©2025 Published in 11th International Symposium on Intelligent Technologies in Engineering and Science 16-18 May 2025 (ISITES2025 Diyarbakır - Turkey)

Reservoir Computing with Fractional-Order Chaos for AES-Compatible Cryptographic Applications

[1]Fatih ŞAHİN and *Gürkan KAVURAN

[1]Malatya Metropolitan Municipality, Turkey
*[2]Faculty of Engineering and Natural Sciences, Department of Electrical and Electronics Engineering, Malatya Turgut Özal University, Turkey

Abstract:

This study presents a method of secure key generation based on fractional-order chaotic systems and Echo State Networks (ESNs). In this case, the time series data of a chaotic system is generated from the fractional order Lorenz system using the Adams–Bashforth–Moulton method that solves the system numerically. An ESN is built with a fixed random reservoir that is trained using one state variable as the input and another as the target output. Afterwards, the output obtained is processed into a binary stream, which is then used for cryptographic key generation. Those keys are integrated into the AES encryption system. There exists a real-time interaction with the chaotic data and the encryption bleeps through a GUI which is MATLAB based. Simulations confirm the initial expectation that the ESN can model the chaotic behavior and provide fragmented keys while still satisfying the criteria for security and randomness. The approach developed here with regards to the proposal is to combine elements of chaos theory and fractional computation with machine learning for implementing advanced cryptographic systems.

Key words: Chaos Theory, Random Number Generator, Echo State Networks, Cryptography, Machine Learning

1. Introduction

Chaos theory provides a groundwork for modeling nonlinear dynamical systems that exhibit sensitivity to initial conditions and long-term unpredictability. Introduced by Lorenz in 1963 [1] chaotic systems have been applied in diverse fields such as meteorology, biology, and engineering. Their inherent randomness has also made them suitable for cryptographic applications, particularly in secure key generation [2, 3]. Traditional pseudorandom number generators (PRNGs) are limited by their deterministic nature, whereas chaos-based random number generators (RNGs) provide higher unpredictability due to their nonlinear dynamics [4]. However, efficiently harnessing chaotic signals for encryption requires accurate numerical modeling and effective data processing strategies. In parallel, Echo State Networks (ESNs)—a form of reservoir computing—have shown strong performance in modeling time-dependent signals [5, 6]. With a fixed randomly connected reservoir and a trainable output layer, ESNs offer a computationally efficient alternative to conventional recurrent neural networks (RNNs). Further enhancing the modeling capacity, fractional calculus generalizes classical differential equations to non-integer orders, enabling the representation of systems with memory and hereditary properties [7-9]. The fractional-order Lorenz system introduces additional flexibility and has been shown to preserve chaotic behavior under appropriate conditions [10, 11].
In this study, we propose a novel framework that combines fractional-order chaos, ESNs, and chaos-based AES encryption. The fractional Lorenz system is numerically solved using the

*Corresponding author: Address: Faculty of Engineering and Natural Sciences, Department of Electrical and Electronics Engineering, Malatya Turgut Özal University, 44100, Malatya TURKEY. E-mail address: gurkan.kavuran@ozal.edu.tr

Adams–Bashforth–Moulton method [9], and the resulting chaotic series is used to train an ESN. Predicted outputs are then utilized for secure key generation. A graphical user interface (GUI) is developed to demonstrate the integration of chaotic dynamics with real-time encryption.

2. Materials and Method

2.1. Echo State Networks

Echo State Networks (ESNs) are a specialized class of recurrent neural networks (RNNs) that offer an efficient framework for processing temporal data, particularly in time series forecasting, speech recognition, and control systems. ESNs are distinguished by their use of a large, sparsely connected reservoir with fixed internal weights and a simple linear readout, enabling fast and efficient training. An ESN typically consists of three main layers [12-14]:

- **Input Layer:** Projects the external inputs into the reservoir.
- **Reservoir Layer:** A dynamic, high-dimensional, nonlinear system capturing the temporal features of the input.
- **Output Layer:** Produces the final output through a trained linear transformation.

The key theoretical elements of ESNs are the reservoir dynamics and the output computation, which are explained below.

