Received 31 October 2022, accepted 29 November 2022, date of publication 2 December 2022,
date of current version 8 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3226515

RESEARCH ARTICLE

Triple Pendulum Based Nonlinear Chaos Generator and Its Applications in Cryptography

BIKRAM PAUL[1], SOURADIP PAL[2], ABHISHEK AGRAWAL[3], AND GAURAV TRIVEDI[1], (Member, IEEE)
[1]Department of Electronics and Electrical Engineering, Indian Institute of Technology at Guwahati, Guwahati, Assam 781039, India
[2]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA
[3]CPU Design Engineer in Google Hardware, Bengaluru, Karnataka 560083, India

Corresponding author: Bikram Paul (bikram@iitg.ac.in)

ABSTRACT A nonlinear chaos generator scheme derived from a mechanical triple pendulum physical system is proposed here. The chaotic behavior of the proposed generator is validated against various standardized tests, such as the Lyapunov exponents test, bifurcation diagrams, sensitivity to parametric and to initial values, ergodicity, key space and sensitivity, histogram, correlation, NPCR and UACI, collision test, etc. and compared with existing contemporary methods. The generated chaotic map is utilized to develop various cryptography applications, such as PRNG and symmetric key encryption schemes, which are realized on an FPGA and an ASIC. Chaos-based PRNG is validated successfully using $NIST - SP800$ benchmarks. The proposed encryption scheme illustrates its usage in low power, high throughput applications, where the power consumption, resource utilization, and throughput are $1.785\times$, $1.825\times$, and $2.396\times$ better than other known contemporary chaos-based encryption methods. The average power and area of its ASIC implementation at 180-nm technology are 61.8836 mW and 0.20374 mm^2 at 250 MHz.

INDEX TERMS Nonlinear system, triple pendulum, chaotic map, pseudo random number, symmetric cryptosystem, differential state-space model, FPGA.

I. INTRODUCTION

CHAOTIC dynamic system-based nonlinear dynamic models are employed in various fields, such as synchronization and control, communication, cryptography, neural network, etc. [1], [2], [3]. Chaotic systems are distinguished by their higher sensitivity to the initial conditions, statistical similarity to random signals, and a continuous broadband power spectrum. These properties garnered interest from cryptanalysts to propose various chaos-based cryptographic systems [4], [5]. The relevance and usefulness of chaos have been demonstrated through comparative studies employing characteristics of a chaotic system, and the requirement of a strong cipher [5], [6] or stronger random number generators or initial key generation. Designing cryptographic schemes based on chaotic dynamics can be divided into two steps. In the first step, an encryption algorithm is modeled based on analog chaotic dynamic systems. Thereafter, discretization with finite precision is introduced during hardware implementation. Discretization is the main trade-off between security and resource utilization, and the synchronization method of multiple chaotic systems has been applied in a few earliest cryptography applications.

The piecewise linear chaotic map (PLCM) based $1-$D chaotic map is implemented in [7], whose security vulnerabilities are improved by introducing logistic polynomial map derived piecewise nonlinear chaotic maps [8]. Further, a combination of the polynomial chaotic map and the piecewise nonlinear chaotic map [9] is employed to enhance the speed and security of a cryptosystem. Recently, many real-world chaotic phenomena are being modeled to generate ideal chaotic characteristics [10], [11]. Thus, physical chaotic dynamical systems have become an active research domain. In [9], a combination of the polynomial chaotic map and the piecewise nonlinear chaotic map is employed to enhance the speed and security of a cryptosystem. It is mentioned that various chaotic block cipher encryption schemes are also developed. A chaotic tent map-based single iterative

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masucci.

encryption scheme is proposed in [6], which is further generalized in [12], and multiple iterative-based dynamical chaotic system models are presented in [13]. A novel symmetric block cipher algorithm based on invertible two-dimensional chaotic maps on a torus is presented in [14], whereas schemes formulated by employing iterative logistic equation and synchronized chaotic systems are discussed in [15].

In another approach, different chaotic attractors for a random chunk of binary plain-text data are presented [16] in order to implement a chaotic-based encryption scheme. As per our knowledge, only a few chaotic map-based images and video encryption schemes have been developed in the past. In [10], a hyperchaotic temperature fluctuation model based on the circuit analysis is proposed for image encryption. Simple first order differential equations ($\dot{x} = z$; $\dot{y} = -z(ay + by^2 + xz)$; $\dot{z} = x^2 + y^2 - |x| - 1$) based a new 3D chaotic system employing a peanut-shaped symmetric equilibrium curve is reported in [17]. A nonlinear continuous-time dynamical chaotic system based on two circles of equilibrium points and its multistability is discussed in [18]. As mentioned above, all these chaotic systems are utilized for image and video encryption only, and no general-purpose chaotic dynamic system-based cryptosystem exists.

Similarly, [19] showcased a new discrete chaotic system merging multiple linear systems, whereas the Hénon map and the multistable hyper-chaotic system are designed in [20] and [21]. A new modified scheme using a skew tent map to overcome drawbacks in conventional PRN sequences with strong cryptographic properties is introduced in [22]. In [23] chaos-based encryption scheme for medical images is developed utilizing SHA-256 hash function, block-based permutation, pixel-based substitution, DNA encoding complementing and decoding, and bit-level diffusion. Reference [24] presented a PRN sequence generator algorithm based on a combination of the three coordinates of the Chen chaotic orbits. A fast RGB image encryption algorithm based on total plain image characteristics and optimized pseudorandom sequence from a 1D logistic map is present in [25]. The work presented in [26] is an image encryption algorithm derived from a 2D chaotic system based on orthogonal Latin squares. Lastly, [27] 2D-PPCS is constructed from nonlinear polynomials, and a modular chaotification of the phase plane yields 2D chaotic maps with robust chaos and desired dynamic properties.

With the advent of smart systems incorporating Internet-of-Things (IoT) enabled devices, data (or information) security has become of utmost importance. Internet of Everything (IoE) is the future and demands a system with very high security, energy efficiency, and reliability. Therefore, it has become imperative to design a general-purpose low-power cryptosystem, which can be used in various applications ranging from IoT-based smart systems to high throughput, low latency systems. In order to address this issue, we endeavor to design a robust and reliable cryptosystem. Since chaotic dynamic systems can produce secure random sequences and present good promise while designing a robust and secure encryption algorithm, we aim to design a cryptosystem based on chaos in the proposed work. Note that the proposed algorithm should satisfy several criteria to establish itself as cryptographically valid, which are given in [28] and [29].

The significant contribution of the research work is summarized below. This work introduces a nonlinear dynamic system derived from the chaotic behavior of a physical triple pendulum configuration. The characteristics of the proposed chaotic system are validated by performing bifurcation, Lyapunov test, NIST benchmark tests, etc., which showcase the applicability of the triple pendulum-based chaotic maps in various cryptography applications, such as pseudorandom number generation (PRNG) and Baptista-type symmetric encryption-decryption scheme development, etc. The proposed work also delineates the methodology to choose the suitable range of initial values for random number seed or a stronger key value for encryption. Further, a general-purpose symmetric key encryption scheme is developed employing the proposed chaotic map.

The organization of the rest of the manuscript is as follows. Section II presents mathematical formulations of chaotic double and triple pendulum dynamic systems. The Lyapunov exponents, bifurcation diagrams, parametric and initial values sensitivity, ergodicity, collision test, etc., are presented in Section III to validate the proposed triple pendulum-based dynamic system's chaotic behavior. Section IV discusses the application of generated chaotic maps as PRNG and its validation employing the NIST test suite. Later, symmetric key generation, encryption, and decryption algorithm are depicted in Section V along with cryptanalysis. A detailed hardware implementation and realization methodology on FPGA and ASIC are discussed in section VI. Finally, the conclusion is drawn in Section VII.

II. MATHEMATICAL MODELING OF TRIPLE PENDULUM SYSTEM

The formulation of a nonlinear triple pendulum chaotic system is similar to a double-pendulum system. For simplicity and to limit the usage of the variables in mathematical formulation, the construction of differential equations of a double pendulum system is demonstrated below. Figure 1 exhibits a basic double pendulum pictorially.

The basic position relations from the above figure are given in equations 1 and 2.

$$x_1 = L_1 sin\theta_1;\ x_2 = L_1 sin\theta_1 + L_2 sin\theta_2; \tag{1}$$

$$y_1 = -L_1 cos\theta_1;\ y_2 = -L_1 cos\theta_1 - L_2 cos\theta_2; \tag{2}$$

The expressions of potential (P) and kinetic (K) energies, as mentioned in equations 3 and 4, are obtained by analyzing position relations.

$$P = -m_1 g L_1 cos\theta_1 - m_2 g(L_1 cos\theta_1 + L_2 cos\theta_2) \tag{3}$$

$$K = \frac{m_1 \dot{\theta_1}^2 L_1^2 + m_2(\dot{\theta_1}^2 L_1^2 + \dot{\theta_2}^2 L_2^2 + 2\dot{\theta_1} L_1 \dot{\theta_2} L_2 cos(\theta_1 - \theta_2))}{2} \tag{4}$$

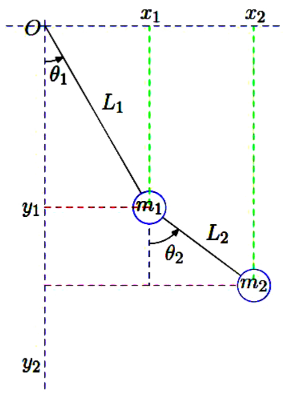


FIGURE 1. Basic double pendulum setup.

Later, Euler-Lagrange condition $\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}}) - \frac{\partial L}{\partial \theta} = 0$ is applied, where $L = K - P$ and can be represented in equation 5.

$$
\begin{aligned}
L = &\frac{1}{2}(m_1 + m_2)L_1^2\dot{\theta_1}^2 + \frac{1}{2}m_2L_2^2\dot{\theta_2}^2 \\
&+ m_2L_1L_2\dot{\theta_1}\dot{\theta_2}cos(\theta_1 + \theta_2) \\
&+ (m_1 + m_2)gL_1cos\theta_1 + m_2L_2gcos\theta_2 \qquad (5)
\end{aligned}
$$

Thereafter, substituting L into above mentioned Euler-Lagrange equation and solving the resultant equation for θ_1 and θ_2, two second-order differential equations are obtained, which are expressed below.

$$
\begin{aligned}
\ddot{\theta_1} = &[-m_2L_1\dot{\theta_1}^2 sin(\theta_1 - \theta_2)cos(\theta_1 - \theta_2) + gm_2sin\theta_2cos(\theta_1 \\
&- \theta_2) - m_2L_2\dot{\theta_2}^2 sin(\theta_1 - \theta_2) - (m_1 \\
&+ m_2)gsin\theta_1]/[L_1(m_1 + m_2) - m_2L_1cos^2(\theta_1 - \theta_2)] \\
\ddot{\theta_2} = &[m_2L_2\dot{\theta_2}^2 sin(\theta_1 - \theta_2)cos(\theta_1 - \theta_2) + gsin\theta_1cos(\theta_1 \\
&- \theta_2)(m_1 + m_2) + L_1\dot{\theta_1}^2 sin(\theta_1 - \theta_2)(m_1 + m_2) \\
&- g\ sin\theta_2(m_1+m_2)]/[L_2(m_1+m_2)-m_2L_2cos^2(\theta_1-\theta_2)]
\end{aligned}
$$

A. FORMULATION OF COMPOUND TRIPLE PENDULUM

As stated above, the steps of the mathematical formulation of a triple pendulum system are similar to the double pendulum system. Here, an abstract representation of the mathematical formulation of the triple pendulum system is presented. Since in practical pendulum system bars have mass, they can be modeled as a compound pendulum. Damping factors are also included in the proposed model to introduce a higher degree of chaos and nonlinearity in the system. We considered m_i, l_i, I_i, and k_i as the mass, length, moment of inertia, and damping coefficient of the bar rotating about its upper joint, respectively, where i denotes the i_{th}-bar. The position and velocity of the bars are defined by six system state variables, $\theta_1, \theta_2, \theta_3$, $\dot{\theta_1}$, $\dot{\theta_2}$, $\dot{\theta_3}$. The relevant governing equations to deduce these variable are given below (stated in equations 6 to 13), where v_i, TKE_i, RKE_i, GPE_i, $\mathcal{L}$ and $\mathcal{D}$ are the magnitude of the velocity of each bar, translational kinetic energy, rotational kinetic energy, gravitational potential energy, Lagrangian and Rayleigh Dissipation Function. Its simplified diagram is presented in Figure 2.

$$y_1 = \frac{l_1}{2}cos\theta_1;\ y_2 = l_1cos\theta_1 + \frac{l_2}{2}cos\theta_2 \qquad (6)$$

$$y_3 = l_1cos\theta_1 + l_2cos\theta_2 + \frac{l_3}{2}cos\theta_3 \qquad (7)$$

$$x_1 = \frac{l_1}{2}sin\theta_1;\ x_2 = l_1sin\theta_1 + \frac{l_2}{2}sin\theta_2 \qquad (8)$$

$$x_3 = l_1sin\theta_1 + l_2sin\theta_2 + \frac{l_3}{2}sin\theta_3 \qquad (9)$$

$$v_i = \sqrt{\dot{x_i}^2 + \dot{y_i}^2};\ TKE_i = \frac{1}{2}m_iv_i^2 \qquad (10)$$

$$RKE_i = \frac{1}{2}I_i\dot{\theta_i};\ GPE_i = m_igy_i \qquad (11)$$

$$\mathcal{L} = T - V = \sum_{i=1}^{3} TKE_i + RKE_i - GPE_i \qquad (12)$$

$$\frac{d}{dt}(\frac{d\mathcal{L}}{\dot{q_i}}) = \frac{d\mathcal{L}}{dq_i} - \frac{d\mathcal{D}}{\dot{q_i}};\ \mathcal{D} = \frac{1}{2}(\sum_{i=1}^{3} k_i\dot{\theta_i}^2) \qquad (13)$$

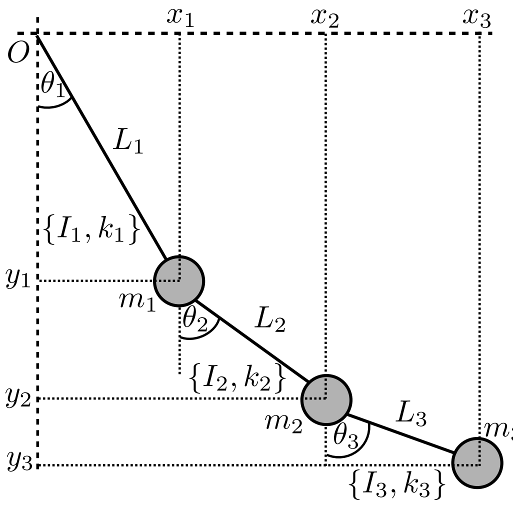
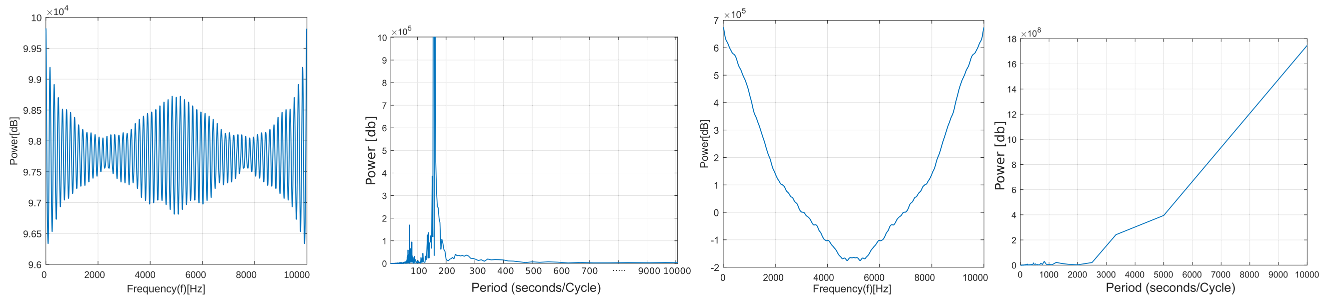


FIGURE 2. Basic Compound Triple Pendulum Setup.

