密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2018, 5(2): 167–179

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

ZUC-256 流密码算法

ZUC 算法研制组

[编者按] ZUC-128 流密码算法, 又称祖冲之密码算法, 是基于 128 比特密钥设计的, 它包含加密算法 128-EEA3 和完整性验证算法 128-EIA3, 主要用于移动通信系统的数据机密性与完整性保护. 在 2011 年 9 月日本福冈召开的第 53 次第三代合作伙伴计划 3GPP 系统架构组 SA 会议上, ZUC-128 算法被批准成为 3GPP 的 LTE 国际标准密码算法. 在近期的 3GPP 会议上, 多个国家提出了 256 比特密钥安全的密码算法需求. 经讨论, 3GPP 明确了 5G 通信中需要 128 比特密钥和 256 比特密钥的对称密码算法, 与 4G 通信保持兼容. 3GPP 确定在今明两年完成 256 比特密码算法应用协议标准化. 为此, ZUC 算法研制组研制了 "ZUC-256 流密码算法", 从 128 比特密钥升级到 256 比特密钥, 升级版的算法与 ZUC-128 算法高度兼容且适应 5G 应用环境, 可提供消息加密和认证. 《ZUC-256 流密码算法》的中文版和英文版已在网上发布, 本刊全文刊载, 以飨读者.

摘 要: 为了应对 5G 通信与后量子密码时代的来临, 本文提出 ZUC-256 流密码. ZUC-256 流密码是 3GPP 机密性与完整性算法 128-EEA3 和 128-EIA3 中采用的 ZUC-128 流密码的 256 比特密钥升级版本, 与 ZUC-128 流密码高度兼容. ZUC-256 流密码的设计目标是提供 5G 应用环境下的 256 比特安全性; 其认证部分在初始向量不可复用的条件下支持多种标签长度.

关键词: 祖冲之算法; 流密码; 256 比特安全性

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000228

中文引用格式: ZUC 算法研制组. ZUC-256 流密码算法[J]. 密码学报, 2018, 5(2): 167–179.

英文引用格式: Design Team. ZUC-256 stream cipher[J]. Journal of Cryptologic Research, 2018, 5(2): 167–179.

ZUC-256 Stream Cipher

Design Team

Abstract: To be well adapted to the 5G communications and the post-quantum cryptography era, we propose the ZUC-256 stream cipher in this paper, a successor of the previous ZUC-128 stream cipher used in the 3GPP confidentiality and integrity algorithms 128-EEA3 and 128-EIA3 which is highly compatible with the ZUC-128 stream cipher. The aim is a new stream cipher that offers the 256-bit security for the upcoming applications in 5G. For the authentication, various tag sizes are supported with the IV-respecting restriction.

Key words: ZUC algorithm; stream ciphers; 256-bit security

1 引言

众所周知, 3GPP 机密性与完整性算法 128-EEA3 和 128-EIA3 的核心是 ZUC-128 流密码 [1]. 随着通信与计算技术的发展, 对未来 5G 应用环境下可提供 256 比特安全性的新型流密码有着迫切的需求.

本文给出 ZUC-256 流密码的完整描述, 在保持与 ZUC-128 流密码高度兼容的基础上, 同时满足 5G 的应用环境. 与 ZUC-128 流密码相比, ZUC-256 流密码在初始化阶段、消息认证码 (MAC, 也称认证标

签或者标签) 生成阶段采用了新的设计方案以满足 5G 应用的各种需求.

本文结构如下: 首先在第2节给出 ZUC-256 流密码的完整描述, 包含了初始化阶段、密钥流生成阶段及消息认证码生成阶段; 第3节总结了全文.

2 算法描述

本节给出 ZUC-256 流密码的完整描述. 首先约定下列符号.

- 记整数的模 2^{32} 加法为 $\boxplus$, 即对于 $0 \leqslant x, y < 2^{32}$, $x \boxplus y$ 就是 $\bmod 2^{32}$ 的整数加法运算;
- 记整数的模 $(2^{31}-1)$ 加法为 $(x+y) \bmod (2^{31}-1)$, 其中 $1 \leqslant x \leqslant 2^{31}-1$, 且 $1 \leqslant y \leqslant 2^{31}-1$;
- 记比特级的异或操作为 $\oplus$;
- 记比特串的连接操作为 $\|$;
- 记比特级的逻辑或运算为 $|$;
- 令 $K = (K_{31}, K_{30}, \cdots, K_1, K_0)$ 为 ZUC-256 流密码采用的 256-比特密钥, 其中 K_i $(0 \leqslant i \leqslant 31)$ 为 8-比特字节;
- 令 $\mathrm{IV} = (\mathrm{IV}_{24}, \mathrm{IV}_{23}, \cdots, \mathrm{IV}_{17}, \mathrm{IV}_{16}, \mathrm{IV}_{15}, \cdots, \mathrm{IV}_1, \mathrm{IV}_0)$ 为 ZUC-256 流密码采用的 184-比特初始向量, 其中 IV_i $(0 \leqslant i \leqslant 16)$ 为 8-比特字节; IV_i $(17 \leqslant i \leqslant 24)$ 为 6-比特长的比特串, 占据一个字节的低 6 位;
- 令 $d_i (0 \leqslant i \leqslant 15)$ 为 ZUC-256 流密码采用的 7-比特常数;
- 记 64-比特操作数的向左循环移位为 $\lll$, $x \lll n$ 即为 $((x \ll n) \mid (x \gg (64-n)))$, 其中 $\ll$ 与 $\gg$ 分别为相应操作数的逻辑左移与逻辑右移.

