Received: August 21, 2022. Revised: September 11, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Using Three-Dimensional Logistic Equations and Glowworm Swarm Optimization Algorithm to Generate S-Box

Sanaa Ali Jabber[1] **Ammar Mohammd Ali Al-Tameemi[2]** **Safa Sami Abdul-Jabbar[3]***

[1]*Faculty of Administration and Economics, AL-Muthanna University, Iraq*
[2]*Chemical Engineering Department, University of Technology, Iraq*
[3]*Computer Science Department, College of Science for Women, University of Baghdad, Iraq*
* Corresponding author's Email: safa.s@csw.uobaghdad.edu.iq

Abstract: A major component of symmetric cryptosystems is the substitution box (S-box), which is primarily utilized in existing cryptographic ciphers to ensure secure data sanctuaries. The cryptographic strength of an S-box used in the encryption directly correlates to the amount of data protection that a cipher offers. This work developed a suitable configuration (8x8) S-box using metaheuristic approaches based on the Glowworm Swarm Optimization (GSO) algorithm and three-dimensional (3D) logistic equations to construct pseudo-random number generators. Chaotic maps boost the initial population of the GSO, making it a suitable place to start. The nonlinearity score of the S-box served as the GSO's objective function. Operations for exploration and exploitation are balanced using the GSO. In comparison to various references, the proposed S-box satisfied all of the S-security box's requirements, including the strict avalanche criterion (SAC), bijection, nonlinearity, output bits independence criteria (BIC), and histogram. The proposed method's nonlinearity property was 107.00. It is regarded as a successful and effective result. BIC and SAC averages were 103.500 and 0.499, respectively. Compared to a simple histogram, the encryption histogram was more uniform. As a result, it has been demonstrated that the suggested S-box is robust and resistant to cryptographic assaults.

Keywords: Chaotic maps, Glowworm swarm optimization algorithm, Security, S-box.

1. Introduction

Encryption is one of the key strategies for safeguarding digital data. Building block ciphers, which use mathematical operations (substitution-permutation networks), is based on Shannon's theory of diffusion and confusion[1, 2]. An input block of n bits is transformed into a new nonlinear block of m bits via a substitution box (S-box). Galois Field theory dictates that its mapping be one-to-one and denoted by the symbol $GF(2^n) \rightarrow GF(2^m)$. A linear transformation that shuffles the input bits is called a permutation. The first-round results of the S-box are permuted before moving on to the second round[2].

The Glow-worm Swarm Optimization (GSO) method was inspired by the way that glow worms naturally entice one another by emitting luciferin (a substance with a luminous property). If the luciferin emissions of the glow worms are higher it considered more attractive compared to other glow-worms. A glow-worm will go toward a brighter glow-worm when it is in the field of their vision. Resolving systems of nonlinear equations and using several robots to locate various sources of a general nutritional profile that is geographically spread throughout a two-dimensional workspace are some uses of the GSO[3].

In this study, we developed a suitable configuration (8x8) S-box using metaheuristic approaches based on the GSO algorithm and three-dimensional (3D) logistic equations to construct pseudo-random number generators works as an initial value of GSO population. GSO is utilized to balance among the operations of the exploitation and the exploration to find the optimal solution.

We experimented with many techniques for generating the S-box, using algorithms inspired by nature, which were not used previously in this process. Therefore, we found that the use of GSO algorithm gives good results, despite the presence of newer algorithms which can be developed and used in the future work. Also, GSO algorithm provides a good method to balancing between exploitation and exploration operations, that is lead to avoid falling in local optima problem.

This study's remaining sections are organized as follows. The related works are explained in Section 2. The S-box issues are described in Section 3. The proposed algorithms are discussed in Section 4. The assessment of the suggested methods is presented in Section 5. The most significant findings are explained in Section 6, along with the conclusion.

2. Related works

There are several ways to construct the S-box, including algebraic methods, random search, and metaheuristic approaches. The S-boxes built using algebraic techniques exhibit the best cryptographic properties. However, using this approach makes it hard to amass a large number of powerful S-boxes. In contrast, S-boxes created using the random search method often have weak cryptographic characteristics. However, Metaheuristic algorithms are effective in both software and hardware implementation[4].

In this paper, we compared the suggested approach with eleven studies. The existing S-boxes were created using continuous or discrete chaotic approaches are find in [5-10]. Unfortunately, this strategy does not produce a good results in nonlinearity or other performance indicators. In contrast, [11] used algebraic methods, while [12] combined an algebraic approach with chaos theory to create S-boxes. Both methods are vulnerable to statistical and algebraic assaults. Additionally, [2], [13] used a hyperchaotic system (five-dimensional). Its implementation in hardware and software is hence inefficient. Finally, [4, 14, 15] used the chaotic system and metaheuristic algorithms. These techniques can produce S-boxes with useful characteristics. In [16] the authors used a Flower Pollination Algorithm and Chaos System method to designing an efficient S-Box. This method was provided a good confusion for the generated S-Box. However, they are not appropriate for all applications. At the same time, there are still a lot of problems with the analysis and design of S-boxes. Nature-inspired metaheuristics have been proven to be a successful approach for machine learning and difficult engineering case studies in several publications. Additionally, in order to avoid becoming trapped in local optima, they could be balancing between local and global search[17-19]. Consequently, the GSO is suggested to address these challenges.