Reservoir Update Equation

At each discrete time step t, the state of the reservoir $\mathbf{x}(t) \in \mathbb{R}^{N_r}$ is updated based on the current input $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ and the previous reservoir state $\mathbf{x}(t-1)$ as follows [6], [15-18]:

$$\mathbf{x}(t) = \tanh(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t-1) + \mathbf{b}) \tag{1}$$

where, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_r \times N_u}$ is the input weight matrix, $\mathbf{W} \in \mathbb{R}^{N_r \times N_r}$ is the reservoir's internal weight matrix, $\mathbf{b} \in \mathbb{R}^{N_r}$ is a bias vector, $\tanh(\cdot)$ is the element-wise hyperbolic tangent activation function, N_r is the number of reservoir neurons and N_u is the number of input features. The matrix $\mathbf{W}$ is typically initialized randomly and scaled to satisfy the spectral radius condition (i.e., the largest absolute eigenvalue $\rho(\mathbf{W}) < 1$) to ensure the echo state property.

Output Computation

The output $\mathbf{y}(t) \in \mathbb{R}^{N_y}$ at time t is generated by a linear transformation of the reservoir states:

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{z}(t) \tag{2}$$

where,$\mathbf{z}(t) = [\mathbf{u}(t); \mathbf{x}(t)]$ is the concatenation of the input and reservoir state and $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (N_r + N_u)}$ is the output weight matrix, learned typically via linear regression. The optimization of $\mathbf{W}_{\text{out}}$ is often done by minimizing the mean squared error (MSE) between the predicted and the target outputs over a set of training samples.

2.2. Fractional Calculus

Fractional calculus extends classical calculus by allowing derivatives and integrals of arbitrary real or complex orders. The fractional derivative of a function $f(x)$, denoted by $D^{\alpha}f(x)$, generalizes the notion of differentiation for $\alpha > 0$ and integration for $\alpha < 0$. This framework provides a continuous spectrum of operators between integer orders, enabling more accurate modeling of systems with memory and hereditary effects. Much like real numbers interpolate between integers, fractional operators interpolate between integer-order differentials and integrals. Although computationally more demanding, the Adams–Bashforth–Moulton (ABM) method provides higher accuracy than the Grünwald–Letnikov approach for solving fractional-order oscillators [19] . As this study is simulation-based, the well-established predictor–corrector ABM algorithm was used. Accordingly, the fractional derivative in Equation (3) corresponds to the Volterra integral form in Equation 4), and the numerical scheme is derived as shown in Equations (6–8).

$$D_t^q y(t) = f(y(t), t), y^{(k)}(0) = y_0^{(k)}, k = 0,1, \dots, m-1 \tag{3}$$

$$y(t) = \sum_{k=0}^{[q]-1} y_0^{(k)} \frac{t^k}{k!} + \frac{1}{\Gamma(q)} \int_0^t (t-\tau)^{q-1} f(\tau, y(\tau)) d\tau \tag{4}$$

$$y_h(t_{n+1}) = \&\sum_{k=}^{m-1} \frac{t_{n+1}^{k}}{k!} y_0^{(k)} + \frac{h^g}{\Gamma(\alpha+2)} f\left(t_{n+1}, y_h^p(t_{n+1})\right) + \frac{h^g}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f\left(t_j, y_n(t_j)\right) \tag{5}$$

$$a_{j,n+1} = \begin{cases} n^{q+1} - (n-q)(n+1)^q & \text{if } j = 0 \\ (n-j+2)^{q+1} + (n-j)^{q+1} + 2(n-j+1)^{q+1} & \text{if } 1 \leqslant j \leqslant n \\ 1 & \text{if } j = n+1 \end{cases} \tag{6}$$

$$y_h^p(t_{n+1}) = \sum_{k=0}^{m-1} \frac{t_{n+1}^{k}}{k!} y_0^{(k)} + \frac{1}{\Gamma(q)} \sum_{j=0}^{n} b_{j,n+1} f\left(t_j, y_n(t_j)\right) \tag{7}$$

$$b_{j,n+1} = \frac{h^g}{q}\left((n+1-j)^q - (n-j)^q\right) \tag{8}$$

2.3. Fractional Order Lorenz System

In 1963, Edward Lorenz conducted studies on simplified equations describing convection cells that arise in atmospheric dynamics. Lorenz was the first to introduce the term "butterfly effect" in the context of chaos theory, referring to the sensitive dependence on initial conditions. The classical Lorenz chaotic system is defined by the following set of nonlinear differential equations [10], [20], 21]:

$$\begin{cases} \frac{dx(t)}{dt} = \sigma(y(t) - x(t)) \\ \frac{dy(t)}{dt} = x(t)(\rho - z(t)) - y(t) \\ \frac{dz(t)}{dt} = x(t)y(t) - \beta z(t) \end{cases} \tag{9}$$

