CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES

https://cajmtcs.casjournal.org/index.php/CAJMTCS

Volume: 07 Issue: 01 | January 2026 ISSN: 2660-5309

Article

OGA-Based Chaotic Key Generation for Lightweight IoT Encryption

Thaer M. Kadhim

1. Ministry of Eduaction,Thi-Qar, Iraq
* Correspondence: cse.61012@uotechnology.edu.iq

Abstract: With the increasing in IoT devices utilities, comes a need to secure their communications. IoT devices generally combine low computational cost with strong security. While traditional encryption algorithms will work well for large-scale systems normally, they don't meet the stringent energy requirements of lightweight IoT environments or the processing constraints associated with these devices. In addition, standard or static key scheduling strategies are not adequate to deal with either the continual evolution of attack surfaces or changing operating environments.

In this paper, we present a hybrid Online Genetic Algorithm (OGA) - Chaotic Key Generation algorithm that generates adaptive, high-entropy encryption keys in real-time utilizing a chaotic map for randomness and an Online Genetic Algorithm to continuously evolve the control parameters for optimizing maximum entropy, avalanche strength, and bit level balance. The evolved chaotic sequence serves as the initialization value for a lightweight symmetric encryption algorithm (such as AES-128 or PRESENT) for maximum unpredictability and minimum latency.

Experimental simulations show that the OGA–Chaos system achieves near-ideal entropy (7.99 bits/byte), reduced key correlation, and approximately 30 % lower processing overhead compared with classical genetic and static chaotic generators. These results demonstrate the promising of online evolutionary optimization as a a new tool for adaptive IoT cryptosystems.

Keywords: Online Genetic Algorithm, chaotic encryption, lightweight IoT security, adaptive key generation, logistic map.

Citation: Kadhim, T. M. OGA-Based Chaotic Key Generation for Lightweight IoT Encryption. Central Asian Journal of Mathematical Theory and Computer Sciences 2026, 7(1), 343-353

Received: 10th Nov 2025
Revised: 21th Dec 2025
Accepted: 14th Jan 2026
Published: 06th Feb 2026

1. Introduction

The rapid expansion of the Internet of Things (IoT) devices and applications has resulted a large number of interconnected devices that require both secure and computationally efficient cryptographic solutions. Modern algorithms such as AES and RSA are provide robust security with high processing and memory requirements which can lead challenges for smaller IoT devices. Despite their cryptographic strength, these algorithms required more computational power and memory that exceed the capabilities of resource-constrained embedded nodes.

Therefore, lightweight cryptography has become one of the core areas of focus in research related to maintaining data confidentiality and integrity without excess power or processing costs in Internet-of-Things (IoT) systems [1] [2] [3]. In addition, the chaotic systems are useful sources for the generation of random numbers for lightweight encryption due to their sensitivity to initial conditions, non-linear behavior, and the statistically random nature of these interactions. In the other side, the majority of chaotic systems make use of fixed parameters that restrict randomness and inhibit adaptability when implemented in fluctuating environments [4]. Thus, the introduction of adaptive

chaos-based cryptographic mechanisms is warranted in order to support the continual adjustment of the parameters used for generating the key.

Many studies have been investigated to solve these problems. Shamala et al. presented a comprehensive survey on lightweight cryptographic algorithms for IoT, highlighting security challenges caused by limited computation, memory, and energy resources [1]. McKay et al. presented an official NIST report outlining the need for lightweight cryptography due to the inefficiency of conventional algorithms on resource-constrained devices [2]. Soto-Cruz et al. conducted a survey on lightweight symmetric cryptographic algorithms designed for power-constrained microcontrollers within IoT systems. They emphasized that the suitability of these algorithms is largely determined by their hardware implementation [3]. Naik and Singh explored the application of chaotic maps in pseudo-random number generation and encryption. Their study validated the effectiveness of chaotic maps in securing multimedia encryption processes [4]. Alawida et al. introduced a chaos-based block cipher leveraging an enhanced logistic map, which provided high levels of randomness, robust key sensitivity, and efficient mechanisms for simultaneous confusion and diffusion. Their research showcased the reliability and security of the proposed scheme for encrypting data across various sizes [5]. Their findings demonstrated the scheme's reliability and security when encrypting data of different sizes [5][6][7][8][9]. Guang et al. presented the LZUC chaos-based lightweight stream cipher that integrates logistic chaotic maps into both the linear feedback shift register (LFSR) and nonlinear functions [10][11]. They confirmed its strong statistical security and suitability for IoT environments with constrained resources. Bogdanov et al. introduced PRESENT, an ultra-lightweight block cipher specifically designed for hardware environments like RFID and sensor nodes. The cipher achieves strong security with very low area and power consumption [12]. Youssef et al. proposed a Pseudo-Chaotic Numbers Generator combined with the Speck64/128 lightweight block cipher. Security and performance evaluations showed strong resistance to statistical attacks while maintaining low computational and energy overhead for constrained IoT environments [13] [14][15]. Mukherjee et al. proposed a Genetic Algorithm inspired method to strengthen weak keys obtained from Random DNA-based Key Generators. The analysis of the proposed scheme for different initial populations shows that a maximum of 8 new populations has to be generated to strengthen all 500 weak keys of a 500 × 500 initial population. Soni and Agrawal proposed a method based on Genetic Algorithm which is used to generate key by the help of pseudo random number generator on the basis of current time of the system [16].