A compound triple-pendulum (Figure 2) is modeled with the help of relations mentioned above and employs ordinary differential equations, describing its chaotic motions. The parameters and initial conditions are constant, which is the part of an individual key. The numerical values corresponding to the angular positions of the bars are obtained within a certain duration as per the predefined precision parameter. This process generates a chaotic map, which is utilized to map plaintext to the ciphertext during encryption operation. After analyzing above mentioned analytic equations, $\ddot{\theta_1}$, $\ddot{\theta_2}$ and $\ddot{\theta_3}$ are obtained, and are given in Appendix A, B and C, respectively. The equations of $\ddot{\theta_1}$, $\ddot{\theta_2}$ and $\ddot{\theta_3}$ can be derived using equation 14 by iterating $t = 0 \rightarrow N$ and are stored after phase wrapped in the range of $[-\pi, \pi]$ in a vector y,



(a) Circular Auto-Correlation for a periodic θ_3

(b) Periodicity for a periodic θ_3

(c) Circular Auto-Correlation for a nonperiodic θ_3

(d) Periodicity for a nonperiodic θ_3

FIGURE 3. Circular auto-correlation and periodicity test for two trajectories of θ_3.

where $N = \frac{T}{\Delta t}$ and expressed as the equation 15.

$$\begin{aligned} \ddot{\theta}_i^{(t+1)} &= f_i(\theta_1^{(t)}, \theta_2^{(t)}, \theta_3^{(t)}, \dot{\theta}_1^{(t)}, \dot{\theta}_2^{(t)}, \dot{\theta}_3^{(t)}) \\ \dot{\theta}_i^{(t+1)} &\leftarrow \dot{\theta}_i^{(t)} + \ddot{\theta}_i^{(t+1)} \Delta t \\ \theta_i^{(t+1)} &\leftarrow \theta_i^{(t)} + \dot{\theta}_i^{(t+1)} \Delta t \end{aligned} \tag{14}$$

$$\hat{y}^i = \{\hat{\theta}_i^{(0)}, \hat{\theta}_i^{(1)}, \ldots, \hat{\theta}_i^{(N-1)}\}; \; \hat{\theta}_i^{(t)} = \theta_i^{(t)} - \lfloor \frac{\theta_i^{(t)}}{2\pi} \rfloor \times 2\pi \tag{15}$$

It is to mention that state variables $\ddot{\theta}_1$, $\ddot{\theta}_2$ and $\ddot{\theta}_3$ are used to generate one-to-one function F, such that,

$$\hat{y} = F(\hat{y}^1, \hat{y}^2, \hat{y}^3).$$

The chaotic numeric map is obtained by iterating through all possible parameters emphasizing nonperiodicity in the selection; subsequently, a set of keys are generated.

TABLE 1. Key parameters and initial conditions.

Parameter/ Condition	Value	Unit
m_1, m_2, m_2	$0.2944, 0.1756, 0.0947$	kg
l_1, l_2, l_3	$0.508, 0.254, 0.127$	m
I_1, I_2, I_3	$(9.526, 1.625, 1.848) \times 10^{-3}$,	kg.m^2
k_1, k_2, k_3	$(5, 0, 0.8) \times 10^{-3}$	N.m.s/rad
$\theta_1, \theta_2, \theta_3$	$-0.4603, -1.2051, -1.5165$	rad
$\dot{\theta_n}$	0	rad/s

III. VALIDATION OF CHAOTIC GENERATOR

The proposed triple pendulum system is implemented using MATLAB to validate the mathematical model's correctness presented in the previous section. Employing random initial conditions given in the Table 1, the phase wrapped angular displacement profiles of $\ddot{\theta}_1$, $\ddot{\theta}_2$ and $\ddot{\theta}_3$ are generated. It can be observed that the proposed system is chaotic because of the aperiodicity of angular displacements.

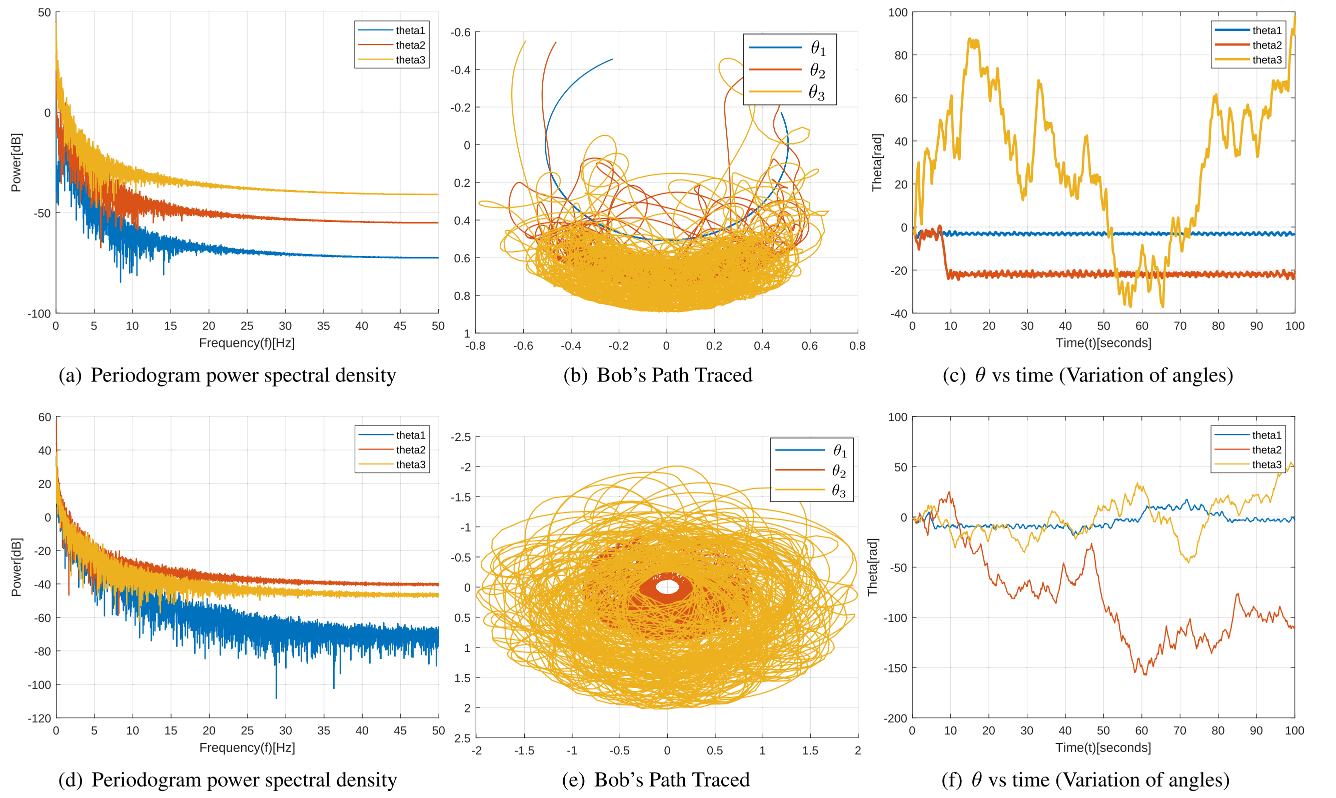
A. AUTO-CORRELATION & FISHER G-STATISTIC TEST

Figure 3 presents a circular auto-correlation, estimated power of θ_3 with respect to its periodicity for periodic and nonperiodic trajectories. The periodic nature of circular auto-correlation signifies the general periodic nature of the trajectories and vice-versa. It can be observed from Figure 3(a) and Figure 3(b) that the power is concentrated around a few frequencies for periodic signals, whereas power is distributed in the whole domain for aperiodic signals, as shown in Figure 3(c) and Figure 3(d), respectively.

Fisher's G-statistic tests are also performed to visualize further similar properties of periodic and non-periodic trajectories, depicted in Figure 4. These tests, illustrated in Figure 4, are periodograms, bob's path traced, and the time variation of angles. The periodogram is the power spectral density at each frequency of a bob. Bob's path traced during chaotic oscillations presents a physical path traced in a 2D plane. The time variation of angles exhibits variations in (θ_1, θ_2, θ_3) with respect to time. The numerical values obtained from the periodicity test, circular auto-correlation, and other analyses differentiate parameters and initial values, leading to the periodic and aperiodic nature of the motion. Thus, iterating through all possible parameters emphasizing nonperiodicity in the desired chaotic map is generated. Lyapunov exponents and bifurcation tests, sensitivity to parametric and initial values, ergodicity, and collision tests are performed to prove that generated chaotic maps are cryptographically viable.

B. LYAPUNOV EXPONENTS TEST

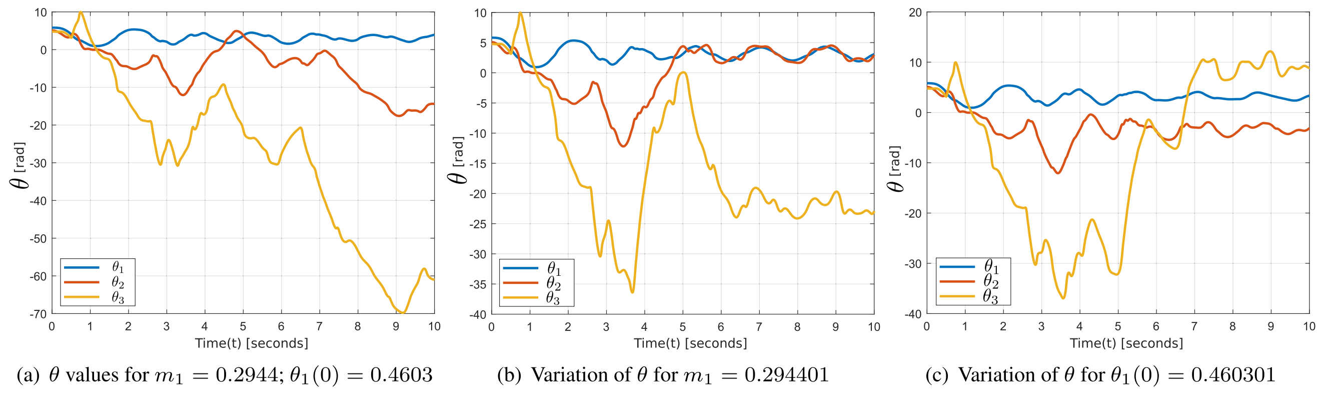
Lyapunov exponents for all the initial parameters are presented in Figures 6 to verify the chaotic nature of our proposed triple pendulum-based nonlinear chaotic map. The chaotic nature of a dynamic system is estimated by the presence of positive exponents in the Lyapunov test, which implies exponential divergence of the system. It is mentioned that chaotic systems do not synchronize with any other physical system in reality. However, their synchronization conditions can be formulated mathematically if all the initial conditions and parameters resemble the actual physical systems, which is only possible in an ideal scenario. Thus, two similar chaotic systems produce significantly different outputs when initial conditions and parameters are marginally different. For estimating Lyapunov exponents [30] of an oscillating chaotic system, a $3D$ Lorenz equation 16 is formulated and compared with different parameters of our



(a) Periodogram power spectral density (b) Bob's Path Traced (c) θ vs time (Variation of angles)

(d) Periodogram power spectral density (e) Bob's Path Traced (f) θ vs time (Variation of angles)

FIGURE 4. Fisher's G-statistic test with a periodic trajectory (a,b,c) and with a nonperiodic trajectory (d,e,f).



(a) θ values for $m_1 = 0.2944$; $\theta_1(0) = 0.4603$ (b) Variation of θ for $m_1 = 0.294401$ (c) Variation of θ for $\theta_1(0) = 0.460301$

FIGURE 5. Sensitivity to parameter value m_1 and θ_1 with $\triangle m = 10^{-6}$ and $\triangle \theta_1 = 10^{-6}$.

proposed dynamic system. It can be observed from Figure 6 that most of the Lyapunov exponents are greater than zero; therefore, the proposed triple pendulum-based dynamic system fulfills Lyapunov criteria and can be considered a suitable chaotic system to be utilized in the design of cryptosystems.

$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= x(\rho - z) - y\\ \dot{z} &= xy - \beta z\end{aligned} \tag{16}$$

C. ERGODICITY AND SENSITIVITY TESTS FOR CHAOS

As we know, these chaotic maps are used to generate keys for encryption and decryption schemes. Therefore, the statistical tests can ensure whether the keys generated in our proposed scheme are ideally indistinguishable from any true random sequence or not. Various standard statistical tests, which are conducted to support our claims, are described below. These tests include ergodicity, sensitivity to parametric values, and sensitivity to initial conditions to be as unpredictable as possible. For completeness, these essential tests are described below.

TABLE 2. Comparison of the Proposed Scheme with Existing Works on NIST *SP*800 – 22 Test Suite.

NIST Tests	p-value $(0.01 \leq p\text{-}value < 1)$					
	Hua et al [31]	Srisubha et al [32]	LCG [33]	BluXor [34]	MPCG [35]	Proposed
Frequency	0.364146	0.49	0.590351	0.506496	0.106628	0.394808
Block Frequency	0.427274	0.33	0.492735	0.484548	0.04156	0.783942
Cumulative Sums	0.670408	0.59	0.525255	0.552932	0.067332	0.197764
Runs	0.213309	0.45	0.572252	0.489091	0.287543	0.335584
Longest Run of 1's	0.204076	0.53	0.284067	0.440516	0.057843	0.284067
Rank	0.819544	0.39	0.539487	0.443327	0.15037	0.197764
DFT/Spectral	0.407091	0.41	0.344484	0.464124	0.41735	0.769057
Non-periodic T.M.	0.472933	0.49	0.493525	0.796262	0.01231	0.278074
Overlapping T.M.	0.602458	0.61	0.494436	0.553845	0.31532	0.554566
Universal Statistical	0.772760	0.29	0.532033	0.454763	0.24316	0.477911
Approx. Entropy	0.287306	0.60	-	-	-	0.365177
Random Excursions	0.442347	0.38	0.052382	0.28161	0.41754	0.513954
Random Excursions Variant	0.356540	0.35	0.036553	0.398311	0.38923	0.242148
Serial	0.175321	0.57	0.489956	0.498307	0.026517	0.156848
Linear Complexity	0.264458	0.58	0.274623	0.543714	0.21839	0.129930

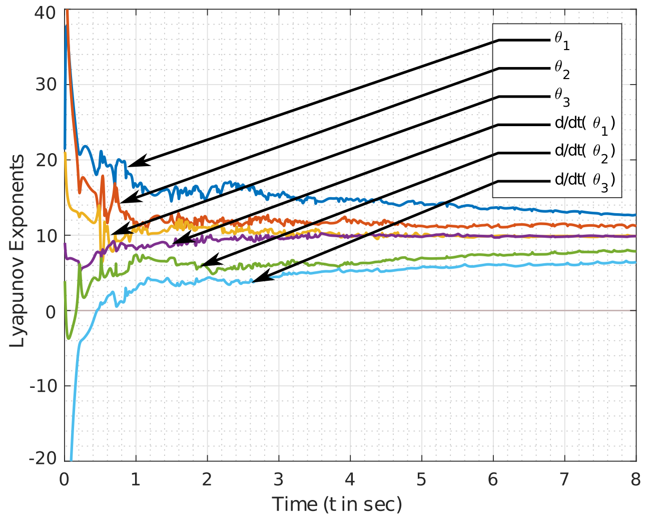


FIGURE 6. Dynamics of Lyapunov exponents upto 8 sec.

