# Implementation of Chaotic Neural Key Generation Algorithm For IoT Devices

ZIED GUITOUNI[1], MOHSEN MACHHOUT[1]

[1]Electronics and Micro-Electronic Laboratory, Faculty of Sciences of Monastir, TUNISIA.

*Abstract:* - This paper presents a new method for generating encryption keys for Internet of Things (IoT) devices. This method combines chaos theory with neural networks to create secure and efficient keys. We use the Lorenz chaotic system to generate complex patterns and a Convolutional Neural Network (CNN) to learn and predict these patterns. This approach is designed to address the unique security challenges of IoT devices, which often have limited computing power. We tested our method with different key sizes and evaluated its performance using accuracy, loss, entropy, and correlation metrics. The results show that key sizes between 256 and 512 bits offer the best balance between model performance and security for IoT devices. We also conducted Diehard statistical tests, which our key generation method passed successfully, demonstrating its ability to produce high-quality random keys.

*Key-Words:* - Internet of Things, Cryptographic Key Generation, Chaotic Systems, Neural Networks, Convolutional Neural Networks, Lorenz System, Randomness.

Received: April 4, 2024. Revised: September 9, 2024. Accepted: September 26, 2024. Published: October 24, 2024.

## 1 Introduction

The Internet of Things (IoT) has emerged as a transformative paradigm, interconnecting billions of devices and revolutionizing various sectors including healthcare, agriculture, and smart cities [1]. However, the proliferation of IoT devices has introduced significant security challenges, particularly in cryptographic key generation and management [2]. The resource-constrained nature of many IoT devices, combined with their vulnerability to physical and remote attacks, necessitates novel approaches to ensure robust security measures [3].

Secure key generation is a cornerstone of IoT security, providing the foundation for encryption, authentication, and data integrity [4]. Traditional key generation methods often prove inadequate for IoT environments due to their computational intensity or lack of sufficient randomness [5]. This paper presents a novel approach to address these challenges: a Chaotic Neural Key Generation Algorithm specifically designed for IoT devices.

Chaos theory, with its inherent properties of sensitivity to initial conditions and unpredictability, offers a promising avenue for cryptographic applications [6]. Chaotic systems can generate highly complex and seemingly random behavior from simple deterministic equations, making them ideal candidates for key generation [7]. When combined with the adaptive learning capabilities of neural networks, chaos theory presents an opportunity to create a robust, efficient, and secure key generation mechanism.

Neural networks, inspired by biological neural systems, have demonstrated remarkable capabilities in pattern recognition, adaptation, and complex non-linear mapping [8]. By leveraging these properties, neural networks can potentially enhance the quality and efficiency of key generation processes [9]. The integration of neural networks with chaotic systems creates a synergistic approach that addresses the unique security requirements of IoT devices.

This paper proposes a new method for generating encryption keys for IoT devices. Our method combines the unpredictability of chaotic systems with the learning capabilities of neural networks. Specifically, we use a type of chaotic system called the Lorenz system to create complex patterns [6]. We then use a Convolutional Neural Network (CNN) to learn and predict these patterns, which helps us generate encryption keys [7].

The remainder of this paper is organized as follows: Section 2 describes the background of Chaotic Systems in Cryptography and the CNN architectures. Section 3 outlines the proposed algorithm for chaotic neural network-based key generation. In Section 4, we present the experimental results from implementing the algorithm, including the training process evaluation and the security analysis. Finally, Section 5 concludes the paper.

# 2 Background

## 2.1 Chaotic Systems in Cryptography

Chaotic systems are complex mathematical models that are very sensitive to their starting conditions [12]. This means that even a tiny change in the beginning can lead to very different results over time. This property makes chaotic systems useful for cryptography, where we need to create unpredictable patterns [7].

One popular chaotic system is the Lorenz system, first described by Edward Lorenz in 1963 [13]. It's a set of three equations that model weather patterns, which are given by:

$$\begin{cases} dx/dt = \sigma * (y - x) & (1) \\ dy/dt = x * (\rho - z) - y & (2) \\ dz/dt = x * y - \beta * z & (3) \end{cases}$$

where x, y, and z represent the system's state variables, and sigma, rho, and beta are parameters. The chaotic sequence is generated by iterating the equations using a given seed value.

