Received: 31 January 2024 | *Revised:* 1 March 2024 | *Accepted:* 14 June 2024
DOI: 10.37917/ijeee.22.1.24 *Vol. 22 | Issue 1 | June 2026*

Iraqi Journal for Electrical and Electronic Engineering
Original Article

Open Access
IJEEE University of Basrah College of Engineering

Adaptive Multi Objective Chicken Swarm Optimization for Solving Nonlinear Stream Cryptosystem

Mohammed H. Ahmed*[1], Asaad N. Hashim[2], Khalid A. Hussein[1]
[1]Computer Science Department, College of Education, Mustansiriyah University, Baghdad, Iraq
[2]Computer Science Department, College of Computer Science and Math, University of Kufa, Najaf, Iraq

Correspondance
*Mohammed Hussein Ahmed
Computer Science Department, College of Education, Mustansiriyah University, Baghdad, Iraq
Email: mohammedalbawi@uomustansiriyah.edu.iq

Abstract
Nonlinear stream ciphers have become a viable alternative to traditional cryptosystems in response to the growing need for secure communication. These ciphers generate a keystream via feedback mechanisms and nonlinear functions, which are then utilized for encryption. Geffe generator system is one of the most keystream generators. Also, these systems have many benefits, like being fast, flexible, and able to create unpredictable and non-repeating keystreams, these systems are susceptible to cryptanalysis attacks, which have the potential to compromise their security. This paper presents the first study of applying chicken swarm optimization (CSO) algorithm in the field of cryptanalysis based on cipher only attack. The standard CSO algorithm and an adaptive multi points CSO (AMPCSO) algorithm are proposed to cryptanalysis nonlinear stream cipher based on Geffe keystream generator. Firstly, the traditional CSO is used to reveal the secret initial values of the Geffe generator. Secondly, an adaptive multi points chicken swarm optimization (AMPCSO) has been proposed to enhance the traditional CSO algorithm to attack Geffe generator systems. The AMPCSO is a new idea to advance the CSO search abilities and improve the foraging behavior of hens and chicks by allowing hens to be influenced by other individuals within the same or different groups and affected by the best individual in the population and enable chicks to learn from four reference points rather than learn from their respective mothers only. Lastly, a new criterion is used to estimate the value of fitness by utilizing a multi-objective fitness function (MOFF), which is grounded on Pareto dominance. The experimental results showed that the CSO and AMPCSO are very effective tools in terms of accuracy, information required, and CPU times when applied to the analysis of nonlinear stream cipher. The AMPCSO required a few characters from ciphertext to attack systems with total LFSRs length up to 59 bits with an appropriate CPU time.

Keywords
Chicken Swarm Optimization, Ciphertext Only Attack, Cryptanalysis, Geffe, LFSR, Nonlinear Stream Cipher.

I. Introduction

Within the field of cryptology, research delves into strategies that assure the secrecy of information. Cryptology consists of two primary domains: cryptography and cryptanalysis [1]. The former involves the creation of cryptosystems that are challenging to decipher, while the latter involves the study of techniques for analyzing these cryptosystems. The cryptanalysis of any given cryptosystem can be framed as an optimization challenge [2]. Depending on the extent of information accessible to the attacker, various techniques can be employed by a cryptanalyst to breach a cipher [3]. The Geffe generator is a notable stream cipher design widely used in cryptography and operates by combining the outputs of three Linear Feedback Shift Registers (LFSRs) through bitwise logical operations to produce pseudorandom bits that can be used to encrypt plaintexts [4, 5]. Like any cryptographic algorithm, the Geffe generator is not immune to attacks. Cryptanalysts have developed various strategies to analyze and potentially

break the cipher, including correlation, divide-and-conquer attacks, evolutionary algorithms, and other methods [6].

The swarm intelligence optimization algorithm belongs to biomimetic random search algorithms that emulate the natural ecosystem mechanisms to address intricate optimization challenges. This algorithm operates without a central controlling entity and relies on straightforward individual interactions [7]. Despite the simplicity of these interactions, they collectively possess the capacity to tackle intricate problems within a limited timeframe efficiently. This characteristic renders swarm intelligence optimization algorithms exceptionally well-suited for the practical resolution of complex optimization problems [8].

Introduced by Meng et al. in 2014 [9], the chicken swarm optimization (CSO) algorithm belongs to the category of swarm intelligence algorithms. This algorithm replicates the hierarchical structure and foraging behavior seen in chicken swarms. Within CSO, each member of the swarm is assigned one of three identities: rooster, hen, or chick, each carrying out distinct functions. When applied to solve optimization problems, individual chickens within the swarm represent potential solutions, and their search strategies are tailored based on their respective identities within the swarm [10]. This paper introduces a new methodology called the AMPCSO algorithm, which enhances the CSO algorithm and focuses on improving the learning strategy of hens and chicks. The CSO and AMPCSO are used in this paper to recover the initial secret value of a nonlinear stream cipher through a cipher only attack using multi objective function. The research organization encompasses a comprehensive review of variation on CSO and the application of intelligent systems in cryptanalysis, covered in Section II. . Section III. provides a concise depiction of CSO-based Swarm techniques. Elaborations on the stream cryptosystem are presented in Section V. . Sections V. and VI. are given the proposed system and multi objective fitness. Applying AMPCSO to cryptanalysis Geffe generator is shown in section VII. . The attained results are thoroughly deliberated in Section VIII. , while Section IX. contains the conclusion of our work.

II. Related Works

Over the past decades, numerous researchers have enhanced CSO (Chicken Swarm Optimization) algorithms to tackle various optimization challenges. As an illustration, in 2015, D. Wu et al. [11] introduced modifications to the CSO algorithm by incorporating knowledge acquired from roosters into the position update formula for chicks. This adaptation aimed to prevent the iterative process from becoming trapped in local optima. Similarly, N. Irsalinda et al. [12] refined the CSO algorithm by removing parameters related to roosters, hens, and chicks. This alteration enabled the algorithm to address optimization problems, effectively outperforming PSO and GA performance.

