Hindawi
Security and Communication Networks
Volume 2017, Article ID 5746976, 15 pages
https://doi.org/10.1155/2017/5746976

Research Article

Digital Implementation of an Improved LTE Stream Cipher Snow-3G Based on Hyperchaotic PRNG

Mahdi Madani,[1,2] Ilyas Benkhaddra,[2] Camel Tanougast,[2] Salim Chitroub,[1] and Loic Sieler[2]

[1] *LISIC Laboratory, Electronics and Computer Science Faculty, University of Science and Technology of Houari Boumediene (USTHB), Algiers, Algeria*
[2] *LCOMS, Universite de Lorraine, Metz, France*

Correspondence should be addressed to Mahdi Madani; mmadani49@gmail.com

Received 23 June 2017; Revised 10 September 2017; Accepted 27 September 2017; Published 27 November 2017

Academic Editor: Angel M. Del Rey

SNOW-3G is a stream cipher used by the 3GPP standards as the core part of the confidentiality and integrity algorithms for UMTS and LTE networks. This paper proposes an enhancement of the regular SNOW-3G ciphering algorithm based on HC-PRNG. The proposed cipher scheme is based on hyperchaotic generator which is used as an additional layer to the SNOW-3G architecture to improve the randomness of its output keystream. The objective of this work is to achieve a high security strength of the regular SNOW-3G algorithm while maintaining its standardized properties. The originality of this new scheme is that it provides a good trade-off between good randomness properties, performance, and hardware resources. Numerical simulations, hardware digital implementation, and experimental results using Xilinx FPGA Virtex technology have demonstrated the feasibility and the efficiency of our secure solution while promising technique can be applied to secure the new generation mobile standards. Thorough analysis of statistical randomness is carried out demonstrating the improved statistical randomness properties of the new scheme compared to the standard SNOW-3G, while preserving its resistance against cryptanalytic attacks.

1. Introduction

Nowadays, the security of communications becomes more and more important to meet the demand for real-time secure data transmission over the open networks [1]. Indeed, in our modern life, all our daily transactions and communications are more made over mobile networks. The Universal Mobile Telecommunications System (UMTS) [2], Long Term Evolution (LTE), and LTE-advanced standards are the 3rd- and 4th-generation mobile networks that enable mobile migrations of Internet applications like Voice over IP (VoIP), video streaming, music downloading, mobile TV, and so on. The security of these mobile standards is based principally on the SNOW-3G stream cipher [3] which is considered as the kernel of the 128-EEA1 confidentiality and 128-EIA1 integrity algorithms of the 4G LTE security [4] replacing the f8 and f9 algorithms of UMTS security [5, 6]. Thereafter, the Third-Generation Partnership Project (3GPP) has already standardized two other kernel algorithms for LTE confidentiality and integrity protection [7]. These algorithms are the Advanced Encryption Standard (AES) in its CounTeR mode (CTR mode) [8] which is composed of 10 rounds processing essentially on the Substitution-Diffusion Network (SDN) principle [9] and the ZUC stream cipher specifically designed for use in China [10, 11].

However, AES algorithm has a long computational time [12]; ZUC stream cipher is recently designed and requires more security analysis to prove its robustness and efficiency [13]. However, several analyses have proved that SNOW-3G encryption algorithm is weak. In particular, improvements to guard against the following:

(i) The short keystream data set attack: this attack discovers a weakness in the initialization process of SNOW-3G stream cipher when 8 from the 15 NIST (National Institute of Standards and Technology) tests fail the short keystream data set test [14].

(ii) The improved Heuristic Guess and Determine (IHGD) attack which reduces the complexity from $O(2^{320})$ to $O(2^{160})$ and the size of guessed basis from

10 to 5: this attack exploits the weakness of main linear feedback polynomials of the cipher [15].

(iii) The fault attack which recovers the secret key with only 22 fault injections on the Linear Feedback Shift Register (LFSR): the attack model assumes that the attacker is able to modify a 32-bit value of one state of the LFSR (S_i^t) during the keystream generation, where i is chosen by the attacker, but he has no full control about t [16].

(iv) The cache-timing attack which is capable of recovering the full cipher state from empirical timing data in a matter of seconds: the attack exploits the cipher using the output from a S-box as input to another S-box [17].

(v) The multiset collision attack on reduced-round: this attack analyzes the resynchronization mechanism of SNOW-3G using multiset collision attacks. This technique has proved itself useful against AES [18]. This technique can be applied to SNOW-3G since its Finite State Machine (FSM) is essentially a 96-bit AES-like cipher in which the LFSR plays a role of a key-schedule. This attack is complicated by the fact that there is a feedback from the FSM to the LFSR during the setup phase (a feature never present in block ciphers) and the attacker sees only 32 bits of output at a time, while the internal state keeps changing constantly [19].

(vi) The sliding property attack of stream ciphers: this attack is used to find sets of related keys. This attack is applicable due to the nature of the initialization process of SNOW-3G; such related keys do not generate slid keystreams, but only keystreams that have several equal words. However, this still allows distinguishing the produced keystream from random keystreams [20].

Consequently, we remark that the majority of the cited attacks exploit mainly the weakness of the initialization process based on the LFSR polynomials and the FSM S-box. Therefore, we can not improve all the cited weaknesses, but we focus in this paper on the improvement of the statistical properties analyzed in [14], and we propose updating the SNOW-3G stream cipher in order to enhance its robustness and being able to resist against new cryptanalysis attacks based on the considered weaknesses.

During the last decade, the use of chaos to design digital chaotic stream ciphers has become a part of cryptography and very important topic of research [21]. With regard to the characteristics of a chaotic dynamical system these are as follows: a deterministic system, ergodic, sensitive to initial conditions and control parameters, and having a random-like behaviour. In addition, these characteristics are related to confusion and diffusion concepts usually used in traditional cryptosystems which allow the application of chaos in cryptography. Considering the possibility for the self-synchronization of chaotic systems, this concept has become a focal topic for research [22]. Therefore, several schemes have been developed to exploit this property of chaotic systems for secure communications. Encryption based chaos has been suggested as a new and efficient way to deal with the problem of fast and highly secure mobile communications. Similarly, several works are proposed to exploit the hyperchaotic (HC) systems providing the nonlinearity and the good randomness. Among them, we can cite the *Rössler*'s hyperchaotic systems [23, 24], *Chen*'s hyperchaotic systems [25, 26], *Lu*'s hyperchaotic system [27], and *Lorenz*'s hyperchaotic systems [28–30].

This paper proposes a new architecture which combines the regular SNOW-3G cipher with a Hyperchaotic Pseudo-Random Number Generator (HC-PRNG) in order to design a hyperchaotic SNOW-3G stream cipher providing good randomness, infinite period, and unpredictability on long term while increasing the complexity of a cryptanalysis attack from the weakness in the initialization mode of the regular SNOW-3G cipher [14]. The considered HC-PRNG is based on a four-dimensional (4D) Lorenz's chaotic system which is used as an additional layer to the regular SNOW-3G while maintaining its standard properties. Contrariwise of the previous simulated system adding one simple linear chaotic map with the FSM layer of the regular SNOW-3G algorithm [31], we propose an enhancement of initialization process and complexity and randomness of output keystream. The main purpose of this added layer is the perturbation of both the digital LFSR sequences and the FSM output. Thus, we suggest this hyperchaotic perturbation technique to investigate the encryption efficiency of LFSR based stream cipher. Consequently, the proposed architecture has the following features. First, the structure of the stream generator key is more complex having a number of variables and parameters, and the time sequence of output is nonrule and not predictable. Thereby, a large key space is provided for use as seed keys of cipher sequences. Second, the standardized core of SNOW-3G stream cipher is respected in order to facilitate the integration of our approach in the current and future mobile communication networks. Moreover, the considered system is irreversible and unpredictable and it may be extended to many other (4D) continuous chaotic systems while being suitable for digital implementation. Considering the numerical resolution of nonlinear differential equations to get a digital implementation, our proposed approach has been designed and verified using a VHDL (VHSIC Hardware Description Language) description with a fixed-point representation of all data on 48 bits leading to a successful hardware implementation targeted to FPGA technology (Field-Programmable Gate Array). The obtained implementation results provide good performance in terms of hardware area and throughput. Finally, the sturdiness of our proposed architecture has been tested using security analyses and statistical tests. Consequently, the experimental results presented in this paper prove that the added HC generator enhances the randomness and robustness of SNOW-3G algorithm against different analyses and statistical attacks.

