JOIV : Int. J. Inform. Visualization, 8(2) - May 2024 819-825

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Dynamic Key Generation Using GWO for IoT System

Balsam A. Hameedi [a,*], Muntaha A. Hatem [b], Jamal N. Hasoon [c]

[a] Palestine High School for Excellence, Ministry of Education-Iraq, Baghdad, Iraq
[b] Department of Missions and Cultural Relations, Ministry of Higher Education and Scientific Research-Iraq, Baghdad, Iraq
[c] Department of Computer Sciences, Faculty of Science, Mustansiriyah University, Falastin St, Baghdad, 10052, Iraq
Corresponding author: *Balsam119@res-rus2.edu.iq

***Abstract*—One well-known technological advancement that significantly impacts many things is the Internet of Things (IoT). These include connectivity, work, healthcare, and the economy. IoT can improve life in many situations, including classrooms and smart cities, through work automation, increased output, and decreased worry. However, cyberattacks and other risks significantly impact intelligent Internet of Things applications. Key generation is essential in information security and the various applications that use a distributed system, networks, or Internet of Things (IoT) systems. Several algorithms have been developed to protect IoT applications from malicious attacks; since IoT devices usually have small memory resources and limited computing and power resources, traditional key generation methods are inappropriate because they require high computational power and memory usage. This paper proposes a method of Dynamic Key Generation Method (DKGM) to overcome the difficulty using a specific chaotic map called the Zaslavskii Map and a swarm intelligent algorithm for optimization called Grey Wolf Optimizer (GWO). DKGM's ability to generate several groups-seed numbers using the Zaslavskii map depends on various initial parameters. GWO selects strong generated numbers depending on the randomness test as a fitness function. Three wolfs GWα, GWβ, and GWΩ, are used to simulate the behavior of a pack of grey wolves when attacking prey. The speed and position of each wolf are updated depending on the best three wolves. Finally, use the sets GW_{α} in the round, GW_{β} in the subkey, and GW_{Ω} in shifting operations of the Chacha20 hash function. The dynamic procedure was used to improve the high-security analysis of the DKGM approach over earlier methods. Simulations show that the suggested method is preferable for IoT applications.**

***Keywords*—Zaslavskii map; Grey Wolf Optimizer (GWO); chacha20 hash function; NIST test.**

Manuscript received 8 Oct. 2023; revised 12 Dec. 2023; accepted 14 Feb. 2024. Date of publication 31 May 2024.

I. INTRODUCTION

The term IoT is applied in various contexts throughout the world. Among the goals are minimizing human effort, increasing efficiency, swiftly comprehending client behavior, speeding up, increasing business value (decision support system), etc. However, confidentiality worries about software vulnerabilities and hacking may lead to many users avoiding IoT devices [1]. IoT device adoption, however, may be avoided by many users due to rising worries about software bugs and hacking [2]. A lightweight cipher has been a preferable choice for intelligent and sensor-based applications [1] due to the restricted devices utilized in the IoT (processing, memory, and power) [3]. Any algorithm for cryptography must include pseudorandom number generators. They are used in several processes, including one-time passwords (OTP), hashing, encryption, seed vectors, and digital signatures [4]. These pseudorandom number generators are deterministic, meaning that the output is determined by the starting seed sequence and the generator's layout [5]. Chaos may generate divergent and disorderly sequences that are random and predict tables while being too sensitive to begin circumstances [6]. The advancement of the dynamics system was represented by chaos theory which found widespread application [7]. One of the most well-known applications in this regard is incorporating chaos theory into optimization algorithms [7]. Several meta-heuristic optimization strategies have been integrated with success [8]. The GWO algorithm is a swarm-intelligent algorithm motivated by the distinct hunting technique of grey wolves. It is one of the metaheuristics that may help to prevent the stagnation of local optima [9]. It also has a high capacity to converge toward the optima. GWO, in general, lends itself well to exploitation. It cannot, however, reliably execute global search. As a result, in certain circumstances, GWO fails to discover the best global

solution. The fundamental GWO search approach is based on random motion to solve the common artificial intelligence issue [10] effectively.

This paper proposes a new dynamic key generation method called DKGM based on the Gray Wolf Optimizer (GWO) with a pseudorandom sequence generated using the Zaslavskii chaotic map. DKGM proposed the fitness function in the GWO algorithm to evaluate and select random numbers from the chaotic function and arrange them in three groups, which are used respectively in rounds, subkeys, and shifting operations of the ChaCha20 hash algorithm.

The paper's outline is as follows: Section 2 summarizes the most critical works on generating secret keys and their application in the IoT, based on chaotic functions. Sections 3, 4, and 5 illustrated the details of the Zaslavskii chaotic map, the GWO, and the Chacha20 lightweight hash algorithm. The flowchart and algorithm of the proposed DKGM are presented in Section 6. The results and experiment of the proposed algorithm are illustrated in Section 7. Finally, the conclusions of the work are shown in Section 8.

