# A New Initialization Scheme of the ZUC-256 Stream Cipher

ZUC Design Team
Chinese Academy of sciences

contact email: martin_zhangbin@hotmail.com

**Abstract.** ZUC-256 stream cipher, together with AES-256 and SNOW 5G, are specified as the future 3GPP confidentiality and integrity algorithms for the air interface by SAGE recently. In this short paper, we describe a new initialization scheme of ZUC-256 with 48 rounds that supports an IV of the exact 128 bits as a positive response to SAGE's recommendation. Compared to the original initialization scheme, this new key/IV setup algorithm avoids the division of the whole key/IV byte and provides a simple and natural-looking initialization scheme for ZUC-256 with a large expected security margin.

**Keywords:** 5G, Stream ciphers, ZUC, 256-bit security.

## 1 Introduction

The core of the 3GPP confidentiality and integrity algorithms 128-EEA3 and 128-EIA3 is the ZUC-128 stream cipher [1]. ZUC-256 is a new member in the ZUC family of stream ciphers, formally proposed in 2018 for the intended usage in the upcoming 5G applications for 3GPP. ZUC-256 stream cipher is industrial-friendly, differing from ZUC-128 only in the initialization phase and in the message authentication codes (MAC) generation phase. The first publicized version of ZUC-256 works with a 256-bit key and a 184-bit initialization vector (IV) and generates a keystream frame of length from 20000 to $2^{32}$ bits after each mixture of the (key, IV) pair.

After the publication of ZUC-256, there is an increasing interest of evaluating its security against various cryptanalytic approaches. At the ZUC-256 international conference in 2018 [7], there are several talks that analyzed different aspects of its security, all of which imply that ZUC-256 is secure against the corresponding cryptanalysis method. Then, a linear distinguishing attack is presented in [5] at FSE 2020, requiring a exceptionally-long keystream frame, which is out of the security claim in [8], as analyzed in [6]. For the analysis of the initialization phase, there is a differential analysis published in [4] and a new cryptanalysis result using modular differences in [3]. In November 2022, SAGE has released a new liaison [2] which specifies AES-256, SNOW 5G and ZUC-256 as the 3GPP 256-bit Confidentiality and Integrity Algorithms for the Air interface and gives the recommendation on ZUC-256 to work with a 48-round

of initialization scheme for a large expected security margin, compared to the scheme presented in [9].

As a positive response to SAGE's recommendation, we propose a new initialization scheme of ZUC-256 that works with a 256-bit key and a 128-bit IV which goes through a 48-round initialization scheme in this paper. This new key/IV setup scheme avoids the division of the whole key/IV byte, is simple and natural-looking, and also provides the 256-bit security in 5G applications with a large expected security margin. A brief cryptanalysis of the new initialization scheme is also yielded.

This paper is structured as follows. In Section 2, we give the detailed description of ZUC-256 with the new initialization scheme. The cryptanalysis related to the new introduced change will be discussed in Section 3. Finally, some conclusions are drawn in Section 4.

## 2 The Description of the New Initialization Scheme for ZUC-256

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

- Denote the integer modular addition by , i.e., for 0 $x < 2^{32}$ and 0 $y <$ $2^{32}$, $x$ $y$ is the integer addition mod $2^{32}$.
- Denote the integer addition modulo $2^{31}$ 1 by $x + y$ mod ($2^{31}$ 1) for 1 $x$ $2^{31}$ 1 and 1 $y$ $2^{31}$ 1.
- Denote the bitwise exclusive OR by .
- Denote the bit string concatenation by *k*.
- $K = (K_{31}, K_{30}, ..., K_2, K_1, K_0)$, the 256-bit secret key used in the ZUC-256 where $K_i$ for 0 $i$ 31 are 8-bit bytes.
- $IV = (IV_{15}, \ldots, IV_1, IV_0)$, the 128-bit initialization vector used in the ZUC-256 where $IV_i$ for 0 $i$ 15 are 8-bit bytes.
- $d_i$ 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 $n$))).