Furthermore, M. Lin et al. [13] proposed a strategy to achieve a better equilibrium between diversification and intensification within the swarm. This was achieved by allowing roosters to explore the search space early in the process and focusing on exploitation later. Moreover, the swarm's diversity was enhanced by enabling hens to learn from chicks. In 2019, D. He et al. [14] introduced enhancements to the chicken position formula to elevate global optimization performance and accelerate convergence. This was achieved by expanding the influence of mother hens' positions and accounting for neighboring roosters' positions. J. Wang et al. [15] introduced an upgraded chick's position updating formula. This improvement involved enabling chicks to learn from roosters within their group, done with a specific degree of influence, to steer clear of local optimization pitfalls. X. Liang et al. [16] proposed advancements to hen's and chick's position formulas. They integrated Levy flight and nonlinear weight reduction strategies into the update process. This adjustment ensured an equitable distribution across the population while preventing premature convergence. M. Gamal et al. [17] introduced a hybrid method known as CSOGA, which merges the effectiveness and solution convergence capabilities of Chicken Swarm Optimization (CSO) and Genetic Algorithm (GA) for text summarization. This amalgamated approach was designed to achieve optimal solutions. The outcomes of their study highlight that the hybrid CSOGA technique exhibited superior performance in terms of text summarization quality. In 2022, G. Yanchun et al. [18] introduced a simplified iteration of CSO. This variant eliminated the chicks, and an adaptive approach was implemented using an inverted S-shaped inertial weight. This adaptive simplified CSO algorithm retained only roosters and hens foraging for solutions. In a separate work in 2022, J. Liang and team [19] enhanced the standard CSO algorithm to overcome the issue of premature convergence in handling multimodal optimization problems. They introduced an Improved Chicken Swarm Optimization (ICSO) algorithm, drawing inspiration from combining PSO's concept with the bacterial foraging algorithm's replication and elimination-dispersal operations. In 2023, J. Liang and colleagues [20] presented the Adaptive Dual-Population Collaborative CSO (ADPCCSO) algorithm. This approach was devised to tackle complex high-dimensional problems. It incorporated an adaptive adjustment strategy for parameter G, a refined foraging behavior strategy, and a dual-population collaborative optimization approach.

Numerous prior studies have advocated advancing the cryptanalysis of stream cipher systems using intelligent approaches. In their work, Maiya Din et al. [21] investigated a pair of strategies: a divide-and-conquer attack and a genetic algorithm for assaulting the Geffe generator. The simulation

outcomes demonstrate the efficient acquisition of accurate initial states for all shift registers with combined lengths of up to 39. Iwona Polak and Mariusz Boryczka [22] employed Genetic Algorithms (GA) to compromise the RC4 stream cipher. As indicated by average findings, this endeavor effectively uncovered 59% to 80% of RC4 keystream bits. In two situations, Klaus Pommerening's work [23] analyzed NLFSR with the BK algorithm. In the first, with no prior knowledge, the BK algorithm wasn't a viable attack method. However, in the second scenario, where the cryptanalyst knew about a limit on variable coefficients in the feedback function, the BK algorithm demonstrated effective cryptanalysis capabilities. Maiya Din et al. [24] developed the Cuckoo Search (CS) evolutionary algorithm to address LFSR with varying polynomial degrees. Their approach necessitated a ciphertext segment of 200 characters to retrieve the secret initial value, with a maximum length of 19. In [25], S. Abbas and R. Kadhum proposed an enhanced PSO algorithm by incorporating Simulated Annealing (SA) to solve the Geffe stream cipher through a ciphertext-only attack. Their method achieved success in retrieving secret keys of lengths up to 12. S. Sadkha and B. Yaseen [26] introduced an approach for attacking stream ciphers using a DNA algorithm alongside parallel computations. Their study demonstrated that the DNA algorithm could efficiently determine the precise initial state with only a tiny ciphertext segment. M. Din et al. [27] attacked the Geffe generator using the binary PSO algorithm. The B_PSO algorithm was introduced to accurately determine the initial state when applied to decrypt ciphertext encoded using the Geffe stream cipher, with a cumulative LFSR length of up to 29. This algorithm showcased its proficiency in decrypting ciphered texts containing 400 characters. I. Polak and M. Boryczka [28, 29] put forward the concept of employing the Tabu search algorithm to target stream cipher systems like VMPC and RC4+. The experimental outcomes notably showcased the successful application of Tabu search in consistently identifying VMPC 10 and VMPC 16 ciphers across all test scenarios. Furthermore, the algorithm demonstrated its capability in accurately determining the correct internal states of RC4+, with an average examination of just 250 potential internal states. G. Mishra et al. [30] introduced a method involving black-box analysis deep learning to authenticate RC4, Trivium, and TRIAD stream ciphers. Their approach dispenses with the necessity for pre-existing mathematical or statistical analysis. The research findings underscore that various factors, including network design, the quantity of training data, and the duration of training, can impact the precision of predictions. R. Rizk-Allah et al. [31] targeted the RC4 stream cipher using a fusion of the B_PSO and Equilibrium Optimizer (EO) techniques. This combined approach was employed to deduce the initial key of the RC4 stream cipher. The outcomes of their study showcased that the devised system effectively retrieved the initial state by examining 104 internal stages through a probable word attack. R. Kadhum [32] merged the PSO and SA algorithms to address nonlinear stream ciphers through a ciphertext-only attack strategy. The devised approach successfully retrieved the secret initial value of the Geffe generator, even when the combined length of LFSRs reached up to 19.

III. Chicken Swarm Optimization (CSO)

In 2014, Meng and colleagues introduced Chicken Swarm Optimization (CSO) as a global optimization algorithm. This algorithm abstractly mimics the foraging behaviour of chickens and incorporates the strengths of genetic algorithms, particle swarm optimization algorithms, and bat algorithms [9]. Chicken Swarm Optimization (CSO) emulates the behaviour of a collective of chickens through various rules, including group grading, inter-group competition, hatching of hens, and chick growth [33]. This algorithm has garnered significant attention in scientific research and offers a novel approach to addressing global optimization challenges across domains like computing, engineering, and management science [34]. Fig. 1 shows the flowchart of the CSO algorithm.

In the foraging context, a strict hierarchical structure exists within the chicken group. During foraging, individuals are categorized into roosters, hens, and chicks. Within this hierarchy, hens follow the lead of roosters while chicks explore their surroundings close to the hens. Consequently, the rooster assumes a leadership role within the chicken group, enjoying a competitive edge during foraging [35]. Meanwhile, chicks with less proficient foraging abilities can only follow the hens in their foraging efforts. The majority of the chicken population consists of hens, and they make random choices regarding which group to join. Additionally, the mother-child connection between hens and chicks must be more active. The dynamics of the dominance and mother-child relationships within a group remain unchanged. They are periodically updated after a certain number of time steps (denoted as "G") [36]. The level of dominance displayed by each individual in the chicken group is determined by a fitness function corresponding to the target function at its specific position. In order to simulate the behaviour of hens tending to their hatchlings and chicks maturing into either roosters or hens, the CSO algorithm assesses each chick's fitness after they complete their foraging. Once graded, the members of the chicken group embark on a new foraging cycle based on an updated position formula, continuing until they satisfy the termination condition. Upon completion of the CSO process, the position held by the fittest individual within the chicken group represents the optimal solution to the optimization problem [37].