Many academics are curious about how secure IoT devices can communicate data with one another, and they have devised various methods. One of these solutions is chaotic maps. Naif et al. [11] developed an IoT-safe system combining a proposed 4D chaotic system with a modified lightweight Advanced Encryption Standard (AES). It has two primary functions: encryption (through chaos-modified lightweight AES) and authentication using hash methods (chaos-SHA3-256 bits). The results demonstrate that the suggested system's calculation time was reduced (by 145%), and it passed the NIST test package.

Guma'a et al. [3] propose modifying the NTRU public key cryptosystem to be safe against lattice-based attacks by using the LLL (Lenstra-Lentra-Lovász) algorithm, as well as a way for dynamically generating a new key sequence. According to the results, the suggested solution reduced the NTRU algorithm implementation time by around 0.2 seconds.

Vohra et al. [12] proposed a novel cryptography technique for encrypting and decrypting information utilizing the origin of an evolutionary algorithm GA. Seed numbers are generated and evaluated as a key in the GA operation for the forward cryptography process formed by a specific function in a chaotic map. This technique yields excellent, desirable cryptographic features since a change in the key produces undesirable consequences on the receiver side.

In Rahman et al. [13], a particular security technique was proposed to enhance security and preserve the integrity of much data transmitted within an IoT system. The authors suggest this technique be applied to an IoT cluster head. It is inspired by enhancing the AES generation technique that uses a three-dimensional Key Generation Mechanism (3DKGM) to elevate the complexity of the AES algorithm. The results showed that the method could only guarantee high data protection against linear and differential attacks. The technique also strengthened data encryption and reduced the encryption time.

Ali [14] proposed and tested a keystream generator based on 3D chaotic maps and "Particle swarm optimization" to generate a random number generator. Ismael and Maolood [15] concentrate on the proposed key for encrypting the data contained in the database using AI approaches and the equation that represented the "Bezier curves" used in the medical platform as an adaptive key. The Particle Swarm Optimization (PSO) technique works with a few networked devices, and the created keys are produced using the logistic map function, which passes NIST tests.

Zalinski Chaotic Map is presented by Zaslavsky [16], a discrete-time dynamical system. The chaotic behavior of the 2-D Zaslavsky map may create random real numbers. The pseudorandom numbers, on the other hand, may be generated using an iterative approach. Coupled Eq. (1) [17] is used to define the 2-D map:

$$\begin{cases} x_{u+1} = x_u + v(1+\mu y_u) + \varepsilon v\mu[cos(2\pi x_u)] mod\ 1 \\ y_{u+1} = e^{-\tau}[y_u + \varepsilon\, cos(2\pi x_u)] \\ \mu = \dfrac{1-e^{-\tau}}{\tau} \end{cases} \quad (1)$$

Initial values are x_0, and y_0, and ε, τ, and are the regulating parameters used to monitor the predicted chaotic behavior, while e is the exponentiation [17], [18]. When the parameters r = 3.0, v = 400/3, and ε = 0.3 are set of the parameters, this map shows chaotic behavior [19].

The swarm intelligence algorithm GWO presented by Mirjalilli, first invented in 2014 [20], simulates grey wolves' distinct hunting and prey-finding behaviors. Grey wolves have a four-level social structure that GWO has adopted, with α wolves at the first level, β at the second, δ at the third, and ω wolves at the last. The grey wolf pack is led and directed by its leader, the wolves [21]. It is also responsible for maintaining order, managing the entire hunting process, and deciding what to hunt, when to rest, and when to wake the pack. The top contender for the α position, the β wolf, provides the α leader with input from other wolves. The wolves of the fourth level are subservient to the wolves of the third level or δ wolves. The Omega wolves, the last rank, are in charge of preserving the integrity and safety of the wolf pack [22]. Equation (2) is used to determine the distances, D_α, D_β, and D_δ, between each of the remaining wolves and the wolves α, β, and δ.