Figure 1 shows a 3D representation of the Lorenz attractor, generated using the Lorenz equations with parameters $\sigma$ = 10, $\rho$ = 28, and $\beta$ = 2.667. The figure displays a butterfly-shaped pattern, a well-known characteristic of the Lorenz system. The trajectory is plotted over 10,000 time steps, with a step size of 0.01, starting at the point (0, 1, 1.05). As the system evolves, the path never repeats, though it remains confined within a specific region of the phase space.

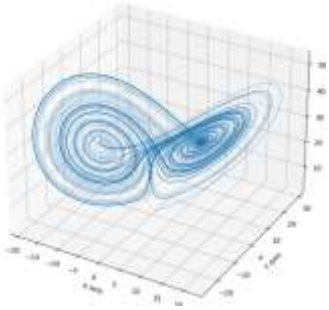

**Figure 1. Lorentz Attractor**

The two lobes of the attractor, resembling butterfly wings, reflect the system's oscillation between two semi-stable states. This pattern illustrates the deterministic but unpredictable nature of chaotic systems, where small variations in initial conditions can lead to drastically different outcomes. The continuous, non-overlapping path emphasizes the system's sensitivity to starting points and its complex, non-linear behavior. This visualization effectively demonstrates the dynamics of chaotic systems and their potential applications in fields like cryptography and secure communication.

## 2.2 Convolutional Neural Networks (CNNs)

CNNs are a powerful type of machine learning tool that are great at finding patterns in data. They're well-known for their use in image recognition, but CNNs have also proven to be valuable in various fields, including cryptography.

The structure of a CNN is like a series of building blocks, each with its specific role. The first blocks are convolutional layers. These layers use filters that move across the input data, searching for specific features. It's similar to how our eyes might scan a picture, noticing things like edges or shapes.

After the convolutional layers, we have pooling layers. These layers reduce the data, retaining only the most crucial information. This not only speeds up the network but also helps it focus on the bigger picture instead of getting caught up in minor details. The final part of a CNN consists of fully connected layers. These function as the brain of the network, taking all the information gathered by the earlier layers and using it to make decisions, such as classifying an image, making a prediction, or solving a complex problem.

What sets CNNs apart is how they learn. As they process more data, they improve at recognizing patterns, even in new information they haven't encountered before. This ability to learn and adapt makes CNNs a valuable tool in many areas of research and technology. The image in Figure 2 illustrates how a CNN is constructed, emphasizing the role of each component in pattern recognition.

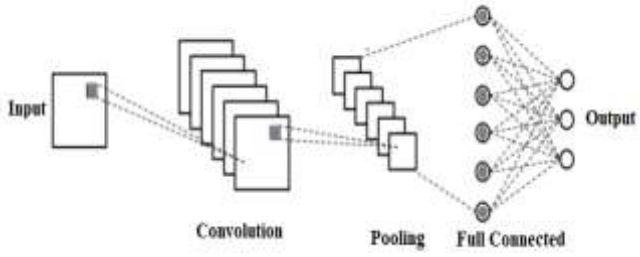


**Figure 2. Convolutional Neural Networks Architecture**

Our study utilizes a specialized CNN to handle chaotic data for generating encryption keys. The network begins with two layers that identify patterns. The first layer has 32 small pattern-finders, while the second has 64. These layers employ ReLU to enhance the network's understanding of complex patterns. After detecting patterns, the network organizes the information into a simple list. This list then undergoes two additional processing steps. The first step, with 256 units, further refines the patterns.

The final step, also with 256 units, determines the likelihood of each potential outcome.

In the following section, we introduce an innovative approach to creating encryption keys. Our method combines two powerful concepts: chaotic systems and neural networks. By bringing these two concepts, we've developed a way to generate secure keys from random data. The chaotic system provides unpredictability, while the neural network learns to work with this randomness efficiently. What makes our method special is that it's designed to work well on devices with limited processing power. This is particularly useful for Internet of Things (IoT) devices, which often have constraints on their computing capabilities.