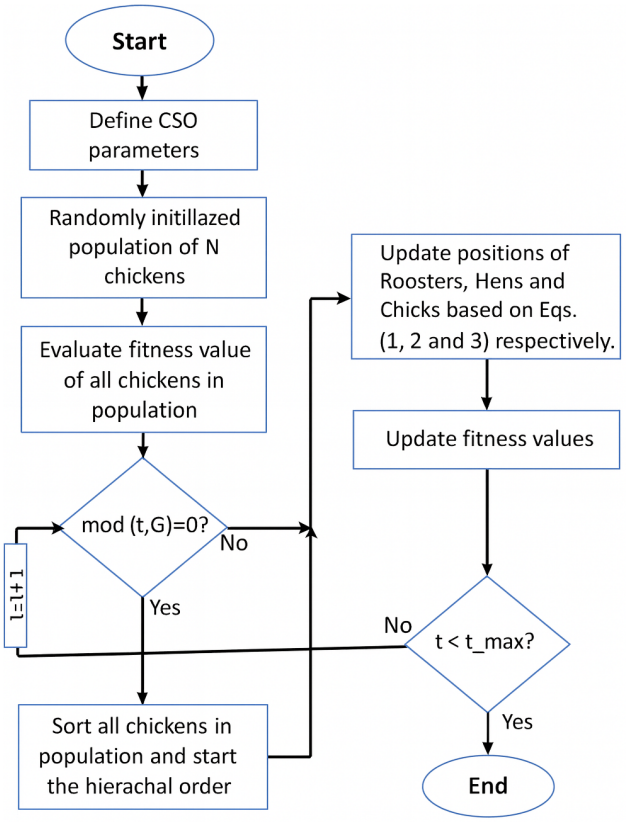

Fig. 1. Flowchart of Basic CSO

A. First Foraging (Roosters)

Within the CSO algorithm, we assume a population of chickens denoted as N, which are organized in ascending order based on their fitness levels. The leading N_R chickens are designated as roosters, the trailing N_C chickens as chicks, and the rest, $N_H = (N - N_R - N_C)$, are categorized as hens. The individual with the highest fitness value in the group can explore a broader range of food sources, enabling enhanced exploration capabilities [38]. The update of a rooster's position $p^t_{i,j}$ is influenced by another randomly chosen rooster. This position update is governed by equation (1):

$$p^{t+1}_{i,j} = p^t_{i,j}\left(1 + R(0,\sigma^2)\right), \tag{1}$$

where

$$\sigma^2 = \begin{cases} 1, & \text{if } ff_i \le ff_k, \\ \exp\left(\frac{ff_k - ff_i}{|ff_i| + \zeta}\right), & \text{otherwise,} \end{cases} \quad k \in \{1,\ldots,N\}, \quad k \neq i.$$

Here, $p^t_{i,j}$ represents the position of rooster i in the j-th dimension at iteration t; $p^{t+1}_{i,j}$ denotes the position at iteration $t+1$. $R(0,\sigma^2)$ represents a Gaussian distribution function with mean value of 0 and variance of σ^2. ff_i and ff_k represents the fitness function of current rooster and random selected rooster respectively. Adding ζ to avoid zero partition error which is the smallest value in computer.

B. Second Foraging (Hens)

Assuming that $p^t_{i,j}$ represents the position of hen in the j-th dimension space within t iteration. The update of hen's positions based on (2).

$$p^{t+1}_{i,j} = p^t_{i,j} + u_1 \cdot \alpha \cdot (p^t_{r_1,j} - p^t_{i,j}) + u_2 \cdot \beta \cdot (p^t_{r_2,j} - p^t_{i,j}) \tag{2}$$

where

$$u_1 = \exp\left(\frac{ff_i - ff_{r_1}}{|ff_i| + \zeta}\right), \quad u_2 = \exp\left(ff_{r_2} - ff_i\right).$$

In this scenario, α and β are both uniformly generated random numbers within the range $[0,1]$. The variable r_1 corresponds to an index representing a rooster selected from the current group, while r_2 denotes an index for a chicken (either rooster or hen) randomly chosen from the overall population of N chickens [39].

C. Third Foraging (Chicks)

The formulation for the chicks' behavior of moving around their mother in search of food is as follows:

$$p^{t+1}_{i,j} = p^t_{i,j} + T \cdot (p^t_{m,j} - p^t_{i,j}) \tag{3}$$

Where $p^t_{m,j}$ represents the location of the mother of the i-th chicks. The parameter T signifies the tendency of the chick to trail its mother while searching for food, with T being randomly selected from a range between 0 and 2. Table I presents the description of abbreviations and symbols used throughout of paper.

IV. Nonlinear Stream Cryptosystem

Stream ciphers are classified as symmetric ciphers, wherein the encryption and decryption of messages necessitate the possession of an identical key by both the sender and the receiver [40]. The primary function of a stream cipher is the generation of a critical stream through the utilization of a random seed which is derived from either the secret key itself or is dependent upon it and LFSRs which is a shift registers where the input bit is determined by a linear function of its prior state. The initial value is modified for each subsequent keystream generation to enhance the randomness of the keystream. The keystream undergoes a bitwise XOR

TABLE I.
DESCRIPTION OF ABBREVIATIONS AND SYMBOLS

Abbreviations & Symbols	Description
CSO	Chicken Swarm Optimization
AMPCSO	Adaptive Multi Points CSO
CSOGA	CSO and Genetic Algorithm
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
ADPCCSO	Adaptive Dual Population Collaborative CSO
BK	Boyar and Krawczyk
CS	Cuckoo Search
SA	Simulated Annealing
B_PSO	Binary PSO
EO	Equilibrium Optimizer
N	Population size
$t, t_{\max}$	Current and maximum iteration
p_i^t	Position of current individual
N_R, N_H, N_C, MH	Number of Roosters, Hens, Chicks, and Mother Hens respectively
ff, G, ζ	Fitness value, threshold of re-ranking, and smallest computer value
α, β	Random numbers in range $[0,1]$
T	Random number in range $[0,2]$
LFSR	Linear Feedback Shift Register
LC	Linear Complexity
$p_r^t, p_b^t, p_n^t, p_m^t$	Positions of rooster, best, random, and mother chickens
ff_{best}	Fitness of the best individual
w, IC	Inertia weight and Index of Coincidence
lb, ub	Lower and upper bounds $[-1,1]$
MOFF	Multi-Objective Fitness Function
MOFF(1), MOFF(2)	MOFF of plaintext and ciphertext
w_1, w_2	Weights in range $[0,1]$
Diff	Difference between fitness of plaintext and ciphertext
LPT	Length of plaintext
MTL	Maximum Text Length
AMOFF, BMOFF	Actual and Best MOFF
TLR	Total Length of LFSRs
BCT	Best Consuming Time
Biter	Best Iteration
RRK	Ratio of Recovered Key
BFA	Brute Force Attack

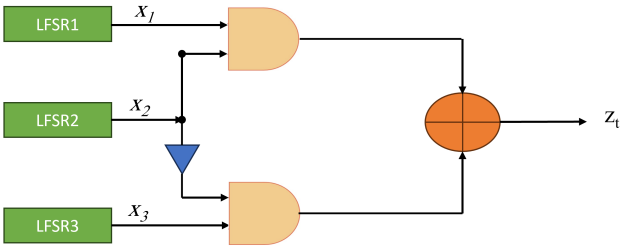

Fig. 2. Geffe Generator

operation with the plaintext to generate the ciphertext. Geffe and Bruer generator systems generate keystreams based on LFSRs [41]. Geffe generator is stream cipher cryptosystem originally proposed by P.R. Geffe in 1973 and is comprised of three LFSRs with different lengths that are relatively prime, with two serving as input multiplexers and the third as a controller [5]. The feedback function polynomial degrees are also assumed to be relatively prime. The Geffe Generator scheme can be visualized in Fig. 2 [42]. The Geffe Generator introduces non-linearity through a function that combines the outputs of the two input LFSRs and the complement of the output from the controller LFSR using the Boolean AND function [43]. The combining function is:

$$f(x_1, x_2, x_3) = x_1 x_2 \oplus x_2 x_3 \oplus x_3 \tag{4}$$

where the symbol $\oplus$ means the XOR logic operation. LFSR2 acts as selector switching the output between LFSR1 and LFSR3. The keystream sequence $\{z_t\}$ obtained from the Geffe generator has period

$$T = (2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$$

and linear complexity

$$LC = L_1 L_2 + L_2 L_3 + L_3$$

The main steps to produce ciphertext using the Geffe generator are as follows [44]:

1. **Initialization:** Each LFSR starts with an initial state. This initial state is usually set as a binary sequence of 0s and 1s. The choice of these initial states is crucial for the quality and randomness of the generated sequence.

