The Journal of Supercomputing (2024) 80:141–201
https://doi.org/10.1007/s11227-023-05413-x

An effective chaotic maps image encryption based on metaheuristic optimizers

Sally Mohamed Sameh[1] · Hossam El-Din Moustafa[1] · Ehab H. AbdelHay[1,3] · Mohamed Maher Ata[2]

Accepted: 17 May 2023 / Published online: 23 June 2023

Abstract

This paper proposes a robust optimization of eight chaotic maps: Logistics, Sine, Gauss, Circle, Tent, Chebyshev, Singer, and Piecewise Maps, for superior image encryption. The proposed model consists of two main processes: chaotic confusion and pixel diffusion. In the chaotic confusion process, the positions of the image's pixels are permuted with the chaotic maps, where the initial condition and the control parameters represent the confusion key. Firstly, the confusion process was performed using the eight chaotic maps without optimization. Then nine metaheuristic optimizers, which are the genetic algorithm, particle swarm optimizations, whale optimization algorithm, dragonfly algorithm, grey wolf optimizer, moth-flame optimizer, sine cosine algorithm, multi-verse optimizer, and ant-lion optimizer, have been used to fine-tune the control parameters of the eight chaotic maps. Then the image's pixel values are changed using the diffusion function in the pixel diffusion process. Multiple performance metrics, such as entropy, histogram, cross-correlation, computation time analysis, the number of pixels change rate (NPCR), unified average changing intensity (UACI), noise attack, data loss, and key analysis metrics, are utilized to evaluate the proposed model. The results demonstrate that the encryption algorithms based on the eight optimized chaotic maps are more resistant to differential attacks than those without optimization. Furthermore, the optimized Gauss chaotic map is the most computationally efficient, while the chaotic circle map has the most robust key. The careful adjustment of initial conditions and control parameters empowers the chaotic maps to create encryption keys with greater randomness and complexity, thereby increasing the security level of the encryption scheme. Experimental analysis indicates that the correlation coefficient values of images encrypted with the proposed scheme are nearly zero, the histogram of the encrypted images is uniform, the execution time of 0.1556 s, the key space of 10^80, NPCR of 99.63%, UACI of 32.92%, and entropy of 7.997. Moreover, the analysis of noise and cropping attacks, along with the comparison with other algorithms, demonstrate the efficiency and robustness of the proposed algorithm.

Extended author information available on the last page of the article

Keywords Encryption · Metaheuristics · Chaotic maps · Optimization

1 Introduction

Due to the rapid progress in smart multimedia devices and communication technologies, securing multimedia data during the transmission process from sender to receiver is extremely critical in digital communication. For this reason, encryption and its applications are critical for improving security standards. Images always have a high value among all other multimedia types due to their visual impact [1]. Images are used to communicate information in various industries, including but not limited to medical, military, aerospace, banking, and education; therefore, they must be protected from unauthorized access [2]. The growing demand for robust image encryption systems has emerged as a consequence of the increased need for the Internet of Things and technological advancements [3]. Image encryption is mainly composed of two phases: confusion and diffusion. The pixels are shuffled throughout the confusion stage [4], while the pixel values are changed during the diffusion stage [5]. An efficient image encryption method includes treating an image as a stream of bits and then encrypting this stream utilizing traditional encryption methods like the Advanced Encryption Standard (AES), the International Data Encryption Algorithm (IDEA), and the Triple Data Encryption Algorithm (TDEA). These methods suffer from a strong correlation between neighbouring pixels and redundant data [6, 7]. As a result, different encryption approaches based on various technologies have been proposed to provide better security to images [8], such as frequency domain transformation [9], chaos theory [10], Deoxyribonucleic Acid (DNA) coding [11], and compressive sensing [12, 13]. One of the most well-known approaches is using chaotic maps with image encryption. Chaos-based image encryption algorithms provide robust results in terms of complexity, security, and speed due to their characteristics, such as pseudo-randomness, ergodicity, non-periodicity, and sensitivity to initial conditions. [14]. Due to the sensitivity of chaotic maps to initial conditions, the control parameters and initial conditions must be carefully selected. Otherwise, the results will be unsatisfactory, the secret keys can be predicted, and unauthorized individuals can access data [15]. Metaheuristics techniques have been used to optimize appropriate control parameters to overcome this problem. Figure 1 presents the block diagram of the proposed chaos-based image encryption algorithm. The main contributions of the proposed work are:

(1) Presenting a novel approach utilizing metaheuristic optimization algorithms to enhance the security and robustness of image encryption based on chaotic maps, leveraging the chaotic maps as objective functions to improve the optimization process through the inherent complexity and randomness of chaotic sequences.
(2) Extensive investigation and fine-tuning of control parameters for eight different chaotic maps using nine metaheuristic optimizers are conducted, resulting in the generation of highly secure encryption keys with improved randomness and complexity.

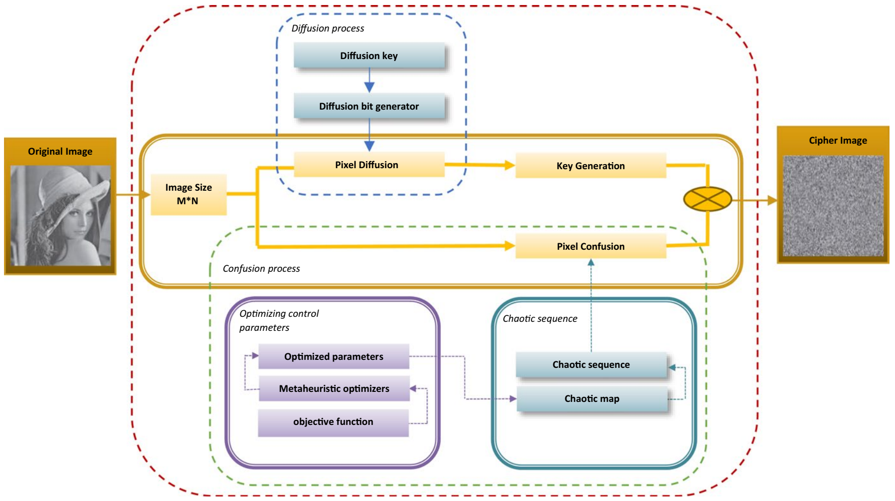


Fig. 1 Structure diagram of the proposed system

(3) Proposing a novel confusion process using optimized chaotic sequences to permute image pixels, thereby strengthening the encryption scheme and introducing an additional layer of security.
(4) Conducting a comparison between encryption methods with and without optimization, the potential of metaheuristic optimizers is evaluated, and the most effective technique in terms of security and robustness is identified.
(5) Introducing a robust image security algorithm that effectively resists various attacks and distortions, focusing on mitigating the impact of periodic noise, thus enhancing image confidentiality and integrity.
(6) Evaluating the performance of the eight proposed chaotic-based image encryption methods on different standardized images demonstrates the applicability and effectiveness of our approach across various image types and characteristics.

The rest of this paper is structured as follows. Section 2 presents an overview of related work. Section 3 illustrates chaotic maps. Section 4 demonstrates the different metaheuristic optimizers that are used in this paper. Section 5 presents the model proposed. Section 6 introduces our proposed model's performance metrics and experimental results. Finally, the conclusions are shown in Sect. 7.

2 Related work

Mozaffari [16] has designed an image encryption system relying on bitplane decomposition and a local binary pattern technique, which converts the plain grayscale image to a group of binary images. After that, pixel substitution and permutation are performed using a genetic algorithm. Finally, the scrambled bitplanes are merged to create an encrypted image. This technology enhanced the encryption speed and allowed it to be used in real-time applications. Talhaoui et al. [17] have suggested an approach to encrypting images relying on the Bülban chaotic map, which produces only a few random columns and rows. Then the substitution-permutation process was used for pixels, and a circular shift was used for rows and columns. At last, the XOR operation and the modulo function are used together to mask the values of the pixels and avoid information leakage. This approach proved to be secure and fast for real-time image processing. Saravanan et al. [18] have proposed an optimized Hybrid Chaotic Map (HCM) approach for image encryption, integrating 2DLCM and PWLCM. Parameter optimization using the Chaos-Induced Whale Optimization Algorithm (CI-WOA) enhances performance and ensures efficient and secure image transmission. Rezaei et al. [19] have presented an evolutionary-based image encryption method using a two-dimensional Henon chaotic map. The parameters of the chaotic map, including α and β, are fine-tuned using the metaheuristic Imperialist Competitive Algorithm (ICA), which ensures that the generated pseudorandom numbers are unique, enhancing the security of the encryption process. Experimental results demonstrate the effectiveness of the proposed method in protecting image content against common attacks. Shahna et al. [20] have introduced a system for encrypting images relying on a symmetric key, cyclic shift operation, scan method, and chaotic map. Diffusion and confusion processes are implemented using the Henon chaotic map and Hilbert curve. Image scrambling involves both bit-level and pixel-level permutations. From the double-scrambled image, the final encrypted image is produced. The proposed technique provides high security. Ghazvini et al. [21] have suggested a hybrid image encryption technique based on chaotic maps and a genetic algorithm. During the confusion phase, Chen's map of chaos was used, while the logistic-sine map was used during the diffusion phase. The encrypted image is then optimized using a genetic algorithm. The proposed technique is resistant to security attacks. Noshadian et al. [22] have performed an image encryption scheme that employs a chaotic logistic function to change the value of the pixels and a genetic algorithm to decrease the correlation among adjacent pixels. Wang et al. [23] have developed a scheme for encrypting images that relies on logistic maps, DNA sequences, and multi-objective particle swarm optimization. The chaotic logistic map and DNA encoding produce random DNA mask images. Then, particle swarm optimization is used to find the best balance between entropy and correlation. The suggested scheme has good security and is resistant to security attacks. Wang et al. [24] have proposed an image encryption method that uses particle swarm optimization (PSO) to select the parameters and initial values of chaotic systems for generating a high-chaotic sequence. The proposed approach uses the chaotic system parameters and initial values as particles, making it more computationally efficient for real-time applications. The encryption algorithm includes key selection, chaotic sequence pre-processing, block scrambling, expansion, confusion, and diffusion, with the PSO-optimized chaotic sequence used for scrambling and diffusion. Experimental results demonstrate that the proposed method is secure. Ferdush et al. [25] have proposed a method for encrypting images that uses a logistic map, a genetic algorithm, and particle swarm optimization. The logistic map is first used to create the initial population. Then, the genetic algorithm is used to encrypt the image, and the particle swarm optimizer is used to make the best-encrypted image. The entropy of the proposed method is high. Alghafis et al. [26] have utilized an image encryption technique based on chaotic sequencing and DNA. Random sequences are generated using logistic, Henon, and Lorenz chaotic systems. Then, pixel confusion and diffusion are created using DNA computations in combination with the chaotic systems. The security tests revealed that this algorithm is well-suited to encrypt digital images and can also protect the privacy of other digital contents, such as audio and video files. Latha et al. [27] proposed an optimized two-dimensional (2D) chaotic mapping for image encryption. A metaheuristic fitness-based Sea Lion optimization algorithm is used to fine-tune the initial parameters of the chaotic system. This algorithm tries to maximize the information entropy model for chaotic key generation, which leads to the best initial parameters. The experimental results demonstrate that the proposed model has high entropy. Kumar et al. [28] have proposed an algorithm to encrypt images by combining the Henon map with a one-dimensional elementary cellular automaton. The diffusion process uses an elementary cellular automaton, and pseudo-random numbers are generated using the Henon map. The proposed algorithm is resistant to statistical attacks. Babaei et al. [29] have developed a method for encrypting images using DNA, a chaotic map, and recursive cellular automata (RCA). The logistic map is utilized for a cellular shift of the image rows and columns during the permutation phase. The RCA and DNA are then utilized to alter the pixel intensity during the diffusion phase. The proposed algorithm is secure and has good entropy. Ghazanfaripour et al. [30] have presented an approach for encrypting images based on hill diffusion, column-row diffusion, and a 3D scale-invariant modular chaotic map. Column-row diffusion and hill diffusion are used for adjacency pixel mixing in the plain image and pixel substitution. Pixel permutation is done by utilizing 3D scale-invariant chaotic maps. The proposed algorithm has a large key space but takes a long time to execute.

3 Chaotic maps

Chaotic maps are subsets of nonlinear dynamical methods like noise signals but are much more specific [32]. One of the essential things about chaotic maps is that they are very sensitive to initial conditions, which indicates that a slight change in the starting value will significantly affect the output, making them suitable for encrypting images [33, 34]. In this paper, eight chaotic maps have been studied, and their properties have been analysed in order to produce chaotic sequences:

3.1 Logistic map

The following equation has been utilized to define the logistic map:

$$x_{n+1} = ax_n\left(1 - x_n\right) \tag{1}$$

where x is the chaotic sequence in the range [0 1], n is the iteration number, and a is the control parameter having values in the range [0 4] [35], while the system is chaotic when a, ϵ [3.5699456, 4] [36].

3.2 Sine map

The sine map has been calculated using the following mathematical definition:

$$x_{n+1} = asin(\pi x_n) \tag{2}$$

where x is the chaotic sequence in the range [0 1], n is the iteration number, and a is the control parameter with values in the range [0 1] [37], while the system is chaotic when $a \in [0.87,1]$ [38].

3.3 Gauss map

Gauss map is also known as mouse map; the gauss map is mathematically formulated as [39]:

$$x_{n+1} = \begin{cases} 10 \\ \dfrac{1}{\mathrm{mod}\,(x_n, 1)} & \text{Otherwise} \end{cases} \cdot \tag{3}$$

where, x is the chaotic sequence in the range [0 1], and n is the iteration number [40].

3.4 Circle map

Circle map is one-dimensional chaotic map, and is mathematically formulated as [41]:

$$X_{i+1} = \mathrm{mod}\left(x_i + \mathrm{b} - \left(\frac{\mathrm{a}}{2\pi}\right)\sin\,(2\pi x_i), 1\right) \tag{4}$$

where x is the chaotic sequence in the range [0 1], It has two parameters a and b, the power of non—linearity is represented by parameter a, while the externally applied frequency is represented by parameter b [42].

3.5 Tent map

The following mathematical definition gives the equation of the tent map:

$$X_{n+1} = \begin{cases} rx_n & if\ \ 0 < x_n < 0.5 \\ r(1 - x_n) & if\ \ 0.5 \le x_n < 1 \end{cases} \tag{5}$$

where x is the chaotic sequence in the range [0 1], n is the number of iterations, and r is the control parameter with values in the range [0 2], while the system is chaotic when $r \in [1.4, 2]$ [43].

3.6 Chebyshev map

The following formula describes the Chebyshev map:

$$x_{n+1} = \cos\left(\mathrm{acos}^{-1}\left(x_n\right)\right) \tag{6}$$

where x is the chaotic sequence in the range [−1 1], n is the iteration number, and a is the control parameter with values in the range [0 2], while the system is chaotic when $a \in [1\ 2]$ [44].

3.7 Singer map

The singer map is mathematically formulated as:

$$x_{n+1} = \mu\left(7.86x_n - 23.31x_n^2 + 28.75x_n^3 - 13,3x_n^4\right) \tag{7}$$

where x is the chaotic sequence in the range [0 1], n is the iteration number, and μ is the control parameter with values in the range [0 2], while the system is chaotic when $\mu \in [0.9\ 1.08]$ [45].

3.8 Piecewise map

The piecewise map is calculated using the following equation:

$$x_{i+1} = \begin{cases} \frac{x_n}{a} & if\ 0 \le x_n \le a \\ \frac{x_n - a}{0\cdot 5 - a} & if\ a \le x_n < 0.5 \\ \frac{1 - a - x_n}{0\cdot 5 - a} & if\ 0.5 < x_n < 1 - a \\ \frac{1 - x_n}{a} & if\ 1 - a \le x_n < 1 \end{cases} \tag{8}$$

where x is the chaotic sequence in the range [0 1], n is the number of iterations, and a is the control parameter with values in the range [0 2], while the system is chaotic when $a \in [1.4, 2]$ [46]. Figure 2 shows the bifurcation diagram of the chaotic maps, where horizontal axis denotes the parameter r and vertical axis denotes x and each trajectory of the map about x with a fixed x is plotted as dots on the figure.

4 Metaheuristics optimizers

Meta-heuristic optimizers are generally used to obtain optimal solutions. In this paper, the chaotic maps' initial conditions and control parameters are fine-tuned using the following meta-heuristic optimizers:

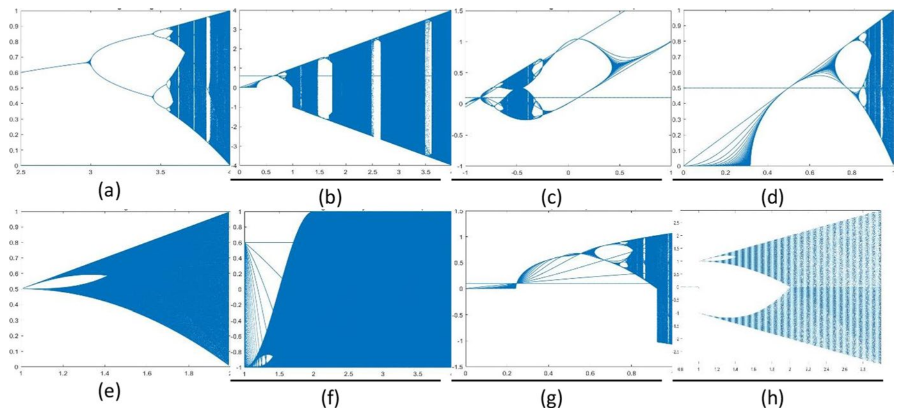


Fig. 2 The bifurcation diagram of **a** Logistic map, **b** Sine map, **c** Gauss map, **d** Circle map, **e** Tent map, **f** Chebyshev map, **g** Singer map, and **h** Piecewis map

4.1 Genetic algorithm (GA)

GA belongs to the wider family of evolutionary algorithms (EA). It involves four steps: selection, mutation, crossover, and inheritance [47, 48]. To simulate the evolution process, GA generates several solutions called population [49]. The initial population is represented using the following equation:

$$x = \left\{ x_1, x_2, x_3, \dots, x_n \right\} \tag{9}$$

GA uses a fitness evaluation function to determine the fitness value of each solution:

$$f\left(x_1\right), f\left(x_2\right), f\left(x_3\right), \dots \dots \dots, f\left(x_n\right) \tag{10}$$

$$P_i = \frac{f\left(x_i\right)}{\sum_{i=1}^{N} f\left(x_i\right)} \tag{11}$$

where, x represents the chromosome, n indicates the size of chromosomes population, i is the number of chromosomes, and P_i is the selection probability of the ith chromosome. Accordingly, the crossover and mutation operators have been applied to the best solutions to generate a new population. It chooses the best chromosomes to crossover exactly as it does in nature [50]. Random individuals' random genes are changing using mutation. The mutation is used to avoid local optimum solutions [51].

4.2 Particle swarm optimization

Eberhart and Kennedy proposed PSO at the end of the 20th century based on their research about bird foraging and migration behaviour [52]. Each group member's unique perception ability allows them to recognize the best local and global individual positions and adjust their next behavior accordingly [53]. Individuals are viewed as particles in a multi-dimensional search space in the algorithm, with each particle representing a potential solution to the optimization problem. The particle characteristics are described using four parameters: velocity, fitness value, and position [54]. The fitness function determines the fitness value. The particle changes its moving direction and distance independently based on the optimal global fitness value and reaches the best solution by iteration [55]. The population of n particles exists in a D-dimensional space represented by the following equations:

$$x = \{x_1, x_2, \ldots, x_n\} \tag{12}$$

$$x_i = [x_{i1}, x_{i2}, \ldots, x_{iM}] \tag{13}$$

The following equations express the location, velocity and position of the *i*th particle [56]:

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{\mathrm{i}D}) \tag{14}$$

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{\mathrm{i}D}) \tag{15}$$

$$P_{best} = (p_{i1}, p_{i2}, \ldots, p_{\mathrm{i}D}) \tag{16}$$

$$g_{best} = (g_{i1}, g_{i2}, \ldots, g_{iD}) \tag{17}$$

where, X_i is the current location, V_i is the current velocity, P_{best} is the best local position, and g_{best} is the best global position. The velocity and position are updated as follows:

$$\begin{aligned} v_{id}(t+1) = v_{id}(t) + c_1 \times \mathrm{Rand} \times \left[Pbest_{id}(t) - x_{id}(t)\right] + c_2 \times \mathrm{Rand} \\ \times \left[Gbest_d(t) - x_{id}(t)\right] \end{aligned} \tag{18}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{19}$$

where, c_1 represents the cognitive acceleration coefficient, and c_2 is the social acceleration coefficients.rticles tend to move towardgood areas in the search space in response to information spreadingthrough the swarm. A particle moves to a new position calculatedby the velocity updated at each time step tby Eq. (3). Equation (4) is then used to calculate the new velocity as the sum of the previou

4.3 Whale optimization algorithm

Mirjalili presented WOA for solving and optimizing problems [57]. This algorithm was inspired by the bubble-net attacking technique used by humpback whales [58]. As a result, the WOA is often used in many fields, such as image encryption, energy, structural optimization, management, and so on [59]. Whales have a unique way of chasing a group of krill or small fish in which they build a nine-shaped net with bubbles that capture the fish, allowing them to easily eat the fish. This technique is known as "spiral bubble-net feeding" [60]. The WOA consists of the following three stages:

(I) ***Encircling prey***

Whales use the following formulas to encircle A group of krill or small fish:

$$\vec{\mathrm{D}} = \left|\vec{C} \cdot \overrightarrow{X^*}(t) - \vec{X}(t)\right| \tag{20}$$

$$\vec{X}(t+1) = \overrightarrow{X^*} - \vec{A} \cdot \vec{D} \tag{21}$$

$$A = 2ar - a \tag{22}$$

$$C = 2r \tag{23}$$

where $\vec{\mathrm{X}}$ represents the position vector, $\overrightarrow{\mathrm{X}^*}$ represents the position vector of the best solution, t represents the current iteration, A and C represent coefficient vectors, a is reduced linearly from 2 to 0, and r is a random vector in the range of [0, 1].