In this system, σ is referred to as the Prandtl number, and ρ is the Rayleigh number. All parameters satisfy $\sigma, \rho, \beta > 0$, with commonly used values being $\sigma = 10, \beta = \frac{8}{3}$, and ρ taken as a variable. For $\rho = 28$, the system exhibits chaotic behavior, while for other values, regular trajectories can be observed. The Lorenz system has three equilibrium points, one of which is clearly the origin, $E_1 = (0,0,0)$. The other two equilibrium points are:

$$E_2 = (\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho-1), E_3 = (-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho-1) \quad (10)$$

The Jacobian matrix of the Lorenz system evaluated at an arbitrary equilibrium point $E^* = (x^*, y^*, z^*)$ is given by:

$$J = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - z^* & -1 & -x^* \\ y^* & x^* & -\beta \end{bmatrix} \quad (11)$$

With the standard parameter values $(\sigma, \rho, \beta) = (10,28,8/3)$, the corresponding equilibrium points are approximately:

$$E_1 = (0,0,0), E_2 = (8.4853,8.4853,27), E_3 = (-8.4853, -8.4853,27) \quad (12)$$

The fractional-order Lorenz system extends the classical model by incorporating derivatives of non-integer order, and is defined as:

$$\begin{cases} {}_0D_t^{q1}x(t) = \sigma(y(t) - x(t)) \\ {}_0D_t^{q2}y(t) = x(t)(\rho - z(t)) - y(t) \\ {}_0D_t^{q3}z(t) = x(t)y(t) - \beta z(t) \end{cases} \quad (13)$$

Here, $q_1, q_2, q_3 \in (0,1]$ are the fractional orders of the derivatives, and ${}_0D_t^{qi}$ denotes the Caputo fractional derivative of order q_i.

3. Simulation Results

In this study, the simulation of the fractional-order chaotic Lorenz system has been carried out. The system, composed of three fractional-order differential equations corresponding to its three state variables, was solved in the time domain using the predictor-corrector PECE (Predict-Evaluate-Correct-Evaluate) variant of the Adams-Bashforth-Moulton method. To verify the chaotic nature of the system, the Lyapunov exponents were computed. The presence of a positive Lyapunov exponent confirms the system's sensitivity to initial conditions, thus indicating chaotic behavior. The system parameters used in the simulation are, $\sigma = 10, \rho = 28, \beta = \frac{8}{3}, q_1 = q_2 = q_3 = 0.995$.

The initial conditions were chosen as: $x(0) = 0.1, y(0) = 0.1, z(0) = 0.1$. The simulation was conducted over a time interval of 100 seconds. Figure 2 presents the time evolution of the state

variable $x = x_1(t)$, Figure 3 illustrates the time response of the state variable $y = x_2(t)$ and Figure 4 shows the time response of the state variable $z = x_3(t)$. These plots demonstrate the characteristic oscillatory and divergent behavior of the fractional-order Lorenz system, confirming the presence of chaos under the specified parameters.