1) **Sensitivity to Parametric values:** A small perturbation in one of the system parameters, for example, m_1, is enough to create two exponential diverging trajectories starting at the same initial point. This is reflected in Figure 5(a) and Figure 5(b) for the parameters given in Table 1.
2) **Sensitivity to Initial Condition:** Two trajectories starting at two different but arbitrarily close initial points denoted by θ_1 diverge from each other at an exponential rate. This is illustrated in Figure 5(a) and Figure 5(c) for the initial conditions given in Table 1.
3) **Ergodicity:** Ergodicity is a chaotic system's phase of space exploration uniformly or randomly. Here, phase space is defined as a relationship between angular velocity, ω, and the angle of the pendulum's bob, ϕ. The triple pendulum system exhibits ergodic nature because phase space is sparse and is not cluttered in any specific region. The phase space of the bottom bob of the proposed triple pendulum system illustrated in Figure 7 under

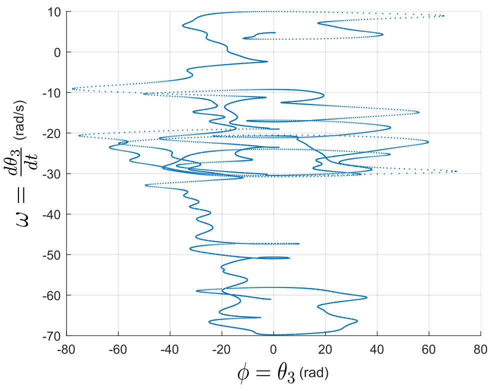


FIGURE 7. Phase space plot for ergodicity test for θ_3.

conditions given in Table 1 exhibits a trajectory that satisfies ergodicity criteria. This leads to the conclusion that a system under specific conditions can be considered ergodic.

The proposed triple pendulum based cryptosystem holds above mentioned properties, which are clearly evident in Figure 5 and Figure 7. It is to mention that the map generated using the proposed compound triple pendulum model possesses essential chaotic properties to be employed in a general cryptosystem.

D. BIFURCATION DIAGRAMS

The bifurcation tests for all the key parameters are presented in Figure 8, which validate the chaotic nature of our proposed dynamic system. Bifurcation diagrams for the proposed nonlinear chaotic map are generated for all the independent parameters m_1, m_2, m_3, l_1, l_2, l_3, k_1, k_2 and k_3 for θ_1, $\dot{\theta}_1$, θ_2, $\dot{\theta}_2$, θ_3 and $\dot{\theta}_3$ each. It is to mention that the generally chaotic nature of systems depends on a single parameter, and its bifurcation diagram is produced by changing the value of that particular parameter. Our proposed chaotic dynamic system has 12 independent parameters constituting a 12 dimensional chaotic map exhibiting highly

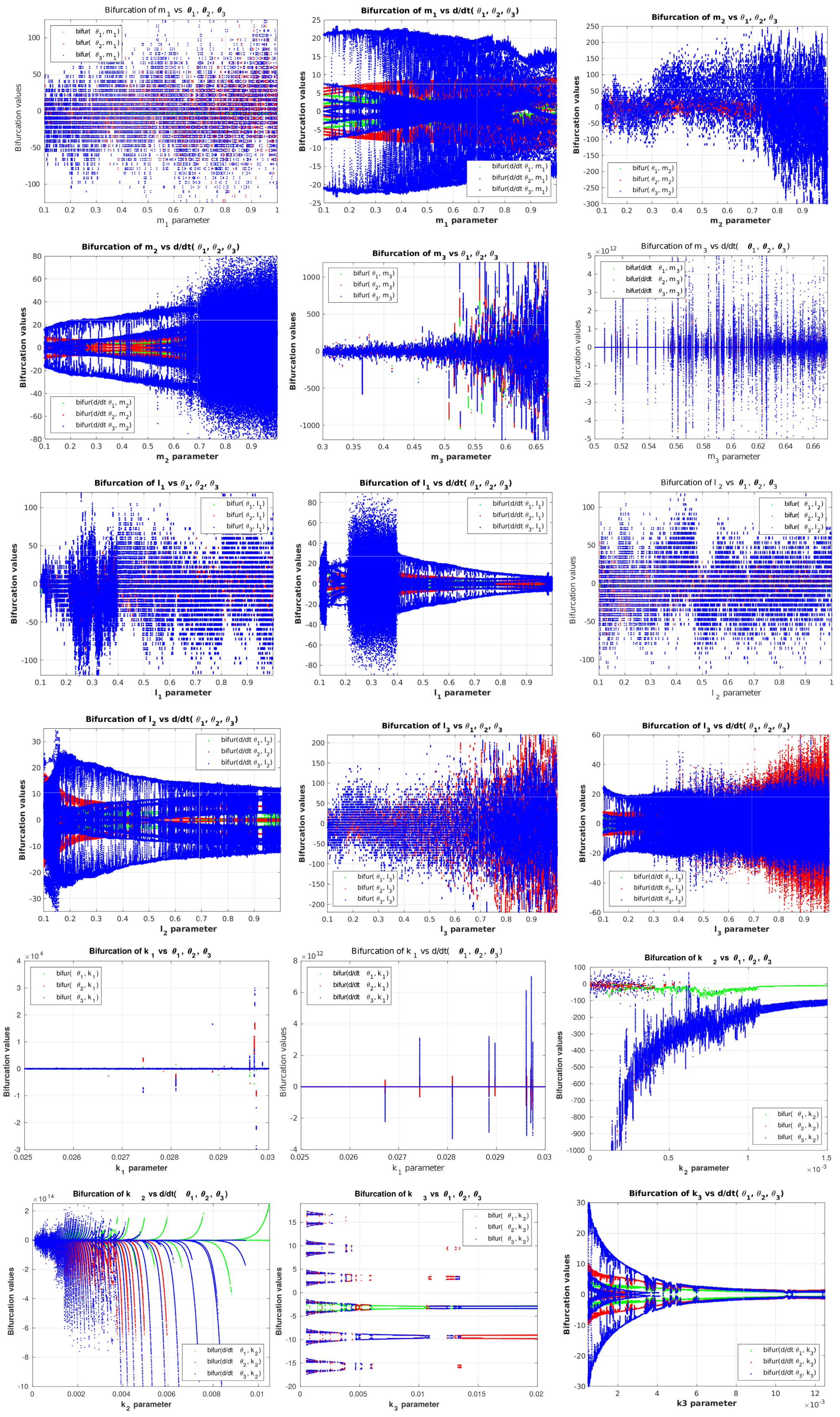


FIGURE 8. Bifurcation diagrams of the independent key parameters for $\{\theta_1, \dot{\theta_1}$ in green$\}$, $\{\theta_2, \dot{\theta_2}$ in red$\}$ and $\{\theta_3, \dot{\theta_3}$ in blue$\}$.

chaotic behavior. All the bifurcation diagrams illustrated in Figure 8 are generated by solving $\ddot{\theta}$ using initial conditions of Table 1. It is to mention that in Figure 8, θ_1 and $\dot{\theta}_1$, θ_2 and $\dot{\theta}_2$, and θ_3 and $\dot{\theta}_3$ are indicated with green, red and blue colors.

The chaotic behavior of the proposed nonlinear compound triple pendulum is validated from the above analytical and statistical tests. Therefore, the proposed chaotic map can be utilized to design various cryptography counterparts, such as pseudo-random number generators, signature generators, lightweight symmetric cryptography schemes, etc. The following sections present critical cryptography applications, a chaos-based pseudo-random number generator, and a lightweight symmetric cryptosystem.

IV. CHAOS BASED PRNG

In order to test the randomness of a chaotic map generated in the proposed method, various statistical tests are performed. These statistical tests are generally employed to validate randomness in pseudo-random-number generators. NIST suite [36] is employed to test the randomness properties of a chaotic map. The information about tests performed on the chaotic map generated during analysis and the corresponding results obtained are presented in Table 2. The proposed chaotic map is tested with existing PRNGs employing standard NIST SP $800-22$ suite [36] to evaluate and compare its randomness quality, tabulated in Table 2. It is to mention that a particular PRNG is said to be passed a test if $0.01 \leq p\text{-}value < 1$.

In the NIST test, our proposed chaotic generator is compared with five recently published methods described in [31], [32], [33], [34], and [35]. In [31], a sine chaotification model is employed to generate a 1-D chaotic map, whereas [32] presents physically unclonable functions (PUF) based chaotic maps. The [31] and [32] are incorporated in the evaluation so that our proposed method is compared fairly with widely used chaotic schemes. On the other hand, the proposed chaotic map-based PRNG is compared with Linear Congruential Generators [33], and also with two newly developed variants [34], [35]. From Table 2, it can be observed that our proposed chaotic map generator passes all the tests successfully. This is largely due to the nonlinear dynamics of the chaotic map having 12 independent parameters and random initial conditions. The outcome of these tests indicates that the proposed chaotic map is indistinguishable with respect to any other random sequence generators. Thus, the proposed triple pendulum-based chaos generator can be utilized in a wide range of security applications due to its cryptographically secure random number generation capability. The following sections present a detailed description of a new encryption scheme, which is an important application of a nonlinear triple pendulum-based chaotic map.

V. CHAOS BASED SYMMETRIC ENCRYPTION SCHEME

This chapter focuses on implementing a chaos-based encryption-decryption technique on Field Programmable Gate Array (FPGA) and as an Application Specific Integrated Circuit (ASIC). We aim to design a system that should be less resource-intense, computationally efficient, and provide as strong security as possible. The necessary steps to be performed, along with design and implementation, are given below for achieving this goal.

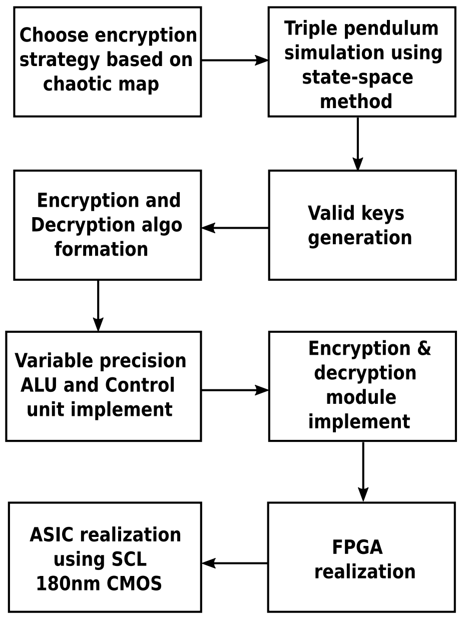

Choose encryption strategy based on chaotic map
Triple pendulum simulation using state-space method
Encryption and Decryption algo formation
Valid keys generation
Variable precision ALU and Control unit implement
Encryption & decryption module implement
ASIC realization using SCL 180nm CMOS
FPGA realization

FIGURE 9. Design steps of the proposed method.

A. STEP-WISE DESIGN METHODOLOGY

Figure 9 depicts design and implementation steps of the proposed cryptosystem. The motivation behind employing the proposed algorithms for encryption and decryption is based on Baptista-type encryption-decryption scheme. It is described using an interval-partitioning of chaotic orbits of a $1D$ logistic map, which enables the construction of swift and secure encryption-decryption schemes because of its less complex structure. The main objective of our scheme is first to map text characters to numerical values, then algorithms are applied to encrypt the message. Decryption is performed by iterating a chaotic map, then corresponding symbols for numerical values are obtained by inverting the process. Since the input to the system is a plain message, the system parameter(s) and the initial conditions of a dynamical system are assumed to be part of the secret key. After analyzing a triple-pendulum system, a valid set of keys for encryption and decryption is generated. The entire mechanical model is implemented on an FPGA along with a custom ALU and a control unit adhering to the specifications of our proposed system. After successfully validating the design on an FPGA, it is realized using Semi-Conductor Laboratory (SCL) 180-nm Bulk CMOS technology.

B. THE PROPOSED ENCRYPTION SCHEME

A detailed description of the proposed methodology, which includes key generation, encryption, and decryption algorithm with a suitable example, is presented here.

1) KEY GENERATION

The key, K, of the proposed symmetric cryptosystem includes parameters, such as initial conditions, time duration, time step, the minimum value of $\hat{y}$, and ϵ. The ϵ is a mapping interval and can be expressed as $\epsilon = \frac{(\hat{y}_{max} - \hat{y}_{min})}{N_c}$. There are total 21 variables constituting basic key structure shown in equation 17.

$$K = \{\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}, \dot{\theta}_1^{(0)}, \dot{\theta}_2^{(0)}, \dot{\theta}_3^{(0)}, m_1, m_2, m_3, l_1, l_2, l_3, I_1, I_2, I_3, k_1, k_2, k_3, g, \hat{y}_{min}, \hat{y}_{min}/\epsilon\} \tag{17}$$

TABLE 3. Key size for various encryption algorithms [37].

Algorithm	Symmetric Key size (bits)	Asymmetric Key size (bits)	Key Memory
Lattice Multivariate	14000	6595	$<$ 1KB
NTRU	6743	6130	$<$ 1KB
Rainbow	740000	991000	125KB
Hash Signature	36000	36000	5KB
Goppa McEliece	92027	8373911	1MB
Quasi-cyclic MDPC McEliece	4384	65542	8KB
SIDH	6144	6144	$<$ 1KB
Proposed (single precision)	672	–	$<$ 1KB
Proposed (double precision)	1344	–	$<$ 1KB

Table 3 compares key size and memory for various security algorithms. It can be observed that the key size generated by our proposed method is the least among other algorithms. For specific parameters or initial conditions, the motion of the bobs of a triple-pendulum exhibits periodic nature after a certain time period. Hence, it becomes imperative to eliminate those parameters or initial conditions for which motion is periodic to preserve the chaotic behavior of the system. Therefore, *Fisher's g-statistic test* [38] is employed to extract the prominent period of the chaotic motion (or signal) using statistical analysis of its spectrum. This test exploits periodic components of the signal derived from its periodogram, which is used to identify a time series's hidden periods (or frequencies). This helps identify dominant periodic behavior in a series. The periodogram indicates the relative occurrence of possible frequencies repeatedly oscillating between intervals. It estimates *Power Spectral Density* of the signal $x_L(n)$ of length L, which is defined below. Here, F_s is the sampling frequency.

$$P_{xx}(f) = \frac{1}{LF_s} \left| \sum_{n=0}^{L-1} x_L(n) e^{-j2\pi fn/F_s} \right|^2 \tag{18}$$

The computation of $P_{xx}(f)$ can be performed only at a finite number of discrete frequency points given by the following relation.

$$f_k = \frac{kF_s}{N}, \; k = 0, 1, \ldots, (N-1) \tag{19}$$

Fisher's $g-$statistic is defined as the ratio of the maximum magnitude of the periodogram ($P_{xx}(f_k)$ or $I(\omega_k)$) to the sum of all the magnitudes of a periodogram as shown below.

$$g = \frac{\max_k I(\omega_k)}{\sum_{k=1}^{[N/2]} I(\omega_k)} \tag{20}$$

Here, $I(\omega_k)$ is the magnitude of a periodogram, and using g, dominant periods of a signal can be estimated. Thus, after removing the range of the parameters in which a signal exhibits possible periodic behavior, the remaining parameters can be utilized to span key space. Further, based on the bifurcation diagrams, the strongest key may be chosen for encryption and decryption steps. Since bifurcation is a resource-intensive process, there is a tradeoff between power and choosing a stronger key for an application. In the following subsection, encryption and decryption methods are explained in detail.

