Analyzing SNOW and ZUC Security Algorithms Using NIST SP 800-22 and Enhancing their Randomness

Zakaria Hassan Abdelwahab[1,*], Talaat A. Elgarf[1] and Abdelhalim Zekry[2]

[1]*Communications Engineering Department, Higher Technological Institute, Ramadan city, Egypt*
[2]*Communications Engineering Department, Faculty of Engineering, Ain shams University, Cairo, Egypt*
E-mail: zakariahassan@windowslive.com; talaat.elgarf@hti.edu.eg; AAAZekry@hotmail.com
[*]*Corresponding Author*

Received 03 May 2020; Accepted 24 November 2020;
Publication 06 February 2021

Abstract

Confidentiality and Integrity algorithms are based on SNOW and ZUC stream cipher algorithms. These standardized algorithms are designed by the 3rd Generation Partnership Project (3GPP) for advanced mobile communication systems (4G-LTE Advanced, LTE Advanced Pro, and 5G-Next Generation). In this paper, twenty configurations of SNOW and ZUC algorithms are studied and analyzed to select the one with the best randomness properties. Each configuration has two different S-boxes in the Finite State Machine (FSM) layer of the SNOW algorithm and Nonlinear Function (NLF) layer of the ZUC algorithm. The two S-boxes are selected from the best five S-boxes published in kinds of literature (Rijndael, Dickson, Feistel structure, New Rijndael, and Improved New Rijndael S-boxes). The NIST SP 800-22 statistical test suite involves 15 tests that are used to assess the randomness properties of each configuration. A complete simulation of each configuration

Journal of Cyber Security and Mobility, Vol. 9_4, 535–576.
doi: 10.13052/jcsm2245-1439.943

SNOW and ZUC with two different S-boxes is applied using C-language. Test results showed that the best pair arrangement of S-boxes in the SNOW algorithm is the configuration (Feistel structure – Rijndael S-boxes) although the standard configuration by 3GPP is (Rijndael – Dickson S-boxes). Also, the best configuration in the ZUC algorithm is (New Rijndael – Rijndael S-boxes) although the standard configuration by 3GPP is (Feistel structure – New Rijndael S-boxes). The best configurations passed all the NIST SP 800-22 suite randomness tests successfully.

Keywords: Mobile security, 4G, 5G, Stream cipher, SNOW algorithm, ZUC algorithm, NIST SP 800-22 statistical analysis test.

1 Introduction

The Confidentiality and Integrity algorithms are applied for the advanced mobile communication systems to secure the user and signaling messages information transmitted over the wireless mobile network in which the kernel structure of these 3GPP standardized algorithms are SNOW and ZUC stream cipher algorithms [1, 2]. The core structure of the SNOW and ZUC algorithms depends on Linear Feedback Shift Registers (LFSR), Finite State Machine (FSM), Nonlinear Function (NLF), Bit Reorganization (BR), and a combination pair of S-boxes (S1, S2) [3, 4]. Each SNOW and ZUC algorithm has two S-boxes in its nonlinear layer structure. SNOW algorithm has S1-box (Rijndael) and S2-box Dickson [3] standardized by 3GPP in its Finite State Machine (FSM) layer while the ZUC algorithm has its two S-boxes as S1-box (Feistel structure) and S2-box (New Rijndael) [4] in its Nonlinear Function (NLF) layer.

During the literature review, it was noticed some threats in the analysis of SNOW [5–7]/ZUC [8] stream cipher algorithms through the statistical evaluation of the short keystream data set attack and cryptanalytic attacks as in SNOW (sliding property attack [9, 10], fault attack [11], cache timing attack [12], multiset collision attack [13], and improved heuristic guess and determine attack [14]), and as in ZUC (alternative algebraic analysis [15, 16] and differential attacks [17]. There is a need to assess such systems that depend on the combination of S-boxes that are used to update the registers in the nonlinear layer structure of these algorithms (FSM-SNOW and NLF-ZUC) to increase their complexity and therefore the time to cryptanalysis. One of the most important parameters is the randomness properties of the output keystream sequences produced by such algorithms.

Up to our knowledge, no recent papers investigate the randomness properties of SNOW and ZUC stream cipher output keystream by using the 3GPP standardized S-boxes in SNOW and ZUC algorithms to select the best pair arrangement of (S1, S2) boxes. In this paper, we analyze the randomness properties of these two algorithms using the standard NIST SP 800-22 test suite. We started with the standard SNOW and ZUC algorithms using the 3GPP original arrangement of the two S-boxes (S1, S2) in the finite state machine part of the algorithm and then test all the different twenty combinations of the two S-boxes (S1, S2) taken from five strong S-boxes (Rijndael, Dickson, Feistel structure, New Rijndael, and Improved New Rijndael S-boxes), to determine the configuration of pair S-boxes (S1, S2) with the best randomness properties.

This paper is organized as follows: In Section 2, modern algorithms applied in the 4th and 5th mobile generation are presented. In Section 3, NIST SP 800-22 randomness tests applied to the output of nonlinear stream ciphers are described. In Section 4, configurations of S-boxes are described. In Section 5, SNOW and ZUC simulation with different S-boxes configurations and NIST SP 800-22 randomness 15-tests verification. Finally, Section 6 concludes this paper.

2 Modern algorithms applied in 4th and 5th Mobile Generation

This section describes the processes of the synchronous stream cipher algorithms (SNOW and ZUC) that are used in the heart core of the confidentiality and integrity algorithms applied in advanced mobile communications.

2.1 Description of SNOW Algorithm

SNOW is a synchronous stream cipher that contains two different processes, the initialization process and the generation of the keystream process, below the rule of 128 bits Ciphering Key/Integrity Key (CK/IK) and 128 bits initialization vector (IV) [3]. The internal structure of the SNOW is collected from the Linear Feedback Shift Register (LFSR), and Finite State machine (FSM) layers. The LFSR layer is created by 16 stages in which each stage consists of 32 bits, the FSM layer is created by three 32 bits registers, nonlinearly clocked (R_1, R_2, and R_3), and the combination of pair S1-box (Rijndael) and S2-box (Dickson) is used in the Finite State machine (FSM) layer [3].

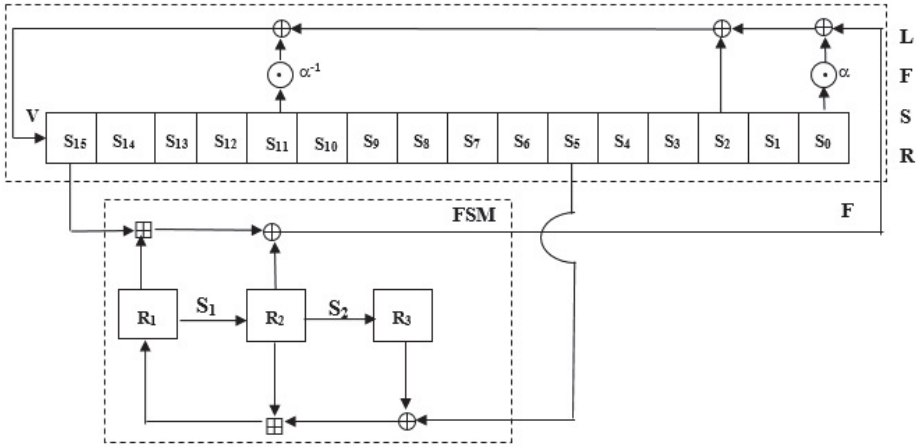


Figure 1 SNOW algorithm during the initialization process [3].

SNOW algorithm during the initialization process is initialized with 128 bits key consisting of four concatenating 32 bits word ($K_0\|K_1\|K_2\|K_3$) and 128 bits initialization variable consisting of four concatenating 32 bits words ($IV_0\|IV_1\|IV_2\|IV_3$) to initiate the internal state of the LFSR registers using a feedback value calculated by a primitive polynomial over the Galois field GF (2^{32}) based on the well-known Mulα and Divα main functions and XOR operations [3]. This process is performed for 32 clock runs without generating an output keystream sequence (as shown in Figure 1). Calculate the intermediate value (V) as following [3]:

$$\begin{aligned} \mathrm{V} = {} & (\mathrm{s}_{0,1}\|\mathrm{s}_{0,2}\|\mathrm{s}_{0,3}\|0\mathrm{x}00) \oplus \mathrm{Mul}\alpha(\mathrm{s}_{0,0}) \oplus \mathrm{s}_2 \\ & \oplus (0\mathrm{x}00\|\mathrm{s}_{11,0}\|\mathrm{s}_{11,1}\|\mathrm{s}_{11,2}) \oplus \mathrm{Div}\alpha(\mathrm{s}_{11,3}) \oplus \mathrm{F} \end{aligned} \tag{1}$$

Clocking the FSM has two input words s_{15} and s_5 from the LFSR. It produces 32 bits output word (F) as [3]:

$$\mathrm{F} = (\mathrm{s}_{15} \boxplus \mathrm{R}_1) \oplus \mathrm{R}_2 \tag{2}$$

where $\boxplus$ is the addition modulo 2^{32}. The update of the FSM is defined by Calculate the intermediate value (r) as follows [3]:

$$\mathrm{r} = \mathrm{R}_2 \boxplus (\mathrm{R}_3 \oplus \mathrm{s}_5) \tag{3}$$

$$\mathrm{R}_3 = \mathrm{S2}(\mathrm{R}_2) \tag{4}$$

$$\mathrm{R}_2 = \mathrm{S1}(\mathrm{R}_1) \tag{5}$$

$$\mathrm{R}_1 = \mathrm{r} \tag{6}$$

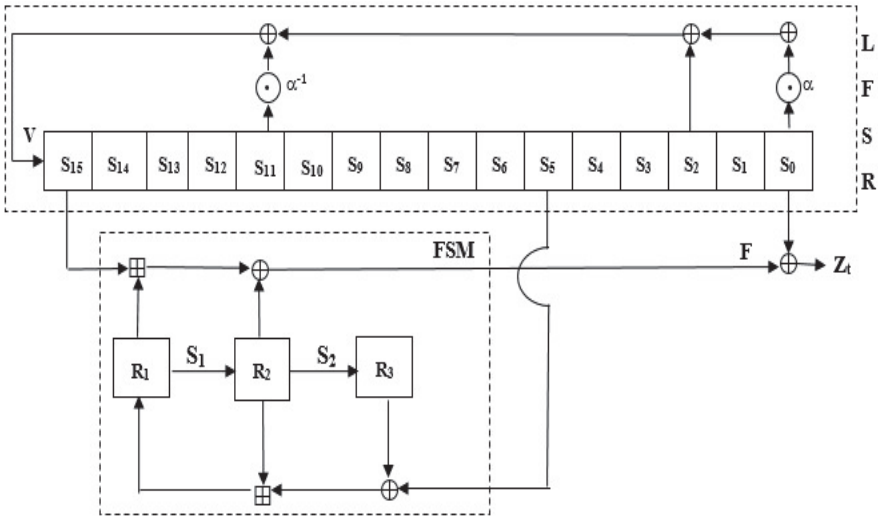


Figure 2 SNOW algorithm during the generation of keystream [3].

