ZUC-256流密码

Abstract. 本文给出ZUC-256流密码的完整描述。ZUC-256是3GPP机密性与完整性算法128-EEA3与128-EIA3中采用的ZUC-128的256比特密钥升级版本。ZUC-256的设计目标是在5G的应用环境下提供256比特的安全性；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。

Keywords: 流密码，祖冲之算法，256-比特安全性

1 引言

众所周知[1]，3GPP机密性与完整性算法128-EEA3和128-EIA3的核心是ZUC-128流密码算法。随着通信与计算技术的发展，对未来5G应用环境下提供256比特安全性的新型流密码算法的需求越发迫切。本文提出ZUC-256流密码，在保持与ZUC-128算法高度兼容的基础上，同时满足5G的应用环境。与ZUC-128相比，新的ZUC-256算法在初始化阶段、消息认证码(MAC)生成阶段采用了新的设计方案。

本文结构如下：在第二节，我们首先给出ZUC-256流密码的详细描述，包含了初始化阶段、密钥流生成阶段及消息认证码(MAC)生成阶段；第三节总结了全文。

2 算法描述

在这一节，我们给出ZUC-256流密码的详细描述，首先约定下列符号。

- 记整数的模2^{32}加法为$\boxplus$，即对于$0 \leq x < 2^{32}$和$0 \leq y < 2^{32}$，$x \boxplus y$就是mod 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$1 \leq x \leq 2^{31}-1$，$1 \leq y \leq 2^{31}-1$；
- 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$K = (K_{31}, K_{30}, ..., K_2, K_1, K_0)$为ZUC-256算法中采用的256-比特密钥，其中K_i $(0 \leq i \leq 31)$为8-比特字节；
- 令$IV = (IV_{24}, IV_{23}, ..., IV_{17}, IV_{16}, IV_{15}, \ldots, IV_1, IV_0)$为ZUC-256算法采用的184-比特初始向量，其中IV_i $(0 \leq i \leq 16)$为8-比特字节；IV_i $(17 \leq i \leq 24)$为6-比特长的比特串，每个占据一个字节的低6位；
- 令$d_i (0 \leq i \leq 15)$为ZUC-256算法采用的7-比特常数；
- 记64-比特操作数的向左循环移位为$\lll$，其中$x \lll n$即为$((x \ll n) \mid (x \gg (64-n)))$。