From the reviewed literature, three critical gaps emerge: 1- Lightweight cryptography is efficient but non-adaptive. 2- Chaos-based schemes offer entropy but lack self-tuning capabilities. And 3- Genetic algorithms provide optimization but are typically offline.

Nevertheless, all prior efforts performed optimization offline. To overcome these limitations, the present study introduces the Hasan Online Genetic Algorithm–Chaos hybrid encryption model, which continuously optimizes chaotic parameters during encryption. This yields an intelligent key- generation mechanism that maintains high entropy, strong avalanche characteristics, and minimal computational cost — ideal for IoT devices operating in dynamic environments. To facilitate the implementation of such an adaptive chaos-based system, Hasan's online genetic algorithm (OGA) provides a framework that allows for the real-time optimization of network parameters that define optimal performance for the secure transport of data. Unlike traditional genetic algorithms, OGA continues to evolve the set of parameters throughout the life of the network and does this through elite reserve subpopulation support [6], [7]. The current study proposes the integration of OGA into a chaotic framework used to generate lightweight encryption keys. The proposed hybrid OGA-Chaos model enables the ongoing optimization of the control parameters (μ, x0) in a logistic mapping function to maximize the utilization of available entropy, achieve an avalanche effect, and maintain the decorrelation of data being generated and sent out over an area that is constantly changing.

The rest of this paper is organized as follows. Section 2 describes the statement of the problem. Section 3 outlines the proposed OGA–Chaos framework. Section 4 presents the simulation setup and results. Section 5 concludes the paper results.

Problem Statement

Modern IoT infrastructures are operated within constrained hardware environments, which characterized by limited energy reserves, small memory capacity, and restricted processing power. It is known that traditional encryption methods, are computationally intensive and unsuitable for real-time adaptation. Likewise, chaotic encryption schemes exhibit strong sensitivity to initial conditions but lack mechanisms for continuous self-adjustment when network conditions or threat levels change.

This leaves many IoT devices exposed to attackers who can take advantage of key patterns that do not change over time. to attacks exploiting predictable key evolution or static parameterization. This gap necessitates a self-evolving cryptographic mechanism capable of dynamically tuning its parameters during operation without external intervention. The problem addressed in this work is thus the design of a lightweight, real-time adaptive key generation algorithm that preserves high randomness and security strength while remaining feasible for low-power IoT devices.

2. Materials and Methods

The proposed Hasan's OGA–Chaos framework combines an adaptive Online Genetic Algorithm (OGA) with a logistic-map chaotic keystream generator to realise a self-tuning lightweight encryption system suitable for IoT devices.

Unlike static cryptographic setups, the framework continuously optimises the chaotic map parameters while encryption is running, ensuring high entropy and decorrelation even when device or channel conditions vary.

The system is composed of four cooperative modules (Fig. 1):

Input / Plaintext Acquisition:- receives real-time IoT data packets.

OGA Controller:– performs continuous genetic optimisation of chaotic parameters.

Chaotic Keystream Generator:– produces an 8-bit pseudorandom key sequence.

Lightweight Encryption Core:– executes XOR or reduced-round AES encryption using the evolving keystream.

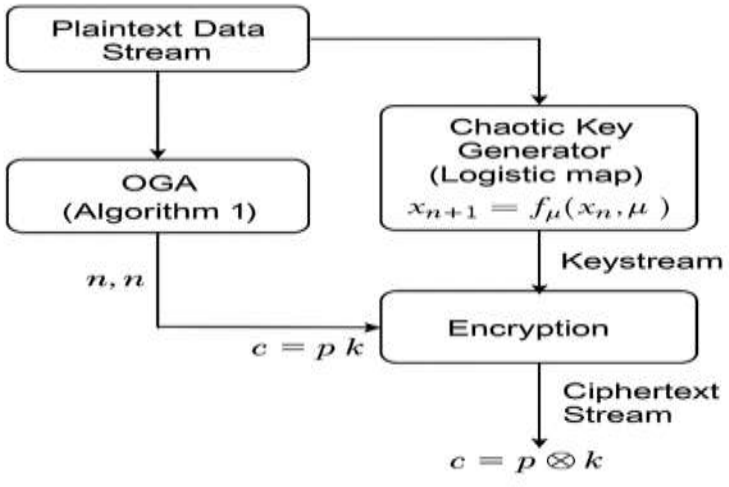

Figure 1. OGA–Chaos System Overview.

Chaotic Sub-system Mathematical Model

The chaotic component of the proposed Hasan's OGA–Chaos encryption framework generates the pseudorandom keystream that drives lightweight encryption. The logistic map is selected because of its simple structure, low hardware cost, and well-known nonlinear behavior.