SNOW algorithm during the generation of the keystream process to generate randomness keystream cipher outputs 32 bits word (Z_t) at each clock cycle. The keystream word is calculated as $Z_t = F \oplus s_0$ (as shown in Figure 2). Calculate the intermediate value (V) as following [3]:

$$V = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL\alpha(s_{0,0}) \oplus s_2 \oplus (0x00 \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV\alpha(s_{11,3}) \tag{7}$$

In both processes the functions MULα and DIVα are used, there are maps 8 bits to 32 bits which are defined as following [3]:

$$\begin{aligned} MUL\alpha(c) = {} & (MULxPOW(c, 23, 0xA9) \| MULxPOW(c, 245, 0xA9) \\ & \| MULxPOW(c, 48, 0xA9) \\ & \| MULxPOW(c, 239, 0xA9)) \end{aligned} \tag{8}$$

$$\begin{aligned} DIV\alpha(c) = {} & (MULxPOW(c, 16, 0xA9) \| MULxPOW(c, 39, 0xA9) \\ & \| MULxPOW(c, 6, 0xA9) \\ & \| MULxPOW(c, 64, 0xA9)) \end{aligned} \tag{9}$$

The Rijndael S1-box used in the Finite State Machine (FSM) layer of the SNOW algorithm is set by the resulting Table 1, the input is mapped to its multiplicative inverse in GF (2^8) = GF(2)[x]/($x^8 + x^4 + x^3 + x + 1$),

the multiplicative inverse is then transformed using the following affine transformation ($S1 = Mx^{-1} + B$, where $B = 0x63$ and M is a matrix of size 8×8) [3, 10]:

$$\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{10}$$

The Rijndael S1-box maps 32 bits input to 32 bits output. Let $X = x_0 \| x_1 \| x_2 \| x_3$ and $S1(X) = y_0 \| y_1 \| y_2 \| y_3$, with y_0 the most and y_3 the least significant byte, then y_0, y_1, y_2, y_3 are defined as follows [3]:

$$\begin{aligned} y_0 &= \mathrm{MULx}(S1(x_0), 0x1B) \oplus S1(x_1) \oplus S1(x_2) \\ &\quad \oplus \mathrm{MULx}(S1(x_3), 0x1B) \oplus S1(x_3), \\ y_1 &= \mathrm{MULx}(S1(x_0), 0x1B) \oplus S1(x_0) \oplus \mathrm{MULx}(S1(x_1), 0x1B) \\ &\quad \oplus S1(x_2) \oplus S1(x_3), \\ y_2 &= S1(x_0) \oplus \mathrm{MULx}(S1(x_1), 0x1B) \oplus S1(x_1) \\ &\quad \oplus \mathrm{MULx}(S1(x_2), 0x1B) \oplus S1(x_3), \\ y_3 &= S1(x_0) \oplus S1(x_1) \oplus \mathrm{MULx}(S1(x_2), 0x1B) \\ &\quad \oplus S1(x_2) \oplus \mathrm{MULx}(S1(x_3), 0x1B). \end{aligned} \tag{11}$$

The Dickson S2-box used in the Finite State Machine (FSM) layer of the SNOW algorithm is formed using the Dickson polynomial $g_{49}(x) = x \oplus x^9 \oplus x^{13} \oplus x^{15} \oplus x^{33} \oplus x^{41} \oplus x^{45} \oplus x^{47} \oplus x^{49}$, for input x (8 bits) in GF(2^8) defined by the polynomial $x^8 + x^6 + x^5 + x^3 + 1$, the output (8 bits) of S2-box matches with $g_{49}(x) \oplus 0x25$ is set by the resulting Table 2 [3, 10]. The Dickson S2-box maps 32 bits input to 32 bits output, let $X = x_0 \| x_1 \| x_2 \| x_3$ and $S2(X) = y_0 \| y_1 \| y_2 \| y_3$, with y_0 the most and y_3 the least significant byte, then y_0, y_1, y_2, y_3 are defined as follows [3]:

$$\begin{aligned} y_0 &= \mathrm{MULx}(S2(x_0), 0x69) \oplus S2(x_1) \oplus S2(x_2) \\ &\quad \oplus \mathrm{MULx}(S2(x_3), 0x69) \oplus S2(x_3), \end{aligned}$$

Table 1 Rijndael S1-box used in FSM- SNOW algorithm [3]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	DO	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

$$
\begin{aligned}
\mathrm{y}_1 &= \mathrm{MULx}(\mathrm{S2}(\mathrm{x}_0), 0\mathrm{x}69) \oplus \mathrm{S2}(\mathrm{x}_0) \oplus \mathrm{MULx}(\mathrm{S2}(\mathrm{x}_1), 0\mathrm{x}69) \\
&\quad \oplus \mathrm{S2}(\mathrm{x}_2) \oplus \mathrm{S2}(\mathrm{x}_3), \\
\mathrm{y}_2 &= \mathrm{S2}(\mathrm{x}_0) \oplus \mathrm{MULx}(\mathrm{S2}(\mathrm{x}_1), 0\mathrm{x}69) \oplus \mathrm{S2}(\mathrm{x}_1) \\
&\quad \oplus \mathrm{MULx}(\mathrm{S2}(\mathrm{x}_2), 0\mathrm{x}69) \oplus \mathrm{S2}(\mathrm{x}_3), \\
\mathrm{y}_3 &= \mathrm{S2}(\mathrm{x}_0) \oplus \mathrm{S2}(\mathrm{x}_1) \oplus \mathrm{MULx}(\mathrm{S2}(\mathrm{x}_2), 0\mathrm{x}69) \\
&\quad \oplus \mathrm{S2}(\mathrm{x}_2) \oplus \mathrm{MULx}(\mathrm{S2}(\mathrm{x}_3), 0\mathrm{x}69). \qquad (12)
\end{aligned}
$$

2.2 Description of ZUC algorithm

ZUC is a modern synchronous stream cipher that contains two different processes, the initialization process and the generation of the keystream process, below the rule of 128 bits Ciphering Key/Integrity Key (CK/IK) and 128 bits Initialization Vector (IV) [4]. The internal structure of the ZUC algorithm is collected from the Linear Feedback Shift Register (LFSR), Bit-Reorganization (BR), and Nonlinear Function (NLF) layers. The LFSR layer

Table 2 Dickson S2-box used in FSM-SNOW algorithm [3]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	25	24	73	67	D7	AE	5C	30	A4	EE	6E	CB	7D	B5	82	DB
1	E4	8E	48	49	4F	5D	6A	78	70	88	E8	5F	5E	84	65	E2
2	D8	E9	CC	ED	40	2F	11	28	57	D2	AC	E3	4A	15	1B	99
3	B2	80	85	A6	2E	02	47	29	07	4B	0E	C1	51	AA	89	D4
4	CA	01	46	B3	EF	DD	44	7B	C2	7F	BE	C3	9F	20	4C	64
5	83	A2	68	42	13	B4	41	CD	BA	C6	BB	6D	4D	71	21	F4
6	8D	B0	E5	93	FE	8F	E6	CF	43	45	31	22	37	36	96	FA
7	BC	0F	08	52	1D	55	1A	C5	4E	23	69	7A	92	FF	5B	5A
8	EB	9A	1C	A9	D1	7E	0D	FC	50	8A	B6	62	F5	0A	F8	DC
9	03	3C	0C	39	F1	B8	F3	3D	F2	D5	97	66	81	32	A0	00
A	06	CE	F6	EA	B7	17	F7	8C	79	D6	A7	BF	8B	3F	1F	53
B	63	75	35	2C	60	FD	27	D3	94	A5	7C	A1	05	58	2D	BD
C	D9	C7	AF	6B	54	0B	E0	38	04	C8	9D	E7	14	B1	87	9C
D	DF	6F	F9	DA	2A	C4	59	16	74	91	AB	26	61	76	34	2B
E	AD	99	FB	72	EC	33	12	DE	98	3B	C0	9B	3E	18	10	3A
F	56	E1	77	C9	1E	9E	95	A3	90	19	A8	6C	09	D0	F0	86

is created by 16 stages in which each stage consists of 31 bits, the bit reorganization excerpts 128 bits from the stages of the LFSR and reshapes 4 of 32-bits word, in which the first three words will be used by the Nonlinear Function (NLF) and the final word will be used to produce the keystream cipher output (Z_t) [4].

The BR layer is shaped by four registers of 32 bits (X_0, X_1, X_2, X_3) filled by the 8 stage registers (s_{15}, s_{14}, s_{11}, s_9, s_7, s_5, s_2, s_0) of the LFSR layer. The BR is defined as following [4]:

$$X_0 = s_{15H} \| s_{14L}$$

$$X_1 = s_{11L} \| s_{9H}$$

$$X_2 = s_{7L} \| s_{5H}$$

$$X_3 = s_{2L} \| s_{0H}$$

where s_{nH} is the leftmost 16 bits of integer s_n and s_{nL} is the rightmost 16 bits of integer s_n.

The Non-Linear function (NLF) has two of 32 bits memory registers R_1 and R_2. The inputs to the nonlinear function are X_0, X_1, and X_2, which come from the outputs of the bit-reorganization, then the function (F) outputs 32 bits word (W), then the nonlinear function (F (X_0, X_1, X_2)) is described as follows [4]:

$$W = (X_0 \oplus R_1) \boxplus R_2 \tag{13}$$

$$W_1 = R_1 \boxplus X_1 \tag{14}$$

$$W_2 = R_2 \oplus X_2 \tag{15}$$

$$R_1 = S(L1(W_{1L} \| W_{2H})) \tag{16}$$

$$R_2 = S(L2(W_{2L} \| W_{1H})) \tag{17}$$

Two S-boxes are used in the Nonlinear Function (NLF) layer of the ZUC algorithm (Feistel structure S1-box and New Rijndael S2-box) [4, 16]. Both L1 and L2 are linear transforms from 32 bits to 32 bits words, the linear transform depended on the Exclusive-OR and the cyclic shift over GF(2)[x]/($x^{32} + 1$) is described as follows [4]:

$$L1(x) = x \oplus (x <<<_{32} 2) \oplus (x <<<_{32} 10) \oplus (x <<<_{32} 18) \oplus (x <<<_{32} 24) \tag{18}$$

$$L2(x) = x \oplus (x <<<_{32} 8) \oplus (x <<<_{32} 14) \oplus (x <<<_{32} 22) \oplus (x <<<_{32} 30) \tag{19}$$

The 32×32 S-box (S) is collected from four contrasted (8×8) S-boxes, S = (S1, S2, S3, S4), where S1-box = S3-box, S2-box = S4-box. The S1-box is designed using a Feistel structure as shown in Table 3 (nonlinearity = 96, differential uniformity = 8, algebraic degree = 5, and algebraic immunity = 2) and the S2-box is based on the inversion over the finite field GF (2^8) defined by the irreducible polynomial $x^8 + x^7 + x^3 + x + 1$ [16].