3. Problem description

The S-box design challenge will be discussed in this section. The National Security Agency has released some suggestions for assessing the cryptographic properties of S-boxes in order to prevent suspicion in DES S-boxes[20]. The following are the performance evaluation standards that are widely acknowledged and regarded as essential for creating cryptographically robust S-boxes[16]:

3.1 Nonlinearity:

Regarding a Boolean function *f(x)*, the Walsh spectrum could be utilized to determine nonlinearity. In Eq. (2), the Walsh Transformation is defined[21]:

$$S_g(r) = \sum_{x \in GF(2^m)} (-1)^{f(x) \oplus x.r} \tag{1}$$

$$N_f = 2^{m-1} - \frac{1}{2} max_{r \in GF(n^m)} \left| S_g(r) \right|, \tag{2}$$

where $x.r$ is the dot product of x and r, $S_g(r)$ is the Walsh spectrum of f(x), and $x.r = x_1 \oplus r_1 + \ldots + x_m \oplus r_m$.

3.2 Bijection:

This characteristic describes a mapping in which every input bit corresponds to a single, unique output bit. The (8x8) S-box in this study is needed to have a range of output values of a period [0, 255].

3.3 Differential approximation probability (DP):

The nonlinearity of an encryption process is accomplished via the S-box. In theory, it may keep differential homogeneity. The differential uniformity is used in this manner, however, to ensure uniform mapping, and the differential input should uniquely map to the differential output map. Differential uniformity is measured using the DP. In Eq. (3), DP is described mathematically as follows[4]:

$$DP = \frac{[\#\{g \in M \mid s(g) \oplus S(g \oplus \Delta_g) = \Delta_h]}{256} \tag{3}$$

Received: August 21, 2022. Revised: September 11, 2022.

Where *M={0,1,...,255}* and Δ_g, Δ_h are the input and output differentials, respectively.

3.4 Output bits independence criteria (BIC):

BIC is one of the key characteristics of an S-box and was developed by AF Webster and SE Tavares. This criteria states that when one bit in the plaintext is complemented, any two ciphertext bits should have a high degree of independence from one another[21].

3.5 Strict avalanche criteria (SAC):

In 1985, AF Webster and SE Tavares proposed the SAC. The SAC essentially states that if the input is altered by a single bit, the entire set of output bits should likewise be altered by half. As a result, an S-box is said to be robust if its SAC value is near to 0.5.

4. The proposed algorithms

The three-dimensional (3D) logistic equations used to determine the initial randomness values for the S-box in the GSO are explained in this section. Also, this section is explaining the GSO and its proposed strategy.

4.1 Generation of initial substitution box using chaos theory

The branch of mathematicians that analyzes how the dynamic systems behave is called a chaos theory. It is an excellent tool to utilize when producing a random number generator[22]. In reality, chaotic system behaviors, such as having a mixed-up attribute, deterministic nature, very sensitive to initial conditions, and being unable to predict the long-term returns. Therefore, these features are useful in cryptography[23]. The chaos logistic equation can be utilized to get enormous pseudorandom numbers, which is described below[24]. Several logistic equations should be used during the generator design phase to improve the unpredictability (the randomness) of the generated keys and raise their security.

4.1.1. Logistic equation

The conventional version of the one-dimensional logistic equation is demonstrated by[25, 26]:

$$F(Xi) = P\,Xi\,(1- Xi) \tag{4}$$

P is the iteration growth rate, and *P* values range between [1, 4]. While *Xi* is the iteration of *X0*, and *Xi* values range between *[0,1]*. Fig. 1 shown the results values of the Logistic equation (*V*) where *P = 3.6*, and *X=0.7842* after applying Eq. (5) to convert the results to integer values between *[0,255]*[16].

$$V = round\,(Xi \times 255) \tag{5}$$

The three-dimensional (3D) logistic equations, which include three fundamental variables (*X*), (*Y*), and (*Z*), as well as three control parameters (*P*, *B*, and *C*) was used in[27].

$$X_{m+1} = PX_m(1 - X_m) + B\,Y_m^2 X_i + CZ_m^3 \tag{6}$$

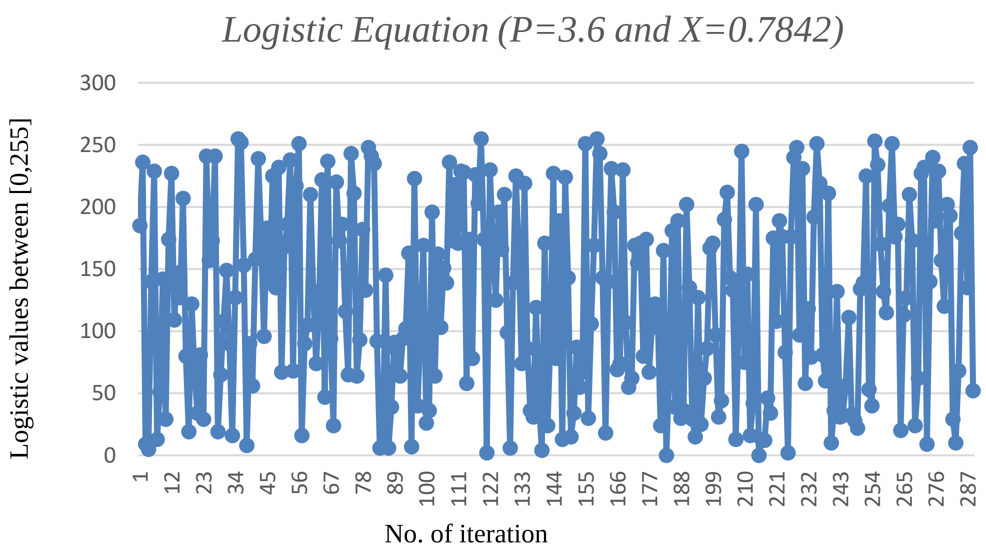

Figure. 1 One dimension logistic equation results

Received: August 21, 2022. Revised: September 11, 2022.