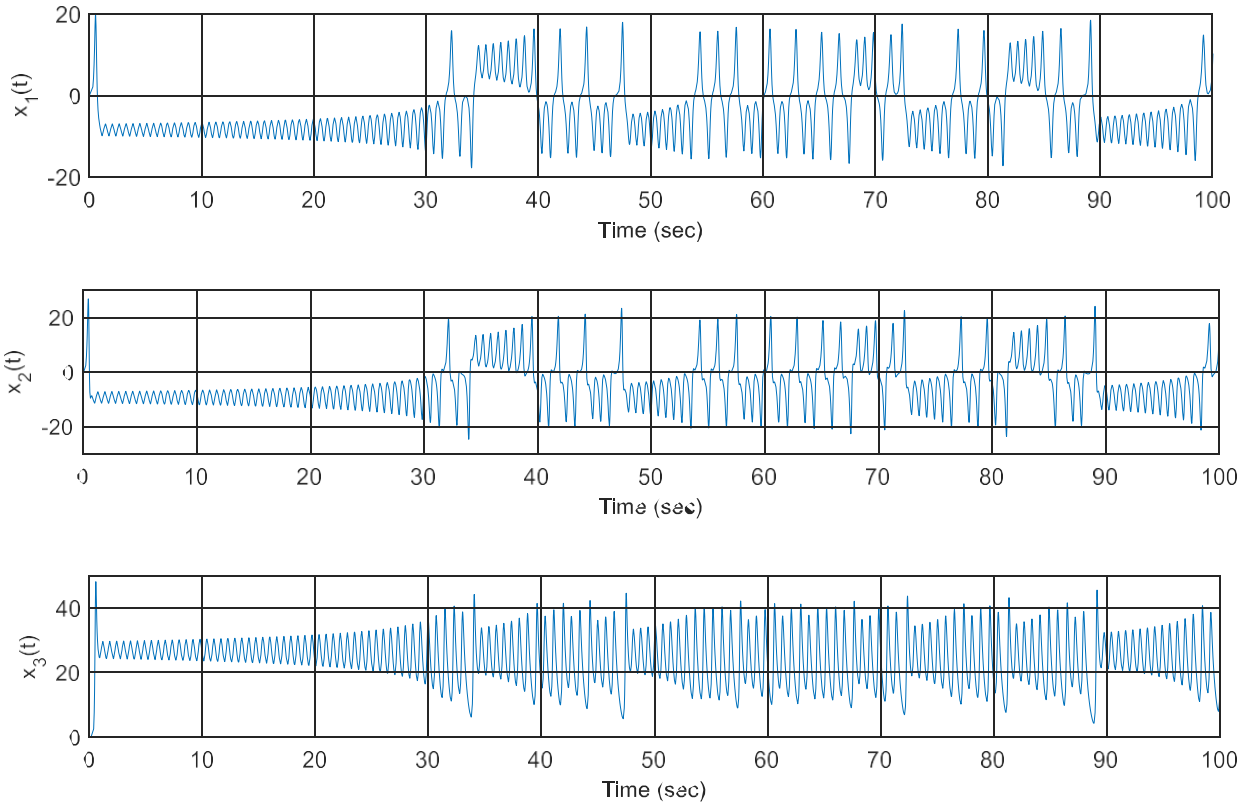

Figure 1. Time evolution of the state variable $x = x_1(t)$, $y = x_2(t)$ and $z = x_3(t)$

The chaotic nature of the system was verified by computing and plotting the Lyapunov exponents, as shown in Figure 5. As a fundamental tool of chaos theory, Lyapunov exponents are used to detect and quantify the presence of chaotic behavior in dynamical systems. They measure the rate at which nearby trajectories diverge (or converge) over time. Let the differential form of the investigated dynamical system be defined as, $x_{t+1} = f(x_t)$. To understand how points in the phase space evolve, the Jacobian matrix is used. For a system of the form $x_{t+1} = f(x_t)$, the Jacobian is defined as, $J(x) = \frac{\partial f}{\partial x}$ At each point in time, the eigenvalues of the Jacobian matrix are computed. The magnitudes of these eigenvalues indicate how small perturbations in the system grow or decay over time. The Lyapunov exponents λ_i provide the average exponential rate of divergence (or convergence) of nearby trajectories and are calculated as, $\lambda_i = \lim_{t\to\infty} \frac{1}{t} \ln \left| \frac{\delta x_i(t)}{\delta x_i(0)} \right|$.

Here, $\delta x_i(t)$ denotes the magnitude of a small perturbation in the i-th state variable at time t. If the largest Lyapunov exponent (LLE) is positive, the system exhibits chaotic behavior. A positive LLE implies that initially close trajectories diverge exponentially over time, which is a hallmark of chaos. Conversely, if all Lyapunov exponents are negative, the system converges to a fixed point or a periodic orbit. As seen in Figure 5, the first Lyapunov exponent (LE1) is positive, confirming that the system behaves chaotically. In Figure 6, the projection of all three state variables onto one another is presented, illustrating the system's characteristic butterfly effect.