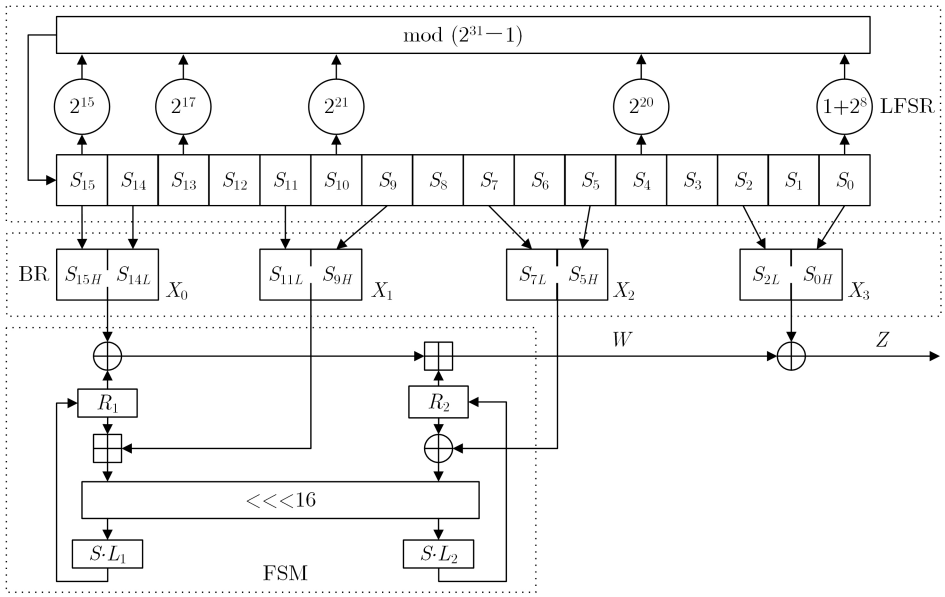

图 1 ZUC-256 流密码的密钥流生成阶段

Figure 1 Keystream generation of ZUC-256 stream cipher

ZUC-256 流密码由 3 部分组成, 如图1与图2所示: 首先是一个 496-比特长的线性反馈移位寄存器 (LFSR), 该 LFSR 定义在域 $\mathrm{GF}(2^{31}-1)$ 上, 由 16 个 31-比特寄存器单元 $(s_{15}, s_{14}, \cdots, s_1, s_0)$ 构成, 这些单元均定义在代表元集合 $\{1, 2, \cdots, 2^{31}-1\}$ 上; 其次是一个比特重组层 (BR), 它主要用来从 LFSR 中抽取一些存储内容, 并拼接成 4 个 32-比特字 (X_0, X_1, X_2, X_3), 用于下面的有限状态自动机 (FSM) 和输出处理; 最后是 FSM 层, 包含 2 个 32-比特字 R_1 与 R_2 作为 FSM 中的记忆单元.

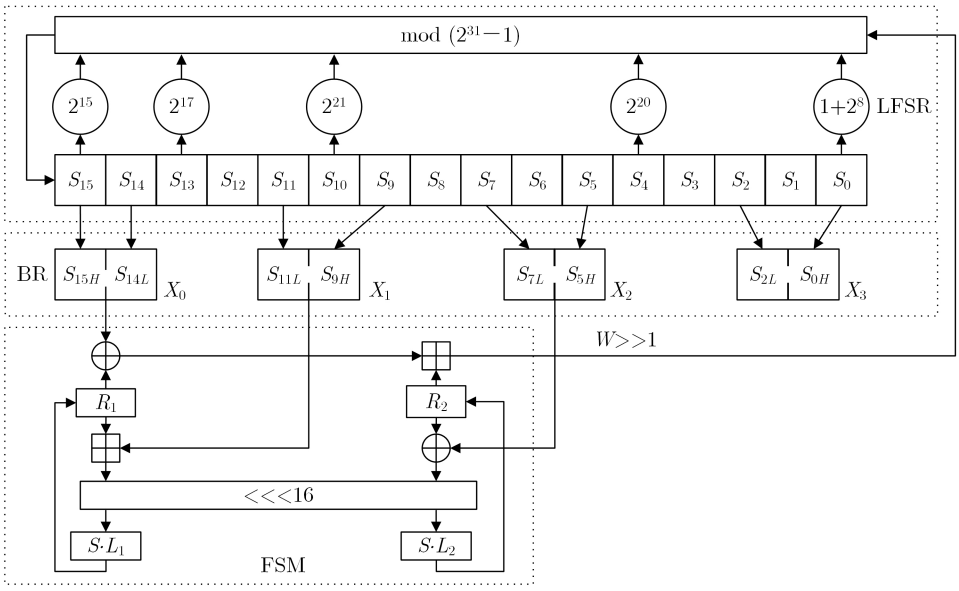

图 2 ZUC-256 流密码的初始化阶段
Figure 2 Initialization of ZUC-256 stream cipher

ZUC-256 流密码的密钥/初始向量装载过程如下.