The rest of this paper is organized as follows. Section 2 briefly reminds the reader with the architecture of the regular SNOW-3G stream cipher. Section 3 details the processing steps and architecture of the enhanced SNOW-3G stream cipher including a Lorenz's HC generator as PRNG to perturb

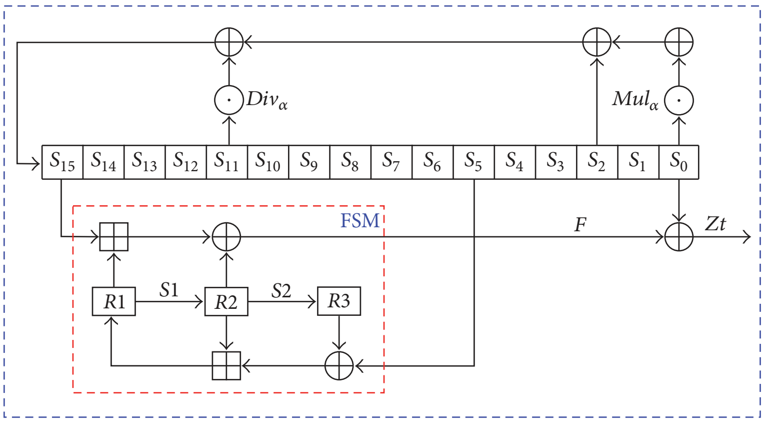

Div_α
Mul_α
S_{15} S_{14} S_{13} S_{12} S_{11} S_{10} S_9 S_8 S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0
FSM
F
Zt
R1
S1
R2
S2
R3

Figure 1: Architecture of the regular SNOW-3G stream cipher.

the digital output of the regular SNOW-3G stream cipher. The digital implementation results on Virtex-5 Xilinx FPGA technology and evaluations are presented in Section 4. In this section, the performance and architectural simulations prove the feasibility and the efficiency of our proposed HC SNOW-3G stream cipher. In Section 5, the robustness of our new scheme is evaluated through three security analyses: exhaustive key search, key sensitivity, and statistical tests to prove that the proposed HC SNOW-3G stream cipher resists against cryptanalysis attacks and exhibits truly random sequences suitable as cipher keys for the data encryption compared to the regular SNOW-3G cipher. Finally, Section 6 concludes this paper.

2. Regular SNOW-3G Algorithm

The SNOW-3G is a cryptographic algorithm which was originally designed as the kernel of the UMTS confidentiality and integrity functions f8 and f9 [5], respectively. After that, it has been kept as the first kernel algorithm for the LTE security functions 128-EEA1 and 128-EIA1, respectively [3, 4]. SNOW-3G is a stream cipher that generates 32-bit word keystream under the control of the 128-bit Ciphering Key (CK) and 128-bit Initialization Vector (IV). It is divided into two interacting layers. The first layer is the LFSR layer which is formed by 16 stages of 32 bits each $(S_0, S_1, \ldots, S_{15})$. The second one is the FSM layer which is formed by three registers ($R1$, $R2$, and $R3$) of 32 bits each. The detailed architecture of the SNOW-3G stream cipher is illustrated in Figure 1, where $\oplus$ represents bitwise XOR operation and $\boxplus$ is adder arithmetic operator. Considering that the regular SNOW-3G can not offer robustness against short keystream data set attacks [14], we propose enhancing its statistical properties and security level by overcoming its weaknesses through an additional layer updating the initialization and keystream modes while keeping its standard.

3. Enhanced SNOW-3G Algorithm Based on HC-PRNG

In this section, we present our designed scheme to enforce the confidentiality and integrity protection for LTE and new generations of mobile standards. We propose adding a HC-PRNG as an additional layer to the regular SNOW-3G architecture which is based on the LFSR and FSM layers. The HC-PRNG is added to the regular SNOW-3G architecture to consolidate the complexity of the regular SNOW-3G stream cipher in order to improve the randomness of the output keystream and to ameliorate the LFSR initialization based on the perturbation effect. More precisely, the HC-PRNG output is mixed with the feedback value of the LFSR layer during the initialization process (see Figure 2), or mixed with the FSM output F and with the register S_0 during the keystream generation process (more details in Section 3.4). Finally, a specific HC SNOW-3G stream cipher is obtained. Figures 2 and 6 depict the considered enhanced architecture. The layers forming the HC SNOW-3G architecture are detailed in the following subsections.

3.1. LFSR Layer. As with the regular SNOW-3G, this layer is formed by 16 stages of 32 bits each $(S_0, S_1, \ldots, S_{15})$ (see Figures 1 and 2). The associated feedback is defined by a primitive polynomial over the finite Galois field GF(2^{32}) using bitwise XOR operations and the well-known Mul_α and Div_α main functions [3]. The LFSR component is clocked in two operating modes described as follows.

Initialization Mode. In this mode, the LFSR is performed by using three stage registers S_0, S_2, and S_{11} and one 32-bit word called F which is the output value of the FSM layer. First, the two 32-bit stage registers S_0 and S_{11} are divided into four bytes as follows:

$$\begin{aligned} S_0 &= S_{0,0} \parallel S_{0,1} \parallel S_{0,2} \parallel S_{0,3}, \\ S_{11} &= S_{11,0} \parallel S_{11,1} \parallel S_{11,2} \parallel S_{11,3}, \end{aligned} \tag{1}$$

where $\parallel$ is the concatenation operation. In this division step, the feedback value v is calculated by

$$\begin{aligned} v = {} & (S_{0,1} \parallel S_{0,2} \parallel S_{0,3} \parallel 0x00) \oplus Mul_\alpha (S_{0,0}) \oplus S_2 \\ & \oplus (0x00 \parallel S_{11,0} \parallel S_{11,1} \parallel S_{11,2}) \oplus Div_\alpha (S_{11,3}) \oplus F. \end{aligned} \tag{2}$$

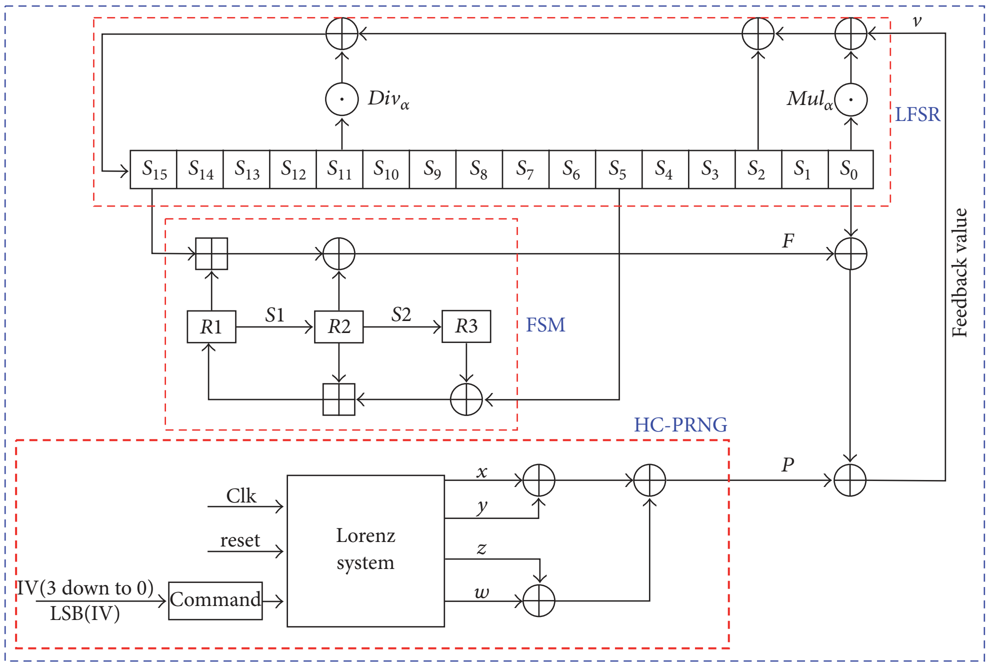


Figure 2: The architecture of our proposed hyperchaotic SNOW-3G stream cipher in its initialization mode.

This feedback value sets the S_{15} stage register ($S_{15} = v$) and the contents of the other stage registers are shifted as follows:

$$S_{14} = S_{15}, S_{13} = S_{14}, \dots, S_1 = S_2, S_0 = S_1. \tag{3}$$

Keystream Mode. In this mode, the main difference with the initialization mode is that the feedback value v is calculated without using the 32-bit word F and by

$$\begin{aligned} v = {} & (S_{0,1} \parallel S_{0,2} \parallel S_{0,3} \parallel 0x00) \oplus Mul_\alpha (S_{0,0}) \oplus S_2 \\ & \oplus (0x00 \parallel S_{11,0} \parallel S_{11,1} \parallel S_{11,2}) \oplus Div_\alpha (S_{11,3}). \end{aligned} \tag{4}$$

All the other steps are similar to the initialization mode.

3.2. FSM Layer. As with the regular SNOW-3G, the FSM layer is formed by three registers $R1$, $R2$, and $R3$ of 32 bits each (see Figures 1 and 2). To generate a 32-bit output keystream word named F, this component uses two substitution boxes $S1$ and $S2$, one linear bitwise XOR operation, and one nonlinear integer modulo 2^{32} adder [3]. The FSM is clocked by using two inputs provided by the two registers S_5 and S_{15} arising from the previous LFSR component (see Figure 1). Therefore, F can be expressed by

$$F = (S_{15} \boxplus R1) \oplus R2. \tag{5}$$

Thereby, an intermediate value r is calculated to set the three registers $R1$, $R2$, and $R3$ at each clock cycle as defined by

$$\begin{aligned} r &= R2 \boxplus (R3 \oplus S_5), \\ R3 &= S_2 (R2), \\ R2 &= S_1 (R1), \\ R1 &= r. \end{aligned} \tag{6}$$

3.3. HC-PRNG Layer. The added HC-PRNG layer is performed by a nonlinear dynamic generator which is very sensitive to initial conditions and presents high recurrences (see Figure 2). This system can be used in cryptography to replace the PRNG. A cryptosystem based on this HC-PRNG consists of using a chaotic nonlinear oscillator as a pseudo-random signal generator which is combined with the message before transmission. At the reception, the similar chaotic oscillator is used in order to regenerate the pseudo-random signal which is combined with the received signal through the inverse operation and in order to recover the original message [32]. Moreover, the added HC-PRNG layer must be synchronized between the emitter and the receiver to ensure encryption/decryption process like any chaotic or HC system. To realize our HC-PRNG, we have implemented the Lorenz HC system which is generated by deriving the generalized Lorenz quadratic controller system (4D) by providing a hyperchaotic behaviour. It is a four-dimensional dynamical controlled system which is characterized by the following [33]:

$$\begin{aligned} \dot{x} &= a(y - x), \\ \dot{y} &= cx - xz - y + w, \\ \dot{z} &= xy - bz, \\ \dot{w} &= -dx, \end{aligned} \tag{7}$$

where a, b, c, and d are the control parameters. Note that a, b, c, and d are >0, and usually $a = 10$, $b = 8/3$, $d = 5$, and c is varied. The system exhibits one hyperchaotic behaviour for $c = 28$. The initial conditions are set as $x_0 = y_0 = z_0 = w_0 = -10$. The two principal requisites presented below must be considered to obtain hyperchaos behaviour:

(1) The minimal dimension of the phase space that embeds a HC attractor should be at least four, which requires the minimum number of coupled first-order autonomous ordinary differential equations to be four.