# 3 Chaotic Neural Key Generation Algorithm

Our key generation method combines chaos theory and machine learning to create secure encryption keys. The main steps in the key generation process outlined by this method are as follows:

- *Chaotic Sequence Creation*: We use the Lorenz system, a type of chaotic math model, to make a long sequence of unpredictable numbers. We start with a seed value and use special equations to generate this sequence.
- *Data Preparation*: We divide the chaotic sequence into smaller chunks, each containing 8 numbers. These chunks become our input data. The next number after each chunk is our target output. We adjust these numbers to make them easier for our neural network to process.
- *Neural Network Setup***:** We use a CNN for our task. This type of network is good at finding patterns in data. Our CNN has layers that look for patterns and layers that make decisions based on those patterns.
- *Training the Network***:** We teach our CNN using the prepared data. The network learns to predict the next number in the sequence based on the previous 8 numbers. We use a method called categorical cross-entropy to measure how well the network is learning.
- *Checking the Network*: After training, we test the CNN with new data to make sure it can accurately predict numbers in the chaotic sequence. This step ensures our network works well and isn't just memorizing the training data.
- *Making the Encryption Key*: Once we're sure our network works well, we use it to predict new numbers based on our chaotic sequence. We convert these predictions to binary (1s and 0s) and mix them with some additional random bits. This creates our final encryption key.

This proposed method is particularly useful for devices with limited processing power, like many Internet of Things (IoT) devices. It creates strong, secure keys without requiring a lot of computational resources. The combination of chaos theory and neural networks makes the keys hard to predict, enhancing security.

# 4 Evaluation of the proposed method

In this section, we present the experimental evaluation of the key generation method.

## 4.1 Evaluation Metrics

We evaluated and compared the neural network architectures based on four key performance metrics:

- Accuracy: Accuracy measures the percentage of predictions matching true labels. It is calculated as the number of correct predictions divided by the total number of samples. We tracked accuracy on both the training and validation sets to gauge model fitting and generalization respectively.
- Loss: Loss quantifies the model's error during training. We utilized categorical cross-entropy loss which calculates the divergence between predicted and true probability distributions. Lower loss values indicate better fitting on the training data.
- Validation Accuracy: To assess generalization, we computed accuracy on a held-out validation set not involved in training. This reflects the model's ability to correctly generate cryptographic keys for previously unseen input data, an important metric for our application.
- Validation Loss: Analogous to validation accuracy, we measured loss on the validation set. This allowed monitoring changes in out-of-sample error to detect potential overfitting as training progressed. A stable or decreasing validation loss indicates the model is still learning useful patterns rather than memorizing the training data.

We recorded accuracy and loss values at the end of each training epoch and plotted them to visualize the learning dynamics.

## 4.2 Training evaluation

To assess the sensitivity of the proposed algorithm concerning key size, we conducted a series of experiments using various key sizes. Figure 2 shows the accuracy of our chaotic neural key generation method for different key sizes

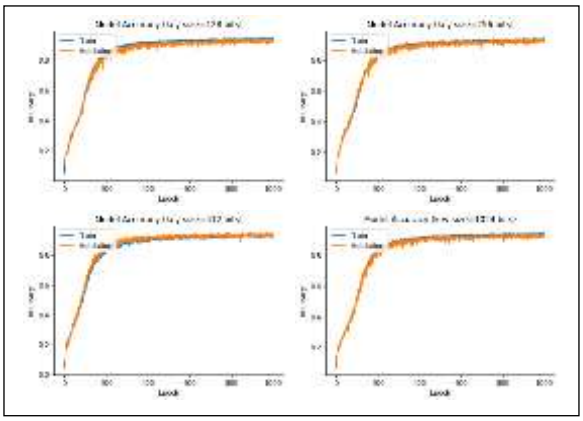

**Figure 3. Precision Variation with Varying Key Sizes**

For all key sizes, we see a significant improvement in accuracy as the model trains. The 256-bit key size performs best, reaching a final training accuracy of 92.56% and a validation accuracy of 91%. The 512-bit key size follows closely, with 92.48% training accuracy and 91.10% validation accuracy. The 128-bit and 1024-bit keys also show good performance, with final training accuracies above 91% and validation accuracies above 88%. Interestingly, while larger key sizes generally lead to better security, they don't always result in higher accuracy in our model. This suggests that a balance between key size and model performance is crucial for optimal security in resource-constrained IoT devices. Figure 4 illustrates the loss values for our chaotic neural key generation method across different key sizes.

