The ZUC-256 Stream Cipher

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 con dentiality and integrity algorithms 128-EEA3 and 128-EIA3 which is highly compatible with the ZUC-128 stream cipher and has its own design features. The aim is a new stream cipher that o ers the 256-bit security for the upcoming applications in 5G. For the authentication, various tag sizes are supported with the IV-respecting restriction.

Keywords: ZUC algorithm, Stream ciphers, 256-bit security.

1 Introduction

The core of the 3GPP con dentiality 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 o ers 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. The new ZUC-256 stream cipher di ers 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 The Description of the 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 , i.e., for $0 \quad x < 2^{32}$ and $0 \quad y < 2^{32}$, $x \quad y$ is the integer addition mod 2^{32}.
- Denote the integer addition modulo $(2^{31} \quad 1)$ by $(x + y) \bmod (2^{31} \quad 1)$ for $1 \quad x \quad 2^{31} \quad 1$ and $1 \quad y \quad 2^{31} \quad 1$.
- Denote the bitwise exclusive OR by .
- Denote the bit string concatenation by k.
- Denote the bitwise logic OR by j.

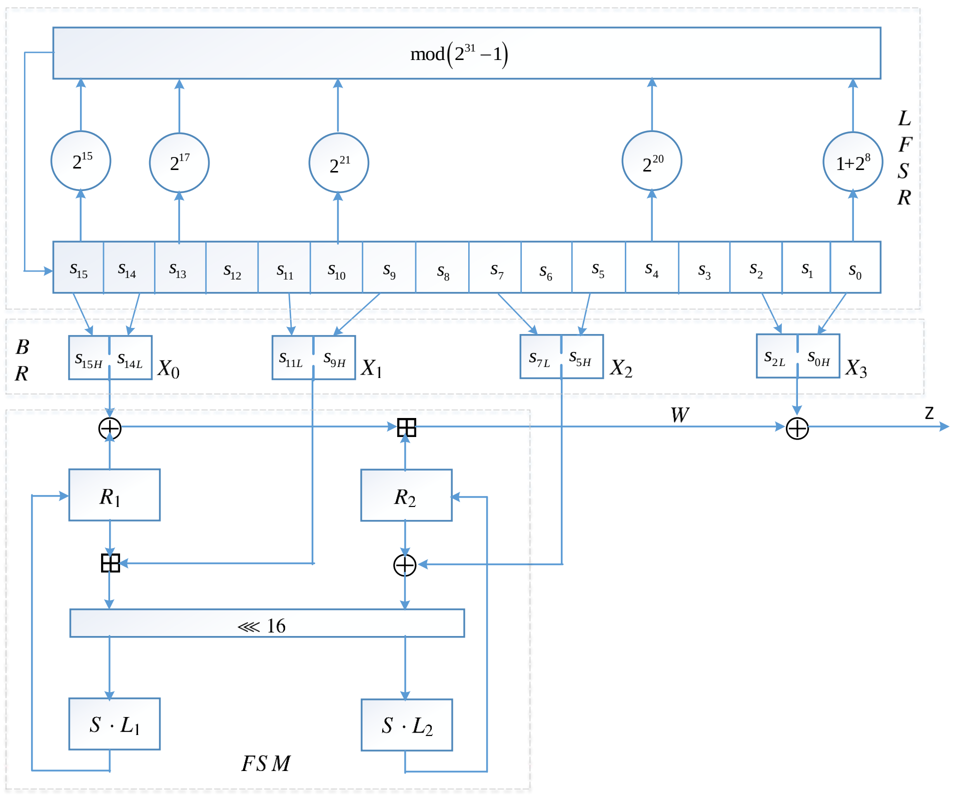

Fig. 1. The keystream generation phase of the ZUC-256 stream cipher

- $K = (K_{31}; K_{30}; :::; K_2; K_1; K_0)$, the 256-bit secret key used in ZUC-256 where K_i for 0 i 31 are 8-bit bytes.
- $IV = (IV_{24}; IV_{23}; :::; IV_{17}; IV_{16}; IV_{15}; :::; IV_1; IV_0)$, the 184-bit initialization vector used in ZUC-256 where IV_i for 0 i 16 are 8-bit bytes and IV_i for 17 i 24 are 6-bit string occupying the 6 least signi cant bits of a byte.
- d_i for 0 i 15 are the 7-bit constants used in the ZUC-256 stream cipher.
- n , the left rotation of a 64-bit operand, x n n means ((x n) j (x (64 n))), where and are the logical left shift and right shift, respectively.

As depicted in Fig.1 and Fig.2, there are 3 parts involved in ZUC-256: a 496-bit linear feedback shift register (LFSR) de ned over the eld GF(2^{31} 1), consisting of 16 31-bit cells ($s_{15}; s_{14};$ $; s_2; s_1; s_0$) de ned over the set $f1; 2;$ $; 2^{31}$ $1g$; a bit reorganization layer (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 nite state machine (FSM); there are 2 32-bit words R_1 and R_2 used as the memory in the FSM.

