A Novel SNOW3G-M Algorithm and Medical

https://doi.org/10.3991/ijoe.v18i03.28013

Rana M. Zaki(✉), Hala Bahjat Abdul Wahab
Computer Science Department, University of Technology, Baghdad, Iraq
Rana.M.Zaki@uotechnology.edu.iq

Abstract—In the past two years, the demand for the use of mobile networks has increased, due to what the world has been exposed to in the face of the COVID-19 pan-demic and the lack of communication between people. People resorted to using of mobile networks in light of the pandemic, and their importance has appeared in several aspects, such as automotives, media and entertainment, and healthcare. This growing demand for the mobile network led to the actual development of the 4G network in terms of providing the speed of transmission and encryption data to maintain security. In this paper, a new member of the SNOW 3G family was proposed, which is one of the fourth generation algorithms called SNOW 3G-M, which has a higher encryption speed in line with the capacity of modern CPUs and is expected to be a move to the fifth generation communication system in the future. The SNOW 3G-M keystream passes all of the tests for long key stream data, short key stream data, and initialization vector data sets.

Keywords—4G mobile security, SNOW3G, stream cipher, NIST-test

1 How to work with this template

For the time being, the security of communications has become an additional and important element at present to ensure the transmission of data over networks is secure [1, 2]. Sure, all of our daily transactions and communications in modern life are done through mobile networks [3, 4]. The 3G and 4G communications technologies that are becoming mobile emigrations of online applications such as Voice over IP (VoIP), video streaming, music download files, mobile communication, and so on are the Universal Mobile Telecommunications System (UMTS), Long Term Evolution (LTE) and LTE advanced standards use stream encryption algorithm such as the snow algorithm, which is the heart of mobile standards' security and is the one viewed as the seed of the 128-EEA1 confidentiality and 128-EIA1 integrity algorithms of 4G LTE security [5–8], which removes the f8 and f9 algorithms of UMTS security [9–11]. Following that, the Third Generation Partnership Project (3GPP) standardized two LTE kernel algorithms.

Snow 3G's algorithm suffers from several weaknesses, the first of which is the shortness of the generated keystream that helps the attack [12], secondly, the cache-timing attack [13], which can recover the filled cipher state in a matter of seconds using empirical timing data: The attack takes advantage of the cipher by feeding the output of one S-box into another S-box. The third is the fault attack on the Linear Feedback Shift Register (LFSR) that retrieves the secret key with only 22 error injec-tions: The attack model suggests that an attacker can change a 32-bit value of one LFSR state all through keystream generation, where i is selected by an attacker, but he does not have complete control over t [15]. Fourth, one such attack uses multiset collision attacks to investigate the SNOW3G resynchronization process. This method has been shown to be effective against AES [16–19]. Because SNOW3G's Finite State Machine (FSM) is a 96-bit AES-like algorithm. Cipher in which the LFSR serves as a key-schedule, this technique can be used. The attacker sees only 32 bits of out-put at a time, while the internal process changes constantly [20]. Because there is feedback from the FSM to the LFSR during the initialization phase, and the attacker needs to accept 32 bits of output during the period, this attack is complex. In this paper, we developed SNOW3G to SNOW3G-M (M for multi-LFSR) with the use of two LFSRs (Linear Feedback Shift Registers) and a FSM (Finite State Machine) updated to best align with vectorized applications by increasing the total size to 64 bits for increased security by encrypting data. In the long run, this proposal provides perfect randomness, and the Snow3G-M algorithm has the strongest because the shortness of the generated keystream does not help the attack and increases the complexity of a cryptanalysis attack due to the failure of the uniform SNOW-3G cipher's initialization method [10]. The remainder of this paper is structured as follows: Section 2: Explain the conventional SNOW3G architecture.Section3 shows the new design and step architecture of the SNOW3G-M. Section 4 results, with evaluation and comparison with the SNOW3G standard. Section 5 explains how the SNOW3G-M algorithm is important in medical use. Section 6 dis-cussions about the proposed algorithm and section 7 conclusions of this paper.

1.1 Group of symbols

= Operator for assigning	
Modulo addition of integers	⊕ Exclusive-OR bitwise operation
|| two operands are concatenated.	<<_(n,t) n-bit register, t-bit left shift.

2 SNOW3G stream cipher traditional

It is one of the important encryption algorithms in the fourth generation that was developed by Thomas Johansson et al. at Lund University. In 2006, it was selection as the prime part of the second set of Universal Mobile Telecommunications System (UMTS) confidentiality and integrity algorithms. In 2008 Böhm using NIST statistical test wing as randomness test gadget with three kinds of test. SNOW3G Down the hold of the 128-bit Cipher Key and 128-bit Initialization Vector (IV), a 32-bit word key-stream is created. It is divided into two layers, each of which is reactive [11].

The LFSR is the first layer that contain 16 stages of 32 bits every ($S0$, $S1$,….., $S15$).

s15 = k3 ⊕ IV0	s14 = k2 s13 = k1	s12 = k0 ⊕ IV1	
s11 = k3 ⊕ 1	s10 = k2 ⊕ 1 ⊕ IV2	s9 = k1 ⊕ 1 ⊕ IV3	s8 = k0 ⊕ 1
s7 = k3 s6 = k2	s5 = k1 s4 = k0		
s3 = k3 ⊕ 1	s2 = k2 ⊕ 1	s1 = k1 ⊕ 1	s0 = k0 ⊕ 1

The FSM is the second layer that contain which three registers (R1, R2, R3) 32 bits per register and have two S-Boxes (⟦ S⟧_R, ⟦ S⟧_Q) 8 bit to 8 bit mapping. The information is presented in hexadecimal format. Figures 1 and 2 depicts the SNOW-3G stream cipher's architecture in detail.