Since the S2-box is affine equivalent to that of the Rijndael S-box, then the New Rijndael S2-box as shown in Table 4, has many cryptographic properties same as the Rijndael S-box including (nonlinearity = 112, differential uniformity = 4, algebraic degree = 7, and algebraic immunity = 2) [16], the multiplicative inverse is then transformed using the following affine transformation ($S2 = Mx^{-1} + B$, where $B = 0x55$ and M is a matrix of

size 8×8) as following [16]:

$$\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{20}$$

Let $X = x_0 \| x_1 \| x_2 \| x_3$ be the input with x_0 the most and x_3 the least significant byte. Let $S(X) = r_0 \| r_1 \| r_2 \| r_3$ with r_0 the most and r_3 the least significant byte, then we have $r_i = Si(w_i)$, i = 0, 1, 2, 3. $S(X) = S1(x_0) \| S2(x_1) \| S3(x_2) \| S4(x_3)$, let x_n be an 8-bit input to Feistel S1-box (or New Rijndael S2-box) then write x_n into two hexadecimal digits as $x = r \| c$, then the entrance at the joint of the r-th row and the c-th column in Table 3 is the output of Feistel S1-box (or in Table 4 is the output of New Rijndael S2-box) [4].

ZUC algorithm during the initialization process, the LFSR obtains 31 bits input word (u), which is obtained by eliminating the rightmost bit from the 32 bits output (W) of the nonlinear function, (u = W>>1), this process is performed for 32 clock runs without generating the output sequence of keystream (as shown in Figure 3) [4]. Let $K = K_0 \| K_1 \| \ldots \| K_{14} \| K_{15}$, $IV = IV_0 \| IV_1 \| \ldots \| IV_{14} \| IV_{15}$, where K_i and $IV_i, 0 \leq i \leq 15$, are all bytes, and D be a 240-bit long constant string combined by 16 substrings of 15 bits, $D = D_0 \| D_1 \| \ldots \| D_{14} \| D_{15}$, then the LFSR registers are initialized as follows [4]:

For $0 \leq i \leq 15$, let $s_i = K_i \| D_i \| IV_i$

$$V = 2^{15} s_{15} + 2^{17} s_{13} + 2^{21} s_{10} + 2^{20} s_4 + (1 + 2^8) s_0 \bmod (2^{31} - 1) \tag{21}$$

$$s_{16} = (v + u) \bmod (2^{31} - 1) \tag{22}$$

If $s_{16} = 0$, then set $s_{16} = 2^{31} - 1$, and $(s_1, s_2, \ldots, s_{15}, s_{16}) \rightarrow (s_0, s_1, \ldots, s_{14}, s_{15})$.

ZUC algorithm during the generation of keystream process to generate randomness keystream cipher outputs 32 bits word ($Z_t = F(X_0, X_1, X_2) \oplus X_3$) at each clock cycle using the output of the NLF layer, Exclusive OR, and the register X_3 of the BR layer (as shown in Figure 4).

Table 3 Feistel structure S1-box used in NLF-ZUC algorithm [4]

	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	AG	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	5V	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	4F	45	7A	19	DF	EE	78	34	60

3 The NIST SP 800-22 Statistical Test Suite

The NIST SP 800-22 test package is established on the mode of statistical hypothesis analysis, if the P-value for an investigation is determined to be equal to one, then the sequence performs to have faultless randomness and the P-value of zero specifies that the sequence performs to be non-random [18, 19]. The significance level (α) be able to select for the tests ($\alpha = 0.01$), if the P-value $\geq \alpha$, subsequently the null hypothesis (H_0) is accepted (sequence performs to exist random) and if the P-value $< \alpha$, subsequently the null hypothesis (H_0) is rejected (sequence performs to exist non-random) [18].

NIST SP 800-22 is a statistical suite that involves 15 tests [18]. There are described as:

- Frequency Test: balanced property [18].
- Block Frequency Test: balanced property for M-sized blocks (subsequences) of the keystream. Given sequence length n, the block size M

Table 4 New Rijndael S2-box used in NLF-ZUC algorithm [4]

	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	D8	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
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

must be that, $M \geq 0.01n$, $n \geq MN$, and non-overlapping blocks $N < 100$, $N = n/M$. For $n = 1048576$, we selected $M = 16384$ [18].

- Runs Test: decides that the number of runs ones and zeros of several lengths is as expected for a random sequence. In precise, this test decides whether the oscillation between such zeros and ones is too fast or too slow, the Runs test carries out a Frequency test as a precondition [18].
- Longest Run Test: decides that the length of the longest run of ones within the tested sequence is reliable with the length of the longest run of ones that would be expected in a random sequence [18].
- Rank Test: checks for linear dependence among fixed-length substrings of the original sequence [18].
- Discrete Fourier Test: detects periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the hypothesis of randomness. The aim is to detect whether the number of peaks above the 95% threshold is significantly different than 5% [18].

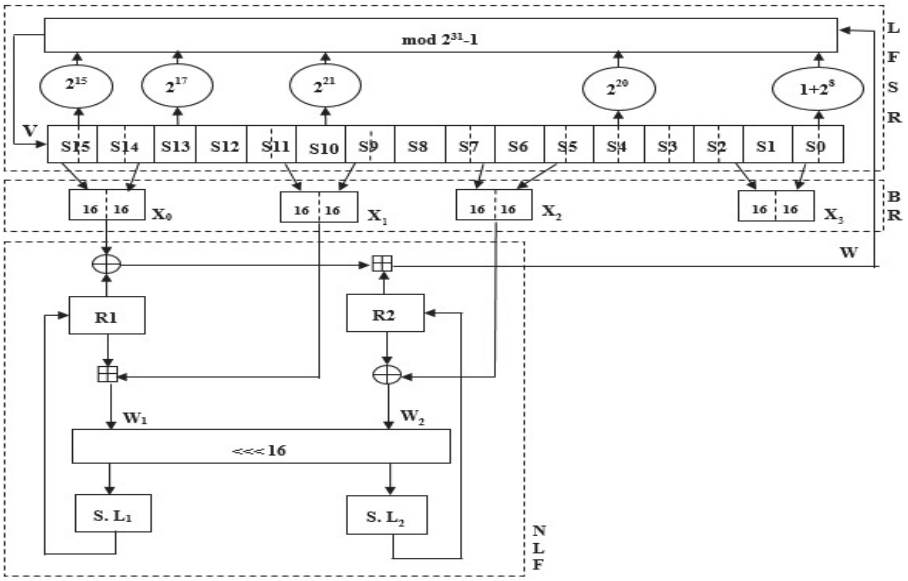


Figure 3 ZUC algorithm during the initialization process [4].

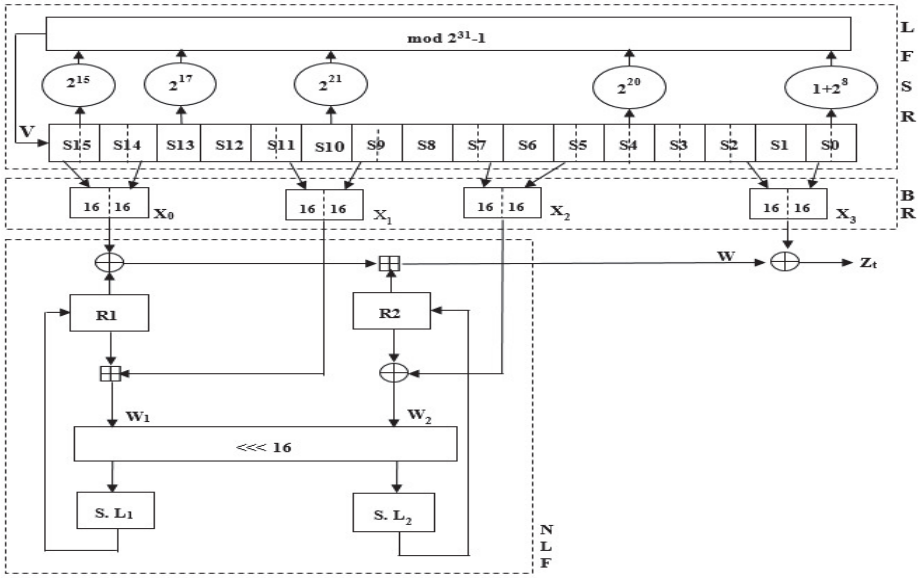


Figure 4 ZUC algorithm during the generation of keystream [4].

- Non-Overlapping Template Matching Test: detects how many times a pre-defined template occurs in the sequence. For this test, an m-bit window is used to search for an exact m-bit pattern. If the pattern is not originating, the window slides a one-bit position. If the pattern is originating, the window is reset to the bit after the originate pattern, and the search continues. Non-overlapping templates involve of 148 subtests (P-values), for each of the 148 templates identical to the default template size, $m = 9$ [18].
- Overlapping Template Matching Test: this test and the Non-Overlapping Template Matching test, use an m-bit window to search for an exact m bit pattern. If the pattern is not originating, the window slides a one-bit position. The difference between them is that when the pattern is originating, the window slides only one bit before continuing the search [18].
- Universal Test: detects whether or not the sequence can be significantly compressed without loss of information [18].
- Linear Complexity Test: decides whether or not the sequence is complex enough to be considered random. Random sequences are characterized by longer LFSRs, using the Berlekamp Massey algorithm, we selected the length of the bit string $n \geq 10^6$, the value of the length in bits of a block (M) must be in the range $500 \leq M \leq 5000$, and (N) independent blocks of (M) bits, $N \geq 200$ [18].
- Serial Test: decides whether the number of existences of the 2^m, (m) bit overlapping patterns is nearly the same as would be expected for a random sequence. Random sequences have uniformity, that is, every m bit pattern (the length in bits of each block) has the same chance of performing as every other m-bit pattern. This test extends the sequence by adding the first (m-1) bits to the end of the sequence for different values of (n). Resolve the frequency of all possible overlapping (m) bit blocks, all probable overlapping (m-1) bit blocks, and all probable overlapping (m-2) bit blocks. This test involves 2 subtests (P_1 and P_2 values). we necessity to select block length (m) such that we have $m < \lfloor \log_2 n \rfloor - 2$, we selected $m = 16$ [18].
- Approximate Entropy Test: compares the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) beside the expected outcome for a random sequence, we need block length (m) such that $m < \lfloor \log_2 n \rfloor - 5$, we selected $m = 10$ [18].

- Cumulative Sums Test: decides whether the cumulative sum of the restricted sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. This test involves 2 subtests, a change for relating the test either Forward P-value or Reverse P-value [18].
- Random Excursions Test: decides if the number of visits to a certain state inside a cycle swerves from what one would expect for a random sequence. This test involves of 8 subtests (P-values), identical to the (x) values $-4 \leq x \leq -1$ and $1 \leq x \leq 4$. Let J = the total number of zero crossings in random walk S′ and if $J < 500$, then discontinue the test. For each of the 8 states of x, compute $\nu_k(x)$ = the total number of cycles in which state x occurs exactly k times among all cycles, for k = 0, 1, ..., 5 (k = 5, all frequencies ≥ 5 are stored in $\nu_5(x)$) [18].
- Random Excursions Variant Test: detects aberrations from the expected number of visits to several states in the random walk. This test involves 18 subtests (P-values), identical to the (x) values –9 $\leq x \leq -1$ and $1 \leq x \leq 9$. For each of the eighteen non-zero states of x, compute $\xi(x)$ = the total number of times that state x occurred across all J cycles [18].