2. **Clocking:** At each clock cycle, all three LFSRs are shifted by one position. The least significant bit of each LFSR's current state becomes the output bit for that clock cycle.

3. **Combining Outputs:** The outputs of the three LFSRs are then fed into a majority function. The majority

function determines the bit that appears most frequently among the three outputs and assigns that value to the next bit in the Geffe generator's output sequence.

4. **Generating Sequence:** The output of the majority function becomes the next bit in the pseudorandom sequence. This bit is then fed back into each LFSR to influence the generation of subsequent bits.

5. **Repeating the Process:** The clocking, combining, and generating steps are repeated for each clock cycle, producing a continuous stream of pseudorandom bits.

6. **Encryption:** The pseudorandom bits are XORed with binary plain text to generate ciphertext.

V. An Adaptive Multi Points Chicken Swarm Optimization (AMPCSO)

CSO can behave intelligently to optimize problems efficiently. However, the algorithm's blind spots in search direction arise from its method of updating the individual's positions. The algorithm's search direction and velocity are primarily dictated by the search proficiency of hens. Furthermore, both hens and chicks predominantly rely on the guidance provided by roosters when searching for food. As a result, the algorithm's performance can be significantly influenced by the actions and information-sharing capabilities of the roosters.

Reducing the occurrence of roosters getting stuck in local optima can enhance the performance of the CSO algorithm when compared to traditional bionics heuristic algorithms. However, a more significant number of roosters within the same group and roosters in the reference non-self-group tend to converge towards a local optimum. In that case, this can lead to hens' positions gravitating towards local optima during their updates. This can result in hens becoming trapped in local optima, hindering the algorithm's ability to find the optimal solution. Algorithm (1) shows the algorithm steps of the AMPCSO algorithm.

To address the limitation associated with hens' foraging behavior, this study enhances the exploration step by enabling hens to be influenced by other individuals within the same or different groups ($p^t_{n,j}$), and it strengthens the exploitation step by allowing hens to be affected by the best individual in the population ($p^t_{b,j}$). These improvements involve the creation of four candidate points, as demonstrated in the equations below:

$$Y^1 = S_1 \cdot \mathrm{R} \cdot (p^t_{r1,j} - p^t_{i,j}) \tag{5}$$

$$Y^2 = S_2 \cdot \mathrm{R} \cdot (p^t_{r2,j} - p^t_{i,j}) \tag{6}$$

$$Y^3 = S_3 \cdot \mathrm{R} \cdot (p^t_{n,j} - p^t_{i,j}) \tag{7}$$

Algorithm 1 AMPCSO

1. Input:
- 1.1 Population size (N), maximum iteration ($t_{\max}$).
- 1.2 Number of Roosters (NR), Hens (NH), Chicks (NC) and Mother Hens (MH).
- 1.3 Lower and upper bounds of search space (lb, ub).
- 1.4 Threshold of re-ranking (G).

2. Output:
- 2.1 Optimal solution (individual with highest fitness in population).

3. Initialization:
- 3.1 Randomly initialize (N) population.

4. Evaluation:
- 4.1 For every element in population:
  - 4.1.1 Calculate fitness values.
  - 4.1.2 Select the individual with best fitness value as best solution.

5. Ranking:
- 5.1 Sort the population based on fitness value in descending order.

6. Population classification:
- 6.1 Determine the highest elements as roosters (R) according to NR.
- 6.2 Take the remaining highest elements as hens (H) according to NH.
- 6.3 The last elements in population are determined as chicks (C) according to NC.
- 6.4 Randomly determine mother hens (MH) from H population according to NM.

7. Subgroups division:
- 7.1 Split population into several predefined subgroups.
- 7.2 Each subgroup must contain at least: one rooster, two hens and chicks.

8. Evolution processes:
- 8.1 For every subgroup in population:
  - 8.1.1 Update Roosters (R) positions based on Eq. (1).
  - 8.1.2 Update Hen (H) positions based on Eqs. [5-18].
  - 8.1.3 Update Chicks (C) positions based on Eqs. [19-31].
  - 8.1.4 Calculate fitness values for the new positions of R, H and C.
  - 8.1.5 Determine the best position of new population.
  - 8.1.6 If old best position $<$ new best position:
    - 8.1.6.1 Update the best position to be new best.
  - 8.1.7 Else:
    - 8.1.7.1 Remain the old best as best solution.

9. Termination conditions:
- 9.1 If the maximum iterations ($t_{\max}$) is reached, go to step 10.
- 9.2 Else, If $\mathrm{Mod}(t, G) = 0$, Repeat steps (5 to 9).
- 9.3 Else, Repeat step (8 and 9).

10. End:
- 10.1 Select the individual position with highest fitness value as optimal solution.

$$Y^4 = S_4 \cdot \mathrm{R} \cdot (p^t_{b,j} - p^t_{i,j}) \tag{8}$$

$$Z^1 = Y^1 + Y^2 \tag{9}$$

$$Z^2 = Y^1 + Y^3 \tag{10}$$

$$Z^3 = Y^1 + Y^4 \tag{11}$$

$$Z^4 = Y^4 \tag{12}$$

$$P^k_{ij} = p^t_{i,j} + Z^k, \quad k = 1,2,3,4 \tag{13}$$

$$S_1 = \exp\left(\frac{ff_i - ff_{r1}}{|ff_i| + \varsigma}\right) \tag{14}$$

$$S_2 = \exp(ff_{r2} - ff_i) \tag{15}$$

$$S_3 = \exp(ff_n - ff_i) \tag{16}$$

$$S_4 = \exp(ff_{\text{best}} - ff_i) \tag{17}$$

Where $p^t_{n,j}$ is randomly chosen from the population N, and $p^t_{b,j}$ refers to the best position in the whole swarm. From the above equations, hens update their position not only based on information provided by the roosters but also based on information from other individuals in the population. The four candidate points are evaluated using:

$$(P^{t+1}_{i,j}, Z^{t+1}_i) = \max_k ff(P^k_{i,j}), \quad k = 1,2,3,4 \tag{18}$$

The final update point is determined by selecting the one with the highest fitness.