As depicted in Fig.1, 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 *f*$1, 2,$ $, 2^{31}$ 1*g*; 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);1

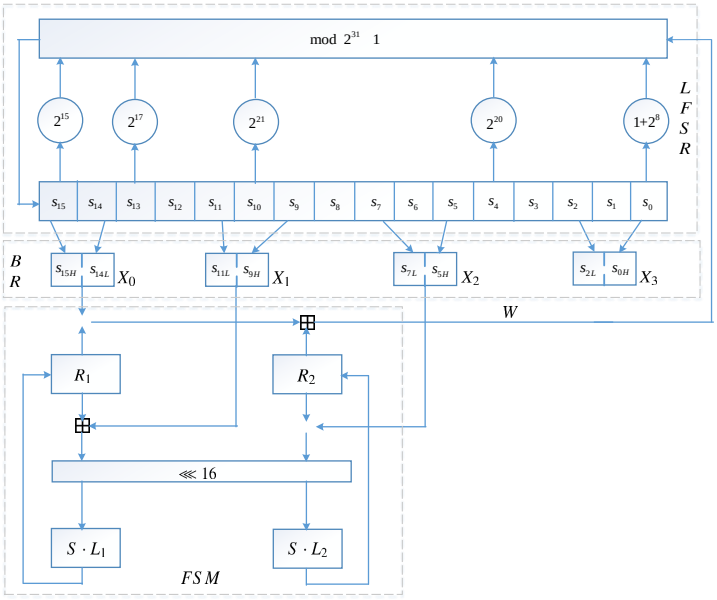


Fig. 1. The initialization phase of the ZUC-256 stream cipher

The Key/IV loading scheme is as follows.

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

where the constants $d_i$ for 0 $i$ 15 are de ned as follows, which are based on the binary expansion of $\pi$ including the integer part.

$d_0$ = 1100100
$d_1$ = 1000011
$d_2$ = 1111011
$d_3$ = 0101010
$d_4$ = 0010001
$d_5$ = 0000101
$d_6$ = 1010001
$d_7$ = 1000010
$d_8$ = 0011010
$d_9$ = 0110001
$d_{10}$ = 0011000
$d_{11}$ = 1100110
$d_{12}$ = 0010100
$d_{13}$ = 0101110
$d_{14}$ = 0000001
$d_{15}$ = 1011100.

Note that there is no hidden weakness introduced in the above constants. There are 48 + 1 = 49 rounds of initialization in this new key/IV setup scheme of ZUC-256, 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 47 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 + $2^8$) $s_0$ mod($2^{31}$ 1)
2. if $v$ = 0 then set $v$ = $2^{31}$ 1
3. $s_{16}$ = $v$ + $u$ mod($2^{31}$ 1)
4. if $s_{16}$ = 0 then set $s_{16}$ = $2^{31}$ 1
5. $(s_{16}, s_{15},$ $, s_2, s_1)$ ! $(s_{15}, s_{14},$ $, s_1, s_0)$.

LFSRWithworkMode()

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

Bitreorganization()

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

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. $W = (X_0 \quad R_1) \quad R_2$ is the FSM output
2. $W_1 = R_1 \quad X_1$
3. $W_2 = R_2 \quad X_2$
4. $R_1 = S(L_1(W_{1L} \; k \; W_{2H}))$
5. $R_2 = S(L_2(W_{2L} \; k \; W_{1H}))$,

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

ZUC-256 generates from 20000-bit up to $2^{32}$-bit keystream for each frame; after that a key/IV re-synchronization is performed with the key/constants xed and the IV changing into a new value.

The MAC generation algorithm of ZUC-256 is as follows. Let $M = (m_0, m_1, \quad , m_{l \quad 1})$ 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 $L = d\frac{l}{32}e + 2 \quad \frac{t}{32}$ words. Denote the keystream bit string by $z_0, z_1, \quad , z_{32 \; L \quad 1}$, where $z_0$ is the most signi cant bit of the rst output keystream word and $z_{31}$ is the least signi cant bit of the rst keystream word.

