde76b59 cb8b96d8 85cfe408 0d1328a0 d636cc0e dc05800b 76acca8f ef672084 d1f52a8b bd8e0993 320992c7 ffbae17c 408441e0 ee883fc8 a8b05e22 f5ff7f8d 1b48c74c 468c467a 028f09fd 7ce91109 a570a2d5 c4d5f4fa 18c5dd3e 4562afe2 4ef77190 1f59af64 5898acef 088abae0 7e92d52e b2de5504 5bb1b7c4 164ef2d7 a6cac15e eb926d7e a2f08b66 e1f759f3 aee44614 725aa3c7 482b3084 4c143ff8 5b53f1e5 83c50125 7dddd096 b81268da a303f172 34c23335 41f0bb8e 190648c5 807c866d 71932286 09adb948 686f7de2 94a802cc 38f7fe52 08f5ea31 96d0167b 9bdd02f0 d2a5221c a508f893 af5c4b4b b9f4f520 fd84289b 3dbe7e61 497a7e2a 584037ea 637b6981 127174af 57b471df 4b2768fd 79c1540f b3edf2ea 22cb69be c0cf8d93 3d9c6fdd 645e8505 91cca3d6 2c0cc000

# 参考文献

1. ETSI/SAGE TS 35.221. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification.
2. ETSI/SAGE TS 35.222. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification.
3. ETSI/SAGE TS 35.223. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 3: Implementor's Test Data.
4. ETSI/SAGE TR 35.924. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 4: Design and Evaluation Report.

# 祖冲之序列密码算法

# 第3部分：基于祖冲之算法的完整性算法

## 1 范围

本部分描述了基于祖冲之算法的完整性算法。该完整性算法可适用于3GPP LTE通信中消息的完整性保护。本部分可用于指导基于祖冲之算法的完整性算法相关产品的研制、检测和使用。

# 2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注明日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GM/T 0001.1 – 2011 祖冲之序列密码算法 第1部分：算法描述

# 3 术语和约定

以下术语和约定适用于本部分。

## **比特 bit**

二进制字符0和1称之为比特。

## **字节 byte**

由8个比特组成的比特串称之为字节。

## **字 word**

由2个以上（包含2个）比特组成的比特串称之为字。

本部分主要使用31比特字和32比特字。

## 

## **字表示 word representation**

本部分字默认采用十进制表示。当字采用其它进制表示时，总是在字的表示之前或之后添加指示符。例如，前缀0x指示该字采用十六进制表示，后缀下角标2指示该字采用二进制表示。

## 

## **高低位顺序 bit ordering**

本部分规定字的最高位总是位于字表示中的最左边，最低位总是位于字表示中的最右边。

# 4 符号和缩略语

## 4.1 符号

下列符合适用于本部分：

* 异或运算

*a*‖*b* 字符串连接符

 不小于的最小整数

<<*k* 左移*k*位

## 4.2 缩略语

下列缩略语适用于本规范：

**IK** 基于祖冲之算法的完整性算法密钥

**KEY** 祖冲之算法的初始密钥

**IV** 祖冲之算法的初始向量

**MAC** 消息认证码

# 5 算法描述

## 5.1 算法输入与输出

本算法的输入参数见表1，输出参数见表2。

**表1 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 | 传输方向标识 |
| **IK** | 128 | 完整性密钥 |
| **LENGTH** | 32 | 输入消息流的比特长度 |
| **M** | **LENGTH** | 输入消息流 |

**表2 输出参数表**

|  |  |  |
| --- | --- | --- |
| **输出参数** | **比特长度** | **备注** |
| **MAC** | 32 | 消息认证码 |

## 5.2 算法工作流程

### 5.2.1 初始化

本算法的初始化主要是指根据完整性密钥**IK**和其他输入参数（见5.1节表1）构造祖冲之算法的初始密钥**KEY**和初始向量**IV**。

记完整性密钥

**IK** = **IK[0]**∥**IK[1]**∥**IK[2]**∥**…**∥**IK[15]**

和祖冲之算法的初始化密钥

**KEY** = **KEY**[0]∥**KEY**[1]∥**KEY**[2]∥…∥**KEY**[15]，

其中**IK**[i]、**KEY** [i](0≤i≤15)都是8比特的字节。则有

**KEY**[i] = **IK**[i], i=0,1,2,…,15。

记计数器

**COUNT**=**COUNT**[0]∥**COUNT**[1]∥**COUNT**[2]∥**COUNT**[3]，

其中**COUNT**[i]为8比特的字节，i＝0,1,2,3。设祖冲之算法的初始向量**IV**为

**IV = IV**[0]∥**IV**[1]∥**IV**[2]∥…∥**IV**[15]，

其中**IV**[i]( 0≤i≤15)为8比特的字节。则有

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

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

**IV**[4] **= BEARER**∥0002，**IV**[5]=000000002，

**IV**[6] = 000000002， **IV**[7]= 000000002，

**IV**[8] **= IV[0]**(**DIRECTION** << 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]****(**DIRECTION** << 7)， **IV**[15] = **IV**[7]。