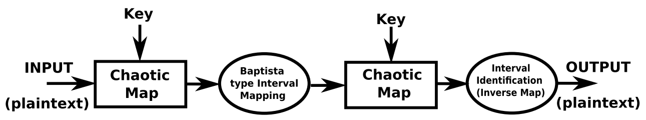

Key
Key
INPUT
(plaintext)
Chaotic Map
Baptista type Interval Mapping
Chaotic Map
Interval Identification (Inverse Map)
OUTPUT
(plaintext)

FIGURE 10. Encryption-Decryption Strategy.

2) ENCRYPTION - DECRYPTION

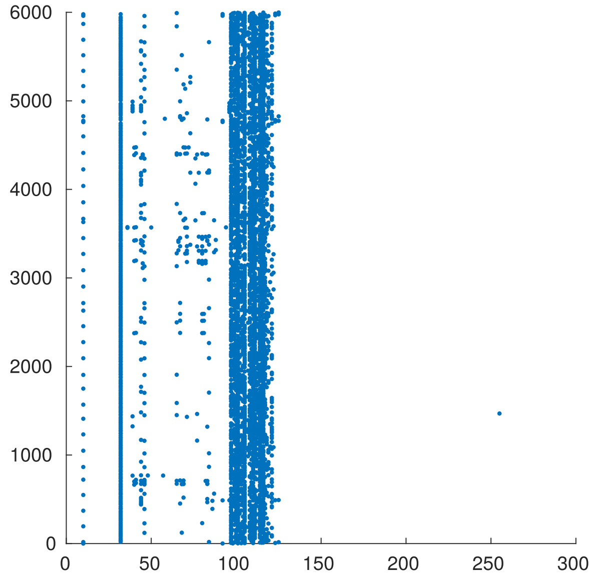
The approach discussed here is to convert plaintext of M characters into an ASCII format and map it to the intervals formulated using state variables of the proposed triple-pendulum system as shown in Figure 10. The initial conditions and control parameters of the differential equations constitute a part of the private key **K**. The differential equations are numerically integrated to get state variables (θ_1, θ_2 and θ_3) followed by phase wrapping in the range of $[0, 2\pi]$ or $[-\pi, \pi]$. Later, employing Baptista-type method, an entire range ($[\hat{y}_{min}, \hat{y}_{max}]$) of chaotic trajectory $\hat{y}$ is partitioned into a number of intervals equal to the number of characters (N_c). Each character in the plaintext **P** is first mapped to a specific interval (k) and is stamped with a time point (i) randomly selected from this interval. A permutation map $\pi(k)$ is employed to determine interval identifiers for each character's ASCII value. This process transforms plaintext into an intermediate value, masked using f_k. Here, f_k is produced by applying variable modular operation (depends on the iteration index) to the private key and converting it to a half-precision floating-point expression. Simultaneously, an intermediate encrypted binary number $D(\pi(\hat{p}), j)$ is rotated left by m bits stored as t, where m is an input plaintext. Finally, the ciphertext **C** is obtained by XORing t and f_k. The proposed encryption scheme is depicted in Algorithm 1. Here, η is a constant and **C** is a ciphertext.

Algorithm 1 Proposed Encryption Algorithm

```
Input: $\hat{y}$, $\mathbf{P}$, $\mathbf{K}$, $N_c$, $\eta$
Output: $\mathbf{C}$
Initialize $D = \{\}$, k = size(K)
Find $\hat{y}_{max} = \max_{1 \le i \le N}(\hat{y}_i)$, $\hat{y}_{min} = \min_{1 \le i \le N}(\hat{y}_i)$
Calculate $\epsilon = \frac{(\hat{y}_{max} - \hat{y}_{min})}{N_c}$
For each $\hat{y}_i$ in $\hat{y}$
Calculate $d = \lfloor \frac{\hat{y}_i - \hat{y}_{min}}{\epsilon} \rfloor$
$D(\pi(d)) \leftarrow D(\pi(d)) \cup \{i\}$
For $m = 1$ to $M$
Initialize $j = 0, p = P(m)$
Calculate $\hat{p} = ASCII(p)$, $l = size(D(\pi(\hat{p})))$
Generate random number r, $0 \le r \le 1$
While $r \le \eta$, do $j \leftarrow (j+1)\%l$
Calculate $f_k = half(K(m\%k))$, where $half(n)$ converts $n$ to Half-Precision Floating Point format
$t = ROL(D(\pi(\hat{p}), j), m)$, where $ROL(n, b)$ rotates $n$ left by $b$-bits
$c = t \oplus f_k$
$C(m) = c$
Return $\mathbf{C}$
```

Algorithm 2 Proposed Decryption Algorithm

```
Input: $\hat{y}$, $\mathbf{C}$, $\mathbf{K}$
Output: $\tilde{P}$
Initialize $k = size(K)$
Find $\hat{y}_{max} = \max_{1 \le i \le N}(\hat{y}_i)$; $\hat{y}_{min} = \min_{1 \le i \le N}(\hat{y}_i)$
Calculate $\epsilon = \frac{(\hat{y}_{max} - \hat{y}_{min})}{N_c}$
For $m = 1$ to $M$
Initialize $c = C(m)$
Calculate $f_k = half(K(m\%k))$, where $half(n)$ converts $n$ to Half-Precision Floating Point format
$t = c \oplus f_k$
$\hat{C}$ = ROR($t$, $m$), where ROR($n$, $b$) rotates $n$ right by $b$-bits
Calculate $\hat{k} = \lfloor \frac{\hat{y}_c - \hat{y}_{min}}{\epsilon} \rfloor$
Compute $k = \pi^{-1}(\hat{k})$
$\tilde{P}(m) \leftarrow CHAR(k)$
Return $\tilde{P}$
```


(a) Initial 6000 ASCII characters distribution of plaintext

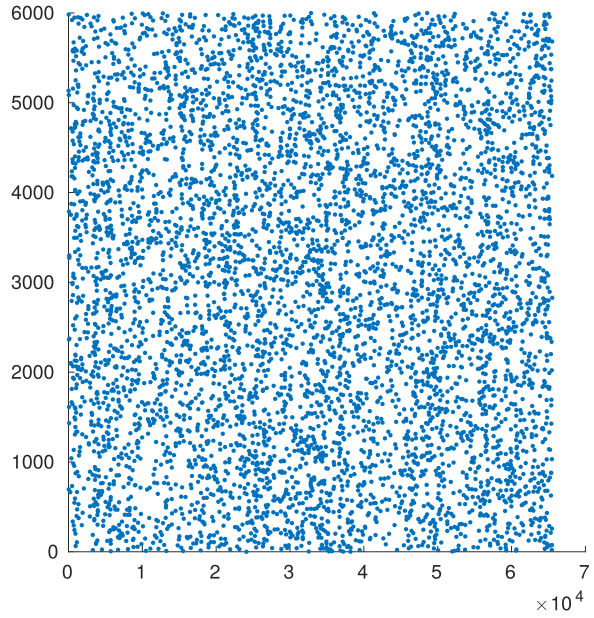

(b) Distribution of ciphertext

FIGURE 11. Text data encryption example for 100KB plaintext based on the key value listed in Table 1.

In the decryption module, the interval in which the character of the ciphertext lies is computed using the same symmetric key. Here **C** is first unmasked by performing the reverse rotation and XORing with f_k after generating it following the similar steps discussed above during encryption. Thereafter, by applying an inverse permutation map ($\pi^{-1}(k)$), corresponding indices are converted back to plaintext ($\tilde{P}$). The message can be decoded by converting $\tilde{P}$ into characters. The proposed decryption scheme is presented in Algorithm 2 for the ready reference.

Besides the plaintext encryption, the proposed method can also be applied to images with RGB values ranging from 0 to 255. The parameter N_c in Algorithm 1 can be set to 256 and a serialized image is fed as an input for the image encryption. This algorithm can also be extended for video encryption by treating each frame as an independent image. For example, the proposed method is applied for image encryption, and the outcome of this process is stated below. Since the proposed method is fast and secure (refer to Section VI), it is suitable to keep live video feeds secure from any adversary attacks.

The proposed encryption and decryption algorithms are demonstrated on three sets of images. The original and encrypted images are illustrated with their corresponding histograms presented in Figure 12. Parameters of the key used for encryption and decryption in both types of applications, text and image, are $m_1 = 0.2944$, $m_2 = 0.8756$, $m_3 = 0.0947$, $l_1 = 0.508$, $l_2 = 0.254$, $l_3 = 0.127$, $k_1 = 0.005$, $k_2 = 0$, $k_3 = 0.0008$, $I_1 = 9.526 \times 10^{-3}$, $I_2 = 1.625 \times 10^{-3}$, $I_3 = 1.848 \times 10^{-4}$, respectively, with $g = 9.81$. Selection of these parameters for a key is accomplished using the bifurcation diagram shown in Figure 8. The following subsection discusses analytical aspects of the security of our proposed chaos-based cryptosystem for completeness.

C. CRYPTANALYSIS OF THE PROPOSED SCHEME

An encryption scheme is needed to withstand various basic cryptanalytic tests, which are discussed below.

1) SECURITY ANALYSIS

The proposed chaos-based encryption scheme is a variant of a Baptista-type chaotic cryptosystem, and general Baptista-type algorithms have the following issues.

- The distribution of ciphertext is non-uniform, with a gradually decaying occurrence probability.
- The ciphertext size is larger than the plaintext size.
- Baptista-type chaotic cryptosystems can be made immune to chosen plaintext attack (CPA), but there is no known mathematical security proofs against chosen ciphertext attack (CCA). This is because of some characteristics of generated chaotic map being obtained from ciphertext C_i.

In our implementation, we attempted to mitigate some of these issues. An intermediate cipher is XORed with the modified expanded key to ensure uniform distribution. Moreover, hardware-level threats were omitted since we developed the entire algorithm on FPGA or ASIC. The security analysis is essential for a chaos-based encryption scheme elaborated below.

a: KEY SPACE

In the case of the proposed chaotic map, there is a total of 21 variables, in which 12 are independent variables, and the rest of the variables can be treated as initial conditions. Compared to other generic linear and logistic chaotic map-based encryption algorithms, our proposed dynamic system has more degree of freedom, and key space is enhanced many folds. Strong keys are selected among the entire key space based on bifurcation diagrams and Fisher g-statistical test. Each above-mentioned independent variable is originally spanned 2^{64}-bit space considering the precision. Therefore, the cumulative, chaotic space is $2^{64\times 12} = 2^{768} \approx 10^{231}$ which is greater than 2^{100} criteria to qualify for a cryptographically valid chaos generator.

b: SENSITIVITY

The proposed scheme presents high sensitivity to the key, as already evident from the sensitivity of the chaotic trajectory to initial conditions and the control parameters shown in section III-C. Various tests are performed to showcase chaotic properties of the proposed chaos generator, such as ergodicity, sensitivity to parametric values, and sensitivity to the initial condition, which exhibits unpredictability in chaotic attractors.

TABLE 4. Correlation Coefficient (Horizontally).

Image	Component	Plain Image	Encrypted Image
Lena (512x512)	R	0.9868	0.0059
	G	0.9809	0.0275
	B	0.9531	-0.0045
Tiger (2192x2192)	R	0.9239	0.0158
	G	0.8957	0.0098
	B	0.8422	0.0142
Lotus (2584x2584)	R	0.9996	-0.0070
	G	0.9983	-0.0040
	B	0.9995	0.0117

c: HISTOGRAM

In the last section Figure 12 showcases original and encrypted image and their corresponding histograms. Thereafter, Figure 11 showcases encryption of the text data, where Figure 11(a) illustrates ASCII distribution of first 6000 characters (plaintext) and Figure 11(b) is the generated ciphertext after applying our proposed encryption algorithm. From Figure 11(a), it can be seen that the distribution of plaintexts is confined to a specific determined domain, whereas, after performing the encryption operation, ciphertexts are transformed in different domains and are randomly distributed over the entire range, which is presented in Figure 11(b). It can be inferred that this distribution resembles the noise distribution. Therefore, it makes ciphertexts more secure and less vulnerable to statistical attacks.

As shown in Figure 12, the histograms of input images are vastly different, but the histograms of encrypted images are similar. This makes image encryption harder to be cracked because of generating similarly encrypted histograms utilizing different keys.

d: CORRELATION

It is known that adjacent pixels in a plain-image are highly correlated since the difference in the pixel values between adjacent pixels (horizontal, vertical, or diagonal) is very small in most cases. Hence if the adjacent pixel values in an encrypted image are highly correlated, then an attacker can run statistical attacks to gain helpful information about the secret key and might successfully recover the plain image. Thus, for strong security, an encrypted image must have no correlation between the adjacent pixels. The first row in Figure 13 shows the RGB correlation for 5000 randomly sampled pixels from the Lena plain image with their adjacent horizontal pixels. They show a high correlation due to similar values. In contrast, the next row shows the RGB correlation for 5000 randomly sampled pixels from the encrypted image where the adjacent pixel values are vastly different, thus having low correlation. Table 4 shows the corresponding correlation coefficients calculated for the above cases. The

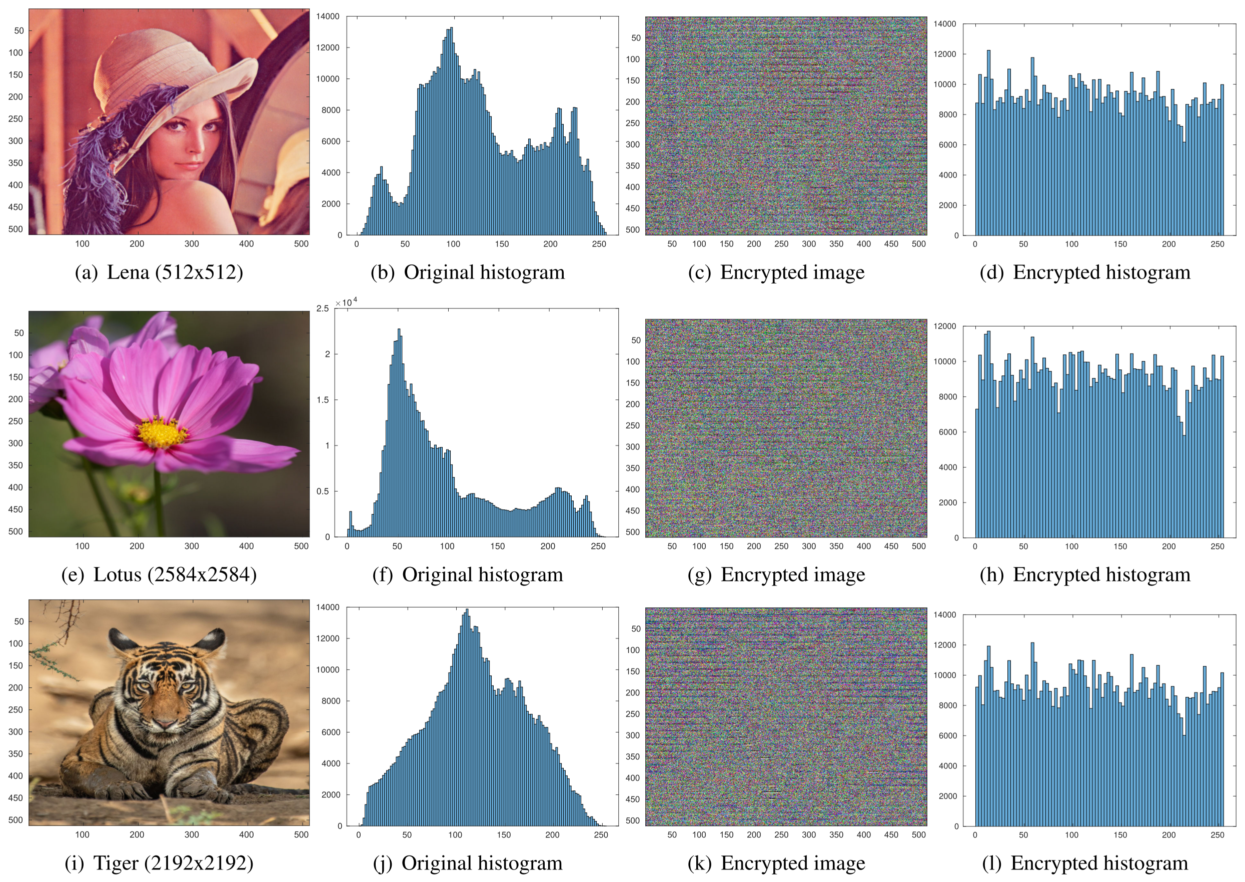


FIGURE 12. Encryption of image with histogram for a particular key value (Table 1).

calculated coefficients correctly quantify the close correlation in plain-images where r_{xy} is close to 1 (high correlation), whereas, for encrypted images, it shows significantly less correlation where r_{xy} close to 0 (low correlation).

e: INFORMATION ENTROPY

Information entropy captures the unpredictability of a message sequence by capturing the randomness of the sequence. This can be mathematically defined as $H(m) = \sum_{i=0}^{2^N-1} p(m_i)\log_2(1/p(m_i))$ where N is the number of bits in the message m, 2^N refers to all possible bit combinations, $p(m_i)$ represents the probability of m_i. Thus, for a highly random sequence of N bits, the information entropy will be closer to its maximum value, i.e., N. 8-bits can represent each component of an RGB image. Therefore the maximum entropy possible for a component of an RGB image is 8. An entropy value closer to 8 implies that the RGB image is highly random and disordered. Table 5 shows that the entropy values calculated for the encrypted image output obtained using the proposed scheme. Notably, the entropy values are much closer to eight than the plain-images, making it harder for any attacker to get adequate information about the secret key involved in the encryption or decryption process.