$$s_0 = K_0 \parallel d_0 \parallel K_{21} \parallel K_{16} \qquad s_8 = K_8 \parallel (d_8 \mid \mathrm{IV}_{20}) \parallel \mathrm{IV}_3 \parallel \mathrm{IV}_{11}$$

$$s_1 = K_1 \parallel d_1 \parallel K_{22} \parallel K_{17} \qquad s_9 = K_9 \parallel (d_9 \mid \mathrm{IV}_{21}) \parallel \mathrm{IV}_{12} \parallel \mathrm{IV}_4$$

$$s_2 = K_2 \parallel d_2 \parallel K_{23} \parallel K_{18} \qquad s_{10} = \mathrm{IV}_5 \parallel (d_{10} \mid \mathrm{IV}_{22}) \parallel K_{10} \parallel K_{28}$$

$$s_3 = K_3 \parallel d_3 \parallel K_{24} \parallel K_{19} \qquad s_{11} = K_{11} \parallel (d_{11} \mid \mathrm{IV}_{23}) \parallel \mathrm{IV}_6 \parallel \mathrm{IV}_{13}$$

$$s_4 = K_4 \parallel d_4 \parallel K_{25} \parallel K_{20} \qquad s_{12} = K_{12} \parallel (d_{12} \mid \mathrm{IV}_{24}) \parallel \mathrm{IV}_7 \parallel \mathrm{IV}_{14}$$

$$s_5 = \mathrm{IV}_0 \parallel (d_5 \mid \mathrm{IV}_{17}) \parallel K_5 \parallel K_{26} \qquad s_{13} = K_{13} \parallel d_{13} \parallel \mathrm{IV}_{15} \parallel \mathrm{IV}_8$$

$$s_6 = \mathrm{IV}_1 \parallel (d_6 \mid \mathrm{IV}_{18}) \parallel K_6 \parallel K_{27} \qquad s_{14} = K_{14} \parallel (d_{14} \mid (K_{31})_H^4) \parallel \mathrm{IV}_{16} \parallel \mathrm{IV}_9$$

$$s_7 = \mathrm{IV}_{10} \parallel (d_7 \mid \mathrm{IV}_{19}) \parallel K_7 \parallel \mathrm{IV}_2 \qquad s_{15} = K_{15} \parallel (d_{15} \mid (K_{31})_L^4) \parallel K_{30} \parallel K_{29}$$

其中 $(K_{31})_H^4$ 是字节 K_{31} 的高 4 位, $(K_{31})_L^4$ 是 K_{31} 的低 4 位, 所使用的填充常数 d_i $(0 \leqslant i \leqslant 15)$ 如下.

$$d_0 = 0100010 \quad d_4 = 1101101 \quad d_8 = 1000000 \quad d_{12} = 1000000$$

$$d_1 = 0101111 \quad d_5 = 1000000 \quad d_9 = 1000000 \quad d_{13} = 1010010$$

$$d_2 = 0100100 \quad d_6 = 1000000 \quad d_{10} = 1000000 \quad d_{14} = 0010000$$

$$d_3 = 0101010 \quad d_7 = 1000000 \quad d_{11} = 1000000 \quad d_{15} = 0110000$$

ZUC-256 流密码的初始化阶段共有 $32 + 1 = 33$ 轮, 其具体描述如下.

1. 按如上所述将密钥、初始向量及常数装载到 LFSR 各单元
2. 初始化记忆单元为 $R_1 = R_2 = 0$
3. for $i = 0$ to 31 do
   Bitreorganization( )
   $W = F(X_0, X_1, X_2)$
   LFSRWithInitializationMode($W \gg 1$)
4. Bitreorganization( )

$W = F(X_0, X_1, X_2)$, 但舍弃 W
LFSRWithworkMode()

下面, 我们逐一给出各个相关子程序的描述.

LFSRWithInitializationMode(u)

1. $v = (2^{15} \cdot s_{15} + 2^{17} \cdot s_{13} + 2^{21} \cdot s_{10} + 2^{20} \cdot s_4 + (1 + 2^8) \cdot s_0) \bmod (2^{31} - 1)$
2. 若 $v = 0$, 则令 $v = 2^{31} - 1$
3. $s_{16} = (v + u) \bmod (2^{31} - 1)$
4. 若 $s_{16} = 0$, 则令 $s_{16} = 2^{31} - 1$
5. $(s_{16}, s_{15}, \cdots, s_2, s_1) \rightarrow (s_{15}, s_{14}, \cdots, s_1, s_0)$, 其中 $\rightarrow$ 表示赋值操作

LFSRWithworkMode()

1. $s_{16} = (2^{15} \cdot s_{15} + 2^{17} \cdot s_{13} + 2^{21} \cdot s_{10} + 2^{20} \cdot s_4 + (1 + 2^8) \cdot s_0) \bmod (2^{31} - 1)$
2. 若 $s_{16} = 0$, 则令 $s_{16} = 2^{31} - 1$
3. $(s_{16}, s_{15}, \cdots, s_2, s_1) \rightarrow (s_{15}, s_{14}, \cdots, s_1, s_0)$

Bitreorganization()

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. $X_3 = s_{2L} \parallel s_{0H}$

其中 s_{iH} 为寄存器单元 s_i 的高 16 位, 而 s_{jL} 为寄存器单元 s_j 的低 16 位.

$F(X_0, X_1, X_2)$

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 $S = (S_0, S_1, S_0, S_1)$ 为 4 个并置的 S-盒, 这一部分与之前 ZUC-128 流密码的同一部件完全相同; 而 L_1 与 L_2 也是 ZUC-128 流密码采用的两个 MDS 矩阵. ZUC-256 流密码每一节拍产生一个 32-比特字作为输出密钥流.

KeystreamGeneration()

1. Bitreorganization( )
2. $Z = F(X_0, X_1, X_2) \oplus X_3$
3. LFSRWithworkMode()

在 5G 应用环境中, ZUC-256 流密码每一帧产生 20000-比特到 2^{32} 比特长的密钥流, 亦即每一帧产生 625 到 2^{27} 个密钥流字; 随后进行一次密钥/初始向量的再同步过程, 在这一过程中保持密钥/常数不变, 而初始向量则演变为一个新值.