(2) The number of terms in the coupled equations giving rise to instability should be at least two, of which at least one should have a nonlinear function [34].

For resolving (7), we use the Runge-Kutta fourth-order resolution method (RK-4 method) [35, 36]. We present a brief review of this method.

For example, a nonlinear system is defined by

$$\begin{aligned}\dot{x} &= F(x, y, z, w),\\ \dot{y} &= G(x, y, z, w),\\ \dot{z} &= Q(x, y, z, w),\\ \dot{w} &= U(x, y, z, w).\end{aligned} \tag{8}$$

Note that $x(t0) = x_0$, $y(t0) = y_0$, $z(t0) = z_0$, and $w(t0) = w_0$, where F, G, Q, and U are nonlinear functions. At each iteration $n + 1$, RK-4 uses four intermediate values from the iteration n. Then, (9) is used to solve (8):

$$\begin{aligned}x_{n+1} &= x_n + \frac{h}{6}(k_0 + 2k_1 + 2k_2 + k_3),\\ y_{n+1} &= y_n + \frac{h}{6}(m_0 + 2m_1 + 2m_2 + m_3),\\ z_{n+1} &= z_n + \frac{h}{6}(n_0 + 2n_1 + 2n_2 + n_3),\\ w_{n+1} &= w_n + \frac{h}{6}(p_0 + 2p_1 + 2p_2 + p_3).\end{aligned} \tag{9}$$

Note that h is the integration step (interval). For each integration step h, four increments are defined for each function (x, y, z, and w): first increment at the beginning, then two increments at the midpoint, and a last increment at the end of the interval h, as follows.

When $t = t_0$, the first increment values are defined based on the slope at the beginning of the interval, as follows:

$$\begin{aligned}k_0 &= F(t_n, x_n),\\ m_0 &= G(t_n, y_n),\\ n_0 &= Q(t_n, z_n),\\ p_0 &= U(t_n, w_n)\end{aligned} \tag{10}$$

and when $t = t + h/2$, the second increment values are defined based on the slope at the midpoint of the interval, using previous values (k_0, m_0, n_0, and p_0), as follows:

$$\begin{aligned}k_1 &= F\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_0\right),\\ m_1 &= G\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}m_0\right),\\ n_1 &= Q\left(t_n + \frac{h}{2}, z_n + \frac{h}{2}n_0\right),\\ p_1 &= U\left(t_n + \frac{h}{2}, w_n + \frac{h}{2}p_0\right)\end{aligned} \tag{11}$$

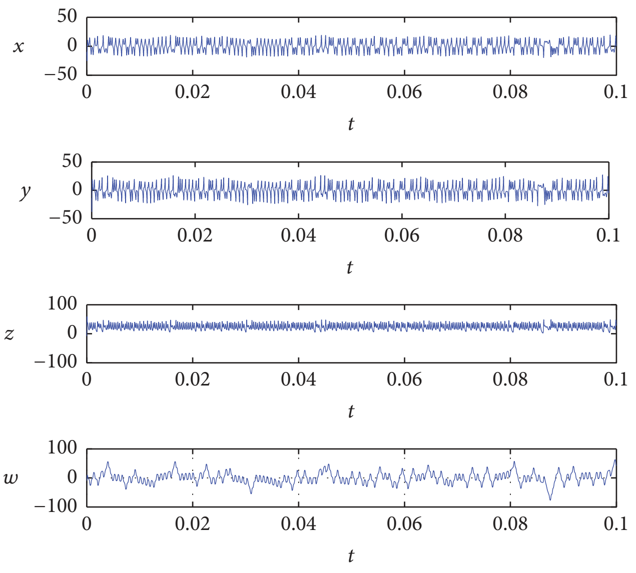
and when $t = t + h/2$, the third increment values are defined based on the slope at the midpoint of the interval, using previous values (k_1, m_1, n_1, and p_1), as follows:

$$\begin{aligned}k_2 &= F\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right),\\ m_2 &= G\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}m_1\right),\\ n_2 &= Q\left(t_n + \frac{h}{2}, z_n + \frac{h}{2}n_1\right),\\ p_2 &= U\left(t_n + \frac{h}{2}, w_n + \frac{h}{2}p_1\right)\end{aligned} \tag{12}$$

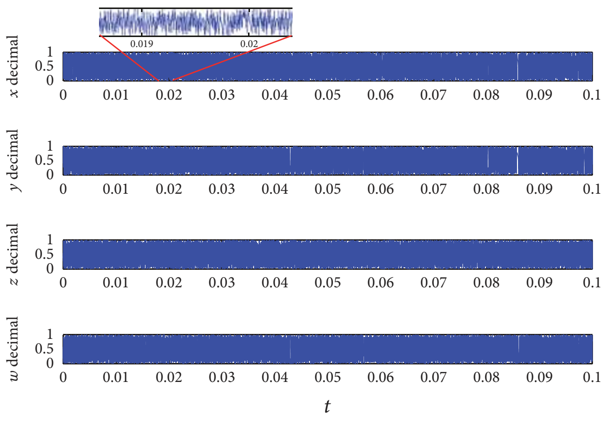
and when $t = t + h$, the fourth increment values are defined based on the slope at the end of the interval, using previous values (k_2, m_2, n_2, and p_2), as follows:

$$\begin{aligned}k_3 &= F(t_n + h, x_n + hk_2),\\ m_3 &= G(t_n + h, y_n + hm_2),\\ n_3 &= Q(t_n + h, z_n + hn_2),\\ p_3 &= U(t_n + h, w_n + hp_2).\end{aligned} \tag{13}$$

Mentioning that HC signals are real, we used the fixed-point format of 48 bits as 16Q32. The real part is encoded on 16 bits and the decimal part is encoded on 32 bits. The used arithmetic fixed-point allows achieving a good compromise between performance and cost. Equation set (8) is simulated in MATLAB platform using the numerical tool RK-4, and the results are illustrated in Figures 3 and 4. A good randomness is obtained in random HC signals (decimal part) as shown in Figure 3(b) compared to the HC signals (real and decimal parts) as shown in Figure 3(a). Similarly, the Lorenz attractor formed by the random HC signals (decimal part) and given in Figure 4(b) presents more randomness compared to the attractor obtained by the HC signals (real and decimal parts) and given in Figure 4(a). Thereafter, we used only the decimal part of hyperchaotic signals encoded on 32 bits as random perturbation signals.



(a) Obtained hyperchaotic signals (real and decimal parts)



(b) Obtained hyperchaotic signals (decimal part)

Figure 3: Simulation results of the considered Lorenz hyperchaotic system.

3.3.1. Digital Implementation of the Proposed HC-PRNG. To realize the HC-PRNG component, we opt for a hardware behaviour description based on Moore's Finite State Machine [35]. This FSM system describes the numerical resolution based on Runge-Kutta method performing (7). Thereby, the RK-4 method is implemented in digital way suitable for FPGA using the FSM system coded in VHDL language and detailed in Figure 5. This figure describes the FSM system corresponding to the internal states of the HC-PRNG layer. A detailed description at each state of the FSM system is presented as follows.

State_1. It is the initialization state. All outputs are set to zero and the state variables are initialized with the initial conditions ($x_0 = y_0 = z_0 = w_0 = -10$).

State_2. In this state, we compute k_0, m_0, n_0, and p_0 and the intermediate points x_{n1}, y_{n1}, z_{n1}, and w_{n1}. RK4 signal is always set to zero. At the next rising edge of clock signal, it jumps to the next state without any condition.

State_3. The values of the intermediate points x_{n1}, y_{n1}, z_{n1}, and w_{n1} are used to update the variables α, β, γ, and δ, respectively. At the next rising edge of clock signal, it jumps to the next state without any condition.

State_4. In this state, we compute the initial slopes k_1, m_1, n_1, and p_1 and the intermediate points x_{n2}, y_{n2}, z_{n2}, and w_{n2}, respectively. RK4 signal is always set to zero. At the next rising edge of clock signal, it jumps to the next state without any condition.

State_5. The values of the intermediate points calculated in state_4 are used to update the variables α, β, γ, and δ. At the next rising edge of clock signal, it jumps to the next state without any condition.