(II) ***Bubble-net attacking method:***

The mathematical model for exploitation phase is as follows:

$$\vec{X}(t+1) = \vec{D}' \cdot \mathrm{e}^{bl} \cdot \cos(2\pi L) + \overrightarrow{X^*}(t) \tag{24}$$

$$\vec{D}' = \left|\vec{x}^* - \vec{x}(t)\right| \tag{25}$$

where L denotes a random number between -1 and 1, b denotes a constant that specifies the logarithmic shape and is the distance between the prey and the whale. Assume that the whale will follow a shrinking mechanism or a spiral path, with a 50% chance of each [61]. The following formula is used to represent this:

$$\vec{X}(t+1) = \begin{bmatrix} \overrightarrow{X^*} - \vec{A} \cdot \vec{D} & (if)\ P < 0.5 \\ \vec{D}^{`} \cdot \mathrm{e}^{bl} \cdot \cos(2\pi L) + \overrightarrow{X^*}(t) & (if)\ P \geq 0.5 \end{bmatrix} \tag{26}$$

where P is a random number in the range [0, 1].

(III) ***Search for prey***

The mathematical model for Exploration phase is as follows:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_{\text{rand}} - \vec{X}\right| \tag{27}$$

$$\vec{X}(t+1) = \vec{X}_{\text{rand}} - \vec{A} \cdot \vec{D} \tag{28}$$

where $\vec{X}_{\text{rand}}$ is position vector selected randomly from the current population.

4.4 Dragonfly algorithm

DA is a randomized search optimizer mimicking a dragonfly swarm's flying movement [62, 63]. Dragonfly swarms follow the following rules: separation, alignment, cohesion, attraction, and deflection [64, 65]:

(I) *Separation (S)* The swarms are separated from other individuals in this step, avoiding clashes with neighbours [65]. Separation is calculated using the following equation:

$$s_i = -\sum_{j=1}^{N} X - X_j \tag{29}$$

Where X represents the current individual's position, X_j represents the position of j-th neighbouring, and N represents the number of neighbouring individuals.

(II) *Alignment (A)* This phase describes the process of a dragonfly's speed adjusting to the speed vectors of other nearby dragonflies in the swarm [66]. This phase is expressed using the following equation:

$$A_i = \frac{\sum_{j=1}^{N} v_j}{N} \tag{30}$$

Where v_j is the velocity of the close individual (j).

(III) *Cohesion (C)* This phase refers to the swarm's attraction to the centre of the swarm's team [67]. The cohesion is calculated using the following equation:

$$c_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \tag{31}$$

Where X_j denotes the location of the jth close individual, and X denotes the location of the current individual.

(IV) *Attraction* This phase shows the attraction of swarms toward the food origin [68]. The attraction is calculated using the following equation:

$$F_i = X^+ - X \tag{32}$$

Where, x^+ denotes the position of the food source, and X denotes the current individual's location.

(V) *Distraction* In this phase, the dragonfly is diverted from the enemy [69]. The distraction is computed using the following equation:

$$E_i = X^- + X \tag{33}$$

Where, X^- denotes the location of the enemy, and X denotes the current individual's location. To locate the best solution to a specified optimization problem, DA assigns a step vector and a position vector to every search agent in the swarm [71]. The step vector is calculated using the following equation:

$$\Delta X_{t+1} = \left(sS_i + aA_i + cC_i + fF_{i+e}E_i\right) + \omega\Delta X_t \tag{34}$$

The position vector is updated using the following equation:

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{35}$$

where $s, a, c, f, ande$ denote the weighting factors for separation, alignment, cohesion, attraction, and distraction, and t denotes the current iteration.

4.5 Grey wolf optimization

GWO is an algorithm inspired by swarm intelligence [72], and it is based on the leadership hierarchy and hunting behaviour of grey wolves in their natural habitat [73]. The leadership hierarchy in the model simulation is divided into four wolf types [74]: alpha (the highest level), beta, delta, and omega (the lowest level). The first level begins with the best solution in alpha and progresses downward in order [75, 76]. The steps of GWO are as follows:

(I) *Encircling prey* The following equations are used to represent the encircling process [76]:

$$\vec{\mathrm{D}} = \left|\vec{\mathrm{C}} \cdot \overrightarrow{\mathrm{X_P}}(\mathrm{t}) - \vec{\mathrm{X}}(\mathrm{t})\right|. \tag{36}$$

$$\vec{X}(t+1) = \overrightarrow{X_P} - \vec{A} \cdot \vec{D} \tag{37}$$

where $\vec{A}$ denotes random vectors, X_P denotes the prey's position vector, t denotes the current iteration, victor $\vec{C}$ has random range of [0 2], and $\vec{X}$ denotes grey wolf's position vector. The vectors of coefficients $\vec{A}$ and $\vec{C}$ are obtained with the following equations:

$$\vec{A} = 2\vec{a} \cdot \overrightarrow{r_1} \tag{38}$$

$$\vec{C} = 2 \cdot \overrightarrow{r_2} \tag{39}$$

(II) *Hunting* is a behavior driven by alpha, while beta and delta might occasionally play a role [77]. The following formulas demonstrate hunting behavior:

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \vec{X}_\alpha - \vec{X}\right| \tag{40}$$

$$\vec{D}_\beta = \left|\overrightarrow{C_2} \cdot \vec{X}_B - \vec{X}\right| \tag{41}$$

$$\vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \tag{42}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \left(\vec{D}_\alpha\right) \tag{43}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \left(\vec{D}_\beta\right) \tag{44}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \left(\vec{D}_\delta\right) \tag{45}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{46}$$

Where $\vec{X}_\alpha, \vec{X}_\beta, \& \vec{X}_\delta$ denote the first three best solutions at iteration t.

(III) *Attacking Prey* When the prey stops moving, the hunting process ends, the wolves attack, and the values of $\vec{a}$ and $\vec{A}$ are decreased mathematically [78].

(IV) *Search for prey* The search phase mainly uses the α, β, and δ positions. Grey wolves look for prey individually before banding together to attack it. To model divergence mathematically [79], the $\vec{A}$ is a random value in which if $\vec{A} > 1$, the wolves will diverge, and if $\vec{A} < 1$, the wolves will converge [80].

4.6 Moth-flame optimization

MFO is an algorithm inspired by swarm intelligence [82]. Moreover, it is based on the moths' transverse orientation navigation method [83]. In MFO, individuals are called moths (M), a population is a group of moths, and the best solution for each moth is called flame (F) [84]. Only one flame for each moth is considered the best solution. This flame will be updated if a better solution arises during the iterations [85]. Moths and flames are generally represented as n * d matrices, where m denotes the number of moths and d denotes the number of dimensions [86]. Following is the moth population matrix:

$$\mathrm{M} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,d} \end{bmatrix} \tag{47}$$

where N denotes the moth population, $m_1, m_2, \cdots, m_n$ denotes n moths. The moth's fitness is calculated once per iteration and saved in a vector called (OM) [87]. Following is the moth's fitness vector:

$$\mathrm{OM} = \begin{bmatrix} OM_1 \\ OM_2 \\ OM_3 \\ OM_N \end{bmatrix} \tag{48}$$

The following is the flame matrix (F) and its fitness vector (OF):

$$\mathrm{F} = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,d} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n,1} & f_{n,2} & \cdots & f_{n,d} \end{bmatrix}. \tag{49}$$

$$\mathrm{OF} = \begin{bmatrix} Of_1 \\ Of_2 \\ Of_3 \\ Of_n \end{bmatrix} \tag{50}$$

where F denotes flame set, $f_1, f_2, \cdots, f_n$ denotes n flames. MFO algorithm is composed of the 4 phases listed below [88]:

In the first phase, the moth's location is updated. The equation to find the best optimal value is given as:

$$M_i = S(M_i, F_j) \tag{51}$$

where M_i denotes the ith moth, Fj denotes the jth Flame, and S denotes the moth's spiral function.

In the second phase, the moth's spiral function is calculated using the following equations:

$$S(M_i, F_j) = D_i * \mathrm{e}^{bt} * \cos(2\pi t) + F_j \tag{52}$$

$$D_i = \left|F_j - M_i\right| \tag{53}$$

where D_i denotes the distance from the ith moth to the jth Flame, t denotes a random value between − 1 and 1, and b denotes the spiral shape factor.

In the third phase, the number of flames is updated using the attenuation function [89, 90]. The attenuation function is defined using the following equation:

$$Nf = round\left(N - R * \frac{N-1}{T}\right) \tag{54}$$

where R denotes the current number of iterations, T denotes the maximum iterations number, and N denotes the maximum flame number.

In the fourth phase, the best moth is obtained if the termination criterion is met. [91].

4.7 Sine cosine algorithm

SCA is a population-based optimizer that starts with a randomly generated population [92, 93]. The location of solutions is updated using sine and cosine equations [94]. The optimization process uses the exploration phase to find the most desirable location in the search area and the exploitation phase to find the best solution in the proposed region [95]. SCA is expressed as follows:

$$x_i^{t+1} = x_i^t + r_1 * \left(\sin\left(r_2\right)\right) * \left|r_3 P_i^t - x_i^t\right| \tag{55}$$

$$x_i^{t+1} = x_i^t + r_1 * \left(\cos\left(r_2\right)\right) * \left|r_3 P_i^t - x_i^t\right| \tag{56}$$

where x_i^t denotes the current solution's position, r_1 denotes a control parameter responsible for achieving the balance between exploitation and exploration, r_2, and r_3 denote random values, and P_i^t denotes the target solution. The two SCA equatns are combined into a single Equation:

$$x_i^t = \begin{bmatrix} x_i^{t+1} = x_i^t + r_1 * \left(\sin\left(r_2\right)\right) * \left|r_3 P_i^t - x_i^t\right| \ (if) r_4 < 0.5 \\ x_{\mathrm{i}}^{t+1} = x_i^t + r_1 * \left(\cos\left(r_2\right)\right) * \left|r_3 P_i^t - x_i^t\right| \ (if) r_4 \geq 0.5 \end{bmatrix}. \tag{57}$$

r1, r2, r3, and r4 are updated using the following equations:

$$r_1 = a - t\frac{a}{T} \tag{58}$$

$$r_2 = 2 * \pi * \text{rand}() \tag{59}$$

$$r_3 = 2 * \text{rand}() \tag{60}$$

$$r_4 = \text{rand}() \tag{61}$$

where a denotes a constant, T denotes the maximum number of iterations, r_2 denotes the next solution's movement direction, r_3 is utilized to identify a random weight for the destination, and r_4 is utilized to toggle between sine and cosine functions [96].

4.8 Multi verse optimization

MVO is a population-based optimizer inspired by the big bang theory and the multiverse theory [97], which starts with a randomly generated population [98]. MVO algorithm comprises three main principles: White hole, Black hole, and Wormhole [99, 100]. The search space is explored by white holes and black holes, while the search space is exploited by warm holes [101]. Individual development for each population can be conducted using one of the theories concerning the existence of multiple universes [102]. For each iteration, every solution is referred to as a "universe" [103]. The universes are sorted depending on the inflation rates, and one of them is picked to have a white hole using the roulette wheel [104]. MVO is expressed mathematically as follows:

$$\text{U} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \tag{62}$$

$$X_i^J = \left\{ \begin{matrix} X_k^J, r_1 < NI(U_i) \\ X_i^J, r_1 \geq NI(U_i) \end{matrix} \right\} \tag{63}$$

where d denotes the number of the parameters, n denotes the universe's number, X_i^J denotes the jth parameter of the ith universe, X_k^J denotes the jth parameter of the kth universe, r1 denotes a random value with range [0 1], and $NI(U_i)$ denotes the ith universe's normalized swell rate. An additional operation used to create local modifications for each universe is outlined below [105]:

$$X_i^J = \left\{ \begin{matrix} \left\{ \begin{matrix} (x_j + TDR * (ub - lb) * r_4 + lb), r_3 < 0.5 \\ (x_j - TDR * (ub - lb) * r_4 + lb), r_3 \geq 0.5 \end{matrix} \right\}, r_2 < WEP \\ X_i^J, r_2 \geq WEP \end{matrix} \right\}. \tag{64}$$

where x_j denotes the jth parameter of the best universe, ub denotes the upper limit, lb denotes the lower limit, TDR denotes travelling distance rate, WEP denotes wormhole existence probability, and r2, r3, and r4 denote random number with range [0 1]. The wormhole existence probability and the Travelling distance rate are calculated using the following equations [106]:

$$WEP = WEP_{\min} + t * \left(\frac{WEP_{max} - WEP_{min}}{t_{max}} \right) \tag{65}$$

$$(TDR) = 1 - \left(\frac{t}{t_{max}} \right)^{1/n} \tag{66}$$

where WEP_{max}&WEP_{min} denotes two constantans in which ($WEP_{max} = 1$ & $WEP_{min} = 0.1$), t denotes iteration, t_{max} denotes the maximum iterations number, and $1/n$ denotes exploitation accuracy constant.

4.9 Ant-lion optimization

ALO is inspired by nature and relies on ant lion hunting mechanisms [107]. It simulates the intelligent behaviour of ant lions through the life cycle of larvae [108]. In the ALO, the roulette wheel and random walks of ants are used to eliminate local optima [109]. The ALO algorithm is divided into six stages, which are as follows [110]:

(I) ***Random walk of prey***

Ants (prey) are supposed to walk randomly [111], The following cumulative sum equation represents the random walk:

$$x(t) = \left[0, Cumsum\left(2r\left(t_i\right)\right) - 1, \ldots, Cum\sin\left(2r(T)\right) - 1\right] \tag{67}$$

where *Cumsum* denotes the sum, t_i denotes random walk step at the i-th iteration, T denotes the total number of iterations, and $r(t)$ denotes a random function, which is represented in the following equation:

$$r(t) = \begin{cases} 1, rand(0,1) > 0.5 \\ 0, rand(0,1) \leq 0.5 \end{cases} \tag{68}$$

A normalization of position in search space is described using Eq. (69):

$$X_j^t = \frac{\left(X_j^t - a_j\right) \times \left(d_j^t - c_J^t\right)}{b_j - a_j} + c_j \tag{69}$$

where, X_j^t denotes the position of *j-th* variable at iteration t, a_j and b_j denote the minimum and maximum of random walk in the *j-th* variable, and c_J^t and d_j^t denote the minimum and maximum of the *j*-th variable at iteration t.

(II) ***Building traps***

Antlion traps affect the random walk ants [112], as shown by the following equations:

$$c_j^t = Antlion_j^t + c^t, \tag{70}$$

$$d_j^t = Antlion_j^t + d^t \tag{71}$$

where $Antlion_j^t$ represents the j-th position of the antlion at the iteration t.

(III) ***Setting trap***

ALO selects antlions based on their fitness value using a roulette wheel to mimic the antlions' hunting process [113].

(IV) ***Sliding of prey in a trap***

Once the ants drop into the trap, the antlion tosses sand into the pit's centre [114]. As a result, the following equations denote the ant's movement:

$$c^t = \frac{c^t}{I}, \tag{72}$$

$$\mathrm{d}^t = \frac{\mathrm{d}^t}{I} \tag{73}$$

where I denotes a factor based on the ratio of a current iteration to the number of total iterations.

(V) ***Catching a prey/rebuilding a trap***

The following equation defines the final prey capture and repositioning of ant lions.

$$Antlion_j^t = Ant_j^t \ \ if\left(Ant_i^t\right) > f\left(Antlion_j^t\right) \tag{74}$$

where Ant_i^t denotes the ant's position at the t-th iteration.

(VI) ***Elitism rule***

The ant's position is updated based on the roulette wheel's average and the elite antlion's movement [115, 116], which is expressed using the following equation:

$$Ant_i^t = \frac{\left(R_A^t\right) + \left(R_E^t\right)}{2} \tag{75}$$

where R_A^t represents the random walk taken by an ant across antlion picked during the process of the roulette wheel selection, and R_E^t represents the random walk of the same ant across the elite antlion at the t-th iteration.

5 Proposed algorithm

Encryption-based chaotic maps are extremely sensitive to initial conditions and control parameters; therefore, the generated sequence will not be chaotic if these parameters are not chosen precisely. The attacker could break the secret key and read the data. Manually picking these seed values takes a long time and is very tedious. As a result, metaheuristic optimizers are used to optimize these parameters. In this algorithm, we compare the performance of different metaheuristic optimizers to tune the control parameters of eight different chaotic maps.

First: Meta heuristic optimizing

Meta-heuristic optimizers are used to fine-tune the chaotic maps' initial conditions and control parameters. In this paper, we compare the performance of nine different meta-heuristic optimizers. The basic steps of optimization are summarized as follows:

Step 1 Produce the initial population of individuals.
Step 2 Set the objective function, the chaotic map function, whose initial condition and control parameters will be tuned.
Step 3 Assessing every individual in the population using the objective function.
Step 4 A new population is formed by altering existing population solutions using evolutionary processes.
Step 5 The search parameter is updated continuously until it reaches the terminating criteria regarding the number of iterations.
Step 6 The algorithm will look for the best solution that meets the objective function at each iteration.
Step 7 The obtained solution is used as the chaotic map's initial condition and control parameter.
Second: Chaotic level Confusion:

The original image's pixels are permuted to decrease the correlation among neighbouring pixels. The basic steps of confusion are summarized as follows:

Step 1 Input the original image (I) with dimensions M*N, where M is the row size, and N is the column size.
Step 2 Convert the original image (I) to a 1*MN-dimensional vector (V).

Step 3 The tuned initial condition and control parameters are used to generate the chaotic sequence vector (x).
Step 4 The elements of the chaotic sequence vector are sorted using the following equation:

$$[so, in] = \text{sort}(x) \tag{76}$$

where so is the newly sorted chaotic sequence vector, and in is the index of the sorted vector.

Step 5
se the following equation to scramble the position of the original image vector (V):

$$V_2(\text{i}) = V(in(i)) \tag{77}$$

Figure 3 shows a simple illustration of the chaotic level permutation process. Fig. 4.

Third diffusion level

The key image is used at this level to diffuse the permuted image. Following is a summary of the fundamental steps of the diffusion level:

Step 1 In order to obtain the diffusion vector (D), the following equation is iterated M*N times:

$$D(t+1) = \cos\left(p\cos^{-1}(D(t))\right) \tag{78}$$

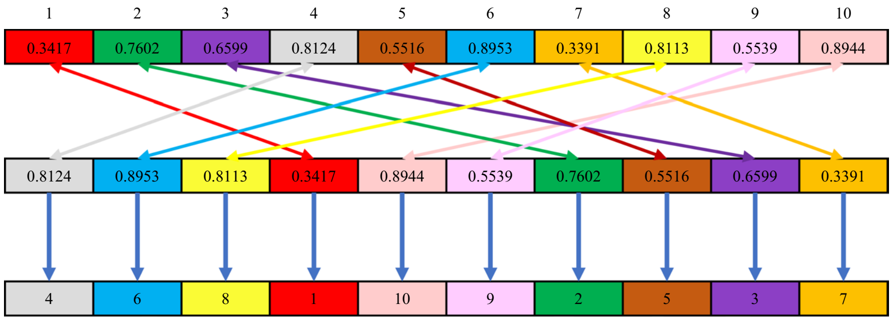


Fig. 3 Chaotic level Permutation process

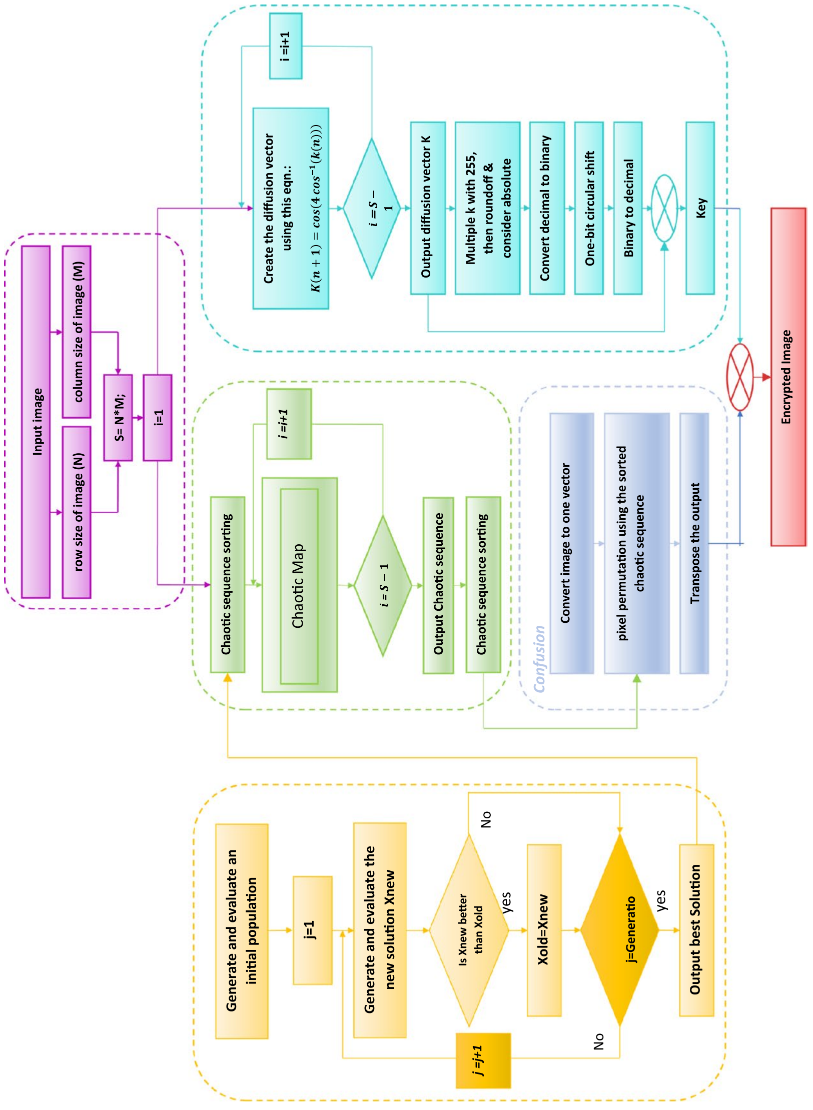


Fig. 4 Block Diagram of the Encryption Process

Step 2: The 1*MN diffusion vector (D) is multiplied by 255, rounded off, and the absolute value (k) is calculated as follows:

$$k = abs(round(D*255) \tag{79}$$

Step 3 The obtained vector is converted from decimal to binary using the *de2bin* function, and then the bits are shifted using a one-bit circular shifting process.
Step 4 The shifted vector is converted from binary to decimal (R), and the obtained vector is combined with victor (k) using the exclusive OR operation to generate a key stream.
Step 5 The generated key stream is XORed with the permuted image to generate the encrypted image. Fig. 4 shows the block diagram of the proposed encrypting scheme. Algorithm 1 presents the pseudo-code for the Grey Wolf Optimization Algorithm with the Logistic Map. It illustrates one of the approaches used in our study to fine-tune the control parameters of chaotic maps for image encryption. It is important to note that similar procedures were applied to the eight chaotic maps using the nine metaheuristic optimizers. Integrating these optimizers with the respective chaotic maps allowed us to optimize their initial conditions and control parameters, enhancing the overall encryption process. Algorithm 2 represents the pseudo-code for the whole encryption process.