ZUC-256 流密码的 MAC 生成原理与 ZUC-128 流密码的 MAC 生成原理相同, 具体过程如下. 令 $M = (m_0, m_1, \cdots, m_{l-1})$ 为一段 l-比特长的明文消息, 其认证标签长度 t 可取为 32, 64 及 128 比特.

MAC_Generation(M)

1. 运行 ZUC-256 流密码产生一段包含 $L = \lceil \frac{l}{32} \rceil + 2 \cdot \frac{t}{32}$ 个字的密钥流. 记该密钥流的二元序列为 $z_0, z_1, \cdots, z_{32 \cdot L-1}$, 其中 z_0 是第一个密钥流字的最高位而 z_{31} 是该字的最低位.
2. 初始化 Tag $= (z_0, z_1, \cdots, z_{t-1})$
3. for $i = 0$ to $l - 1$ do

   令 $W_i = (z_{t+i}, \cdots, z_{i+2t-1})$

   若 $m_i = 1$, 则 Tag $=$ Tag $\oplus W_i$
4. $W_l = (z_{l+t}, \cdots, z_{l+2t-1})$
5. Tag $=$ Tag $\oplus W_l$
6. 返回 Tag

对于不同长度的认证标签, 为了防止伪造攻击, 各长度所采用的常数如下.

1. 对于 32-比特的标签长度, 所采用的常数如下.

$d_0 = 0100010$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100101$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

2. 对于 64-比特的标签长度, 所采用的常数如下.

$d_0 = 0100011$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100100$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

3. 对于 128-比特的标签长度, 所采用的常数如下.

$d_0 = 0100011$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100101$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

ZUC-256 流密码的测试向量如下. 首先对于密钥流生成阶段, 有如下测试向量.

1. 令密钥 $K_i =$ `0x00`, 对于 $0 \leqslant i \leqslant 31$, 而初始向量 $\mathrm{IV}_i =$ `0x00`, 对于 $0 \leqslant i \leqslant 24$, 则头 20 个密钥流字为
   - `58d03ad6,2e032ce2,dafc683a,39bdcb03,52a2bc67,`
   - `f1b7de74,163ce3a1,01ef5558,9639d75b,95fa681b,`
   - `7f090df7,56391ccc,903b7612,744d544c,17bc3fad,`
   - `8b163b08,21787c0b,97775bb8,4943c6bb,e8ad8afd`
2. 令密钥 $K_i =$ `0xff`, 对于 $0 \leqslant i \leqslant 31$, 而初始向量 $\mathrm{IV}_i =$ `0xff`, 对 $0 \leqslant i \leqslant 16$ 及 $\mathrm{IV}_i =$ `0x3f`, 对 $17 \leqslant i \leqslant 24$, 则头 20 个密钥流字为
   - `3356cbae,d1a1c18b,6baa4ffe,343f777c,9e15128f,`
   - `251ab65b,949f7b26,ef7157f2,96dd2fa9,df95e3ee,`
   - `7a5be02e,c32ba585,505af316,c2f9ded2,7cdbd935,`

- `e441ce11,15fd0a80,bb7aef67,68989416,b8fac8c2`

其次对于消息认证码生成阶段, 有如下测试向量.

1. 令密钥 $K_i = \texttt{0x00}$, 对于 $0 \leqslant i \leqslant 31$, 而初始向量 $\mathrm{IV}_i = \texttt{0x00}$, 对于 $0 \leqslant i \leqslant 24$, $l = 400$-比特的消息为 $M = 0x\underbrace{00, \cdots, 00}_{100}$, 则相应的 32-比特标签, 64-比特标签与 128-比特标签分别如下
   - 32-比特认证标签为 `9b972a74`
   - 64-比特认证标签为 `673e5499 0034d38c`
   - 128-比特认证标签为 `d85e54bb cb960096 7084c952 a1654b26`
2. 令密钥 $K_i = \texttt{0x00}$, 对于 $0 \leqslant i \leqslant 31$, 而初始向量 $\mathrm{IV}_i = \texttt{0x00}$, 对于 $0 \leqslant i \leqslant 24$, $l = 4000$-比特的消息为 $M = 0x\underbrace{11, \cdots, 11}_{1000}$, 则相应的 32-比特标签, 64-比特标签与 128-比特标签分别如下
   - 32-比特认证标签为 `8754f5cf`
   - 64-比特认证标签为 `130dc225 e72240cc`
   - 128-比特认证标签为 `df1e8307 b31cc62b eca1ac6f 8190c22f`
3. 令密钥 $K_i = \texttt{0xff}$, 对于 $0 \leqslant i \leqslant 31$, 而初始向量 $\mathrm{IV}_i = \texttt{0xff}$, 对于 $0 \leqslant i \leqslant 16$ 及 $\mathrm{IV}_i = \texttt{0x3f}$, 对于 $17 \leqslant i \leqslant 24$, $l = 400$-比特的消息为 $M = 0x\underbrace{00, \cdots, 00}_{100}$, 则 32-比特标签, 64-比特标签与 128-比特标签分别如下
   - 32-比特认证标签为 `1f3079b4`
   - 64-比特认证标签为 `8c71394d 39957725`
   - 128-比特认证标签为 `a35bb274 b567c48b 28319f11 1af34fbd`
4. 令密钥 $K_i = \texttt{0xff}$, 对于 $0 \leqslant i \leqslant 31$, 而初始向量 $\mathrm{IV}_i = \texttt{0xff}$, 对于 $0 \leqslant i \leqslant 16$ 及 $\mathrm{IV}_i = \texttt{0x3f}$, 对于 $17 \leqslant i \leqslant 24$, $l = 4000$-比特的消息为 $M = 0x\underbrace{11, \cdots, 11}_{1000}$, 则 32-比特标签, 64-比特标签与 128-比特标签分别如下
   - 32-比特认证标签为 `5c7c8b88`
   - 64-比特认证标签为 `ea1dee54 4bb6223b`
   - 128-比特认证标签为 `3a83b554 be408ca5 494124ed 9d473205`

ZUC-256 流密码的安全性目标是提供 5G 应用环境下的 256 比特安全性. 对于认证部分的伪造攻击, ZUC-256 流密码可提供相当于标签长度的安全性, 这里特别强调, 在 ZUC-256 流密码中, 初始向量不可复用; 在认证标签验证失败时, 不可产生任何的输出.