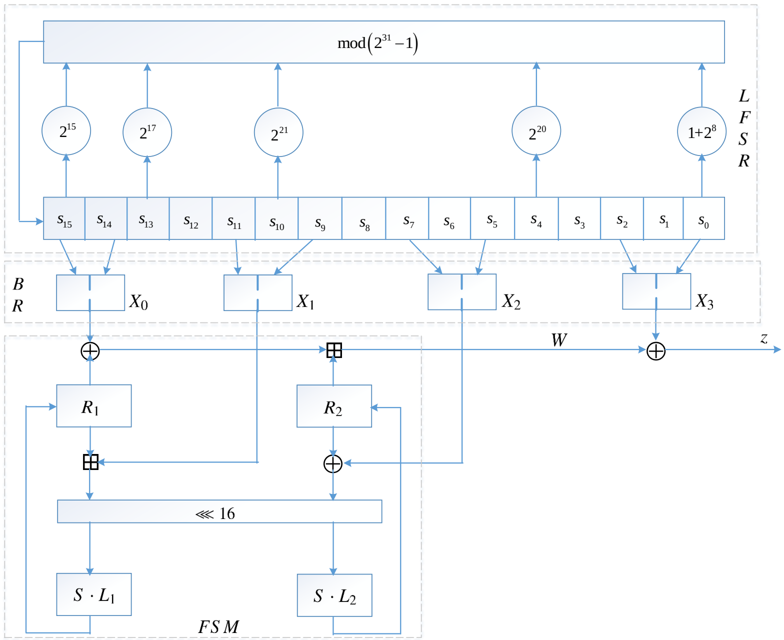

Fig. 1. ZUC-256流密码的密钥流生成阶段

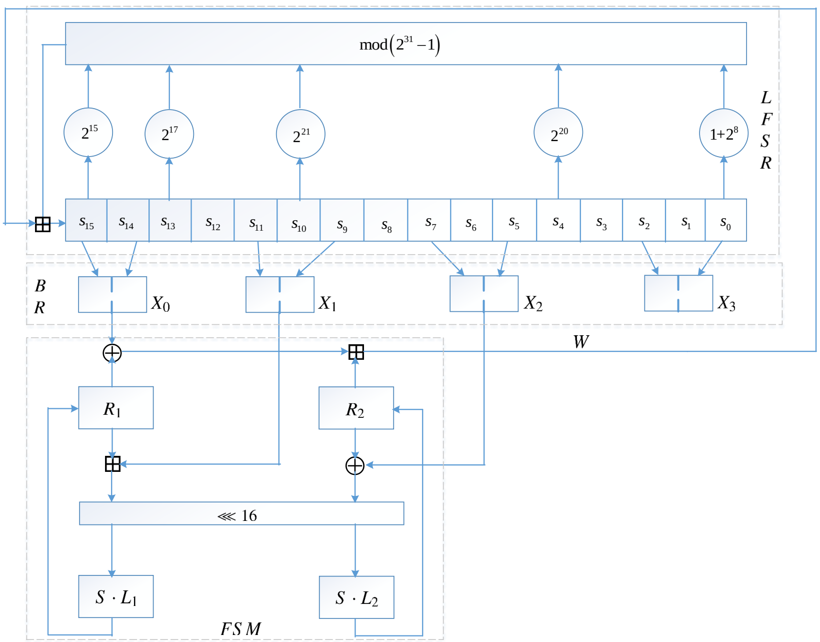

Fig. 2. ZUC-256流密码的初始化阶段

如图1与图2所示，ZUC-256算法由3部分组成: 上部是一个496-比特长的线性反馈移位寄存器(LFSR)，该LFSR定义在域GF($2^{31}-1$) 上，由16个31-比特寄存器单元($s_{15}, s_{14}, \cdots, s_2, s_1, s_0$)构成，这些单元均定义在代表元集合$\{1, 2, \cdots, 2^{31}-1\}$上；中部是一个比特重组层(BR)，主要用来从LFSR中抽取一些寄存器单元内容，并拼接成4个32-比特的字(X_0, X_1, X_2, X_3)，以用于下部的有限状态自动机(FSM)；下部是FSM层，主要包含2个32-比特字R_1与R_2，作为FSM中的记忆单元。

ZUC-256的密钥/初始向量装载算法如下。

$$
\begin{aligned}
s_0 &= K_0 \parallel d_0 \parallel K_{21} \parallel K_{16} \\
s_1 &= K_1 \parallel d_1 \parallel K_{22} \parallel K_{17} \\
s_2 &= K_2 \parallel d_2 \parallel K_{23} \parallel K_{18} \\
s_3 &= K_3 \parallel d_3 \parallel K_{24} \parallel K_{19} \\
s_4 &= K_4 \parallel d_4 \parallel K_{25} \parallel K_{20} \\
s_5 &= IV_0 \parallel (d_5 \mid IV_{17}) \parallel K_5 \parallel K_{26} \\
s_6 &= IV_1 \parallel (d_6 \mid IV_{18}) \parallel K_6 \parallel K_{27} \\
s_7 &= IV_{10} \parallel (d_7 \mid IV_{19}) \parallel K_7 \parallel IV_2 \\
s_8 &= K_8 \parallel (d_8 \mid IV_{20}) \parallel IV_3 \parallel IV_{11} \\
s_9 &= K_9 \parallel (d_9 \mid IV_{21}) \parallel IV_{12} \parallel IV_4 \\
s_{10} &= IV_5 \parallel (d_{10} \mid IV_{22}) \parallel K_{10} \parallel K_{28} \\
s_{11} &= K_{11} \parallel (d_{11} \mid IV_{23}) \parallel IV_6 \parallel IV_{13} \\
s_{12} &= K_{12} \parallel (d_{12} \mid IV_{24}) \parallel IV_7 \parallel IV_{14} \\
s_{13} &= K_{13} \parallel d_{13} \parallel IV_{15} \parallel IV_8 \\
s_{14} &= K_{14} \parallel (d_{14} \mid (K_{31})_H^4) \parallel IV_{16} \parallel IV_9 \\
s_{15} &= K_{15} \parallel (d_{15} \mid (K_{31})_L^4) \parallel K_{30} \parallel K_{29},
\end{aligned}
$$

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

$$
\begin{aligned}
d_0 &= 0100010 \\
d_1 &= 0101111 \\
d_2 &= 0100100 \\
d_3 &= 0101010 \\
d_4 &= 1101101 \\
d_5 &= 1000000 \\
d_6 &= 1000000 \\
d_7 &= 1000000 \\
d_8 &= 1000000 \\
d_9 &= 1000000
\end{aligned}
$$

$$d_{10} = 1000000$$
$$d_{11} = 1000000$$
$$d_{12} = 1000000$$
$$d_{13} = 1010010$$
$$d_{14} = 0010000$$
$$d_{15} = 0110000.$$

ZUC-256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( )
   - $Z = F(X_0, X_1, X_2)$
   - LFSRWithInitializationMode($Z \gg 1$)
4. - Bitreorganization( )
   - $Z = F(X_0, X_1, X_2)$，但舍弃Z
   - LFSRWithworkMode()。

下面，我们逐一给出各个相关子程序的描述。

LFSRWithInitializationMode(u)

1. $v = 2^{15} \cdot s_{15} + 2^{17} \cdot s_{13} + 2^{21} \cdot s_{10} + 2^{20} \cdot s_4 + (1+2^8) \cdot s_0 \mod(2^{31}-1)$
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. $(s_{16}, s_{15}, \cdots, s_2, s_1) \to (s_{15}, s_{14}, \cdots, s_1, s_0)$.