Algorithm 1: Grey wolf optimization algorithm with logistic map

```
1. Initialize the population of grey wolves
2.   while stopping criterion is not met do:
3.       for each grey wolf in the population do:
4.           if fitness of the grey wolf is better than the current best
             then
5.           update the current best
6.           calculate the a and A values for the current grey wolf
7.               for each dimension in the search space do:
8.                   calculate the prey position based on the logistic map equation:
                     prey_position = r * current_position * (1 - current_position)
9.                   evaluate the objective function (logistic map equation) at the prey position:
                     fitness = objective_function(prey_position)
10.                     update the position of the grey wolf using the prey position and the grey
wolf's position:
                     new_position = prey_position + A * (prey_position - current_position) * rand()
11.              end for
12.      end for
13.      update the exploration and exploitation parameters:
         a = a - a * iteration / maximum_iterations
         A = 2 * A * (1 - iteration / maximum_iterations)
14.  end while
15. return the best solution found
```

Algorithm 2. Encryption Algorithm

```
Input  :  A plain-image of size M*N
output :   Encrypted image

A: INIT() // Metahuristics optimizers
1. Generate an initial population
2. evaluate an initial population
3. while(stopping criterion not satisfied) do
         X(new)<--Generate the new solution
         X(new)<--evaluate the new solution
               if X(new) better than X(old)
               X(old)=X(new)
               else
               check stopping criteriona
               end
         update best found solution
4. end
5. return best found Solution x(1)
B: Generating chaotic sequence
6. img <-- Read input image
7. M,N <-- size img
8. S   <-- M*N
9. x(1) <-- use optimized solution as initial parameter
10.   for(n=1:s-1)
         x(n+1) <-- chaotic map function
11.   end
12.   x(n) <-- chaotic sequence generated
13.   [so,in] <-- sort chaotic sequence elements
14.   return sorted chaotic sequence
C: Creation of diffusion key
15.   k(1) <-- set initial condition for k
16.   for (n=1:s-1)
         k(n+1) <-- cos(4*acos(k(n)))
17.   end
18.   k <-- K*255,then roundoff & consider absolute.
19.   R <-- convert k from decimal to binary
20.   R <-- One-bit circular shift of R
21.   R <-- convert R from Binary to decimal
22.   Key <-- K xor R
23.   return key
D: Confusion
24.   img <-- Read input image
25.   img <-- reshapes img to a one-dimensional column vector
26.   for (m=1:size of img)
      a. img(in(m)) <-- img(m), pixel permutation using the sorted chaotic
         sequence
27.   end
28.   img  <-- Return permuted vector
29.   timg <-- transpose the permuted vector
30.   E: Final Encryption
31.   Eimg <-- timg xor key
32.   Eimg <-- reshapes Eimg vector to a M*N matrix
33.   return encrypted image
```

6 Results

6.1 Lyapunov exponent

The Lyapunov exponent (LE) provides a numerical measure for assessing the sensitivity of control parameters in chaotic maps. Its mathematical representation is as follows:

$$LE = \lim_{n\to\infty} \frac{1}{n} \sum_{n=0}^{n-1} \ln\left|f'\left(x_i\right)\right| \tag{80}$$

where $f'\left(x_i\right)$ represents the derivative function of the function f(x), f(x) represents a 1D chaotic map. The parameter n denotes the number of iterations of the chaotic map. Table 1 shows Lyapunov exponents for optimized chaotic maps using metaheuristic optimizers compared to non-optimized maps. A higher Lyapunov exponent indicates stronger chaotic behaviour. The optimized chaotic maps have significantly higher (LE) values, indicating increased chaos. Notably, the optimized maps have higher (LE) values than the non-optimized ones, demonstrating the effectiveness of optimization in enhancing chaotic behaviour.

6.2 Performance metrics

The following performance metrics are used to measure and compare the performance of encryption algorithms based on eight optimized chaotic maps:

6.2.1 Statistical attack analysis

6.2.1.1 Histogram analysis It represents the frequency of an image's pixel distribution. The histograms of the primary image are non-uniform. In contrast, the histograms of encrypted images should be uniform to resist statistical attacks. The histograms of plain and encrypted images are shown in Fig. 5. It is noticeable that the histogram distributions of the plain images are unequal. In contrast, the histogram distributions of the encrypted images using encryption algorithms based on the eight chaotic maps are uniform, indicating that each chaotic map resists statistical attacks.

6.2.1.2 Entropy analysis Entropy measures the randomisation of the encrypted image's pixel intensities. An encryption algorithm with an entropy value of approxi-

Table 1 Lyapunov exponents

	Logistics	Sine	Gauss	Circle	Tent	Chebyshev	Singer	Piecewise
GA	1290.116	1354.917	1351.267	1237.972	1343.830	1408.528	1355.651	1351.400
WO	1186.008	1358.774	1351.381	1244.888	1339.396	1424.119	1355.947	1351.842
PSO	1290.704	1354.497	1347.615	1236.283	1352.456	1423.036	1356.638	1353.648
DA	1239.075	1358.180	1349.777	1243.109	1342.835	1414.821	1356.192	1352.447
GWO	1231.094	1356.368	1348.375	1238.377	1341.455	1391.594	1356.458	1350.004
MFO	1224.998	1351.094	1349.988	1238.208	1341.254	1406.309	1356.204	1352.375
SCA	1266.729	1357.351	1355.395	1238.211	1336.504	1420.995	1354.152	1356.028
MVO	1181.129	1358.977	1351.881	1238.246	1347.997	1424.709	1353.937	1349.570
ALO	1185.169	1350.651	1349.690	1241.536	1352.923	1430.378	1357.682	1353.648
No Optimizer	1040.021	1151.292	921.024	1207.726	1261.153	0.283	1289.922	1220.607

mately 8 is a robust algorithm. The entropy is calculated using the following formulas:

$$E(X) = \sum_{i=0}^{L-1} p(x_i) \log_2 \frac{1}{p(x_i)} \tag{81}$$

where, E(X) denotes the entropy of the plain image X, $p(x_i)$ denotes the occurrence probability, and L denotes total number of the tested pixels. The results in Fig. 6 demonstrate that the encrypted images' entropy is near the optimal value. However, encrypted images using Tent Chaotic map optimized with SCA has the best entropy. Table 2 demonstrates entropy comparison with other recent algorithms. From the previous results, it is concluded that the proposed algorithm provides the highest entropy value and is robust against entropy attacks.

6.2.1.3 Correlation analysis The correlation coefficient measures how closely two adjacent pixels in a picture are related. It is a term that describes the degree of similarity between two neighbouring pixels in the vertical, horizontal, and diagonal directions. A robust algorithm for image encryption decreases the correlation among neighbouring pixels in the encrypted image. The following formulas are used to calculate the correlation coefficient:

$$r_{xy} = \frac{\operatorname{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{82}$$

$$\operatorname{cov}(x, y) = \frac{1}{L} \sum_{i=1}^{L} \left(x_i - E(x)\right)\left(y_i - E(y)\right) \tag{83}$$

$$E(x) = \frac{1}{L} \sum_{i=1}^{L} x_i, \tag{84}$$

$$E(y) = \frac{1}{L} \sum_{i=1}^{L} y_i, \tag{85}$$

$$D(x) = \frac{1}{L} \sum_{i=1}^{L} \left(x_i - E(x)\right)^2, \tag{86}$$

$$D(x) = \frac{1}{L} \sum_{i=1}^{L} \left(y_i - E(y)\right)^2, \tag{87}$$

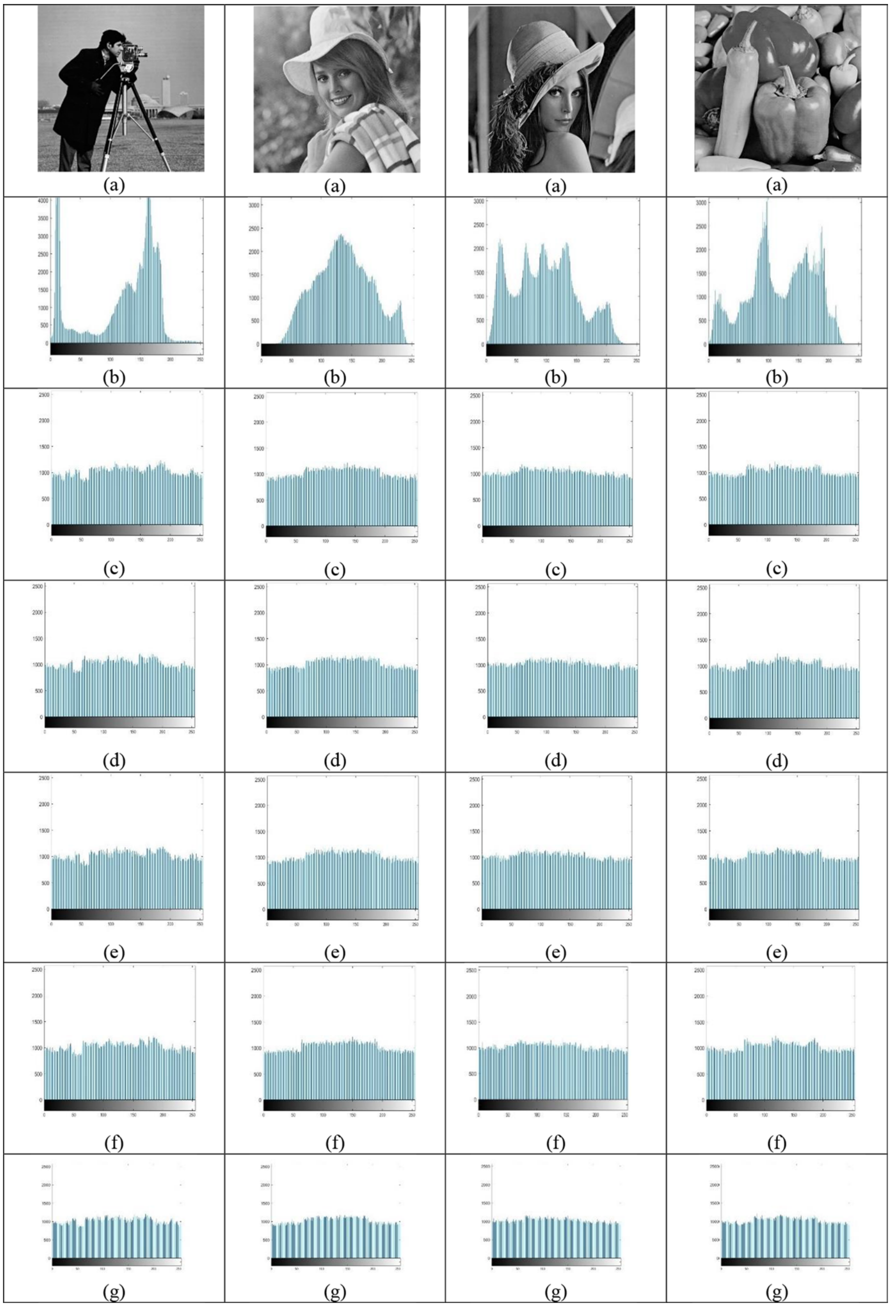


Fig. 5 **a** Original images, **b** Original image's histogram, The histogram of the encrypted images using **c** Logistic map, **d** Sine map, **e** Gauss map, **f** Circle map, **g** Tent map, **h** Chebyshev map, **i** Singer map, and **j** Piecewis map

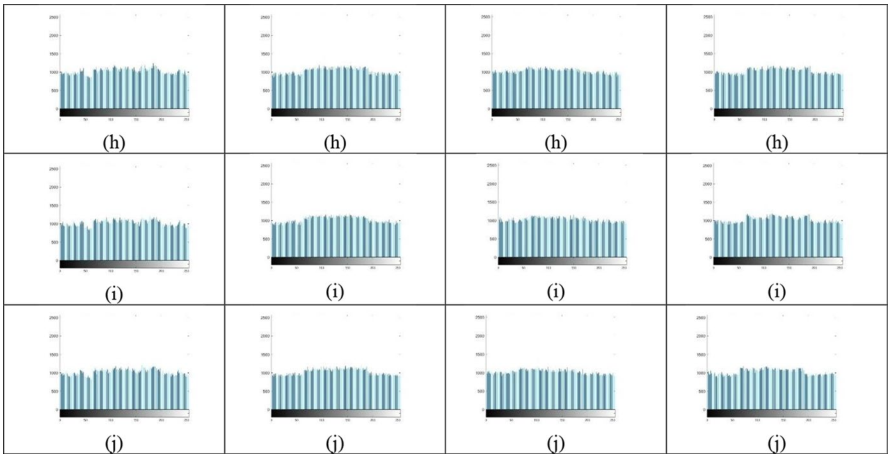


Fig. 5 (continued)

where x denotes the original image, y denotes the cipher image, and L denotes the number of pixels. Figures 7, 8, 9 and 10 illustrate the correlation of neighboring pixels in the three directions for the four images before and after encryption. As per Figs. 7, 8, 9 and 10, the correlation distribution of neighboring pixels in plain images is generally centered along the diagonal, indicating a significant relationship among neighboring pixels. While the correlation of the encrypted images is distributed randomly over the entire area, demonstrating a weak correlation between adjacent pixels. Table 3 displays the numerical results of the correlation among neighbouring pixels of the encrypted images, whereas the results achieved for the encrypted images are close to zero. Table 4 shows the comparison results with other methods. The correlation coefficients of our encryption algorithm are better than the results of the other method. Tables 3 and 4 prove that the proposed algorithm resists correlation attacks.

6.2.2 Peak signal to noise ratio analysis (PSNR)

PSNR gives the error between the plain image and the decrypted image. The PSNR is computed based on the mean squared error (*MSE*).

$$MSE = \frac{1}{M * N} \sum_{m=1}^{M} \sum_{n=1}^{N} \left| X(m,n) - X_d(m,n) \right|^2, \tag{88}$$

$$PSNR = 10 * \log_{10} \left[\frac{255^2}{MSE} \right], \tag{89}$$

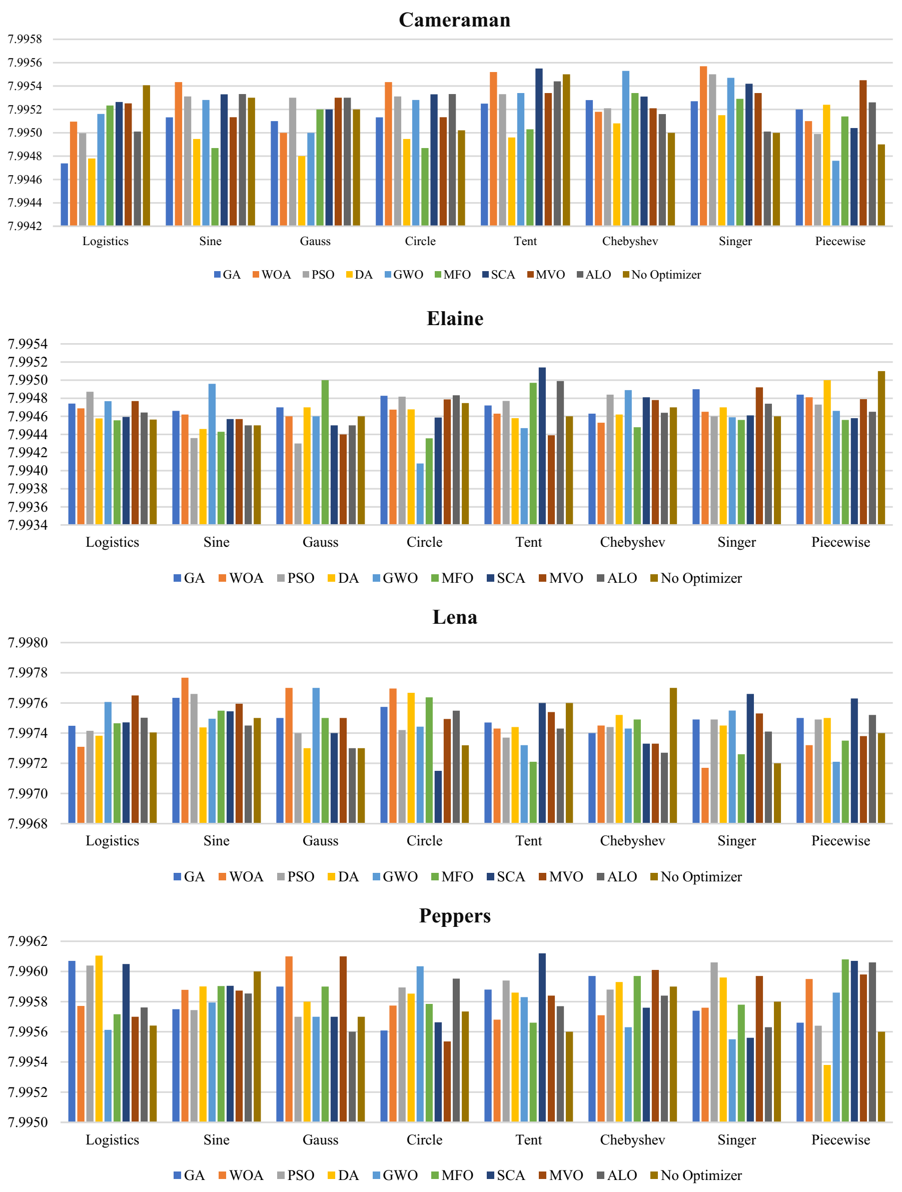


Fig. 6 Entropy of different images

where, $X(m, n)$ presents the original image, $X_d(m, n)$ presents the decrypted image, and M*N presents the size of both images. All of the PSNR values for the images are 99%, meaning that the original and decrypted images are the same, demonstrating that the proposed model is robust.

Table 2 Entropy Comparison with other recent algorithms

Image name	Proposed	Ref. [132]	Ref. [133]	Ref. [135]	Ref. [136]
Cameraman	7.99557	7.9862	–	7,9947	–
Elaine	7.99514	7.9895	–	–	7.981
Lena	7.99777	7.9948	7.9902	7.9970	7.924
Peppers	7.99614	7.9827	7.9885		–

6.2.3 Robust analysis

The encryption scheme's robustness indicates recovering valuable data even when noise disturbs the encrypted image or some data is lost during transmission. In the tests, some noise and different data loss amounts are added.

6.2.3.1 Noise attack analysis Salt and Pepper, Gaussian, Poisson, and Speckle noise have been added to the encrypted images of Cameraman, Elaine, Lenna, and peppers. Figures 11(a–d) show the encrypted images with the Salt and Pepper intensity of 0.05, the Gaussian noise variance of 0.01, the Poisson, and the Speckle noise, respectively. Figures 11(e–h) show the corresponding decrypted images. From Fig. 11, it can be seen that the decrypted images are clearly visible, which indicates that the algorithm is robust against noise attacks. Moreover, Fig. 12 shows a PSNR comparative analysis of the eight chaotic maps with and without optimization. It can be recognized from Figs. 11 and 12 that Gauss chaotic map with Dragonfly optimizer has the best visual recovery and most reasonable PSNR.

6.2.3.2 Periodic noise In order to test the algorithm's robustness, periodic noise has been added to images. The following steps have been taken to recover the image:

(1) Convert the image from the spatial domain to the frequency domain by computing the fast Fourier transform.
(2) Find the periodic noise spikes in the frequency image and block them.
(3) Convert the image back to the spatial domain using the inverse Fourier transform.

Figure 13 demonstrates that despite adding periodic noise, the essential details of the original image can still be retrieved. Figure 14 shows that the decrypted images' PSNRs are high, demonstrating that the proposed algorithm strongly resists periodic noise attacks.