3 结束语

在本文中我们给出了 ZUC-256 流密码的完整描述, 欢迎专家学者对 ZUC-256 流密码进行密码分析.

References

[1] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 and 128-EIA3, Document 4: Design and Evaluation Reprot[OL]. http://www.gsmworld.com/documents/EEA3_EIA3_Design_Evaluation_v1_1.pdf.

联系人: 张斌, E-mail: zhangbin@tca.iscas.ac.cn

密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2018, 5(2): 167–179

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

ZUC-256 Stream Cipher

Design Team

[**Editorials**] Called also as Zu Chongzhi's cipher, ZUC-128 stream cipher is designed on the basis of 128-bit key. Including encryption algorithm 128-EEA3 and integrity algorithm 128-EIA3, it is mainly used for data encryption and integrity protection of mobile communication systems. In September 2011, ZUC-128 was approved as the LTE international standard cipher for 3GPP at the 53rd 3GPP Meeting for System Architecture Group held at Fukuoka, Japan. Several countries raised the need for cipher of 256-bit secured key at recent 3GPP meetings. After the discussion, 3GPP clarified that in 5G communications, symmetric cryptography is needed for 128-bit key and 256-bit key, and it should be compatible with 4G communications. It is also cleared that the application protocol standardization of 256-bit cipher should be finalized within two years. Designating to fulfill the above demand, ZUC Design Team has carried out "ZUC-256 Stream Cipher", with key upgraded from 128-bit to 256-bit. The upgraded algorithm is highly compatible with ZUC-128 stream cipher and adaptive well to 5G applications, thus it can provide message encryption and ID authentication. The paper "ZUC-256 Stream Cipher", both in Chinese and in English, was archived at the website. Journal of Cryptologic Research now publishes the full version of the paper for the convenience of our fellow readers.

Abstract: To be well adapted to the 5G communications and the post-quantum cryptography era, we propose the ZUC-256 stream cipher in this study, a successor of the previous ZUC-128 stream cipher used in the 3GPP confidentiality and integrity algorithms 128-EEA3 and 128-EIA3 which is highly compatible with the ZUC-128 stream cipher and is of new design features. The aim of ZUC-256 is a new stream cipher that offers the 256-bit security for the upcoming applications in 5G. For the authentication, various tag sizes are supported with the IV-respecting restriction.
Key words: ZUC algorithm; stream ciphers; 256-bit security

DOI: 10.13868/j.cknl.jcr.000228
Cite: Design Team. ZUC-256 stream cipher[J]. Journal of Cryptologic Research, 2018, 5(2): 167–179.

1 Introduction

The core of the 3GPP confidentiality and integrity algorithms 128-EEA3 and 128-EIA3 is the ZUC-128 stream cipher [1]. With the development of the communication and computing technology, there is an emerging need for the new core stream cipher in the upcoming 5G applications which offers 256-bit security. To be highly compatible with the current 128-bit version, we present the ZUC-256 stream cipher, which is a successor of the previous ZUC-128 stream cipher and is adaptive to 5G applications at the same time. The new ZUC-256 stream cipher differs from ZUC-128 only in the initialization phase and in the message authentication codes (MAC, also called authentication tag or tag) generation phase, other aspects are all the same as the previous ZUC-128 algorithm.

This paper is structured as follows. In Section 2, we give the detailed description of the new ZUC-256 stream cipher, including both the initialization phase, the keystream generation phase, and the MAC generation phase. Finally, some conclusions are drawn in Section 3.

2 Description of Cipher

In this section, we will present the detailed description of the ZUC-256 stream cipher. The following notations will be used hereafter.

- Denote the integer modular addition by $\boxplus$, i.e., for $0 \leqslant x < 2^{32}$ and $0 \leqslant y < 2^{32}$, $x \boxplus y$ is the integer addition $\bmod 2^{32}$;
- Denote the integer addition modulo $(2^{31}-1)$ by $(x+y) \bmod (2^{31}-1)$ for $1 \leqslant x \leqslant 2^{31}-1$ and $1 \leqslant y \leqslant 2^{31}-1$;
- Denote the bitwise exclusive OR by $\oplus$;
- Denote the bit string concatenation by $\|$;
- Denote the bitwise logic OR by $|$;
- $K = (K_{31}, K_{30}, \cdots, K_1, K_0)$, the 256-bit secret key used in ZUC-256 where K_i for $0 \leqslant i \leqslant 31$ are 8-bit bytes;
- $\mathrm{IV} = (\mathrm{IV}_{24}, \mathrm{IV}_{23}, \cdots, \mathrm{IV}_{17}, \mathrm{IV}_{16}, \mathrm{IV}_{15}, \cdots, \mathrm{IV}_1, \mathrm{IV}_0)$, the 184-bit initialization vector used in ZUC-256 where IV_i for $0 \leqslant i \leqslant 16$ are 8-bit bytes and IV_i for $17 \leqslant i \leqslant 24$ are 6-bit string occupying the 6 least significant bits of a byte;
- d_i for $0 \leqslant i \leqslant 15$ are the 7-bit constants used in the ZUC-256 stream cipher;
- $\lll$, the left rotation of a 64-bit operand, $x \lll n$ means $((x \ll n) \mid (x \gg (64-n)))$, where '$\ll$' and '$\gg$' are the logical left shift and right shift, respectively.