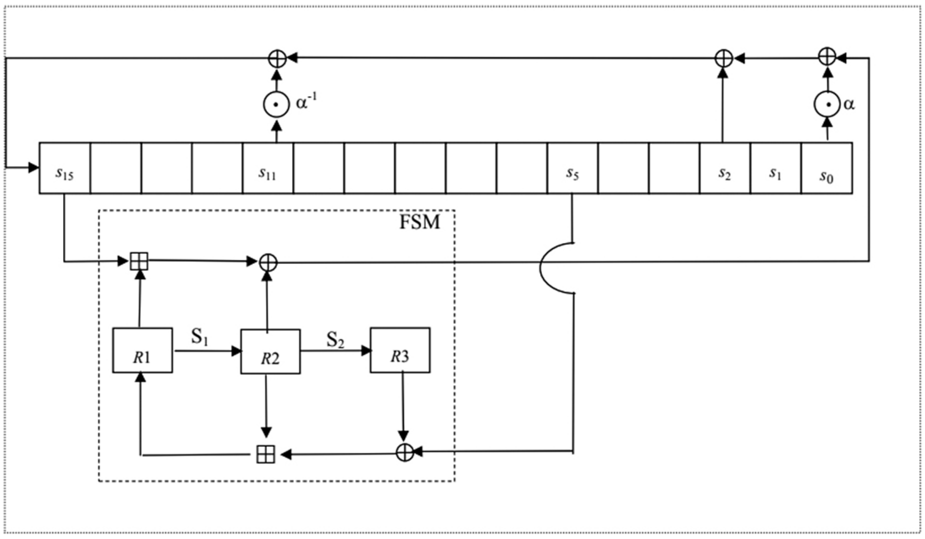

Fig. 1. Throughout key initialization, the SNOW 3G algorithms are used [11]

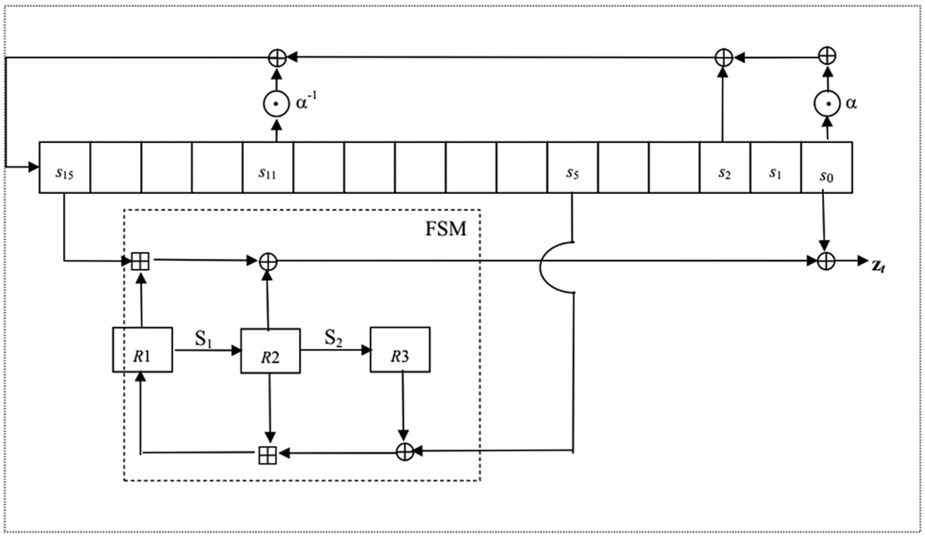

Fig. 2. Throughout key generation, the SNOW 3G algorithms are used [11]

Given that the traditional SNOW 3G is vulnerable to shortened keystream data of set attacks [14], it is proposed to improve its statistical property and security level while maintaining the standard by overcoming its flaws in the initialization and keystream modes.

3 The new design of the SNOW3G-M algorithm

Snow 3G-M architecture has LFSR and FSM, but with a different design than traditional SNOW 3G, and we will explain its design in the steps below

3.1 LFSR (Linear Feedback Shift Register)

In the snow 3G-M algorithm, two LFSRs are used instead of one, called (LFSR-R, LFSR2), both of which have 16 stages, each stage consisting of 32 bits and represented in F(2^16). LFSR-R (r0, r1, r2,....., r14, r15) and LFSR-H (h0, h1, h2,...., h14, h15). The polynomial equation for LFSR-R is:

$$R(x) = x^{16} + x^{12} + x^{3} + x^{1} + 1 \qquad (1)$$

The polynomial equation for LFSR-R is:

$$H(x) = x^{16} + x^{12} + x^{4} + x^{2} + x^{1} + 1 \qquad (2)$$

Functions used in various components of the LFSR. The SNOW3G-M uses the same function as the SNOW3G traditional [11].

i. **The function MUL α:** MULα is a function that converts 8 bits to 32 bits. Allow for 8-bit input values for V1 and C1. MUL x is then defined as follows:

   If the leftmost (most significant) bit V Equals 1, Then

   MULx (V1, C1) = $(V1 \ll_8 1) \oplus C1$

   Else

   MULx (V1, C1) = $V1 \ll_8 1$

ii. **The function DIVα:** DIVα is a function that converts 8 bits to 32 bits. Allow for 8-bit input values for V1 and C1. MUL x is then defined as follows:

   If rightmost (lest significant) bit of V1 Equals 1, Then

   DIVα (V1, C1) = $(V1 \ll_8 1) \oplus C1$

   Else

   DIVx (V1, C1) = $V1 \ll_8 1$

iii. **MULxPOW:** MULxPOW is a function that converts 16 bits and a positive integer i1 to 8 bits. MULxPOW (V, i1, c) is recursively defined if V1 and C1 are 8-bit input values [11]:

   If i1 Equals 0, Then

   MULxPOW (V1, i1, C1) = V,

Else

MULxPOW (V1, i1, C1) = MULx (MULxPOW (V1, i1–1, C1), C1).

3.2 FSM (Finite State Machine)

- The FSM has three registers (R1, R2, and R3) each register have 64 bits.
- The FSM has four substitution-boxes (S1, S2, S3, and S4) called SR1, SR2, SM1 and SM2 respectively, that are used to update the R2 and R3 registers, as shown in Figure 3. In the SNOW 3G traditional algorithm, the Rijndael S-box-SR and the S-box-SQ were used, both of which used 32-bit input to a 32-bit output. As for the snow 3G-M algorithm, S1, S2, S3 and S4 were reconstructed according to the Permutation of Last Layer algorithms (PLL) algorithm to replace the place (permutation) of the pieces generated from the Magic Cube and, instead of using static S-Boxes, this algorithm was used to build s-boxes that enjoyed high confusion. The PLL algorithm has the ability to generate two million S-Boxes when there is a change in the permutation of position by using permutation, but we adopted the use of only two, which are S-Box (SR1) and S-Box (SR2), which are shown in Tables 1 and 2 below, where the tables were built in hexadecimal, which is a two-dimensional matrix. Each position in the matrix is 8 bits, but the SM1 and SM2 represent the inverse of the SR1 and SR2.