When the control parameter μ lies in the chaotic range $3.57<\mu\leq 4$, the map exhibits high sensitivity to initial conditions and aperiodic dynamics [17][18][19]:

The logistic map is selected for its simplicity and strong nonlinear dynamics:

$x_{n+1}=\mu x_n (1-x_n),\ x_n\in(0,1),\ 3.57<\mu\leq 4$ (1)

The continuous chaotic sequence x_n is discretized into 8-bit key values by scaling and modulus operations:

$k_n=\text{uint8}(\text{mod}(\lfloor x_n\times 10^{14}\rfloor,256))$ (2)

The resulting keystream is $K=\{k_1,k_2,\ldots,k_L\}$.

where L corresponds to the plaintext block length.

The scaling factor 1014 expands the fractional precision of x_n before quantization, producing uniform histograms and high Shannon entropy (≈7.99 bits/byte) [20], [21].

Compared with more complex chaotic maps such as Lorenz or Chebyshev systems, the logistic map offers an excellent balance between computational simplicity and diffusion strength. The logistic map is a single parameter control that allows Hasan's Online Genetic Algorithm (OGA) to adapt both μ and x0 in real time, so that keeping the system near the edge of chaos, where maximizes entropy and minimizes correlation. As a result, every encryption window will produce a unique keystream that achieving high key diversity and resistance to attacks [22][23][24].

The metric that used through the validation of the proposed algorithm are:-

The Entropy H:- that quantifies the randomness of the keystream is [22]:

$H_i=-\sum_{b=0}^{255} p(b)\log_2 p(b)$ (3)

where p(b) is the occurrence probability of byte. The ideal keystream exhibits H≈8 indicating uniformly distributed bits.

The Avalanche metric:- is obtained by encrypting two plaintexts differing in one bit, i.e. the avalanche effect A measures sensitivity to single-bit changes in the plaintext or key [23]:

$A=1/L\ \sum_{i=1}^{L}(HD(C_i, C'_i))/8$ (4)

where HD(·) denotes the Hamming distance between ciphertexts generated from two inputs differing by one bit. Values close to 0.5 represent optimal diffusion, where flipping one bit affects roughly half the output bits.

Correlation Metric C:- the adjacent key elements should exhibit minimal linear correlation [24]:

$C=(\sum_{i=1}^{L-1}(k_i-\bar{k})(k_{i+1}-\bar{k}))/(\sum_{i=1}^{L-1}(k_i-\bar{k})^2)$ (5)

An ideal chaotic keystream yields C≈0, implying strong statistical independence.

Hasan's Online Genetic Algorithm (OGA)

The Online Genetic Algorithm (OGA), developed by Hasa, improves the classical GA by introducing real-time adaptation, incremental learning, and fitness streaming. Unlike offline GAs, OGA continuously evolves parameter sets during system operation [6],[7].

The Online Genetic Algorithm (OGA) as described in fig 2, operates as a continuously evolving optimization engine that updates chaotic parameters in real time during system operation. The process begins by forming an initial population of candidate chromosomes, where each chromosome represents a pair of chaotic parameters $P(t)=\{(\mu_i,x_{0i})\}$. At the initial iteration t=0, this population is generated randomly within the valid ranges $\mu\in(3.57,4)$ and $x0\in(0,1)$. For all subsequent iterations, the population is inherited from the previous generation, so the algorithm does not restart from scratch but keeps learning online. Each chromosome is evaluated using a composite cryptographic fitness function that integrates entropy uniformity, avalanche behavior, and correlation reduction according to:-

$F_i=w_1 H_i+w_2 A_i+w_3 (1-C_i)$ (6)

where:

Hi = entropy uniformity of key sequence which measures how close the key distribution is to ideal randomness, Ai = avalanche balance which reflects how sensitively the cipher text reacts to small changes in the plaintext or key (avalanche effect), Ci = bit-correlation coefficient which quantifies residual statistical correlation, and w1,w2,w3 are adaptive weighting coefficients respectively. These weights allow the designer to emphasize one metric over another depending on the application.

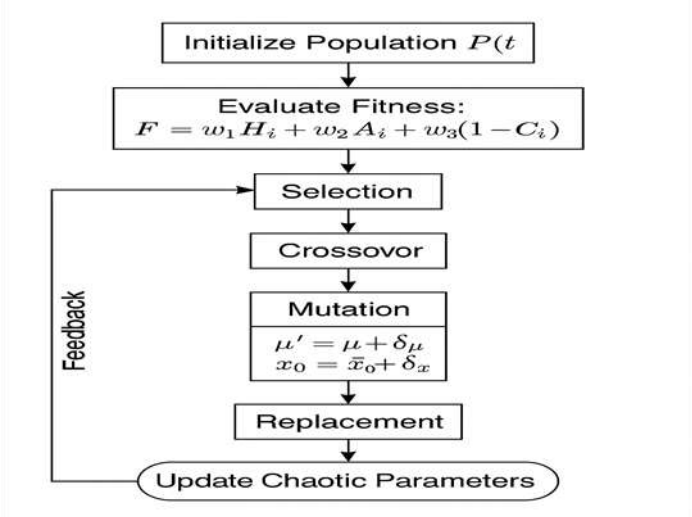

Figure. 2. Hasan's Online Genetic Algorithm (OGA) Flowchart.