$$Y_{m+1} = PY_m(1 - Y_m) + B\,Z_m^2 Y_i + CX_m^3 \quad (7)$$

$$Z_{m+1} = PZ_m(1 - Z_m) + B\,X_m^2 Z_m + CY_m^3 \quad (8)$$

The values of the control parameters should be *$3.53 < P < 3.81$, $0 < B < 0.022$, and $0 < C < 0.015$*. While the values of the variables (*X*), (*Y*) and (*Z*) must be in range *[0, 1]*. Fig. 2 shown the results values of the 3D Logistic equation.

The initial S-box result was obtained by using 3D Logistic equations, which helped to speed up the best solution and use less memory. This is an excellent starting point for searching using the Glowworm Swarm Optimization (GSO) algorithm rather than beginning from scratch. Table 1 displays the S-Box of GSO's starting values as created using 3D Logistic equations. To build an effective and robust S-box that can be used for cryptographic operations, GSO must alter this S-box.

4.2 Glowworm swarm optimization (GSO) algorithm

Swarm intelligence is a kind of artificial intelligence that is based on the self- organized systems and the decentralized collective behavior. Its main objective is to investigate the collective behavior of populations of simple agents that interacts with one another and with their

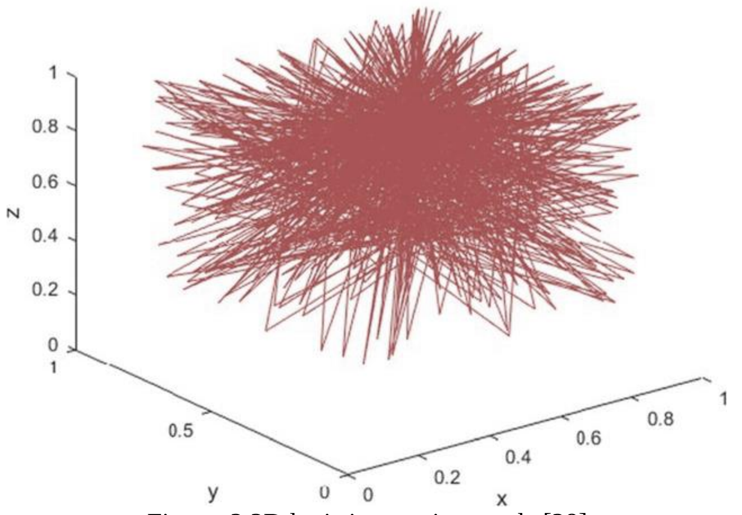

Figure. 2 3D logistic equation results[28]

Table 1. The initial values S-Box of GSO generated by 3D logistic equations with nonlinearity =104.8750

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	221	82	77	206	128	144	75	108	158	154	79	121	28	62	30	16
1	204	85	210	167	237	39	112	254	124	11	37	97	242	84	184	42
2	78	227	41	107	38	110	207	25	74	250	123	58	22	71	20	155
3	170	49	135	213	68	33	40	57	147	60	255	14	220	113	139	63
4	171	194	212	7	138	251	253	24	126	17	48	245	51	181	208	149
5	177	240	54	94	100	89	3	129	143	111	191	185	174	4	199	218
6	197	92	99	93	125	29	67	35	247	236	164	166	190	8	45	69
7	224	81	43	201	15	178	95	193	225	20	148	160	244	145	66	137
8	88	105	64	176	90	104	83	168	55	198	98	127	163	101	46	61
9	13	133	115	0	222	195	117	157	44	228	162	249	182	226	189	172
A	141	246	175	106	211	114	10	231	26	233	151	238	80	239	27	118
B	215	59	87	219	205	52	183	217	248	72	214	96	2	186	140	152
C	142	180	91	56	116	241	19	5	73	169	216	232	209	229	159	235
D	130	161	21	150	179	165	234	131	12	86	32	34	192	9	1	18
E	23	200	230	103	173	132	50	102	109	156	252	6	70	53	136	119
F	122	202	134	153	65	36	120	146	188	47	223	243	76	187	31	203

Received: August 21, 2022. Revised: September 11, 2022.