Table 1. S-box (SR1) produced by the PLL algorithm

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	FC	2B	6B	06	15	71	E7	79	1E	BF	4D	80	88	41	B6	FE
1	5C	56	73	92	1B	7C	54	B1	A3	3E	91	E6	C3	3A	78	C6
2	6E	59	50	CB	58	26	E3	95	69	34	AC	6A	EE	EB	29	F9
3	00	42	6D	72	9F	37	C7	7A	C4	CA	2C	E8	0A	10	D5	FB
4	76	31	85	A1	B4	98	2F	9D	04	13	AD	74	DC	22	61	4E
5	DF	8D	FF	CF	07	16	65	9C	67	1F	A5	4B	97	CE	47	FA
6	02	0D	77	C2	89	1C	C5	53	B3	A7	48	AA	F2	D3	35	F4
7	5A	19	F1	49	D6	A0	44	5E	BC	E4	38	B0	E5	D2	ED	2A
8	5B	C9	45	75	7B	A6	3C	93	86	82	E2	2D	EA	05	14	E1
9	5D	94	32	7E	A9	BA	87	30	83	08	17	90	8C	F5	1D	FD
A	B5	F8	9B	27	CD	0B	11	C8	9E	60	1A	D1	4C	D7	81	3D
B	5F	01	0E	68	D0	7D	23	B9	52	99	AB	3F	DE	8F	C0	3B
C	57	AE	20	66	51	B7	D8	43	DD	D4	C1	39	B8	DA	40	EF
D	4F	62	A2	46	8A	64	96	33	A4	BD	D9	E9	2E	E0	0C	18
E	A8	F3	6C	36	84	8E	6F	BB	25	EC	09	0F	7F	9A	F7	24
F	55	70	8B	63	28	CC	03	12	BE	AF	B2	21	F0	4A	F6	DB

Table 2. S-box (SR2) produced by the PLL algorithm

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	06	0B	03	04	05	09	0A	02	01	0C	08	07	0D	11	0F	10
1	0E	12	13	17	15	16	14	18	1F	1A	1B	1C	23	22	21	1D
2	19	24	20	1E	2B	2C	27	28	29	30	25	26	2D	2E	2F	2A
3	31	38	39	3A	35	36	37	32	33	34	3B	3C	3D	41	3F	40
4	3E	46	43	44	45	42	47	48	49	4A	51	52	50	4E	4F	4D
5	4B	4C	53	54	5B	56	55	60	59	58	57	5C	5D	5E	5F	5A
6	61	68	63	64	6B	66	67	62	69	6A	65	6C	75	74	6D	78
7	71	72	73	77	6F	70	6E	76	79	7A	7B	7C	7D	82	81	83
8	7F	7E	80	84	85	86	8B	8A	89	8E	8D	8C	87	88	8F	90
9	99	92	97	96	95	94	93	98	91	9C	9B	9A	9D	A7	9F	A0
A	9E	A2	A3	A4	A5	A6	A1	A8	AB	AA	A9	B4	B3	AE	AF	B0
B	B1	B2	AD	AC	B5	B9	B7	B8	BF	BA	BB	BC	BD	BE	B6	C0
C	C1	C2	C7	C6	CB	C4	C3	C8	C9	CA	C5	CC	CD	D7	CE	D0
D	D1	D6	D5	D4	D3	D2	CF	D8	D9	DD	DF	DE	DA	DC	DB	E0
E	El	E2	E3	E4	ED	E9	E5	F0	EF	EA	EB	EC	E7	E8	E6	EE
F	FI	F2	FA	F9	F5	F6	F7	F8	F4	F3	F8	FC	FF	00	FD	FE

3.3 SNOW3G-M of operations

Initialization Operation. The SNOW3G-M algorithm has a key (K) and an initial vector (IV) length of 128 bits divided into four words. Every word has 32 bits.

W3	W2	W1	W0
K3=32bits	K2=32bits	K1=32bits	K0=32bits
IV3=32bits	IV1=32bits	IV2=32bits	IV0=32bits

We fill in the LFSR-R and LFSR-H assuming that the value of one is (0xffffffff).

LFSR-R:

$r_{15} = K_3 \oplus IV_0$ $r_{14} = K_2$ $r_{13} = K_1$ $r_{12} = K_0 \oplus IV_1$
$r_{11} = K_3 \oplus (0\text{xffffffff})$ $r_{10} = K_2 \oplus (0\text{xffffffff}) \oplus IV_2$ $r_9 = K_1 \oplus (0\text{xffffffff}) \oplus IV_3$
$r_8 = K_0 \oplus (0\text{xffffffff})$ $r_7 = K_3$ $r_6 = K_2$ $r_5 = K_1$ $r_4 = K_0$
$r_3 = K_3 \oplus (0\text{xffffffff})$ $r_2 = K_2 \oplus (0\text{xffffffff})$ $r_1 = K_1 \oplus (0\text{xffffffff})$
$r_0 = K_0 \oplus (0\text{xffffffff})$

LFSR-H:

$\mathrm{h}_{15} = \mathrm{K}_3 \oplus \mathrm{IV}_0$ $\mathrm{h}_{14} = \mathrm{K}_3$ $\mathrm{h}_{13} = \mathrm{K}_2$ $\mathrm{h}_{12} = \mathrm{K}_1$ $\mathrm{h}_{11} = \mathrm{K}_0$
$\mathrm{h}_{10} = \mathrm{K}_3 \oplus (0\text{xffffffff})$ $\mathrm{h}_9 = \mathrm{K}_2 \oplus (0\text{xffffffff})$ $\mathrm{h}_8 = \mathrm{K}_1 \oplus (0\text{xffffffff})$
$\mathrm{h}_7 = \mathrm{K}_3 \oplus (0\text{xffffffff})$ $\mathrm{h}_6 = \mathrm{K}_2 \oplus (0\text{xffffffff}) \oplus \mathrm{IV}_2$ $\mathrm{h}_5 = \mathrm{K}_1 \oplus (0\text{xffffffff}) \oplus \mathrm{IV}_3$
$\mathrm{h}_4 = \mathrm{K}_0 \oplus (0\text{xffffffff})$ $\mathrm{h}_3 = \mathrm{K}_2$ $\mathrm{h}_2 = \mathrm{K}_1$ $\mathrm{h}_1 = \mathrm{K}_0 \oplus IV_1$ $\mathrm{h}_0 = K_0 \oplus (0\text{xffffffff})$

The initialized Finite State Machine (FSM) to three registers is:

$$R1=R2=R3=0$$

The process of initializing the key takes 32 clocks and consists of two steps:

Step one:

The FSM has two input from LFSR2-H ($h_{14}||h_{15}$) and LFSR1-R ($r_0||r_1$) to produce F have 64 bits (two words), according to the equation below:

$$x = (h_{14}||h_{15}) \boxplus R1 \tag{3}$$

$$F = x \oplus R2 \tag{4}$$

The registers are then updated. Calculate the intermediate value r according to the equation below:

$$a = R3 \oplus (r_0||r_1) \tag{5}$$

$$r= R2 \boxplus a \tag{6}$$

Then, to update (R1, R2 and R3) by used S-Boxes built using the PLL algorithm (see 3.2). Follow as

$$R1=r \tag{7}$$

$$R2= SR\ (R1) \tag{8}$$

$$R3= SM\ (R2) \tag{9}$$

The S-Box is based on the Permutation of Last Layer algorithms (PLL). To generate values [0–255], they used permutation position to construct S-Box 16*16.