In the standard CSO algorithm, chicks exclusively learn from their respective mothers. Consequently, when a mother encounters a local minimum, her accompanying chicks inevitably get stuck in the same local minimum, thereby diminishing the optimization performance. To boost the algorithm's global optimization capacity and accelerate its convergence, chick position updating is refined by incorporating four reference points. These reference points allow the chicks to adjust their positions by following their respective mother, rooster, and the best individual, as shown below:

$$A^1 = T \cdot (p^t_{m,j} - p^t_{i,j}) \tag{19}$$

$$A^2 = C \cdot (p^t_{r,j} - p^t_{i,j}) \tag{20}$$

$$A^3 = S_5 \cdot \mathrm{Rand} \cdot (p^t_{m,j} - p^t_{i,j}) + S_6 \cdot \mathrm{Rand} \cdot (p^t_{r,j} - p^t_{i,j}) \tag{21}$$

$$A^4 = FL \cdot (p^t_{b,j} - p^t_{i,j}) \tag{22}$$

$$B^1 = A^1 \tag{23}$$

$$B^2 = A^1 + A^2 \tag{24}$$

$$B^3 = A^3 \tag{25}$$

$$B^4 = A^1 + A^4 \tag{26}$$

$$P^J_{ij} = w \cdot p^t_{i,j} + B^J, \quad J = 1,2,3,4 \tag{27}$$

$$S_5 = \exp\left(\frac{ff_i - ff_m}{|ff_i| + \varsigma}\right) \tag{28}$$

$$S_6 = \exp\left(\frac{ff_i - ff_r}{|ff_i| + \varsigma}\right) \tag{29}$$

Where T and C are cooperative coefficients in the range $[0,1]$, indicating the degree to which chicks learn from mother hens and roosters in their own group, respectively. From the above equations, the chicks update their positions not only based on their mother's information but also by learning from the group rooster and the best element from the population at the current iteration.

To enhance chicks' foraging, a nonlinear decreasing inertia weight w is computed as follows:

$$w = \frac{w^{\max} - w^{\min} \cdot t}{t_{\max}} \tag{30}$$

In this paper, $w^{\max} = 0.95$ and $w^{\min} = 0.4$. The chick's next position is selected from the four candidate points based on the highest fitness value:

$$(P^{t+1}_{i,j}, B^{t+1}_i) = \max_J ff(P^J_{i,j}), \quad J = 1,2,3,4 \tag{31}$$

VI. Fitness Critireon

Optimization and evolutionary algorithms frequently employ the concept of a fitness function, a mathematical framework that accepts a potential solution as input and generates a scalar value as output. This scalar value signifies the performance or quality of the candidate solution concerning the given optimization problem and reflects the individual's survival chances. This work aims to maximize the value of the fitness function [45].

A. Number of Zeros (NZ) Fitness Function

The first fitness criterion used in this work to assess candidate solutions is based on the count of zeros and ones in the candidate's plaintext. This function is adjusted to accommodate variations in plaintext sizes. In the English language, approximately 60% of characters translate into zeros in binary plaintext, although this value may vary with the text length [25]. The fitness value is computed using the following equation:

$$NZ(P_{iD}^{m}) = \frac{S_z}{N} \tag{32}$$

where P_{iD}^{m} refers to the candidate solution, S_z denotes the total number of zeros in the plaintext, and N represents the total number of bits in the candidate solution.

B. Index of Coincidence (IC) as Fitness Function

The Index of Coincidence (IC) is a technique used to measure how similar a frequency distribution is to a uniform distribution. IC can be calculated by comparing two texts side-by-side and counting how often identical letters appear in the same position. This count can be expressed either as a raw ratio or normalized by dividing by the expected count for a random model. The value of IC is obtained by multiplying the result by a normalizing coefficient, commonly set to 26 for English text [46]. The IC is given by:

$$IC = c\left(\frac{n_a}{N}\cdot\frac{n_a-1}{N-1}+\frac{n_b}{N}\cdot\frac{n_b-1}{N-1}+\cdots+\frac{n_z}{N}\cdot\frac{n_z-1}{N-1}\right) \tag{33}$$

Where c is the normalizing coefficient, n_a is the frequency of the letter 'a' in the text and N is length of the text. IC can also be calculated as mentioned in (34):

$$\mathrm{IC} = \sum_{i=1}^{m}\frac{n_i(n_i-1)}{N(N-1)} \tag{34}$$

C. Multi Objective Fitness Function (MOFF)

Using NZ and IC objective functions independently in the analysis of nonlinear stream ciphers has given rise to numerous challenges. This is primarily due to the statistical nature of these measures, which renders them imperfect. Consequently, the efficacy of these measures can be influenced by various factors, including the length of the ciphertext and the linguistic attributes of the underlying language.

This study gives a new multi objective function to address the challenges above. Utilizing a multi objective fitness function (MOFF) achieves an optimal balance and makes trade-offs between the fitness functions of NZ and IC.

The presented fitness function (MOFF) is grounded on Pareto dominance. Pareto dominance is a fundamental idea employed in multi-objective optimization, enabling the comparison and ranking of diverse solutions based on their performance across multiple objectives. The MOFF identifies potential solutions along the Pareto front, which consists of configurations that maximize the importance of IC and minimize the importance of NZ in the candidate plaintext [47]. The mathematical model of the proposed fitness function (MOFF) can be structured as a weighted sum of the two measures (NZ and IC) to reflect the relative importance of each metric as depicted in (35).

$$\mathrm{MOFF} = w_1\cdot f_1 + w_2\cdot f_2 \tag{35}$$

Where f_1 and f_2 are the IC and NZ respectively. The w_1 and w_2 have a range between $[0,1]$ and their sum equals one which represent weights of importance that control the trade-off between f_1 and f_2.

A comprehensive investigation was undertaken to ascertain the optimal objective function for employment in the cryptanalysis of nonlinear stream ciphers. The fitness function that shows a clear difference between plaintext and ciphertext is the principal measure for this work. Tables II and III presents the values of w_1 and w_2, as well as the impact of those values on the fitness function calculation when applied to the plaintext of 20 and 30 characters which is called MOFF(1) and fitness calculation of the ciphertext (Non plaintext) with the same lengths that produces by utilizing the Geffe generator system which is called MOFF(2).

Tables II and III demonstrated that there are issues with computing the fitness function value based on NZ, because there are some circumstances where the NZ value is good, but the text is not plain. For example, the NZ fitness function for plaintext with lengths of (LPT) 20 and 30 was 0.63 and 0.632, respectively, whereas the NZ fitness function for non-plaintext was 0.66 and 0.6429. Diff column refers to the difference between real plaintext (Plaintext) and ciphertext (Non Plaintext). Fitness criteria should be distinguished between the actual plaintext and the candidate plaintext that result from cryptanalysis processes. So, the Diff value must be equal to zero when the actual plaintext is equal to the candidate plaintext and must be as large as feasible when there is a difference between them. In tables above we suppose that the candidate plaintext is not the same of actual plaintext, so the Diff value must be large. From different experiences, this work found that the highest Diff is reached when the values of w_1 and w_2 0.9 and 0.1, respectively. This indicates that the importance of IC should be maximized, while the importance of NZ should be minimized to obtain a decent fitness function.