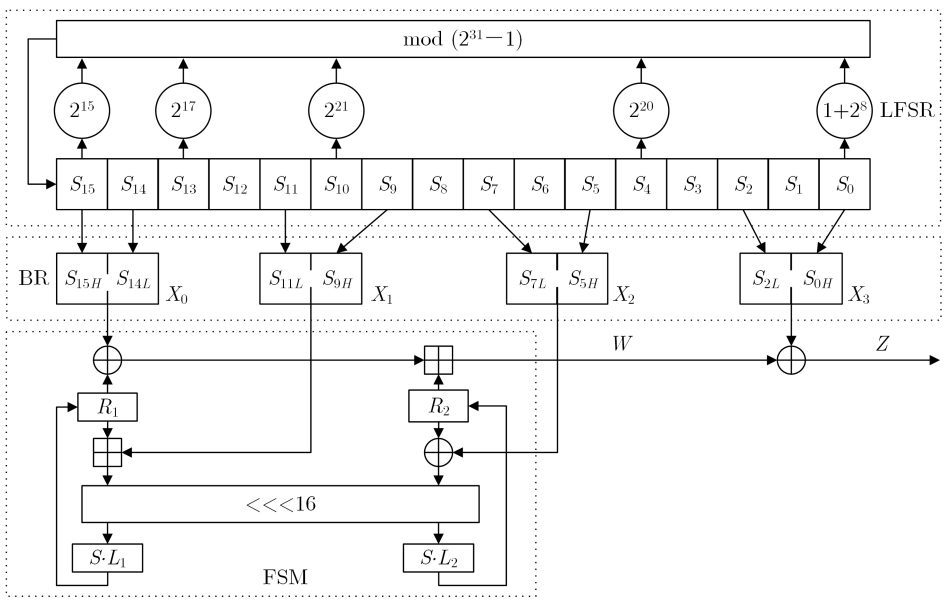

Figure 1 Keystream generation of ZUC-256 stream cipher

As depicted in Fig.1 and Fig.2, there are 3 parts involved in ZUC-256: linear feedback shift register (LFSR), bit reorganization layer (BR), and finite state machine (FSM). A 496-bit LFSR defined over the field $\mathrm{GF}(2^{31}-1)$, consisting of 16 31-bit cells $(s_{15}, s_{14}, \cdots, s_1, s_0)$ defined over the set $\{1, 2, \cdots, 2^{31}-1\}$. A bit BR, which extracts the content of the LFSR to form 4 32-bit words, (X_0, X_1, X_2, X_3), used in the following FSM and output processing. There are 2 32-bit words R_1 and R_2 used as the memory in the FSM.

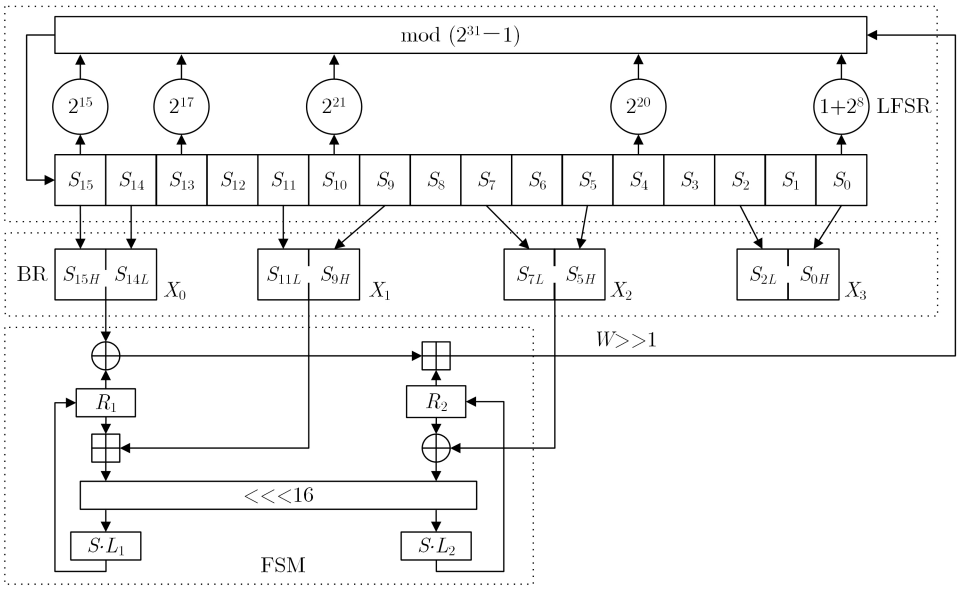

Figure 2 Initialization of ZUC-256 stream cipher

The key/IV loading scheme of ZUC-256 stream cipher is as follows.