Look a 64-bit input w = w0 || w1 || w2 || w3. Each W0, W1, W2 and W3 has 16 bits. Hence Wi = $w_{i0} || w_{i1} || w_{i2} || w_{i3} || w_{i4} || w_{i5} || w_{i6} || w_{i7} || w_{i8} || w_{i9} || w_{i10} || w_{i11} || w_{i12} || w_{i13} || w_{i14} || w_{i15}$ with wi0 the most significant bit and w_{i15} the least significant bit. We will divide 16 bits into two bytes. Each byte enters SR1 and SR2 then the result is worked for concatenation by following the equation to the S-Boxes (SR1, SR2):

$$Sr_{w0} = SR1\ (W_0)\ ||\ SR2\ (W_0) \tag{10}$$

$$Sr_{w1} = SR1\ (W_1)\ ||\ SR2\ (W_1) \tag{11}$$

$$Sr_{w2} = SR1\ (W_2)\ ||\ SR2\ (W_2) \tag{12}$$

$$Sr_{w3} = SR1\ (W_3)\ ||\ SR2\ (W_3) \tag{13}$$

Then the output S1 (w) = r = r0 || r1 || r2 || r3 using these equations

$$r0 = (MULx(Sr_{w0},0x1B1B) \oplus (Sr_{w1}) \oplus (Sr_{w2}) \oplus (MULx(Sr_{w3},0x1B1B)\oplus(Sr_{w3}))) \quad (14)$$

$$r1 = ((MULx\ (Sr_{w0},0x1B1B) \oplus (Sr_{w0})) \oplus MULx(Sr_{w1},0x1B1B) \oplus (Sr_{w2}) \oplus (Sr_{w3})) \quad (15)$$

$$r2 = ((Sr_{w0}) \oplus (MULx(Sr_{w1},0x1B1B)\oplus(Sr_{w1})) \oplus MULx(Sr_{w2},0x1B1B) \oplus (Sr_{w3})) \quad (16)$$

$$r3 = ((Sr_{w0}) \oplus (Sr_{w1})\oplus(MULx(Sr_{w2},0x1B1B) \oplus (Sr_{w2})) \oplus MULx(Sr_{w3},0x1B1B)) \quad (17)$$

In the same way as above, the S-Box (SM1, SM2) used to update R3 can be calculated using the equation below.

$$Sm_{w0} = SM1\ (W_0) \parallel SM2\ (W_0) \quad (18)$$

$$Sm_{w1} = SM1\ (W_1) \parallel SM2\ (W_1) \quad (19)$$

$$Sm_{w2} = SM1\ (W_2) \parallel SM2\ (W_2) \quad (20)$$

$$Sm_{w3} = SM1\ (W_3) \parallel SM2\ (W_3) \quad (21)$$

$$r0 = (MULx(Sm_{w0},0x6969) \oplus (Sm_{w1}) \oplus (Sm_{w2}) \oplus (MULx(Sm_{w3},0x6969) \oplus (Sm_{w3}))) \quad (22)$$

$$r1 = ((MULx(Sm_{w0},0x6969) \oplus (Sm_{w0})) \oplus MULx(Sm_{w1},0x6969) \oplus (Sm_{w2}) \oplus (Sm_{w3})) \quad (23)$$

$$r2 = ((Sm_{w0}) \oplus (MULx(Sm_{w1},0x6969) \oplus (Sm_{w1})) \oplus MULx(Sm_{w2},0x6969) \oplus (Sm_{w3})) \quad (24)$$

$$r3 = ((Sm_{w0}) \oplus (Sm_{w1}) \oplus (MULx(Sm_{w2},0x6969) \oplus (Sm_{w2})) \oplus MULx(Sm_{w3}, 0x6969)) \quad (25)$$

Step two:

The output from the FSM represented by (F) with 64bits is input to LFSR1-R and LFSR2-H with 32 bits each. As in the two equations below:

$$V1 = ((r_{0,1} \parallel r_{0,2} \parallel r_{0,3} \parallel 0x00) \oplus MUL\alpha\ (r_{0,0}) \oplus (r_2) \oplus (0x00 \parallel r_{11,0} \parallel r_{11,1} \parallel r_{11,2}) \oplus DIV\alpha(r_{11,3}) \oplus F\ [0:32]) \quad (26)$$

$$V2 = ((h_{0,1} \parallel h_{0,2} \parallel h_{0,3} \parallel 0x00) \oplus MUL\alpha\ (h_{0,0}) \oplus (h_1) \oplus (h_3) \oplus (0x00 \parallel h_{11,0} \parallel h_{11,1} \parallel h_{11,2} \parallel) \oplus DIV\alpha\ (h_{11,3}) \oplus F\ [32:64]) \quad (27)$$

Through these two equations, each stage is updated in LFSR1-R and LFSR2-H, show that

LFSR1-R:
$r_0 = r_1$, $r_1 = r_2$, $r_2 = r_3$, $r_3 = r_4$, $r_4 = r_5$, $r_5 = r_6$, $r_6 = r_7$,
$r_7 = r_8$, $r_8 = r_9$, $r_9 = r_{10}$, $r_{10} = r_{11}$, $r_{11} = r_{12}$, $r_{12} = r_{13}$, $r_{13} = r_{14}$,
$r_{14} = r_{15}$, $r_{15} = v_1$.

LFSR1-H:
$h_0 = r_1$, $h_1 = h_2$, $h_2 = h_3$, $h_3 = h_4$, $h_4 = h_5$, $h_5 = h_6$, $h_6 = h_7$,
$h_7 = h_8$, $h_8 = h_9$, $h_9 = h_{10}$, $h_{10} = h_{11}$, $h_{11} = h_{12}$, $h_{12} = h_{13}$, $h_{13} = h_{14}$,
$h_{14} = h_{15}$, $h_{15} = v_2$.