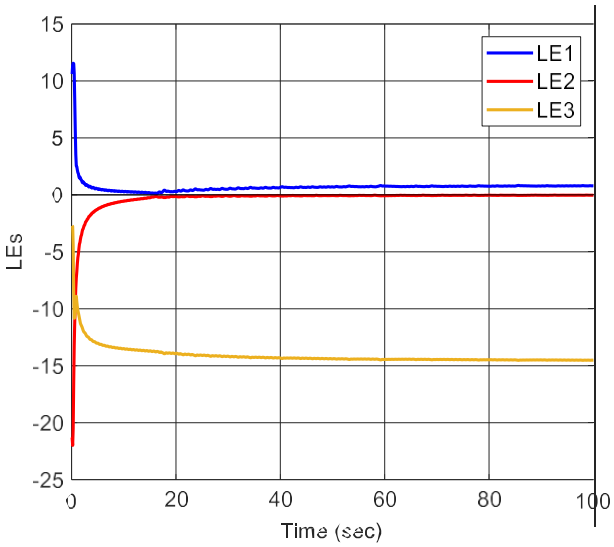

Figure 2. Evolution of Lyapunov Exponents

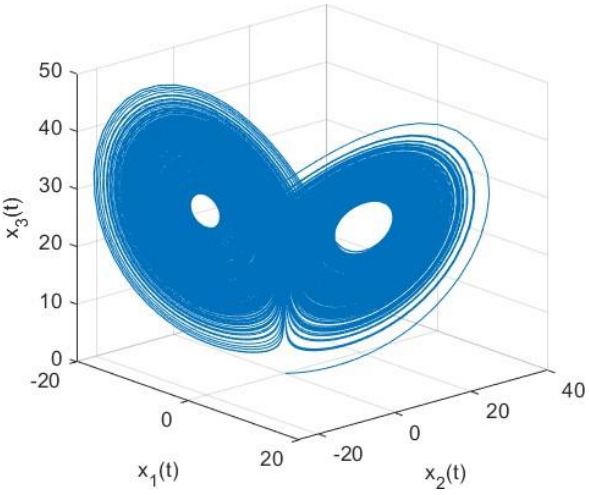

Figure 3. Phase-space trajectories showing the relationship among $x_1(t)$, $x_2(t)$ and $x_3(t)$

The chaotic time series vector obtained from the simulation, $\mathrm{X} = [x_1(t), x_2(t), x_3(t)]$, constitutes the dataset used in the Echo State Network (ESN) model. In this study, $x_1(t)$ was designated as the input signal, while $x_2(t)$ was used as the target output. A total of 20,000 data points were recorded over a simulation duration of 100 seconds. To implement reservoir computing, a reservoir composed of 30 internal units was constructed. The spectral radius was set to 0.5, and the connection sparsity was defined as 0.1. The hyperbolic tangent function (tanh) described in Equation (2) was used as the activation function. The model includes a single input and a single output unit. The 20,000 samples of $x_1(t)$ were provided as input to the reservoir, while the corresponding $x_2(t)$ values were used for training and testing. The dataset was divided equally, with 10,000 samples used for training and 10,000 for testing. From the test set, the first 100 samples were discarded to eliminate initialization transients, and analyses were conducted on the remaining 9,900 samples. A visualization of the first four units of the constructed reservoir is presented in Figure 7. The reservoir states transmitted to the output layer were processed using linear regression. The output weights were updated by minimizing the error between the predicted and actual values. In this optimization process, the Normalized Root Mean Square Error (NRMSE) criterion was employed. The performance of the model was evaluated separately for the training and testing

phases. In Figure 8, the predicted values of the time series $x_2(t)$ during training are shown in blue, while the actual values are depicted in red, revealing a strong correspondence. The testing phase results, shown in Figure 9, also demonstrate high agreement between the predicted and actual values.