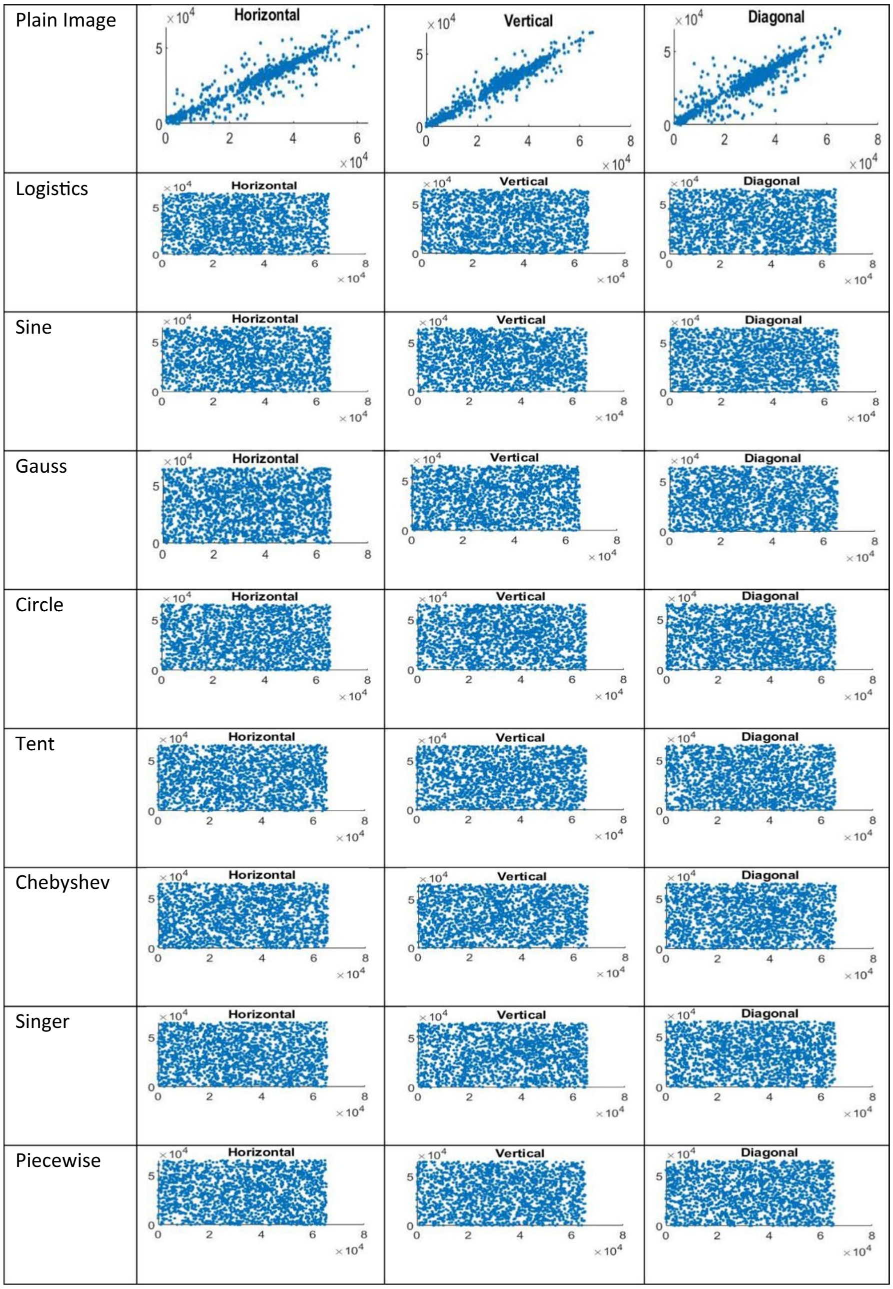


Fig. 7 Shows the correlation for cameraman plain image and the encrypted image

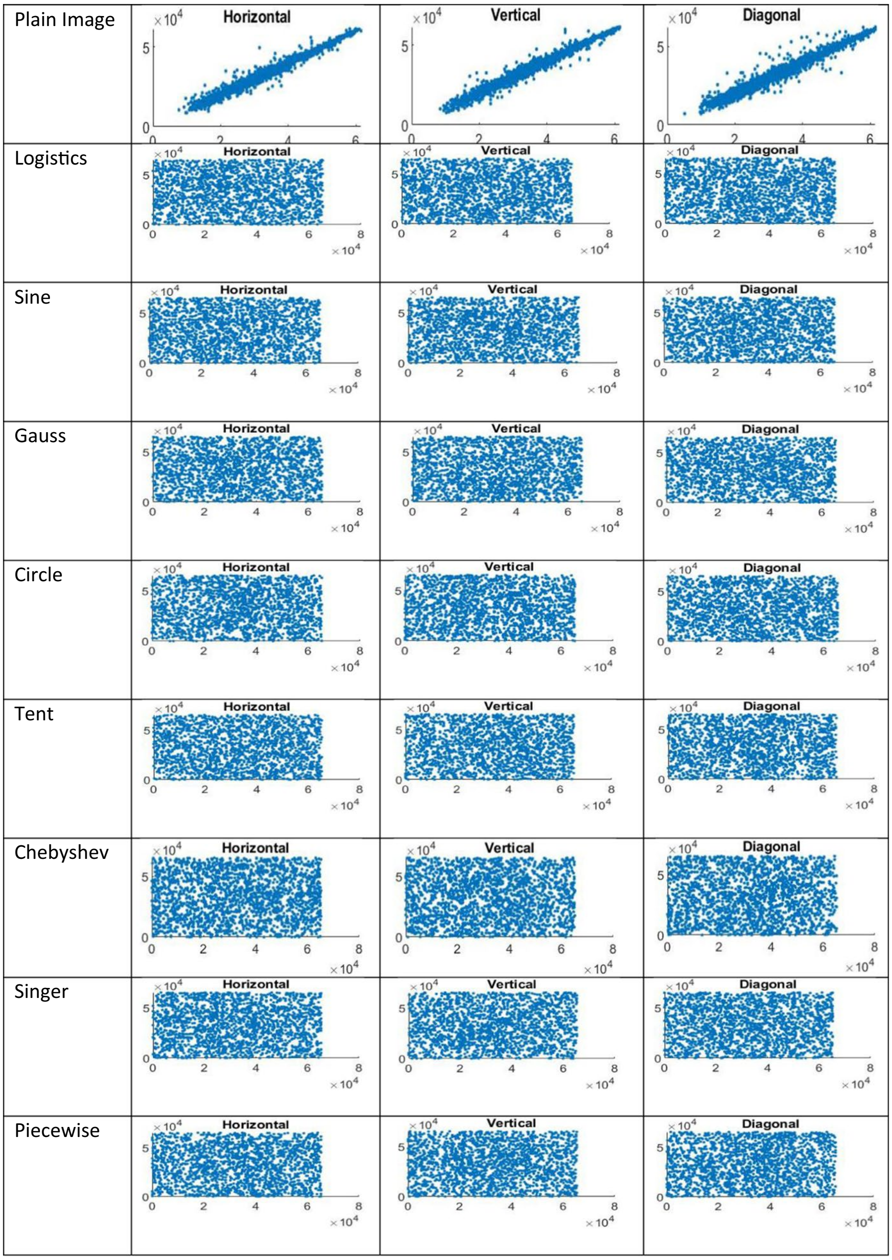


Fig. 8 Shows the Correlation for Elaine plain image and encrypted image

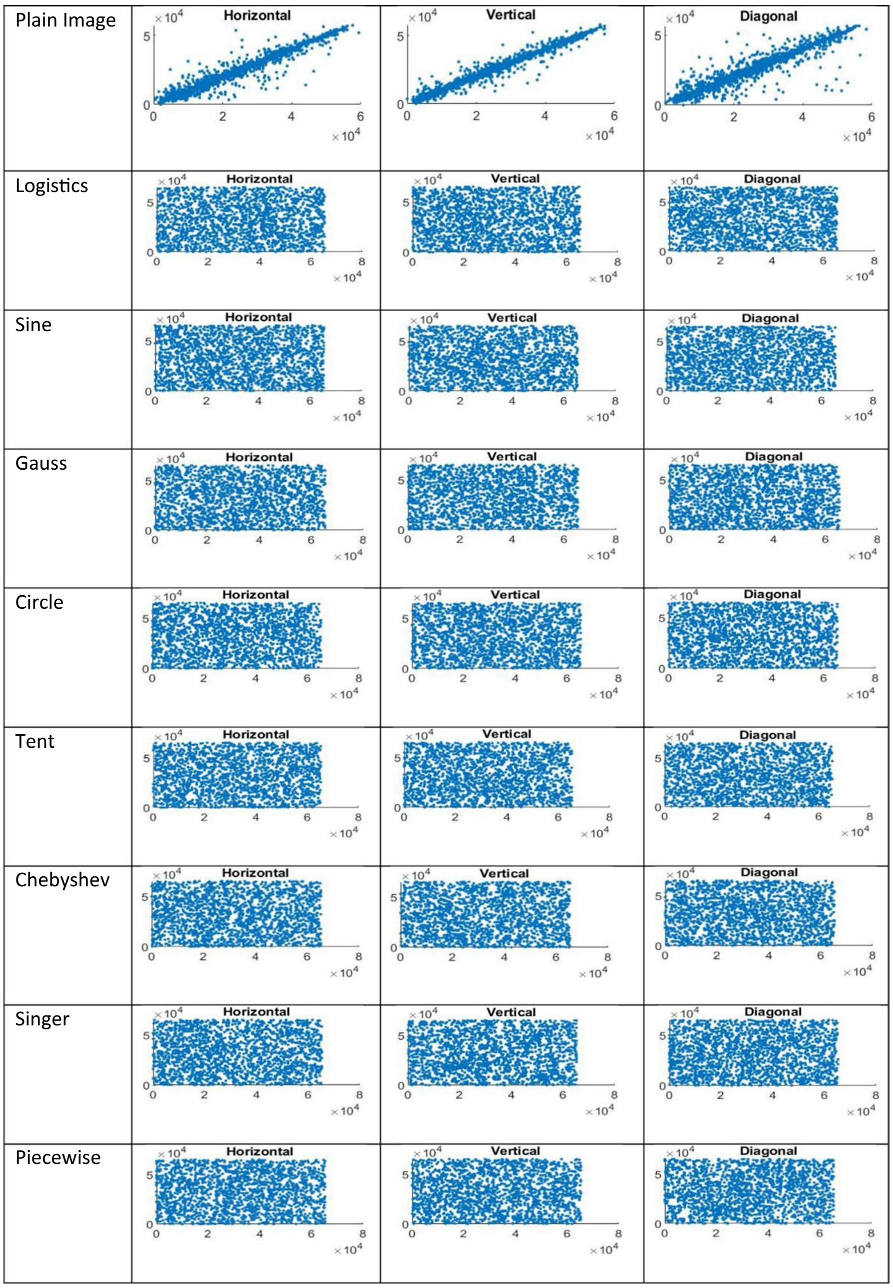


Fig. 9 Shows the Correlation for Lena plain image and the encrypted image

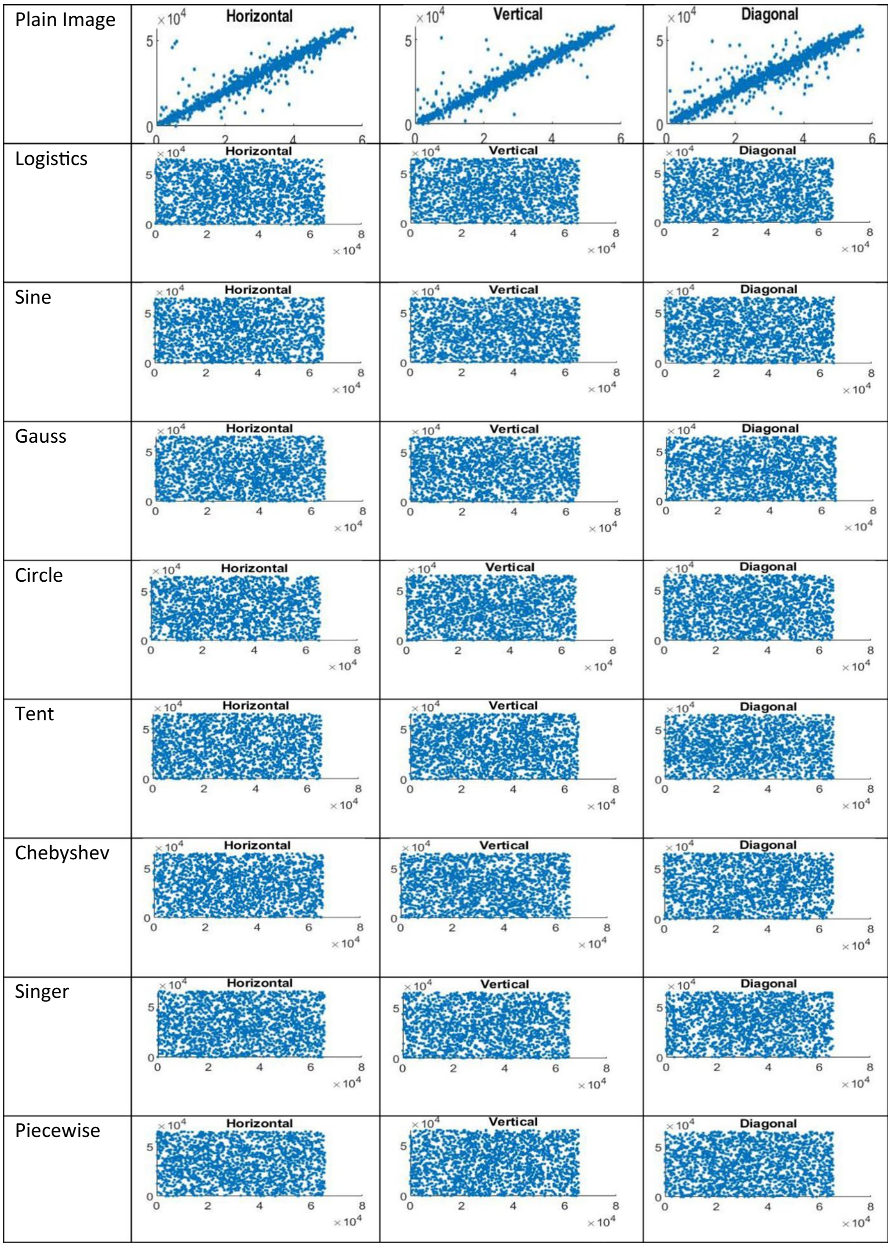


Fig. 10 Shows the Correlation for Peppers plain image and encrypted image

Table 3 Shows the correlation of the encryption algorithms using eight optimized chaotic Maps

		Cameraman			Elaine			Lenna			Peppers		
		Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
1-Logistics	GA	0.0024	−0.0024	0.0023	0.0026	0.0001	−0.0005	−0.0024	0.0079	0.0002	0.0013	0.0007	0.0004
	WOA	0.0030	−0.0022	0.0026	−0.0017	0.0016	0.0032	−0.0003	0.0084	−0.0003	−0.0049	−0.0001	0.0000
	PSO	0.0011	−0.0018	−0.0005	0.0005	0.0013	−0.0052	−0.0018	0.0071	0.0000	−0.0034	−0.0011	0.0013
	DA	0.0013	−0.0034	0.0009	0.0019	−0.0002	−0.0017	0.0013	0.0044	−0.0027	0.0008	−0.0034	0.0018
	GWO	0.0024	−0.0007	0.0007	−0.0034	−0.0033	0.0005	0.0038	0.0047	0.0018	−0.0020	0.0023	−0.0004
	MFO	−0.0003	0.0003	−0.0012	−0.0040	−0.0011	−0.0013	0.0005	0.0033	−0.0003	−0.0019	−0.0005	−0.0023
	SCA	0.0003	0.0009	0.0040	0.0018	0.0021	−0.0005	−0.0006	0.0074	−0.0005	−0.0001	−0.0005	0.0023
	MVO	−0.0022	−0.0016	−0.0009	0.0012	0.0015	−0.0018	−0.0021	0.0059	−0.0018	0.0012	0.0032	0.0017
	ALO	0.0017	−0.0012	−0.0016	0.0011	0.0002	0.0005	0.0018	0.0066	0.0021	0.0006	0.0054	0.0021
	No optimizer	0.0017	−0.0014	−0.0030	−0.0006	0.0006	−0.0003	0.0045	0.0036	−0.0018	−0.0022	0.0051	−0.0025
2-Sine	GA	−0.0045	−0.0021	0.0005	−0.0047	0.0003	−0.0015	−0.0016	0.0038	−0.0004	0.0021	−0.0016	−0.0011
	WOA	−0.0004	−0.0026	0.0018	−0.0008	-0.0041	−0.0017	0.0000	0.0045	0.0003	0.0040	0.0040	0.0004
	PSO	−0.0004	0.0002	−0.0020	0.0016	−0.0024	−0.0006	0.0006	0.0063	−0.0026	0.0001	0.0018	0.0026
	DA	−0.0003	−0.0011	− 0.0027	−0.0010	−0.0036	−0.0030	−0.0001	0.0074	−0.0028	−0.0041	0.0022	−0.0016
	GWO	−0.0002	−0.0029	0.0004	0.0005	−0.0024	0.0007	0.0014	0.0049	−0.0027	0.0005	0.0046	0.0022
	MFO	0.0006	−0.0015	0.0022	0.0004	−0.0007	0.0002	0.0014	0.0016	−0.0022	0.0004	0.0037	0.0022
	SCA	−0.0012	−0.0014	−0.0027	0.0022	0.0009	−0.0007	0.0015	0.0053	0.0024	−0.0068	0.0002	0.0014
	MVO	−0.0007	−0.0007	−0.0029	0.0006	0.0020	0.0027	−0.0003	0.0044	0.0000	−0.0006	0.0024	−0.0012
	ALO	−0.0008	0.0019	−0.0033	−0.0004	0.0009	−0.0014	−0.0010	0.0001	−0.0024	−0.0009	0.0017	0.0026
	No optimizer	−0.0014	−0.0013	−0.0013	−0.0009	0.0040	−0.0033	0.0017	0.0052	0.0011	0.0027	−0.0005	−0.0014

Table 3 (continued)

		Cameraman			Elaine			Lenna			Peppers		
		Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
3-Gauss	GA	−0.0007	−0.0035	−0.0008	−0.0018	0.0043	0.0026	−0.0022	0.0019	−0.0013	0.0015	−0.0004	0.0009
	WOA	0.0009	−0.0013	−0.0045	−0.0006	0.0041	−0.0001	0.0002	0.0031	−0.0013	0.0006	−0.0018	−0.0010
	PSO	0.0004	0.0003	−0.0013	0.0032	0.0025	0.0012	0.0009	0.0002	0.0016	0.0017	−0.0002	0.0028
	DA	−0.0038	0.0024	0.0003	−0.0005	0.0024	−0.0008	0.0018	0.0093	−0.0027	−0.0010	0.0040	−0.0008
	GWO	−0.0001	−0.0021	−0.0003	−0.0007	0.0002	0.0020	0.0031	0.0042	−0.0030	0.0006	0.0008	−0.0023
	MFO	0.0004	−0.0024	0.0027	0.0013	0.0008	0.0025	−0.0001	0.0053	−0.0008	−0.0003	0.0030	0.0010
	SCA	0.0003	0.0008	−0.0023	−0.0023	0.0042	0.0008	−0.0030	0.0060	−0.0020	−0.0005	−0.0023	0.0009
	MVO	−0.0026	0.0007	−0.0002	0.0018	0.0006	0.0027	0.0002	0.0055	0.0032	−0.0015	0.0000	−0.0020
	ALO	−0.0005	0.0022	0.0010	0.0027	0.0022	0.0019	0.0012	0.0042	−0.0019	−0.0022	−0.0006	0.0004
	No optimizer	−0.0029	−0.0018	−0.0015	0.0012	0.0021	0.0027	−0.0021	0.0051	−0.0012	−0.0001	0.0023	−0.0013
4-Circle	GA	0.0064	−0.0058	−0.0015	0.0018	−0.0014	−0.0018	−0.0007	0.0034	0.0007	0.0018	0.0001	−0.0011
	WOA	−0.0005	−0.0004	−0.0001	0.0003	0.0029	−0.0046	−0.0017	0.0056	-0.0029	0.0018	0.0033	0.0005
	PSO	0.0020	−0.0009	0.0014	−0.0017	0.0037	−0.0010	0.0008	0.0078	−0.0029	−0.0010	0.0006	0.0021
	DA	0.0010	−0.0039	−0.0006	−0.0003	0.0043	−0.0002	0.0021	0.0036	0.0004	0.0024	0.0007	0.0019
	GWO	−0.0039	−0.0045	−0.0001	0.0026	0.0060	−0.0017	0.0034	0.0032	0.0008	0.0002	0.0010	−0.0044
	MFO	−0.0016	−0.0007	−0.0021	0.0009	0.0004	0.0000	−0.0008	0.0055	−0.0009	0.0023	−0.0003	−0.0006
	SCA	0.0004	−0.0034	−0.0063	0.0000	0.0005	−0.0020	−0.0011	0.0037	0.0019	0.0019	0.0015	0.0003
	MVO	0.0014	−0.0024	−0.0019	0.0039	−0.0034	−0.0002	−0.0003	0.0052	−0.0009	0.0041	−0.0013	−0.0019
	ALO	0.0004	−0.0027	−0.0013	−0.0004	0.0007	−0.0018	−0.0004	0.0083	0.0026	0.0014	0.0034	0.0030
	No optimizer	−0.0016	0.0015	−0.0008	0.0021	0.0006	0.0030	−0.0015	0.0055	−0.0012	0.0010	0.0000	−0.0015

Table 3 (continued)

		Cameraman			Elaine			Lenna			Peppers		
		Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
5-Tent	GA	−0.0013	−0.0035	0.0007	0.0021	−0.0010	0.0012	0.0016	0.0040	0.0009	0.0014	0.0011	0.0001
	WOA	−0.0016	0.0006	0.0018	−0.0007	0.0021	0.0011	−0.0017	0.0020	−0.0019	−0.0010	0.0025	−0.0022
	PSO	0.0019	−0.0023	−0.0007	−0.0009	0.0015	−0.0014	−0.0021	0.0035	0.0032	0.0002	0.0018	−0.0018
	DA	0.0019	0.0022	0.0035	0.0031	0.0006	−0.0007	−0.0021	0.0068	0.0019	−0.0028	0.0016	−0.0002
	GWO	0.0025	−0.0019	0.0014	0.0013	0.0027	0.0017	0.0001	0.0063	0.0012	0.0022	−0.0013	0.0025
	MFO	0.0004	−0.0036	−0.0017	−0.0003	0.0003	−0.0003	0.0031	0.0054	0.0021	−0.0019	0.0036	0.0025
	SCA	0.0003	−0.0030	−0.0003	0.0039	−0.0009	−0.0006	0.0016	0.0058	−0.0004	−0.0005	−0.0015	−0.0011
	MVO	0.0009	−0.0021	0.0016	0.0039	0.0012	−0.0003	0.0006	0.0030	0.0006	−0.0006	−0.0018	0.0010
	ALO	−0.0036	−0.0015	−0.0044	0.0030	0.0010	0.0025	−0.0028	0.0031	0.0002	−0.0019	0.0010	−0.0034
	No optimizer	0.0065	0.0077	0.0012	0.0032	−0.0035	0.0024	0.0058	0.0189	0.0004	0.0045	0.0031	0.0030
6-Chebyshev	GA	−0.0003	−0.0004	0.0011	0.0028	0.0042	0.0005	0.0031	0.0055	0.0022	0.0010	0.0018	−0.0012
	WOA	−0.0003	−0.0022	0.0039	−0.0013	−0.0003	−0.0015	0.0039	0.0038	0.0016	−0.0008	0.0022	0.0033
	PSO	0.0025	−0.0046	0.0020	−0.0005	0.0013	0.0005	−0.0015	0.0056	−0.0027	0.0020	0.0006	−0.0014
	DA	−0.0031	0.0026	−0.0018	0.0047	0.0018	−0.0040	−0.0022	0.0051	0.0001	−0.0022	−0.0012	−0.0015
	GWO	0.0020	−0.0015	−0.0033	0.0014	0.0002	−0.0014	−0.0002	0.0055	−0.0018	0.0024	−0.0002	−0.0014
	MFO	0.0012	0.0015	−0.0019	0.0004	−0.0010	0.0005	0.0053	0.0039	−0.0014	0.0013	−0.0021	0.0021
	SCA	0.0007	−0.0003	0.0027	−0.0017	0.0037	−0.0015	0.0016	−0.0012	0.0018	0.0028	0.0008	0.0004
	MVO	0.0011	0.0002	−0.0005	0.0028	0.0030	0.0021	−0.0032	0.0053	−0.0024	0.0009	−0.0010	−0.0050
	ALO	0.0001	−0.0042	0.0012	−0.0019	−0.0007	−0.0022	−0.0011	0.0072	0.0019	−0.0024	−0.0002	−0.0007
	No optimizer	−0.0011	−0.0051	−0.0029	−0.0008	0.0011	0.0008	0.0017	0.0044	0.0042	−0.0004	0.0003	−0.0003