VII. Applying AMPCSO Algorithm to Attack Stream Cipher

The AMPCSO algorithm is the proposed enhancement of the traditional CSO algorithm, aimed at improving the exploration and exploitation phases of hens and chicks to efficiently reach the optimal solution, as described in Section V. The AMPCSO algorithm is applied to attack nonlinear stream cryptosystems, particularly those based on the Geffe generator, by identifying the secret initial values of combined LFSRs with different polynomial degrees.

The brute-force time complexity of breaking such systems is given by:

$$(2^{L_1}-1)\times(2^{L_2}-1)\times(2^{L_3}-1) \tag{36}$$

which is computationally infeasible to perform using modern

TABLE II.
CALCULATION OF THE MOFF FOR PLAINTEXT WITH SIZE 20 CHARACTERS BASED ON VARIANTS VALUES OF w_1 AND w_2

LPT	w_1	w_2	Plaintext			Ciphertext			Diff
			f_1	f_2	M(1)	f_1	f_2	M(2)	
20	0.1	0.9	0.066	0.63	0.574	0.0047	0.66	0.5944	0.0204
	0.2	0.8		0.518			0.5289		0.0109
	0.3	0.7		0.461			0.4634		0.0024
	0.4	0.6		0.405			0.3979		0.0071
	0.5	0.5		0.348			0.3323		0.0157
	0.6	0.4		0.292			0.2668		0.0252
	0.7	0.3		0.235			0.2013		0.0337
	0.8	0.2		0.179			0.1358		0.0432
	0.9	0.1		0.123			0.0728		0.0502

TABLE III.
CALCULATION OF THE MOFF FOR PLAINTEXT WITH SIZE 30 CHARACTERS BASED ON VARIANTS VALUES OF w_1 AND w_2

LPT	w_1	w_2	Plaintext			Non Plaintext			Diff
			f_1	f_2	MOFF(1)	f_1	f_2	MOFF(2)	
30	0.1	0.9	0.0655	0.632	0.5753	0.0017	0.6429	0.5787	0.0034
	0.2	0.8		0.5187			0.5146		0.0041
	0.3	0.7		0.462			0.4505		0.0115
	0.4	0.6		0.4054			0.3864		0.019
	0.5	0.5		0.3487			0.3223		0.0264
	0.6	0.4		0.2921			0.2581		0.034
	0.7	0.3		0.2354			0.194		0.0414
	0.8	0.2		0.1788			0.1299		0.0489
	0.9	0.1		0.1221			0.0658		0.0563

systems. In contrast, the AMPCSO algorithm is capable of recovering the secret initial values with minimal exploration, thereby achieving optimal timing for cryptanalysis.

The main steps of attacking a stream cipher using the AMPCSO algorithm are as follows:

1. Randomly initialize the population of N chickens. Each chicken in the population has its own position $p^t_{i,j}$, where the vector length of the positions equals the size of the secret key (seed).

2. Convert each chicken's position value to its binary representation using:
$$p^t_{i,j} = \begin{cases} 1, & \text{if } p^t_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (37)$$

3. Each chicken (candidate solution) represents a possible secret initial key. Insert each candidate key into the Geffe generator to produce a potential keystream.

4. Decrypt the ciphertext by XORing it with the potential keystream of each candidate key.

5. Evaluate each chicken in the population by computing the MOFF using Equation (35).

6. Rank the chickens in descending order (this work uses maximized fitness, as discussed previously) based on their fitness function values. Establish the hierarchical order to determine the number of roosters (RN), hens (HN), chicks (CN), and mother hens (MN).

7. Divide the population into subgroups, each containing at least one rooster, two hens, and several chicks.

8. Re-rank the entire population after a predefined period G to update the hierarchical structure.

9. **Evolving process:**

- Update roosters' secret initial keys $p_{R,j}^{t+1}$ using Equation (1).
- Update hens' secret initial keys $p_{H,j}^{t+1}$ using Equation (18).
- Update chicks' secret initial keys $p_{CN,j}^{t+1}$ using Equation (31).
- Update the best secret initial key $p_{b,j}^{t}$ in the entire population.

10. Repeat steps 2–9 until the actual plaintext is revealed or the maximum number of iterations is reached.

11. Selected the individual with the best MOFF as the optimal secret initial key in the population.

VIII. Experimental Results

The primary objective of this section is to evaluate the effectiveness of the CSO and AMPCSO algorithms in attacking nonlinear stream cipher systems based on the Geffe keystream generator using a ciphertext-only attack. The experimental results for the brute-force attack, CSO, and the enhanced CSO (AMPCSO) algorithm are presented in Tables V, VI, and VII, respectively. These results are based on the Mean Objective Fitness Function (MOFF) values.

Furthermore, the outcomes of previous research efforts are compared with those of the AMPCSO algorithm, as shown in Table IV . The cryptanalysis processes were implemented in MATLAB R2022b, and all experiments were conducted on a laptop equipped with a 10th Generation Intel Core i7 processor, 16 GB of RAM.

Table IV provides the actual fitness values for plaintexts of varying lengths.

Results from Tables V and VI showed that the CSO and AMPCSO algorithms are efficient methods to attack nonlinear stream cryptosystems. Where the brute force attack required about 10866.872 seconds to recover initial value with lengths 27 bits, while the CSO and AMPCSO algorithms required only 72.86 and 14.7 seconds respectively. Also, CSO algorithm required only 50 characters from ciphertext and 165 seconds to recover about 33 bits from 3LFSRs, whereas 20 characters and 79 seconds were enough for AMPCSO to achieve a successful attack. Table VII illustrates that the AMPCSO algorithm was able to recover keys up to 59 bits with appropriate consuming time and a minimum number of characters from the ciphertext. Table VIII compares the AMPCSO algorithm with the preview works in attacking the Geffe keystream generator. According to the above tables, our proposed system was an efficient algorithm to attack the Geffe keystream generator. AMPCSO has the best consuming time and requires fewer characters from cipher [10-200] to attack Geffe with total LFSRs up to 59 based on cipher-only attacks where the most previews work consumes more time and requires about 400 characters from the ciphertext. Fig. 2 illustrates the fitness values of CSO and AMPCSO with the keylengths and the comparison with the related works.

IX. Conclusion

This paper introduced the use of the CSO algorithm in the cryptanalysis process, which is considered the first attempt at using such an algorithm in the field of cryptanalysis. Where the CSO algorithm is applied to solve nonlinear stream cipher based on Geffe keystream generators. The traditional CSO was also developed to obtain an updated and valuable version of the cryptanalysis process called the AMPCSO algorithm. The AMPCSO algorithm attempts to prevent the hens and chicks from being trapped in local optima by enhancing the exploration step of hens and updating the learning of chicks by incorporating four reference points. To implement the cryptanalysis process, a new multi objective function was proposed that depends on the number of zeros and the decrypted message's coincidence index. This paper proved that the CSO and AMPCSO were efficient algorithms in the field of cryptanalysis. CSO could recover the keystream of the system up to 33 bits in an appropriate time and required only 50 characters from the ciphertext, according to Table IV. Results from Tables V and 6 showed that the AMPCSO algorithm required only ten characters to recover the initial key of Geffe with a length of up to 27 bits in time less than 15 seconds and required about 200 characters when the initial key is up to 59 bits, where the best-related work is required 400 characters with long consuming time. These simulation results showed that the AMPCSO algorithm is successful in finding the correct initial states of Geffe keystream generators with minimum information and optimal consuming time. The proposed algorithms CSO and AMPCSO can be used to cryptanalysis other keystream generators and block cryptosystems.