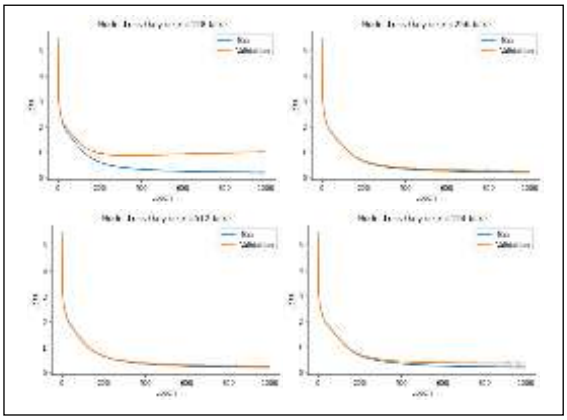

**Figure 4. Loss Variation with Varying Key Sizes**

According to Figure4, all key sizes show a significant decrease in loss during training, indicating that our model is learning effectively. The 256-bit key size demonstrates the best performance, with the lowest final training loss of 0.2476 and validation loss of 0.7513. The 512-bit key size follows closely, achieving a final training loss of 0.2856 and the lowest validation loss of 0.4445. Interestingly, while the 1024-bit key has the highest initial loss, it still achieves a competitive final training loss of 0.3102. However, its validation loss remains higher at 1.3198, suggesting some overfitting. The 128-bit key, despite its smaller size, performs reasonably well with a final training loss of 0.2532. These results suggest that medium-sized keys (256 and 512 bits) offer the best balance between model performance and potential security for our IoT-focused key generation method.

Based on the results shown in Figures 3 and 4, we can conclude that for IoT devices using our chaotic neural key generation method, key sizes between 256 and 512 bits work best. These key sizes offer a good balance between how well the model performs and how secure the keys are. This balance is important for IoT devices, which need strong security.

### 4.3 Security Evaluation

In this subsection, we assess the security of the key generation method. To evaluate the randomness of the generated keys, we examine key aspects including entropy variation, correlation variation, and statistical tests using Diehard.

#### *4.3.1 Entropy Variation*

Entropy testing is a key metric for evaluating the randomness and unpredictability of generated keys [16]. Higher entropy values reflect a stronger degree of randomness, which in turn enhances security. By examining entropy variation, we can assess the quality and strength of the chaotic neural key generation process.

In our analysis, a chaotic system with a specific seed was used to generate a series of chaotic values. These values were fed into a neural network to produce random keys. Entropy testing was then performed on the keys generated for various key lengths: 128, 256, 512, and 1024 bits.

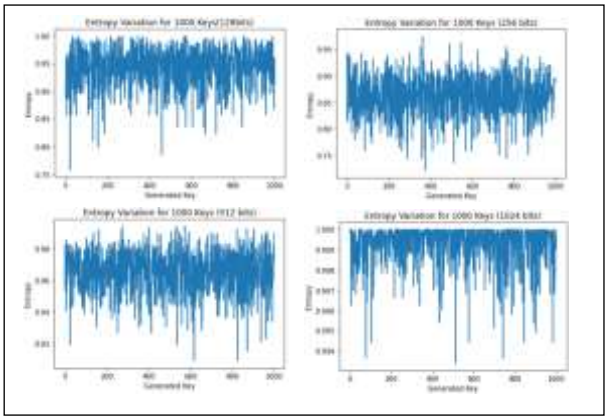

**Figure 5. Entropy Variation**

Figure 5 shows how entropy varies across these different key sizes. The results provide important insights into the security of our chaos-based neural key generation method. We found a clear link between key length and entropy: as the key length increased, so did the entropy values, indicating higher randomness and stronger security.