$$s_0 = K_0 \parallel d_0 \parallel K_{21} \parallel K_{16} \qquad s_8 = K_8 \parallel (d_8 \mid \mathrm{IV}_{20}) \parallel \mathrm{IV}_3 \parallel \mathrm{IV}_{11}$$

$$s_1 = K_1 \parallel d_1 \parallel K_{22} \parallel K_{17} \qquad s_9 = K_9 \parallel (d_9 \mid \mathrm{IV}_{21}) \parallel \mathrm{IV}_{12} \parallel \mathrm{IV}_4$$

$$s_2 = K_2 \parallel d_2 \parallel K_{23} \parallel K_{18} \qquad s_{10} = \mathrm{IV}_5 \parallel (d_{10} \mid \mathrm{IV}_{22}) \parallel K_{10} \parallel K_{28}$$

$$s_3 = K_3 \parallel d_3 \parallel K_{24} \parallel K_{19} \qquad s_{11} = K_{11} \parallel (d_{11} \mid \mathrm{IV}_{23}) \parallel \mathrm{IV}_6 \parallel \mathrm{IV}_{13}$$

$$s_4 = K_4 \parallel d_4 \parallel K_{25} \parallel K_{20} \qquad s_{12} = K_{12} \parallel (d_{12} \mid \mathrm{IV}_{24}) \parallel \mathrm{IV}_7 \parallel \mathrm{IV}_{14}$$

$$s_5 = \mathrm{IV}_0 \parallel (d_5 \mid \mathrm{IV}_{17}) \parallel K_5 \parallel K_{26} \qquad s_{13} = K_{13} \parallel d_{13} \parallel \mathrm{IV}_{15} \parallel \mathrm{IV}_8$$

$$s_6 = \mathrm{IV}_1 \parallel (d_6 \mid \mathrm{IV}_{18}) \parallel K_6 \parallel K_{27} \qquad s_{14} = K_{14} \parallel (d_{14} \mid (K_{31})_H^4) \parallel \mathrm{IV}_{16} \parallel \mathrm{IV}_9$$

$$s_7 = \mathrm{IV}_{10} \parallel (d_7 \mid \mathrm{IV}_{19}) \parallel K_7 \parallel \mathrm{IV}_2 \qquad s_{15} = K_{15} \parallel (d_{15} \mid (K_{31})_L^4) \parallel K_{30} \parallel K_{29}$$

where $(K_{31})_H^4$ is the high 4 bits of the byte K_{31} and $(K_{31})_L^4$ is the low 4 bits of K_{31}, and the constants d_i for $0 \leqslant i \leqslant 15$ are defined as follows.

$d_0 = 0100010$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100100$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

There are $32 + 1 = 33$ rounds of initialization in the ZUC-256 stream cipher, which is depicted as follows.

1. Load the key, IV, and constants into the LFSR as specified above.
2. Let $R_1 = R_2 = 0$.
3. for $i = 0$ to 31 do

   Bitreorganization( )

   $W = F(X_0, X_1, X_2)$

LFSRWithInitializationMode($W \gg 1$)

4. Bitreorganization( )
   $W = F(X_0, X_1, X_2)$ and discard W
   LFSRWithworkMode()

Now we specify the relevant subroutines one-by-one.

LFSRWithInitializationMode(u)

1. $v = (2^{15} \cdot s_{15} + 2^{17} \cdot s_{13} + 2^{21} \cdot s_{10} + 2^{20} \cdot s_4 + (1 + 2^8) \cdot s_0) \bmod (2^{31} - 1)$
2. if $v = 0$ then set $v = 2^{31} - 1$
3. $s_{16} = v + u \bmod (2^{31} - 1)$
4. if $s_{16} = 0$ then set $s_{16} = 2^{31} - 1$
5. $(s_{16}, s_{15}, \cdots, s_2, s_1) \rightarrow (s_{15}, s_{14}, \cdots, s_1, s_0)$, where $\rightarrow$ is the assignment operation

LFSRWithworkMode()

1. $s_{16} = 2^{15} \cdot s_{15} + 2^{17} \cdot s_{13} + 2^{21} \cdot s_{10} + 2^{20} \cdot s_4 + (1 + 2^8) \cdot s_0 \bmod (2^{31} - 1)$
2. if $s_{16} = 0$ then set $s_{16} = 2^{31} - 1$
3. $(s_{16}, s_{15}, \cdots, s_2, s_1) \rightarrow (s_{15}, s_{14}, \cdots, s_1, s_0)$

Bitreorganization()

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. $X_3 = s_{2L} \parallel s_{0H}$

where s_{iH} is the high 16 bits of the cell s_i and s_{jL} is the low 16 bits of the cell s_j.

$F(X_0, X_1, X_2)$

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

where $S = (S_0, S_1, S_0, S_1)$ is the 4 parallel S-boxes which are the same as those used in the previous ZUC-128, while L_1 and L_2 are the two MDS matrices used in the ZUC-128. The ZUC-256 stream cipher generates a 32-bit keystream word at each time instant.

KeystreamGeneration()

1. Bitreorganization( )
2. $Z = F(X_0, X_1, X_2) \oplus X_3$
3. LFSRWithworkMode()

ZUC-256 generates 20 000-bit to 2^{32}-bit keystream for each frame, i.e., for each frame it produces 625 to 2^{27} keystream words; after that a key/IV re-synchronization is performed with the key/constants fixed and the IV changing into a new value.

In the 5G applications, the MAC generation algorithm of ZUC-256 is similar to that of ZUC-128, which is described as follows. Let $M = (m_0, m_1, \cdots, m_{l-1})$ be the l-bit length plaintext message and the size t of the tag is selectively to be of 32, 64, and 128 bits.