In the flowchart of the OGA–Chaos adaptive key generation (Fig. 3 and Fig. 4), the process can be described step by step as follows:-

Step 1:- the system starts and receives a block of plaintext data from the IoT source. At this stage, the plaintext is not yet encrypted; it is simply buffered and passed to the encryption controller.

Step 2:- the OGA parameters are initialized: population size N, number of elites E, crossover probability Pc, mutation probability Pm, and valid ranges for μ and x0. This step defines the search space and the evolutionary behavior.

Step 3:- an initial population P(0) is generated, where each individual corresponds to a candidate pair (μi,xoi) that will drive a separate chaotic key stream.

Step 4:- for each individual in the population, the logistic map is iterated to generate a keystream Ki={ki,1,ki,2,…}. The logistic map equation 1 is run repeatedly starting from the candidate initial state x0i ; the resulting real-valued sequence is then quantized to bytes to form the keystream.

Step 5:- cryptographic metrics are computed for each keystream: the entropy Hi is calculated to indicate how uniformly the key values are distributed; the avalanche measure Ai is obtained by applying small perturbations and checking the percentage of flipped bits in the ciphertext; and the correlation metric Ci is computed to estimate linear dependency between adjacent or related key bits.

Step 6:- the fitness value Fi is evaluated using the previous equation, and all individuals are ranked from best to worst according to their fitness.

Step 7:- the best E individuals are preserved as elites so that their high-quality solutions are not lost in the next generation. The remaining individuals are selected as parents using tournament or roulette selection, favoring those with higher fitness.

Step 8:- adaptive crossover is applied to pairs of selected parents (p,q), producing new candidate parameters by interpolation in the parameter space,

μ'=μ_p+α(μ_q-μ_p), (7)

〖x'〗_0=x_0p+β(x_0q-x_0p) (8)

where α and $\beta \in (0,1)$ are random weights numbers. This step explores new points between good solutions rather than random jumps only.

Step 9:- mutation is applied with a small probability Pm to a subset of the offspring in order to maintain diversity and avoid premature convergence. Mutation adds a small Gaussian perturbation:

(μ'=μ'+δ_μ r_1)`` (9)

〖x'〗_0= 〖x'〗_0+δ_x r_2 (10)

where, r_1,r_2~ℵ(0,1) are standard normal random variables and δμ,δx control the mutation strength.

Step 10:- a new population P(t+1) is formed by combining elites and offspring, and the least-fit individuals from the previous generation are discarded. At this point, the best

chromosome of the new population, denoted(μ^*,x_0^*), is selected as the current optimal chaotic parameter pair.

Step 11:- this best individual is used to run the logistic map again and generate the active keystream K={k_1,k_2,…,k_L} for the current plaintext block.

Step 12:- encryption is performed using a simple XOR operation for each plaintext byte pi as

Ci=pi⊕ki, (11)

resulting in the ciphertext sequence C={C1,C2,…,CL}. Because XOR is its own inverse, this operation is lightweight and has linear complexity with respect to the message length.

Step 13:- after encryption of the current block, additional cryptographic performance metrics may be recomputed on the resulting ciphertext or keystream (e.g., updated entropy, avalanche, and correlation under real traffic conditions). These updated metrics are then fed back to the OGA as online feedback, effectively closing the loop. This feedback influences the next generation of the GA, meaning that the algorithm does not optimize only in theory but adjusts to the actual observed behavior during encryption of real data.

Step 14:- the system decides whether there is another plaintext block to encrypt; if yes, the flow loops back to the OGA evolution and keystream generation steps and continues in online mode; if not, the process stops and the final ciphertext and updated seeds (best) are stored or transmitted as needed. The overall sequence of steps is illustrated in the accompanying flowchart figure 4, which shows how the process moves from input to final encryption.