Table 3 (continued)

		Cameraman			Elaine			Lenna			Peppers		
		Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
7-Singer	GA	0.0065	0.0092	−0.0004	0.0033	0.0001	0.0026	0.0049	0.0178	0.0020	0.0044	0.0055	0.0048
	WOA	0.0051	0.0082	0.0016	−0.0018	−0.0027	0.0013	0.0018	0.0173	0.0025	0.0022	0.0027	−0.0001
	PSO	0.0002	−0.0026	−0.0016	0.0005	0.0015	−0.0011	−0.0020	0.0061	0.0006	−0.0003	0.0015	−0.0007
	DA	0.0045	−0.0030	0.0011	−0.0001	0.0007	−0.0025	0.0011	0.0028	−0.0009	0.0005	0.0002	0.0014
	GWO	0.0065	0.0077	0.0012	0.0032	−0.0035	0.0024	0.0058	0.0190	0.0004	0.0045	0.0031	0.0030
	MFO	−0.0048	−0.0037	0.0023	0.0013	−0.0008	0.0018	−0.0015	0.0030	0.0005	−0.0001	−0.0007	0.0007
	SCA	0.0073	0.0090	−0.0003	0.0050	−0.0015	−0.0008	0.0029	0.0181	0.0006	0.0020	0.0007	0.0012
	MVO	0.0043	0.0101	−0.0004	0.0024	−0.0018	0.0006	0.0050	0.0177	0.0008	0.0047	0.0016	0.0015
	ALO	0.0061	0.0065	0.0023	0.0023	−0.0017	0.0021	0.0037	0.0174	0.0030	0.0040	0.0012	0.0020
	No optimizer	0.0051	0.0082	0.0016	−0.0018	−0.0027	0.0013	0.0018	0.0173	0.0025	0.0022	0.0027	−0.0001
8-Piecewise	GA	−0.0007	−0.0013	−0.0017	0.0018	−0.0024	−0.0025	−0.0022	0.0039	0.0005	−0.0008	0.0029	−0.0003
	WOA	−0.0012	0.0004	0.0028	−0.0036	−0.0006	−0.0022	0.0021	0.0061	−0.0007	−0.0027	−0.0032	0.0010
	PSO	−0.0026	0.0001	0.0030	0.0037	−0.0013	−0.0003	−0.0007	0.0023	0.0025	0.0017	−0.0013	0.0021
	DA	0.0013	0.0021	−0.0021	0.0008	−0.0014	0.0016	0.0032	0.0053	0.0013	0.0006	0.0026	0.0039
	GWO	−0.0003	−0.0008	−0.0003	0.0025	−0.0009	−0.0021	−0.0001	0.0031	0.0011	−0.0022	0.0031	0.0013
	MFO	0.0025	−0.0001	0.0022	−0.0011	0.0028	−0.0021	−0.0003	0.0051	−0.0001	−0.0025	0.0020	−0.0020
	SCA	−0.0002	−0.0017	−0.0012	−0.0013	−0.0016	−0.0020	0.0004	0.0031	−0.0010	−0.0004	0.0016	−0.0002
	MVO	−0.0007	−0.0024	−0.0014	0.0005	0.0035	−0.0012	0.0025	0.0090	0.0003	0.0027	0.0035	−0.0002
	ALO	−0.0016	0.0013	−0.0012	−0.0022	−0.0012	0.0004	0.0004	0.0038	0.0042	−0.0025	0.0032	0.0008
	No optimizer	−0.0038	−0.0035	−0.0018	0.0017	0.0006	−0.0003	−0.0004	0.0070	−0.0007	0.0010	−0.0003	−0.0033

Table 4 Correlation comparison with others

Encryption algorithm	Cipher image		
	Horizontal	Vertical	Diagonal
Proposed algorithm (Cameraman)	−0.0039	−0.0045	−0.0001
Ref. [118]	−0.0031	−0.0006	0.0011
Ref. [2]	0.00002	−0.0048	−0.0029
Ref. [124]	0.0044	−0.0041	0.0016
Proposed algorithm (Elaine)	−0.0010	−0.0036	−0.0030
Ref. [121]	0.0005	−0.0133	0.0419
Ref. [2]	−0.0061	−0.00007	0.0007
Ref. [122]	0.0080	−0.0004	−0.0006
Ref. [123]	0.0014	0.0007	−0.0013
Ref. [125]	0.0019	0.0008	0.0010
Proposed algorithm (Lena)	−0.0032	−0.0012	−0.0024
Ref. [117]	0.0068	−0.0000	−0.0028
Ref. [118]	0.0040	−0.0012	−0.0021
Ref. [2]	0.0020	−0.0005	−0.0009
Ref. [122]	0.0026	0.0006	0.0005
Ref. [125]	0.0007	−0.0004	0.0011
Ref. [127]	0.0002	0.0022	−0.001
Ref. [128]	−0.0007	0.00001	−0.0010
Ref. [130]	0.0008	0.0004	0.0020
Proposed algorithm (Peppers)	−0.0022	−0.0012	−0.0015
Ref. [119]	0.0046	0.0089	0.0128
Ref. [120]	−0.0121	−0.0079	0.0051
Ref. [121]	0.0295	0.0187	0.0393
Ref. [2]	0.004	0.0009	−0.0030
Ref. [125]	0.0002	−0.0018	0.0014
Ref. [127]	−0.0036	0.0023	0.0022
Ref. [129]	−0.0020	0.00044	0.0040
Ref. [130]	0.0046	0.0039	−0.0018

6.2.3.3 Cropping attack Different degrees of cropping attack tests have been applied to the encrypted images. The cipher images are cropped by 2.25, 12.5, 25, and 50%, respectively, and then decrypted. Figure 16 shows that the decrypted image obtained from the non-optimized Singer chaotic map reveals a partial loss of information, highlighting the importance of chaotic map optimization in achieving a robust image encryption algorithm, where optimization plays a crucial role in preserving the integrity of the encrypted data during decryption. Moreover, Fig. 17 shows the PSNR

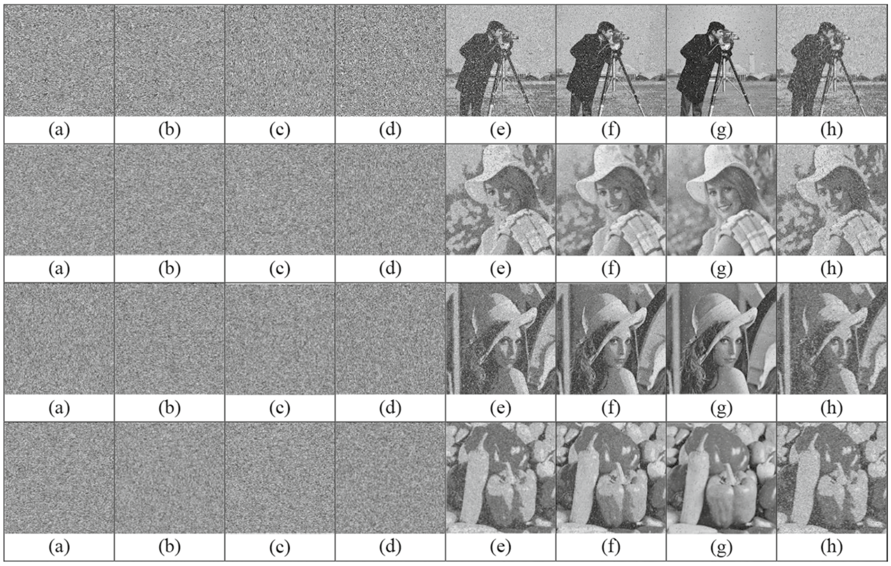


Fig. 11 Results of Cipher image with **a** Salt andPepper noise of intensity 0.05. **b** Gaussian noise of mean 0 and variance 0.01. **c** Poisson noise. **d** Speckle noise; Decryption image with **e** Salt and Pepper of intensity 0.05; Decryption image with **f** Gaussian noise of mean 0 and variance 0.01. **g** Poisson noise. **h** Speckle noise

between the original and recovered images Even though the Singer Chaotic map has the highest PSNR, cropped cipher images can only be reconstructed with optimization, and part of the data is lost, as shown in Figs. 15, 16, and 17. The Gauss chaotic map gives a good PSNR, and the primary information of the plain images can be recovered. It can be concluded that the Gauss Chaotic map has the best numerical and visual results compared to the other chaotic maps, and the Dragonfly algorithm is the best optimizer in terms of robust analysis.

6.2.3.4 Cropping attack with noise In this test, a 25% cropping attack with noise has been added to cipher images. It can be shown from Fig. 18 that the decrypted images using the optimized chaotic map successfully reconstructed the original content (Fig. 19), whereas Fig. 20, representing the non-optimized chaotic map, failed to reconstruct the decrypted images. This emphasizes the significance of optimization in achieving robust image encryption. Figure 20 shows the PSNR between the original image and the reconstructed images that match it. Based on these results, it can be declared that the Gauss Chaotic Map encryption algorithm is robust against noise attacks.

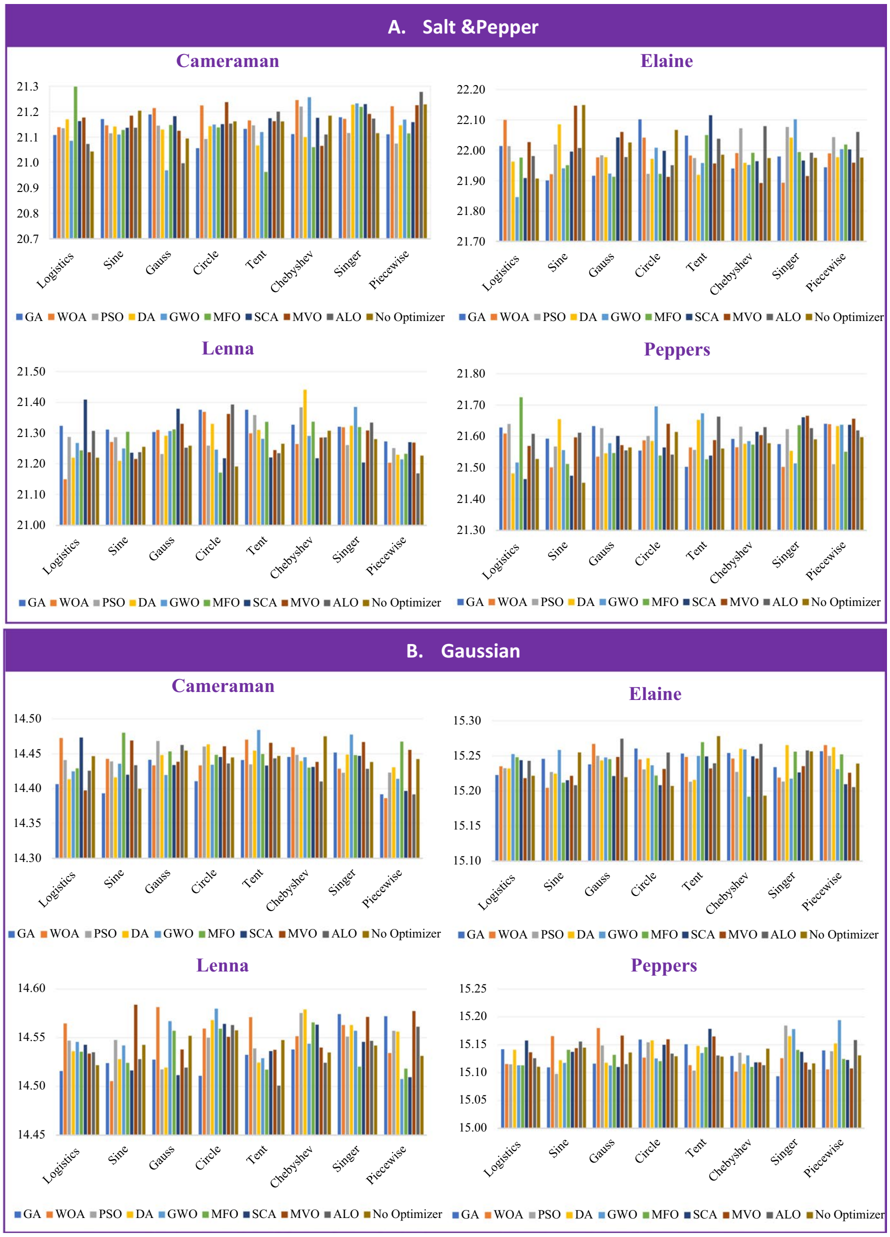


Fig. 12 PSNR of the recovered images comparative analysis- Different Chaotic maps with different optimizers and with no optimizer

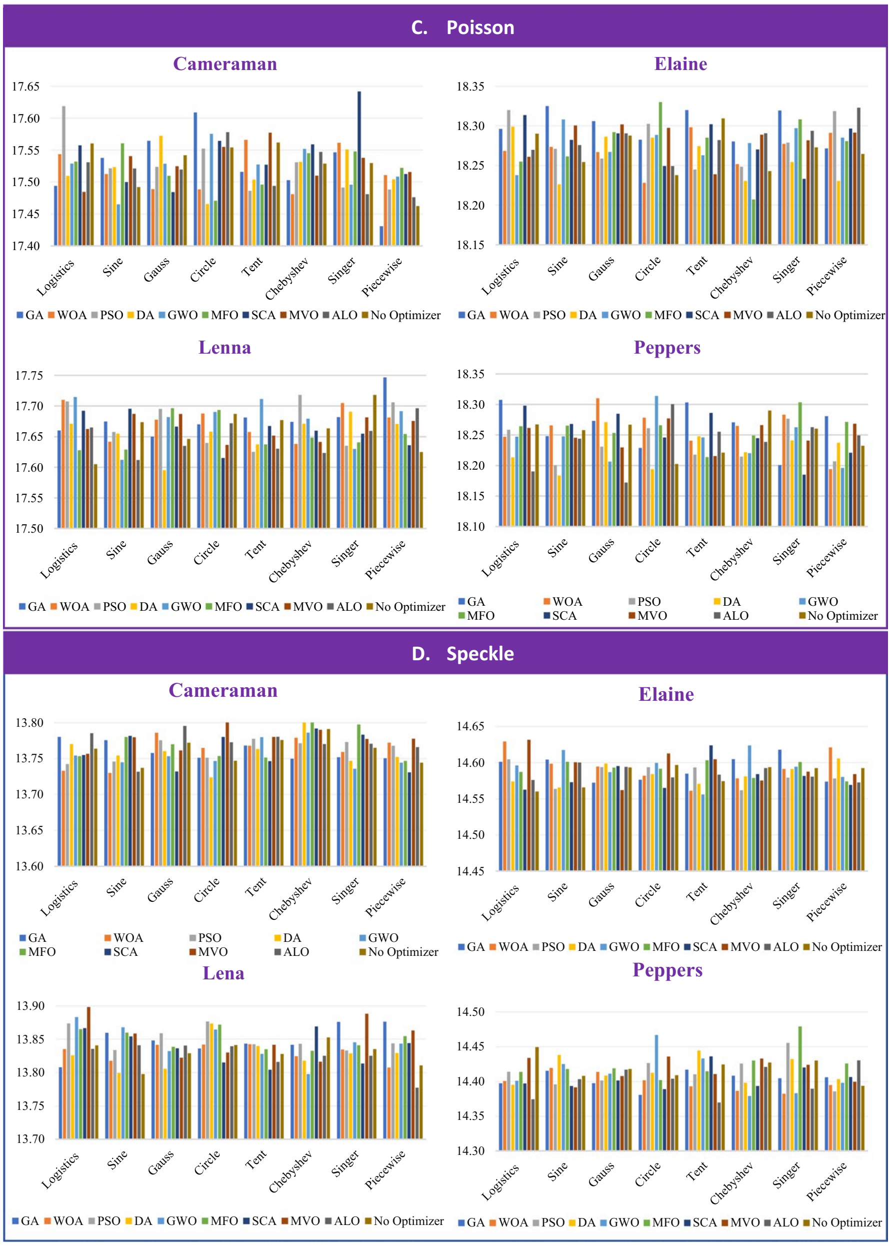


Fig. 12 (continued)

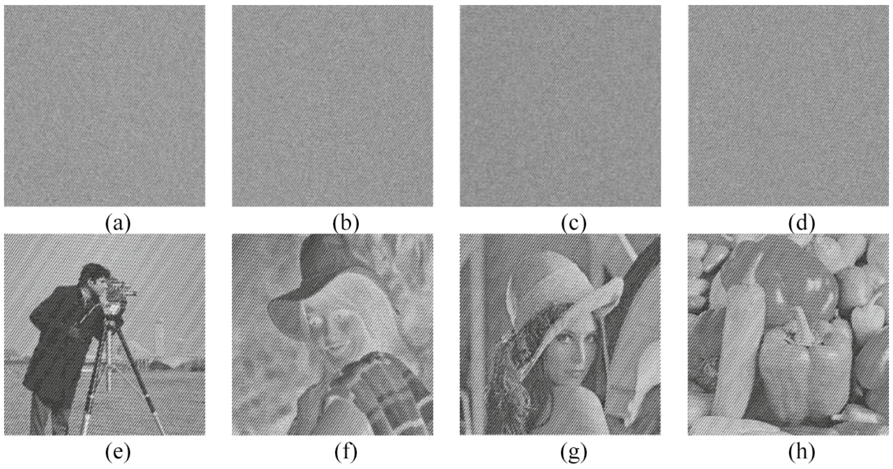


Fig. 13 Results of Periodic noise attack on images "cameraman," "Elaine," "Lenna," and "Peppers": (**a**–**d**) Ciphertext images. (**e**–**h**) Decrypted images

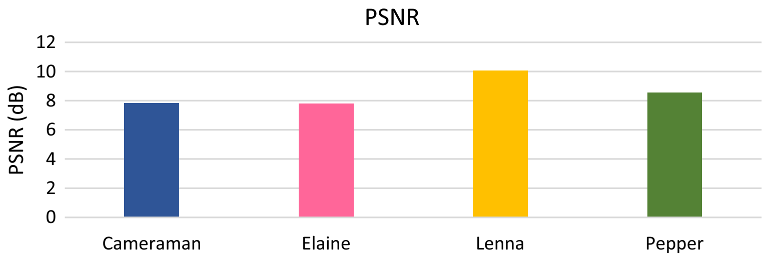


Fig. 14 PSNRs of the Periodic noise decrypted images and the corresponding plain image

6.2.4 Key analysis

6.2.4.1 Key space analysis The analysis of the key space refers to the collection of all possible encryption keys. A robust encryption algorithm must be sensitive to secret keys and have enough space to withstand any brute-force attack. The proposed algorithm uses the chaotic map parameters, initial conditions, and the parameters of the diffusion equation as keys. The key space of the proposed algorithm for the different types of chaotic maps with a computation precision of (10^{-16}) is shown in Table 5, which demonstrates that the key space of the proposed encryption algorithm is sufficiently large to resist brute-force attacks, with the Circle Chaotic map having the most robust key space. Table 6 compares the key spaces of different encryption techniques. The proposed algorithm has a large key space and is resistant to brute-force attacks, as seen in Table 6.