According to Figure 5, the minimum entropy values recorded were 0.75, 0.72, 0.90, and 0.99 for key lengths of 128, 256, 512, and 1024 bits, respectively. The maximum entropy values rose from 0.99 for the 128-bit key to 1.0 for the 1024-bit key. Similarly, the average entropy values followed the same pattern, with 0.94, 0.86, 0.96, and 0.99 for the respective key lengths.

These results emphasize the importance of using longer key lengths in chaos-based neural key generation to boost entropy and enhance security. Longer keys provide a larger key space, making it harder for attackers to guess the key through brute-force or statistical methods. The higher entropy associated with longer keys improves randomness and increases resistance to cryptographic attacks.

### *4.3.2 Correlation Variation*

The correlation values observed during cryptographic key generation are important indicators of the quality and security of the process [17]. In our study, we generated 1000 keys and analyzed their correlation values. The results, presented in Figure 6, provide valuable insights into the performance of the key generation process.

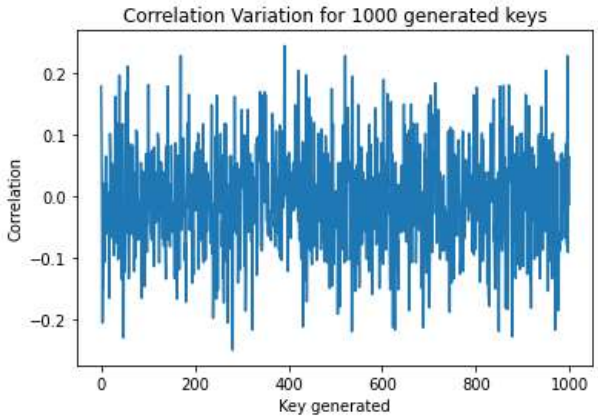


**Figure 6. Correlation Variation**

The minimum correlation value of -0.25 and the maximum of 0.24 highlight the effectiveness of the key generation process in achieving bit decorrelation. These values, close to ±0.25, show that the bits making up the keys are well-separated from one another, which is essential for preventing the prediction or partial reconstruction of one key based on another. Additionally, the average correlation value of -0.005 indicates an almost perfect statistical independence between the bits. This level of independence ensures high entropy and confusion in the generated keys, further strengthening their security.

### *4.3.3 Diehard Test*

The Diehard Test includes a series of statistical tests designed to identify weaknesses or vulnerabilities in random number generators [18]. By analyzing the results of these tests, we can assess the strength of the key generation process and its ability to withstand statistical attacks.

Table 1 presents the outcomes of 15 Diehard tests conducted on the generated keys. The results show that the key generation process successfully passed all 15 tests. The p-values from each test are higher than the significance threshold, indicating that the keys demonstrate high-quality randomness and meet the required standards. This consistent performance across all tests reinforces the reliability and effectiveness of the key generation method.

TABLE 1
Results of Diehard Tests for Key Generation Using FFNN

| Test Name | P-Value | Result |
|---|---|---|
| Birthday spacing | 0.241 | Passed |
| Binary rank 31*31 | 0.633 | Passed |
| Binary rank 32*32 | 0.400 | Passed |
| Binary rank 6*8 | 0.625 | Passed |
| Count the1 | 0.312 | Passed |
| Parking lot | 0.097 | Passed |
| Minimum distance | 0.345 | Passed |
| 3D sphere | 0.879 | Passed |
| the Squezze | 0.588 | Passed |
| Overlapping sum | 0.196 | Passed |
| Run up 1 | 0.584 | Passed |
| Run up 2 | 0.701 | Passed |
| Run down 1 | 0.931 | Passed |
| Run down 2 | 0.106 | Passed |
| Craps of throws | 0.783 | Passed |
| Craps of wins | 0.727 | Passed |

These findings are important for the security of IoT devices, where randomness plays a critical role in ensuring the confidentiality and integrity of sensitive data. The results demonstrate that the proposed algorithm generates random keys with sufficient randomness, contributing to enhanced security for IoT devices.