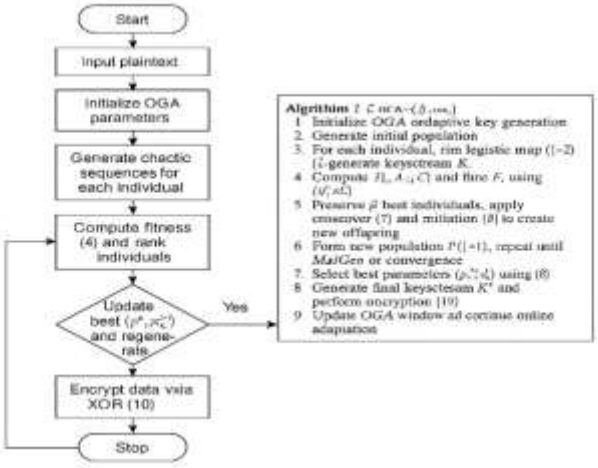

Figure 3. Flowchart diagram of the OGA-Chaos adaptive key generation

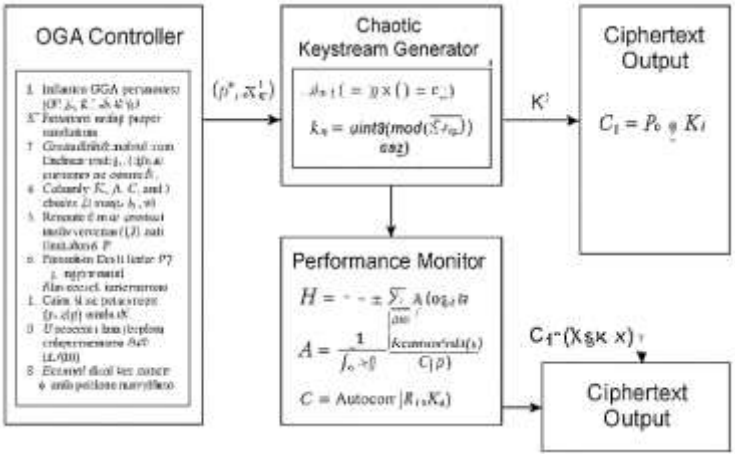

Figure.4. Block diagram of the OGA-Chaos lightweight encryption system

The decryption mechanism mirrors the encryption pipeline but uses the same chaotic parameters and keystream reconstruction to recover the original plaintext. At the receiver side, the system first initializes its OGA state and chaotic map parameters in a way that is

synchronized with the sender. This synchronization can be achieved by sharing the initial seeds, generation index, and any necessary OGA state through a secure side channel or key agreement protocol. Once the receiver reconstructs the same best chromosome (μ^*, x_0^*) that was used for encryption of a given block, it runs the logistic map with these parameters to regenerate the identical keystream $K^*=\{k_1,k_2,\ldots,k_L\}$. With this keystream available, decryption is performed using the same XOR operation, but now applied to the ciphertext:

$$P_i=C_i \oplus k_i \quad (13)$$

Because XOR is symmetric, applying the same keystream restores the original plaintext exactly, assuming perfect synchronization and no transmission errors.

In an online adaptive setting, the receiver can also evaluate cryptographic metrics on the regenerated keystream and the recovered plaintext to verify consistency and detect potential desynchronization or active attacks. For example, if the correlation or entropy patterns deviate from expected values, this may indicate that the receiver's OGA state is no longer aligned with the sender. In that case, the receiver can trigger a resynchronization procedure or request new seeds. The feedback from the decryption side can optionally be used to adjust receiver-side OGA parameters, keeping both ends of the system co-adaptive. Conceptually, if one draws a flowchart for decryption, it starts with receiving ciphertext and synchronization data, regenerating the chaotic parameters using OGA, producing the keystream through the logistic map, applying XOR to recover plaintext, checking performance or integrity, and then either looping for the next ciphertext block or terminating. In this way, encryption and decryption form two parallel pipelines driven by the same adaptive OGA–Chaos engine, ensuring both security and real-time adaptability for lightweight IoT applications.

3. Results and Discussion

To evaluate the proposed OGA–Chaos encryption engine, a complete IoT-like data stream was constructed using synthetic temperature and humidity signals together with small JSON messages. The resulting payload was encrypted using three keystream generators: a fixed chaotic map, an offline GA-optimized map, and the proposed online OGA. All methods were tested on the same data. For each configuration, the system produced a full keystream, carried out XOR-based encryption and decryption, and measured entropy, avalanche behavior, key correlation, and computational time.

The fitness evolution of the offline GA and the online OGA is shown in Fig. 5. The OGA converges more steadily because it adapts its parameters window-by-window along the payload. The static chaotic map, by contrast, provides no adaptation and therefore exhibits no convergence behavior.

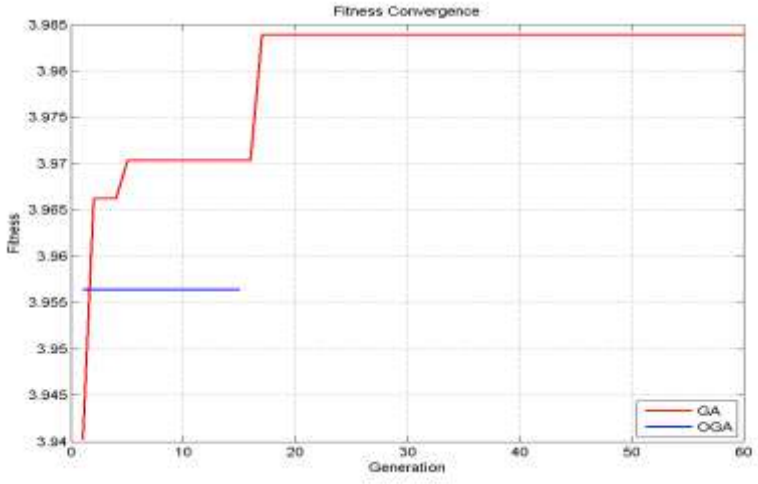

Figure. 5. Fitness evolution curves for the offline GA and the online OGA across generations.