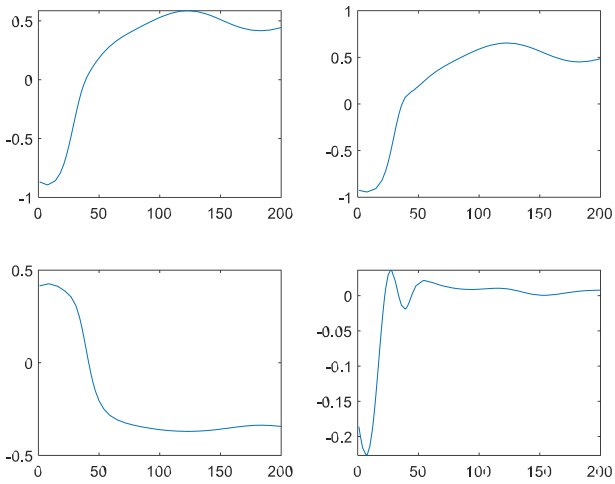

Figure 4. Structure of the first four units

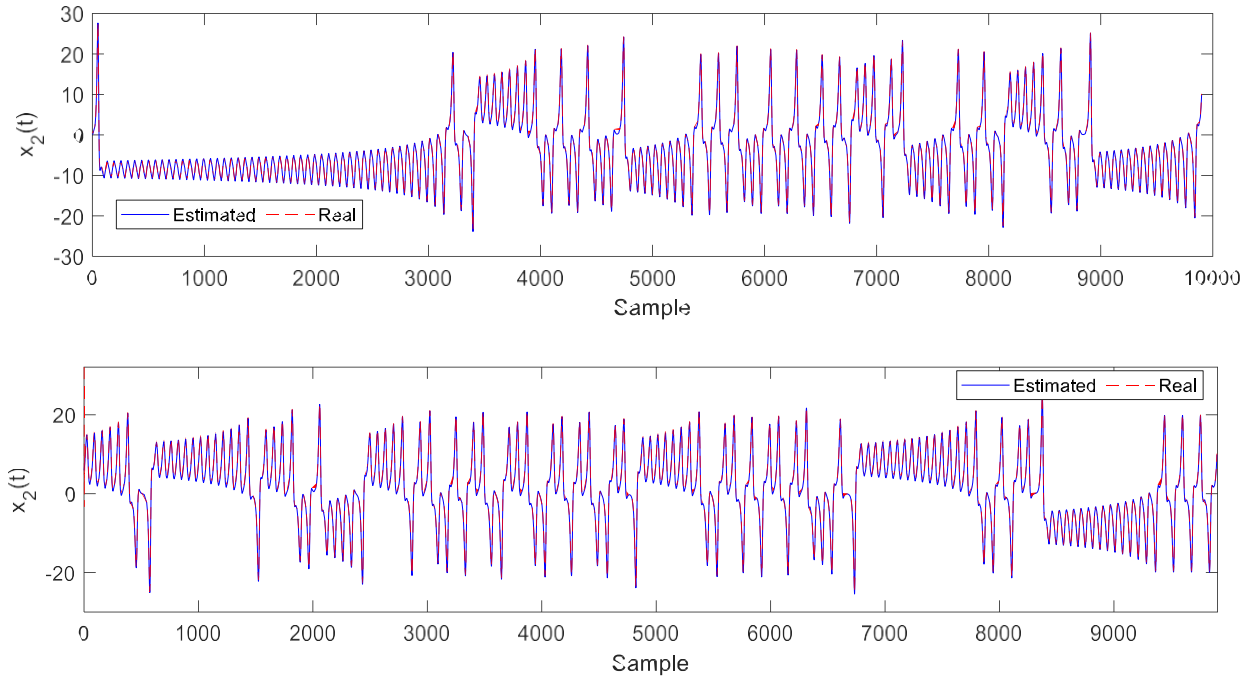

Figure 5. Predicted vs. actual values in the testing and training phase of $x_2(t)$

The results obtained from both phases confirm the reliability and accuracy of the proposed reservoir computing model. It is anticipated that the ESN architecture can achieve similar success in other

datasets and application domains. Future studies incorporating larger datasets and alternative parameter configurations are expected to further enhance the generalizability and performance of the model.

4. Chaotic Signal-Based Key Generation and AES Encryption

In this study, the Advanced Encryption Standard (AES) algorithm is integrated with a chaotic key generation process to enhance the unpredictability and security of the encryption system. A chaotic signal is used as the entropy source for key generation, leveraging its inherent sensitivity to initial conditions and complex dynamic behavior. The complete process is described below.

First, synthetically generated chaotic signal is normalized to the interval $[0,1]$ to ensure a uniform dynamic range for thresholding. The normalization is performed as:

$$x_{\text{norm}}(t) = \frac{x(t) - \min(x)}{\max(x) - \min(x)} \tag{14}$$

Next, a binary bitstream $b(t)$ is extracted from the normalized signal via thresholding. A global threshold value $\theta \in (0,1)$ is defined such that:

$$b(t) = \begin{cases} 1 & \text{if } x_{\text{norm}}(t) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

From this binary stream, a specified number of bits N (typically $N = 128$ for AES-128) are grouped into bytes to construct the encryption key. Each byte k_i is formed from 8 successive bits using:

$$k_i = \sum_{j=0}^{7} b_{8(i-1)+j+1} \cdot 2^{7-j}, \text{ for } i = 1,2,\dots,\frac{N}{8} \tag{16}$$

Once the key has been derived, the text is transformed into a byte format and padded using the PKCS#7 technique until it meets the required AES block size of 16 bytes. The padding value P can be defined as P = 16 - (length of plaintext mod 16). Subsequently, P bytes with a value of P are appended to the message. The AES algorithm is first applied in Electronic Codebook (ECB) mode, using the Java javax.crypto framework integrated with MATLAB. The final ciphertext is displayed in hex. Decryption, done with the same key, is executed after the original text is restored by removing the padding. It is evident that AES key, which depends on chaotic properties, highlights the fact that a highly sensitive system in terms of initial conditions and parameters is utilized. The dynamic AES key increases resistance to brute force and statistical attacks due to the disorderly and unpredictable behavior of chaos.