2. Initialize $Tag$ = $(z_0, z_1, \quad , z_{t\ 1})$
3. for $i$ = 0 to $l$ 1 do
   { let $W_i$ = $(z_{t+i}, \quad , z_{i+2t\ 1})$
   { if $m_i$ = 1 then $Tag$ = $Tag$ $W_i$
4. $W_l$ = $(z_{l+t}, \quad , z_{l+2t\ 1})$
5. $Tag$ = $Tag$ $W_l$
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

$$
\begin{aligned}
d_0 &= 1100100\\
d_1 &= 1000011\\
d_2 &= 1111010\\
d_3 &= 0101010\\
d_4 &= 0010001\\
d_5 &= 0000101\\
d_6 &= 1010001\\
d_7 &= 1000010\\
d_8 &= 0011010\\
d_9 &= 0110001\\
d_{10} &= 0011000\\
d_{11} &= 1100110\\
d_{12} &= 0010100\\
d_{13} &= 0101110\\
d_{14} &= 0000001\\
d_{15} &= 1011100.
\end{aligned}
$$

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

$$
\begin{aligned}
d_0 &= 1100101\\
d_1 &= 1000011\\
d_2 &= 1111011\\
d_3 &= 0101010\\
d_4 &= 0010001
\end{aligned}
$$

$d_5$ = 0000101
$d_6$ = 1010001
$d_7$ = 1000010
$d_8$ = 0011010
$d_9$ = 0110001
$d_{10}$ = 0011000
$d_{11}$ = 1100110
$d_{12}$ = 0010100
$d_{13}$ = 0101110
$d_{14}$ = 0000001
$d_{15}$ = 1011100.

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

$d_0$ = 1100101
$d_1$ = 1000011
$d_2$ = 1111010
$d_3$ = 0101010
$d_4$ = 0010001
$d_5$ = 0000101
$d_6$ = 1010001
$d_7$ = 1000010
$d_8$ = 0011010
$d_9$ = 0110001
$d_{10}$ = 0011000
$d_{11}$ = 1100110
$d_{12}$ = 0010100
$d_{13}$ = 0101110
$d_{14}$ = 0000001
$d_{15}$ = 1011100.

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

1. let $K_i$ = 0x00 for 0 $i$ 31 and $IV_i$ = 0x00 for 0 $i$ 15, then the rst 20 keystream words are
   - 0234e932,f0c22292,38853662,aa624def,7f99a4c7,
   - e47a0282,b2fde38d,f4cb89c5,3c17ab18,87ef5093,
   - 15c53d45,af1de542,7d278dbb,839af54e,e9375674,
   - 01d3207e,7f1d6fb3,b5770472,c4f98e41,637788d9

2. let $K_i$ = 0xff for 0 $i$ 31 and $IV_i$ = 0xff for 0 $i$ 15, then the rst 20 keystream words are
   - 3985e2af,3533d429,338580f0,e0d80ce9,0649e5be,
   - 4961b8a2,d23a44d3,9c18ce98,75f7c424,082ecf47,
   - e1d384b8,91ace320,e46f0b16,cf903c77,f097f1a9,
   - 4bcb2079,fb5c6cc1,6e9f3e05,6eff3261,89ea0373

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 $i$ 15, $M$ = $0x\underbrace{00,\ldots,00}_{100}$ with the length $l$ = 400-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit tag is d51f12fc
   - The 64-bit tag is 3f4aaa58 99158f4a
   - The 128-bit tag is cf4bc324 7d0f6ae5 ce498d54 4556c247
2. let $K_i$ = 0x00 for 0 $i$ 31 and $IV_i$ = 0x00 for 0 $i$ 15, $M$ = $0x\underbrace{11,\ldots,11}_{1000}$ with the length $l$ = 4000-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit tag is 55f6c1a1
   - The 64-bit tag is 972be021 f8152288
   - The 128-bit tag is f9d9a922 37cba79b 42394e2c 3df7a9e4
3. let $K_i$ = 0xff for 0 $i$ 31 and $IV_i$ = 0xff for 0 $i$ 15, $M$ = $0x\underbrace{00,\ldots,00}_{100}$ with the length $l$ = 400-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit tag is 5aea7964
   - The 64-bit tag is 11720876 83515f4b
   - The 128-bit tag is 70469592 61c0dc2e e4f88400 5f1e4368
4. let $K_i$ = 0xff for 0 $i$ 31 and $IV_i$ = 0xff for 0 $i$ 15, $M$ = $0x\underbrace{11,\ldots,11}_{1000}$ with the length $l$ = 4000-bit, then the 32-bit tag, 64-bit tag and 128-bit tag are
   - The 32-bit tag is 06637506
   - The 64-bit tag is 7a6cfe5c 74615bfe
   - The 128-bit tag is dd3a4017 357803a5 1c3fb9a5 7a96feda