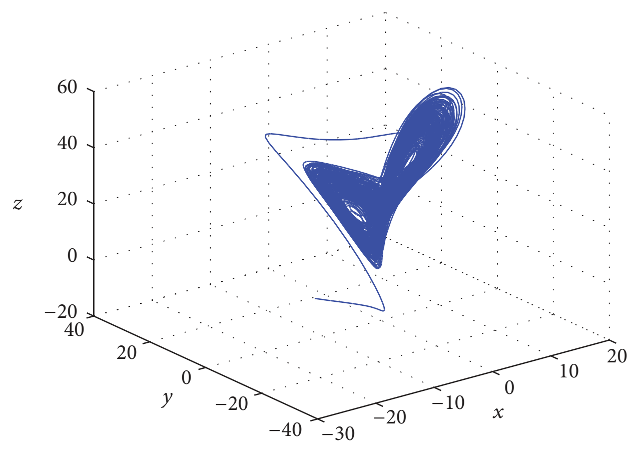
State_6. In this state, we compute the initial slopes k_2, m_2, n_2, and p_2 and the intermediate points x_{n3}, y_{n3}, z_{n3}, and w_{n3}, respectively. RK4 signal is always set to zero. At the next rising edge of clock signal, it jumps to the next state without any condition.

State_7. The values of the intermediate points calculated in state_6 are used to update the variables α, β, γ, and δ. At the next rising edge of clock signal, it jumps to the next state without any condition.

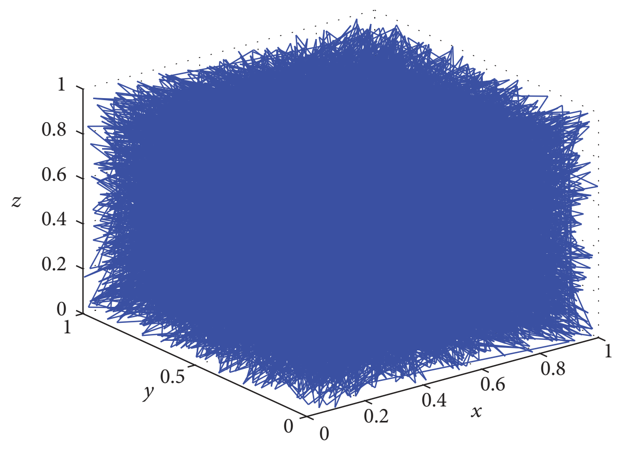
State_8. In this state, the HC solutions x, y, z, and w of (8) are calculated, and the RK4 signal is set to one to indicate the end of the process. At the next rising edge of clock signal, it jumps to state_end without any condition.

State_End. It is the last state of the FSM, the first chaotic key values x, y, z, and w are generated to perturb SNOW-3G, and they are also used to update the variables α, β, γ, and δ, respectively, for the next chaotic key calculation. At the next rising edge of clock signal, if $Cp < N$, then it jumps to state_2 to start new generation key cycle. But if $Cp = N$, which indicates that all the required keys are generated, the FSM jumps to state_1 to wait a new order of N keystream generation. Cp is a command to control the end of output keystreams and N is the number of keystreams to be generated.

3.4. Keystream Generation. To ensure the synchronization of the enhanced SNOW-3G stream cipher parts, all its components are ordered by the same clock and reset signals. Thereby, two main 128-bit inputs CK and IV are loaded to the LFSR stages before starting the process. And then the algorithm is performed according to two operating modes (initialization and keystream modes) depending on the LFSR mode. The perturbation of the LFSR and FSM layers is ordered by a specific block command which defines the starting of perturbation (see Figures 2 and 6). This block provides one perturbation command which is defined by the 4 Least Significant Bits (LSB) of the Initial Vector (LSB(IV) = IV[3 down to 0]). The keystream generator is performed from the following processing steps.



(a) Attractors (x, y, and z) (real and decimal parts)



(b) Attractors (x, y, and z) (decimal part)

FIGURE 4: Obtained attractors of the proposed Lorenz hyperchaotic system.

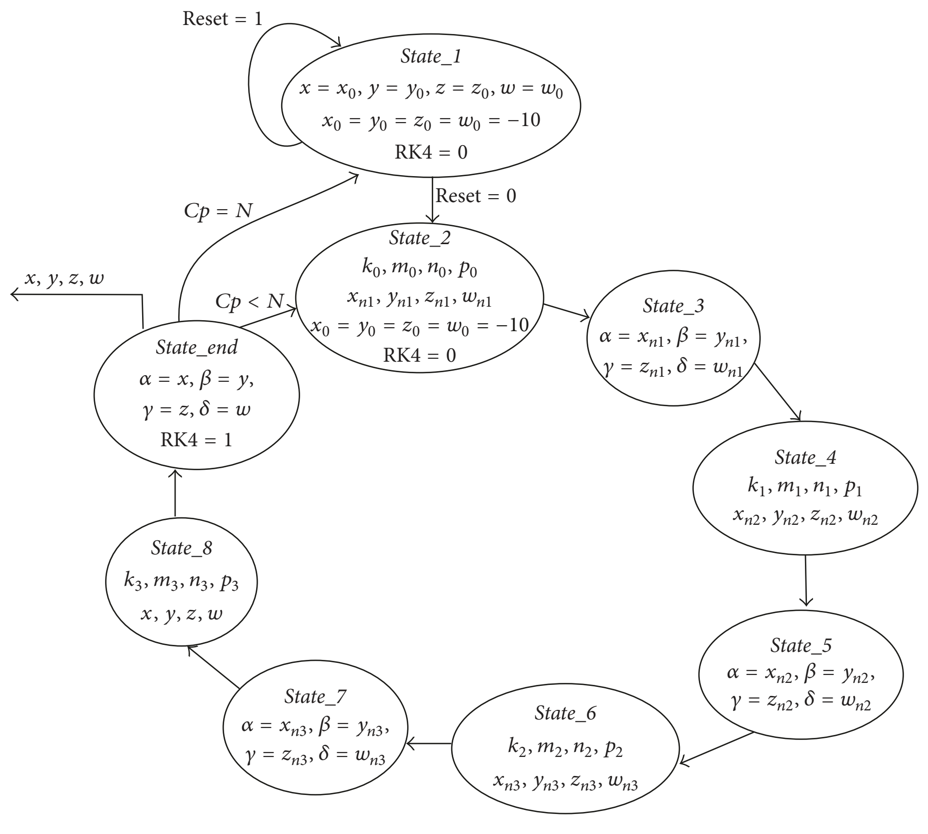


FIGURE 5: Description of the FSM system performing the RK-4 resolution of Lorenz's HC generator.

Key Loading. This step is performed before the initialization mode. First, it begins by dividing each of the two 128-bit input words CK and IV into 32-bit words as follows:

$$\begin{aligned} \mathrm{IV} &= \mathrm{IV0} \parallel \mathrm{IV1} \parallel \mathrm{IV2} \parallel \mathrm{IV3}, \\ K &= K0 \parallel K1 \parallel K2 \parallel K3. \end{aligned} \tag{14}$$

Then, the 16 LFSR stage registers are loaded by the formed 32-bit words as follows:

$$S_{15} = K3 \oplus \mathrm{IV0},$$

$$S_{14} = K2,$$

$$S_{13} = K1,$$

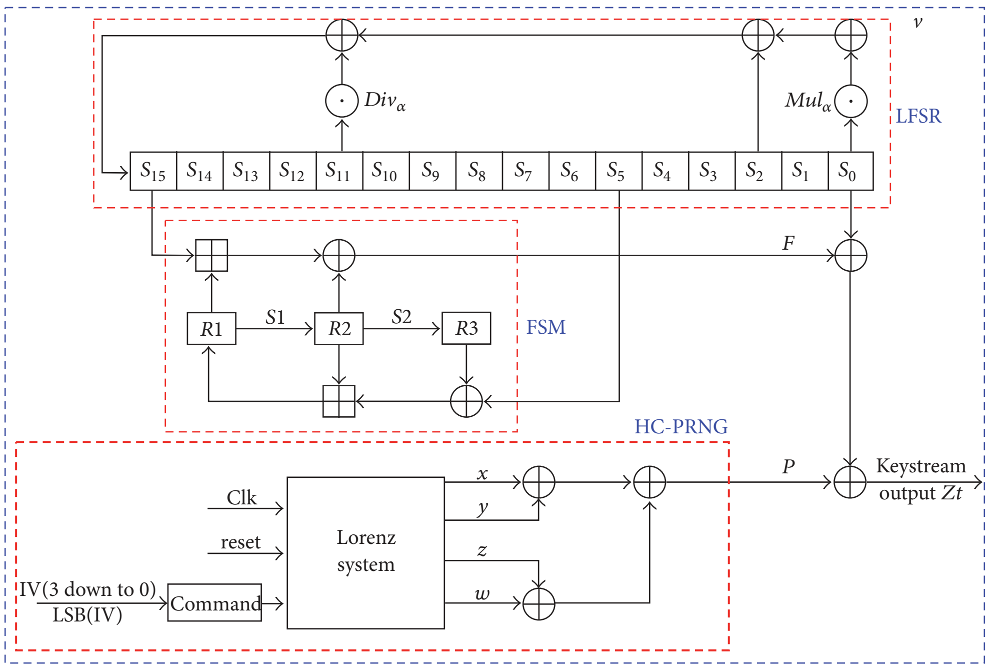


FIGURE 6: The architecture of our proposed hyperchaotic SNOW-3G stream cipher in its keystream mode.