TABLE 5. Information Entropy.

Image	Component	Plain Image	Encrypted Image
Lena (512x512)	R	7.2531	7.9962
	G	7.5940	7.9960
	B	6.9684	7.9936
Tiger (2192x2192)	R	7.7222	7.9977
	G	7.5394	7.9971
	B	7.2159	7.9936
Lotus (2584x2584)	R	7.4546	7.9967
	G	6.9713	7.9934
	B	7.1275	7.9960

f: CHOSEN PLAINTEXT ATTACK (CPA)

We introduce the masking of a temporary variable t by employing p. In this step, p is first left rotated by b-bits, and the outcome is XORed with f to produce the ciphertext c. It is to mention that b depends on p and is explained in Algorithm 1. It can be observed that b and f are chosen randomly utilizing the proposed chaotic map. Therefore, even if an adversary attacks with the prior knowledge of c and p using the chosen plaintext, it cannot determine b and f with significant accuracy. Thus, the determination of f employing chosen plaintext attack would take at least eight steps if each outcome is stored in eight bits. For rotating plaintext of length n by b bits, the minimum number of steps required is 8^n. If we

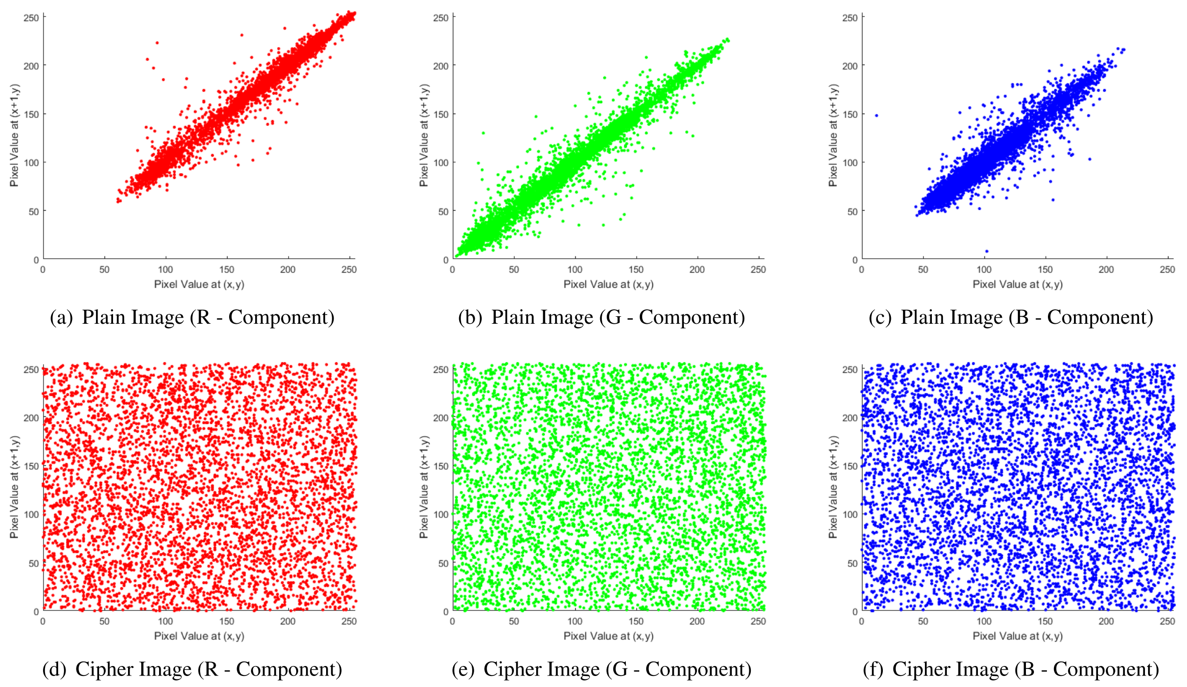


FIGURE 13. **Correlation (Horizontal) of Plain vs Encrypted Lena Image.**

mask multiple times, the total steps required to operate as mentioned above is $8^{n^{n^{\cdots^{n}}}}$, exhibiting P-SPACE complexity.

g: DIFFERENTIAL ATTACK

A good cryptosystem should be highly sensitive to the input plain message to resist differential attack, also known as Chosen Plaintext Attack(CPA). For example, in the case of images, a small difference in a few of the pixel values in an input image should ideally produce large differences in the pixel values of the encrypted output image. Currently, there are two widely used metrics to measure the sensitivity of the cryptosystem with respect to the input images: NPCR(Net Pixel Change Rate) and UACI(Unified Average Changing Intensity). NPCR measures the percentage of the pixels that differ between two encrypted images, whereas UACI measures how much the two images differ in intensity on average. Mathematically, for each RGB component of two encrypted images E_1 and E_2 of size $M \times N$, will have the following relations.

$$NPCR = \frac{\sum_{i=1}^{i=M} \sum_{j=1}^{j=N} W(i,j)}{M \times N} \times 100 \tag{21}$$

where

$$W(i,j) = \begin{cases} 0, & \text{if } E_1(i,j) = E_2(i,j) \\ 1, & \text{otherwise} \end{cases} \tag{22}$$

and

$$UACI = \frac{\sum_{i=1}^{i=M} \sum_{j=1}^{j=N} |E_1(i,j) - E_2(i,j)|}{M \times N \times 255} \times 100 \tag{23}$$

TABLE 6. **Values of NPCR and UACI.**

Image	Component	NPCR	UACI
Lena (512x512)	R	96.84	31.54
	G	97.19	31.92
	B	97.13	31.66
Tiger (2192x2192)	R	97.14	31.71
	G	97.27	31.9
	B	97.08	31.68
Lotus (2584x2584)	R	96.96	31.65
	G	97.17	31.78
	B	97.12	31.86

For our analysis, both these metrics were calculated using two encrypted images: the first image is the encrypted image E_1 of the Lena plain image, and the second is an encrypted image E_2 of a slighted modified Lena plain image obtained by incrementing the value of a random pixel by 1 (a single bit difference).

Table 6 shows the NPCR and UACI results. The NPCR values are close to 100%, i.e. both E_1 and E_2 are very different, and UACI values are around 33%, i.e., both E_1 and E_2 are 33% different in magnitude. Therefore, the proposed algorithm is susceptible to the input images and is robust against differential attacks.

h: COLLISION ATTACK

Collision resistance is the property of cryptography algorithms, which makes it difficult to find two inputs that give the same encrypted output. It is well-known that finding collisions is computationally very hard. As discussed earlier, an entire range of $\hat{y}$ are partitioned according to the number

TABLE 7. Comparison of the Proposed Scheme with Literature, where ✓ means "Achieved" and × means "Not Achieved" and - means "Data Not Available."

Test	Proposed work	[19] (2022)	[21] (2021)	[20] (2019)	[22] (2019)	[23] (2019)	[24] (2017)	[25] (2015)	[26] (2021)	[27] (2022)
Chaos Properties										
Chaotic map class	Triple Pendulum System	Chaotified Linear System	Multistable Hyper-chaotic System	Hénon map	Cross-coupled Skew Tent map	Chaotic Hash	Chen chaotic system	1D logistic map	2D LSM	2D PPCS
Lyapunov exponent	✓	×	✓	✓	-	×	✓	×	✓	✓
Bifurcation diagram	✓	×	✓	✓	✓	×	-	×	✓	✓
Ergodicity	✓	×	×	×	×	×	×	×	×	×
Sensitivity	✓	-	✓	✓	✓	✓	×	×	×	✓
Randomness Analysis										
NIST $800-22$	✓	-	×	✓	✓	×	✓	×	×	✓
Security Analysis										
Key Space	✓	✓	×	✓	×	✓	✓	✓	×	×
key Sensitivity	✓	×	✓	✓	×	✓	✓	✓	✓	×
NPCR and UACI	✓	✓	✓	✓	×	✓	×	✓	✓	×
Histogram	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
Correlation	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
Information Entropy	✓	✓	✓	✓	×	✓	×	✓	✓	×
Side channel	✓	×	×	×	×	✓	×	×	✓	×
Collision Attack	✓	×	×	×	×	✓	×	×	×	×
Throughput										
Speed (>MB/s)	✓	-	-	✓	-	-	×	✓	×	×
Comparison Analysis	✓	-	✓	✓	×	×	×	×	×	×
Hardware Evaluation										
FPGA platform	✓	✓	✓	✓	✓	×	×	×	×	×
ASIC $180nm$	✓	×	×	×	×	×	×	×	×	×

of characters. For example, $N_c = 256$, i.e., $[\hat{y}_{min} + k\epsilon, \hat{y}_{min} + (k+1)\epsilon]$ for $k = 0, 1, \ldots, 255$, which are mapped to different intervals. In the proposed method, two different inputs having different characters are mapped to different intervals always. Thus, the proposed scheme can be considered to be collision resistant. Further, the complexity of the implementation of the proposed algorithm is presented below.

i: SIDE CHANNEL ATTACK

Since hardware is designed indigenously, as described in the next section, there is no chance of corrupting it with any hardware trojan. Therefore, the security threat regarding side-channel attacks and other hardware design-related attacks is impossible.

Table 7 summarized all the analyses performed for the proposed chaos-based scheme and compared it with existing contemporary algorithms. Our proposed scheme passed all the essential criteria to qualify for a chaos-based cryptosystem mentioned in [28] and [29].

D. COMPLEXITY ANALYSIS

The computational complexity of the proposed scheme is described below.

- The computation of state variables (θ) for a duration t and time step ΔT requires N iterations, where $N = \frac{T}{\Delta t}$. In each iteration, only six equations are analyzed to find six state variables $(\theta_1, \theta_2, \theta_3, \dot{\theta_1}, \dot{\theta_2}, \dot{\theta_3})$ in $O(1)$ time. Thus, the time required for generating an asymptotical map is $O(N)$.
- The space complexity for storing state variables is $O(N)$.
- Baptista type partitioning takes constant time, if range of the variables is already computed in the previous step. For assigning variables to corresponding intervals, $O(N)$ time is required for each variable. The $K_c = 21$ variables constituting symmetric key in our proposed method, $K_cO(N) \approx O(N)$ time is required.
- The encryption of a plaintext having M characters utilizes M iterations. In each iteration, an encrypted value is selected randomly in an interval in a constant time. Therefore, time complexities for setting up of an encryption module and processing each plaintext through are $O(N)$ and $O(M)$, respectively.
- Space complexity of an encryption module is $O(N)$.
- The time complexities of setting up decryption module and its execution are $O(N)$ and $O(M)$, respectively. The information rate of the proposed cryptosystem is defined as the ratio of the size of plain-text, SZ_{pt}, to the size of cipher-text, SZ_{ct}, which is defined below.

$$R = \frac{SZ_{pt}}{SZ_{ct}} = \frac{8 * M}{\lceil \log_2 N \rceil * M} = \frac{8}{\lceil \log_2 N \rceil} \tag{24}$$

The overall time complexity of the complete encryption and decryption methods is the sum of individual complexities mentioned above, which add up to $O(N)$. Due to this, it can be utilized in many applications efficiently to make them secure without sacrificing performance significantly. Since both the time and space complexities of the proposed method are linear, it offers excellent scope for its parallelization to boost hardware performance. Due to its immunity against various attacks, scalability, and computational simplicity, our proposed method becomes a prominent candidate to provide much-needed security to various applications, such as IoT-based smart systems, low latency systems, etc.

VI. HARDWARE IMPLEMENTATION AND RESULTS

The proposed cryptosystem is implemented using System Verilog on Digilent Zed-Board having Xilinx Zynq-7 FPGA, and a synthesized netlist is generated by Synopsis Design Compiler (DC) employing SCL 180nm Bulk CMOS technology PDKs. For hardware realization of the proposed scheme, a permutation map ($\pi(k)$) is fixed as an identity map, and a maximum of three state variables are stored in a particular interval during an encryption phase. As a result, it is required to search over a total number of characters, N_c, for an interval during the decryption cycle. For evaluating equation 14, its RHS expression is converted into a postfix format, and the resultant postfix expression is stored in a ROM on an FPGA board. Later, this expression is evaluated using a stack-based postfix evaluation technique. The angle combinations of sinusoidal terms present in the expressions are cached in a LUT and are appropriately decoded while evaluating these expressions. The overall block diagram of this scheme is presented in Figure 14 and its finite state machine is illustrated in Figure 15.

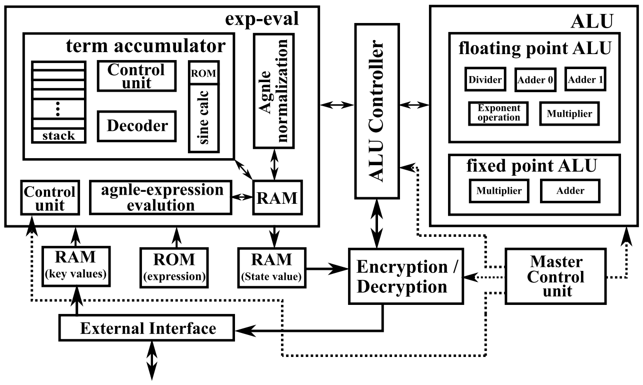


FIGURE 14. Overall block diagram of proposed scheme.