The positive Diehard test results boost confidence in the proposed method, showing its suitability for cryptographic applications in IoT devices. Secure key generation enables IoT devices to establish safe communication, authenticate users, and protect sensitive data from unauthorized access and cyberattacks. Therefore, these findings

positively impact the overall security and privacy of IoT ecosystems.

## 5 Conclusion

This paper introduced a new approach to cryptographic key generation for IoT devices by integrating chaotic systems with neural networks. Our method addresses the specific security needs of resource-limited IoT devices while ensuring strong and efficient key generation. The evaluation of our chaotic neural key generation algorithm produced promising outcomes. For example, with a 256-bit key, we achieved a training accuracy of 92.56% and a validation accuracy of 91%. Entropy analysis showed high levels of randomness, with mean entropy values ranging from 0.86 for 256-bit keys to 0.99 for 1024-bit keys. Correlation testing revealed an average correlation of -0.005, demonstrating strong statistical independence between key bits. Additionally, our method passed all 15 Diehard statistical tests, proving its ability to generate high-quality random keys. These findings indicate that our approach offers an effective solution for secure key generation in IoT environments. Its balance between security and efficiency makes it well-suited for devices with limited computational resources.

*References:*

[1] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things—A survey of topics and trends," Information Systems Frontiers, vol. 17, no. 2, pp. 261-274, 2015.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125-1142, 2017.

[3] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2483-2495, 2018.

[4] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," Computer Networks, vol. 76, pp. 146-164, 2015.

[5] K. Sha, W. Wei, T. A. Yang, Z. Wang, and W. Shi, "On security challenges and open issues in Internet of Things," Future Generation Computer Systems, vol. 83, pp. 326-337, 2018.

[6] L. Kocarev, "Chaos-based cryptography: a brief overview," IEEE Circuits and Systems Magazine, vol. 1, no. 3, pp. 6-21, 2001.

[7] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," International Journal of Bifurcation and Chaos, vol. 16, no. 08, pp. 2129-2151, 2006.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[9] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, 1976.

[10] C. Li, F. Min, and C. Li, "Multiple coexisting attractors of the Lorenz system," International Journal of Bifurcation and Chaos, vol. 29, no. 3, 1950033, 2019.

[11] W. Wang, J. Chen, X. Wang, T. Huang, and L. Wang, "Encryption Key Generation in IoT Systems: Current Trends and Challenges," IEEE Internet of Things Journal, vol. 8, no. 9, pp. 7212-7231, 2021.

[12] S. H. Strogatz, "Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering," CRC Press, 2018.

[13] E. N. Lorenz, "Deterministic Nonperiodic Flow," Journal of the Atmospheric Sciences, vol. 20, no. 2, pp. 130-141, 1963.

[14] L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, "PHY-Layer Spoofing Detection With Reinforcement Learning in Wireless Networks," IEEE Transactions on Vehicular Technology, vol. 65, no. 12, pp. 10037-10047, 2016.

[15] H. Wang, J. Zhao, Y. Zhang, and C. Luo, "Lightweight Key Generation Scheme for Edge-enabled IoT Based on CNN and PUF," IEEE Internet of Things Journal, vol. 8, no. 10, pp. 8084-8094, 2021.

[16] Y. Choi, Y. Yeom, and J.-S. Kang. "Practical Entropy Accumulation for Random Number Generators with Image Sensor-Based Quantum Noise Sources", Entropy, vol. 25, no. 7, p. 1056, Jul. 2023

[17] Zhu, H., Zhao, C., Zhang, X., & Yang, L . "A novel iris and chaos-based random number generator", Computers & Security, 36, 40-48, 2013.

[18] Axel Gebbert and Andreas Thor, "Usability Issues of Dieharder for Empirical Testing of Pseudorandom Number Generators." J. UCS 24(2), pp.130-157, 2018.

## .Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

• Zied Guitouni and Mohsen Machhout proposed the idea of the paper.
• Zied Guitouni carried out the simulation and optimization of the proposed algorithm.
• All authors organized and executed the experiments in Section 4.
• All authors were responsible for the paper's writing and editing.

**Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself**
No funding was received for conducting this study.

**Conflict of Interest**
The authors have no conflicts of interest to declare that are relevant to the content of this article.