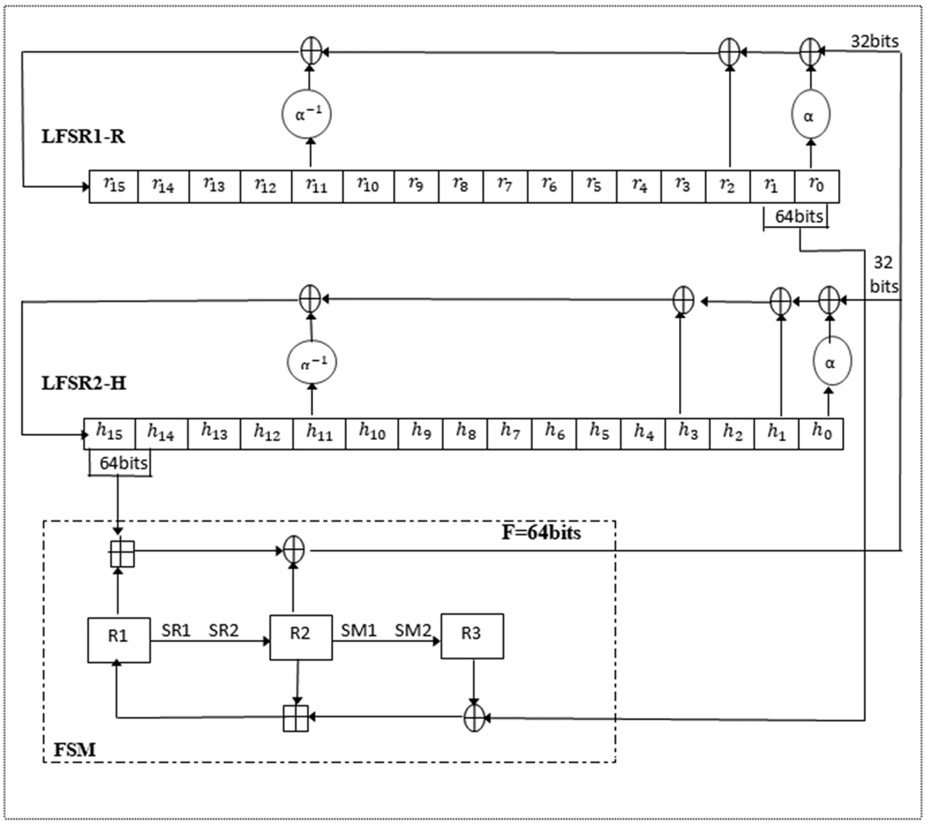
32bits
LFSR1-R
α⁻¹
α
r15 r14 r13 r12 r11 r10 r9 r8 r7 r6 r5 r4 r3 r2 r1 r0
64bits
32 bits
LFSR2-H
h15 h14 h13 h12 h11 h10 h9 h8 h7 h6 h5 h4 h3 h2 h1 h0
64bits
F=64bits
R1
SR1 SR2
R2
SM1 SM2
R3
FSM

Fig. 3. Initialization key in SNOW3G-M

Generation Keystreams Operation. After clock FSM to produce F then keystream generating of length 64 bits, as in the equation below:

$$Z = (F \oplus (r_0 || h_0)) \tag{28}$$

When generating the keystream, there is no entry from FSM to LFSR. That is, there is no F as show Figure 4, so the equation V will be as follows:

$$V1 = ((r_{0,1} \parallel r_{0,2} \parallel r_{0,3} \parallel 0x00) \oplus MUL\alpha\ (r_{0,0}) \oplus (r_2) \oplus (0x00 \parallel r_{11,0} \parallel r_{11,1} \parallel r_{11,2}) \oplus DIV\alpha(r_{11,3})) \quad (29)$$

$$V2 = ((h_{0,1} \parallel h_{0,2} \parallel h_{0,3} \parallel 0x00) \oplus MUL\alpha\ (h_{0,0}) \oplus (h_1) \oplus (h_3) \oplus (0x00 \parallel h_{11,0} \parallel h_{11,1} \parallel h_{11,2} \parallel) \oplus DIV\alpha\ (h_{11,3})) \quad (30)$$

In the Keystream phase, the LFSR is clocked to update the LFSR-R and LFSR-H.

LFSR1-R:
$r_0 = r_1$, $r_1 = r_2$, $r_2 = r_3$, $r_3 = r_4$, $r_4 = r_5$, $r_5 = r_6$, $r_6 = r_7$,
$r_7 = r_8$, $r_8 = r_9$, $r_9 = r_{10}$, $r_{10} = r_{11}$, $r_{11} = r_{12}$, $r_{12} = r_{13}$, $r_{13} = r_{14}$,
$r_{14} = r_{15}$, $r_{15} = v_1$.

LFSR1-H:
$h_0 = r_1$, $h_1 = h_2$, $h_2 = h_3$, $h_3 = h_4$, $h_4 = h_5$, $h_5 = h_6$, $h_6 = h_7$,
$h_7 = h_8$, $h_8 = h_9$, $h_9 = h_{10}$, $h_{10} = h_{11}$, $h_{11} = h_{12}$, $h_{12} = h_{13}$, $h_{13} = h_{14}$,
$h_{14} = h_{15}$, $h_{15} = v_2$.

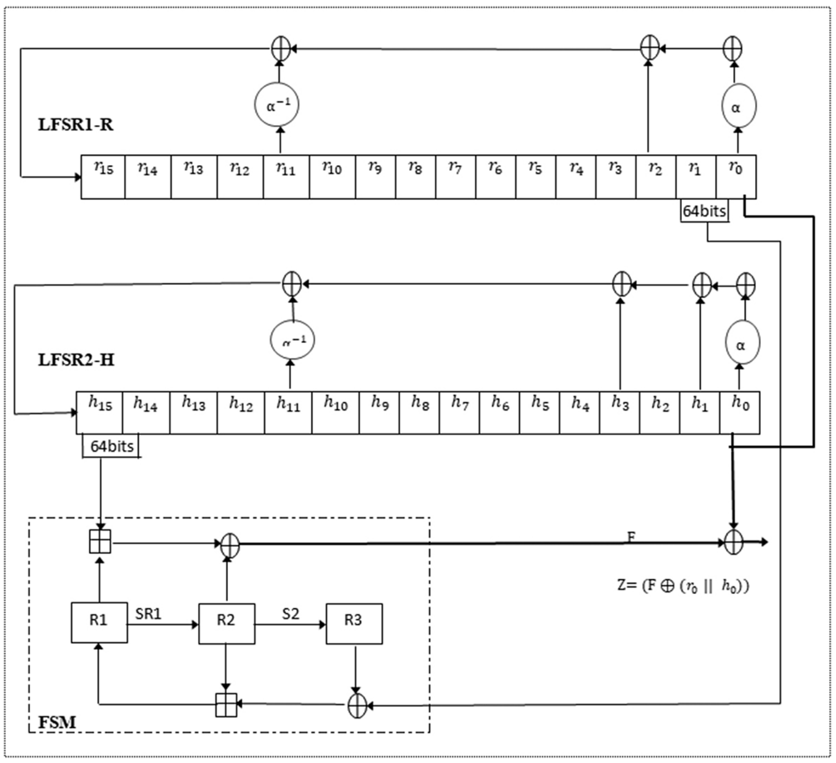
LFSR1-R
α^{-1}
α
r_{15} r_{14} r_{13} r_{12} r_{11} r_{10} r_9 r_8 r_7 r_6 r_5 r_4 r_3 r_2 r_1 r_0
64bits
LFSR2-H
α^{-1}
α
h_{15} h_{14} h_{13} h_{12} h_{11} h_{10} h_9 h_8 h_7 h_6 h_5 h_4 h_3 h_2 h_1 h_0
64bits
F
Z= (F ⊕ (r_0 || h_0))
R1
SR1
R2
S2
R3
FSM