The effect of α, β, and δ wolves on the prey [7] can be calculated using Eq. (3).

$$\overrightarrow{D_\alpha} = |\overrightarrow{C_1}\cdot\overrightarrow{X_\alpha} - \overrightarrow{X}|, \overrightarrow{D_\beta} = |\overrightarrow{C_2}\cdot\overrightarrow{X_\beta} - \overrightarrow{X}|, \overrightarrow{D_\delta} = |\overrightarrow{C_3}\cdot\overrightarrow{X_\delta} - \overrightarrow{X}| \quad (2)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\overrightarrow{D_\alpha}, \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.\overrightarrow{D_\beta}, \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.\overrightarrow{D_\delta} \quad (3)$$

$$\overrightarrow{A} = 2\overrightarrow{a}.\overrightarrow{r}_1 - \overrightarrow{a}, \overrightarrow{C} = 2.\overrightarrow{r}_2 \quad (4)$$

$$\overrightarrow{X}(t+1) = (\overrightarrow{X}_1 + \overrightarrow{X}_2 + \overrightarrow{X}_3)/3 \quad (5)$$

The algorithm's control parameters A and C are calculated using Eq. (4). The random vectors in this case are r_1 and r_2 they fall between [0, 1]. Wolves can travel through these vectors to any location between their prey and themselves. Vector "a" controls the activity of the GWO algorithm and is used to calculate "A," which decreases linearly in periods [2,0] through iteration [23]. C used to append weight on the prey, making it difficult for the wolves to find it. Finally, using Equation (5), all other wolves update their positions [24].

The Chacha stream cipher family is a high-throughput stream cipher designed for software platforms. In this part, we characterize ChaCha20, as defined by RFC7539, as a conservative security instantiation [22], [23]. The ChaCha20's total input size is 512 bits, as illustrated in Fig. (1). These bits serve as seeds; each one contains thirty-two bits, which are made up of [24,25]:

- A 256-bit function as the entire key size (k1... k8).
- The entire amount of the nonce (n1, n2) and constants [c1... c4] is 192 bits.
- The entire size of the block message counter is 64 bits (b1, b2).

c1	c2	c3	c4	▶	x0	x1	x2	x3
k1	k2	k3	k4		x4	x5	x6	x7
k5	k6	k7	k8		x8	x9	x10	x11
b1	b2	n1	n2		x12	x13	x14	x15

Fig. 1 Input of ChaCha20 [24]

Consequently, three methods combine the ChaCha20 input with the original data for ChaCha20 encryption to create a sequence of 512 bits representing the keystream through an XOR operation [26]. One lightweight approach is addition, which involves adding two 32-bit numbers. Another is exclusive OR, which consists of XORing two 32-bit numbers [27]. The third lightweight way is rotation, which involves supervising the rotation of 32-bit integers by the e bit [y <<e], where e functions as a constant number. [28]. The three elementary procedures are reduced to two dual functions. The central component of the dual function, the Quarter Round Function (QRF), is responsible for updating the state matrix after every round. QRF is applied to the columns of the state matrix before moving on to its diagonals, as shown in Fig. 2.

Column Form	Diagonal Form
QR (x0, x4, x8, x12)	QR (x0, x5, x10, x15)
QR (x1, x5, x9, x13)	QR (x1, x6, x11, x12)
QR (x2, x6, x10, x14)	QR (x2, x7, x8, x13)
QR (x3, x7, x11, x15)	QR (x3, x4, x9, x14)

Fig. 2 Quarter-round function of ChaCha20 [24]

QRF takes four 32-bit inputs and modifies the outputs depending on the three lightweight techniques stated in Eq. (6,7,8, and 9) [29]. An additional operation between the most recently modified matrix and the input's initial seed constitutes the final stage of ChaCha20 encryption algorithm [30].

$$a = a + b, d = (d \oplus a) \lll 16 \quad (6)$$

$$c = c + d, b = (b \oplus c) \lll 12 \quad (7)$$

$$a = a + b, d = (d \oplus a) \lll 8 \quad (8)$$

$$c - c + d, b - (b \oplus c) \lll 7 \quad (9)$$

II. Materials and Methods

The proposed method is used in IoT applications that need to generate a vital key for immunity purposes. The process is represented by using the Zaslavskii map to generate numbers by controlling the initial values used in equations to create numbers. Different numbers are generated, and the most random set of numbers is selected by applying GWO after several iterations. A shifting operation is performed for these numbers before they are used in the Chacha function [29], and the result is used as keys in multiple applications that require crucial generation, as explained in Fig. 3.

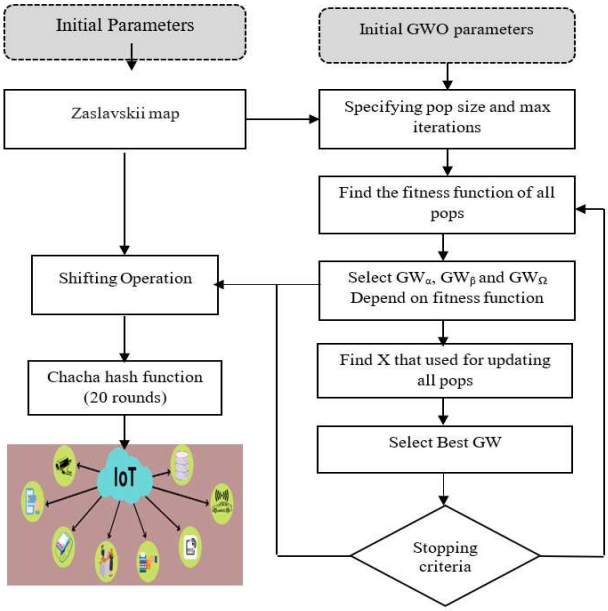

Fig. 3 General Framework of Proposed Method

A. Applying Zaslavskii Map

The second-order equations of the Zaslavskii map are applied, which need initial values and control values, and these values control the series of numbers generated, which are numbers of a real type that are later processed and converted into binary numbers that are examined [30], as shown in Table 1.