4 Configurations of S-Boxes

We have four 3GPP-standardized S-boxes applied in SNOW and ZUC algorithms (4G, 5G), these are described as follows:

- Rijndael (R) (S1-box), and Dickson (D) (S2-box) are used in the SNOW algorithm [3].
- Feistel structure (F) (S1-box), and New Rijndael (NR) (S2-box) are used in the ZUC algorithm [4].

We added the most complex S-box (Improved New Rijndael (INR) S-box) which had been tested by Jie Cui, Hong Zhong, Jiankai Wang, and Runhua Shi using cryptographic properties of S-box that has the strongest resistance against algebraic attacks ($\Gamma = 2^{160}$) [20]. The S-box (Improved New Rijndael) as shown in Table 5 [21], the input is mapped to its multiplicative inverse in GF (2^8) = GF(2)[x]/$(x^8 + x^4 + x^3 + x + 1)$, a new affine transformation ($L_{5B,5D}$) to substitute the original one. There are described as [21]:

- Set the affine transformation $\mathrm{L_{5B,5D}}$: $\mathrm{x'} = \mathrm{L_{5B,5D}(x)} = \mathrm{Lb} \times \mathrm{x} + \mathrm{5D}$

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathrm{x_7} \\ \mathrm{x_6} \\ \mathrm{x_5} \\ \mathrm{x_4} \\ \mathrm{x_3} \\ \mathrm{x_2} \\ \mathrm{x_1} \\ \mathrm{x_0} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \tag{23}$$

- Proceeding the multiplicative inverse:

$$x'' = (x')^{-1} \tag{24}$$

- Set the affine transformation again:

$$y = L_{5B,5D}(x'') = Lb \times x'' + 5D \tag{25}$$

To assess the randomness properties of SNOW and ZUC algorithms, we selected two S-boxes out of the five ones mentioned above (Rijndael (R), Dickson (D), Feistel structure (F), New Rijndael (NR), and Improved New Rijndael (INR) S-boxes). So, we have twenty different configurations of pair S-boxes for both SNOW and ZUC algorithms.

5 SNOW and ZUC Simulation with Different S-boxes Configurations and NIST SP 800-22 Randomness 15-tests Verifications

This section discusses the SNOW and ZUC simulation with different S-boxes configurations and NIST tests verification.

5.1 Simulation of SNOW and ZUC Algorithms with Different Twenty Pair S-boxes Configurations

We implemented the SNOW and ZUC algorithms by using C language to evaluate the performance and test the best pair S-boxes configurations of SNOW and ZUC algorithms output keystream sequences ($\mathrm{Z_t}$). NIST SP 800-22 statistical test suite will be applied on SNOW and ZUC algorithms output sequences of length ten Megabits, then with every clock pulse, it produces a 32-bit word of output $\mathrm{Z_t} = [\mathrm{Z_1} \| \mathrm{Z_2} \| \ldots \| \mathrm{Z_{312500}}]$.

Table 5 Improved New Rijndael S-box [21]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	FA	62	a0	EA	81	C9	2F	22	E5	A9	BD	1E	13	4D	65	C8
1	87	17	BB	88	B8	45	57	95	F3	0B	9E	D7	68	11	8A	B2
2	3B	A8	1D	A5	F8	5D	3E	8F	D2	0E	80	06	54	4B	3D	6E
3	F0	28	02	6D	E9	63	32	23	82	1C	C3	B3	15	B4	0G	C7
4	12	39	19	58	7C	99	A1	26	89	B7	77	C2	D5	66	73	DD
5	BF	40	72	0D	4A	97	5C	2B	FD	BE	6B	D1	44	9A	69	0A
6	75	6F	70	16	EB	FB	BA	33	36	3F	78	21	74	2E	B1	8E
7	5B	7B	7A	AD	4E	7E	AF	A4	F6	10	B5	C1	48	F1	3C	A6
8	09	E7	CE	8B	24	20	DE	D4	9F	AE	79	07	61	A2	DB	5E
9	D8	4C	EE	ED	7D	C6	71	FE	29	FF	31	C5	59	FC	DA	98
A	2A	6A	E6	42	B0	CD	04	91	F9	14	47	27	83	34	1F	EC
B	2D	18	5A	76	60	E4	50	25	3A	56	03	D9	85	6C	90	E8
C	41	94	92	30	05	38	84	D6	CA	51	AC	43	8C	D3	A7	C0
D	0C	1A	67	AB	D0	F4	1B	BC	8D	F7	5F	AA	08	46	35	B6
E	00	E0	9B	CF	EF	86	9D	4F	E3	DF	E1	93	B9	E2	53	64
F	7F	CB	CC	01	9C	2C	A3	F5	52	55	49	96	DC	C4	F2	37

5.1.1 Simulation of SNOW Algorithm with Different Twenty Configurations of S-boxes

SNOW is a word-oriented stream cipher that generates 32 bits word keystream $Z_t = [Z_1 \| Z_2 \| \ldots \| Z_{312499} \| Z_{312500}]$ as shown in Table 6, under control of 128 bits Key (CK/IK) = [2BD6459F82C5B300952C4910488 1FF48] and 128 bits IV = [EA024714AD5C4D84DF1F9B251C0BF45F] as standardized in test set one by 3GPP [22].

5.1.2 Simulation of ZUC Algorithm with Different Twenty Configurations of S-boxes

ZUC is a word oriented stream cipher that generates 32 bits word keystream $Z_t = [Z_1 \| Z_2 \| \ldots \| Z_{312499} \| Z_{312500}]$ as shown in Table 7, under control of 128 bits Key (CK/IK) = [3D4C4BE96A82FDAEB58F641DB17B455B], D = [44D726BC626B135E578935E2713509AF4D782F1 36BC41AF15E26 3C4D789A47AC] and 128 bits IV = [84319AA8DE6915CA1F6BDA6bFBD 8C766] as standardized in test set three by 3GPP [23].

Table 6 Results implementation of the SNOW algorithm with different twenty configurations of (S1-box) and (S2-box)

Sample (1): R (S1-box|) and D (S2-box) , as in standardized 3GPP SNOW [22]
$Z_t = [ABEE9704 \| 7AC31373 \| \ldots \| 04244B83 \| 9CFEA3D6]$
Sample (2): D (S1-box|) and R (S2-box)
$Z_t = [C111D662 \| 8B110B9B \| \ldots \| E77EE597 \| 126E98C2]$
Sample (3): [R (S1-box|) and INR (S2-box)]
$Z_t = [A8061F97 \| ECB76A82 \| \ldots\ldots \| 1F799FE0 \| 466E650D]$
Sample (4): INR (S1-box|) and R (S2-box)
$Z_t = [C95308B9 \| 516283AB \| \ldots\ldots \| 3A3CD518 \| 696F68B1]$
Sample (5):D (S1-box|) and INR (S2-box)
$Z_t = [BA725F13 \| 53DFCEBC \| \ldots \| 41005D82 \| 48D0E266]$
Sample (6): INR (S1-box|) and D (S2-box)
$Z_t = [208C06E8 \| 3BCA8820 \| \ldots\ldots \| 7FF86529 \| E7628318]$
Sample (7): NR (S1-box|) and F (S2-box)
$Z_t = [1D1B215E \| C0773ED1 \| \ldots \| E4A519C2 \| 8020E7F0]$
Sample (8): F (S1-box|) and NR (S2-box)
$Z_t = [0F33FB21 \| BE4E49C0 \| \ldots\ldots \| 80D19C48 \| AE8A3BBF]$
Sample (9): F (S1-box|) and INR (S2-box)
$Z_t = [94039631 \| F3415052 \| \ldots\ldots \| 471D9A7A \| 893F1455]$
Sample (10): INR (S1-box|) and F (S2-box)
$Z_t = [B4463505 \| C0380DC8 \| \ldots\ldots \| 2ECA8FEF \| 41C9F241]$
Sample (11): NR (S1-box|) and INR (S2-box)
$Z_t = [6CC1CCCB \| 0937AD76 \| \ldots\ldots \| 44784AC5 \| 79BCD4C3]$
Sample (12): INR (S1-box|) and NR (S2-box)
$Z_t = [ADD456D5 \| 86FA2492 \| \ldots\ldots \| 5569FE42 \| E4D87277]$
Sample (13): R (S1-box|) and NR (S2-box)
$Z_t = [2E535E84 \| D591FF55 \| \ldots \| 46255592 \| 0B737D22]$
Sample (14): NR (S1-box|) and R (S2-box)
$Z_t = [1635CBFD \| 104EA1AF \| \ldots\ldots \| CE07C416 \| 4D74CCFE]$
Sample (15): R (S1-box|) and F (S2-box)
$Z_t = [3B17BA47 \| 07B9112F \| \ldots\ldots \| 0E55D7A8 \| 95923E4F]$
Sample (16): F (S1-box|) and R (S2-box)
$Z_t = [4D6DCBB7 \| 3461E626 \| \ldots\ldots \| 819886D8 \| D3388432]$
Sample (17): D (S1-box|) and NR (S2-box)
$Z_t = [68956D31 \| B94EF1EB \| \ldots\ldots \| 3265985A \| 42CF8F6B]$
Sample (18): NR (S1-box|) and D (S2-box)
$Z_t = [4AC53315 \| 0351F2B3 \| \ldots\ldots \| AED5D835 \| 830F8CF1]$
Sample (19): D (S1-box|) and F (S2-box)
$Z_t = [7022CC9E \| 93F93DB6 \| \ldots\ldots \| 6071D858 \| 3D258954]$
Sample (20): F (S1-box|) and D (S2-box)
$Z_t = [C3B0F1FE \| B57C8162 \| \ldots\ldots \| 147425DC \| 280FD770]$

Table 7 Results implementation of the ZUC algorithm with different twenty configurations of (S1-box) and (S2-box)