LFSRWithworkMode()

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

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算法的同一部件完全相同；而L_1与L_2亦是ZUC-128中采用的两个MDS矩阵。ZUC-256算法每一节拍产生一个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应用中，ZUC-256每一帧产生20000-比特的密钥流，亦即每一帧产生625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的消息认证码生成算法如下。令$M = (m_0, m_1, \cdots, m_{l-1})$为$l$-比特长的明文消息，认证标签长度$t$可取为32, 64及128比特。

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

对于不同长度的MAC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。

$$
\begin{aligned}
d_0 &= 0100010 \\
d_1 &= 0101111 \\
d_2 &= 0100101 \\
d_3 &= 0101010 \\
d_4 &= 1101101 \\
d_5 &= 1000000 \\
d_6 &= 1000000
\end{aligned}
$$

$$
\begin{aligned}
d_7 &= 1000000\\
d_8 &= 1000000\\
d_9 &= 1000000\\
d_{10} &= 1000000\\
d_{11} &= 1000000\\
d_{12} &= 1000000\\
d_{13} &= 1010010\\
d_{14} &= 0010000\\
d_{15} &= 0110000.
\end{aligned}
$$

2. 对于64比特的标签长度，所采用的常数如下。

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

3. 对于128比特的标签长度，所采用的常数如下。

$$
\begin{aligned}
d_0 &= 0100011\\
d_1 &= 0101111\\
d_2 &= 0100101\\
d_3 &= 0101010\\
d_4 &= 1101101
\end{aligned}
$$

$$
\begin{aligned}
d_5 &= 1000000 \\
d_6 &= 1000000 \\
d_7 &= 1000000 \\
d_8 &= 1000000 \\
d_9 &= 1000000 \\
d_{10} &= 1000000 \\
d_{11} &= 1000000 \\
d_{12} &= 1000000 \\
d_{13} &= 1010010 \\
d_{14} &= 0010000 \\
d_{15} &= 0110000.
\end{aligned}
$$

ZUC-256算法的测试向量如下。首先，对于密钥流生成，有如下测试向量。

1. let $K_i = \texttt{0x00}$ for $0 \leq i \leq 31$ and $IV_i = \texttt{0x00}$ for $0 \leq i \leq 24$, 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 $K_i = \texttt{0xff}$ for $0 \leq i \leq 31$ and $IV_i = \texttt{0xff}$ for $0 \leq i \leq 16$ and $IV_i = \texttt{0x3f}$ for $17 \leq i \leq 24$, 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`

其次，对于消息认证码生成阶段，测试向量如下。

1. let $K_i = \texttt{0x00}$ for $0 \leq i \leq 31$ and $IV_i = \texttt{0x00}$ for $0 \leq i \leq 24$, $M = 0x\underbrace{00, \cdots, 00}_{100}$ with the length $l = 400$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit mac is `9b972a74`
   - The 64-bit mac is `673e5499 0034d38c`
   - The 128-bit mac is `d85e54bb cb960096 7084c952 a1654b26`
2. let $K_i = \texttt{0x00}$ for $0 \leq i \leq 31$ and $IV_i = \texttt{0x00}$ for $0 \leq i \leq 24$, $M = 0x\underbrace{11, \cdots, 11}_{1000}$ with the length $l = 4000$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit mac is `8754f5cf`
   - The 64-bit mac is `130dc225 e72240cc`
   - The 128-bit mac is `df1e8307 b31cc62b eca1ac6f 8190c22f`

3. let $K_i = \texttt{0xff}$ for $0 \le i \le 31$ and $IV_i = \texttt{0xff}$ for $0 \le i \le 16$ and $IV_i = \texttt{0x3f}$ for $17 \le i \le 24$, $M = 0x\underbrace{00, \cdots, 00}_{100}$ with the length $l = 400$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit mac is `1f3079b4`
   - The 64-bit mac is `8c71394d 39957725`
   - The 128-bit mac is `a35bb274 b567c48b 28319f11 1af34fbd`
4. let $K_i = \texttt{0xff}$ for $0 \le i \le 31$ and $IV_i = \texttt{0xff}$ for $0 \le i \le 16$ and $IV_i = \texttt{0x3f}$ for $17 \le i \le 24$, $M = 0x\underbrace{11, \cdots, 11}_{1000}$ with the length $l = 4000$-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit mac is `5c7c8b88`
   - The 64-bit mac is `ea1dee54 4bb6223b`
   - The 128-bit mac is `3a83b554 be408ca5 494124ed 9d473205`

ZUC-256算法的安全性目标是在5G应用环境下提供256比特的安全性。对于认证部分的伪造攻击，ZUC-256可提供相当于标签长度的安全性；这里特别强调，在ZUC-256算法中，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，我们给出了新的ZUC-256算法的详细描述。欢迎密码分析。

References

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