Fig. 4. Generation keystream in SNOW3G-M

4 Results with statistical tests

Our proposed stream cipher plan is subjected to statistical analyses in order to demonstrate its robustness, such as randomness. NIST Test (National Institute of Standards and Technology) has a number of statistical trails that can be applied to assess the randomness of the series, including NIST [21–24], DIEH. [25] and NESS. [26]. The SNOW3G-M stream cipher is evaluated using NIST statistical tests. We iterate the 3 statistical trials in [12] to evaluate a pass of the produced keystreams to demonstrate the importance of the proposed design.

We used the same sample size sequences and the same weighted importance level applied in the NIST trial [21] to compare the results with previous work [12]. As a result, the significance level in all of the samples was fixed at 300 sequences, and the size of the experiment was reduced to 0.01. The following are the definitions of the tests:

a. Long keystream set of data test: To obtain 300 sequences of 1048576 bits, generate bits for the entire 300 random key (put the initial vector to zero in all cases). The result sequences are then evaluated using the 15 NIST tests.
b. Short keystream set of data test: Generate bits for each of the 307200 random keys (in all cases, set the initial vector to zero). The concatenation of all the bit keystreams yields 300 sequences with a total length of 1048576 bits. The result sequences are then evaluated using the 15 NIST tests.
c. The initial vector set of data starts by generating 256 (4096 bits) sequences for 300 random keys. It's worth noting that each key's first sequence is generated with IV put to zero. It is then increased to the next 255 sequences. As a result, we have 300 sequences totaling 1048576 bits for NIST's 15 tests to evaluate.

There are two accesses to interpreting the NIST test score, the ratio of sequences that pass in to a statistical test and allocation of P-value to examination for uniformity [4]. The domain of acceptable ratio is determined using the trust interval, defined as:

$$\hat{P} \mp 3\sqrt{\frac{\hat{P}(1-\hat{P})}{S}} \tag{31}$$

Let $\alpha = 0.01$ $\hat{P} = 1-0.01$ $S = 300$

If the ratio does not fall within this range, the data is not random.

The uniformity of P-value is determined using the X^2 goodness of fit test, which involves computing:

$$X_{test}^2 = \sum_{i=1}^{t} \frac{O_i - E_i}{E_i} \tag{32}$$

The number of levels is represented by l, the key stream was tested to see if it was distributed uniformly. With a total of 300 keystreams, a minimum threshold ratio value of 0.97 is acceptable.

When we implemented the SNOW3G-M algorithm, a keystream is random, as per Table 3, because P-values are distributed uniformly and the P-values ratio is accepted as shown in Table 4. Finally, the proposed SNOW3G-M algorithm was compared with the SNOW3G traditional algorithm in Table 5.

Table 3. P-values results by NIST-tests

No. Test	NIST Test	SNOW3G-M Long-Keystream	SNOW3G-M Short-Keystream	SNOW3G-M IV-Keystream
1	Monobit-test(frequency)	0.501535	0.501496	0.501488
2	Frequency-within-block-test	0.500128	0.500138	0.500160
3	Runs-test	0.985206	0.498141	0.498116
4	Longest-run-ones-in-block-test	0.487928	0.487928	0.487864
5	Binary-matrix-rank-test	0.398332	0.524316	0.412435
6	Discrete fourier transform-test	0.477582	0.477565	0.477557
7	Non-overlapping-template-matching-test	0.446077	0.445748	0.446222
8	Overlapping-template-matching-test	0.079792	0.628880	0.453612
9	Maurer-universal-test	0.994265	0.992731	0.994359
10	Linear-complexity-test	0.255588	0.311035	0.382952
11	Serial-test	0.500788	0.500777	0.500698
12	Approximate-entropy-test	0.495596	0.495587	0.495507
13	Cumulative-sums-test	0.520659	0.520631	0.520657
14	Random-excursion-test	0.458387	0.570036	0.317417
15	Random-excursion-variant-test	0.322805	0.879336	0.582429

Table 4. P-values ratio is acceptable by NIST-tests

No. Test	NIST Test	SNOW3G-M Long-Keystream	SNOW3G-M Short-Keystream	SNOW3G-M IV-Keystream
1	Monobit-test(frequency)	0.985392	0.985392	0.985392
2	Frequency-within-block-test	0.986974	0.986974	0.986974
3	Runs-test	0.985206	0.985206	0.985206
4	Longest-run-ones-in-block-test	0.986509	0.986509	0.986509
5	Binary-matrix-rank-test	0.978444	0.975428	0.980444
6	Discrete fourier transform-test	0.982043	0.982043	0.982043
7	Non-overlapping-template-matching-test	0.970986	0.973312	0.975917
8	Overlapping-template-matching-test	1.0	1.0	1.0
9	Maurer-universal-test	1.0	1.0	1.0
10	Linear-complexity-test	1.0	1.0	1.0
11	Serial-test	0.978321	0.978321	0.978321
12	Approximate-entropy-test	0.985485	0.985485	0.985485
13	Cumulative-sums-test	0.983531	0.983531	0.983531
14	Random-excursion-test	1.0	0.0	1.0
15	Random-excursion-variant-test	1.0	1.0	1.0

Table 5. Compared the SNOW3G traditional [14] with the SNOW3G-M algorithm using NIST-tests