$$
\begin{aligned}
S_{12} &= K0 \oplus \mathrm{IV}1,\\
S_{11} &= K3 \oplus 1,\\
S_{10} &= K2 \oplus 1 \oplus \mathrm{IV}2,\\
S_9 &= K1 \oplus 1 \oplus \mathrm{IV}3,\\
S_8 &= K0 \oplus 1,\\
S_7 &= K3,\\
S_6 &= K2,\\
S_5 &= K1,\\
S_4 &= K0,\\
S_3 &= K3 \oplus 1,\\
S_2 &= K2 \oplus 1,\\
S_1 &= K1 \oplus 1,\\
S_0 &= K0 \oplus 1.
\end{aligned}
\tag{15}
$$

In parallel of this key loading step, the HC-PRNG layer command is initialized by the IV as follows:

$$Command = \mathrm{IV}\,(3\ downto\ 0)\,. \tag{16}$$

Initialization Mode. When the key loading process is completed, the FSM layer is executed using S_{15} and S_5 stage registers from LFSR layer to generate a 32-bit output word F expressed by (5). The registers of the FSM layer are set to zero ($R1 = R2 = R3 = 0$) before computing F. At the same time, the HC-PRNG is executed to produce the 32-bit word P perturbation signal. After waiting n clock cycles, P is mixed with F by a bitwise XOR operator in order to perturb the LFSR updating. Consequently, the new feedback value is calculated from the following equation:

$$
\begin{aligned}
v = {} & (S_{0,1} \parallel S_{0,2} \parallel S_{0,3} \parallel 0x00) \oplus Mul_\alpha (S_{0,0}) \oplus S_2\\
& \oplus (0x00 \parallel S_{11,0} \parallel S_{11,1} \parallel S_{11,2}) \oplus Div_\alpha (S_{11,3}) \oplus F\\
& \oplus P.
\end{aligned}
\tag{17}
$$

The number of clock cycles is defined by $n = 0, 1, \ldots, 15$, corresponding to the value contained in *Command* which depends on IV. The 16 LFSR stage registers are updated as follows:

$$S_{15} = v, S_{14} = S_{15}, S_{13} = S_{14}, \ldots, S_1 = S_2, S_0 = S_1. \tag{18}$$

The FSM and HC-PRNG layers run in the initialization mode 32 clock cycles, producing the two 32-bit words F and P, which are used by the LFSR feedback value as defined in (17). Note that the initialization mode does not produce any output keystream Zt. The detailed architecture of the proposed HC SNOW-3G stream cipher in its initialization mode is given in Figure 2.

Keystream Mode. At the end of initialization mode, the FSM is performed during one clock cycle, and the associated output word Zt is discarded. Then, the LFSR feedback value is calculated by (4), and the 16 stage registers are updated as follows:

$$S_{15} = v, S_{14} = S_{15}, S_{13} = S_{14}, \ldots, S_1 = S_2, S_0 = S_1. \tag{19}$$

Table 1: FPGA implementation results and comparison.

Resource	HC SNOW-3G (proposed)	Regular SNOW-3G	PLCM SNOW-3G [31]
Device	Virtex XC5vfx70t-1ff1136	Virtex XC5vfx70t-1ff1136	Virtex XC5vfx70t-1ff1136
Number of slice registers	2391	912	1131
Number of slice LUTs	7881	1108	2495
Number of fully used LUT-FFpairs	1713	881	1046
Number of bonded IOBs	56	56	56
Number of BUFG/BUFGCTRLs	1	1	1
Number of blockRAM/FIFO	10	10	10
Maximum frequency (Mhz)	28.84	287.81	13.83
Throughput (Mbps)	922.88	8264.32	442.56
Power (Watts)	1.36	1.34	1.36

For the next clock cycles, the FSM 32-bit output word F generated by (5) is mixed with the HC-PRNG 32-bit output word P and the register S_0 of the LFSR in order to produce the 32-bit keystream word Zt as defined by (20). The architecture of the proposed HC SNOW-3G stream cipher in its keystream mode is given in Figure 6:

$$Zt = F \oplus S_0 \oplus P. \tag{20}$$

Therefore, in this processing mode, the LFSR operates on keystream mode, and both FSM and HC-PRNG layers are clocked at each clock cycle to generate a 32-bit keystream word Zt as mentioned before. This mode runs j times where j is the number of 32-bit blocks to be encrypted.

4. FPGA Implantation of the Proposed Design

The proposed architecture is mainly based on three layers, as already discussed: the HC-PRNG layer and the FSM and LFSR layers of the regular SNOW-3G. The added HC-PRNG layer is used in order to perturb the LFSR synchronization in the initialization mode and the FSM output in the keystream mode, as described in Figures 2 and 6, respectively.

The RTL description of the proposed architecture has been implemented on Xilinx Virtex-5 FPGA (XC5vfx70t-1) through the ML507 Virtex Development platform [37] using VHDL structural description. ISE 13.1i of Xilinx tools have been used for this digital implementation allowing obtaining the logic resource requirements and the associated real-time constraints. The Xilinx Synthesis Technology (XST) results after place and route show the performance analysis of our implementation in Table 1. To exhibit the importance of this architecture, we also implemented the regular SNOW-3G and the PLCM (Piecewise Linear Chaotic Map) SNOW-3G proposed in [31] on the same FPGA device. The results are presented in Table 1. This table depicts comparison in terms of hardware resources, maximum time frequency, throughput, and the power consumption. Our logic implementation on a Xilinx Virtex-5 device requires only 2391 Slice registers, 56 bonded IOBs, and 10 block RAMs. In addition, the low power consumption estimated by 1.36 Watts is suitable for embedded electronic applications thanks to the considered FSM encoding algorithm and the fixed-point coding data.

To evaluate the performance of the proposed stream cipher, the throughput rate metric is considered. The throughput rate is defined as the number of bits of key in a unit of time for a keystream. In view of the performance results (see Table 1), we have achieved a maximal throughput of 922.88 Mbps for the proposed HC SNOW-3G and 422.56 Mbps for the PLCM SNOW-3G, where the regular SNOW-3G achieves a maximal throughput of 8264.32 Mbps. Thus, the proposed architecture is two times faster than PLCM SNOW-3G architecture and nine times slower than the regular SNOW-3G architecture. On the other hand, the proposed HC SNOW-3G architecture occupies greater area compared to the regular and PLCM SNOW-3G architectures. The difference in terms of hardware area is due to the following:

(1) The HC system is for order 4 where the PLCM system is for order 3.

(2) The HC system is more complex and has more nonlinearity than the PLCM system.

(3) The HC system complexity provides good randomness and security compared to the PLCM system.

Based on the obtained results and the discussed comparison, we conclude that the proposed HC SNOW-3G is better than the PLCM SNOW-3G in terms of randomness, security, and high throughput.

To ensure that the proposed architecture enhances the complexity of the standardized SNOW-3G stream cipher suitable for increasing the robustness of data encryption, security and statistical tests are performed in order to prove the high level of security obtained from our architecture.

5. Cryptographic Strength and Performance

The security and statistical analyses of our designed architecture are presented in this section. The proposed stream cipher HC SNOW-3G algorithm is subjected to two security tests: (1) key sensitivity and (2) key space and several statistical analyses such as uniformity and randomness. to demonstrate its satisfactory security and good randomness with regard to the added HC-PRNG layer.

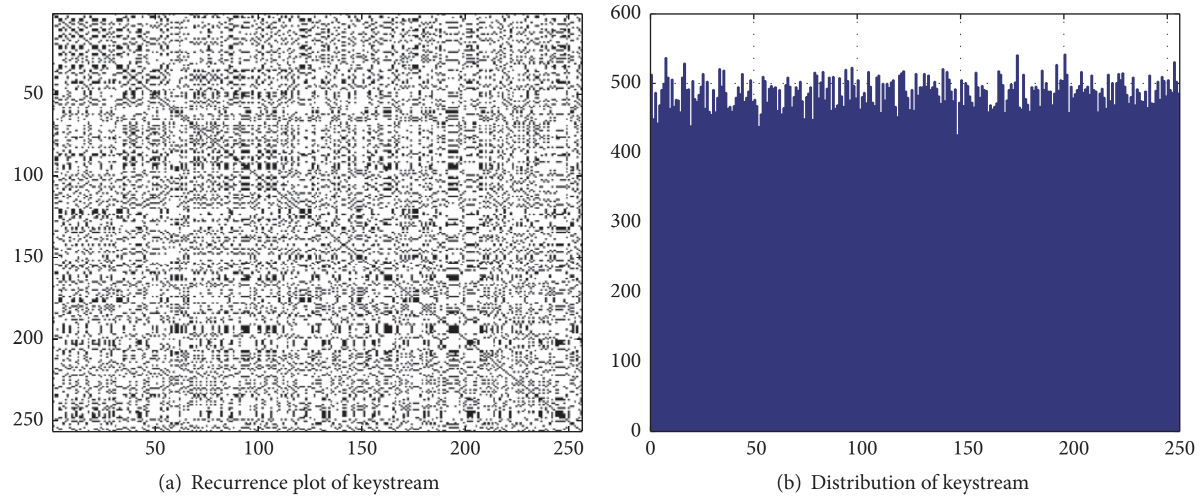


FIGURE 7: Uniformity of keystream generated.