Sample (1): R (S1-box|) and D (S2-box)
$Z_t = [9E31F6BD\|\|88EAD710\|\| \ldots \|\|5AFF0720\|\|EEF967AC]$
Sample (2): D (S1-box|) and R (S2-box)
$Z_t = [77D0A9B5\|\|D6685E74\|\| \ldots \|\|A548D129\|\|13D5C471]$
Sample (3): R (S1-box|) and INR (S2-box)
$Z_t = [722AE874\|\|EAA0594C\|\| \ldots \|\|72E696E6\|\|D36E5DAA]$
Sample (4): INR (S1-box|) and R (S2-box)
$Z_t = [79BB2DEE\|\|79CD0EFE\|\| \ldots \|\|C5DD7A80\|\|4D1A3AE5]$
Sample (5): D (S1-box|) and INR (S2-box)
$Z_t = [0FE7D8B7\|\|920519FC\|\| \ldots \|\|03B9814C\|\|A1EE26E9]$
Sample (6): INR (S1-box|) and D (S2-box)
$Z_t = [5D3E35A4\|\|A70B0DB1\|\| \ldots \|\|C6B51E89\|\|6A5E84FC]$
Sample (7): NR (S1-box|) and F (S2-box)
$Z_t = [A062A431\|\|5ED61AB4\|\| \ldots \|\|253465A5\|\|96B8D82D]$
Sample (8): F (S1-box|) and NR (S2-box), as in standardized 3GPP ZUC [23]
$Z_t = [14F1C272\|\|3279C419\|\| \ldots \|\|A99B9FA9\|\|1AF5209F]$
Sample (9): F (S1-box|) and INR (S2-box)
$Z_t = [6973FC71\|\|3529160A\|\| \ldots \|\|2C448FAD\|\|D1562A98]$
Sample (10): INR (S1-box|) and F (S2-box)
$Z_t = [D629ABE7\|\|26F61100\|\| \ldots \|\|7D9DEC7C\|\|9C44FAC7]$
Sample (11): NR (S1-box|) and INR (S2-box)
$Z_t = [A508A0BA\|\|E98A7CC9\|\| \ldots \|\|B1B27EC4\|\|B028CA0E]$
Sample (12): INR (S1-box|) and NR (S2-box)
$Z_t = [CEA4FEF0\|\|88B8B386\|\| \ldots \|\|CAA8D0C8\|\|7BE51175]$
Sample (13): R (S1-box|) and NR (S2-box)
$Z_t = [87DD182D\|\|3D75A9FB\|\| \ldots \|\|369CFE4C\|\|6C5C3EA3]$
Sample (14): N R (S1-box|) and R (S2-box)
$Z_t = [D3B53312\|\|289AE6E4\|\| \ldots \|\|5555624D\|\|5866A1B6]$
Sample (15): R (S1-box|) and F (S2-box)
$Z_t = [1CE3EE0F\|\|BA1D6B42\|\| \ldots \|\|D9FA9A02\|\|28F65185]$
Sample (16): F (S1-box|) and R (S2-box)
$Z_t = [3CAD3F95\|\|B846C393\|\| \ldots \|\|B97ED94F\|\|1FB34064]$
Sample (17): D (S1-box|) and NR (S2-box)
$Z_t = [F2AC4FAC\|\|0183ACA7\|\| \ldots \|\|A91DF17B\|\|093A1F22]$
Sample (18): NR (S1-box|) and D (S2-box)
$Z_t = [5b6e9220\|\|4f7b6903\|\| \ldots \|\|378ebd89\|\|6cb640d4]$
Sample (19): D (S1-box|) and F (S2-box)
$Z_t = [928EC747\|\|538E47DF\|\| \ldots \|\|500B38BB\|\|4A7660D3]$
Sample (20): F (S1-box|) and D (S2-box)
$Z_t = [392A7ECA\|\|092B37FD\|\| \ldots \|\|20230FEE\|\|333EF5FE]$

5.2 NIST SP 800-22 Statistical Analysis, Comparison, and Evaluation of SNOW and ZUC Stream Cipher Outputs

NIST analysis, comparison, and evaluation of SNOW and ZUC stream cipher outputs with the different twenty configurations of S-boxes are applied using 15-tests of the NIST SP 800-22 statistical test suite. The total length of stream cipher output is $10(2^{20})$ bits. These $10(2^{20})$ bits are divided into ten iterations each iteration consists of (2^{20}) bits is tested to give P-value. An average P-value is the result of these ten iteration tests.

5.2.1 NIST Statistical Analysis, Comparison, and Evaluation of SNOW Stream Cipher Output

Through these NIST (15-tests) statistical analyses, comparison, and evaluation of SNOW algorithm stream cipher output as shown in Tables 8 and 9, we noticed the following:

- The standard permutation (according to 3GPP-SNOW standardized S-boxes) (Sample (1) Rijndael (R) S1-box and Dickson (D) S2-box): failed in Serial test ($P2 = 0.004301 < 0.01$), weak in a Cumulative Sums test (forward test, $P = 0.035174$), weak in Longest Runs test ($P = 0.066882$), and weak in Non-Overlapping Template test (total number of failed P-value = 15 from 148 subtests).
- Permutation (Sample (8) Feistel structure (F) S1-box and New Rijndael (NR) S2-box) (according to 3GPP-ZUC standardized S-boxes): failed in Random Excursions test (number of discontinuities the test = 8 from 10 iterations of P-values ($J<500$), failed in Random Excursions Variant test (number of discontinuities the test = 8, and number of failed P-value = 3 from 18 subtests), weak in Non-Overlapping Template test (total number of failed P-value = 18 from 148 subtests), and weak in Discrete Fourier Transform test ($P = 0.066882$).
- Best permutation (Sample (16) Feistel structure (F) S1-box and Rijndael (R) S2-box): passes all the NIST SP 800-22 randomness 15-tests successfully as shown in Figures 5(a–o). Table 10 shows the comparison between the existing proposes of SNOW that verified by NIST SP 800-22 tests.

Table 8 NIST SP800-22 statistical tests (Approximate Entropy, Block Frequency, Cumulative Sums, DFT, Frequency, Linear Complexity, Longest Run, and Overlapping Template tests) verifications (P-values) for the different twenty samples of configuration S-boxes used in the SNOW algorithm

NIST Test P-value	Approximate Entropy Test-1	Block Frequency Test-2	Cumulative Sums (Forward/Reverse) Test-3	DFT Test-4	Frequency Test-5	Linear Complexity Test-6	Longest Run Test-7	Overlapping Template Test-8
Sample number (S1-box and S2-box) applied in the SNOW algorithm								
Sample 1 (R-D)) as in 3GPP-SNOW	0.534146	0.350485	0.035174 0.534146	0.350485	0.911413	0.991468	0.066882	0.739918
Sample 2 (D-R)	0.004301	0.739918	0.122325 0.534146	0.350485	0.739918	0.122325	0.534146	0.122325
Sample 3 (R-INR)	0.066882	0.911413	0.534146 0.122325	0.739918	0.122325	0.739918	0.739918	0.534146
Sample 4 (INR-R)	0.534146	0.534146	0.739918 0.534146	0.739918	0.911413	0.739918	0.066882	0.122325
Sample 5 (D-INR)	0.739918	0.739918	0.534146 0.350485	0.122325	0.350485	0.350485	0.350485	0.739918
Sample 6 (INR-D)	0.534146	0.350485	0.122325 0.739918	0.122325	0.534146	0.066882	0.739918	0.066882
Sample 7 (NR-F)	0.739918	0.739918	0.122325 0.739918	0.739918	0.739918	0.350485	0.122325	0.534146
Sample 8 (F-NR)	0.911413	0.739918	0.350485 0.534146	0.066882	0.534146	0.739918	0.534146	0.739918

(*Continued*)

Table 8 Continued

NIST Test P-value	Approximate Entropy Test-1	Block Frequency Test-2	Cumulative Sums (Forward/Reverse) Test-3	DFT Test-4	Frequency Test-5	Linear Complexity Test-6	Longest Run Test-7	Overlapping Template Test-8
Sample 9 (F-INR)	0.534146	0.213309	0.534146 0.213309	0.350485	0.534146	0.534146	0.350485	0.122325
Sample 10 (INR-F)	0.534146	0.739918	0.534146 0.534146	0.911413	0.534146	0.739918	0.350485	0.350485
Sample 11 (NR-INR)	0.534146	0.122325	0.122325 0.534146	0.911413	0.534146	0.122325	0.911413	0.739918
Sample 12 (INR-NR)	0.534146	0.534146	0.350485 0.739918	0.213309	0.350485	0.122325	0.739918	0.911413
Sample 13 (R-NR)	0.739918	0.739918	0.739918 0.739918	0.534146	0.739918	0.213309	0.122325	0.350485
Sample 14 (NR-R)	0.350485	0.739918	0.122325 0.350485	0.739918	0.911413	0.534146	0.534146	0.739918
Sample 15 (R-F)	0.534146	0.534146	0.534146 0.534146	0.350485	0.739918	0.534146	0.534146	0.350485
Sample 16 (F-R)	0.213309	0.739918	0.739918 0.739918	0.911413	0.739918	0.122325	0.911413	0.911413
Sample 17 (D-NR)	0.534146	0.534146	0.911413 0.739918	0.350485	0.350485	0.739918	0.122325	0.350485
Sample 18 (NR-D)	0.739918	0.350485	0.534146 0.534146	0.534146	0.739918	0.350485	0.066882	0.213309
Sample 19 (D-F)	0.213309	0.213309	0.911413 0.739918	0.911413	0.739918	0.534146	0.350485	0.534146
Sample 20 (F-D)	0.911413	0.739918	0.534146 0.739918	0.739918	0.350485	0.739918	0.350485	0.534146

Table 9 NIST SP800-22 statistical tests (Non-Overlapping Template, Random Excursions, Random Excursions Variant, Rank, Runs, Serial, and Universal tests) verifications (P-values) for the different twenty samples of configuration S-boxes used in the SNOW algorithm

NIST Test P-value	Non Overlapping Test-9	Random Excursions Test-10		Random Excursions Variant Test-11		Rank Test-12	Runs Test-13	Serial (P1&P2) Test-14	Universal Test-15
Sample number (S1-box and S2-box) applied in the SNOW algorithm	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value				
Sample 1 (R-D) as in 3GPP-SNOW	15	2	0	2	0	0.534146	0.739918	0.122325 0.004301	0.911413
Sample 2 (D-R)	14	3	1	3	6	0.739918	0.122325	0.911413 0.350485	0.122325
Sample 3 (R-INR)	22	3	0	3	5	0.350485	0.350485	0.739918 0.739918	0.213309
Sample 4 (INR-R)	12	7	0	7	0	0.534146	0.350485	0.017912 0.122325	0.534146
Sample 5 (D-INR)	17	5	0	5	0	0.739918	0.008879	0.122325 0.534146	0.911413
Sample 6 (INR-D)	14	10	0	10	0	0.739918	0.122325	0.534146 0.350485	0.911413
Sample 7 (NR-F)	16	2	1	2	0	0.739918	0.122325	0.122325 0.739918	0.911413

(*Continued*)