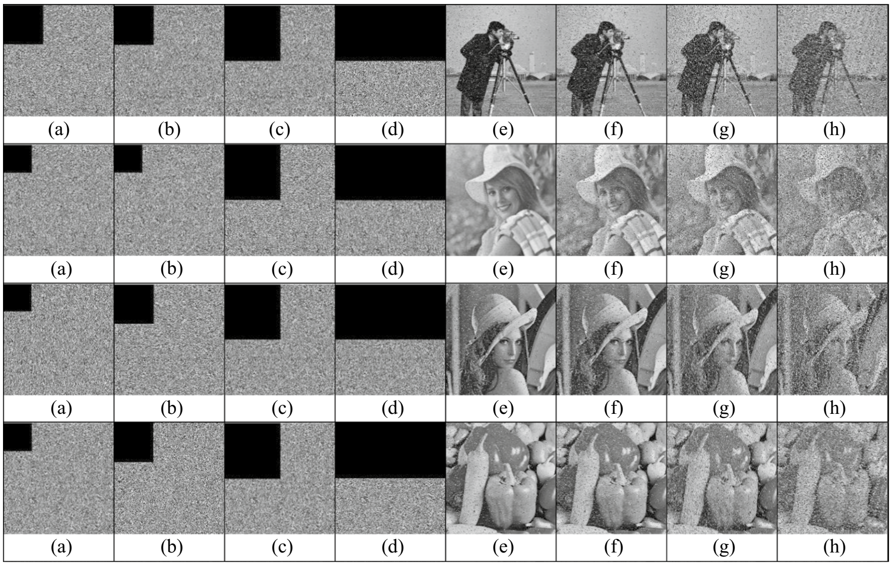


Fig. 15 Results of cropping attacks on cipher image using Gauss chaotic map optimized by dragonfly with: **a** 6.25% cropped. **b** 12.5% cropped. **c** 25% cropped. **d** 50% cropped; Decryption image with **e** 6.25% cropped. **f** 12.5% cropped. **g** 25% cropped. **h** 50% cropped

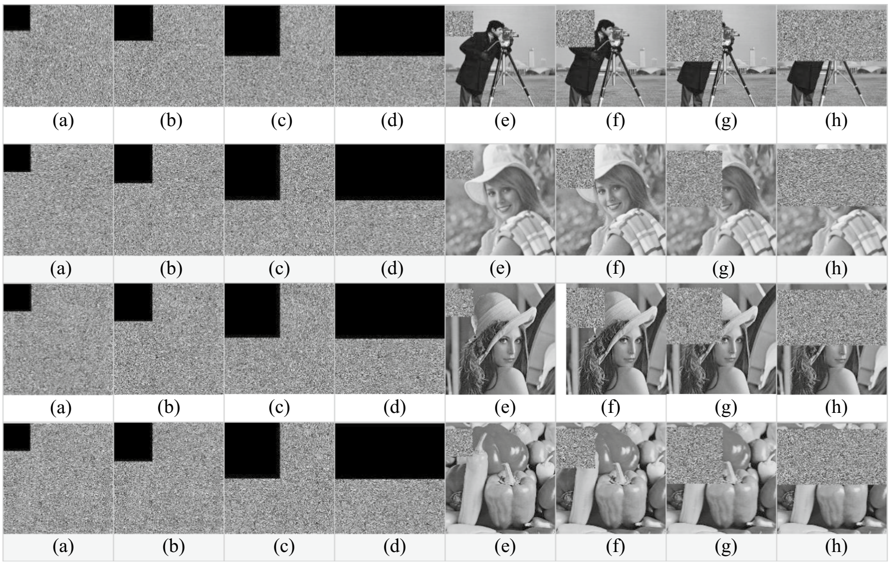


Fig. 16 Results of cropping attacks on cipher image using Singer chaotic map without optimization with: **a** 6.25% cropped. **b** 12.5% cropped. **c** 25% cropped. **d** 50% cropped; Decryption image with **e** 6.25% cropped. **f** 12.5% cropped. **g** 25% cropped. **h** 50% cropped

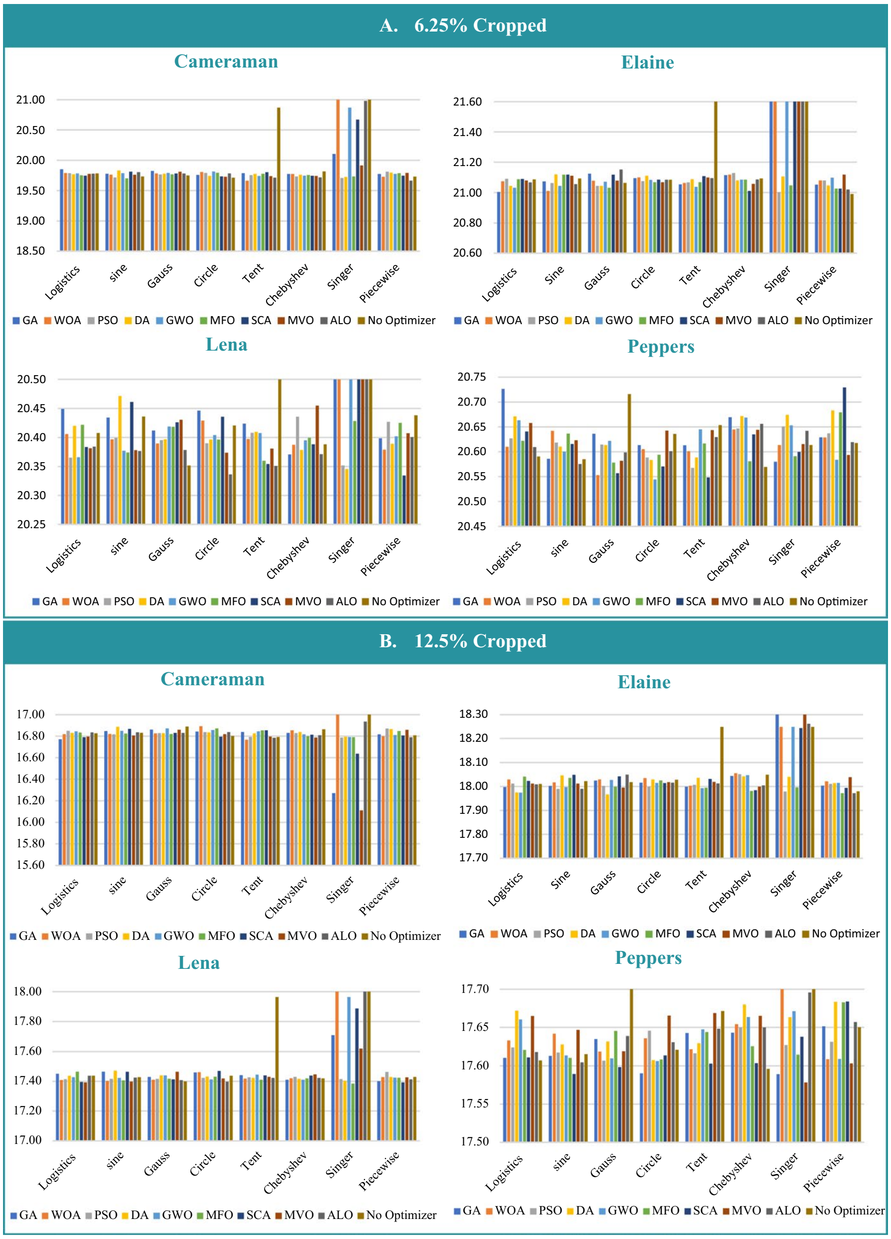


Fig. 17 Comparative analysis of PSNRs of the decrypted images after applying different percentages of cropping attack to the encrypted -Different Chaotic maps with different optimizers and with no optimizer

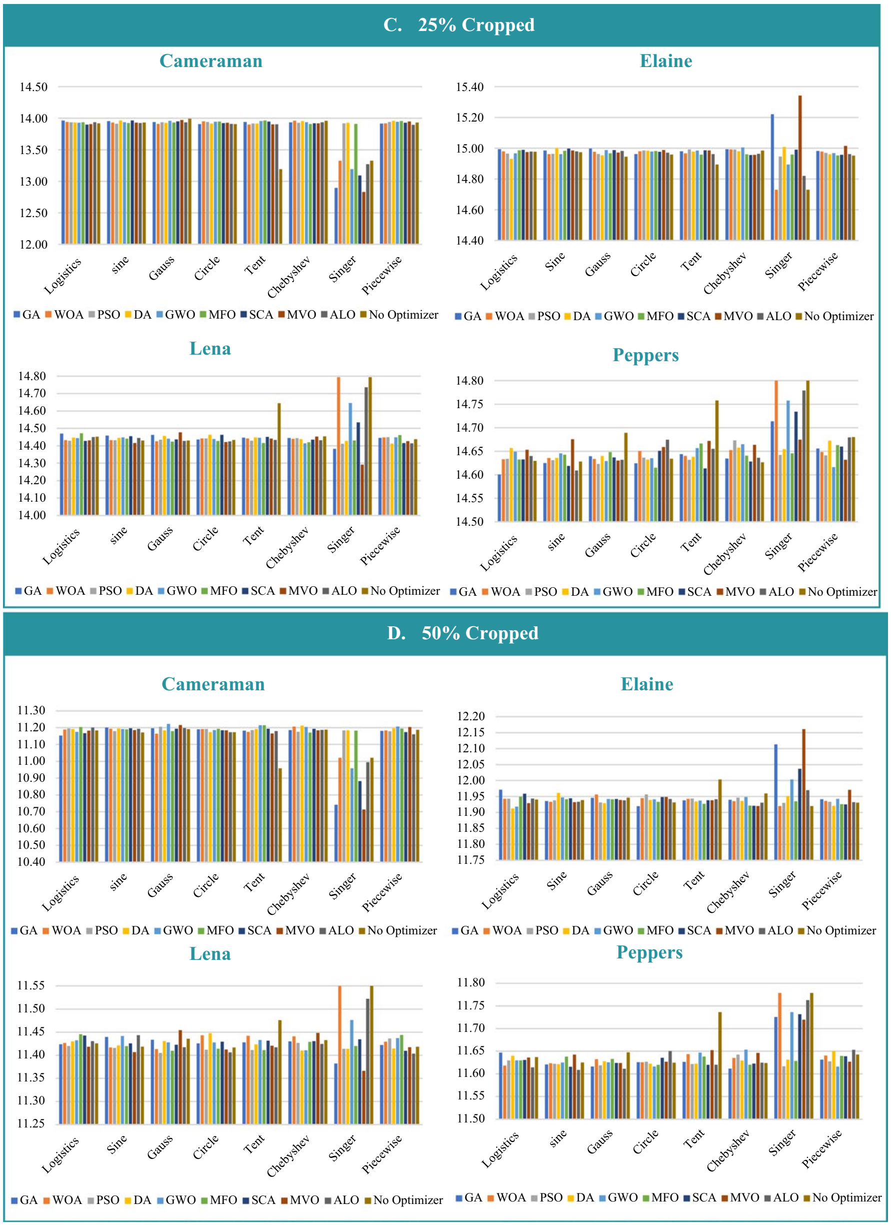


Fig. 17 (continued)

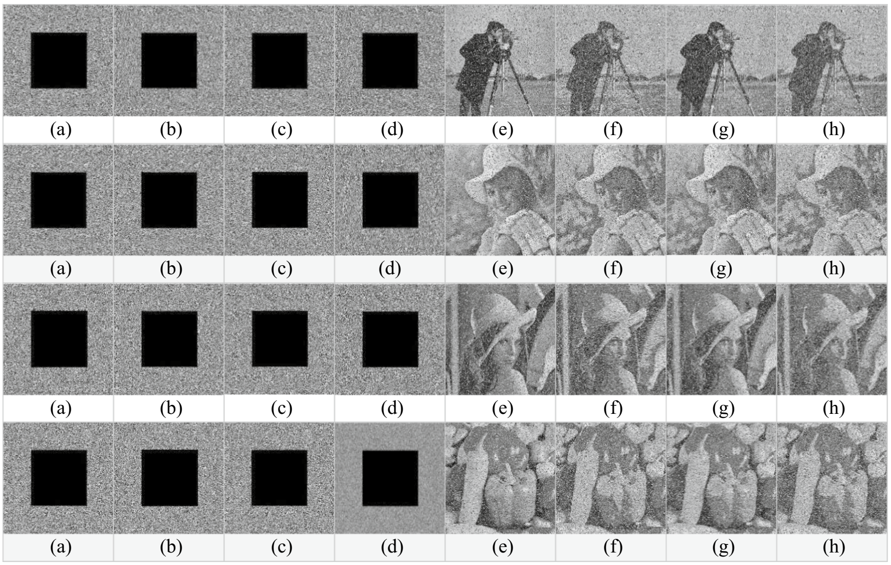

(a) (b) (c) (d) (e) (f) (g) (h)
(a) (b) (c) (d) (e) (f) (g) (h)
(a) (b) (c) (d) (e) (f) (g) (h)
(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 18 Results of 25% cropping attacks on cipher image using Gauss chaotic map optimized by dragonfly with **a** Salt andPepper noise of intensity 0.05. **b** Gaussian noise of mean 0 and variance 0.01. **c** Poisson noise. **d** Speckle noise; Decryption image with **e** Salt and Pepper of intensity 0.05; Decryption image with **f** Gaussian noise of mean 0 and variance 0.01. **g** Poisson noise. **h** Speckle noise

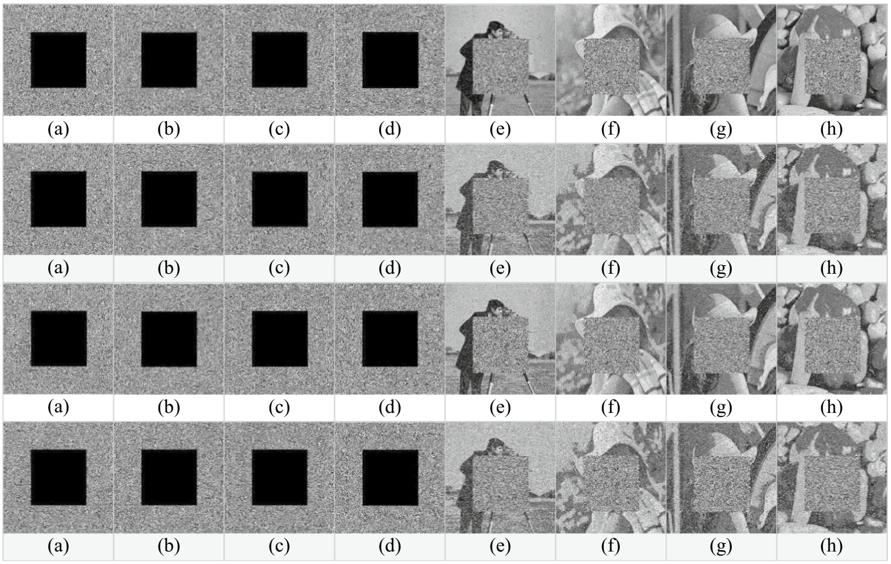

(a) (b) (c) (d) (e) (f) (g) (h)
(a) (b) (c) (d) (e) (f) (g) (h)
(a) (b) (c) (d) (e) (f) (g) (h)
(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 19 Results of 25% cropping attacks on cipher image using Singer chaotic map without optimization with **a** Salt andPepper noise of intensity 0.05. **b** Gaussian noise of mean 0 and variance 0.01. **c** Poisson noise. **d** Speckle noise; Decryption image with **e** Salt and Pepper of intensity 0.05; Decryption image with **f** Gaussian noise of mean 0 and variance 0.01. **g** Poisson noise. **h** Speckle noise

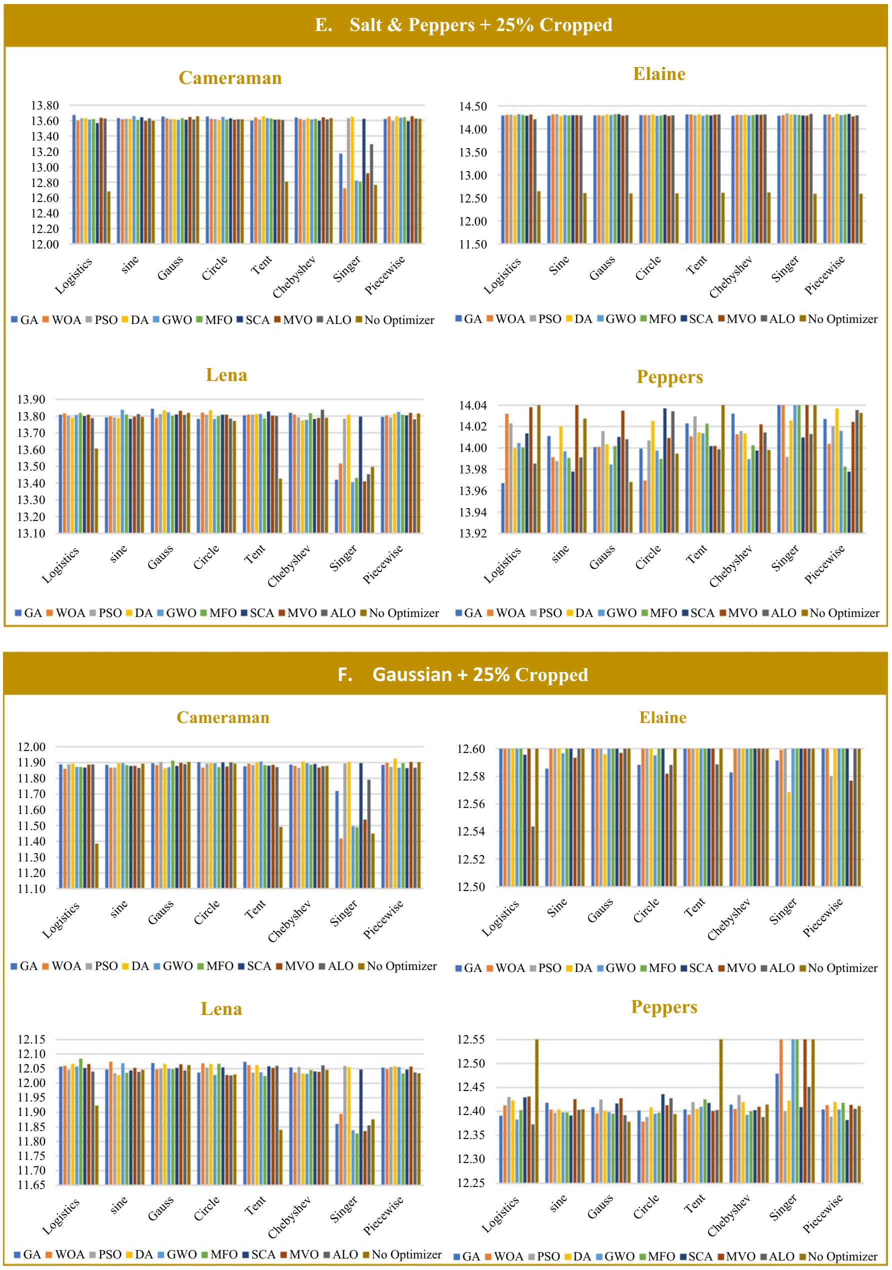
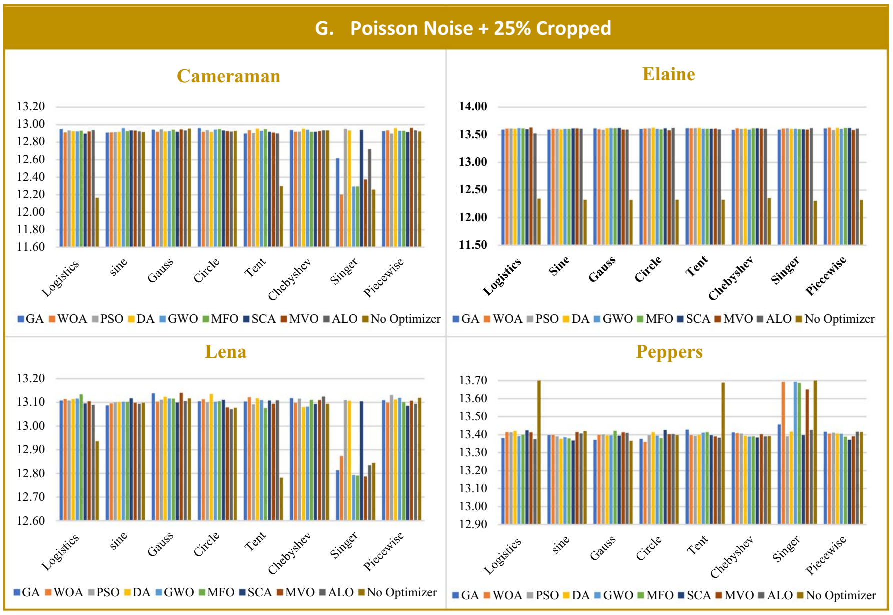


Fig. 20 Comparative analysis of PSNRs of the decrypted images after applying noise and cropping attack to the encrypted -Different Chaotic maps with different optimizers and with no optimizer



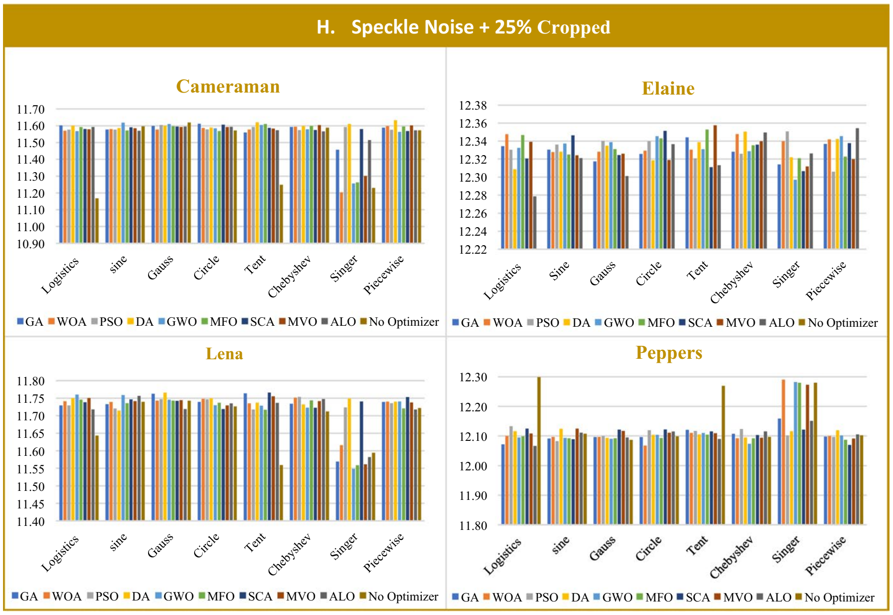


Fig. 20 (continued)

Table 5 Key Space of each Chaotic Map

Chaotic map	Parameters and initial conditions	Key space
Logistics map	2 + 2	$((10^{16})^4)$
Circle map	3 + 2	$((10^{16})^5)$
Gauss map	1 + 2	$((10^{16})^3)$
Sine map	2 + 2	$((10^{16})^4)$
Tent map	2 + 2	$((10^{16})^4)$
Chebyshev map	2 + 2	$((10^{16})^4)$
Singer map	2 + 2	$((10^{16})^4)$
Piecewise map	2 + 2	$((10^{16})^4)$

Table 6 Comparisons of the image encryption algorithms key space

Algorithm	Key space
Proposed algorithm	(10^{80})
Ref. [119]	(10^{56})
Ref. [128]	(10^{45})
Ref. [129]	(10^{36})
Ref. [130]	(10^{64})
Ref. [131]	(10^{42})
Ref. [34]	(10^{40})

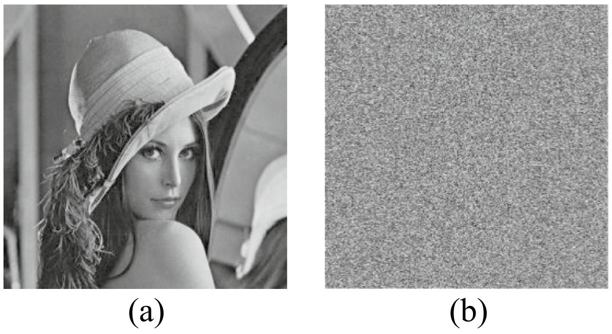


Fig. 21 Decrypting Lena image using the **a** correct key and **b** wrong key

6.2.4.2 Key sensitivity analysis Key sensitivity examines the sensitivity of a cipher image to a very slight change in the secret key. Figure 21a indicates decryption with the correct key x = 0.777; the encrypted image is identical to the original image; then the