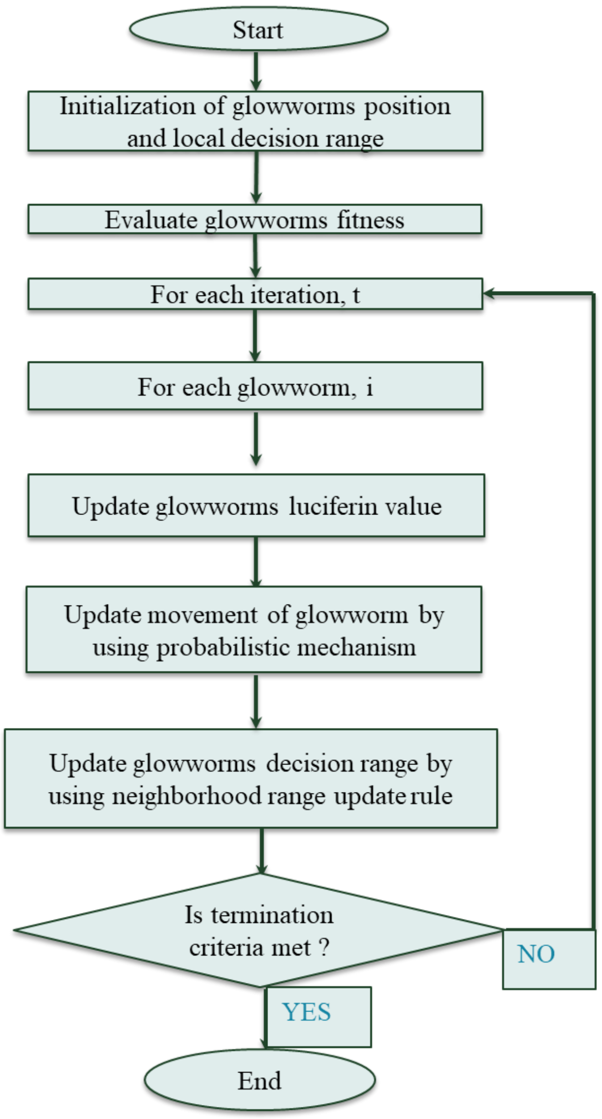
Start
Initialization of glowworms position and local decision range
Evaluate glowworms fitness
For each iteration, t
For each glowworm, i
Update glowworms luciferin value
Update movement of glowworm by using probabilistic mechanism
Update glowworms decision range by using neighborhood range update rule
Is termination criteria met ?
NO
YES
End

Figure. 3 The flowchart of the GSO algorithm[29]

environment locally. In 2005, GSO is designed by D. Ghose and K. N. Krishnanad[3]. It has been applied to problems of function optimization as well as sensor noisy text and robot simulation. The main idea of the GSO algorithm inspired by the nighttime activities of glowworms in nature. Glowworms exercise in groups and communicate with one another using luciferin. The increasing amount of the light emits from the glowworms means that more glowworms drawn to it[29]. Fig. (3) depicts the GSO algorithm's flowchart.

Each glowworm has its own luciferin and has a unique vision field, which is referred to as the local-decision range. These glowworms are randomly dispersed in the objective function's declaration space. The intensity of each glowworm's luciferin serves as a measure of the objective function's fitness. The position of this glowworm corresponds better to the value of the objective function in the glowworm with brighter luciferin. The glowworm's movement pattern is to scan its visual field for nearby glowworms until it locates the one with the highest amount of luciferin, at which point it moves in its direction. The various glowworm populations in the area have an impact on the local-decision range of glowworms. If there aren't enough glowworms nearby, the glowworms can expand their search area to discover additional glowworms; otherwise, they can reduce their search area. Glowworms initial distribution, updates to luciferin, glowworm movement, and updates to local-decision range are all phases in the algorithm[30].

4.2.1. Phase of initialization

GSO starts by distributing a population of n glowworms around the search region at random. Initially, the luciferin 0 concentration in each glowworm is the same. Each algorithm cycle includes three phases: phase of luciferin update, phase of movement, and phase of neighborhood range update[30].

4.2.2. Phase of luciferin update

The luciferin updates are influenced by the function value at the glowworm position. During the luciferin-update phase, each glowworm raises its level of luciferin by an amount proportional to the fitness of its present position in the objective function space. The following provides the luciferin update rule[29]:

$$l_j(m+1) = (1-P)l_j(m) + \chi J(y_j(m+1)) \quad (9)$$

Where P is the luciferin decay constant ($0<P<1$), χ is the constant for luciferin enhancement, and $J(y_j(m))$ represents the objective function's value at the position of glowworm J at time m. Where l_j stands for the quantity of luciferin related to glowworm j at time m[30].

4.2.3. Phase of movement

This phase used a probabilistic method, each glowworm chooses to move in the direction of a neighbor whose luciferin value is greater than its own. In other words, neighbors that brighter more intensely attract glowworms. The probability of moving from g to h is:

$$P_j(m) = \frac{l_h(m) - l_g(m)}{\sum_{k \in N_g(m)} l_k(m) - l_g(m)} \quad (10)$$

where, $h \in N_g(m)$, $N_g(m) = \{h : d(g,h) < r_d^g ; l_h(m) < l_g(m)\}$ is the set of glowworm $g's$ neighbors at time m, $d(g, h)$ is the Euclidean distance among glowworms g and h at time m, and $r_d^g(t)$ is the glowworm g's variable neighborhood range at time m.

Whereas the moving from location g to location h is:

$$X_{\dot{g}}(m+1) = X_g(m) + S\left(\frac{X_{\dot{h}}(m) - X_g(m)}{\|X_h(m) - X_g(m)\|}\right) \quad (11)$$

where S (> 0) is the step size, ||.|| denotes the Euclidean norm operator, and $X_g \in R^m$ is the location of the glowworm g in the m-dimensional real space R^m at time m[29].