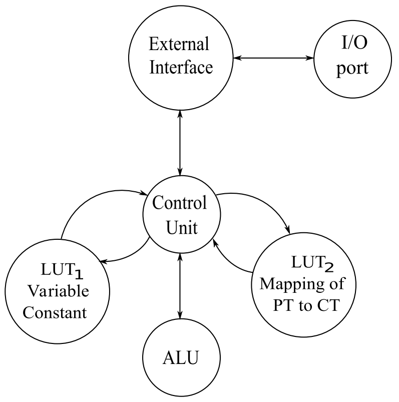


FIGURE 15. Overview of FPGA Implementation.

A. DETAILED DESCRIPTION OF HARDWARE MODULE

The hardware design can be divided into three main parts: a LUT-Memory, an ALU, and a Control Unit. A brief description of these units is elaborated below.

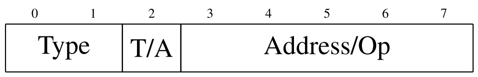
1) LOOK UP TABLE (LUT) AND MEMORY

During FPGA implementation, five LUTs and a stack memory are utilized for solving differential equations by evaluating postfix expressions, which depend on angle combinations presented in Table 8. These five LUTs are used to store *Parameters, Angle Combinations, State Variables, Numeric Constants* and *Postfix Evaluations*. Parameters represented in Table 1 are employed for the above-mentioned purpose.

TABLE 8. LUT Details.

Angle Combinations	$t = 0$	Angle Combinations	$t = 0$
$2\theta_1 - 2\theta_3$	2.1124	θ_2	5.0781
$2\theta_1 - 2\theta_2$	1.4896	$\theta_2 - \theta_3$	0.3114
$\theta_1 - \theta_2$	0.7448	$2\theta_1 - \theta_2 - \theta_3$	1.801
$\theta_1 + \theta_2 - 2\theta_3$	1.3676	$2\theta_1 - \theta_2$	6.5677
$\theta_1 - \theta_3$	1.0562	$\theta_2 - 2\theta_3$	−4.4553
$\theta_1 - 2\theta_2 + \theta_3$	0.4334	$2\theta_1 + \theta_2 - 2\theta_3$	7.1905
θ_1	5.8229	θ_3	4.7667
$\theta_1 - 2\theta_3$	−3.7105	$2\theta_1 - \theta_3$	6.8791
$2\theta_2 - 2\theta_3$	0.6228	$2\theta_2 - \theta_3$	5.3895
$\theta_1 - 2\theta_2$	−4.3333	$2\theta_1 - 2\theta_2 + \theta_3$	6.2563

As mentioned earlier, in implementing postfix evaluations, RHS expressions in equation 14 are converted into postfix format. The resultant postfix expression constitutes numeric constants, state variables or parameters, operations, and sinusoidal terms. Each of these terms is encoded to an 8-bit code. Then an entire postfix expression is stored in a LUT in the coded format. During postfix evaluation, these codes are decoded, and an appropriate data value is fetched from a corresponding address. The 8-bit encoding format can be expressed as shown below.



The mathematical analytical model of triple pendulum system presented in appendix A, B and C are parsed and evaluated by applying the information presented in Table 9. After that, stack memory is configured to execute encryption and decryption modules.

Stack: A stack consisting of two immediate registers and memory is implemented to evaluate postfix expressions. The immediate registers store the top two variables on the stack. An operation is performed in an expression employing these two immediate registers, and its result is stored in the first register. Another register is again filled with a variable popped from the top of the stack for the next operation. It continues until all the operations related to a mathematical expression are completed.

When a complex operation requires more resources than two immediate registers, the above operation is performed in two or more steps. In this case, the variable stored in the second immediate register is pushed onto the stack, and

TABLE 9. Encoding for Mathematical Operations by 8-bit Postfix Format.

Type	T/A	Address/Operation	Remarks
00	X	XXXXX	Constant terms
01	0	0x00 to 0x1F (addr.)	Parameters from Table I & II
	1	0x00 to 0x1F (addr.)	State variables
10	0	0x00 to 0x1F (addr.)	Sine values from Table IV
	1	0x00 to 0x1F (addr.)	Cosine values from Table IV
01	X	XX000 (op.)	Exponentiation
		XX001 (op.)	Multiplication
		XX010 (op.)	Division
		XX011 (op.)	Addition
		XX100 (op.)	Subtraction
11	1	11111 (op.)	End of expressions

X = don't care; op. = operation; addr. = memory location of LUT.

the corresponding memory location and program counter are updated. This process repeats until a complex operation is executed, producing the correct output. This method helps in reducing resource utilization while performing this operation. Another advantage of using stack-based push-pop architecture is using it for solving any differential equation expressed in the form of a state-space equation. Only angle combinations and mnemonics in the stack need to be changed for this operation, keeping hardware architecture identical for all the operations.

2) ARITHMETIC AND LOGIC UNIT (ALU)

Arithmetic and logical unit (ALU) consisting of two addition, one multiplication, one exponentiation, and one division modules are presented in this section. All these modules perform floating point operations in the standard IEEE-754 format. The datapath in the addition module is split into four stages. The clock frequency of this operation is kept at 100 MHz to meet the timing constraints of the FPGA employed to realize the proposed design. As mentioned earlier, the datapath in the multiplication module is divided into five stages to achieve the clock frequency.

Moreover, if required, the multiplication module can be configured to function in a pipelined mode. The exponentiation module evaluates a floating point input raised to an integer as an exponent. It instantiates the multiplication module internally for the evaluation of an exponent. Similarly, the datapath of the division module is sliced into multiple stages to meet the target clock frequency of 100 MHz. However, the division operation requires significant clock cycles compared to other operations. Therefore, the design of the proposed algorithms is optimized to keep the number of division operations as less as possible.

3) CONTROL UNIT

The control unit of the proposed design is essentially a finite state machine (FSM), which performs required tasks in a well-defined sequence. In order to implement it efficiently, the complete FSM is factored into smaller state machines hierarchically. The top-level FSM serves as a controller for all the smaller FSMs. Detailed descriptions for these state machines are given below.

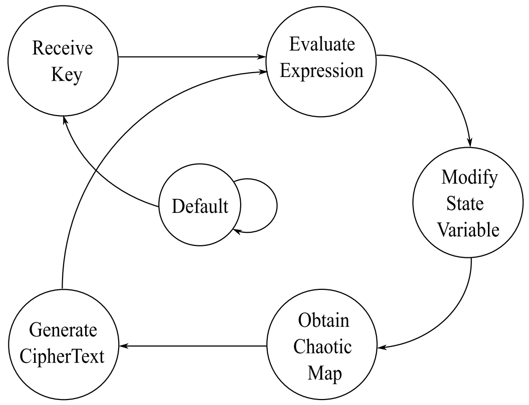


FIGURE 16. Top-Level FSM.

(i) **Top Level State Machine:** Figure 16 exhibits a simplified FSM of the top module. Description of each state is given below.
- DEFAULT: It is an initial state of FSM after reset.
- RECEIVE_KEY: It receives encryption or decryption keys through USB or Ethernet interfaces.
- EVALUATE_EXPRESSION: It evaluates state-space expression using given initial conditions for a chaotic dynamic system.
- MODIFY_STATE_VARIABLES: In this state, state variables are updated based on the previous state. It involves multiply and addition modules to perform updated operations.
- OBTAIN_CHAOTIC_MAP: It evaluates a chaotic map through a linear combination of all state variables and calculates a linear combination of θ, $\dot{\theta}$ and $\ddot{\theta}$. It is to mention that θ corresponds to the chaotic map. Further, it calculates intervals, in which θ lies and adds it to corresponding address in LUT.
- GENERATE_CIPHERTEXT: It generates a ciphertext for a particular plaintext string using a generated chaotic map and utilizes an LUT to convert plaintext into ciphertext.

(ii) **Evaluate State-Space Expression:** Figure 17 exhibits an FSM of a state space solver module and its detailed description is presented below.
- DEFAULT: This is an initial state of FSM after reset.
- FETCH_INITIAL_CONDITIONS: It fetches required initial conditions, θ and $\dot{\theta}$, from Block-RAM (BRAM).
- ANGLE_COMBINATION_CALC: This state generates linear combinations of state variables.
- ANGLE_NORMALIZATION: In this state, trigonometric inputs are normalized in the range of $[-\pi, \pi]$.
- POSTFIX_TERM_ACCUMULATE: It evaluates a state-space expression stored in the postfix form.
- STORE_OUTPUT_DATA: This state stores state-space expression obtained in the previous state in the memory.

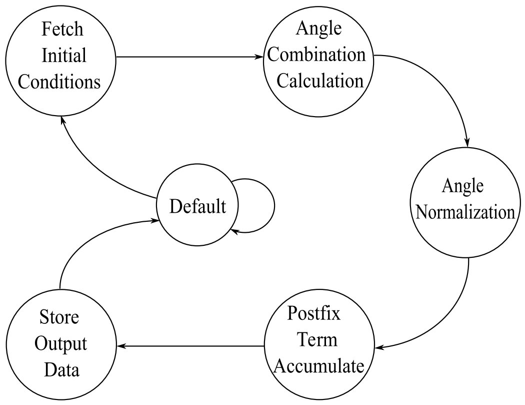


FIGURE 17. FSM for State-Space solver.

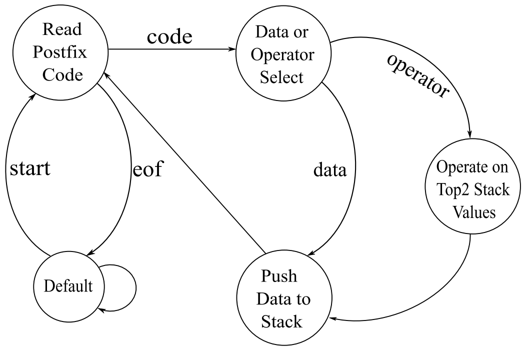


FIGURE 18. FSM for Postfix Evaluator.

(iii) **Postfix Expression Evaluation:** Figure 18 presents an FSM of postfix expression evaluation, which outputs encrypted/decrypted values. A description of each state is described below.

- DEFAULT: This is an initial state of FSM after reset.
- READ_POSTFIX_CODE: In this state, mnemonic of the next term of a postfix expression is fetched from the stack.
- DATA OR OPERATOR: This state checks whether a fetched code corresponds to data or an operator.
- OPERATE_ON_TOP_TWO_STACK_VALUES: It performs an operation based on top two data values stored in a stack.
- PUSH_DATA_TO_STACK: It pushes data in a postfix expression or result of an ALU operation onto the stack.

It is to mention that the differentiating factor between encryption and decryption operations in *Postfix Expression Evaluation FSM* depends on the postfix expressions, which are stored in postfix evaluation LUTs. Employing the aforementioned FSMs, encryption and decryption algorithms proposed in this chapter are implemented and validated against various benchmarks. In the next section, implementation details of these methods on FPGA and ASIC are presented.

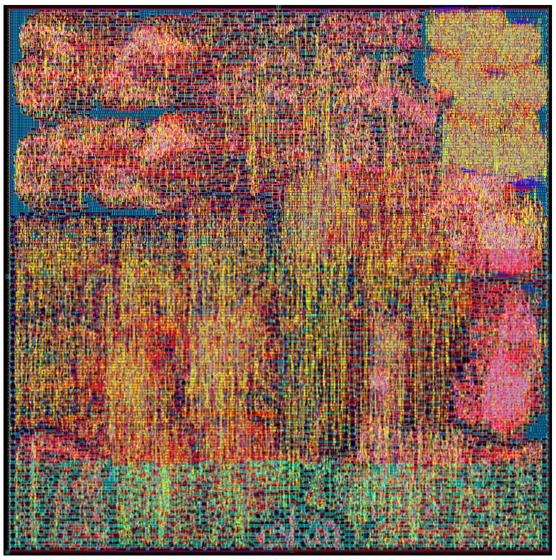


FIGURE 19. Chip layout at 180nm Bulk CMOS technology node.

B. FPGA AND ASIC IMPLEMENTATION DETAILS

A detailed analysis of the proposed encryption and decryption algorithms is presented in previous sections to validate their correctness and efficacy. These methods are implemented on hardware using System Verilog and are synthesized using Xilinx Vivado [39].

Table 10 exhibits a detailed comparison and resource utilization of our hardware implementation with other contemporary research, such as Adrián et al. [40], Pérez-Resa et al. [41] and two variants of Richard et al. [42] etc., reported in the literature. It can be observed that the proposed design consumes 1.825×, 3.446×, 2.275×, and 1.115× less resource and produces 2.396×, 2.396×, 58.3×, and 21.863× greater throughput than [40] and [41] and the two variants presented in [42], respectively. Further, the proposed scheme is 1.785× and 4.167× power efficient, and exhibits 120.6× and 5.025× less delay while compared with [40] and [41], respectively. It is to mention that our proposed method exploits maximum hardware resources, memory, and LUTs, in storing state variables only. Although it enables the proposed design to be memory-optimized, most of the static power is consumed in the memory operations. Since FPGA is the target device, a few DSP slices (DSP48) are employed during the synthesis of the proposed design, which enables efficient implementation of trigonometric functions. This makes the proposed scheme a resource and delay efficient and presents better throughput than other state-of-the-art methods.

It can be observed further that the power consumption of FPGA implementation of our proposed design is 186 mW, and its throughput is 2396.164 MBps at 100 MHz. It has the least power-delay product compared with other designs, which indicates that our proposed design is energy efficient. Further, the proposed design is realized employing 180 nm Bulk CMOS technology available with Semi-Conductor Lab

TABLE 10. **Hardware Realization of the Proposed Chaos Based Cryptosystem With Existing Ones.**

Work	LUT/FF/Slice/BRAM/ DSP/ BUFG	Comparative Resource Utilization	Power (mW)	Delay (ns)	PDP^a (nJ)	Throughput (MBps)
Adrián et al. [40]	10943/3629/ 3190/*/144/*	**1.825**×	332	192	63.744	1000
Pérez-Resa et al. [41]	16978/11127/ 5636/77/*/*	**3.446**×	775	8	6.2	1000
Richard-1 et al. [42]	7426/11285/ 3268/343/*/*	**2.275**×	∼	∼	∼	41.1
Richard-2 et al. [42]	3587/5592/ 1596/170/*/*	**1.115**×	∼	∼	∼	109.6
Proposed	3042/3519/ 3237/5/8/2	**1**×	186	1.592	0.296	2396.164

∼: not specified; *: Resource not utilized; ^{a}: Power-delay-product.

TABLE 11. **Actual Resource Utilization on SCL's 180nm technology node at 250MHz.**

Parameters	Value
Total std. cells	6229
Combinational cells	5086
Sequential cells	1105
Number of Buf/Inv	1075
Number of references	38
Combinational area	113800.501382 μm^2
Buf/Inv area	9217.829951 μm^2
Non combinational area	83402.028854 μm^2
Net Interconnect area	6533.662340 μm^2
Total cell area	197202.530237 μm^2
Total Area	**203736.192577** μm^2
Internal Power	51.1217 mW
Switching Power	10.7619 mW
Leakage Power	1.0772 μW
Total Power	**61.8836 mW**

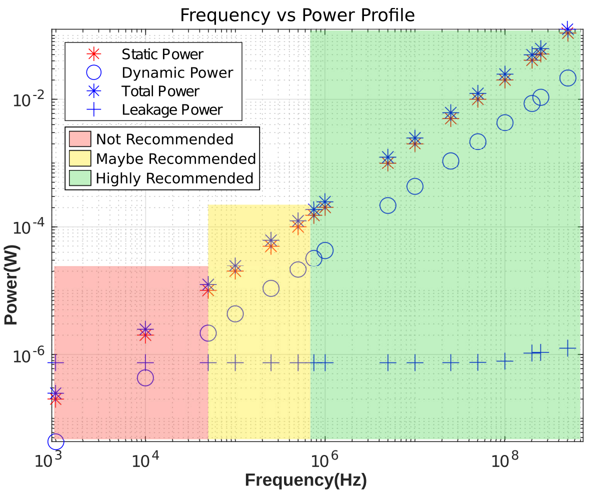


FIGURE 20. **Power profile of the proposed design at various operating frequencies (1KHz to 500MHz).**

(SCL), India. Table 11 presents the actual physical area, power, and resource utilization estimated using above mentioned technology node at 250 MHz. The fully synthesized HDL from Vivado is exported to the Design Compiler (DC) tool of Synopsys design suite and a custom Synopsys design constraint (SDC) file is generated, containing all the timing specifications of input-output ports and clocking information. In DC, the SCL's process design kit (PDK) libraries are linked, and based on the PDKs, SDC, synthesizable HDL is converted to a netlist file. The generated netlist file is fed to the IC Compiler (ICC) tool of Synopsys design suite, which physically places all the standard cells in a predefined silicon die area. Subsequently, the routing of all the standard cells is completed along with the power rails (VDD, VSS), and the clock tree is synthesized after the routing operation is completed. After successfully completing all these steps, accurate power and area estimation can be obtained. The chip layout of the proposed design illustrated in Figure 19 and all the design parameters exhibited in Table 11 are extracted through IC Compiler.