TABLE I
GENERATED NUMBERS USING ZASLAVSKII MAP

#	First dimension	Second dimension	#	First dimension	Second dimension
1	0.908898478	-0.072964052	7	0.553737282	0.11935367
2	0.622973332	0.55893907	8	0.30046067	-0.6223176
3	0.544474361	-0.408892309	9	0.315076185	-0.2951879
4	0.222860721	-0.705765605	10	0.318391236	-0.3089872
5	0.495027491	0.019314246	11	0.309634718	-0.3237281
6	0.199646495	-0.67373228	12	0.32729623	-0.2914683

These values from the previous TABLE are converted to decimal values by getting the digits after floating point in fixed numbers for all values (10, 11, 12, 15, or more digits), as explained in Table 2.

TABLE II
DECIMAL VALUES OF GENERATED NUMBERS

#	First dimension	Second dimension
1	908898477721977	72964052410067
2	622973332344945	558939069814865
3	544474360695031	408892309107034
4	222860721491073	705765604862273
5	495027490802712	19314245746619
6	199646494751517	673732279569791
7	553737281641506	119353677117366
8	300460669988326	622317580951473
9	315076184992073	295187920912175
10	318391235860893	308987195407013
11	309634718342016	323728084256895

As explained in Table 3, decimal values are converted to hexadecimal values. These numbers satisfy the requirements for generated keys used in the security requirements.

TABLE III
THE HEXADECIMAL FORM OF GENERATED NUMBERS

#	First dimension	Second dimension
1	33AA36AE6C979	0425C446CDED3
2	23697482DD071	1FC5A24B51851
3	1EF3250DBC8F7	173E2A8D5F15A
4	0CAB0CE238C81	281E3DC2C2941
5	1C23991BE4C18	01190F2C1BFBB
6	0B593D2667B1D	264C185328D7F
7	1F79F01F89C22	06C8D3186F7B6
8	111447380EDE6	235FE9A599FB1
9	11E8F64156949	10C78CB2AFF2F
10	121933C83599D	11905B005BAA5
11	1199C73659B80	1266DD177487F
12	129AC97844F4C	10916C53C662F

B. Applying GWO Algorithm

At this stage, a set of initial values is used to create a set of sequences using the Zaslavskii map. These numbers are real, and the characteristics of them are chaotic phenomena. All these sequences are converted into decimal form, converted to binary form, and differentiated between them through the target function, which counts the number of zeros relative to the number of ones. When the two values are roughly equal, the best result is obtained. The best three sequences are selected and considered as α, β, and δ Gray-Wolfe, and all existing values are updated depending on these values. The evaluation of all sequences is measured via the fitness function by converting the generated numbers into binary numbers and measuring the number of zeros, the number of ones, and the series containing the best converging values. The GWO algorithm is represented by selecting the best three solutions as G_{α}, G_{β}, and G_{δ} are done through the fitness function and the update process is done through the set of equations, and a set of numbers is generated against each set, and the fitness function of the new set is found if it is better to be adopted and if it is worse, it is neglected and the best solution is adopted after a set of iterations. In each iteration, a new gray-Wolfe is obtained that is compared with the previous one, and the Greedy selection is applied to select the best from the current and last solution.