4.2.4. Phase of neighborhood range update

Each agent h is a member of a neighborhood whose radial range ($0< r_d^h \leq r_s$) is constantly shifting. GSO uses an adaptive neighborhood range to detect the presence of several peaks in a multimodal function landscape. Let r_0 represent each glowworm's initial neighborhood range. Therefore, the following rule is used to adaptively update each glowworm's neighborhood range:

$$r_d^h(m+1) = \min\{r_s, \max\{0, r_d^h(m) + B(n_m - |N_h(m)|)\}\} \quad (12)$$

B is a parameter value and n_h is a parameter that controls the number of neighbors[30].

4.3 Adapted GSO

According to the bijectivity condition, each element of the S-box must be unique. However, the S-box produced by the GSO algorithm does not satisfies this condition. As a result, each GSO S-box must modified and carried out as follows:

1. Since the S-box only holds 256 discrete values, the S-box values must be transformed using Eq. (5) into values in the range (0–255) in order to ensure the bijection condition.
2. Based on Table 1, the initial population values S-Box of GSO generated by 3D Logistic equations is used. It is hence appropriate for S-box condition.
3. To assess the current best solution, Equation (2) is applied as an objective function.
4. Replacement element numbers are based on the initialized parameters.

By doing this, we guarantee that the S-box values are distinct numbers and fall between the higher and lower boundaries. The modified GSO is shown in Algorithm 1. While, the generated S-box based on modified GSO is shown in Table 2 with nonlinearity 107.00.

5. Evaluation of the proposed S-box

To ascertain modified GSO efficiency, several tests based on S-box generation were conducted. On a machine running Windows 10 (64-bit OS), MATLAB Release 2021a, and an Intel(R) CoreTM i7-8565U CPU clocked at 3.79 GHz, all tests were conducted.

5.1 Performance evaluation of the produced S-box

The sturdy S-box must meet the following conditions, some of which are detailed in Section 3.

Algorithm 1. S-box produced based on modified GSO

```
input: S-box initial values of Table 1, Step size S, initialise parameters.
output: Manipulation of S-box with Largely nonlinear.
Using Eq. 2, evaluate the S-box and determine the best current solution;
While (stopping criterion is not met) do
    For each glowworm
        Calculate the luciferin value using equation (9)// Update phase
    End for
    For each glowworm
        Calculate the probability of moving using equation (10)
        Select glowworm with the large probability;
        Moving from location g to location h using equation (11)// Movement phase
        Update each glowworm's neighborhood range using equation (12)// Neighborhood Range Update phase
        Using Eq. 2, evaluate the generated S-box and determine the best new solution;
    End for
    Store the current best solution;
End while.
```

Table 2. S-box produced based on modified GSO

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	221	111	77	206	128	144	75	108	158	154	239	121	28	47	30	104
1	204	85	210	167	237	39	112	254	124	11	37	97	242	84	184	42
2	78	227	41	107	38	110	207	25	74	250	123	58	22	71	196	155
3	170	49	135	213	68	55	40	57	147	1	255	14	220	113	139	63
4	171	194	212	7	138	251	253	24	126	17	48	245	51	181	208	149
5	66	240	54	94	100	89	3	129	6	82	191	185	174	4	199	218
6	197	92	99	93	125	29	67	35	247	236	164	166	190	8	45	69
7	224	81	43	201	15	178	95	193	225	20	148	160	244	145	177	137
8	88	105	64	176	90	16	83	168	33	198	98	127	163	101	46	61
9	13	133	115	0	222	195	117	157	44	228	162	249	182	226	189	172
A	141	246	175	106	211	114	60	231	26	233	151	238	80	79	27	118
B	215	59	87	219	205	52	183	217	248	72	214	96	2	186	140	152
C	142	180	91	56	116	203	19	5	73	169	216	232	209	229	159	235
D	130	161	21	150	179	165	234	131	12	86	32	34	192	9	10	18
E	23	200	230	103	173	132	50	102	109	156	252	143	70	53	136	119
F	122	202	134	153	65	36	120	146	188	62	223	243	76	187	31	241

Fourteen S-boxes were created using different methods, and the results of the eleven S-boxes created utilizing GSO methodologies were compared.

5.1.1. Nonlinearity property:

Eq. (2) calculates the nonlinearities of eight Boolean functions of the created S-boxes compared to other S-boxes generated using various methods, and Table 3 displays the results. The objective function was to obtain an S-box with nonlinearity scores of 108 108 104 108 108 106 108 106 with an average of 107.00. When compared to other ways, the GSO approach is deemed to be an adequate method with acceptable outcomes for obtaining high nonlinearity. Although there are some references that have a higher nonlinearity of S-box than the proposed algorithm but the proposed approach achieved higher results in other S-box

Table 3. Analysis of S-boxes nonlinearity

Ref.	N1	N2	N3	N4	N5	N6	N7	N8	Max.	Min.	Avg.
Proposed	108	108	104	108	108	106	108	106	108	104	107.00
[15]	108	108	108	108	108	108	108	108	108	108	108
[16]	110	106	106	108	108	108	106	106	110	106	107.25
[14]	108	106	106	106	106	110	106	108	110	106	107
[12]	108	106	108	110	110	108	104	100	110	100	106.75
[8]	108	108	106	106	106	106	106	106	108	106	106.5
[13]	110	106	108	106	106	106	104	106	110	104	106.5
[7]	108	106	104	106	108	106	106	106	108	104	106.25
[9]	106	106	106	106	105	106	107	106	107	105	106
[5]	104	106	108	106	102	104	106	106	108	102	105.25
[11]	104	100	108	106	102	106	104	108	108	100	104.75
[6]	102	102	100	106	102	100	104	98	106	98	101.75