5.1. Statistical Analysis. Many statistical analyses are applied to our proposed stream cipher scheme to demonstrate its robustness such as randomness, uniformity, mixing nature, low coefficient correlation between original and encrypted data packets, and NIST tests. The simulations are done using keystreams of 125000 bytes in length which are normally distributed with a mean equal to 128 and standard deviation equal to 8.

5.1.1. Randomness and Uniformity Analysis. Any performing stream cipher should produce high random keystreams and with best uniformity representation. Therefore, a keystream of 1000000 bits produced by our design is analyzed. The recurrence plot representation (Figure 7(a)) and distribution (Figure 7(b)) of the generated keystream indicate clearly a good degree of randomness and uniformity.

The high randomness of generated keystream is also proved by the random-excursion and random-excursion-variant tests of NIST [38]. To perform these tests, we analyzed 250 keystream sequences where each sequence is composed of 50 packets of 1000000 bits. The proportion value of each sequence is shown in Figure 8. The determined range of acceptable proportion using the confidence interval is defined by

$$\widehat{P} \mp 3\sqrt{\frac{\widehat{P}\left(1-\widehat{P}\right)}{m}}, \tag{21}$$

where m is the sample size, $\widehat{P} = 1 - \alpha$, and $\alpha = 0.01$.

In our case, the confidence interval is $0.99 \mp 3\sqrt{(0.99 \times 0.01)/50}$ (i.e., the proportion value should be laid above 0.947786) which is represented in Figure 8 by the red line. The simulation results indicate clearly the high randomness of keystreams. Indeed, all random-excursion and random-excursion-variant proportion values are above the minimum proportion value and close to ideal value 1. In addition, other tests proving randomness and uniformity of our proposed stream cipher are presented in the following subsections.

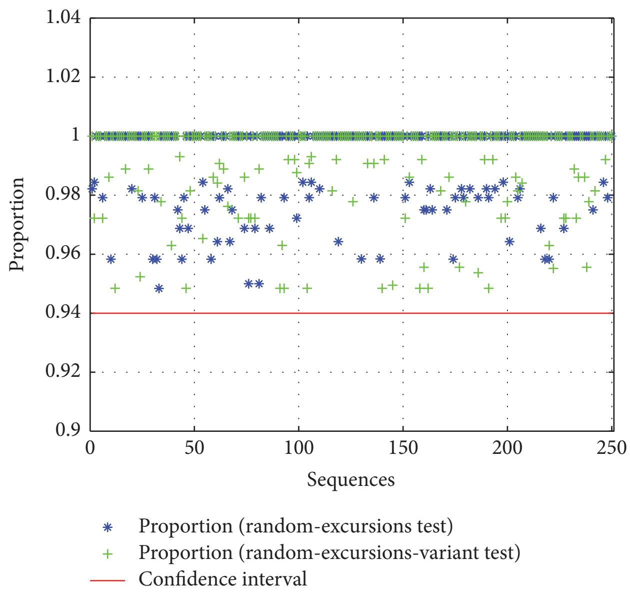


FIGURE 8: Proportion values of random-excursion and random-excursion-variant NIST tests.

5.1.2. Low Correlation Coefficient. The correlation is an important test to prove the independence of encrypted data from the original ones. Indeed, the encrypted data are increasingly different from the original ones when the correlation coefficient becomes as low as possible. The correlation coefficient is computed according to [39]

$$\rho_{x,y} = \frac{\operatorname{cov}(x,y)}{\sqrt{D(x) \times D(y)}}, \tag{22}$$

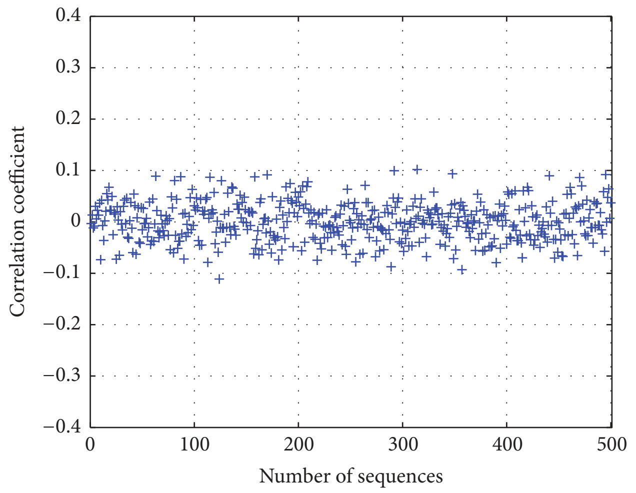


Figure 9: Correlation coefficient between original and encrypted packets.

where

$$\begin{aligned} \operatorname{cov}(x,y) &= E\left[\{x-E(x)\}\{y-E(y)\}\right], \\ E(x) &= \frac{1}{n}\times\sum_{i=1}^{n} x_i, \\ D(x) &= \frac{1}{n}\times\sum_{i=1}^{n}\{x_i - E[x]\}. \end{aligned} \tag{23}$$

For performing this test, we have analysed 500 encrypted packets with different keys, and in each case we have calculated the correlation coefficient between original packet and its corresponding encrypted packet. The results are shown in Figure 9. The illustrated results indicate clearly that no detectable correlation exists between the original and corresponding cipher packets considering that the correlation coefficient is always close to ideal value zero.

5.1.3. Distribution and Chi-Square Uniformity. An additional statistical approach, the distribution of frequency counts which can define the uniformity of generated keystreams, is performed. Any sequence is categorized uniform if its corresponding distribution is close to a uniform distribution. In Figures 10(a) and 10(b) distributions of an original and its corresponding encrypted packet obtained by our proposed stream cipher are presented, respectively. From these results, we deduce that the contents of the encrypted packet are spread over the entire space and have a uniform distribution thanks to a good level of mixing included by our added HC-PRNG layer. To validate this uniformity, the Chi-square test is applied [40]. The Chi-square distance is computed according to

$$\chi^2_{\text{test}} = \sum_{i=1}^{l}\frac{o_i - e_i}{e_i}. \tag{24}$$

Note that l represents the number of levels (here 256), o_i represents the observed occurrence frequencies of each level of field size (0–255) in the histogram of ciphered generation contents, and e_i is the expected occurrence frequency of uniform distribution. Before applying the test, we fixed the significant level to 0.05. After test, we analyzed the result according to the χ^2_{test} value. Indeed, if $\chi^2_{\text{test}} \leq \chi^2_{\text{theory}}(255, 0.05) \approx 293$ (in our case study), and the null hypothesis is not rejected, then the distribution of the histogram is uniform. In Figure 11, the results of the Chi-square test for 500 different ciphered packets of 125000 bytes each are shown. Based on these results, we conclude that a stronger mixing property is obtained with uniform distribution exceeding ≈95% considering all values are below the value of χ^2_{theory} presented by a red line in Figure 11.

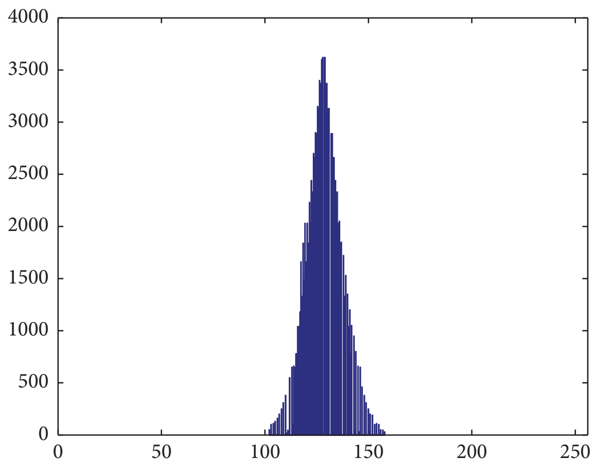
5.1.4. NIST Keystream Analysis. Various statistical tests can be used to evaluate the randomness of a sequence, for example, NIST [38], DIEHARD [41], NESSIE [42]. In our case study, the NIST statistical tests are used to evaluate our HC SNOW-3G stream cipher. To illustrate the importance of the proposed design, we repeat the three statistical tests done in [14] to evaluate the randomness of the generated keystreams. To perform comparison results with previous work [14], we have considered the usual significance level used in the NIST tests [38] and the same sample size sequences. Therefore, in all the experiments the significance level was set to 0.01 and sample size was set to 300 sequences. The tests are defined as follows:

(1) Long keystream data set test: it begins by generating 2^{20} bits for each of the 300 random CK (set IV to zero in all the cases) to obtain 300 vectors of 1048576 bits. Then, it evaluates the result vectors by the 15 NIST tests.

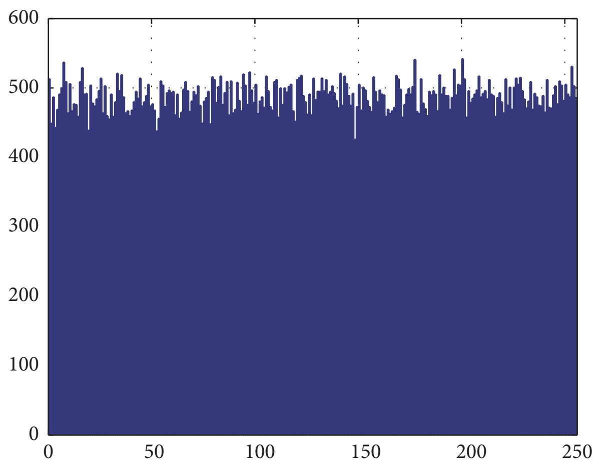
(2) Short keystream data set test: it begins by generating 2^{10} bits for each of the 307200 random CK (set IV to zero in all the cases). The concatenation of all the keystreams of 2^{10} bits allows obtaining 300 vectors of 1048576 bits. Then, it evaluates the result vectors by the 15 NIST tests.