Acknowledgment

We thank Mustansiriyah University and the University of Kufa for their help with our research.

Conflict of Interest

The authors have no conflict of relevant interest to this article.

References

[1] C. Chris, "Review of history of cryptography and cryptanalysis by john dooley," *Cryptologia*, vol. 43, no. 6, pp. 536–538, 2019.

TABLE IV. AMOFF OF PLAINTEXT WITH DIFFERENT MTL VALUES

MTL	10	20	50	80	100	200
AMOFF	0.1096	0.1231	0.1227	0.1224	0.1223	0.1227

TABLE V. THE CONSUMING TIMES OF BRUTE FORCE ATTACK TO BREAK TWO GEFFE SYSTEMS WITH DIFFERENT MTL VALUES

TLR	[3, 5, 7]			[7, 9, 11]		
MTL	10	20	50	10	20	50
BCT	1.4028	1.1282	1.2271	10866.872	5379.371	4577.341

TABLE VI.
APPLYING CSO AND AMPCSO TO SOLVE GEFFE STREAM CIPHERS BASED ON 3 LFSRS WITH TLR UP TO 33 BITS AND MTL [10–50] CHARACTERS

TLR	**MI/N**	**MTL**	**CSO**			**AMPCSO**		
			BMOFF	BCT	BIter	BMOFF	BCT	BIter
[3,5,7]	30/30	10	0.1096	0.41	5	0.1096	0.29	3
		20	0.1230	0.32	3	0.1230	0.069	1
		50	0.1226	0.26	2	0.1226	0.116	2
[7,9,11]	300/50	10	0.1061	72.86	215	0.1096	14.7	78
		20	0.1230	46.03	109	0.1230	7.39	37
		50	0.1226	36.93	92	0.1226	5.09	34
[7,11,15]	500/50	10	0.0841	159.47	488	0.0961	131.01	401
		20	0.1069	140.92	419	0.1230	79.28	283
		50	0.1226	165.78	492	0.1226	128.61	379

TABLE VII.
APPLYING CSO AND AMPCSO TO ATTACK GEFFE STREAM CIPHERS BASED ON 3 LFSRS WITH TLR UP TO 61 BITS AND MTL [10–200]

TLR	**MI/N**	**MTL**	**CSO**				**AMPCSO**			
			BMOFF	BCT	BIter	RRK%	BMOFF	BCT	BIter	RRK%
[9,11,19]	800/150	10	0.0692	415.62	798	63.13	0.091	392.01	753	83
		50	0.0951	410.83	783	77.5	0.1125	328.92	630	91.4
		80	0.0986	412.93	789	80.5	0.1224	369.39	712	100
		100	0.1004	406.88	781	82.12	0.12226	212.71	415	100
		200	0.1095	395.31	759	89.24	0.1227	204.36	393	100
[13,17,19]	1200/150	10	0.0531	1116.68	1158	47.53	0.071	1012.3	1050	64.78
		50	0.067	1146.64	1187	54.6	0.0830	952.9	985	67.67
		80	0.069	1061.82	1098	56.37	0.0985	1113.21	1012	80.4
		100	0.073	1151.39	1194	59.73	0.12226	604.09	631	100
		200	0.0813	1138.86	1181	66.24	0.1227	545.26	592	100
[7,23,23]	1500/250	10	0.0501	1821.32	1315	45.71	0.052	1980.3	1430	47.44
		50	0.0594	2001.38	1445	48.4	0.0808	1928.9	1412	65.9
		80	0.0649	2036.01	1470	53.02	0.0867	1823.12	1394	70.83
		100	0.0701	1923.2	1390	57.36	0.0975	1299.01	1010	79.78
		200	0.0789	2043.31	1496	64.3	0.1227	1014.29	915	100

[2] N. Munir, M. Khan, T. Shah, A. Alanazi, and I. Hussain, "Cryptanalysis of nonlinear confusion component based encryption algorithm," *Integration*, vol. 79, pp. 41–47, 2021.

TABLE VIII.
COMPARISON BETWEEN THE PROPOSED SYSTEM AND THE RELATED WORKS IN ATTACKING GEFFE KEYSTREAM GENERATORS

Ref	TLR	Related Works			AMPCSO		
		BCT	BIter	MTL	BCT	BIter	MTL
[32]	15	0.55	300	10	0.29	3	10
[21]	23	481	Na	400	5.81	18	10
[27]	27	1.6	16	400	14.7	78	10
[25]	–	0.70	100	40	–	–	–
[21]	29	877	Na	400	5.41	34	30
	39	3693	Na	400	369.39	712	80
[27]	49	95.2	376	400	604.09	631	100
	53	1168.8	4805	400	1014.29	915	200
	59	3925.2	25820	400	2295.33	1928	200

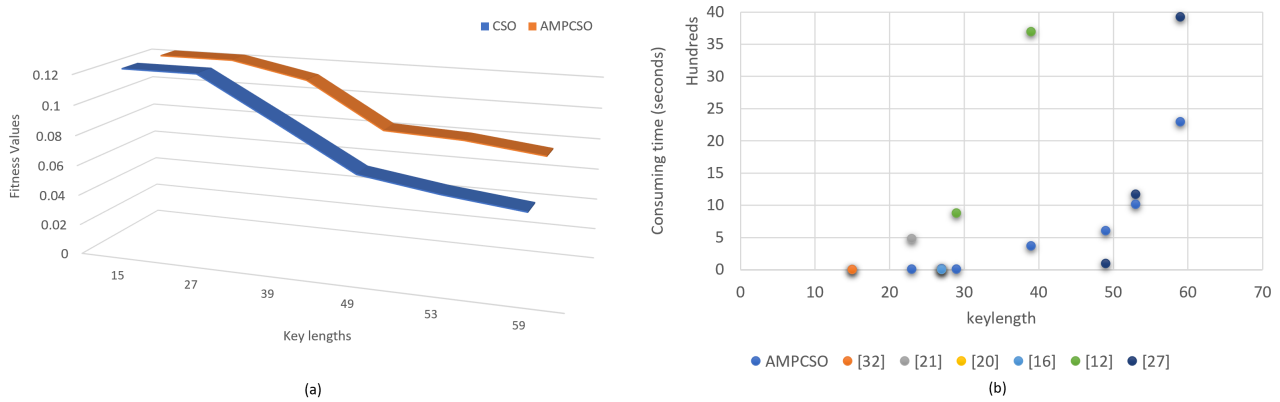

Fig. 3.
Attacking Geffe Stream: (a) Fitness Values of CSO And AMPCSO Algorithms Based On 50 Characters and (b) The Consuming Time Of Proposed System and Previews Works

[3] M. Ahmed, A. Shibeeb, and A. Mohammed, "Solve polyalphabetic cipher based on intelligent system," in *Proc. of International Conf. On Communication & Information Technology (ICICT)*, (Basrah, Iraq), pp. 290–296, 2021.