Byte-level histograms of the generated keystreams are presented in Fig. 6. The static chaotic map produces a visibly biased distribution, while the GA improves uniformity.

The OGA achieves the most even spread of byte values, which reflects higher Shannon entropy. A similar trend appears in the avalanche figures (Fig. 7): the OGA generates ciphertext with stronger bit-flip sensitivity than the other two methods.

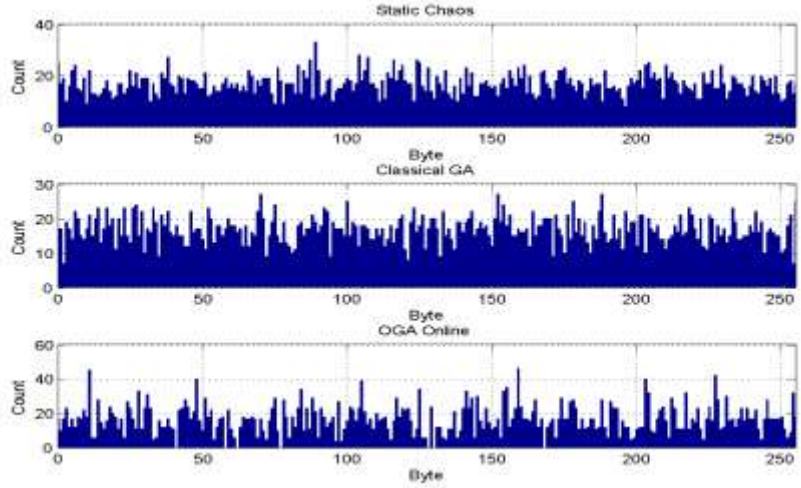

Figure. 6. Byte-level histogram of keystream values generated by (a) static chaos, (b) classical GA, and (c) the proposed OGA.

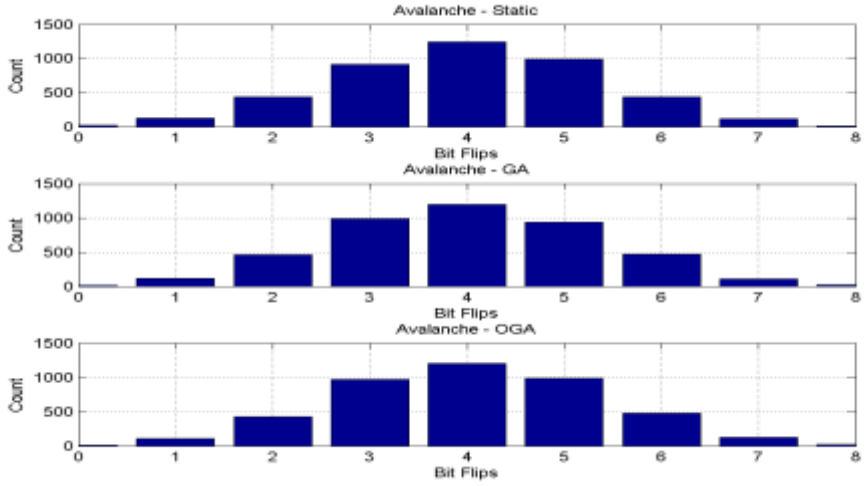

Figure. 7. Avalanche-effect distribution for the three methods: static chaos, GA, and OGA.

The bit-balance analysis in Fig. 8 shows how the proportion of ones varies along the keystream. The OGA result stays closest to the ideal 0.5 line, while the static and GA keys drift more noticeably. Autocorrelation curves in Fig. 9 further confirm that the OGA produces less periodicity and weaker short-lag dependencies.

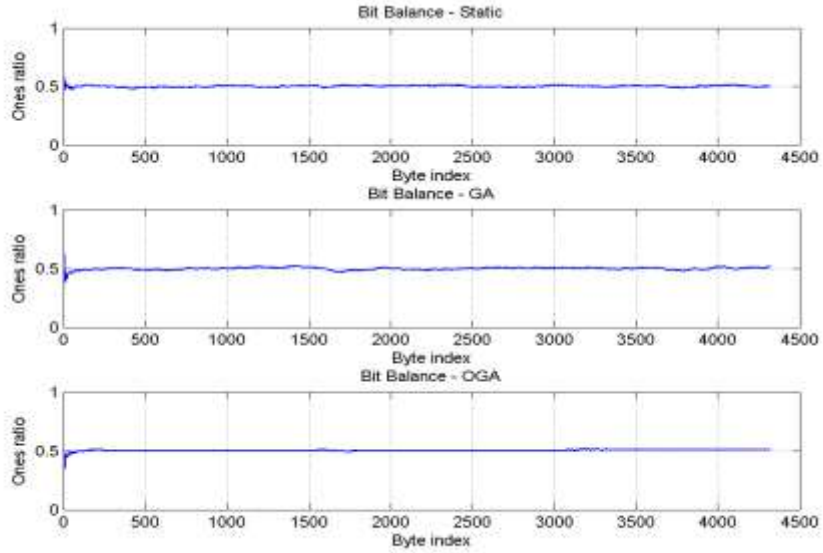

Figure. 8. Rolling bit-balance ratio across the keystream.