Table 4. BIC-nonlinearity test for the suggested S-box

0	106	102	102	102	108	104	104
106	0	102	106	104	106	106	102
102	102	0	106	104	96	102	108
102	106	106	0	102	108	104	100
102	104	104	102	0	102	104	104
108	106	96	108	102	0	98	108
104	106	102	104	104	98	0	98
104	102	108	100	104	108	98	0

Table 5. Dependency matrix for the suggested S-box

0.4688	0.5	0.5469	0.5781	0.4062	0.4375	0.4844	0.5
0.4531	0.5625	0.4531	0.4531	0.5625	0.5	0.5312	0.5156
0.4375	0.4688	0.5	0.4531	0.5156	0.4688	0.4688	0.4688
0.5156	0.5	0.5156	0.5	0.4531	0.4688	0.4844	0.5938
0.5625	0.4844	0.4844	0.4688	0.4844	0.5156	0.5312	0.4219
0.5156	0.5156	0.5156	0.5781	0.4531	0.5312	0.5156	0.5469
0.5	0.5156	0.5	0.5312	0.5938	0.5	0.5781	0.4688
0.4844	0.4844	0.4062	0.4844	0.5156	0.5156	0.4844	0.4844

Table 6. Comparison of BIC and SAC results

Ref.	BIC Avg.	SAC Avg.
Proposed S-box	103.500	0.49903
[15]	103.35	0.5068
[16]	104.7857	0.498
[14]	104.21	0.5014
[12]	104	0.5002
[8]	104.07	0.5001
[13]	104.57	0.4995
[7]	103	0.4996
[9]	102.64	0.5065
[5]	103.57	0.4994
[11]	105.07	0.4938
[6]	102.64	0.5017

measurements. Therefore, it is possible to use the obtained S-box with the applications that are compatible with the results we got.

5.1.2. Bijective property:

Because of the fact that all of the proposed S-box values fall inside the range [0,255], the produced S-box is bijective. All Boolean functions have the same Hamming weight, which is 128 128 128 128 128 128 128.