(3) IV data set: it begins by generating 256 sequences of 4096 bits each for 300 random CK. Note that the first sequence of each key is generated with IV set to zero. Then, it is incremented for the following 255 sequences. As a result, we obtain 300 vectors of 1048576 bits for evaluation by the 15 NIST tests.

To perform the analysis, we compare the results of our proposed HC SNOW-3G stream cipher with results of the regular SNOW-3G done in [14]. The aim of this analysis is to prove the amelioration of the initialization process and randomness of short keystream data set. As a result, we find that both algorithms (regular SNOW-3G and our proposed design) pass in success all the fifteen NIST tests for test set 1 (long keystream data set) and test set 3 (IV data set). However, only our proposed design passes in success all the fifteen NIST tests for test set 2 (short keystream data set).



(a) Distribution of one original sequence



(b) Distribution of corresponding encrypted sequence

FIGURE 10: The distribution comparison between original and encrypted sequences.

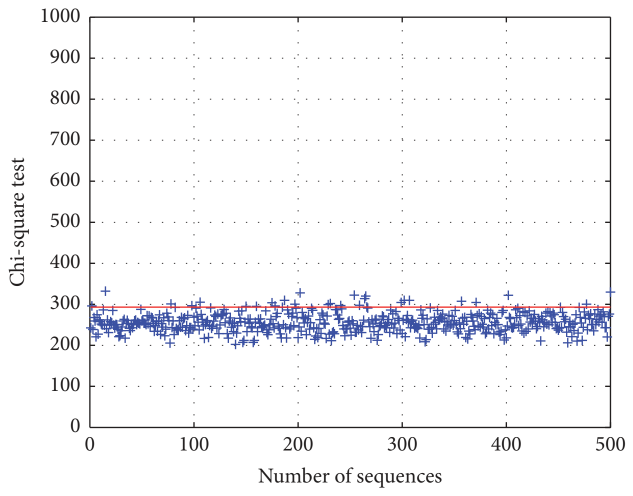


FIGURE 11: Variation of χ^2_{test} of cipher sequence versus 500 random keys.

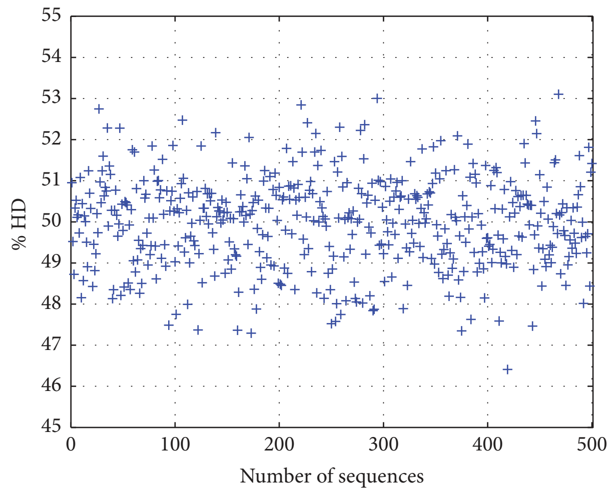


FIGURE 12: The key sensitivity analysis results.

Indeed, the regular SNOW-3G fails in the eight NIST tests as it is presented in Table 2. More precisely, this table gives a comparison of short keystream generated by the regular SNOW-3G cipher with short keystream generated by our proposed HC SNOW-3G design. We can clearly observe that all tests are passed compared to the regular SNOW-3G cipher where NIST tests based on the short keystream data set have failed [14]. Therefore, we can conclude that our proposed design present an enhanced initialization process and generates very good pseudo-random keystream, which justify its high level of security. Indeed, it is very difficult to find a plaintext masked by a sequence generated by this stream cipher which can be used in the 128-EEA1 confidentiality algorithm. Similarly, it is theoretically impossible to generate the same MAC by the 128-EEIA1 integrity algorithm based on this stream cipher without knowing original message.

5.2. Key Sensitivity Analysis.

The key sensitivity analysis is one among the most important security tests of cryptosystems. In other words, an ideal cryptosystem should be sensitive with respect to the secret key. For testing the key sensitivity of the proposed architecture, we analyzed 500 random keys, where in each case a single bit in the secret key is changed. Then, the percent of Hamming Distance (HD) is calculated according to

$$\text{HD} = \frac{\sum_{k=1}^{N} C_i \oplus C_i'}{N} \times 100\%, \qquad (25)$$

where N is the length in bit level of the encrypted packet and C_i and C_i' are the corresponding ciphers packets using two secret keys different in one bit K_i and K_i', respectively. The obtained results are shown in Figure 12 in which a high sensitivity is indicated, while the average Hamming Distance

Table 2: Comparison results of NIST tests.

The number of test	The type of test	Regular SNOW-3G [14]	Proposed SNOW-3G based on HC-PRNG
1	Frequency (monobit) test	Success	Success
2	Frequency test with a block	Success	Success
3	Runs test	Fail	Success
4	Long runs of ones test	Fail	Success
5	Binary matrix rank test	Fail	Success
6	Discrete Fourier transform (spectral) test	Fail	Success
7	Nonoverlapping template matching test	Fail	Success
8	Overlapping template matching test	Fail	Success
9	Maurer's "universal statistical" test	Success	Success
10	Linear complexity test	Success	Success
11	Serial test	Fail	Success
12	Approximate entropy test	Fail	Success
13	Cumulative sums (Cusum) test	Success	Success
14	Random-excursion test	Success	Success
15	Random-excursion-variant test	Success	Success

percent is closer to the optimal value 50% in bit level. We conclude that a change of one bit in the secret key leads to thoroughly different keystreams with respect to the avalanche effect [43].

5.3. Key Space Analysis. The proposed enhanced stream cipher HC SNOW-3G has the advantage of being able to generate a usable random number while maintaining a large enough key space and high level security. The key space achieved in the present work is increased from 2^{128} in the regular SNOW-3G stream cipher to 2^{132} by addition of 2^4 through the controlling command used to perform the perturbation process which is coded in 16 bits.

Additionally, considering all the initial values ($x0, y0, z0$, and $w0$) and control parameters (a, b, c, and d) of our HC generator (defined by (7)) as internal keys, the key space of the generator is based upon those 8 parameters. First, we fix the initial values ($x0, y0, z0$, and $w0$) and the three parameters a, b, and c whereas d is considered the key of the random number generator (we change slightly the value of d) to generate a new binary sequence. The experiment is done more than 1000 times. The obtained results show that the variance ratio of each bit is approximated to 99%, even if the change of initial value or control parameter is an extremely small value 10^{-15}, which means that the system is extremely sensitive to the key changes. Consequently, the key space is larger than 10^{15} for this considered key (d). Similarly, the 4 initial values and the 3 remaining control parameters can be considered as key of our random sequence generator. Finally, the key space is defined by $(10^{15})^8 = 10^{120} \approx 2^{398}$, which is large enough to withstand attacks limited by a key space of 2^{128}.

Otherwise, the HC generator key space of 2^{398} is much better than the PLCM generator key space of 2^{124} [44]. Finally, we conclude that our HC generator is sufficiently large to make the brute-force attacks infeasible.

5.4. Synthesis and Discussion. In cryptography, any cipher must exhibit the confusion and diffusion defined on Shannon's theory with success to be considered secure enough and able to resist attacks [45]. Our designed cipher is resistant against the powerful attacks that are based on statistical analysis. Moreover, the avalanche effect is attained when repeating the tests several times [43], and then a significant difference is caused in the generated keystream by changing of a single bit in the Ciphering Key, which makes our cipher resistant against a special case of differential cryptanalysis. The proposed cipher is also resistant against the statistical attacks proved by the obtained statistical analysis results. Furthermore, the key space of 2^{132} is achieved, which is sufficiently large to make the brute-force attack infeasible. Finally, the added nonrevertible HC generator limits the ability of the attackers who try to break the cipher. The high randomness and security of our cipher is achieved according to the new combined HC technique with both regular LFSR and FSM layers in addition to the strength of the original SNOW-3G against contemporary cryptanalytic.

6. Conclusion

In this paper, an enhanced SNOW-3G stream cipher is proposed. The objective is the improvement of the randomness and the complexity of the regular SNOW-3G stream cipher which presents a weakness in its initialization mode leading to failure of the NIST test based on the short keystream data set. The adopted original approach is to include a HC-PRNG (Hyperchaotic Pseudo-Random Number Generator) based on 4D Lorenz system in the regular SNOW-3G cipher as an additional layer while keeping standardization and in order to generate a more robustness keystream. The proposed architecture performs the perturbation concept of digital LFSR (Linear Feedback Shift Register) sequences and the digital output of the regular SNOW-3G cipher part through one hyperchaotic generator which is used as PRNG *(Pseudo-Random Number Generator)*. The obtained statistical and security tests proved that the proposed stream cipher is more secure than the regular SNOW-3G cipher as long as it passes all analyses presented in this paper. Therefore, the proposed stream cipher can replace the regular SNOW-3G stream cipher without modification in its mechanism while respecting its standard requirements, mainly the length of input and output parameters. Consequently, this paper proposed a new digital hyperchaotic SNOW-3G stream cipher architecture which can be used as the kernel of the confidentiality and integrity algorithms used in the UMTS and LTE standards. The hardware implementation results of our proposed hyperchaotic SNOW-3G cipher provide a good trade-off between high security, performance, and hardware resources. Indeed, the proposed implemented design in FPGA Virtex-5 device consumes a low logic area cost while it achieves a throughput of 922.88 Mbps. Finally, our technique exhibits attractive performance and can be extended to enhancing the 4G security level by scrambling the plaintext for the LTE 128-EEA1 confidentiality algorithm or controlling the integrity of a signalling message for 128-EIA1 algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] 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.