In order to validate the use of our proposed design in various applications, its power profile is analyzed by varying operating frequency from 1 KHz to 500 MHz, which is presented in Figure 20. The total power measured at a particular frequency is the sum of static, dynamic, and leakage power. It can also be observed in Figure 20 that the static power contributes significantly to the total power. Below 100 KHz, leakage power becomes comparable to the dynamic power, which is not recommended for reliable circuit performance at 180 nm technology. Above 100 KHz, leakage power is insignificant compared to static and dynamic power. Further, the leakage power does not rise substantially at 500 MHz. This implies that our proposed design can be used reliably at more than 100 KHz to high operating frequencies at a 180-nm technology node. The operating range is anticipated to be different with the smaller technology nodes. Therefore, the chaotic map generated utilizing a nonlinear triple pendulum is cryptographically secure and can be used in designing PRNG and symmetric key encryption schemes. Based on the experimental and analytical results, it can be emphasized that the proposed scheme is better than any other contemporary chaos-based encryption algorithm. Because of its smaller key size, optimal area, high throughput, and low power consumption, it can also be used efficiently to enhance the security of IoT devices and any low latency, high-performance, secure application.

VII. CONCLUSION

A triple pendulum nonlinear dynamic system based nonlinear chaotic generator is proposed in this work. Mathematical analysis of this chaos generator using bifurcation diagrams, Lyapunov exponent test, etc., is performed to verify its chaotic property. The efficacy of the chaotic map generated through our proposed methodology is ratified by employing it in developing a cryptographically secure pseudo-random number generator (PRNG). The randomness of this PRNG is successfully validated through various standard benchmark tests, viz., sensitivity to parametric and initial values, ergodicity, collision test, NIST randomness test, etc. The proposed chaotic generator is also utilized to develop a symmetric key encryption scheme. It is realized on the FPGA platform exhibiting 0.186 W power consumption, 1.592 ns delay, and 2396.164 MBps throughput. The power consumption, resource utilization, and throughput of the encryption scheme are 1.785×, 1.825×, and 2.396× better than the other known contemporary methods, respectively. The proposed cryptosystem is further implemented using 180nm Bulk CMOS technology occupying 0.20374 mm^2 die area and consuming 61.88 mW power at 250MHz. It is illustrated that this system can operate from low (at least 100KHz) to high operating frequencies efficiently without showing any performance deterioration. The efficient area utilization of the proposed chaos-based cryptosystem with high throughput at a low power enables us to utilize it in the IoT-based systems and in high-performance applications to provide them desired strength against malicious attacks.

APPENDIX A
DIFFERENTIAL EQUATION OF θ_1

$$\ddot{\theta}_1 = -(2((l_3^2 m_3^2 sin(2\theta_1 - 2\theta_3)(4I_2 - l_2^2 m_2) + l_2^2 sin(2\theta_1 - 2\theta_2)(m_2 + 2m_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3)))l_1^2 \dot{\theta}_1^{\,2} + (l_2(sin(\theta_1 - \theta_2)((m_2 m_3(m_2 + 3m_3)l_3^2 + 4I_3(m_2^2 + 6m_2 m_3 + 8m_3^2))l_2^2 + 4I_2(m_3(m_2 + m_3)l_3^2 + 4I_3(m_2 + 2m_3))) + l_3^2 m_3^2 sin(\theta_1 + \theta_2 - 2\theta_3)(4I_2 - l_2^2 m_2)) \ldots \theta_2^{\,2} - 4k_2 l_2(cos(\theta_1 - \theta_2)(m_3(m_2 + m_3)l_3^2 + 4I_3(m_2 + 2m_3)) - l_3^2 m_3^2 cos(\theta_1 + \theta_2 - 2\theta_3))\dot{\theta}_2 + l_3 m_3(sin(\theta_1 - \theta_3)(8I_3 m_3 l_2^3 + 4I_3 m_3 l_3^2 + 16I_2 I_3) + l_2^2 sin(\theta_1 - 2\theta_2 + \theta_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3)))\dot{\theta}_3^{\,2} - 4k_3 l_3 m_3(cos(\theta_1 - \theta_3)(2m_3 l_2^2 + 4I_2) - l_2^2 cos(\theta_1 - 2\theta_2 + \theta_3)(m_2 + 2m_3))\dot{\theta}_3 - g(sin(\theta_1)((m_3(m_1 m_2 + 2m_1 m_3 + 3m_2 m_3 + m_2^2)l_3^2 + 4I_3(m_2^2 + 6m_2 m_3 + m_1 m_2 + 4m_3^2 + 4m_1 m_3))l_2^2 + 4I_2(m_3(m_1 + 2m_2 + m_3)l_3^2 + 4I_3(m_1 + 2m_2 + 2m_3))) + l_3^2 m_3^2(sin(\theta_1 - 2\theta_3)(4I_2 l_2^2 m_2) - 2l_2^2 cos(2\theta_2 - 2\theta_3)sin(\theta_1)(m_1 + m_2)) + l_2^2 sin(\theta_1 - 2\theta_2)(m_2 + 2m_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3))))l_1 + 2k_1(4I_2(m_3 l_3^2 + 4I_3) + l_2^2(m_3(m_2 + 2m_3)l_3^2 + 4I_3(m_2 + 4m_3)) - 2l_2^2 l_3^2 m_3^2 cos(2\theta_2 - 2\theta_3))\theta_1))/(64I_1 I_2 I_3 + 8I_3 l_1^2 l_2^2 m_2^2 + 8I_1 l_2^2 l_3^2 m_3^2 + 8I_2 l_1^2 l_3^2 m_3^2 + 32I_3 l_1^2 l_2^2 m_3^2 + 16I_2 I_3 l_1^2 m_1 + 16I_1 I_3 l_2^2 m_2 + 4I_1 l_2^2 l_3^2 m_2 m_3 + 16I_2 l_1^2 l_3^2 m_2 m_3 + 48I_3 l_1^2 l_2^2 m_2 m_3 - 8I_1 l_2^2 l_3^2 m_3^2 cos(2\theta_2 - 2\theta_3) - 2l_1^2 l_2^2 cos(2\theta_1 - 2\theta_2)(m_2 + 2m_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3)) - 2l_1^2 l_3^2 m_3^2 cos(2\theta_1 - 2\theta_3)(-m_2 l_2^2 + 4I_2) + 2l_1^2 l_2^2 l_3^2 m_1 m_3^2 + 6l_1^2 l_2^2 l_3^2 m_2 m_3^2 + 2l_1^2 l_2^2 l_3^2 m_2^2 m_3 + l_1^2 l_2^2 l_3^2 m_1 m_2 m_3 - 2l_1^2 l_2^2 l_3^2 m_1 m_3^2 cos(2\theta_2 - 2\theta_3) - 4l_1^2 l_2^2 l_3^2 m_2 m_3^2 cos(2\theta_2 - 2\theta_3))$$

APPENDIX B
DIFFERENTIAL EQUATION OF θ_2

$$\ddot{\theta}_2 = (2((l_1^2 sin(2\theta_1 - 2\theta_2)(m_2 + 2m_3)(m_2 m_3 l_3^2 + 4I_3(m_3 + 2m_3)) - l_3^2 m_3^2 sin(2\theta_2 - 2\theta_3)((m_1 + 2m_2)l_1^2 + 4I_1))l_2^2 \dot{\theta}_2^{\,2} + l_1(sin(\theta_1 - \theta_2)((m_3(m_1(m_2 + m_3) + 2m_2(2m_2 + 3m_3))l_3^2 + 4I_3(m_2 + 2m_3)(m_1 + 4m_2 + 4m_3))l_1^2 + 4I_1(m_3(m_2 + m_3)l_3^2 + 4I_3(m_2 + 2m_3))) - l_3^2 m_3^2 sin(\theta_1 + \theta_2 - 2\theta_3)((m_1 + 2m_2)l_1^2 + 4I_1))l_2 \dot{\theta}_1^{\,2} + 4k_1 l_1(cos(\theta_1 - \theta_2)(m_3(m_2 + m_3)l_3^2 + 4I_3(m_2 + 2m_3)) - l_3^2 m_3^2 cos(\theta_1 + \theta_2 - 2\theta_3))l_2 \dot{\theta}_1 + (-l_3 m_3(sin(\theta_2 - \theta_3)((m_3(m_1 + 3m_2)l_3^2 + 4I_3(m_1 + 3m_2 + 2m_3))l_1^2 + 4I_1(m_3 l_3^2 + 4I_3)) - l_1^2 sin(2\theta_1 - \theta_2 - \theta_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3)))\dot{\theta}_3^{\,2} + 4k_3 l_3 m_3(cos(\theta_2 - \theta_3)((m_1 + 3m_2 + 2m_3)l_1^2 + 4I_1) - l_1^2 cos(2\theta_1 - \theta_2 - \theta_3)(m_2 + 2m_3))\dot{\theta}_3 + g(sin(\theta_2)((m_2 m_3(2m_2 + 3m_3)l_3^2 + 8I_3(m_2^2 + 3m_2 m_3 + 2m_3^2))l_1^2 + 4I_1(m_3(m_2 + m_3)l_3^2 + 4I_3(m_2 + 2m_3))) - l_1^2 sin(2\theta_1 - \theta_2)(m_3(m_1(m_2 + m_3) + m_2(2m_2 + 3m_3))l_3^2 + 4I_3(m_2 + 2m_3)(m_1 + 2m_2 + 2m_3)) + l_3^2 m_3^2(sin(\theta_2 - 2\theta_3)(m_2 l_1^2 + 4I_1) + l_1^2 sin(2\theta_1 + \theta_2 - 2\theta_3)(m_1 + m_2))))l_2 - 2k_2(4I_1(m_3 l_3^2 + 4I_3) + l_1^2(m_3(m_1 + 4m_2 + 2m_3)l_3^2 + 4I_3(m_1 + 4m_2 + 4m_3)) - 2l_1^2 l_3^2 m_3^2 cos(2\theta_1 - 2\theta_3))\dot{\theta}_2))/(64I_1 I_2 I_3 + 8I_3 l_1^2 l_2^2 m_2^2 + 8I_1 l_2^2 l_3^2 m_3^2 + 32I_3 l_1^2 l_2^2 m_3^2 + 16I_2 I_3 l_1^2 m_1 + 16I_1 I_3 l_2^2 m_1 m_3 + 4I_1 l_2^2 l_3^2 m_2 m_3 + 16I_2 l_1^2 l_3^2 m_2 m_3 + 48I_3 l_1^2 l_2^2 m_2 m_3 - 8I_1 l_2^2 l_3^2 m_3^2 cos(2\theta_2 - 2\theta_3) - 3l_1^2 l_2^2 cos(2\theta_1 - 2\theta_2)(m_2 + 2m_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3)) - 2l_1^2 l_3^2 m_3^2 cos(2\theta_1 - 2\theta_3)(-m_2 l_2^2 + 4I_2) + 2l_1^2 l_2^2 l_3^2 m_1 m_3^2 + 6l_1^2 l_2^2 l_3^2 m_2 m_3^2 + 2l_1^2 l_2^2 l_3^2 m_2^2 m_3 + l_1^2 l_2^2 l_3^2 m_1 m_2 m_3 - 2l_1^2 l_2^2 l_3^2 - 2l_1^2 l_2^2 l_3^2 m_1 m_3^2 cos (2\theta_2 - 2\theta_3) - 4l_1^2 l_2^2 l_3^2 m_2 m_3^2 cos(2\theta_2 - 2\theta_3))$$