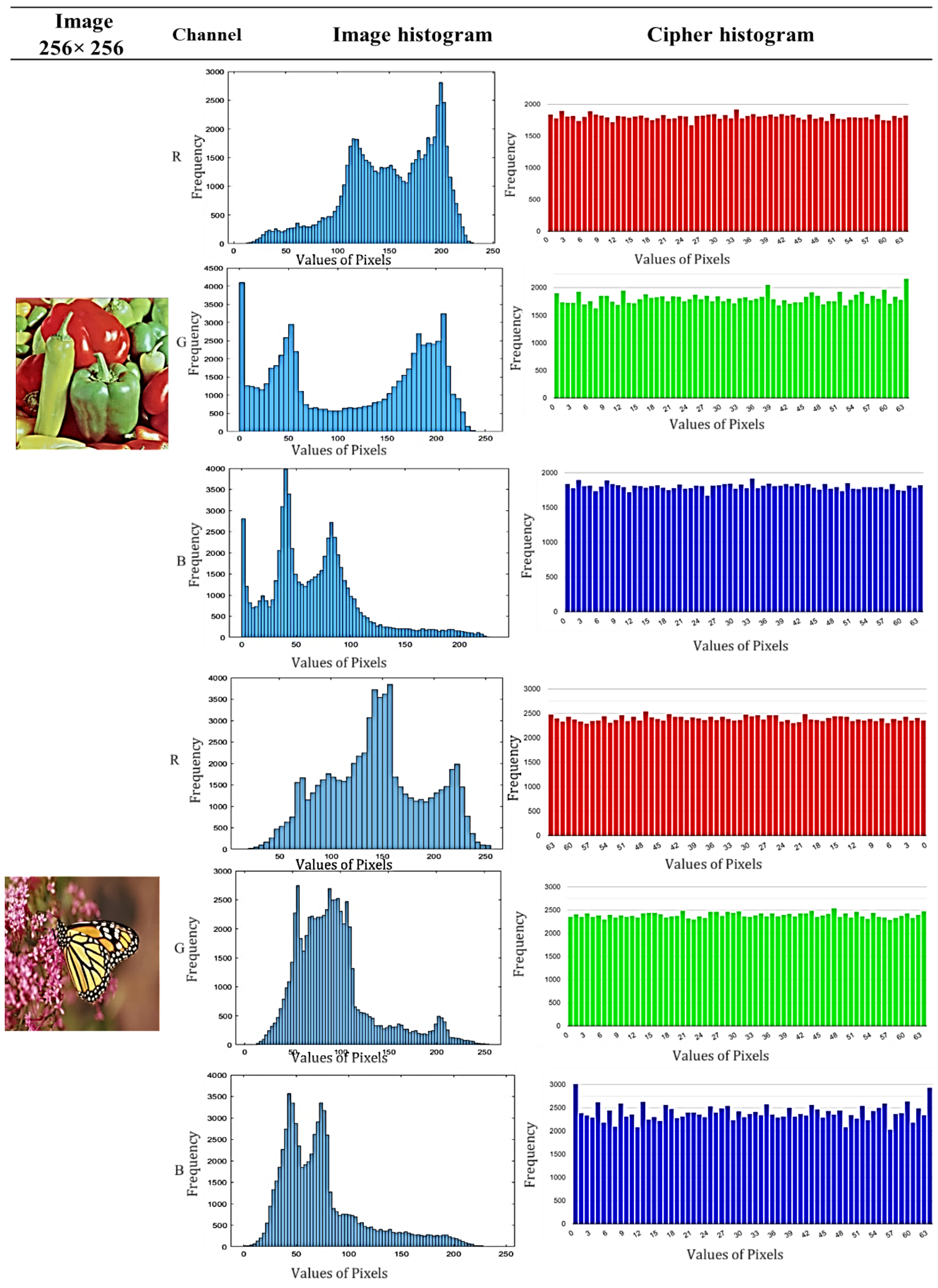

Figure. 4. Images histogram

5.1.3. BIC property:

Table 4 displays the results of the suggested S-boxes. The resulting S-boxes have a minimum BIC-nonlinearity of 96. Table 6 also includes a comparison of our S-box with the aforementioned sources. The GSO S-box had average values of 103.500. The GSO did not employ BIC as an objective function. Consequently, the goal of the suggested technique was not to generate an S-box with a substantial BIC.

5.1.4. SAC property:

As shown in Table 5, the dependence matrix is used to describe the SAC of the proposed S-box.

The generated S-box has an average SAC value of 0.499. It's nearly at the ideal value (0.5). Table 6 compares the SAC of the proposed S-boxes for this investigation. It has been demonstrated that the GSO method yields products with acceptable SAC properties.

5.2 Analysis of histogram

The histogram[31] displays an image's pixel distribution. The encryption picture may be attacked by the eavesdropper via a histogram analysis [32]. To avoid statistical histogram attacks, the encryption picture's histogram should be as uniform as feasible. Various photos (256x256) are shown in Fig. 4, of the R, G and B channels for original images before and after encryption. The encryption histogram appears to be more uniform than the ordinary histogram. Thus, the system is secure and immune to histogram attacks.

6. Conclusion

In this paper, a new method for creating a strong 8x8 S-box was proposed. A comparison of the resulting S-box with various studies was done. One of the most significant findings was the nonlinearity property, which came in at 107.0. Accordingly, this result is regarded as efficient and suitable result in cryptography operations. BIC and SAC averages were 103.500 and 0.499 respectively. The GSO did not use those criteria as an objective function. However, those results were accepted and it was better than other compared references. Additionally, a histogram was used to demonstrate the resistance to differential analysis. Therefore, the suggested S-box is strong and resistant to cryptographic assaults, and it also offers good confusion. In future work, the got S-box will be further upgraded into a dynamic S-box by using a more advanced metaheuristic algorithm approach.

Conflicts of Interest

No conflicts of interest are disclosed by the authors.

Author Contributions

The methodology, paper's idea, formal analysis, and software was written by the first author. While the second author was responsible for the resources and data curation. The third author was responsible of project administration and oversight.

References

[1] F. Ishfaq, “A MATLAB Tool for the Analysis of Cryptographic Properties of S-boxes”, *Capital University*, 2018.

[2] A. Alhudhaif, M. Ahmad, A. Alkhayyat, N. Tsafack, A. K. Farhan, and R. Ahmed, “Block Cipher Nonlinear Confusion Components Based on New 5-D Hyperchaotic System”, *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3090163.

[3] K. N. Krishnanand and D. Ghose, “Detection of multiple source locations using a glowworm metaphor with applications to collective robotics”, In: *Proc. of 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pp. 84-91, 2005, doi: 10.1109/SIS.2005.1501606.

[4] H. S. Alhadawi, D. Lambić, M. F. Zolkipli, and M. Ahmad, “Globalized firefly algorithm and chaos for designing substitution box”, *J. Inf. Secur. Appl.*, Vol. 55, p. 102671, 2020.

[5] F. Özkaynak, “On the effect of chaotic system in performance characteristics of chaos based s-box designs”, *Phys. A Stat. Mech. its Appl.*, Vol. 550, p. 124072, 2020.

[6] Z. M. Z. Muhammad and F. Özkaynak, “A Cryptographic Confusion Primitive Based on Lotka–Volterra Chaotic System and Its Practical Applications in Image Encryption”, In: *Proc. of 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, pp. 694-698, 2020.

[7] M. A. B. Farah, A. Farah, and T. Farah, “An image encryption scheme based on a new hybrid chaotic map and optimized substitution box”, *Nonlinear Dyn.*, pp. 1-24, 2019.

[8] D. Lambić, “A new discrete-space chaotic map based on the multiplication of integer numbers and its application in S-box design”, *Nonlinear Dyn.*, Vol. 100, No. 1, pp. 699-711, 2020.

[9] V. M. S. García, R. F. Carapia, C. R. Márquez, B. L. Benoso, and M. A. Pérez, “Substitution box generation using Chaos: An image encryption application”, *Appl. Math. Comput.*, Vol. 332, pp. 123-135, 2018.

[10] R. S. Salman, A. K. Farhan, and A. Shakir,

"Creation of S-Box based One-Dimensional Chaotic Logistic Map: Colour Image Encryption Approach", *Int. J. Intell. Eng. Syst.*, Vol. 15, No. 5, p. 2022, doi: 10.22266/ijies2022.1031.33.

[11] I. Hussain, T. Shah, M. A. Gondal, W. A. Khan, and H. Mahmood, "A group theoretic approach to construct cryptographically strong substitution boxes", *Neural Comput. Appl.*, Vol. 23, No. 1, pp. 97-104, 2013.