MAC_Generation(M)

1. Let ZUC-256 produce a keystream of $L = \lceil \frac{l}{32} \rceil + 2 \cdot \frac{t}{32}$ words. Denote the keystream bit string by $z_0, z_1, \cdots, z_{32 \cdot L-1}$, where z_0 is the most significant bit of the first output keystream word and z_{31} is the least significant bit of the keystream word.
2. Initialize Tag $= (z_0, z_1, \cdots, z_{t-1})$
3. for $i = 0$ to $l - 1$ do
   let $W_i = (z_{t+i}, \cdots, z_{i+2t-1})$
   if $m_i = 1$ then Tag $=$ Tag $\oplus W_i$
4. $W_l = (z_{l+t}, \cdots, z_{l+2t-1})$
5. Tag $=$ Tag $\oplus W_l$
6. return Tag

For the different sizes of the MAC tag, to prevent the forgery attack, the constants are specified as follows.

1. For the tag size of 32 bits, the constants are

$d_0 = 0100010$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100101$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

2. For the tag size of 64 bits, the constants are

$d_0 = 0100011$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100100$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

3. For the tag size of 128 bits, the constants are

$d_0 = 0100011$	$d_4 = 1101101$	$d_8 = 1000000$	$d_{12} = 1000000$
$d_1 = 0101111$	$d_5 = 1000000$	$d_9 = 1000000$	$d_{13} = 1010010$
$d_2 = 0100101$	$d_6 = 1000000$	$d_{10} = 1000000$	$d_{14} = 0010000$
$d_3 = 0101010$	$d_7 = 1000000$	$d_{11} = 1000000$	$d_{15} = 0110000$

The test vectors of the ZUC-256 stream cipher for the keystream generation phase are as follows.

1. Let $K_i =$ `0x00` for $0 \leqslant i \leqslant 31$ and $\mathrm{IV}_i =$ `0x00` for $0 \leqslant i \leqslant 24$, then the first 20 keystream words are
   - `58d03ad6,2e032ce2,dafc683a,39bdcb03,52a2bc67,`
   - `f1b7de74,163ce3a1,01ef5558,9639d75b,95fa681b,`
   - `7f090df7,56391ccc,903b7612,744d544c,17bc3fad,`
   - `8b163b08,21787c0b,97775bb8,4943c6bb,e8ad8afd`

2. Let K_i = `0xff` for $0 \leqslant i \leqslant 31$ and IV_i = `0xff` for $0 \leqslant i \leqslant 16$ and IV_i = `0x3f` for $17 \leqslant i \leqslant 24$, then the first 20 keystream words are
   - `3356cbae,d1a1c18b,6baa4ffe,343f777c,9e15128f,`
   - `251ab65b,949f7b26,ef7157f2,96dd2fa9,df95e3ee,`
   - `7a5be02e,c32ba585,505af316,c2f9ded2,7cdbd935,`
   - `e441ce11,15fd0a80,bb7aef67,68989416,b8fac8c2`

The test vectors of the ZUC-256 stream cipher for the tag authentication phase are as follows.

1. Let K_i = `0x00` for $0 \leqslant i \leqslant 31$ and IV_i = `0x00` for $0 \leqslant i \leqslant 24$, $M = 0x\underbrace{00,\cdots,00}_{100}$ with the length $l = 400$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are as follows, respectively.
   - The 32-bit authentication tag is `9b972a74`
   - The 64-bit authentication tag is `673e5499 0034d38c`
   - The 128-bit authentication tag is `d85e54bb cb960096 7084c952 a1654b26`
2. Let K_i = `0x00` for $0 \leqslant i \leqslant 31$ and IV_i = `0x00` for $0 \leqslant i \leqslant 24$, $M = 0x\underbrace{11,\cdots,11}_{1000}$ with the length $l = 4000$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are as follows, respectively.
   - The 32-bit authentication tag is `8754f5cf`
   - The 64-bit authentication tag is `130dc225 e72240cc`
   - The 128bit authentication tag is `df1e8307 b31cc62b eca1ac6f 8190c22f`
3. Let K_i = `0xff` for $0 \leqslant i \leqslant 31$ and IV_i = `0xff` for $0 \leqslant i \leqslant 16$ and IV_i = `0x3f` for $17 \leqslant i \leqslant 24$, $M = 0x\underbrace{00,\cdots,00}_{100}$ with the length $l = 400$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are as follows, respectively.
   - The 32-bit authentication tag is `1f3079b4`
   - The 64-bit authentication tag is `8c71394d 39957725`
   - The 128-bit authentication tag is `a35bb274 b567c48b 28319f11 1af34fbd`
4. Let K_i = `0xff` for $0 \leqslant i \leqslant 31$ and IV_i = `0xff` for $0 \leqslant i \leqslant 16$ and IV_i = `0x3f` for $17 \leqslant i \leqslant 24$, $M = 0x\underbrace{11,\cdots,11}_{1000}$ with the length $l = 4000$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are as follows, respectively.
   - The 32-bit authentication tag is `5c7c8b88`
   - The 64-bit authentication tag is `ea1dee54 4bb6223b`
   - The 128-bit authentication tag is `3a83b554 be408ca5 494124ed 9d473205`

The security claim of the ZUC-256 stream cipher is the 256-bit security in the 5G application settings. For the forgery attacks on the authentication part, the security level is the same as the tag size and the IV is not allowed to be re-used. If the tag verification failed, no output should be generated.

3 Conclusions

In this paper, we have presented the details of the new ZUC-256 stream cipher. Any cryptanalysis is welcome.

References

[1] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 and 128-EIA3, Document 4: Design and Evaluation Reprot.
http://www.gsmworld.com/documents/EEA3_EIA3_Design_Evaluation_v1_1.pdf.

Corresponding author: ZHANG Bin, E-mail: zhangbin@tca.iscas.ac.cn