APPENDIX C
DIFFERENTIAL EQUATION OF θ_3

$$\ddot{\theta}_3 = -(2(32I_1 I_2 k_3 \dot{\theta}_3 - l_2 l_3 m_3 \dot{\theta}_2^{\,2}(sin)\theta_1 - \theta_3)((l_2^2(m_1 m_2 + 4m_1 m_3 + 6m_2 m_3 + m_2^2) + 4I_2(m_1 + 3m_2 + 2m_3))l_1^2 + 4I_1(4I_2 + l_2^2(m_2 + 4m_3))) + l_1^2 sin(2\theta_1 - \theta_2 - \theta_3)(m_2 + 2m_3)(4I_2 - m_2 l_2^2)) - l_1 l_3 m_3 \dot{\theta}_1^{\,2}(sin(\theta_1 - \theta_3)(8I_1(m_3 l_2^2 + 2I_2) + 2l_1^2((m_1 m_3 - m_2^2)l_2^2 + 2I_2(m_1 + 4m_2 + 4m_3))) - l_2^2 sin(\theta_1 - 2\theta_2 + \theta_3)(m_2 + 2m_3)((m_1 + 2m_2)l_1^2 + 4I_1)) + 4k_3 l_1^2 l_2^2 m_2^2 \dot{\theta}_3 + 16l_3 l_1^2 l_2^2 m_3^2 \dot{\theta}_3 + 8I_2 k_3 l_1^2 m_1 \dot{\theta}_3 + 8I_1 k_3 l_2^2 m_2 \dot{\theta}_3 + 32I_2 k_3 l_1^2 m_2 \dot{\theta}_3 + 32I_1 k_3 l_2^2 m_3 \dot{\theta}_3 + 32I_2 k_3 l_1^2 m_3 \dot{\theta}_3 - 4k_1 l_1 k_3 m_3 \dot{\theta}_1(cos(\theta_1 - \theta_3)(2m_3 l_2^2 + 4I_2) - l_2^2 cos(\theta_2 - 2\theta_2 + \theta_3)(m_2 + 2m_3)) - 16I_1 I_2 g l_3 m_3 sin(\theta_3) - 4I_2 l_1^2 l_3^2 m_3^2 \dot{\theta}_3^{\,2} sin(2\theta_1 - 2\theta_3) - 4I_1 l_2^2 l_3^2 m_3^2 \dot{\theta}_3 sin(2\theta_2 - 2\theta_3) + 8I_2 g l_1^2 l_3 m_3^2 sin(2\theta_1 - \theta_3) + 8I_1 g l_2^2 l_3 m_3^2 sin(2\theta_2 - \theta_3) + 2k_3 l_1^2 l_2^2 m_1 m_2 \dot{\theta}_3 + 8k_3 l_1^2 l_2^2 m_1 m_3 \dot{\theta}_3 + 24k_3 l_1^2 l_2^2 m_2 m_3 \dot{\theta}_3 - 4k_2 l_2 l_3 m_3 \dot{\theta}_2(cos(\theta_2 - \theta_3)((m_1 + 3m_2 + 2m_3)l_1^2 + 4I_1) - l_1^2 cos(2\theta_1 - \theta_2 - \theta_3)(m_2 + 2m_3)) - 8I_1 g l_2^2 l_3 m_3^2 sin(\theta_3) - 8I_2 g l_1^2 l_3 m_3^2 sin(\theta_3) - 4k_3 l_1^2 l_2^2 m_2^2 \dot{\theta}_3 cos(2\theta_1 - 2\theta_2) - 16k_3 l_1^2 l_2^2 m_3^2 \dot{\theta}_3 cos(2\theta_1 - 2\theta_2) - 8I_2 g l_1^2 l_3 m_2 m_3 sin(\theta_3) - 16k_3 l_1^2 l_2^2 m_2 m_3 \dot{\theta}_3 cos(2\theta - 2\theta_2) - l_1^2 l_2^2 l_3^2 m_1 m_3^2 \dot{\theta}_3^{\,2} sin(2\theta_2 - 2\theta_3) + l_1^2 l_2^2 l_3^2 m_2 m_3^2 \dot{\theta}_3^{\,2} sin(2\theta_1 - 2\theta_3) - 2l_1^2 l_2^2 l_3^2 m_2 m_3^2 \dot{\theta}_3^{\,2} sin(2\theta_2 2\theta_3) + 2g l_1^2 l_2^2 l_3 m_1 m_3^2 sin(2\theta_2 - \theta_3) - g l_1^2 l_2^2 l_3 m_2^2 m_3 sin(2\theta_1 - \theta_3) + 2g l_1^2 l_2^2 l_3 m_2 m_3^2 sin(2\theta_2 - \theta_3) + g l_1^2 l_2^2 l_3 m_2 m_3 sin(2\theta_2 - \theta_3) + 4I_2 g l_1^2 l_3 m_1 m_3 sin(2\theta_1 - \theta_3) + 8I_2 g l_1^2 l_3 m_2 m_3 sin(2\theta_1 - \theta_3) + 4I_1 g l_2^2 l_3 m_2 m_3 sin(2\theta_2 - \theta_3) - 2g l_1^2 l_2^2 l_3 m_1 m_3^2 sin(2\theta_1 - 2\theta_2 + \theta_3) - 2g l_1^2 l_2^2 l_3 m_2 m_3^2 sin(2\theta_1 - 2\theta_2 + \theta_3) - g l_1^2 l_2^2 l_3 m_2^2 m_3 sin(2\theta_1 - 2\theta_2 + \theta_3) + g l_1^2 l_2^2 l_3 m_2^2 m_3 sin(\theta_3) - g l_1^2 l_2^2 l_3 m_1 m_2 m_3 sin(2\theta_1 - 2\theta_2 + \theta_3)))/(64I_1 I_2 I_3 + 8I_3 l_1^2 l_2^2 m_2^2 + 8I_1 l_2^2 l_3^2 m_3^2 + 8I_2 l_1^2 l_3^2 m_3^2 + 32I_3 l_1^2 l_2^2 m_3^2 + 16I_2 I_3 l_1^2 m_1 + 16I_1 I_3 l_2^2 m_2 + 64I_2 I_3 l_1^2 m_2 + 16I_1 I_2 l_3^2 m_3 + 64I_1 I_3 l_2^2 m_3 + 64I_2 I_3 l_1^2 m_3 + 4I_3 l_1^2 l_2^2 m_1 m_2 + 4I_2 l_1^2 l_3^2 m_1 m_3 + 16I_3 l_1^2 l_2^2 m_1 m_3 + 4I_1 l_2^2 l_3^2 m_2 m_3 + 16I_2 l_1^2 l_3^2 m_2 m_3 + 48I_3 l_1^2 l_2^2 m_2 m_3 - 8I_1 l_2^2 l_3^2 m_3^2 cos(2\theta_2 - 2\theta_3) - 2l_1^2 l_2^2 cos(2\theta_1 - 2\theta_2)(m_2 + 2m_3)(m_2 m_3 l_3^2 + 4I_3(m_2 + 2m_3)) - 2l_1^2 l_3^2 m_3^2 cos(2\theta_1 - 2\theta_3)(4I_2 - m_2 l_2^2) +$$

$$2l_1^2l_2^2l_3^2m_1m_3^2+6l_1^2l_2^2l_3^2m_2m_3^2+2l_1^2l_2^2l_3^2m_2^2m_3+l_1^2l_2^2l_3^2m_1m_2m_3-2l_1^2l_2^2l_3^2m_1m_3^2cos(2\theta_2-2\theta_3)-4l_1^2l_2^2l_3^2m_2m_3^2cos(2\theta_2-2\theta_3))$$

REFERENCES

[1] X. Wu, J. Li, and G. Chen, "Chaos in the fractional order unified system and its synchronization," *J. Franklin Inst.*, vol. 345, no. 4, pp. 392–401, Jul. 2008.

[2] E. E. May, M. A. Vouk, D. L. Bitzer, and D. I. Rosnick, "An error-correcting code framework for genetic sequence analysis," *J. Franklin Inst.*, vol. 341, nos. 1–2, pp. 89–109, Jan. 2004.

[3] R. Schmitz, "Use of chaotic dynamical systems in cryptography," *J. Franklin Inst.*, vol. 338, no. 4, pp. 429–441, Jul. 2001.

[4] T. Yang, "A survey of chaotic secure communication systems," *Int. J. Comput. Cognit.*, vol. 2, no. 2, pp. 81–130, Jun. 2004.

[5] L. Kocarev, "Chaos-based cryptography: A brief overview," *IEEE Circuits Syst. Mag.*, vol. 1, no. 3, pp. 6–21, Mar. 2001.

[6] T. Habutsu, Y. Nishio, I. Sasase, and S. Mori, "A secret key cryptosystem by iterating a chaotic map," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Brighton, U.K.: Springer, Apr. 1991, pp. 127–140.

[7] F. Huang and Z.-H. Guan, "A modified method of a class of recently presented cryptosystems," *Chaos, Solitons Fractals*, vol. 23, no. 5, pp. 1893–1899, Mar. 2005.

[8] L. Kocarev and G. Jakimoski, "Logistic map as a block encryption algorithm," *Phys. Lett. A*, vol. 289, no. 4, pp. 199–206, Oct. 2001.

[9] A. Akhavan, H. Mahmodi, and A. Akhshani, "A new image encryption algorithm based on one-dimensional polynomial chaotic maps," in *Proc. Int. Symp. Comput. Inf. Sci.* İstanbul, Turkey: Springer, Nov. 2006, pp. 963–971.

[10] S. Vaidyanathan, A. T. Azar, K. Rajagopal, A. Sambas, S. Kacar, and U. Cavusoglu, "A new hyperchaotic temperature fluctuations model, its circuit simulation, FPGA implementation and an application to image encryption," *Int. J. Simul. Process. Model.*, vol. 13, no. 3, pp. 281–296, 2018.

[11] S. S. Askar, A. A. Karawia, and A. Alshamrani, "Image encryption algorithm based on chaotic economic model," *Math. Problems Eng.*, vol. 2015, Jan. 2015, Art. no. 341729.

[12] Z. Kotulski and J. Szczepański, "Discrete chaotic cryptography," *Annalen der Physik*, vol. 509, no. 5, pp. 381–394, 1997.

[13] Z. Kotulski, J. Szczepański, K. Górski, A. Paszkiewicz, and A. Zugaj, "Application of discrete chaotic dynamical systems in cryptography—DCC method," *Int. J. Bifurcation Chaos*, vol. 9, no. 6, pp. 1121–1135, 1999.

[14] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *Int. J. Bifurcation Chaos*, vol. 8, no. 6, pp. 1259–1284, Jun. 1998.

[15] Y.-H. Chu and S. Chang, "Dynamical cryptography based on synchronized chaotic systems," *Electron. Lett.*, vol. 35, no. 12, pp. 974–975, 1999.

[16] E. Alvarez, A. Fernández, P. García, J. Jiménez, and A. Marcanoc, "New approach to chaotic encryption," *Phys. Lett. A*, vol. 263, nos. 4–6, pp. 373–375, 1999.

[17] A. Sambas, S. Vaidyanathan, E. Tlelo-Cuautle, B. Abd-El-Atty, A. A. A. El-Latif, O. Guille-Fernandez, Sukono, Y. Hidayat, and G. Gundara, "A 3-D multi-stable system with a peanut-shaped equilibrium curve: Circuit design, FPGA realization, and an application to image encryption," *IEEE Access*, vol. 8, pp. 137116–137132, 2020.

[18] A. Sambas, S. Vaidyanathan, E. Tlelo-Cuautle, S. Zhang, O. Guillen-Fernandez, Sukono, Y. Hidayat, and G. Gundara, "A novel chaotic system with two circles of equilibrium points: Multistability, electronic circuit and FPGA realization," *Electronics*, vol. 8, no. 11, p. 1211, Oct. 2019.

[19] M. Maazouz, A. Toubal, B. Bengherbia, O. Houhou, and N. Batel, "FPGA implementation of a chaos-based image encryption algorithm," *J. King Saud Univ. Comput. Inf. Sci.*, early access, Jan. 2022, doi: 10.1016/j.jksuci.2021.12.022.

[20] M. O. Meranza-Castillón, M. A. Murillo-Escobar, R. M. López-Gutiérrez, and C. Cruz-Hernández, "Pseudorandom number generator based on enhanced Hénon map and its implementation," *AEU Int. J. Electron. Commun.*, vol. 107, pp. 239–251, Jul. 2019.

[21] S. Vaidyanathan, A. Sambas, E. Tlelo-Cuautle, A. A. A. El-Latif, B. Abd-El-Atty, O. Guillén-Fernández, K. Benkouider, M. A. Mohamed, M. Mamat, and M. A. H. Ibrahim, "A new 4-D multi-stable hyperchaotic system with no balance point: Bifurcation analysis, circuit simulation, FPGA realization and image cryptosystem," *IEEE Access*, vol. 9, pp. 144555–144573, 2021.

[22] R. A. Elmanfaloty and E. Abou-Bakr, "Random property enhancement of a 1D chaotic PRNG with finite precision implementation," *Chaos, Solitons Fractals*, vol. 118, pp. 134–144, Jan. 2019.

[23] A. Belazi, M. Talha, S. Kharbech, and W. Xiang, "Novel medical image encryption scheme based on chaos and DNA encoding," *IEEE Access*, vol. 7, pp. 36667–36681, 2019.

[24] R. Hamza, "A novel pseudo random sequence generator for image-cryptographic applications," *J. Inf. Secur. Appl.*, vol. 35, pp. 119–127, Aug. 2017.

[25] M. A. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, and R. M. Lopez-Gutiérrez, "A RGB image encryption algorithm based on total plain image characteristics and chaos," *Signal Process.*, vol. 109, pp. 119–131, Apr. 2015.

[26] Z. Hua, Z. Zhu, Y. Chen, and Y. Li, "Color image encryption using orthogonal Latin squares and a new 2D chaotic system," *Nonlinear Dyn.*, vol. 104, no. 4, pp. 4505–4522, Jun. 2021.

[27] Z. Hua, Y. Chen, H. Bao, and Y. Zhou, "Two-dimensional parametric polynomial chaotic system," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 7, pp. 4402–4414, Jul. 2022.

[28] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006.

[29] M. A. Murillo-Escobar, M. O. Meranza-Castillón, R. M. López-Gutiérrez, and C. Cruz-Hernández, "Suggested integral analysis for chaos-based image cryptosystems," *Entropy*, vol. 21, no. 8, p. 815, Aug. 2019.

[30] K. Ramasubramanian and M. S. Sriram, "A comparative study of computation of Lyapunov spectra with different algorithms," *Phys. D, Nonlinear Phenomena*, vol. 139, no. 1, pp. 72–86, May 2000.

[31] Z. Hua, B. Zhou, and Y. Zhou, "Sine chaotification model for enhancing chaos and its hardware implementation," *IEEE Trans. Ind. Electron.*, vol. 66, no. 2, pp. 1273–1284, 2019.

[32] S. Kalanadhabhatta, D. Kumar, K. K. Anumandla, S. A. Reddy, and A. Acharyya, "PUF-based secure chaotic random number generator design methodology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 7, pp. 1740–1744, Jul. 2020, doi: 10.1109/TVLSI.2020.2979269.

[33] J. Boyar, "Inferring sequences produced by a linear congruential generator missing low-order bits," *J. Cryptol.*, vol. 1, no. 3, pp. 177–184, Oct. 1989.

[34] B. Paul, G. Trivedi, P. Jan, and Z. Nemec, "Efficient PRNG design and implementation for various high throughput cryptographic and low power security applications," in *Proc. 29th Int. Conf. Radioelektronika (Radioelektronika)*, Apr. 2019, pp. 1–6.

[35] B. Paul, A. Khobragade, S. Javvaji Sai, S. S. P. Goswami, S. Dutt, and G. Trivedi, "Design and implementation of low-power high-throughput PRNGs for security applications," in *Proc. 32nd Int. Conf. VLSI Design 18th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2019, pp. 535–536.

[36] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Gaithersburg, MD, USA: NIST, Apr. 2010.

[37] K. S. Roy and H. K. Kalita, "A survey on post-quantum cryptography for constrained devices," *Int. J. Appl. Eng. Res.*, vol. 14, no. 11, pp. 2608–2615, 2019.

[38] D. Bernstein, J. Buchmann, and E. Dahmen, *Post-Quantum Cryptography*. Berlin, Germany: Springer-verlag, 2009.

[39] *Xilinx Vivado Eda Tool*, Xilinx (AMD), San Jose, CA, USA, 2012.

[40] A. Perez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "Chaotic encryption applied to optical Ethernet in industrial control systems," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 12, pp. 4876–4886, Dec. 2019.

[41] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "Physical layer encryption for industrial Ethernet in gigabit optical links," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3287–3295, Apr. 2019.

[42] R. Agbeyibor, J. Butts, M. Grimaila, and R. Mills, "Evaluation of format-preserving encryption algorithms for critical infrastructure protection," in *Proc. Int. Conf. Crit. Infrastruct. Protection*. Arlington, VA, USA: Springer, Mar. 2014, pp. 245–261.

BIKRAM PAUL received the M.Tech. degree from the Department of Electronics and Electrical Engineering, National Institute of Technology, Agartala, in 2014. He is currently pursuing the Ph.D. degree with the Indian Institute of Technology, Guwahati. His research interests include cryptographic processor design for quantum secure application, post-quantum cryptography, hardware accelerator, VLSI circuits design for the IoT applications, and quantum electronics circuit design.

SOURADIP PAL received the B.Tech. degree from the Department of Electronics and Electrical Engineering, Indian Institute of Technology, Guwahati, in 2019. He is currently pursuing the M.S. degree in electrical and computer engineering with Purdue University, West Lafayette, IN, USA. He worked as a Software Development Engineer at Adobe Systems India Pvt Ltd., where he is developing cloud services for e-commerce applications. His research interests include algorithm design and automation, high-performance computing, cloud computing, machine learning, and NLP.

ABHISHEK AGRAWAL received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Technology, Guwahati, in 2019. He was a Digital Design Engineer at Texas Instruments. He is currently working at Google Hardware. His research interests include VLSI design for high-speed wireless communication, architectures for ML/AI processing, and electronic design automation.

GAURAV TRIVEDI (Member, IEEE) received the Ph.D. degree from the Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, in 2007. In 2011, he joined the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, where he is currently an Associate Professor. His research interests include circuit simulation (analog and digital) and VLSI CAD, high-performance computing, VLSI cryptographic circuits design, and hardware security.