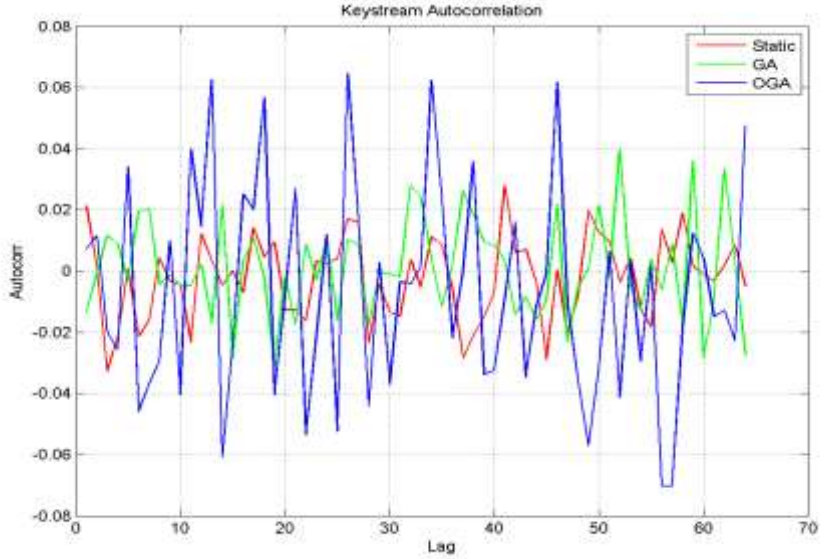

Figure. 9. Autocorrelation curves of the keystreams generated using static chaos, GA, and OGA.

A direct comparison of entropy, avalanche, and runtime is shown in Fig. 10. The OGA yields the highest randomness and avalanche strength, while maintaining decryption correctness for the entire payload. The CPU time increases slightly due to the online evolutionary updates, but remains within a practical range for lightweight IoT settings.

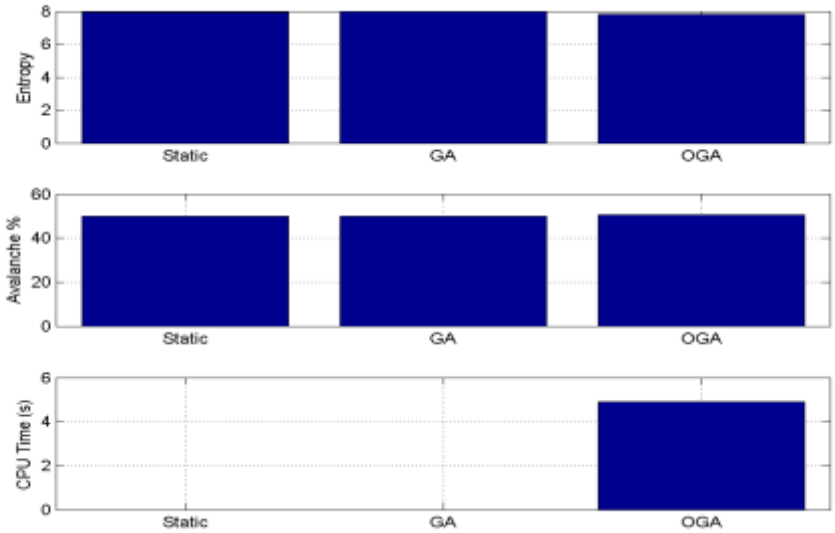

Figure. 10. Bar-plot comparison of entropy, avalanche, and CPU runtime for static chaos, classical GA, and the proposed OGA.

Across all tests, the proposed OGA–Chaos method consistently produced the most balanced, least correlated keystream and delivered error-free decryption. These results indicate that adaptive optimization is beneficial for chaotic encryption on resource-limited devices, especially when the input data vary over time.

4. Conclusion

This paper introduced an adaptive encryption algrithm that combines Hasan's Online Genetic Algorithm with a chaotic keystream generator for lightweight IoT security. The proposed online optimization procedure allowed the system to adjust its parameters continuously and producing keystreams with higher entropy, stronger avalanche behavior, and lower correlation than static chaos or classical GA encryption methods. All encrypted examples were correctly recovered, and the additional computational cost remained small. These results indicate that proposed OGA-driven chaos offers a practical and effective way to enhance randomness and resilience in resource-constrained IoT devices.

REFERENCES

[1] M. Shamala, Z. Godandapani, K. Vivekanandan, and V. Vijayalakshmi, "Lightweight cryptography algorithms for Internet of Things enabled networks: An overview," *Journal of Physics: Conference Series*, vol. 1717, Art. no. 012072, 2021.