5. Graphical User Interface (GUI) Implementation

In order to facilitate user interaction with the proposed chaotic key-based encryption system, a graphical user interface (GUI) was developed using MATLAB's App Designer framework. The interface enables users to perform AES encryption and decryption operations by utilizing externally provided chaotic signals as the basis for dynamic key generation. The GUI includes the following main components:

- A file selection button for uploading chaotic signal data from .mat or .txt files;
- A text input area for entering the plaintext message to be encrypted;
- Numeric input fields for threshold value and block size (in bits), both of which directly influence the bitstream extraction and key formation processes;
- An encryption trigger button;
- Read-only display fields for visualizing the hexadecimal representation of the encrypted message and the final decrypted output.

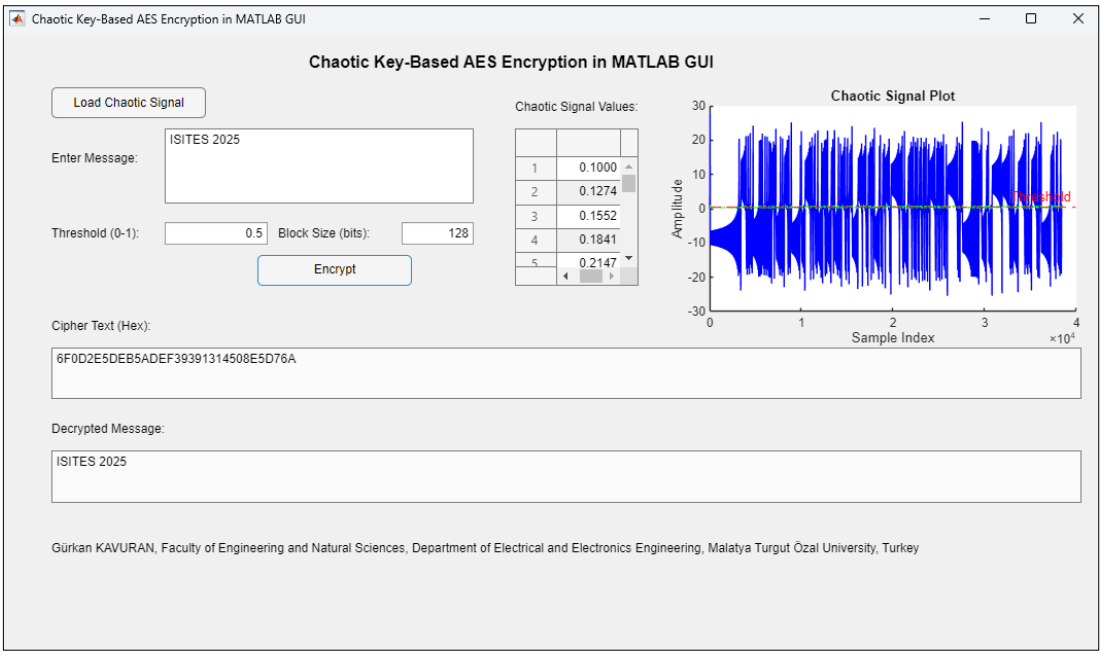

Figure 4. MATLAB GUI of the proposed system [22]

Upon execution, the uploaded chaotic signal is normalized and thresholded to produce a binary sequence. A specified number of bits (e.g., 128 for AES-128) are grouped and converted into bytes to form the encryption key. The plaintext is padded according to the PKCS#7 scheme to satisfy the block size requirements of the AES algorithm. Java's javax.crypto library is employed within MATLAB to perform AES encryption and decryption in ECB mode. This user-friendly and parameter-adjustable interface supports rapid experimentation and testing, and serves as an

educational tool for demonstrating the integration of chaotic systems with modern cryptographic frameworks.

Conclusions

In this paper, we present and analyze a novel approach of three-way fusion of fractional order chaotic systems, Echo State Networks (ESNs), and AES encryption within a cryptographic framework. Time series data was created using the Adams-Bashforth-Moulton method, and the fractional order Lorenz system was simulated to develop complex, yet light, memory time series data. Model estimation of these signals via ESN asserts that learning non-linear temporal sequences is indeed possible. The derived dynamic cryptographic keys from ESN simulated data can be utilized in AES encryption frameworks, yielding better results after the implementation of thresholding and normalization processes at this stage. The predictive accuracy of ESN simulations and the efficacy of the key generation process are confirmed through the simulation results. The MATLAB GUI facilitates interactive control over the operation and parameters of the encryption-decryption system, which allows for manually changing and monitoring system parameters in real-time during simulations. Results demonstrated the promise of integrating fractional dynamics with reservoir computing towards adaptable responsive cryptographic systems. Future studies could target hardware implementations using other fractional chaotic systems for non-system dependent use case scenarios.