The security claim of the ZUC-256 stream cipher with the new initialization scheme is the 256-bit security in the 5G application setting. 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 The Analysis Related to the New Change

In this section, we will present the cryptanalysis of the new initialization scheme, other aspects of the security analysis that is not e ected by the newly introduced change will remain the same as before, and will not cover here.

### 3.1 Differential Attacks

Chosen IV/Key attacks aim at the initialization stage of a stream cipher. For a good stream cipher, after the initialization, each bit of the IV/Key should contribute to each bit of the internal states, and any di erence in the IV/Key will result in an almost-uniform and unpredictable di erence in the internal states. In stream cipher domain, it is more frequent to change the IV than to change the key. And since the IV is known to the public, so the chosen-IV attack is more feasible. The main idea of chosen-IV attack is to choose some di erences in some IV bits and study the propagation of the di erences during the initialization of the cipher. To evaluate the di usion of the input di erence in IV e ectively, we

Table 1. Least number of rounds that every bit is inserted into FSM

| Location | Di usion rounds |
|---|---|
| 0 | 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 |
| 1 | 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 |
| 2 | 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 |
| 3 | 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 |
| 4 | 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 |
| 5 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 |
| 6 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 |
| 7 | 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
| 8 | 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 |
| 9 | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 |
| 10 | 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 |
| 11 | 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
| 12 | 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 |
| 13 | 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 |
| 14 | 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 |
| 15 | 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 |

make the following two assumptions:

1. If the input di erence to the FSM is not zero, then the output di erence of FSM is all 1. As L is an MDS matrix and S is non-linear permutation, the di erence is di used su ciently and faster than the LFSR.
2. The modular addition operation is reduced to be the traditional xor. The di usion of di erence in modular addition is related to the values of states.

Meanwhile, the modular addition can be seen as the xor with carry. Thus, it is reasonable to follow this assumption to a certain extent.

In order to evaluate the property of initialization stage comprehensively, we give three kinds of analysis.

Firstly, we want to know the minimum number of iteration steps to guarantee that each bit of feedback is in uenced by each bit in the initialization state. We divide the initialization state into 16 31 = 496 bits and test the least number of rounds that each bit is inserted into FSM. The obtained results are shown in Table 1, where 'Location' denotes the index $i$ of $s_i$ and the leftmost location is denoted by 15. Combining with the key/IV loading procedure, we can conclude that the di erence of IV is di used to FSM after at most four rounds and the memory cells are in uenced after at most ve rounds.

The conclusion mentioned above is tested under two assumptions and may be di erent to the real results. We randomly choose $2^{11}$ $(K, IV)$ pairs to test the di usion property in practice. In details, just exhaustively test all the possible di erences for $i$-th (0 $i$ 15) word of IV to get the maximum iteration steps for causing di erence in memory cells $R_1$ and $R_2$. The results of experiments are summarized as Table 2. We can see that the memory cells are in uenced after

Table 2. Maximum steps to lead di erence in $R_1$ and $R_2$

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $R_1$ | 1 | 2 | 3 | 4 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 2 | 3 | 4 | 5 |
| $R_2$ | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 |

at most ve rounds from Table 2 and the result matches the above conclusion from Table 1.

Secondly, the minimum number of iterations to guarantee that each bit of the register state is in uenced by each bit of initialization state will be focused. In a word, if the input to FSM is di erent, then the state of the feedback word $s_{15}$ will have some di erence in the next round. The di erential state will be shifted to the rightmost location in the next 15 rounds and all the states will be a ected.