### 5.2.2 产生密钥流

利用5.2.1节生成的初始密钥**KEY**和初始向量**IV**，祖冲之算法产生L个字的密钥流。将生成的密钥流用比特串表示为**k**[0]，**k**[1]，…，**k**[32\*L-1]，其中**k**[0]为祖冲之算法生成的第一个密钥字的最高位比特，**k**[31]为最低位比特，其它依此类推。为了计算**LENGTH**比特消息的**MAC**值，L的取值为L=⎡**LENGTH**/32⎤+2。

对于i=0，1，2，……，32\*(L-1)，令

**k**i = **k**[i]∥**k**[i+1]∥…∥**k**[i+31]。

则**k**i为32比特字。

### 5.2.3 计算MAC

设**T**为32比特字变量。置**T** = 0。

对i=0,1,…,**LENGTH**-1，如果**M**[i] = 1，那么

**T**=**T****ki**。

计算

**T**=**T****kLENGTH**。

最后计算

**MAC**=**T****k**32\*(L-1)。

（资料性附录）

算法计算实例

以下为本算法的计算实例。数据采用16进制表示。

第一组实例：

**IK** = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

**COUNT** = 0

**BEARER** = 0

**DIRECTION** = 0

**LENGTH** = 1

**M**: 00000000

**MAC**: c8a9595e

第二组实例：

**IK** = c9 e6 ce c4 60 7c 72 db 00 0a ef a8 83 85 ab 0a

**COUNT** = a94059da

**BEARER** = a

**DIRECTION** = 1

**LENGTH** = 241

**M**: 983b41d4 7d780c9e 1ad11d7e b70391b1 de0b35da 2dc62f83 e7b78d63 06ca0ea0 7e941b7b

e91348f9 fcb170e2 217fecd9 7f9f68ad b16e5d7d 21e569d2 80ed775c ebde3f40 93c53881

00000000

**MAC**: fae8ff0b

第三组实例：

**IK** = 6b 8b 08 ee 79 e0 b5 98 2d 6d 12 8e a9 f2 20 cb

**COUNT** = 561eb2dd

**BEARER** = 1c

**DIRECTION** = 0

**LENGTH** = 1626

**M**: 5bad7247 10ba1c56 d5a315f8 d40f6e09 3780be8e 8de07b69 92432018 e08ed96a 5734af8b

ad8a575d 3a1f162f 85045cc7 70925571 d9f5b94e 454a77c1 6e72936b f016ae15 7499f054

3b5d52ca a6dbeab6 97d2bb73 e41b8075 dce79b4b 86044f66 1d4485a5 43dd7860 6e0419e8

059859d3 cb2b67ce 0977603f 81ff839e 33185954 4cfbc8d0 0fef1a4c 8510fb54 7d6b06c6

11ef44f1 bce107cf a45a06aa b360152b 28dc1ebe 6f7fe09b 0516f9a5 b02a1bd8 4bb0181e

2e89e19b d8125930 d178682f 3862dc51 b636f04e 720c47c3 ce51ad70 d94b9b22 55fbae90

6549f499 f8c6d399 47ed5e5d f8e2def1 13253e7b 08d0a76b 6bfc68c8 12f375c7 9b8fe5fd

85976aa6 d46b4a23 39d8ae51 47f680fb e70f978b 38effd7b 2f7866a2 2554e193 a94e98a6

8b74bd25 bb2b3f5f b0a5fd59 887f9ab6 8159b717 8d5b7b67 7cb546bf 41eadca2 16fc1085

0128f8bd ef5c8d89 f96afa4f a8b54885 565ed838 a950fee5 f1c3b0a4 f6fb71e5 4dfd169e

82cecc72 66c850e6 7c5ef0ba 960f5214 060e71eb 172a75fc 1486835c bea65344 65b055c9

6a72e410 52241823 25d83041 4b40214d aa8091d2 e0fb010a e15c6de9 0850973b df1e423b

e148a237 b87a0c9f 34d4b476 05b803d7 43a86a90 399a4af3 96d3a120 0a62f3d9 507962e8

e5bee6d3 da2bb3f7 237664ac 7a292823 900bc635 03b29e80 d63f6067 bf8e1716 ac25beba

350deb62 a99fe031 85eb4f69 937ecd38 7941fda5 44ba67db 09117749 38b01827 bcc69c92

b3f772a9 d2859ef0 03398b1f 6bbad7b5 74f7989a 1d10b2df 798e0dbf 30d65874 64d24878

cd00c0ea ee8a1a0c c753a279 79e11b41 db1de3d5 038afaf4 9f5c682c 3748d8a3 a9ec54e6

a371275f 1683510f 8e4f9093 8f9ab6e1 34c2cfdf 4841cba8 8e0cff2b 0bcc8e6a dcb71109

b5198fec f1bb7e5c 531aca50 a56a8a3b 6de59862 d41fa113 d9cd9578 08f08571 d9a4bb79

2af271f6 cc6dbb8d c7ec36e3 6be1ed30 8164c31c 7c0afc54 1c000000

**MAC**: 0ca12792

# 参考文献

1. ETSI/SAGE TS 35.221. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 1: 128-EEA3 and 128-EIA3 Specification.
2. ETSI/SAGE TS 35.222. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification.
3. ETSI/SAGE TS 35.223. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 3: Implementor's Test Data.
4. ETSI/SAGE TR 35.924. Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 4: Design and Evaluation Report.