No. Test	NIST Test	SNOW3G Long Traditional	SNOW3G Short Traditional	SNOW3G IV Traditional	Proposed SNOW3G-M Long	Proposed SNOW3G -M Short	Proposed SNOW3G-M IV
1	Monobit-test(frequency)	pass	pass	pass	pass	pass	pass
2	Frequency-within-block-test	pass	pass	pass	pass	pass	pass
3	Runs-test	pass	fail	pass	pass	pass	pass
4	Longest-run-ones-in-block-test	pass	fail	pass	pass	pass	pass
5	Binary-matrix-rank-test	pass	fail	pass	pass	pass	pass
6	Discrete fourier transform-test	pass	fail	pass	pass	pass	pass
7	Non-overlapping-template-matching-test	pass	fail	pass	pass	pass	pass
8	Overlapping-template-matching-test	pass	fail	pass	pass	pass	pass
9	Maurer-universal-test	pass	pass	pass	pass	pass	pass
10	Linear-complexity-test	pass	pass	pass	pass	pass	pass
11	Serial-test	pass	fail	pass	pass	pass	pass
12	Approximate-entropy-test	pass	fail	pass	pass	pass	pass
13	Cumulative-sums-test	pass	pass	pass	pass	pass	pass
14	Random-excursion-test	pass	pass	pass	pass	pass	pass
15	Random-excursion-variant-test	pass	pass	pass	pass	pass	pass

5 SNOW3G-M algorithm and medical uses

The proposed snow3G-M algorithm has several benefits, especially in medical uses, because of its speed in encrypting data, and this thus leads to a speedy transmission of data between the sender and recipient over the network, where the doctor monitors the patient's condition and gives him instructions via phone call, in addition to using several applications to register to obtain vaccines against COVID-19 around the world [27–31].

6 Discussion

The proposed SNOW3G-M algorithm was designed using 2-LFSR (Two Linear Feedback Shift Register) to eliminate the fault attack vulnerability represented in the initialization process and to prevent the attacker from exploiting the previous weakness in LFSR, where it was possible to recover the secret key by injecting 22 injections into the LFSR. The new design prevents the attacker from modifying 64-bit during the key streaming generation process. Where the difficulty lies in controlling the LFSR as it is not possible to control t, represented by the time clock, and not control i, represented by the stage number of the linear. The proposed algorithm also used 4 S-Boxes created with the permutation last-layer algorithm to prevent an attacker from exploiting the cipher by using the time of the exit from S-box 1 as an entrance to another S-box. When two boxes are worked on at the same time to prevent a cache-timing-attack. The process of feedback from FSM to LFSR during the initialization mode is 64 bits, as 32 bits enter the LFSR1-R and the other 32 bits enter the LFSR2-H, which confuses the attacker in trying a multiset collision attack where all the LFSR and FSM are in a state of constant change.

7 Conclusion

The proposed SNOW3G-M algorithm was designed based on the basic structure of the traditional SNOW3G algorithm with an update in its work while maintaining the basic requirements of the hardware.as well as providing a perfect balance of high security, high performance, and limited hardware resources.

The main purpose of using 2 LFSR and 4 S-boxes is to generate a keystream that is more powerful, more complex, and more random to increase the complexity of the cipher. In addition to eliminating the weakness of the setup process that led to the failure of NIST tests on the short data set. Both the initialization and generation stages in the proposed algorithm result in 64 bits (2 words). Both the initialization and generation stages in the proposed algorithm result in 64 bits, which speeds up the encryption process and reduces the attack.

Based on our findings, we conclude that the SNOW 3G-M keystream passes all of the tests for long key stream data, short key stream data, and initialization vector data sets. SNOW3G-M stream cipher architecture was proposed in this paper that was used as the foundation for the UMTS and LTE standards' confidentiality and integrity algorithms.as well as in the fifth generation.

8 References

[1] R. M. Zaki, T. W. Khairi, and A. E. Ali, "Secure data sharing based on linear congruetial method in cloud computing," in *Next Generation of Internet of Things*: Springer, 2021, pp. 129–140. https://doi.org/10.1007/978-981-16-0666-3_13

[2] T. W. Khairi, R. M. Zaki, and W. A. Mahmood, "Stock price prediction using technical, fundamental and news based approach," in *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, 2019: IEEE, pp. 177–181. https://doi.org/10.1109/SCCS.2019.8852599

[3] R. M. Zaki and H. B. A. Wahab, "4G network security algorithms: Overview," *International Journal of Interactive Mobile Technologies,* vol. 15, no. 16, 2021. https://doi.org/10.3991/ijim.v15i16.24175

[4] K. E. Mayes and K. Markantonakis, "Mobile communication security controllers an evaluation paper," *Information Security Technical Report,* vol. 13, no. 3, pp. 173–192, 2008. https://doi.org/10.1016/j.istr.2008.09.004

[5] A. Jalal and I. Uddin, "Security architecture for third generation (3G) using GMHS cellular network," in *2007 International Conference on Emerging Technologies*, 2007: IEEE, pp. 74–79. https://doi.org/10.1109/ICET.2007.4516319

[6] M. Madani, I. Benkhaddra, C. Tanougast, S. Chitroub, and L. Sieler, "Digital implementation of an improved LTE stream cipher snow-3G based on hyperchaotic PRNG," *Security and Communication Networks,* vol. 2017, 2017. https://doi.org/10.1109/CoDIT.2017.8102758

[7] M. Madani and C. Tanougast, "Combined and robust SNOW-ZUC algorithm based on chaotic system," in *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2018: IEEE, pp. 1–7. https://doi.org/10.1109/CyberSecPODS.2018.8560677

[8] G. Orhanou, S. E. Hajji, Y. Bentaleb, and J. Laassiri, "EPS confidentiality and integrity mechanisms algorithmic approach," *arXiv preprint arXiv:1102.5191,* 2011.

[9] G. M. Koien, "An introduction to access security in UMTS," *IEEE Wireless Communications,* vol. 11, no. 1, pp. 8–18, 2004. https://doi.org/10.1109/MWC.2004.1269712

[10] M. Madani, S. Chitroub, and C. Tanougast, "Two KASUMI components for an optimal implementation of the A5/3 algorithm," in *2017 International Conference on Circuits, System and Simulation (ICCSS)*, 2017: IEEE, pp. 124–128. https://doi.org/10.1109/CIRSYSSIM.2017.8023195

[11] P. Bohm, "Statistical evaluation of stream cipher snow 3G," in Constantin Brancusi University of Targu Jiu Engineering Faculty Scientific Conference with international participation, 2008, pp. 363–366.

[12] A. G. Sulaiman and I. F. Al Shaikhli, "Comparative study on 4G/LTE cryptographic algorithms based on different factors," *International Journal of Computer Science and Telecommunications,* vol. 5, no. 7, pp. 7–10, 2014.