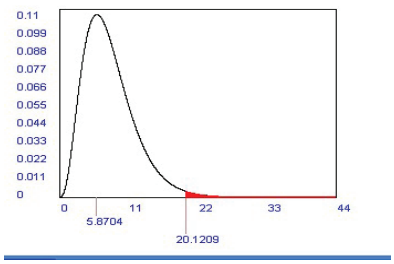
Table 9 Continued

NIST Test P-value	Non Overlapping Test-9	Random Excursions Test-10		Random Excursions Variant Test-11		Rank Test-12	Runs Test-13	Serial (P1&P2) Test-14	Universal Test-15
Sample number (S1-box and S2-box) applied in the SNOW algorithm	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value				
Sample 8 (F-NR)	18	8	0	8	3	0.911413	0.739918	0.534146 0.911413	0.122325
Sample 9 (F-INR)	17	4	0	4	0	0.534146	0.350485	0.911413 0.911413	0.122325
Sample 10 (INR-F)	16	6	0	6	1	0.350485	0.911413	0.534146 0.350485	0.350485
Sample 11 (NR-INR)	13	4	0	4	0	0.350485	0.534146	0.350485 0.350485	0.122325
Sample 12 (INR-NR)	22	4	1	4	2	0.350485	0.739918	0.213309 0.035174	0.008879
Sample 13 (R-NR)	14	4	1	4	0	0.534146	0.739918	0.534146 0.911413	0.350485
Sample 14 (NR-R)	17	3	2	3	0	0.739918	0.122325	0.739918 0.213309	0.911413
Sample 15 (R-F)	19	5	0	5	0	0.534146	0.911413	0.911413 0.739918	0.534146

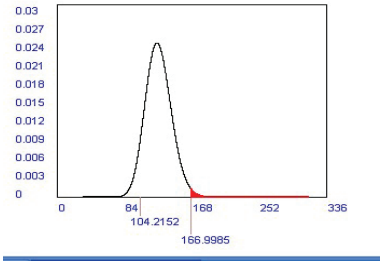
(*Continued*)

Table 9 Continued

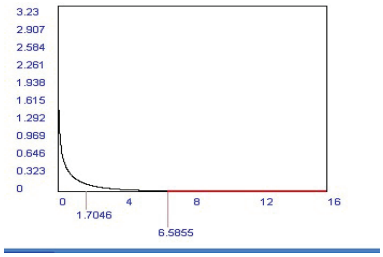
NIST Test P-value	Non Overlapping Test-9	Random Excursions Test-10		Random Excursions Variant Test-11		Rank Test-12	Runs Test-13	Serial (P1&P2) Test-14	Universal Test-15
Sample number (S1-box and S2-box) applied in the SNOW algorithm	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value				
Sample 16 (F-R)	7	2	0	2	0	0.739918	0.350485	0.534146 0.534146	0.122325
Sample 17 (D-NR)	19	5	0	5	0	0.739918	0.534146	0.534146 0.911413	0.213309
Sample 18 (NR-D)	18	3	0	3	0	0.911413	0.739918	0.534146 0.739918	0.739918
Sample 19 (D-F)	30	4	0	4	0	0.213309	0.350485	0.350485 0.534146	0.739918
Sample 20 (F-D)	9	2	1	2	0	0.739918	0.122325	0.739918 0.534146	0.350485



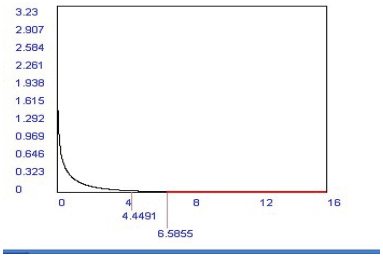
(a) Approximate Entropy Test



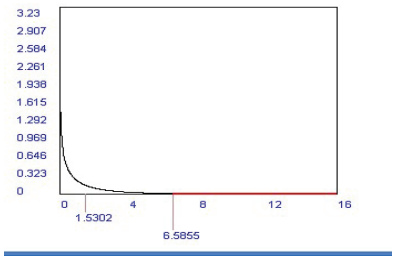
(b) Block Frequency Test



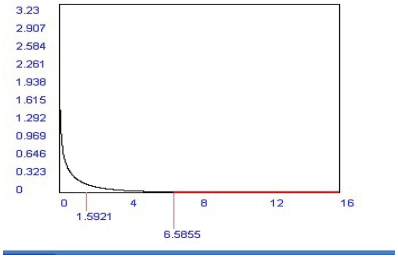
(c) Cumulative Sums Test



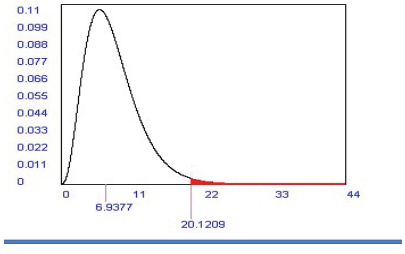
(d) DFT Test



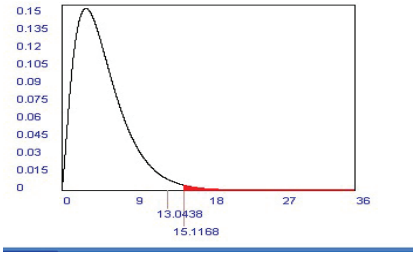
(e) Frequency Test



(f) Linear Complexity Test



(g) Longest Run Test



(h) Overlapping Template Test

Figure 5 Continued

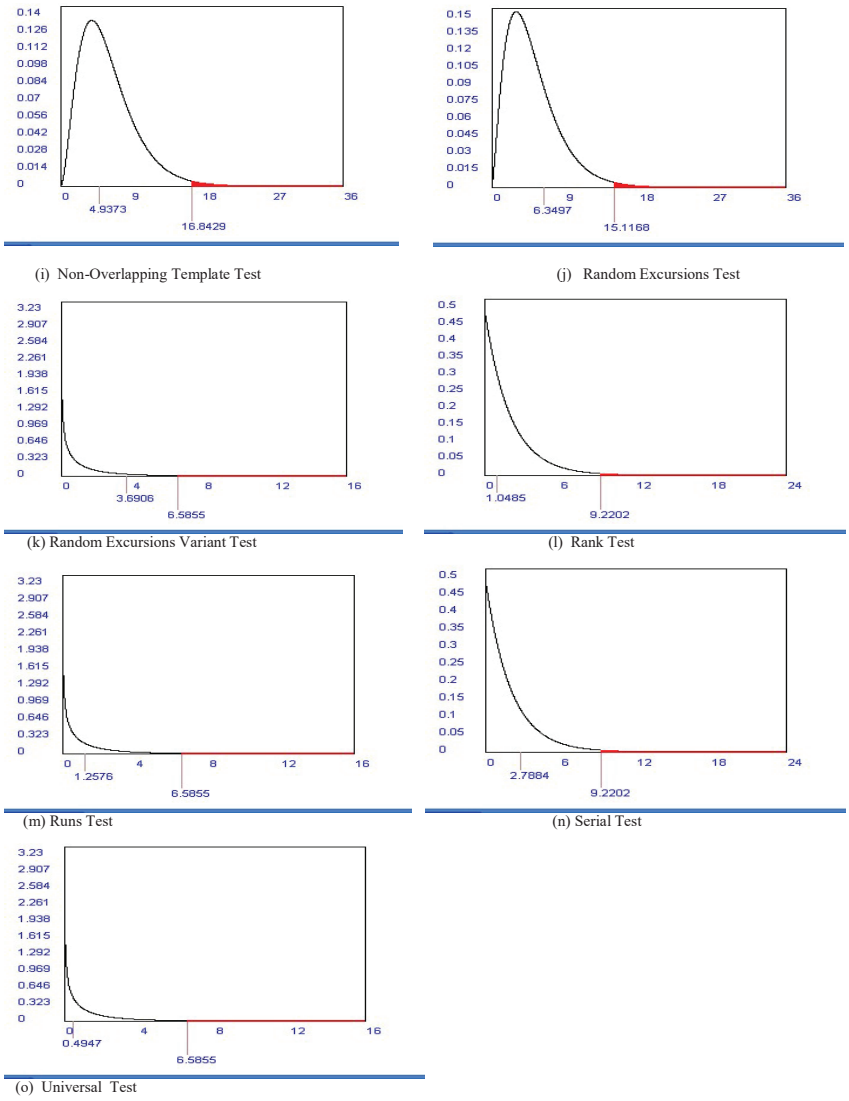


Figure 5 (a–o) NIST SP 800-22 results showed that the upgrading SNOW algorithm by using the best permutation (Sample (16) Feistel structure S1-box and Rijndael S2-box passed all NIST randomness tests successfully).

Table 10 Comparison between the existing proposes of SNOW that verified by NIST SP 800-22 statistical test suite

NIST SP 800-22 Tests	3GPP Standardized SNOW (Rijndael – Dickson S-boxes)	Proposal The Best Arrangement Pair of S-boxes in SNOW (Feistel – Rijndael S-boxes)
Approximate Entropy	Success	Success
Block Frequency	Success	Success
Cumulative Sums	Success	Success
DFT	Success	Success
Frequency	Success	Success
Linear Complexity	Success	Success
Longest Runs	Success	Success
Non-Overlapping Template	Success	Success
Overlapping Template	Success	Success
Random Excursions	Success	Success
Random Excursions Variant	Success	Success
Rank	Success	Success
Runs	Success	Success
Serial	Fail	Success
Universal	Success	Success

5.2.2 NIST Statistical Analysis, Comparison, and Evaluation of ZUC Stream Cipher Output

Through these NIST statistical analyses, comparison, and evaluation of ZUC algorithm stream cipher output as shown in Tables 11 and 12, we noticed the following:

- Permutation (Sample (1) Rijndael (R) S1-box and Dickson (D) S2-box) according to (3GPP-SNOW standardized S-boxes): weak in Block Frequency test (P = 0.066882), weak in a Cumulative Sums test (Reverse test, P = 0.035174), weak in Non-Overlapping Template test (total number of failed P-value = 19 from 148 subtests), and weak in Overlapping Template test (P = 0.035174).
- The standard Permutation (according to 3GPP-ZUC standardized S-boxes) (Sample (8) Feistel structure (F) S1-box and New Rijndael (NR) S2-box): failed in Random Excursions test (number of discontinuities

Table 11 NIST SP800-22 statistical tests (Approximate Entropy, Block Frequency, Cumulative Sums, DFT, Frequency, Linear Complexity, Longest Run, and Overlapping Template tests) verifications (P-values) for the different twenty samples of configuration S-boxes used in the ZUC algorithm

NIST Test P-value	Approximate Entropy Test-1	Block Frequency Test-2	Cumulative Sums (Forward/Reverse) Test-3	DFT Test-4	Frequency Test-5	Linear Complexity Test-6	Longest Run Test-7	Overlapping Template Test-8
Sample number (S1-box and S2-box) applied in the ZUC								
Sample 1 (R-D)	0.122325	0.066882	0.534146 0.035174	0.350485	0.534146	0.739918	0.534146	0.035174
Sample 2 (D-R)	0.350485	0.534146	0.350485 0.534146	0.066882	0.534146	0.213309	0.534146	0.911413
Sample 3 (R-INR)	0.122325	0.350485	0.739918 0.739918	0.213309	0.350485	0.534146	0.350485	0.213309
Sample 4 (INR-R)	0.534146	0.911413	0.213309 0.213309	0.534146	0.534146	0.739918	0.739918	0.911413
Sample 5 (D-INR)	0.534146	0.534146	0.534146 0.035174	0.213309	0.350485	0.017912	0.534146	0.911413
Sample 6 (INR-D)	0.350485	0.739918	0.350485 0.350485	0.350485	0.035174	0.534146	0.534146	0.739918
Sample 7 (NR-F)	0.534146	0.350485	0.066882 0.350485	0.213309	0.534146	0.350485	0.991468	0.350485
Sample 8 (F-NR)) as in 3GPP-ZUC	0.534146	0.739918	0.122325 0.350485	0.739918	0.122325	0.350485	0.213309	0.066882