References

[1] Lorenz EN. Deterministic nonperiodic flow. *J Atmos Sci* 1963;20(2):130–41.
[2] Kantz H, Schreiber T. *Nonlinear Time Series Analysis*. 2nd ed. Cambridge: Cambridge University Press; 2004.
[3] Álvarez G, Li S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int J Bifurcat Chaos* 2006;16(8):2129–51.
[4] Cuomo KM, Oppenheim AV. Circuit implementation of synchronized chaos with applications to communications. *Phys Rev Lett* 1993;71(1):65–8.
[5] Jaeger H. The ‘echo state’ approach to analysing and training recurrent neural networks. 2001. Available from: https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf
[6] Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training. *Comput Sci Rev* 2009;3(3):127–49.
[7] Oldham KB, Spanier J. *The Fractional Calculus*. New York: Academic Press; 1974.
[8] Podlubny I. *Fractional Differential Equations*. Vol. 198. San Diego: Academic Press; 1999.
[9] Diethelm K, Ford NJ, Freed AD. A predictor-corrector approach for the numerical solution of fractional differential equations. *Nonlinear Dyn* 2002;29:3–22. doi:10.1023/A:101659221934

[10] Grigorenko I, Grigorenko E. Chaotic dynamics of the fractional Lorenz system. Phys Rev Lett 2003;91(3):034101. doi:10.1103/PhysRevLett.91.034101

[11] Kavuran G. *Utilization of Fractional Order Differentiation in Nonlinear Control Methods and Signal Processing Techniques* [PhD thesis]. Malatya: Inonu University; 2017.

[12] Rodan A, Tiňo P. Minimum complexity echo state network. *IEEE Trans Neural Netw* 2011;22(1):131–44. doi:10.1109/TNN.2010.2089641

[13] Long J, Zhang S, Li C. Evolving Deep Echo State Networks for Intelligent Fault Diagnosis. *IEEE Trans Ind Inform* 2020;16(7):4928–37. doi:10.1109/TII.2019.2938884

[14] Wang Z, Zhao H, Zheng M, Niu S, Gao X, Li L. A novel time series prediction method based on pooling compressed sensing echo state network and its application in stock market. *Neural Netw* 2023;164:216–27. doi:10.1016/j.neunet.2023.04.031

[15] Karunasinghe DSK, Liong SY. Chaotic time series prediction with a global model: Artificial neural network. *J Hydrol (Amst)* 2006;323(1–4):92–105. doi:10.1016/j.jhydrol.2005.07.048

[16] Wang Z, Yao X, Huang Z, Liu L. Deep Echo State Network with Multiple Adaptive Reservoirs for Time Series Prediction. *IEEE Trans Cogn Dev Syst* 2021;13(3):693–704. doi:10.1109/TCDS.2021.3062177

[17] Kashinath K, et al. Physics-informed machine learning: Case studies for weather and climate modelling. *Philos Trans R Soc A Math Phys Eng Sci* 2021;379(2194). doi:10.1098/RSTA.2020.0093

[18] Wang Q, et al. Chaotic time series prediction based on physics-informed neural operator. *Chaos Solitons Fractals* 2024;186. doi:10.1016/j.chaos.2024.115326

[19] Kavuran G. When machine learning meets fractional-order chaotic signals: detecting dynamical variations. *Chaos Solitons Fractals* 2022;157:111908. doi:10.1016/J.CHAOS.2022.111908

[20] Petras I. *Fractional-Order Nonlinear Systems: Modeling, Analysis and Simulation*. Berlin: Springer; 2011.

[21] Atangana A, Koca I. Chaos in a simple nonlinear system with Atangana–Baleanu derivatives with fractional order. *Chaos Solitons Fractals* 2016;89:447–54. doi:10.1016/j.chaos.2016.02.012

[22] Kavuran G. chaotic_AES_gui [software]. MATLAB Central File Exchange; 2025. Available from: https://www.mathworks.com/matlabcentral/fileexchange/180914-chaotic_aes_gui