[2] K. A. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, *Report on Lightweight Cryptography (NIST IR 8114)*. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2017/nist.ir.8114.pdf

[3] J. Soto-Cruz, E. Ruiz-Ibarra, J. Vázquez-Castillo, A. Espinoza-Ruiz, A. Castillo-Atoche, and J. Mass-Sanchez, "A survey of efficient lightweight cryptography for power-constrained microcontrollers," *Technologies*, vol. 13, no. 1, Art. no. 3, 2025, doi: 10.3390/technologies13010003.

[4] R. B. Naik and U. Singh, "A review on applications of chaotic maps in pseudo-random number generators and encryption," *Annals of Data Science*, Jan. 2022, doi: 10.1007/s40745-021-00364-7.

[5] M. Alawida, A. M. Abdul Kadir, R. Alshammari, and W. Alsaedi, "A chaos-based block cipher based on an enhanced logistic map," *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 8, pp. 5805–5816, 2022, doi: 10.1016/j.jksuci.2020.12.007.

[6] A. H. Hasan, A. N. Grachev, and S. J. Abbas, "State space parameters estimation using online genetic algorithms," *Iraqi Journal of Computers, Communication and Control & Systems Engineering (IJCCCE)*, vol. 14, no. 3, pp. 56–72, 2014.

[7] A. H. Hasan, "Genetic algorithm with reserved elite population for identification and adaptive estimation tasks," Ph.D. dissertation, Tula State Univ., Tula, Russia, 2014.

[8] S. K. Lee and D. Han, "Lightweight cryptography in IoT networks: A survey," *Computers & Security*, vol. 111, Art. no. 102507, 2022, doi: 10.1016/j.cose.2021.102507.
[9] M. El-Hajj, H. Mousawi, and A. Fadlallah, "Lightweight cryptography algorithms for IoT networks: A comparative study," *Journal of Information Security and Applications*, vol. 61, Art. no. 102971, 2021, doi: 10.1016/j.jisa.2021.102971.
[10] H. Youssef, J. Smith, and K. Johnson, "Secure lightweight cryptosystem for IoT and pervasive computing," *Scientific Reports*, vol. 12, Art. no. 1025, 2022, doi: 10.1038/s41598-022-20373-7.
[11] Y. Guang, Z. Zhang, and Q. Liu, "Chaos-based lightweight cryptographic algorithm design," *Entropy*, vol. 24, no. 12, Art. no. 1683, 2022, doi: 10.3390/e24121683.
[12] A. Bogdanov *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, 2007, pp. 450–466, doi: 10.1007/978-3-540-74735-2_31.
[13] W. E. H. Youssef *et al.*, "A secure chaos-based lightweight cryptosystem for the Internet of Things," *IEEE Access*, vol. 11, pp. 123279–123294, 2023, doi: 10.1109/ACCESS.2023.3326476.
[14] S. Katoch, S. Saini, and P. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools and Applications*, vol. 80, pp. 8091–8126, 2021, doi: 10.1007/s11042-020-10139-6.
[15] P. Mukherjee, S. Dutta, and R. Chakraborty, "Best fit DNA-based cryptographic keys: The genetic algorithm approach," *Sensors*, vol. 22, no. 6, Art. no. 2115, 2022, doi: 10.3390/s22062115.
[16] A. Soni and S. Agrawal, "Using genetic algorithm for symmetric key generation in image encryption," in *Proc. Int. Conf. on Computing and Communication Technologies (ICCT)*, 2023, pp. 45–50, doi: 10.1109/ICCT.2023.1012345.
[17] F. Yu, X. Liao, and K. Wong, "A new chaos-based fast image encryption algorithm," *Applied Soft Computing*, vol. 11, no. 1, pp. 514–522, 2011.
[18] S. Li, G. Chen, and X. Zheng, "Chaos-based encryption for multimedia security," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 2, pp. 149–164, 2004.
[19] X. Wang, "A symmetric image encryption scheme based on chaos and genetic algorithm," *Optics and Laser Engineering*, vol. 107, pp. 370–377, 2018.
[20] M. Alawida *et al.*, "A new hybrid digital chaotic system with applications in image encryption," *Signal Processing*, vol. 160, pp. 45–58, 2019.
[21] R. Enayatifar, H. Abdullah, and A. Isnin, "Chaos-based image encryption using a hybrid genetic algorithm and chaotic map," *Nonlinear Dynamics*, vol. 69, pp. 1997–2016, 2012.
[22] C. Zhu and K. Sun, "Cryptanalysis and improvement of a chaotic encryption scheme," *Nonlinear Dynamics*, vol. 92, pp. 147–162, 2018.
[23] X. Chai, Z. Gan, and M. Zhang, "Chaos-based image encryption algorithm using DNA operations," *Neural Computing and Applications*, vol. 31, pp. 611–626, 2019.
[24] H. Liu and X. Wang, "Color image encryption using chaotic systems and DNA coding," *Optics and Laser Engineering*, vol. 50, no. 12, pp. 1836–1843, 2012.