(*Continued*)

Table 11 Continued

NIST Test P-value	Approximate Entropy Test-1	Block Frequency Test-2	Cumulative Sums (Forward/Reverse) Test-3	DFT Test-4	Frequency Test-5	Linear Complexity Test-6	Longest Run Test-7	Overlapping Template Test-8
Sample 9 (F-INR)	0.739918	0.066882	0.213309 0.350485	0.350485	0.534146	0.017912	0.739918	0.911413
Sample 10 (INR-F)	0.534146	0.739918	0.066882 0.534146	0.739918	0.911413	0.122325	0.739918	0.350485
Sample 11 (NR-INR)	0.911413	0.122325	0.739918 0.350485	0.066882	0.911413	0.350485	0.350485	0.739918
Sample 12 (INR-NR)	0.534146	0.350485	0.534146 0.534146	0.911413	0.350485	0.739918	0.534146	0.008879
Sample 13 (R-NR)	0.122325	0.534146	0.911413 0.213309	0.739918	0.213309	0.739918	0.350485	0.122325
Sample 14 (NR-R)	0.739918	0.739918	0.534146 0.739918	0.911413	0.534146	0.991468	0.035174	0.534146
Sample 15 (R-F)	0.213309	0.534146	0.017912 0.350485	0.534146	0.122325	0.534146	0.739918	0.004301
Sample 16 (F-R)	0.350485	0.534146	0.122325 0.350485	0.017912	0.534146	0.534146	0.213309	0.911413
Sample 17 (D-NR)	0.534146	0.739918	0.122325 0.739918	0.739918	0.350485	0.739918	0.534146	0.122325
Sample 18 (NR-D)	0.911413	0.017912	0.122325 0.350485	0.350485	0.213309	0.350485	0.534146	0.122325
Sample 19 (D-F)	0.739918	0.213309	0.350485 0.534146	0.066882	0.534146	0.739918	0.350485	0.534146
Sample 20 (F-D)	0.534146	0.122325	0.739918 0.350485	0.350485	0.739918	0.350485	0.534146	0.350485

(*Continued*)

Table 12 NIST SP800-22 statistical tests (Approximate Entropy, Block Frequency, Cumulative Sums, DFT, Frequency, Linear Complexity, Longest Run, and Overlapping Template tests) verifications (P-values) for the different twenty samples of configuration S-boxes used in the ZUC algorithm

NIST Test P-value	Non Overlapping Test-9	Random Excursions Test-10		Random Excursions Variant Test-11		Rank Test-12	Runs Test-13	Serial (P1&P2) Test-14	Universal Test-15
Sample number (S1-box and S2-box) applied in the SNOW algorithm	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value				
Sample 1 (R-D)	19	2	0	2	0	0.350485	0.911413	0.350485 0.739918	0.534146
Sample 2 (D-R)	11	4	1	4	2	0.350485	0.350485	0.534146 0.122325	0.350485
Sample 3 (R-INR)	15	5	0	5	2	0.035174	0.213309	0.911413 0.350485	0.122325
Sample 4 (INR-R)	19	5	0	5	0	0.350485	0.350485	0.739918 0.739918	0.534146
Sample 5 (D-INR)	15	5	1	5	0	0.213309	0.739918	0.213309 0.991468	0.213309
Sample 6 (INR-D)	17	2	4	2	5	0.035174	0.213309	0.739918 0.911413	0.534146
Sample 7 (NR-F)	15	6	0	6	0	0.122325	0.534146	0.534146 0.350485	0.739918

(*Continued*)

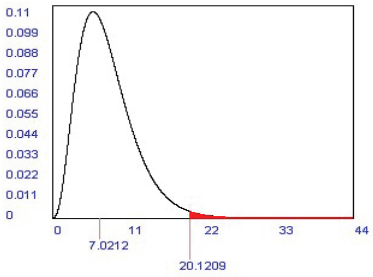
Table 12 Continued

NIST Test P-value	Non Overlapping Test-9	Random Excursions Test-10		Random Excursions Variant Test-11		Rank Test-12	Runs Test-13	Serial (P1&P2) Test-14	Universal Test-15
Sample number (S1-box and S2-box) applied in the SNOW algorithm	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value				
Sample 8 (F-NR) as in 3GPP-ZUC	20	2	1	2	4	0.739918	0.534146	0.739918 0.911413	0.350485
Sample 9 (F-INR)	10	5	0	5	0	0.350485	0.534146	0.739918 0.739918	0.739918
Sample 10 (INR-F)	14	1	4	1	5	0.008879	0.739918	0.350485 0.350485	0.991468
Sample 11 (NR-INR)	17	6	1	6	2	0.534146	0.739918	0.911413 0.534146	0.739918
Sample 12 (INR-NR)	23	2	2	2	2	0.911413	0.534146	0.739918 0.911413	0.739918
Sample 13 (R-NR)	20	5	0	5	1	0.350485	0.911413	0.534146 0.534146	0.350485
Sample 14 (NR-R)	14	5	0	5	0	0.911413	0.534146	0.739918 0.534146	0.534146
Sample 15 (R-F)	13	5	0	5	3	0.911413	0.017912	0.739918 0.122325	0.122325

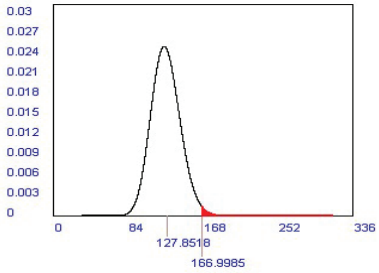
(*Continued*)

Table 12 Continued

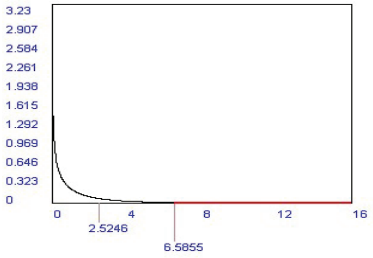
NIST Test P-value	Non Overlapping Test-9	Random Excursions Test-10		Random Excursions Variant Test-11		Rank Test-12	Runs Test-13	Serial (P1&P2) Test-14	Universal Test-15
Sample number (S1-box and S2-box) applied in the SNOW algorithm	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value	Total No. of Discontinued the Test	Total No. of Failed P-value				
Sample 16 (F-R)	14	5	0	5	3	0.035174	0.122325	0.739918 0.739918	0.350485
Sample 17 (D-NR)	14	6	0	6	0	0.122325	0.739918	0.350485 0.534146	0.911413
Sample 18 (NR-D)	18	2	1	2	1	0.739918	0.534146	0.122325 0.350485	0.534146
Sample 19 (D-F)	16	3	0	3	0	0.350485	0.534146	0.066882 0.213309	0.122325
Sample 20 (F-D)	17	5	0	5	1	0.534146	0.911413	0.739918 0.534146	0.350485



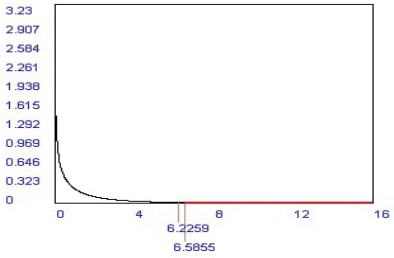
(a) Approximate Entropy Test



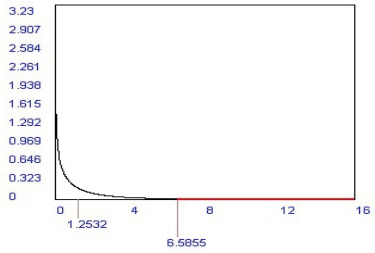
(b) Block Frequency Test



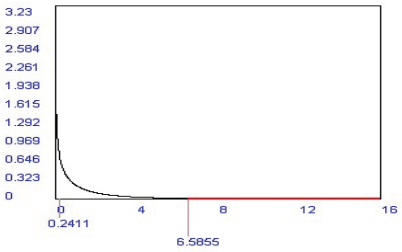
(c) Cumulative Sums Test



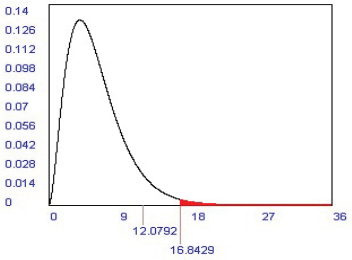
(d) DFT Test



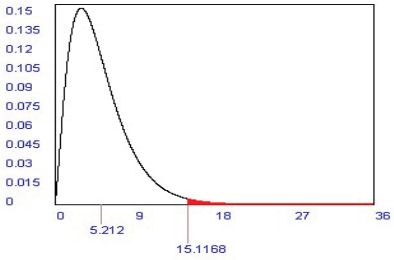
(e) Frequency Test



(f) Linear Complexity Test

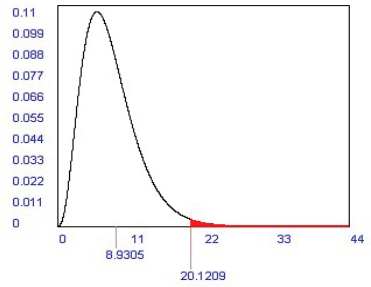


(g) Longest Run Test

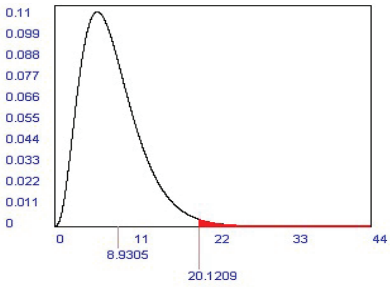


(h) Overlapping Template Test

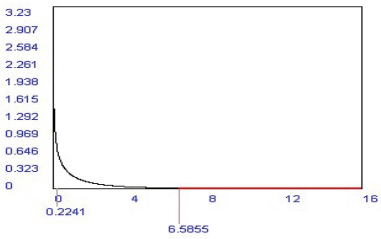
Figure 6 Continued



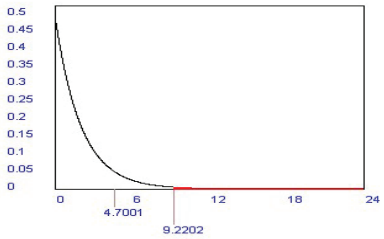
(i) Non-Overlapping Template Test



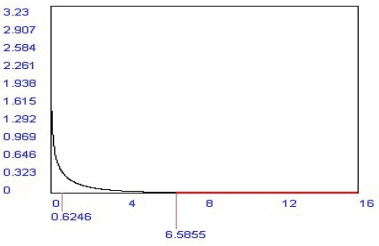
(j) Random Excursions Test



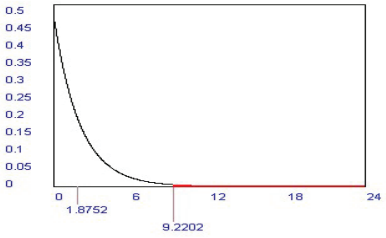
(k) Random Excursions Variant Test



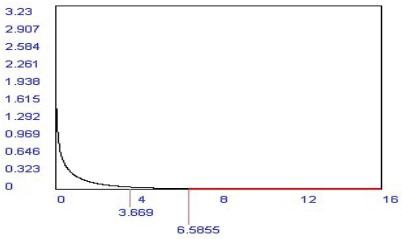
(l) Rank Test



(m) Runs Test



(n) Serial Test



(o) Universal Test

Figure 6 (a–o) NIST SP 800-22 results showed that the upgrading ZUC algorithm by using the best permutation (Sample (14) New Rijndael S1-box and Rijndael S2-box) passed all NIST randomness tests successfully.

