

祖冲之序列密码算法

ZUC stream cipher algorithm

目 次

目 次.....	II
祖冲之序列密码算法.....	3
1 术语和定义.....	3
2 符号和缩略语.....	3
2.1 运算符.....	3
2.2 符号.....	4
2.3 缩略语.....	4
3 算法描述.....	4
3.1 算法整体结构.....	4
3.2 线性反馈移位寄存器 LFSR.....	5
3.3 比特重组 BR.....	6
3.4 非线性函数 F.....	6
3.5 密钥装入.....	6
3.6 算法运行.....	7
附 录 A S 盒.....	8
附 录 B 模 $2^{31}-1$ 乘法和模 $2^{31}-1$ 加法的实现.....	10
附 录 C 算法计算实例.....	11
参考文献.....	15

祖冲之序列密码算法

1 术语和定义

下列术语和定义适用于本文件。

1.1

比特 bit

二进制字符0或1称之为比特。

1.2

字节 byte

由8个比特组成的比特串称之为字节。

1.3

字 word

由2个以上（包含2个）比特组成的比特串称之为字。

本文主要使用31比特字和32比特字。

1.4

字表示 word representation

本文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。

1.5

高低位顺序 bit ordering

本文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。

2 符号和缩略语

2.1 运算符

+	算术加法运算
ab	整数a和b的乘积
=	赋值操作符
mod	整数模运算
\oplus	按比特位逐位异或运算
\boxplus	模 2^{32} 加法运算
\parallel	字符串或字节串连接符
\cdot_H	取字的最高16比特
\cdot_L	取字的最低16比特
$\lll k$	32比特字循环左移k位

$\gg k$ 32比特字右移 k 位
 $\mathbf{a} \rightarrow \mathbf{b}$ 向量 \mathbf{a} 赋值给向量 \mathbf{b} ，即按分量逐分量赋值

2.2 符号

下列符号适用于本文：

$s_0, s_1, s_2, \dots, s_{15}$	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
W_1	R_1 与 X_1 进行模 2^{32} 加法运算输出的 32 比特字
W_2	R_2 与 X_2 按比特位逐位异或运算输出的 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
i_V	初始向量
d_i	15 比特的字符串常量， $i=0, 1, 2, \dots, 15$
F	非线性函数
L	输出密钥字长度