Table 7 UACI and NPCR comparison

Chaotic maps	Optimizers	Cameraman		Elaine		Lenna		Peppers	
		UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR
1-Logistics	GA	0.3254	0.9881	0.3216	0.9936	0.3266	0.9943	0.3242	0.9941
	WO	0.3253	0.9896	0.3220	0.9939	0.3270	0.9942	0.3243	0.9939
	PSO	0.3248	0.9895	0.3216	0.9942	0.3266	0.9940	0.3246	0.9939
	DA	0.3252	0.9896	0.3215	0.9938	0.3272	0.9941	0.3246	0.9943
	GWO	0.3252	0.9897	0.3217	0.9938	0.3269	0.9939	0.3243	0.9942
	MFO	0.3253	0.9895	0.3219	0.9938	0.3267	0.9940	0.3243	0.9941
	SCA	0.3257	0.9892	0.3221	0.9940	0.3270	0.9941	0.3245	0.9943
	MVO	0.3253	0.9895	0.3215	0.9941	0.3271	0.9942	0.3247	0.9940
	ALO	0.3249	0.9898	0.3219	0.9941	0.3270	0.9942	0.3243	0.9942
	No optimizer	0.2921	0.9896	0.3020	0.9941	0.2970	0.9942	0.3123	0.9943
2-Sine	GA	0.3252	0.9898	0.3221	0.9940	0.3274	0.9943	0.3245	0.9940
	WO	0.3251	0.9894	0.3216	0.9939	0.3267	0.9941	0.3242	0.9945
	PSO	0.3253	0.9895	0.3214	0.9940	0.3264	0.9942	0.3244	0.9940
	DA	0.3249	0.9894	0.3217	0.9937	0.3271	0.9942	0.3242	0.9942
	GWO	0.3257	0.9897	0.3216	0.9937	0.3268	0.9941	0.3245	0.9942
	MFO	0.3253	0.9892	0.3215	0.9938	0.3269	0.9942	0.3246	0.9942
	SCA	0.3252	0.9895	0.3220	0.9939	0.3270	0.9941	0.3244	0.9941
	MVO	0.3250	0.9895	0.3218	0.9938	0.3269	0.9941	0.3245	0.9941
	ALO	0.3258	0.9895	0.3219	0.9938	0.3269	0.9942	0.3240	0.9942
	No optimizer	0.2925	0.9895	0.3018	0.9937	0.2976	0.9943	0.3139	0.9942
3-Gauss	GA	0.3252	0.9887	0.3219	0.9935	0.3270	0.9942	0.3245	0.9942
	WO	0.3252	0.9892	0.3217	0.9941	0.3271	0.9939	0.3240	0.9940
	PSO	0.3252	0.9889	0.3218	0.9938	0.3271	0.9938	0.3245	0.9939
	DA	0.3249	0.9891	0.3218	0.9937	0.3266	0.9942	0.3244	0.9943
	GWO	0.3247	0.9889	0.3216	0.9938	0.3268	0.9941	0.3241	0.9941
	MFO	0.3251	0.9890	0.3216	0.9936	0.3266	0.9937	0.3248	0.9941
	SCA	0.3253	0.9892	0.3217	0.9939	0.3267	0.9939	0.3247	0.9943
	MVO	0.3250	0.9891	0.3219	0.9937	0.3271	0.9940	0.3243	0.9940
	ALO	0.3255	0.9889	0.3217	0.9935	0.3270	0.9940	0.3246	0.9940
	No optimizer	0.2899	0.9890	0.3007	0.9938	0.2966	0.9939	0.3130	0.9939
4- Circle	GA	0.3249	0.9912	0.3216	0.9939	0.3270	0.9945	0.3249	0.9948
	WO	0.3252	0.9868	0.3212	0.9938	0.3273	0.9932	0.3247	0.9938
	PSO	0.3252	0.9884	0.3211	0.9938	0.3265	0.9943	0.3246	0.9940
	DA	0.3251	0.9893	0.3214	0.9942	0.3266	0.9943	0.3246	0.9943
	GWO	0.3254	0.9900	0.3217	0.9940	0.3275	0.9944	0.3249	0.9945
	MFO	0.3252	0.9848	0.3215	0.9927	0.3271	0.9938	0.3245	0.9929
	SCA	0.3252	0.9886	0.3221	0.9939	0.3269	0.9941	0.3242	0.9941
	MVO	0.3252	0.9879	0.3220	0.9932	0.3270	0.9941	0.3240	0.9937
	ALO	0.3245	0.9901	0.3214	0.9939	0.3274	0.9944	0.3251	0.9946
	No optimizer	0.2906	0.9894	0.3021	0.9941	0.3006	0.9947	0.3154	0.9946

Table 7 (continued)

Chaotic maps	Optimizers	Cameraman		Elaine		Lenna		Peppers	
		UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR
5-Tent	GA	0.3268	0.9960	0.3235	0.9960	0.3281	0.9960	0.3261	0.9960
	WO	0.3268	0.9960	0.3238	0.9961	0.3283	0.9961	0.3264	0.9960
	PSO	0.3260	0.9960	0.3237	0.9960	0.3278	0.9963	0.3265	0.9961
	DA	0.3272	0.9960	0.3238	0.9960	0.3290	0.9961	0.3261	0.9961
	GWO	0.3277	0.9962	0.3229	0.9960	0.3287	0.9961	0.3266	0.9963
	MFO	0.3270	0.9961	0.3241	0.9960	0.3282	0.9961	0.3259	0.9960
	SCA	0.3279	0.9960	0.3238	0.9960	0.3282	0.9960	0.3261	0.9960
	MVO	0.3267	0.9959	0.3233	0.9961	0.3281	0.9959	0.3262	0.9961
	ALO	0.3280	0.9962	0.3233	0.9960	0.3280	0.9962	0.3264	0.9961
	No optimizer	0.1582	0.9332	0.2249	0.9852	0.1751	0.9739	0.2036	0.9769
6-Chebysev	GA	0.3277	0.9961	0.3234	0.9962	0.3285	0.9960	0.3263	0.9960
	WO	0.3267	0.9961	0.3227	0.9962	0.3288	0.9959	0.3269	0.9962
	PSO	0.3281	0.9961	0.3236	0.9961	0.3282	0.9960	0.3255	0.9960
	DA	0.3265	0.9958	0.3230	0.9962	0.3285	0.9961	0.3269	0.9961
	GWO	0.3276	0.9963	0.3241	0.9960	0.3288	0.9961	0.3269	0.9963
	MFO	0.3277	0.9963	0.3228	0.9960	0.3288	0.9963	0.3265	0.9960
	SCA	0.3270	0.9960	0.3242	0.9959	0.3272	0.9961	0.3260	0.9961
	MVO	0.3265	0.9960	0.3235	0.9960	0.3289	0.9961	0.3274	0.9962
	ALO	0.3271	0.9962	0.3230	0.9962	0.3282	0.9963	0.3262	0.9963
	No optimizer	0.2923	0.9895	0.3024	0.9937	0.2978	0.9944	0.3144	0.9943
7-Singer	GA	0.3272	0.9960	0.3232	0.9959	0.3288	0.9961	0.3268	0.9962
	WO	0.3265	0.9960	0.3235	0.9960	0.3282	0.9961	0.3265	0.9961
	PSO	0.3268	0.9961	0.3240	0.9960	0.3290	0.9961	0.3269	0.9962
	DA	0.3273	0.9958	0.3240	0.9960	0.3282	0.9960	0.3265	0.9960
	GWO	0.3273	0.9961	0.3234	0.9962	0.3285	0.9960	0.3267	0.9962
	MFO	0.3266	0.9959	0.3246	0.9961	0.3284	0.9962	0.3264	0.9960
	SCA	0.3276	0.9961	0.3232	0.9961	0.3289	0.9961	0.3263	0.9959
	MVO	0.3273	0.9962	0.3236	0.9962	0.3287	0.9961	0.3262	0.9961
	ALO	0.3272	0.9959	0.3240	0.9960	0.3290	0.9959	0.3268	0.9962
	No optimizer	0.1529	0.9329	0.2168	0.9833	0.1672	0.9724	0.1951	0.9743
8-Piecewise	GA	0.3267	0.9960	0.3237	0.9960	0.3288	0.9961	0.3268	0.9961
	WO	0.3267	0.9961	0.3234	0.9959	0.3286	0.9961	0.3261	0.9960
	PSO	0.3270	0.9961	0.3239	0.9962	0.3284	0.9960	0.3267	0.9961
	DA	0.3277	0.9962	0.3234	0.9962	0.3284	0.9961	0.3267	0.9961
	GWO	0.3262	0.9963	0.3233	0.9962	0.3278	0.9960	0.3263	0.9961
	MFO	0.3273	0.9959	0.3232	0.9960	0.3284	0.9959	0.3265	0.9960
	SCA	0.3267	0.9961	0.3234	0.9959	0.3292	0.9960	0.3272	0.9960
	MVO	0.3274	0.9960	0.3237	0.9960	0.3276	0.9960	0.3266	0.9959
	ALO	0.3266	0.9959	0.3233	0.9960	0.3288	0.9962	0.3265	0.9961
	No optimizer	0.2894	0.9887	0.3000	0.9936	0.2953	0.9939	0.3122	0.9944

Table 8 Comparisons of the image NPCR of encrypted image

Image name	Proposed	Ref. [118]	Ref. [2]	Ref. [125]	Ref. [134]
Cameraman	0.99632	0.99630	–	–	0.99630
Elaine	0.99624	–	0.99650	0.99618	–
Lena	0.99630	0.99620	0.9960	0.99628	0.99620
Peppers	0.99628	–	0.99610	0.99620	–

Table 9 Comparisons of the image UACI of encrypted image

Image name	Proposed	Ref. [130]	Ref. [131]	Ref. [133]	Ref. [134]	Ref. [135]
Cameraman	32.8082	–	–	–	31.2802	25.492
Elaine	32.4573	–	–	–	–	–
Lena	32.9224	–	34.7772	28.7491	30.7972	–
Peppers	32.7370	32.6708	34.4092	30.7957	–	–

ciphered image is decrypted using one bit changed key x = 0.7770000000000000001. as shown in Fig. 21b, shows that the cipher image could not be decrypted then the ciphered image is decrypted using a one-bit changed key x = 0.7770000000000000001. As shown in Fig. 21b, the cipher image could not be decrypted. Consequently, the proposed algorithm is highly sensitive to secret keys.

6.2.5 Differential analysis

6.2.5.1 Number of pixel change rate It is the proportion of changed pixel numbers between two encrypted images with a one-pixel change in the original image. The following are the formulas used to compute the NPCR:

$$NPCR = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{D(i,j)}{M * N} * 100\% \tag{90}$$

$$D(\mathrm{i},j) = \begin{cases} 0, \textit{otherwise} \\ 1, \textit{if } Y_1(i,j) = Y_2(i,j) \end{cases} \tag{91}$$

where $Y_1(i,j) \& Y_2(i,j)$ denote two encrypted images corresponding to the same original image differing only in one bit, and M*N is the size of the images.

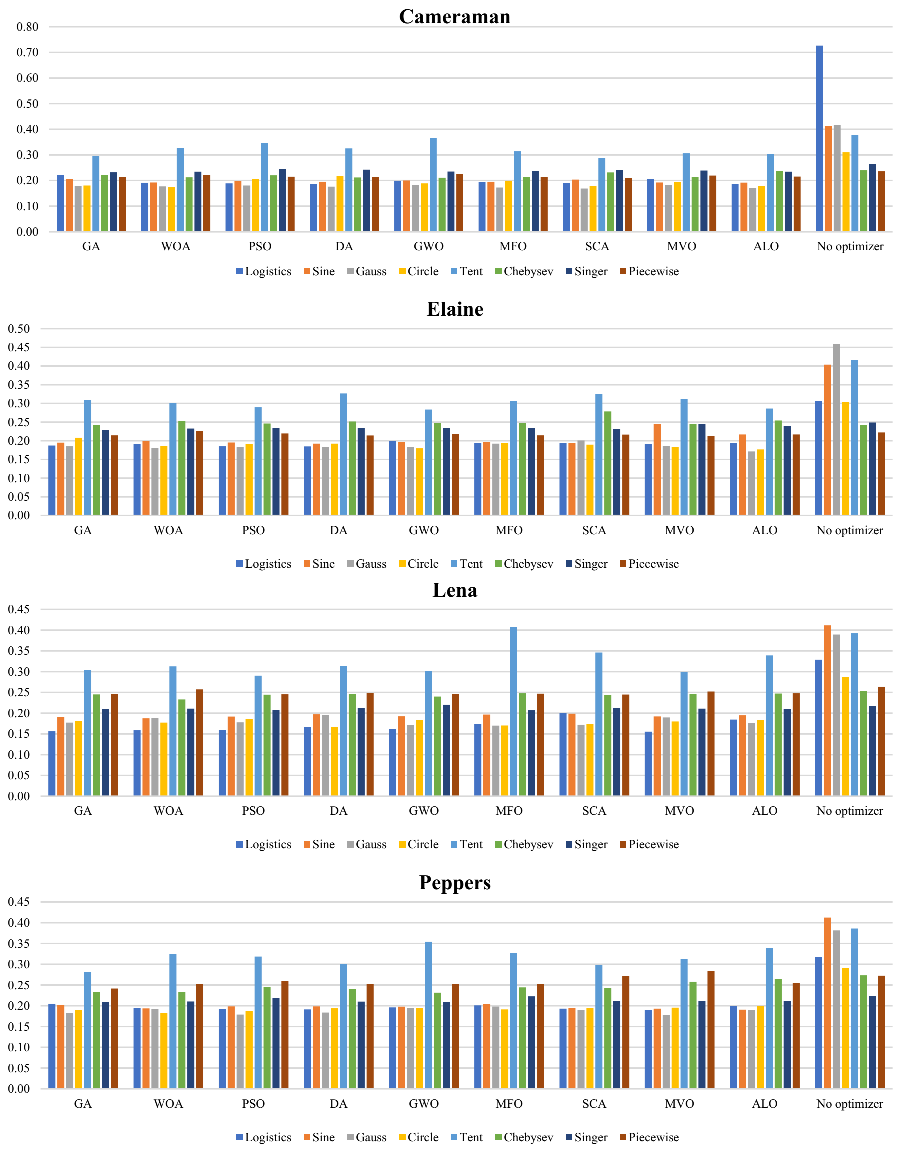


Fig. 22 Encryption time/ decryption time (sec) of the proposed algorithm

6.2.5.2 Unified average changing intensity It calculates the average difference in intensity between two ciphered images, which corresponds to a change of one pixel in the original image. The following equation is utilized to calculate the UACI:

Table 10 Comparisons of time complexity of encrypted and decryption process

Encryption algorithm	Execution time (seconds)
Proposed algorithm (Cameraman)	0.1689
Ref. [123]	0.5539
Ref. [36]	4.711
Proposed algorithm (Elaine)	0.1715
Ref. [126]	0.176
Ref. [122]	1.3105
Proposed algorithm (Lena)	0.1556
Ref. [126]	0.179
Ref. [127]	0.62
Ref. [123]	0.5454
Ref. [128]	1.1044
Ref. [34]	0.25
Proposed algorithm (Peppers)	0.1776
Ref. [126]	0.183
Ref. [129]	0.27444
Ref. [123]	0.5474
Ref. [36]	5.2199
Ref. [131]	0.36

$$UACI = \frac{1}{M \times N}\left[\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{Y_1(i,j) - Y_2(i,j)}{255}\right] \times 100\% \tag{92}$$

where $Y_1(i,j) and Y_2(i,j)$ denote two ciphered images corresponding to the same plain image, differing only by one bit, and M*N is the size of the images. Table 7 shows the results of UACI and NPCR for encryption algorithms using the eight optimized chaotic maps. As seen from Table 4, all results of UACI and NPCR are closer to theoretical values except when using all the chaotic maps without optimization, giving the lowest UACI and NPCR. From the previous results, the optimized eight chaotic maps demonstrate effective resistance to differential attacks. Tables 7 and 8 compare UACI and NPCR with other encryption algorithms, which indicates that the proposed algorithm has the highest sensitivity and is competitive with other proposed encryption methods (Table 9).

6.2.6 Computational speed analysis

It is the time required to execute the algorithm, measured in seconds or milliseconds. Figure 22 compares the time of execution for the eight optimized chaotic maps. It can be observed that the chaotic tent map has a relatively higher computational time than the other chaotic maps. The results regarding computational time for all images demonstrate that the gauss chaotic map has the lowest computational time compared to the other chaotic maps and that using the chaotic maps without optimization has the highest computational time. The proposed algorithm provides the fastest encryption speed, as shown in Table 10.

7 Conclusion

This paper aims to propose an image encryption scheme using optimized chaotic maps. Eight chaotic maps (logistics, sine, gauss, circle, tent, Chebyshev, singer, and piecewise map) are optimized using various metaheuristic optimizers. The encryption process involves chaotic confusion and pixel diffusion. Performance metrics such as entropy, histogram, cross-correlation, computation time, NPCR, UACI, noise attack, data loss, and key analysis are employed to evaluate the proposed scheme. The evaluation of the proposed encryption scheme yielded significant findings. The correlation coefficient values of encrypted images are nearly zero, indicating high security. The histogram analysis reveals uniform distributions, preventing information leakage. The execution time for encryption is recorded as 0.1556 ms, demonstrating real-time applicability. The key space reaches 10^80, ensuring a large key space for robustness. The NPCR value is measured at 99.63%, and the UACI value is 32.92%, further indicating the effectiveness of the encryption scheme. Moreover, the analysis of noise and cropping attacks confirms the robustness of the proposed scheme. Comparative evaluations against other algorithms support the superior performance of the optimized encryption algorithms.

Acknowledgements The authors declare that there is no conflict of interest regarding the manuscript.

Author Contribution SMS Conceptualization, Methodology, Software, Data curation, Visualization and Investigation. HE-DM Supervision, Reviewing and EEHA Supervision, Reviewing and Editing. MMA Supervision, Methodology, Software, Data curation, Writing- Original draft preparation, Reviewing and Editing.

Funding There is no funding to declare regarding this manuscript.

Data availability The data that support the findings of this study are available within this article.

Declarations

Conflict of interest We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. No potential competing interest was reported by the authors.

References

1. Hua Z, Zhu Z, Yi S, Zhang Z, Huang H (2021) Cross-plane colour image encryption using a two-dimensional logistic tent modular map. Inf Sci 546:1063–1083
2. Ahmad M, Agarwal S, Alkhayyat A, Alhudhaif A, Alenezi F, Zahid AH, Aljehane NO (2022) An image encryption algorithm based on new generalized fusion fractal structure. Inf Sci 592:1–20

3. Roy M, Chakraborty S, Mali K, Roy D, Chatterjee S (2021) A robust image encryption framework based on DNA computing and chaotic environment. Microsyst Technol 27(10):3617–3627
4. Elkamchouchi H, Anton R, Abouelseoud Y (2022) New encryption algorithm for secure image transmission through open network. Wirel Personal Commun 2022:1–18
5. Fu J, Gan Z, Chai X, Yang Lu (2022) Cloud-decryption-assisted image compression and encryption based on compressed sensing. Multimed Tools Appl 81(12):17401–17436
6. Arif J, Khan MA, Ghaleb B, Ahmad J, Munir A, Rashid U, Al-Dubai AY (2022) A novel chaotic permutation-substitution image encryption scheme based on logistic map and random substitution. IEEE Access 10:12966–12982
7. Abduljabbar ZA, Abduljaleel IQ, Ma J, Al Sibahee MA, Nyangaresi VO, Honi DG, Abdulsada AI, Jiao X (2022) Provably secure and fast color image encryption algorithm based on s-boxes and hyperchaotic map. IEEE Access 10:26257–26270
8. Song W, Zheng Yu, Chong Fu, Shan P (2020) A novel batch image encryption algorithm using parallel computing. Inf Sci 518:211–224
9. Huang X, Dong Y, Zhu H, Ye G (2022) Visually asymmetric image encryption algorithm based on SHA-3 and compressive sensing by embedding encrypted image. Alex Eng J 61(10):7637–7647
10. Mir UH, Singh D, Lone PN (2022) Color image encryption using RSA cryptosystem with a chaotic map in Hartley domain. Inf Secur J Global Perspect 31(1):49–63
11. Bao W, Zhu C (2022) A secure and robust image encryption algorithm based on compressive sensing and DNA coding. Multimed Tools Appl 81(11):15977–15996
12. Ye G, Liu M, Mingfa Wu (2022) Double image encryption algorithm based on compressive sensing and elliptic curve. Alex Eng J 61(9):6785–6795
13. Isaac SD, Njitacke ZT, Tsafack N, Tchapga CT, Kengne J (2022) Novel compressive sensing image encryption using the dynamics of an adjustable gradient Hopfield neural network. Eur Phys J Special Topics 231(10):1995–2016
14. Belazi A, Kharbech S, Aslam MN, Talha M, Xiang W, Iliyasu AM, Abd El-Latif AA (2022) Improved sine-tangent chaotic map with application in medical images encryption. J Inf Secur Appl 66:103131
15. Muthu JS, Murali P (2021) Review of chaos detection techniques performed on chaotic maps and systems in image encryption. SN Comput Sci 2(5):1–24
16. Mozaffari S (2018) Parallel image encryption with bitplane decomposition and genetic algorithm. Multimed Tools Appl 77(19):25799–25819
17. Talhaoui MZ, Wang X, Midoun MA (2021) Fast image encryption algorithm with high security level using the Bülban chaotic map. J Real-Time Image Process 18(1):85–98
18. Saravanan S, Sivabalakrishnan M (2021) A hybrid chaotic map with coefficient improved whale optimization-based parameter tuning for enhanced image encryption. Soft Comput 25:5299–5322
19. Rezaei B, Ghanbari H, Enayatifar R (2023) An image encryption approach using tuned Henon chaotic map and evolutionary algorithm. Nonlinear Dyn 111(10):9629–9647
20. Shahna KU, Mohamed A (2020) A novel image encryption scheme using both pixel level and bit level permutation with chaotic map. Appl Soft Comput 90:106162
21. Ghazvini M, Mirzadi M, Parvar N (2020) A modified method for image encryption based on chaotic map and genetic algorithm. Multimed Tools Appl 79(37):26927–26950
22. Noshadian S, Ebrahimzade A, Kazemitabar SJ (2020) Breaking a chaotic image encryption algorithm. Multimed Tools Appl 79(35):25635–25655
23. Wang X, Li Y (2021) Chaotic image encryption algorithm based on hybrid multi-objective particle swarm optimization and DNA sequence. Opt Lasers Eng 137:106393
24. Wang J, Song X, El-Latif AA (2022) Single-objective particle swarm optimization-based chaotic image encryption scheme. Electronics 11(16):2628
25. Ferdush J, Mondol G, Prapti AP, Begum M, Sheikh MN, Galib SM (2021) An enhanced image encryption technique combining genetic algorithm and particle swarm optimization with chaotic function. Int J Comput Appl 43(9):960–967
26. Alghafis A, Firdousi F, Khan M, Batool SI, Amin M (2020) An efficient image encryption scheme based on chaotic and Deoxyribonucleic acid sequencing. Math Comput Simul 177:441–466
27. Latha HR, Ramaprasath A. (2022) Optimized two-dimensional chaotic mapping for enhanced image security using sea lion algorithm. In: Emerging research in computing, information, communication and applications: ERCICA 2020, Springer Singapore, *Vol 2*, pp 981–998
28. Kumar A, Raghava NS (2021) An efficient image encryption scheme using elementary cellular automata with novel permutation box. Multimed Tools Appl 80(14):21727–21750