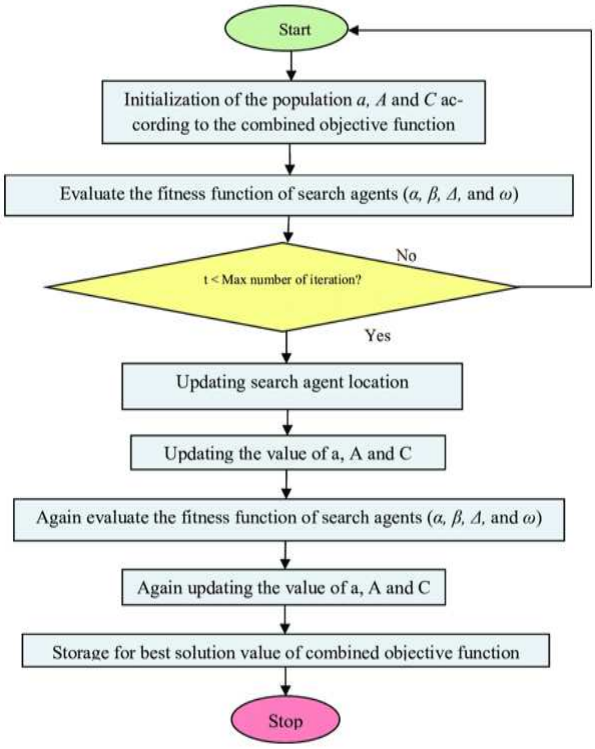

Fig. 4 GWO Flow Chart of Proposed Method

C. Shifting Operation

When generated numbers are entered into the Chacha algorithm, the shifting operations used in the results are shifted. This process of shifting is of a different amount each time to increase complexity and for the final numbers to become different in the case of obtaining the same generated sequences, and this process uses one of the dimensions used in moderate key generation

D. Chacha Function

The hash functions in the Chacha function are used as the final stage in the generation process as it needs a set of inputs and performs a set of special operations for a specific number of cycles, and the last result is generated numbers that can be used in encryption processes and applications that need numbers as keys.

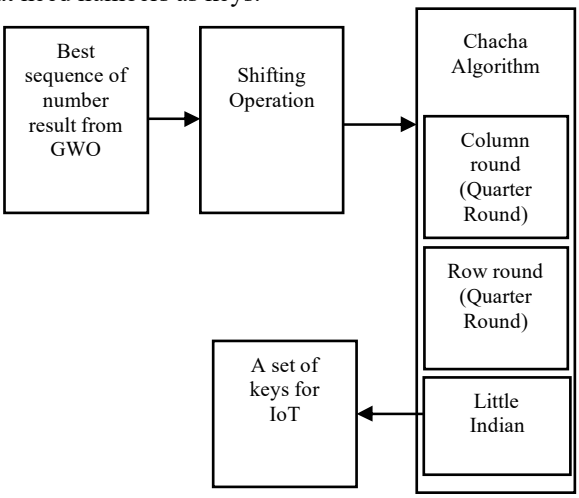

Fig. 5 Chacha Hash Function in Proposed Method

The ChaCha cipher is a stream cipher designed to be secure, fast, and simple to implement. It is a variant of the Salsa20 cipher, which was itself derived from the original ChaCha cipher. The ChaCha cipher uses a block size of 64 bytes, which is divided into 16 32-bit words. The key and nonce are also divided into 32-bit words and are used to initialize the state of the cipher, as shown in Table 4.

TABLE IV
THE INITIAL STATE OF CHACHA

Constant	Constant	Constant	Constant
Key	Key	key	Key
Key	Key	key	Key
input	input	input	Input

The cipher consists of 20 rounds of operations, which are performed on the state using three main processes: addition, XOR, and rotation. In each round, the state is first updated by adding the keywords and constants to the state. Next, the state is XORed with the input block. Finally, the state is rotated by a certain number of bits, and the results are added to the keywords. This process is then repeated for a total of 20 rounds. The ChaCha cipher is a strong and secure stream cipher, and it is widely used in various applications, including encryption of internet traffic, secure communication, and more.

OR(X0,X4,X8,X12) OR(X1,X5,X9,X13) OR(X2,X6,X10,X14) OR(X3,X7,X11,X15)	OR(X0,X5,10,X15) OR(X1,X6,X11,X12) OR(X2,X7,X8,X13) OR(X3,X4,X9,X14)
a=a+b, d=(d⊕a)≪16 c=c+d, b=(b⊕c)≪12 a=a+b, d=(d⊕a)≪8 c=c+d, b=(b⊕c)≪7	

Fig. 6 Quarter Round Function and Processing Function in Proposed Method

III. RESULT AND DISCUSSION

The proposed method is evaluated with multiple tests to ensure its efficiency and applicability. The Zaslavskii Map is a two-dimensional function that generates two seemingly random sequences of floating numbers, as shown in Fig. 7.