[13] B. B. Brumley, R. M. Hakala, K. Nyberg, and S. Sovio, "Consecutive S-box lookups: A timing attack on SNOW 3G," in *International Conference on Information and Communications Security*, 2010: Springer, pp. 171–185. https://doi.org/10.1007/978-3-642-17650-0_13

[14] A. Kircanski and A. M. Youssef, "On the sliding property of SNOW 3 G and SNOW 2.0," *IET Information Security*, vol. 5, no. 4, pp. 199–206, 2011. https://doi.org/10.1049/iet-ifs.2011.0033

[15] H. T. AlRikabi, A. H. M. Alaidi, A. S. Abdalrada, and F. T. Abed, "Analysis of the efficient energy prediction for 5G wireless communication technologies," *International Journal of Emerging Technologies in Learning*, vol. 14, no. 8, 2019. https://doi.org/10.3991/ijet.v14i08.10485

[16] H. A. Abdulmohsin, H. B. A. Wahab, and A. M. J. A. Hossen, "A new hybrid feature selection method using T-test and fitness function," *CMC-Computers Materials & Continua,* vol. 68, no. 3, pp. 3997–4016, 2021. https://doi.org/10.32604/cmc.2021.014840

[17] M. Al-dabag, H. S. ALRikabi, and R. Al-Nima, "Anticipating atrial fibrillation signal using efficient algorithm," 2021. https://doi.org/10.3991/ijoe.v17i02.19183

[18] H. A. Abdulmohsin, "A new proposed statistical feature extraction method in speech emotion recognition," *Computers & Electrical Engineering*, vol. 93, p. 107172, 2021. https://doi.org/10.1016/j.compeleceng.2021.107172

[19] H. A. Abdulmohsin, H. B. A. Wahab, and A. M. J. A. Hossen, "A novel classification method with cubic spline interpolation," *Intelligent Automation and Soft Computing,* vol. 31, no. 1, pp. 339–355, 2022. https://doi.org/10.32604/iasc.2022.018045

[20] D.-S. Kundi, A. Aziz, and N. Ikram, "A high performance ST-Box based unified AES encryption/decryption architecture on FPGA," *Microprocessors and Microsystems,* vol. 41, pp. 37–46, 2016. https://doi.org/10.1016/j.micpro.2015.11.015

[21] E. B. Smid, S. Leigh, M. Levenson, M. Vangel, A. DavidBanks, and S. JamesDray, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *Her Research Interest Includes Computer Security, Secure Operating Systems, Access Control, Distributed Systems, Intrusion Detection Systems*, 2010.

[22] H. B. A. Wahab and T. A. Jaber, "Using Chebyshev Polynomial and Quadratic Bezier Curve for Secure Information Exchange," *Engineering and Technology Journal*, vol. 34, no. 5 Part (B) Scientific, 2016.

[23] H. B. A. Wahab, S. M. Kadhem, and E. A. R. Kadhim, "Proposed approach for key generation based on elliptic curve (EC) algebra and metaheuristic algorithms," *Engineering and Technology Journal,* vol. 32, no. 2 Part (B) Scientific, 2014.

[24] H. Bahjat and M. Ali, "Improvement majority function in A5/1 stream cipher algorithm," *Engineering and Technology Journal,* vol. 34, no. 1 Part (B) Scientific, 2016.

[25] G. Marsaglia and W. W. Tsang, "The 64-bit universal RNG," *Statistics & Probability Letters,* vol. 66, no. 2, pp. 183–187, 2004. https://doi.org/10.1016/j.spl.2003.11.001

[26] B. Preneel, "New European Schemes for Signature, Integrity and Encryption (NESSIE): a status report," in *International Workshop on Public Key Cryptography*, 2002: Springer, pp. 297–309. https://doi.org/10.1007/3-540-45664-3_21

[27] P. Beňo, F. Schauer, S. Šprinková, and T. Komenda, "Optimization of the cloud for setup of hardware of remote laboratories," 2020. https://doi.org/10.3991/ijoe.v16i12.16699

[28] H. Naman, N. Hussien, M. Al-dabag, and H. Alrikabi, "Encryption system for hiding information based on internet of things," 2021. https://doi.org/10.3991/ijim.v15i02.19869

[29] H. Muljo, A. Perbangsa, and B. Pardamean, "Assessment of online learning application for health education," 2019. https://doi.org/10.3991/ijoe.v15i12.11157

[30] P. Podder, A. Khamparia, M. R. H. Mondal, M. A. Rahman, and S. Bharati, "Forecasting the spread of COVID-19 and ICU requirements," 2021. https://doi.org/10.20944/preprints202103.0447.v1

[31] K. Munawar and F. R. Choudhry, "Exploring stress coping strategies of frontline emergency health workers dealing Covid-19 in Pakistan: A qualitative inquiry," *American Journal of Infection Control,* vol. 49, no. 3, pp. 286–292, 2021. https://doi.org/10.1016/j.ajic.2020.06.214

9 Authors

Rana M. Zaki received the BSc and MSc degrees in Computer Sciences in 2003 and 2016, respectively from the University of Technology. In 2007 she worked at the Department of Computer Sciences/University of Technology, Baghdad, Iraq, and completed her as an assistant lecturer in 2016. She published five articles in image processing, multimedia, and data encryption. Her research interests are Mobile networks, Encryption algorithms, and cloud computing. (Department Computer Sciences, University of Technology, Baghdad, Iraq) (email: Rana.M.Zaki@uotechnology.edu.iq).

Hala Bahjat Abdul Wahab, She author became a Reviewer (R) of IEEE in 2010. Was born in Basra, Iraq in 1969. She received the B.S. degree in 1990 in computer science, Basra University, M.S. degree in 2001 in computer science, Technology University, and finally the Ph.D. degree in 2006 in computer science security from the department of computer science, University of Technology, Baghdad, Iraq. She received a professor degree in 2018 from the technical university. From 1991 to 1995, Dr. Hala was a lecturer assistant in the computer science department at the Basra University, Iraq. From 1995 to 2020, Dr. Hala was a lecturer in the computer science department at the Technology University, Iraq. She received the Assistant professor's degree in 2006 and the professor's degree in 2018 from the Technology University. Prof. Hala is the author of more than 65 articles. Prof. Hala research interests include Information and network Security. Prof. Hala is a co-author in the "PGP Protocols and its Applications" book in IN TECH 2012. (Department Computer Sciences, University of Technology, Baghdad, Iraq) (email: Hala.B.AbdulWahab@uotechnology.edu.iq).

Article submitted 2021-11-04. Resubmitted 2021-12-19. Final acceptance 2021-12-19. Final version published as submitted by the authors.