We have also made some experiments to evaluate this di usion property in practice. For the injected di erence position on each bit of IV, we have chosen $2^{22}$ (K, IV) pairs to get the least number of steps that all the register states are in uenced. The experimental results show that each bit of register state is a ected by each bit of IV after at most 19 rounds.

Thirdly, we will investigate the di erential characteristic aspect of the initialization scheme when the injected di erence positions covering all the possible key and IV loading positions.

We have searched the minimum number of active S-box of the initialization scheme under the simpli cation that the $2^{31}$ 1 addition operation of the LFSR

Tabl

| Index |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| |
| 16 |

be replaced by the traditional exclusive or. The searching result shows that there are some input key di erences such that the minimum number of active S-box is zero after 11 rounds of initialization, when the hamming weight of the input di erence is restricted to be less than 11. The following are the only 43 input di erence patterns, shown in Table 3. Given that $s_2$ and $s_6$ are chosen to have

Table 4. The di erences of the LFSR in each round of iteration

| Round | $S_{15}$ | $S_{14}$ | $S_{13}$ | $S_{12}$ | $S_{11}$ | $S_{10}$ | $S_9$ | $S_8$ | $S_7$ | $S_6$ | $S_5$ | $S_4$ | $S_3$ | $S_2$ | $S_1$ | $S_0$ | $R_1$ | $R_2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 | B | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 | B | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 | B | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 |
| 7 | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | D | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | E | D | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | F | E | D | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | G | F | E | D | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | H | G | F | E | D | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * |
| 13 | * | H | G | F | E | D | C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | * |
| 14 | * | * | H | G | F | E | D | C | | | | | | | | | | |

Table 4 gives the di erence propagation of the 16 cells of the LFSR after $i$ rounds of iteration ($i$ = 1, 2, , 27) in a step-by-step manner. It is easy to see that the memory cells $R_1$ and $R_2$ have no di erence even after 11 rounds, thus the differences $C$-$H$ are generated only from the $2^{31}$ 1 addition operation. We expect that the state cells $s_{15}$ and $s_{14}$ will have good di erential randomness after 48 rounds. Hence, we believe that, after 49 rounds of iteration, the di erence in the rst 32-bit keystream word will be fairly random and unpredictable. Note that the above analysis will naturally be converted into a related key attack scenario, which is di cult, in stream cipher domain, to detect the related key pairs given only the corresponding keystream segments with the randomly generated IVs.

From the above analysis, we could conclude that the new initialization scheme of ZUC-256 could provide the 256-bit security in 5G application settings with a large expected security margin.

## 4 Conclusions

In this paper, we have presented the details of a new initialization scheme with 48 rounds for the ZUC-256 stream cipher that works with the 128-bit initialization vector. Any cryptanalysis of the new initialization scheme 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.
2. SAGE liaison, Speci cation of the 256-bit air interface algorithms, 14{18th November, 2022.
3. Fukang L., Willi M., Santanu S., Gaoli W., Ryoma I.,Takanori I., New Cryptanalysis of ZUC-256 Initialization Using Modular Di erences, *IACR Trans. Symmetric Cryptol*, 2022(3), pp. 152{190, 2022.
4. Babbage S. and Maximov A., Di erential analysis of the ZUC-256 initialisation, https://eprint.iacr.org/2020/1215.pdf
5. Jing Y., Thomas J., and Alexander M., Spectral analysis of ZUC-256. *IACR Trans. Symmetric Cryptol*, 2020(1), pp. 266{288, 2020.
6. The ZUC design team, On the linear distinguishing attack against ZUC-256 stream cipher, https://eprint.iacr.org/2020/1046.pdf
7. https://eurocrypt.2018.rump.cr.yp.to/f2efa67f85b309013f8506364c002ce5.pdf
8. The ZUC design team. The ZUC-256 Stream Cipher. http://www.is.cas.cn/ztzl2016/zouchongzhi/201801/W020180126529970733243.pdf, 2018.
9. The ZUC design team. An Addendum to the ZUC-256 Stream Cipher. https://eprint.iacr.org/2021/1439.