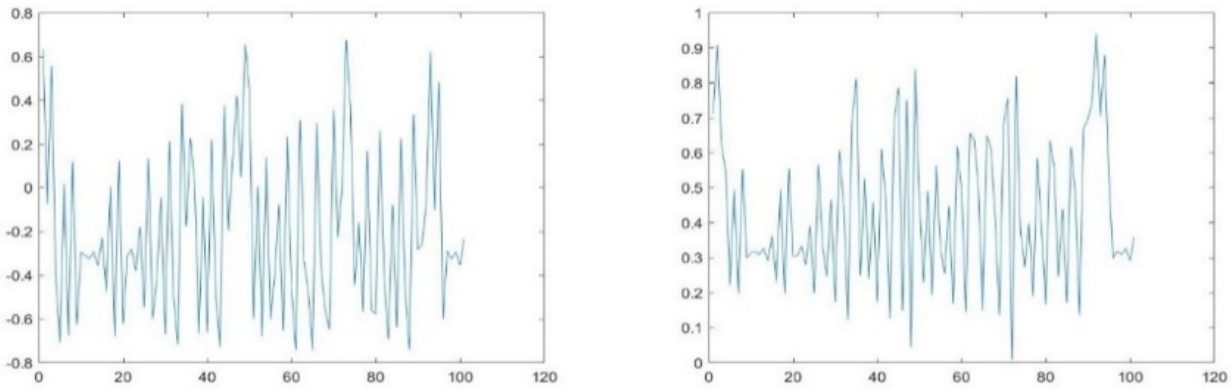

Fig. 7 The Visualization of a Two-Dimensional Zaslavskii Map

The numbers generated by the Zaslavskii map are processed and converted into a series of bit-streams. These bit streams are subjected to NIST tests to ensure their randomness. The total results of the NIST test are explained in Table 5.

TABLE V
THE HEXADECIMAL FORM OF GENERATED NUMBERS

Test number	P-Value (minimum)	P-Value (Mean)	P-value (maximum)	Status
1	0.00460	0.01297	0.01697	Pass
2	0.00015	0.00315	0.00214	Pass
3	0.01672	0.05486	0.01856	Pass
4	0.05953	0.04725	0.11200	Pass
5	0.01220	0.04363	0.17485	Pass
6	0.00002	0.00016	0.00052	Pass
7	0.00000	0.00001	0.00000	Pass
8	0.00006	0.00008	0.00009	Pass
9	0.00175	0.00146	0.00040	Pass
10	0.00021	0.00356	0.00949	Pass
11	0.01820	0.11482	0.03854	Pass
12	0.00042	0.00046	0.00099	Pass

The NIST test (test keys are the keys used in each experiment). A collection of statistical tests for assessing variables include the run test, serial test, random excursion variant test, random excursion test, frequency mono bit test, non-overlapping template matching test, universal Maurer's statistical test, the longest run of ones in a block test, linear complexity test, frequency test within a block test, discrete Fourier transform test, and cumulative sums test.

These tests aim to verify that the key generation technique is safe from various attacks. The randomness of the critical generation process, which is essential for preserving the security of the generated key, may be evaluated using the findings of the NIST testing. Because different statistical tests may identify various types of biases or patterns in the generated numbers, multiple tests are frequently employed to evaluate the quality of a random number generator. The final sequences produced by the suggested method are examined by determining the correlation between each to clarify any weak correlations in Table 6.

TABLE VI
THE CORRELATION TEST OF GENERATED NUMBERS

Seq#	1	2	3	4	5	6	7	8	9	10
1	0.001	0.121	0.189	0.126	0.102	0.056	0.184	0.078	-0.019	0.161
2	0.091	0.01	0.105	0.086	0.121	0.261	0.137	0.117	0.011	0.192
3	0.185	0.137	0.001	0.054	0.143	0.036	0.095	0.114	0.032	0.262
4	0.008	0.151	0.066	0.01	0.201	0.187	0.171	0.134	0.208	0.002
5	0.049	0.142	0.084	0.215	0.001	0.004	0.021	0.146	0.091	0.006
6	0.117	0.232	0.061	0.104	0.072	0.001	0.161	-0.034	0.053	0.081
7	0.219	-0.012	0.239	0.201	0.098	0.161	0.001	0.104	0.151	0.013
8	0.079	0.059	0.262	0.131	0.201	0.108	0.091	0.001	0.139	0.161
9	0.107	-0.013	0.049	0.141	0.109	0.141	0.023	0.097	0.001	0.152
10	0.181	0.078	0.223	0.101	0.069	0.195	0.006	0.109	0.203	0.001

Independent output bits are a technique used to evaluate the security of the generated sequences. This is a fundamental component of many cryptographic algorithms as it can help identify weaknesses that could potentially be exploited by attackers. It is just one of many techniques that can be used to evaluate the security of cryptographic algorithms, and it is essential to use various techniques to assess the security of any given algorithm thoroughly.

Table 7 shows how the suggested algorithm compares against some of the works described in the previous related works based on various criteria, such as the random number generation algorithm, the encryption method, and the standards employed. It also shows how the proposed algorithm improves security and performance when used in Internet of Things use cases.