[2] C. Blanchard, “Security for the third generation (3G) mobile system,” *Information Security Technical Report*, vol. 5, no. 3, pp. 55–65, 2000.

[3] Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2; Document 2: SNOW 3G specification, Technical Specification (TS) TS 35.216 V12.0.0, 3GPP (Sep 2014-09).

[4] Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2; Document 1: UEA2 and UIA2 specifications, Technical Specification (TS) TS 35.215 V12.0.0, 3GPP (Sep 2014-09).

[5] Tech. Rep., 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 specification, Technical Specification (TS) TS 35.201 V12.0.0, 3GPP (Sep 2014-09).

[6] 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI specification, Technical Specification (TS) TS 35.202 V12.0.0, 3GPP (Sep 2014-09).

[7] Digital cellular telecommunications system (phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; 3GPP System Architecture Evolution (SAE); Security Architecture, Technical Specification (TS) ETSI TS 133 401 V11.5.0, 3GPP (Oct 2012-10).

[8] P. Szalachowski, B. Ksiezopolski, and Z. Kotulski, “CMAC, CCM and GCM/GMAC: advanced modes of operation of symmetric block ciphers in wireless sensor networks,” *Information Processing Letters*, vol. 110, no. 7, pp. 247–251, 2010.

[9] 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.

[10] Specification of the 3GPP Confidentiality and Integrity Algorithms EEA3 & EIA3; Document 1: EEA3 and EIA3 specifications, Technical Specification (TS) TS 35.221 V12.0.0, 3GPP (Sep 2014-09).

[11] Specification of the 3GPP Confidentiality and Integrity Algorithms EEA3 & EIA3; Document 2: ZUC specification, Technical Specification (TS) TS 35.222 V12.0.0, 3GPP (Sep 2014-09).

[12] S. Murphy, “The advanced encryption standard (AES),” *Information Security Technical Report*, vol. 4, no. 4, pp. 12–17, 1999.

[13] F. Lafitte, O. Markowitch, and D. Van Heule, “SAT based analysis of LTE stream cipher ZUC,” *Journal of Information Security and Applications*, vol. 22, pp. 54–65, 2015.

[14] P. Bohm, “Statistical Evaluation of Stream Cipher SNOW 3G,” in *Proceedings of the “Constantin Brancusi” University of Targu Jiu Engineering Faculty Scientific Conference with International Participation*, Targu Jiu, Romania, 13th edition, 2008.

[15] M. S. N. Nia and T. Eghlidos, “Improved Heuristic guess and determine attack on SNOW 3G stream cipher,” in *Proceedings of the 2014 7th International Symposium on Telecommunications, IST 2014*, pp. 972–976, September 2014.

[16] B. Debraize and I. M. Corbella, “Fault analysis of the stream cipher snow 3G,” in *Proceedings of the 6th International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009*, pp. 103–110, Lausanne, Switzerland, September 2009.

[17] B. B. Brumley, R. M. Hakala, K. Nyberg, and S. Sovio, “Consecutive S-box lookups: a timing attack on SNOW 3G,” in *Proceedings of the 12th International Conference, ICICS 2010*, pp. 171–185, Barcelona , Spain, 2010.

[18] H. Gilbert and M. Minier, “A Collision Attack on 7 Rounds of Rijndael,” in *Proceedings of the AES Candidate Conference*, pp. 230–241, 2000.

[19] A. Biryukov, D. Priemuth-Schmid, and B. Zhang, “Multiset Collision Attacks on Reduced-Round SNOW 3G and SNOW 3G ⊕,” in *Proceedings of the 8th International Conference, ACNS 2010*, pp. 139–153, Beijing, China, 2010.

[20] A. Kircanski and A. M. Youssef, “On the sliding property of SNOW 3G and SNOW 2.0,” *IET Information Security*, vol. 5, no. 4, pp. 199–206, 2011.

[21] S. Sadoudi, C. Tanougast, and M. S. Azzaz, “First experimental solution for channel noise sensibility in digital chaotic communications,” *Progress in Electromagnetics Research C*, vol. 32, pp. 181–196, 2012.

[22] M. S. Azzaz, C. Tanougast, S. Sadoudi, and A. Bouridane, “Synchronized hybrid chaotic generators: application to real-time wireless speech encryption,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 8, pp. 2035–2047, 2013.

[23] O. E. Rossler, "An equation for hyperchaos," *Physics Letters A*, vol. 71, no. 2-3, pp. 155–157, 1979.

[24] S. Nikolov and S. Clodong, "Occurrence of regular, chaotic and hyperchaotic behavior in a family of modified Rossler hyperchaotic systems," *Chaos, Solitons & Fractals*, vol. 22, no. 2, pp. 407–431, 2004.

[25] A. Chen, J. Lu, J. Lü, and S. Yu, "Generating hyperchaotic Lü attractor via state feedback control," *Physica A: Statistical Mechanics and its Applications*, vol. 36, no. 4, pp. 103–110, 2006.

[26] T. Gao, Z. Chen, Z. Yuan, and G. Chen, "A hyperchaos generated from Chen's system," *International Journal of Modern Physics C*, vol. 17, no. 4, pp. 471–478, 2006.

[27] Y. Li, W. K. S. Tang, and G. Chen, "Generating hyperchaos via state feedback control," *International Journal of Bifurcation and Chaos*, vol. 15, no. 10, pp. 3367–3375, 2005.

[28] G. Tiegang, C. Zengqiang, G. Qiaolun, and Z. Yuan, "A new hyper-chaos generated from generalized Lorenz system via nonlinear feedback," *Chaos, Solitons & Fractals*, vol. 35, no. 2, pp. 390–397, 2008.

[29] Q. Jia, "Hyperchaos generated from the Lorenz chaotic system and its control," *Physics Letters A*, vol. 366, no. 3, pp. 217–222, 2007.

[30] W. Xingyuan and W. Mingjun, "A hyperchaos generated from Lorenz system," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 14, pp. 3751–3758, 2008.

[31] M. Wasi, A. Arif, and S. Windarta, "Modified SNOW 3G: Stream cipher algorithm using piecewise linear chaotic map," in *Proceedings of the AIP Conference Proceedings*, vol. 1707, Issue 1, 2016.

[32] G. Alvarez, G. P. F. Monotoya, G. Pastor, and M. Romera, "Chaotic cryptosystems," in *Proceedings of the IEEE 33rd Annual International Carnahan Conference on Security Technology*, pp. 332–338, Madrid, Spain, October 1999.

[33] E. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, no. 20, pp. 130–141, 1963.

[34] T. Tsubone and T. Saito, "Hyperchaos from a 4-D manifold piecewise-linear system," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, no. 9, pp. 889–894, 1998.

[35] S. Sadoudi, C. Tanougast, M. S. Azzaz, and A. Dandache, "Design and FPGA implementation of a wireless hyperchaotic communication system for secure real-time image transmission," *Eurasip Journal on Image and Video Processing*, vol. 2013, article 43, pp. 1–18, 2013.

[36] J.-P. Demailly, "Analyse numérique et équations différentielles, EDP Sciences 4," *Collection Grenoble Sciences*, 2006.

[37] Virtex 5 FPGA Configuration User Guide, ug702, Xilinx (sep 2012).

[38] A. Rukhin, J. Sota, J. Nechvatal et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Special Publication NIST 800-22, National Institute of Standards and Technology, 2010.

[39] J. L. Rodgers and A. W. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. 59–66, 1988.

[40] C. Bates and B. Carl, "The Chi-square Test of Goodness of Fit for Bivariate Normal Distribution," Defence Technical Information Center.

[41] G. Marsaglia, "The Marsaglia Random Number CDROM, with The Diehard Battery of Tests of Randomness," Florida State University under a grant from The National Science Foundation, 1985.

[42] B. Preneel, "New European Schemes for Signature, Integrity and Encryption (NESSIE): A Status Report," in *Proceedings of the International Workshop on Public Key Cryptography (PKC) 2002: Public Key Cryptography*, pp. 297–309, 2002.

[43] D. Han, L. Min, and G. Chen, "A Stream Encryption Scheme with Both Key and Plaintext Avalanche Effects for Designing Chaos-Based Pseudorandom Number Generator with Application to Image Encryption," *International Journal of Bifurcation and Chaos*, vol. 26, no. 5, article 1650091, 2016.

[44] Y. Hu, C. Zhu, and Z. Wang, "An improved piecewise linear chaotic map based image encryption algorithm," *The Scientific World Journal*, vol. 2014, Article ID 275818, 7 pages, 2014.

[45] C. E. Shannon, "Communication theory of secrecy systems," *Bell Labs Technical Journal*, vol. 28, pp. 656–715, 1949.