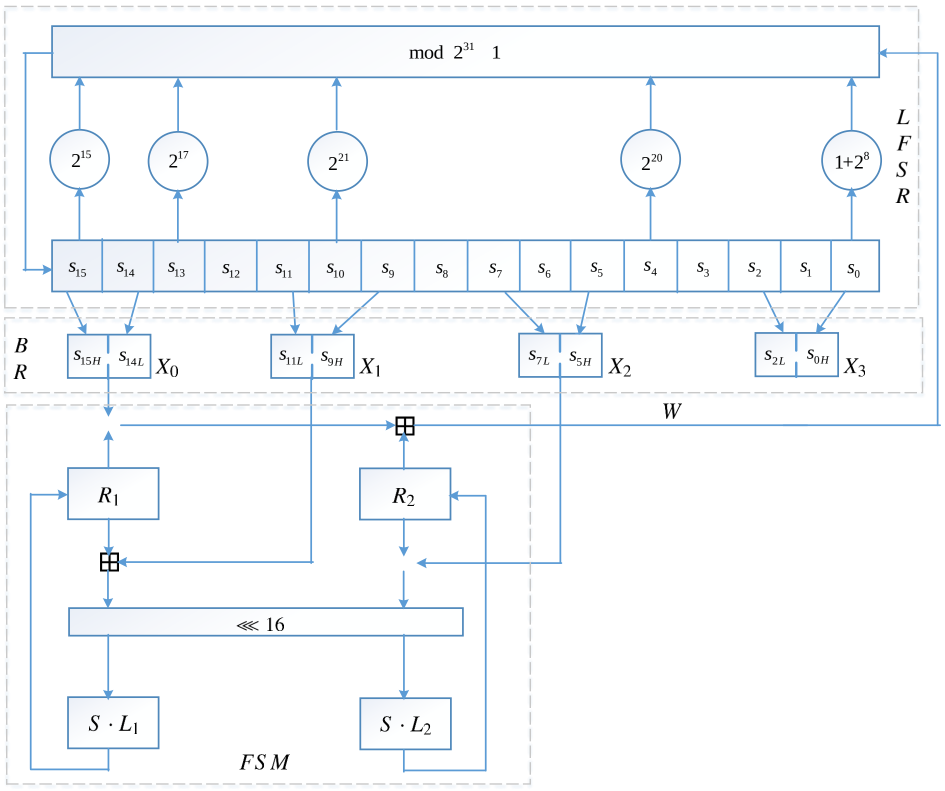

Fig. 2. The initialization phase of the ZUC-256 stream cipher

The Key/IV loading scheme of ZUC-256 is as follows.

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

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

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

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 speci ed 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($W$ 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. $v = 2^{15}\ s_{15} + 2^{17}\ s_{13} + 2^{21}\ s_{10} + 2^{20}\ s_4 + (1 + 2^8)\ s_0\ \mathrm{mod}(2^{31}\ 1)$
2. if $v = 0$ then set $v = 2^{31}\ 1$
3. $s_{16} = v + u\ \mathrm{mod}(2^{31}\ 1)$
4. if $s_{16} = 0$ then set $s_{16} = 2^{31}\ 1$
5. $(s_{16}; s_{15};\ ; s_2; s_1)\ !\ (s_{15}; s_{14};\ ; s_1; s_0)$, where ! is the assignment op-
   eration.
```

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. if $s_{16} = 0$ then set $s_{16} = 2^{31}\ \ 1$
3. $(s_{16}, s_{15}, \ \ , s_2, s_1) \to (s_{15}, s_{14}, \ \ , s_1, s_0)$.

Bitreorganization()

1. $X_0 = s_{15H} \| s_{14L}$
2. $X_1 = s_{11L} \| s_{9H}$
3. $X_2 = s_{7L} \| s_{5H}$
4. $X_3 = s_{2L} \| s_{0H}$,

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. $W = (X_0\ \ R_1)\ \ R_2$
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. $R_2 = S(L_2(W_{2L} \| W_{1H}))$,

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 and 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. $Z = F(X_0, X_1, X_2)\ \ X_3$
3. LFSRWithworkMode().

ZUC-256 generates 20000-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 xed 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 $M = (m_0, m_1, \ \ , m_{l\ 1})$ 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)

bS 1. Let ZUC-256 produce a keystream of $L = \lceil \frac{l}{32} \rceil + 2\ \ \frac{t}{32}$ words. Denote the

```
3. for $i$ = 0 to $l$   1 do
   { let $W_i = (z_{t+i};\quad ; z_{i+2t\ \ 1})$
   { if $m_i = 1$ then $Tag = Tag \quad W_i$
4. $W_l = (z_{l+t};\quad ; z_{l+2t\ \ 1})$
5. $Tag = Tag \quad W_l$
6. return $Tag$
```

For the di erent sizes of the MAC tag, to prevent the forgery attack, the constants are speci ed as follows.

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

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

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

$$\begin{aligned}
d_0 &= 0100011\\
d_1 &= 0101111\\
d_2 &= 0100100\\
d_3 &= 0101010\\
d_4 &= 1101101\\
d_5 &= 1000000\\
d_6 &= 1000000\\
d_7 &= 1000000\\
d_8 &= 1000000
\end{aligned}$$

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

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

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

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

1. let K_i = `0x00` for $0 \quad i \quad 31$ and IV_i = `0x00` for $0 \quad i \quad 24$, then the rst 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 $0 \quad i \quad 31$ and IV_i = `0xff` for $0 \quad i \quad 16$ and IV_i = `0x3f` for $17 \quad i \quad 24$, then the rst 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 i 31 and IV_i = 0x00 for 0 i 24, M = $0x\underbrace{00;\ ;00}_{100}$ 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 i 31 and IV_i = 0x00 for 0 i 24, M = $0x\underbrace{11;\ ;11}_{1000}$ 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 128-bit authentication tag is df1e8307 b31cc62b eca1ac6f 8190c22f
3. let K_i = 0xff for 0 i 31 and IV_i = 0xff for 0 i 16 and IV_i = 0x3f for 17 i 24, M = $0x\underbrace{00;\ ;00}_{100}$ 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 i 31 and IV_i = 0xff for 0 i 16 and IV_i = 0x3f for 17 i 24, M = $0x\underbrace{11;\ ;11}_{1000}$ 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 veri cation 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. Speci cation of the 3GPP Con dentiality 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.

A Document History

25-01-2018	Online publication	version 1.0
15-04-2018	Revision to the Figures	version 1.1