2.3 缩略语

下列缩略语适用于本文：

ZUC	祖冲之序列密码算法或者祖冲之算法
LFSR	线性反馈移位寄存器
BR	比特重组

3 算法描述

3.1 算法整体结构

祖冲之算法逻辑上分为上中下层，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。

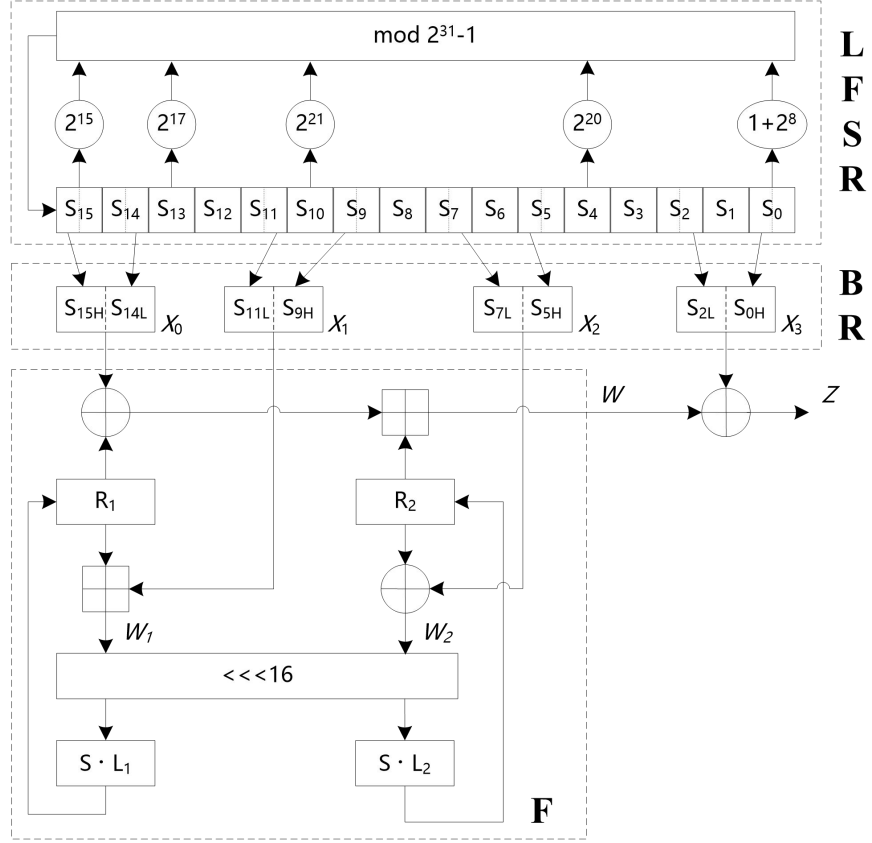


图 1 祖冲之算法结构图

3.2 线性反馈移位寄存器 LFSR

3.2.1 概述

LFSR 包括 16 个 31 比特寄存器单元变量 s_0, s_1, \dots, s_{15} 。

LFSR 的运行模式有 2 种：初始化模式和工作模式。

3.2.2 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 $u = W \gg 1$ 。在初始化模式下，LFSR 计算过程如下：

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)  $(s_1, s_2, \dots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \dots, s_{14}, s_{15})$ 。
}
```

模 $2^{31}-1$ 乘法和模 $2^{31}-1$ 加法的实现参见附录 B。

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

3.3 比特重组 BR

输入为 LFSR 寄存器单元变量 $s_0, s_2, s_5, s_7, s_9, s_{11}, s_{14}, s_{15}$, 输出为 4 个 32 比特字 X_0, X_1, X_2, X_3 。计算过程如下:

```
BitReconstruction()
{
    (1)  $X_0 = s_{15H} \parallel s_{14L}$ ;
    (2)  $X_1 = s_{11L} \parallel s_{9H}$ ;
    (3)  $X_2 = s_{7L} \parallel s_{5H}$ ;
    (4)  $X_3 = s_{2L} \parallel s_{0H}$ 。
}
```

3.4 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 。计算过程如下:

```
F( $X_0, X_1, X_2$ )
{
    (1)  $W = (X_0 \oplus R_1) \boxplus R_2$ ;
    (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)  $R_2 = S(L_2(W_{2L} \parallel W_{1H}))$ 。
}
```

其中 S 为 32 比特的 S 盒变换, S 盒定义见附录 A; L_1 和 L_2 为 32 比特线性变换, 定义如下:

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

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

3.5 密钥装入

将初始密钥 k 和初始向量 iv 分别扩展为 16 个 31 比特字作为 LFSR 寄存器单元变量 s_0, s_1, \dots, s_{15} 的初始状态。设 k 和 iv 分别为

$$k_0 \parallel k_1 \parallel \dots \parallel k_{15}$$

和

$$iv_0 \parallel iv_1 \parallel \dots \parallel iv_{15},$$

其中 k_i 和 iv_i 均为 8 比特字节, $0 \leq i \leq 15$ 。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 比特的字串:

$$D = d_0 \parallel d_1 \parallel \dots \parallel d_{15}$$

其中:

$$d_0 = 100010011010111_2,$$

$$d_1 = 010011010111100_2,$$

$$d_2 = 110001001101011_2,$$

$$d_3 = 001001101011110_2,$$

$$\begin{aligned}
d_4 &= 101011110001001_2, \\
d_5 &= 011010111100010_2, \\
d_6 &= 111000100110101_2, \\
d_7 &= 000100110101111_2, \\
d_8 &= 100110101111000_2, \\
d_9 &= 010111100010011_2, \\
d_{10} &= 110101111000100_2, \\
d_{11} &= 001101011110001_2, \\
d_{12} &= 101111000100110_2, \\
d_{13} &= 011110001001101_2, \\
d_{14} &= 111100010011010_2, \\
d_{15} &= 100011110101100_2.
\end{aligned}$$

(2) 对 $0 \leq i \leq 15$, 有 $s_i = k_i \parallel d_i \parallel iv_i$ 。

3.6 算法运行

3.6.1 初始化阶段

首先把128比特的初始密钥 k 和初始向量 iv 按照3.5的密钥装入方法装到LFSR的寄存器单元变量 s_0, s_1, \dots, 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:

重复下述过程32次:

- (1) BitReconstruction();
- (2) $W = F(X_0, X_1, X_2)$;
- (3) 输出 32 比特字 W ;
- (4) LFSRWithInitialisationMode ($W \gg 1$)。

3.6.2 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) $Z = F(X_0, X_1, X_2) \oplus X_3$;
- (3) 输出 32 比特密钥字 Z ;
- (4) LFSRWithWorkMode()。

附 录 A

S 盒

32 比特 S 盒 S 由 4 个小的 8×8 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 A.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 $x = h \parallel l$, 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:

$$X = x_0 \parallel x_1 \parallel x_2 \parallel x_3,$$

$$Y = y_0 \parallel y_1 \parallel y_2 \parallel y_3,$$

其中 x_i 和 y_i 均为 8 比特字节, $i = 0, 1, 2, 3$ 。则有 $y_i = S_i(x_i)$, $i = 0, 1, 2, 3$ 。

表 A.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 A.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

表 A.2 S_1 盒 (续)

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。

附 录 B

模 $2^{31}-1$ 乘法和模 $2^{31}-1$ 加法的实现

B.1 模 $2^{31}-1$ 乘法

两个 31 比特字模 $2^{31}-1$ 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 $2^{31}-1$ 加法运算实现。例如，计算 $ab \bmod (2^{31}-1)$ ，其中 $b=2^i+2^j+2^k$ 。则

$$ab \bmod (2^{31}-1) = (a \lll_{31} i) + (a \lll_{31} j) + (a \lll_{31} k) \bmod (2^{31}-1),$$

其中： \lll_{31} 表示31比特左循环移位运算。

B.2 模 $2^{31}-1$ 加法

在 32 位处理平台上，两个 31 比特字 a 和 b 模 $2^{31}-1$ 加法运算 $c = a + b \bmod (2^{31}-1)$ 可以通过下面的两步计算实现：

- 1) $c = a + b;$
- 2) $c = (c \& 0x7FFFFFFF) + (c \gg 31)。$

附录 C 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:

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
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	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	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:

$R_1 = 14cfd44c$

$R_2 = 8c6de800$

密钥流:

t	X_0	X_1	X_2	X_3	R_1	R_2	z	S_{15}
0	7c37ba6b	b1367f6c	1e426568	dd0bf9c2	3512bf50	a0920453	286dafa5	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

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

输出:

z_1 : 0657cfa0

z_2 : 7096398b

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	7fc4d7ff	7fa6bcff	7fe26bff	7f935eff	7fd789ff	7fb5e2ff	7ff135ff	7f89afff
8	7fcd78ff	7faf13ff	7feb4c4f	7f9af1ff	7fde26ff	7fbc4dff	7ff89aff	7fc7acff
t	X_0	X_1	X_2	X_3	R_1	R_2	W	S_{15}
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	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	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：

$$R_1 = \text{b8017bd5}$$

$$R_2 = \text{9ce2de5c}$$

密钥流：

t	X_0	X_1	X_2	X_3	R_1	R_2	z	S_{15}
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：

$$\text{密钥 } k: \quad 3d \ 4c \ 4b \ e9 \ 6a \ 82 \ fd \ ae \ b5 \ 8f \ 64 \ 1d \ b1 \ 7b \ 45 \ 5b$$

$$\text{初始向量 } iv: \quad 84 \ 31 \ 9a \ a8 \ de \ 69 \ 15 \ ca \ 1f \ 6b \ da \ 6b \ fb \ d8 \ c7 \ 66$$

输出：

$$z_1: \text{14f1c272}$$

$$z_2: \text{3279c419}$$

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	1ec4d784	2626bc31	25e26b9a	74935ea8	355789de	4135e269	7ef13515	5709afca
8	5acd781f	47af136b	326bc4da	0e9af16b	58de26fb	3dbc4dd8	22f89ac7	2dc7ac66
t	X_0	X_1	X_2	X_3	R_1	R_2	W	S_{15}
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	8d89dddde	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	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	10da5941	5b6acb6f	17060ce1	35368174	5cf4385a	479943df	2753bab2	73775d6a
8	43930a37	77b4af31	15b2e89f	24ff6e20	740c40b9	026a5503	194b2a57	7a9a1cff

有限状态机内部状态：

$$R_1 = 860a7dfa$$

$$R_2 = bf0e0ffc$$

密钥流：

t	X_0	X_1	X_2	X_3	R_1	R_2	z	S_{15}
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.