Table 13 Comparison between the existing proposes of ZUC that verified by NIST SP 800-22 statistical test suite

NIST SP 800-22 Tests	3GPP Standardized ZUC (Feistel – New Rijndael S-boxes)	Proposal The Best Arrangement Pair of S-boxes in ZUC (New Rijndael – Rijndael S-boxes)
Approximate Entropy	Success	Success
Block Frequency	Success	Success
Cumulative Sums	Success	Success
DFT	Success	Success
Frequency	Success	Success
Linear Complexity	Success	Success
Longest Runs	Success	Success
Non-Overlapping Template	Success	Success
Overlapping Template	Success	Success
Random Excursions	Fail	Success
Random Excursions Variant	Fail	Success
Rank	Success	Success
Runs	Success	Success
Serial	Success	Success
Universal	Success	Success

the test = 2 from 10 iterations of P-values (J<500) and number of failed P-value = 1 from 8 subtests), failed in Random Excursions Variant test (number of discontinuities the test = 2 and number of failed P-value = 4 from 18 subtests) weak in Non-Overlapping Template test (total number of failed P-value = 20 from 148 subtests), and weak in Overlapping Template test (P = 0.066882).

- Best permutation (Sample (14) (New Rijndael (NR) S1-box and Rijndael (R) S2-box): passes all the NIST SP 800-22 randomness tests successfully as shown in Figures 6(a–o). Table 13 shows the comparison between the existing proposes of ZUC that verified by NIST SP 800-22 tests.

6 Conclusions

In this paper, we investigate the randomness properties of the keystream sequences produced by the two algorithms (SNOW and ZUC) that are used for confidentiality and integrity processes in advanced mobile communication systems. NIST 800-22 statistical test suite with its 15 tests is used for this purpose. Twenty different configurations of pair S-boxes in the nonlinear layer structure of the two algorithms are tested to select the best permutation (S1 and S2 boxes).

A complete simulation of SNOW and ZUC algorithms is presented using C-language. Samples of the keystream sequences with the length ten Megabits are used (according to the NIST standard). Test results showed that the best pair arrangement of S-boxes in the SNOW algorithm is the configuration (Feistel structure – Rijndael S-boxes) although the standard configuration by 3GPP is (Rijndael (R)-Dickson (D)S-boxes). Also, the best configuration in the ZUC algorithm is (New Rijndael – RijndaelS-boxes) although the standard configuration by 3GPP is (Feistel structure-New RijndaelS-boxes).

The best configurations passed all the NIST suite tests successfully. Applying the best configurations of S-boxes will lead to better randomness properties of the keystream sequences generated by SNOW and ZUC stream cipher algorithms and therefore increase the security level of both algorithms.

References

[1] 3GPP TS 35.501, Third Generation Partnership Project, Technical Specification Group Services and System Aspects, Security architecture and procedures for 5G system, March 2020.

[2] 3GPP TS 35.401, Third Generation Partnership Project, Technical Specification Group Services and System Aspects, 3GPP System Architecture Evolution (SAE), and security architecture for 4G, March 2020.

[3] 3GPP TS 35.216, Specification of the 3GPP Confidentiality and Integrity algorithms UEA2 and UIA2, Document 2: SNOW 3G specification, June 2018.

[4] 3GPP TS 35.222, Specification of the 3GPP Confidentiality and Integrity algorithms EEA3 and EIA3, Document 2: ZUC specification, June 2018.

[5] Patrick Bohm, 'Statistical Evaluation of Stream Cipher SNOW 3G', In Constantin Brancusi University of Targu Jiu Engineering Faculty

Scientific Conference with international participation, 13th edition, pp. 363–366, Targu Jiu, 7–8 Nov., 2008.

[6] Mahdi Madani, Ilyas Benkhaddra, Camel Tanougast, Salim Chitroub, and Loic Sieler, 'FPGA Implementation of an enhanced SNOW-3G Stream Cipher based on a Hyper-Chaotic System', In the 4th international conference on Control, Decision and Information Technologies, IEEE, pp. 1168–1173, Barcelona, Spain, 5–7 April 2017.

[7] Mahdi Madani, Ilyas Benkhaddra, Camel Tanougast, Salim Chitroub, and Loic Sieler, 'Digital Implementation of an Improved LTE Stream Cipher Snow-3G Based on Hyperchaotic PRNG', Security and Communication Networks, Wiley and Hindawi, 15 pages, 2017.

[8] Mahdi Madani, Ilyas Benkhaddra, Camel Tanougast, Salim Chitroub, and Loic Sieler, 'Enhanced ZUC Stream Cipher Based on a Hyperchaotic Controller System', In the Euromicro Conference on Digital System Design (DSD), Vienna, Austria, 30 Aug.–1 Sept.2017.

[9] Aleksandar Kircanski and Amr M. Youssef, 'On the sliding property of SNOW 3G and SNOW 2.0', IET Information Security, Vol. 5(4), pp. 199–206, 2011.

[10] 3GPP TS 35.919, Specification of the 3GPP Confidentiality and Integrity algorithms UEA2 and UIA2, Document 5: Design and Evaluation report, June 2018.

[11] Alex Biryukov, Deike Priemuth-Schmid, and Bin Zhang, 'Fault Analysis of the Stream Cipher Snow 3G', In 6^{th} International Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), IEEE, pp. 103–110, Lausanne, Switzerland, Sept., 2010.

[12] Alex Biryukov, Deike Priemuth-Schmid, and Bin Zhang: "A Timing Attack on SNOW 3G," In International Conference on Information and Communications Security, pp. 171–185, Spain, Dec., 2010.

[13] Alex Biryukov, Deike Priemuth-Schmid, and Bin Zhang, 'Multiset Collision Attacks on Reduced-Round SNOW 3G and SNOW 3G$\oplus$', In International Conference on Applied Cryptography and Network Security, pp. 191–198, China, Dec., 2010.

[14] Mohammad Sadegh Nemati Nia and Malek e Ashtar, 'Improved Heuristic guess and determine attack on SNOW 3G stream cipher', In 7th International Symposium on Telecommunications (IST), IEEE, pp. 972–976, Iran, Sept., 2014.

[15] Mufeed Juma AlMashrafi, 'A different algebraic analysis of the ZUC stream cipher', In Proceedings of the 4th International Conference on

Security of Information and Networks (SIN), pp. 191–198, Australia, Nov., 2011.

[16] 3GPP TR 35.924, Specification of the 3GPP Confidentiality and Integrity Algorithms EEA3 & EIA3, Document 4: Design and Evaluation report, June 2018.

[17] Wu Hongjun, Huang Tao, Ha.Nguyen Phuong, Wang Huaxiong, and Ling San, 'Differential Attacks against Stream Cipher ZUC', In 18th International Conference on the Theory and Application of Cryptology and Information Security, Springer, pp. 262–277, China, 2–6 Dec., 2012.

[18] National Institute of Standards and Technology, A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST Special publication 800-22, April 2010.

[19] Carmina Georgescu, Emil Simion, Alina Petrescu Nita, Antonela Toma, 'A view on NIST randomness tests independence', Proc. 9th International Conference on Electronics, Computers, and Artificial Intelligence, IEEE, Romania, 29 June–1 July 2017.

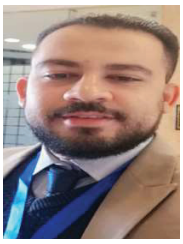
[20] Jie Cui, Hong Zhong, Jiankai Wang, Runhua Shi, 'Generation and optimization of Rijndael S-box equation system', Information Technology Journal, Vol. 13(15), pp. 2482–2488, 2014.

[21] Jie Cui, Liusheng Huang, Hong Zhong, Chinchen Chang, Wei Yang: 'An improved AES S-box and its performance analysis', International Journal of Innovative Computing, Information, and Control, Vol. 75(A), pp. 2291–2302, 2011.

[22] 3GPP TS 35.217, Specification of the 3GPP Confidentiality and Integrity algorithms UEA2 and UIA2, Document 3: Implementer test data, June 2018.

[23] 3GPP TS 35.223, Specification of the 3GPP Confidentiality and Integrity algorithms EEA3 & EIA3, Document 3: Implementer test data, June 2018.

Biographies



Zakaria Hassan Abdelwahab was born in Ismailia, Egypt, in 1987. He received the B.S. degree in electrical engineering from the Higher Technological Institute (HTI), Tenth of Ramadan City, Egypt, in 2009, the M.S. degree in electrical engineering (electronics and communications engineering dept.) from the faculty of Engineering, Ain Shams University, Cairo, Egypt, in 2014 and the Ph.D. degree in electrical engineering (electronics and communications) at the faculty of Engineering, Ain Shams University, Egypt in 2020. In 2009, Zakaria joined the department of electrical engineering, HTI, as a teaching assistant and he is a member of the Egyptian engineering syndicate since 2009. His main areas of research interest are security telecommunications and network security.

Talaat A. Elgarf was born in Telwana-Menofia, Egypt, in 1953. He received the B.S. degree in electrical engineering (communications), from the Military Technical College, Cairo, Egypt, in 1976, the M.S. degree in electrical engineering from the faculty of Engineering (electronics and communications), Ain Shams University, Egypt, in 1990 and the Ph.D. degree in electrical engineering (electronics and communications) from the faculty of Engineering, Ain Shams University, Egypt, in 1993. He is currently visiting professor, Military Technical College, faculty of Engineering, Ain Shams University,

Cairo, Egypt, and professor of communications, Higher Technological Institute (HTI), Tenth of Ramadan City, Egypt, since 2005.

Abdelhalim Zekry is a professor of electronics and communications at the faculty of Engineering, Ain Shams University, Cairo, Egypt. He worked as a staff member in several universities. He published more than 240 conference and periodical papers. He also supervised more than 110 Master thesis and 30 Doctorate thesis in the area of electronics and electronics for communications as well as photovoltaics. Prof. Zekry focuses his research programs on the field of microelectronics and electronic applications including communications and photovoltaics.