TABLE VII
COMPARED THE PROPOSED ALGORITHM WITH SOME RELATED WORKS

Reference	Title	Methodology	disadvantage
Naif et al. [11]	"Internet of Things security using the new chaotic system and lightweight AES"	a secure system evaluated by NIST standard requirements and used the modified lightweight Advanced Encryption Standard (AES) in conjunction with the (4D) chaos system Lyapunov.	Not suitable for big data
Guma'a et al. [3]	"Dynamic key generation for the Internet of Things"	modifications to the NTRU public key cryptosystem	High computation
Vohra et al. [12]	"An efficient chaos-based optimization algorithm approach for cryptography"	Dynamic generation pseudorandom number used 2d Hénon chaotic map and genetic algorithm for cryptography data	High computation
Rahman et al. [13]	"Chaos and logistic map-based key generation technique for AES-driven IoT security"	3 D-Dynamic key generation method (3DKGM) based on logistic chaotic and Linear Feedback Shift Register for encoding data using LZ78 Algorithm and tested by NIST standard criteria	3DKGM is unable to generate a key of length greater than 600 bytes
Ali [14]	Random Number Generator based on Hybrid Algorithm between Particle Swarm Optimization (PSO) Algorithm and 3D-Chaotic System and its Application	Proposed 3D-LMPSO method for Generation key stream based on using a non-linear 3D chaotic map and PSO algorithm and tested by five standard criteria	Time encryption is not known
Ismael and Maolood [15]	Proposed Secure Key for Healthcare Platform	Using Bezier Curve, Logistic Map Chaotic, and PSO to generate an encrypted key and test the proposed method using five NIST tests	It has not been treating different types of attacks
Our proposed algorithm	Dynamic Key Generation using GWO for IoT System	Proposes a dynamic key generation method (DKGM) using the Zaslavskii map and the Grey Wolf Optimizer (GWO) and a testing method using 11 NIST tests	-

IV. CONCLUSION

Information security is becoming increasingly crucial with the abundance of valuable information available today. Organizations are aware of the true meaning of information security and why every organization should begin implementing it. It is always related to the key generation required to be high randomness. The traditional key generation methods didn't satisfy the security. To overcome the challenge, this work presented a GWO, Zaslavskii Map, and Chacha hash function as a dynamic key generation method. DKGM generates multiple groups-seed numbers depending on different initial values. GWO is a tool for choosing robust produced numbers. The proposed method uses the Zaslavskii map to generate many random numbers and then uses the Grey Wolf Optimizer (GWO) to select the optimal groups of numbers from this set. The chosen groups are then used in the Chacha20 hash function in terms of rounds, subkeys, and shifting operations to generate the final key. The use of the dynamic process is intended to enhance the security of the proposed method compared to existing methods. The best solution in the final step is obtaining the final key. The resulting key is tested using the NIST test and passes all tests that provide suitable keys for various applications in IoT systems. The correlation test is applied to the sequences of generated numbers to find dependencies.

The test simulation results suggest that the proposed method is effective and suitable for use in IoT applications and could be combined with several lightweight encryption methods to enhance the system's performance.

REFERENCES

[1] Z. Rahman, X. Yi, M. Billah, M. Sumi, and A. Anwar, "Enhancing AES Using Chaos and Logistic Map-Based Key Generation Technique for Securing IoT-Based Smart Home," Electronics, Vol.11, no.7, pp. 1083,2022.

[2] Wax, J. Zhang, S. Huang, C. Luo, and W. Li, "Key generation for Internet of Things: a contemporary survey," ACM Computing Surveys (CSUR), vol.54, no.1, pp. 1-37,2021.

[3] O. S. Guma'a, Q.M. Hussein, and Z. T. Mustafa," Dynamic keys generation for Internet of things," TELKOMNIKA Indonesian Journal of Electrical Engineering, vol.18, no.2, pp.4897-4909,2019

[4] R. B Naik and U. Singh, "A Review on Applications of Chaotic Maps in Pseudorandom Number Generators and Encryption," Annals of Data Science, pp.1-26,2022.

[5] M. T. Taha and J.M. Al-Tuwaijari,"Improvement of Chacha20 Algorithm based on Tent and Chebyshev Chaotic Maps," Iraqi Journal of Science, pp.2029-2039,2021.

[6] U. Zia, M. McCartney, B. Scotney, J. Martinez, and A. Sajjad,"A novel pseudorandom number generator for IoT based on a coupled map lattice system using the generalized symmetric map," SN Applied Sciences, vol.4, no.2,1-17,2022.

[7] M. Kohli and S. Arora," Chaotic grey wolf optimization algorithm for constrained optimization problems," Journal of Computational Design and Engineering, vol.5, no.4, pp. 458-472,2018.

[8] A.R. Kashani, M. Gandomi, C.V. Camp, and A.H. Gandomi," Optimum design of shallow foundation using evolutionary algorithms," Soft Computing, vol.24, pp.6809-6833,2020.