[12] S. S. Jamal, A. Anees, M. Ahmad, M. F. Khan, and I. Hussain, "Construction of cryptographic S-boxes based on mobius transformation and chaotic tent-sine system", *IEEE Access*, Vol. 7, pp. 173273-173285, 2019.

[13] T. Farah, R. Rhouma, and S. Belghith, "A novel method for designing S-box based on chaotic map and teaching–learning-based optimization", *Nonlinear Dyn.*, Vol. 88, No. 2, pp. 1059-1074, 2017.

[14] M. Ahmad, D. Bhatia, and Y. Hassan, "A novel ant colony optimization based scheme for substitution box design", *Procedia Comput. Sci.*, Vol. 57, pp. 572-580, 2015.

[15] Y. Wang, K. W. Wong, C. Li, and Y. Li, "A novel method to design S-box based on chaotic map and genetic algorithm", *Phys. Lett. A*, Vol. 376, No. 6-7, pp. 827-833, 2012.

[16] S. A. Jassim and A. K. Farhan, "Designing a novel efficient substitution-box by using a flower pollination algorithm and chaos system", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 1, pp. 176-187, 2022, doi: 10.22266/ijies2022.0228.17.

[17] H. A. Abdulwahab, A. Noraziah, A. A. Alsewari, and S. Q. Salih, "An enhanced version of black hole algorithm via levy flight for optimization and data clustering problems", *IEEE Access*, Vol. 7, pp. 142085-142096, 2019.

[18] Z. M. Yaseen, A. M. A. Juboori, U. Beyaztas, N. A. Ansari, K. Chau, C. Qi, M. Ali, S. Q. Salih, and S. Shahid, "Prediction of evaporation in arid and semi-arid regions: A comparative study using different machine learning models", *Eng. Appl. Comput. Fluid Mech.*, Vol. 14, No. 1, pp. 70-89, 2020.

[19] Z. M. Yaseen, M. F. Allawi, H. Karami, M. Ehteram, S. Farzin, A. N. Ahmed, S. B. Koting, N. S. Mohd, W. Z. B. Jaafar, H. A. Afan, and A. E. Shafie, *Neural Comput. Appl.*, Vol. 31, No. 12, pp. 8807-8821, 2019.

[20] D. Teodorović, "Bee colony optimization (BCO)", In: *Proc. of Innovations in Swarm Intelligence*, pp. 39-60, 2009.

[21] T. Akhtar, N. Din, and J. Uddin, "Substitution box design based on chaotic maps and cuckoo search algorithm", In: *Proc. of 2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*, pp. 1-7, 2019.

[22] M. Francois, T. Grosges, D. Barchiesi, and R. Erra, "A new pseudo-random number generator based on two chaotic maps", *Informatica*, Vol. 24, No. 2, pp. 181-197, 2013.

[23] C. Guyeux, Q. Wang, and J. M. Bahi, "A pseudo random numbers generator based on chaotic iterations: application to watermarking", In: *Proc. of International Conference on Web Information Systems and Mining*, pp. 202-211, 2010.

[24] A. S. Hamad and A. K. Farhan, "Image Encryption Algorithm Based on Substitution Principle and Shuffling Scheme", *Eng. Technol. J.*, Vol. 38, No. 3B, pp. 98-103, 2020.

[25] M. S. Fadhil, A. K. Farhan, and M. N. Fadhil, "Designing Substitution Box Based on the 1D Logistic Map Chaotic System", In: *Proc. of IOP Conference Series: Materials Science and Engineering*, Vol. 1076, No. 1, p. 12041, 2021.

[26] S. A. Jassim, A. K. Farhan, and A. H. Radie, "Using a Hybrid Pseudorandom Number Generator for Cryptography in the Internet of Things", In: *Proc. of 2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA)*, pp. 264-269, 2021.

[27] F. A. Kadhim and M. H. Emad, "Mouse movement with 3D chaotic logistic maps to generate random numbers", *Diyala J. Pure Sci.*, Vol. 13, No. 3-part 2, pp. 24-39, 2017.

[28] S. Patel, K. P. Bharath, and R. Kumar, "Symmetric keys image encryption and decryption using 3D chaotic maps with DNA encoding technique", *Multimed. Tools Appl.*, Vol. 79, No. 43, pp. 31739-31757, 2020.

[29] K. N. Kaipa and D. Ghose, *Glowworm Swarm Optimization: Theory, Algorithms, and Applications*, Vol. 698, 2017.

[30] Y. Zhou, G. Zhou, Y. Wang, and G. Zhao, "A glowworm swarm optimization algorithm based tribes", *Appl. Math. Inf. Sci.*, Vol. 7, No. 2, pp. 537-541, 2013.

[31] S. A. Mohseni, H. R. Wu, J. A. Thom, and A. B. Hadiashar, "Recognizing Induced Emotions With Only One Feature: A Novel Color Histogram-Based System", *IEEE Access*, Vol. 8, pp. 37173-37190, 2020, doi: 10.1109/ACCESS.2020.2975174.

[32] A. A. Shah, S. A. Parah, M. Rashid, and M. Elhoseny, "Efficient image encryption scheme

based on generalized logistic map for real time image processing”, *J. Real-Time Image Process.*, Vol. 17, No. 6, pp. 2139-2151, 2020.