29. Babaei A, Motameni H, Enayatifar R (2020) A new permutation-diffusion-based image encryption technique using cellular automata and DNA sequence. Optik 203:164000
30. Ghazanfaripour H, Broumandnia A (2020) Designing a digital image encryption scheme using chaotic maps with prime modular. Opt Laser Technol 131:106339
31. Abd Elminaam DS, Ibrahim SA, Houssein EH, Elsayed SM (2022) An efficient chaotic gradient-based optimizer for feature selection. IEEE Access 10:9271–9286
32. Naik RB, Singh U (2022) A review on applications of chaotic maps in pseudo-random number generators and encryption. Ann Data Sci 18:1–26
33. Munir N, Khan M, Abd Al Karim Haj Ismail A, Hussain I (2022) Cryptanalysis and improvement of novel image encryption technique using hybrid method of discrete dynamical chaotic maps and brownian motion. Multimed Tools Appl 81(5):6571–6584
34. Lyle M, Sarosh P, Parah SA (2022) Adaptive image encryption based on twin chaotic maps. Multimed Tools Appl 81(6):8179–8198
35. Wang X, Chen S, Zhang Y (2021) A chaotic image encryption algorithm based on random dynamic mixing. Opt Laser Technol 138:106837
36. Yan X, Wang X, Xian Y (2021) Chaotic image encryption algorithm based on arithmetic sequence scrambling model and DNA encoding operation. Multimed Tools Appl 80(7):10949–10983
37. Demir FB, Tuncer T, Kocamaz AF (2020) A chaotic optimization method based on logistic-sine map for numerical function optimization. Neural Comput Appl 32(17):14227–14239
38. Mansouri A, Wang X (2020) A novel one-dimensional sine powered chaotic map and its application in a new image encryption scheme. Inf Sci 520:46–62
39. Mozaffari A, Emami M, Fathi A (2019) A comprehensive investigation into the performance, robustness, scalability and convergence of chaos-enhanced evolutionary algorithms with boundary constraints. Artif Intell Rev 52(4):2319–2380
40. Arora S, Sharma M, Anand P (2020) A novel chaotic interior search algorithm for global optimization and feature selection. Appl Artif Intell 34(4):292–328
41. Khan M, Masood F (2019) A novel chaotic image encryption technique based on multiple discrete dynamical maps. Multimed Tools Appl 78(18):26203–26222
42. Arora S, Anand P (2019) Chaotic grasshopper optimization algorithm for global optimization. Neural Comput Appl 31(8):4385–4405
43. Parameshachari BD, Panduranga HT (2022) Medical image encryption using SCAN technique and chaotic tent map system. In: Recent advances in artificial intelligence and data engineering, Springer, Singapore, pp 181–193
44. Ryu J, Kang D, Won D (2022) Improved secure and efficient chebyshev chaotic map-based user authentication scheme. IEEE Access 10:15891–15910
45. Aydemir SB (2022) A novel arithmetic optimization algorithm based on chaotic maps for global optimization. Evolut Intell 12:1–16
46. Ali TS, Ali R (2022) A novel color image encryption scheme based on a new dynamic compound chaotic map and S-box. Multimed Tools Appl 81(15):20585–20609
47. Sokhangoee ZF, Rezapour A (2022) A novel approach for spam detection based on association rule mining and genetic algorithm. Comput Electr Eng 97:107655
48. Pal S, Kalita K, Haldar S (2022) Genetic algorithm-based fundamental frequency optimization of laminated composite shells carrying distributed mass. J Inst Eng India: Series C 2022:1–13
49. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80(5):8091–8126
50. Saleh A, Yuzir A, Sabtu N, Abujayyab SK, Bunmi MR, Pham QB (2022) Flash flood susceptibility mapping in urban area using genetic algorithm and ensemble method. Geocarto Int 21:1–30
51. Blanchard AE, Shekar MC, Gao S, Gounley J, Lyngaas I, Glaser J, Bhowmik D (2022) Automating genetic algorithm mutations for molecules using a masked language model. IEEE Trans Evolut Comput 26(4):793–799
52. Xu Y, Caihong Hu, Qiang Wu, Jian S, Li Z, Chen Y, Zhang G, Zhang Z, Wang S (2022) Research on particle swarm optimization in LSTM neural networks for rainfall-runoff simulation. J Hydrol 608:127553
53. Jahandideh-Tehrani M, Bozorg-Haddad O, Loáiciga HA (2020) Application of particle swarm optimization to water management: an introduction and overview. Environ Monit Assess 192(5):1–18

54. Shami TM, El-Saleh AA, Alswaitti M, Al-Tashi Q, Summakieh MA, Mirjalili S (2022) Particle swarm optimization: a comprehensive survey. IEEE Access 10:10031–10061
55. Huda RK, Banka H (2020) New efficient initialization and updating mechanisms in PSO for feature selection and classification. Neural Comput Appl 32(8):3283–3294
56. Parouha RP, Verma P (2022) A systematic overview of developments in differential evolution and particle swarm optimization with their advanced suggestion. Appl Intell 52(9):10448–10492
57. Chakraborty S, Saha AK, Sharma S, Mirjalili S, Chakraborty R (2021) A novel enhanced whale optimization algorithm for global optimization. Comput Ind Eng 153:107086
58. Dewi SK, Utama DM (2021) A new hybrid whale optimization algorithm for green vehicle routing problem. Syst Sci Control Eng 9(1):61–72
59. Chakraborty S, Sharma S, Saha AK, Saha A (2022) A novel improved whale optimization algorithm to solve numerical optimization and real-world applications. Art Intell Rev 1:1–112
60. Xu X, Liu C, Zhao Y, Lv X (2022) Short-term traffic flow prediction based on whale optimization algorithm optimized BiLSTM_Attention. Concurr Comput: Practice Exp 34(10):e6782
61. Tang C, Sun W, Xue M, Zhang X, Tang H, Wei Wu (2022) A hybrid whale optimization algorithm with artificial bee colony. Soft Comput 26(5):2075–2097
62. Chatterjee S, Biswas S, Majee A, Sen S, Oliva D, Sarkar R (2022) Breast cancer detection from thermal images using a Grunwald-Letnikov-aided Dragonfly algorithm-based deep feature selection method. Comput Biol Med 141:105027
63. Yang D, Mingliang Wu, Li Di, Yunlang Xu, Zhou X, Yang Z (2022) Dynamic opposite learning enhanced dragonfly algorithm for solving large-scale flexible job shop scheduling problem. Knowl-Based Syst 238:107815
64. Jothi S, Chandrasekar A (2022) An efficient modified dragonfly optimization based mimo-ofdm for enhancing qos in wireless multimedia communication. Wireless Pers Commun 122(2):1043–1065
65. Latchoumi TP, Parthiban L (2022) Quasi oppositional dragonfly algorithm for load balancing in cloud computing environment. Wireless Pers Commun 122(3):2639–2656
66. Meraihi Y, Ramdane-Cherif A, Acheli D, Mahseur M (2020) Dragonfly algorithm: a comprehensive review and applications. Neural Comput Appl 32(21):16625–16646
67. Zhong L, Zhou Y, Luo Q, Zhong K (2021) Wind driven dragonfly algorithm for global optimization. Concurr Comput: Pract Exp 33(6):e6054
68. Chantar H, Tubishat M, Essgaer M, Mirjalili S (2021) Hybrid binary dragonfly algorithm with simulated annealing for feature selection. SN Comput Sci 2(4):1–11
69. Vedik B, Kumar R, Deshmukh R, Verma S, Shiva CK (2021) Renewable energy-based load frequency stabilization of interconnected power systems using quasi-oppositional dragonfly algorithm. J Control, Autom Electr Syst 32(1):227–243
70. Pitchipoo P, Muthiah A, Jeyakumar K, Manikandan A (2021) Friction stir welding parameter optimization using novel multi objective dragonfly algorithm. Int J Lightweight Mater Manufact 4(4):460–467
71. Too J, Mirjalili S (2021) A hyper learning binary dragonfly algorithm for feature selection: a COVID-19 case study. Knowl-Based Syst 212:106553
72. Musharavati F, Khoshnevisan A, Alirahmi SM, Ahmadi P, Khanmohammadi S (2022) Multi-objective optimization of a biomass gasification to generate electricity and desalinated water using Grey Wolf Optimizer and artificial neural network. Chemosphere 287:131980
73. Meidani K, Hemmasian A, Mirjalili S, Barati Farimani A (2022) Adaptive grey wolf optimizer. Neural Comput Appl 34(10):7711–7731
74. Cui F, Al-Sudani ZA, Hassan GS, Afan HA, Ahammed SJ, Yaseen ZM (2022) Boosted artificial intelligence model using improved alpha-guided grey wolf optimizer for groundwater level prediction: comparative study and insight for federated learning technology. J Hydrol 606:127384
75. Qaraad M, Amjad S, Hussein NK, Elhosseini MA (2022) Large scale salp-based grey wolf optimization for feature selection and global optimization. Neural Comput Appl 34(11):8989–9014
76. Zhou J, Huang S, Zhou T, Armaghani DJ, Qiu Y (2022) Employing a genetic algorithm and grey wolf optimizer for optimizing RF models to evaluate soil liquefaction potential. Art Intell Rev 55(7):5673–5705
77. Mirjalili S, Aljarah I, Mafarja M, Heidari AA, Faris H (2020) Grey wolf optimizer: theory, literature review, and application in computational fluid dynamics problems. Nat-inspired Optim 2020:87–105

78. Khalilpourazari S, Doulabi HH, Çiftçioğlu AÖ, Weber GW (2021) Gradient-based grey wolf optimizer with Gaussian walk: application in modelling and prediction of the COVID-19 pandemic. Expert Syst Appl 177:114920
79. Nadimi-Shahraki MH, Taghian S, Mirjalili S (2021) An improved grey wolf optimizer for solving engineering problems. Expert Syst Appl 166:113917
80. Bansal JC, Singh S (2021) A better exploration strategy in Grey Wolf Optimizer. J Amb Intell Human Comput 12(1):1099–1118
81. Ghalambaz M, Yengejeh RJ, Davami AH (2021) Building energy optimization using grey wolf optimizer (GWO). Case Stud Thermal Eng 27:101250
82. Zhao X, Fang Y, Liu Le, Miao Xu, Li Q (2022) A covariance-based Moth–flame optimization algorithm with Cauchy mutation for solving numerical optimization problems. Appl Soft Comput 119:108538
83. Li X, Qi X, Liu X, Gao C, Wang Z, Zhang F, Liu J (2022) A discrete moth-flame optimization with an $ l_2 $-Norm constraint for network clustering. IEEE Trans Netw Sci Eng 9(3):1776–1788
84. Long B, Yang W, Hu Q, Guerrero JM, Garcia C, Rodriguez J, Chong KT (2022) Moth–Flame-optimization-based parameter estimation for FCS-MPC-controlled grid-connected converter with LCL filter. IEEE J Emerg Selected Topics Power Electron 10(4):4102–4114
85. Zhang Yu, Wang P, Yang H, Cui Qi (2022) Optimal dispatching of microgrid based on improved moth-flame optimization algorithm based on sine mapping and Gaussian mutation. Syst Sci Control Eng 10(1):115–125
86. Shehab M, Alshawabkah H, Abualigah L, Al-Madi N (2021) Enhanced a hybrid moth-flame optimization algorithm using new selection schemes. Eng Comput 37(4):2931–2956
87. Chatterjee S, Mohammed AN (2022) Performance evaluation of novel moth flame optimization (MFO) technique for AGC of hydro system. In: IOT with smart systems. Springer, Singapore, pp 377–392
88. Wu H-H, Ke G, Wang Y, Chang Y-T (2022) Prediction on recommender system based on bi-clustering and moth flame optimization. Appl Soft Comput 120:108626
89. Mittal T (2022) A hybrid moth flame optimization and variable neighbourhood search technique for optimal design of IIR filters. Neural Comput Appl 34(1):689–704
90. Neelamkavil Pappachan S (2022) Hybrid red deer with moth flame optimization for reconfiguration process on partially shaded photovoltaic array. Energy Sourc, Part A: Recov, Utiliz Environ Effects 5:1–27
91. Zhang Bo, Tan R, Lin C-J (2021) Forecasting of e-commerce transaction volume using a hybrid of extreme learning machine and improved moth-flame optimization algorithm. Appl Intell 51(2):952–965
92. Xia J, Yang D, Zhou H, Chen Y, Zhang H, Liu T, Heidari AA, Chen H, Pan Z (2022) Evolving kernel extreme learning machine for medical diagnosis via a disperse foraging sine cosine algorithm. Comput Biol Med 141:105137
93. Al-Betar MA, Awadallah MA, Zitar RA, Assaleh K (2022) Economic load dispatch using memetic sine cosine algorithm. J Amb Intell Human Comput 7:1–29
94. Abualigah L, Ewees AA, Al-qaness MA, Elaziz MA, Yousri D, Ibrahim RA, Altalhi M (2022) Boosting arithmetic optimization algorithm by sine cosine algorithm and levy flight distribution for solving engineering optimization problems. Neural Comput Appl 34(11):8823–8852
95. Gupta S, Zhang Yi, Rong Su (2022) Urban traffic light scheduling for pedestrian–vehicle mixed-flow networks using discrete sine–cosine algorithm and its variants. Appl Soft Comput 120:108656
96. Abdel-Mawgoud H, Fathy A, Kamel S (2022) An effective hybrid approach based on arithmetic optimization algorithm and sine cosine algorithm for integrating battery energy storage system into distribution networks. J Energy Storage 49:104154
97. Rayaguru NK, Sekar S (2022) Modified multiverse optimization, perturb and observer algorithm-based MPPT for grid-connected photovoltaic system. In: Proceedings of International Conference on Power Electronics and Renewable Energy Systems. Springer, Singapore, pp 647–658
98. Anshuman A, Eldho TI (2022) Entity aware sequence to sequence learning using LSTMs for estimation of groundwater contamination release history and transport parameters. J Hydrol 608:127662

99. Fu Y, Zhou MC, Guo X, Qi L, Sedraoui K (2021) Multiverse optimization algorithm for stochastic biobjective disassembly sequence planning subject to operation failures. IEEE Trans Syst, Man, Cyber: Syst 52(2):1041–1051
100. Chouksey M, Jha RK (2021) A multiverse optimization based colour image segmentation using variational mode decomposition. Expert Syst Appl 171:114587
101. Chouksey M, Jha RK (2021) A joint entropy for image segmentation based on quasi opposite multiverse optimization. Multimed Tools Appl 80(7):10037–10074
102. Zhu W, Huang L, Mao L, Esmaeili-Falak M (2022) Predicting the uniaxial compressive strength of oil palm shell lightweight aggregate concrete using artificial intelligence-based algorithms. Struct Concr. https://doi.org/10.1002/suco.202100656
103. Liu J, Wei J, Heidari AA, Kuang F, Zhang S, Gui W, Chen H, Pan Z (2022) Chaotic simulated annealing multi-verse optimization enhanced kernel extreme learning machine for medical diagnosis. Comput Biol Med 144:105356
104. Nazari N, Mousavi S, Mirjalili S (2021) Exergo-economic analysis and multi-objective multiverse optimization of a solar/biomass-based trigeneration system using externally-fired gas turbine, organic Rankine cycle and absorption refrigeration cycle. Appl Therm Eng 191:116889
105. Zhang X, Nguyen H, Choi Y, Bui X-N, Zhou J (2021) Novel extreme learning machine-multi-verse optimization model for predicting peak particle velocity induced by mine blasting. Nat Resour Res 30(6):4735–4751
106. Aljarah I, Faris H, Heidari AA, Mafarja MM, Ala'M AZ, Castillo PA, Merelo JJ (2021) A robust multi-objective feature selection model based on local neighborhood multi-verse optimization. IEEE Access 9:100009–100028
107. Kavitha J, Thirupathi Rao K (2022) Dynamic resource allocation in cloud infrastructure using ant lion-based auto-regression model. Int J Commun Syst 35(6):e5071
108. Li Q, Li D, Zhao K, Wang L, Wang K (2022) State of health estimation of lithium-ion battery based on improved ant lion optimization and support vector regression. J Energy Storage 50:104215
109. Manikanda Selvam S, Yuvaraj T. (2022) Optimal allocation of capacitor using ant lion optimization algorithm. In: Proceedings of International Conference on Data Science and Applications. Springer, Singapore, pp 279–286
110. Soesanti I, Syahputra R (2022) Multiobjective ant lion optimization for performance improvement of modern distribution network. IEEE Access 10:12753–12773
111. Zhou Z, Ji H, Yang X (2022) Illumination correction of dyed fabric based on extreme learning machine with improved ant lion optimizer. Color Res Appl 47(4):1065–1077
112. Kesarwani S, Verma RK (2022) Ant Lion Optimizer (ALO) algorithm for machinability assessment during Milling of polymer composites modified by zero-dimensional carbon nano onions (0D-CNOs). Measurement 187:110282
113. Niu G, Li X, Wan X, He X, Zhao Y, Yi X, Chen C, Xujun L, Ying G, Huang M (2022) Dynamic optimization of wastewater treatment process based on novel multi-objective ant lion optimization and deep learning algorithm. J Clean Prod 345:131140
114. Vennila H, Rajesh R. (2022) Ant lion optimization for solving combined economic and emission dispatch problems. In: Machine learning and autonomous systems, Springer, Singapore pp 639–649
115. M'Hamdi B, Benmahamed Y, Teguar M, Taha IB, Ghoneim SS (2022) Multi-objective optimization of 400 kV composite insulator corona ring design. IEEE Access 10:27579–27590
116. Veramalla R, Arya SR, Gundeboina V, Jampana B, Chilipi R, Madasthu S (2022) Meta-heuristics algorithms for optimization of gains for dynamic voltage restorers to improve power quality and dynamics. Opt Control Appl Methods 44(2):1006–1025
117. Midoun MA, Wang X (2021) Talhaoui MZ A sensitive dynamic mutual encryption system based on a new 1D chaotic map. Optics Lasers Eng 139:106485
118. Zhang S, Liu L (2021) A novel image encryption algorithm based on SPWLCM and DNA coding. Math Comput Simul 190:723–744
119. Wang X, Sun H (2020) A chaotic image encryption algorithm based on improved Joseph traversal and cyclic shift function. Opt Laser Technol 122:105854
120. Muñoz-Guillermo M (2021) Image encryption using q-deformed logistic map. Inf Sci 552:352–364
121. Liu X, Xiao Di, Liu C (2021) Three-level quantum image encryption based on Arnold transform and logistic map. Quantum Inf Process 20(1):1–22

122. Wang X, Guan N, Liu P (2022) A selective image encryption algorithm based on a chaotic model using modular sine arithmetic. Optik 258:168955
123. Guo Z, Sun P (2022) Improved reverse zigzag transform and DNA diffusion chaotic image encryption method. Multimed Tools Appl 81(8):11301–11323
124. Liu X, Tong X, Wang Z, Zhang M (2022) A new n-dimensional conservative chaos based on generalized hamiltonian system and its' applications in image encryption. Chaos, Solitons Fractals 154:111693
125. Patro KA, Acharya B (2021) An efficient dual-layer cross-coupled chaotic map security-based multi-image encryption system. Nonlinear Dyn 104(3):2759–2805
126. Zareai D, Balafar M, Derakhshi MRF (2021) A new Grayscale image encryption algorithm composed of logistic mapping, Arnold cat, and image blocking. Multimed Tools Appl 80(12):18317–18344
127. Khalil N, Sarhan A, Alshewimy MA (2021) An efficient color/grayscale image encryption scheme based on hybrid chaotic maps. Opt Laser Technol 143:107326
128. Wang X, Gao S (2021) A chaotic image encryption algorithm based on a counting system and the semi-tensor product. Multimed Tools Appl 80(7):10301–10322
129. Wang X, Gao S (2020) Image encryption algorithm for synchronously updating Boolean networks based on matrix semi-tensor product theory. Inf Sci 507:16–36
130. Mondal B, Behera PK, Gangopadhyay S (2021) A secure image encryption scheme based on a novel 2D sine–cosine cross-chaotic (SC3) map. J Real-Time Image Process 18(1):1–18
131. Jan A, Parah SA, Malik BA (2022) IEFHAC: image encryption framework based on hessenberg transform and chaotic theory for smart health. Multimed Tools Appl 81(13):18829–18853
132. Xu S, Chang C-C, Liu Y (2021) A high-capacity reversible data hiding scheme for encrypted images employing vector quantization prediction. Multimed Tools Appl 80(13):20307–20325
133. Wu H, Li F, Qin C, Wei W (2019) Separable reversible data hiding in encrypted images based on scalable blocks. Multimed Tools Appl 78(18):25349–25372
134. Kamrani A, Zenkouar K, Najah S (2020) A new set of image encryption algorithms based on discrete orthogonal moments and Chaos theory. Multimed Tools Appl 79(27):20263–20279
135. Yu W, Liu Ye, Gong L, Tian M, Liangqiang Tu (2019) Double-image encryption based on spatiotemporal chaos and DNA operations. Multimed Tools Appl 78(14):20037–20064
136. Premkumar R, Anand S (2019) Secured and compound 3-D chaos image encryption using hybrid mutation and crossover operator. Multimed Tools Appl 78(8):9577–9593

Authors and Affiliations

Sally Mohamed Sameh[1] · Hossam El-Din Moustafa[1] · Ehab H. AbdelHay[1,3] · Mohamed Maher Ata[2]

✉ Mohamed Maher Ata
mmaher844@yahoo.com

[1] Department of Electronics and Communications Engineering, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt

[2] Department of Communications and Electronics Engineering, MISR Higher Institute for Engineering and Technology, Mansoura 35516, Egypt

[3] Faculty of Engineering, Mansoura National University, Mansoura, Egypt