[9] D.Yang, G. Li, and G. Cheng,"On the efficiency of chaos optimization algorithms for global optimization Chaos, " Solitons & Fractals, vol.34, no.4, pp.1366-1375,2007.

[10] G. Kaur and S. Arora, "Chaotic whale optimization algorithm," Journal of Computational Design and Engineering, vol. 5, no. 3, pp.275-284,2018.

[11] J. R. Naif, G.H. Abdul-majeed, and A.K.& Farhan, "Internet of things security using the new chaotic system and lightweight AES, " Journal of Al-Qadisiyah for computer science and mathematics, vol.11, no.2, pp. 45-52,2019.

[12] R. Vohra and B. Patel, "An efficient chaos-based optimization algorithm approach for cryptography," Communication Network Security, vol. 1, no.4, pp.75-79,2012.

[13] Z. Rahman, X. Yi, I. Khalil, and M. Sumi, "Chaos and logistic map-based key generation technique for AES-driven IoT security," In International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (pp. 177-193). Springer, Cham,2021.

[14] R. A. Ali,"Random Number Generator based on Hybrid Algorithm between Particle Swarm Optimization (PSO) Algorithm and 3D-Chaotic System and its Application," Iraqi Journal of Information Technology. V, vl.8, no.3, 2018.

[15] M.H. Ismael and A.T. Maolood, "Proposed Secure Key for Healthcare Platform," Iraqi Journal of Computers, Communications, Control and Systems Engineering, vol.22, no.1,2022.

[16] M. Khan and T. Shah "A novel construction of substitution box with Zaslavskii chaotic map and symmetric group," Journal of Intelligent & Fuzzy Systems, vol. 28, no.4, pp. 1509-1517,2015.

[17] R. Hamza and F. Titouna,"A novel sensitive image encryption algorithm based on the Zaslavsky chaotic map," Information Security Journal: A Global Perspective, vol.25.no.4-6, pp. 162-179,2016.

[18] N. Balaska, Z. Ahmida, A. Belmeguenai, and S. Boumerdassi, "Image encryption using a combination of Grain-128a algorithm and Zaslavsky chaotic map," IET Image Processing, vol. 14, no.6, pp.1120-1131,2020.

[19] S. Arunkumar and M. Krishnan, "Enhanced Audio Encryption using 2-D Zaslavsky Chaotic Map," In 2022 International Conference on Computer Communication and Informatics (ICCCI) (pp. 1-4). IEEE,2022.

[20] Abdel-Basset, Mohamed, et al. "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection." Expert Systems with Applications 139 (2020): 112824.

[21] H. Faris, I., Aljarah, M.A. Al-Betar, and S. Mirjalili,"Grey wolf optimizer: a review of recent variants and applications, "Neural computing and applications, vol.30, no.2, pp.413-435,2018.

[22] N.M. Hatta, A. M., Zain, R. Sallehuddin, Z. Shayfull, and Y. Yusoff, " Recent studies on optimization method of Grey Wolf Optimiser (GWO): a review (2014–2017), "Artificial Intelligence Review, vol. 52, no.4, pp.2651-2683, 2019.

[23] Meidani, Kazem, et al. "Adaptive grey wolf optimizer." Neural Computing and Applications 34.10 (2022): 7711-7731.

[24] Al-Tashi, Qasem, et al. "A review of grey wolf optimizer-based feature selection methods for classification." Evolutionary Machine Learning Techniques: Algorithms and Applications (2020): 273-286.

[25] H. H. Alyas and A.A. Abdullah, "Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps," In Next Generation of Internet of Things (pp. 91-107). Springer, Singapore,2021.

[26] Nadimi-Shahraki, Mohammad H., Shokooh Taghian, and Seyedali Mirjalili. "An improved grey wolf optimizer for solving engineering problems." Expert Systems with Applications 166 (2021): 113917.

[27] Degabriele, Jean Paul, et al. "The security of chacha20-poly1305 in the multi-user setting." Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021.

[28] J. N. Hasoon, B.A. Khalaf, R.S. Hameed, S.A. Mostafa, and A. H. Fadil, "A Lightweight Stream Ciphering Model Based on Chebyshev Chaotic Maps and One Dimensional Logistic," In International Conference on Advances in Cyber Security (pp. 35-46). Springer, Singapore,2021.

[29] M. S. Mahdi, N.F. Hassan, and G.H. Abdul-Majeed, " An improved chacha algorithm for securing data on IoT devices," SN Applied Sciences, vol.3, no.4, pp.1-9,2021.

[30] P. Yadav, I. Gupta, and S.K. Murthy, "Study and analysis of eSTREAM cipher Salsa and ChaCha," In 2016 IEEE International Conference on Engineering and Technology (ICETECH) (pp. 90-94), 2016.