[4] Y. Feng, H. Wang, C. Chang, H. Lu, F. Yang, and C. Wang, "A novel nonlinear pseudorandom sequence generator for the fractal function," *Fractal and Fractional*, vol. 6, no. 10, p. 589, 2022.

[5] S. Gupta, P. Singh, N. Shrotriya, and T. Baweja, "Lfsr next bit prediction through deep learning," *Journal of Informatics Electrical and Electronics Engineering (JIEEE)*, vol. 2, no. 2, pp. 1–9, 2021.

[6] S. Karthika and K. Singh, "Cryptanalysis of stream cipher lizard using division property and milp based cube attack," *Discrete Applied Mathematics*, vol. 325, pp. 63–78, 2023.

[7] A. K. Shibeeb and M. H. Ahmed, "Use of a new approach to automated break transposition cipher system," *IOP Conference Series Materials Science and Engineering*, vol. 518, p. 052020, May 2019.

[8] G. Wang, D. Cheng, D. Xia, and H. Jiang, "Swarm intelligence research: From bio-inspired single-population swarm intelligence to human-machine hybrid swarm intelligence," *Machine Intelligence Research*, vol. 20, no. 1, pp. 121–144, 2023.

[9] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: chicken swarm optimization," in *Proc. of 5th International Conf of Advances in Swarm Intelligence (ICSI 2014)*, (Hefei, China), pp. 86–94, Springer, 2014.

[10] Z. Wang, C. Qin, B. Wan, W. W. Song, and G. Yang, "An adaptive fuzzy chicken swarm optimization algorithm," *Mathematical Problems in Engineering*, vol. 2021, p. 1–17, Mar. 2021.

[11] D. Wu, F. Kong, W. Gao, Y. Shen, and Z. Ji, "Improved chicken swarm optimization," in *Proc. of International Conf of IEEE on Cyber Technology in Automation Control and Intelligent Systems (CYBER)*, (Shenyang, China), pp. 681–686, 2015.

[12] N. Irsalinda, A. Thobirin, and W. Wijayanti, "Chicken swarm as a multi step algorithm for global optimization," *International Journal of Engineering Science Invention*, vol. 6, no. 1, pp. 8–14, 2017.

[13] M. Lin, Y. Zhong, J. Lin, and X. Lin, "Enhanced chicken swarm optimisation for function optimisation problem," *International Journal of Wireless and Mobile Computing*, vol. 15, no. 3, pp. 258–269, 2018.

[14] D. He, G. Lu, and Y. Yang, "Research on optimization of train energy-saving based on improved chicken swarm optimization," *IEEE Access*, vol. 7, pp. 121675–121684, 2019.

[15] J. Wang, F. Zhang, H. Liu, J. Ding, and C. Gao, "Interruptible load scheduling model based on an improved chicken swarm optimization algorithm," *Journal of Power and Energy Systems*, vol. 7, no. 2, pp. 232–240, 2021.

[16] X. Liang, D. Kou, and L. Wen, "An improved chicken swarm optimization algorithm and its application in robot path planning," *IEEE Access*, vol. 8, pp. 49543–49550, 2020.

[17] M. Gamal, A. El-Sawy, and A. AbuEl-Atta, "Hybrid algorithm based on chicken swarm optimization and genetic algorithm for text summarization," *International Journal of Intelligent Engineering and Systems*, vol. 14, p. 319–331, May 2021.

[18] Y. Gu, H. Lu, L. Xiang, and W. Shen, "Adaptive simplified chicken swarm optimization based on inverted s-shaped inertia weight," *Chinese Journal of Electronics*, vol. 31, p. 367–386, Mar. 2022.

[19] L. Liang, L. Wang, and M. Ma, "An improved chicken swarm optimization algorithm for solving multimodal optimization problems," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[20] L. Liang, L. Wang, and M. Ma, "An adaptive dual-population collaborative chicken swarm optimization algorithm for high-dimensional optimization," *Biomimetics*, vol. 8, no. 2, p. 210, 2023.

[21] M. Din, A. Bhateja, and R. Ratan, "Cryptanalysis of geffe generator using genetic algorithm," in *Proc. of International Conf. On Soft Computing for Problem Solving: SocProS 2013*, vol. 259, (New Delhi), pp. 509–515, Springer, 2014.

[22] I. Polak and M. Boryczka, "Genetic algorithm in stream cipher cryptanalysis," in *Proc. of International Conf. On Computational Collective Intelligence, ICCCI 2015*, vol. 9330, (Madrid, Spain), pp. 149–158, 2015.

[23] K. Pommerening, "Cryptanalysis of nonlinear feedback shift registers," *Cryptologia*, vol. 40, no. 4, pp. 303–315, 2016.

[24] M. Din, S. Pal, S. Muttoo, and A. Jain, "Applying cuckoo search for analysis of lfsr based cryptosystem," *Perspectives in Science*, vol. 8, pp. 435–439, 2016.

[25] S. Al-Ageelee and R. Kadhum, "Cryptanalysis of nonlinear stream cipher cryptosystem based on improved particle swarm optimization," *International Journal of applied information systems*, vol. 19, no. 1, pp. 78–84, 2017.

[26] S. Sadkhan and B. Yaseen, "A dna-sticker algorithm for cryptanalysis lfsrs and nlfsrs based stream cipher," in *Proc. of International Conf. On Advanced Science and Engineering (ICOASE)*, (Duhok, Iraq), pp. 301–305, 2018.

[27] M. Din, S. K. Pal, and S. K. Muttoo, *Applying PSO based technique for analysis of GEFFE Generator Cryptosystem*, p. 741–749. Aug. 2018.

[28] I. Polak and M. Boryczka, "Tabu cryptanalysis of vmpc stream cipher," *Tatra Mountains Mathematical Publications*, vol. 73, no. 1, pp. 145–162, 2019.

[29] I. Polak and M. Boryczka, "Tabu search in revealing the internal state of rc4+ cipher," *Applied Soft Computing*, vol. 77, pp. 509–519, 2019.

[30] G. Mishra, I. Gupta, S. Murthy, and S. Pal, "Deep learning based cryptanalysis of stream ciphers," *Defence Science Journal*, vol. 71, no. 4, pp. 499–506, 2021.

[31] R. Rizk-Allah, H. Abdulkader, S. Elatif, W. Elkilani, E. Al Maghayreh, H. Dhahri, and A. Mahmood, "A novel binary hybrid pso-eo algorithm for cryptanalysis of internal state of rc4 cipher," *Sensors*, vol. 22, no. 10, p. 3844, 2022.

[32] R. Jawad, "Proposed hybrid technique in cryptanalysis of cryptosystem based on pso and sa," *Iraqi Journal of Science*, vol. 63, no. 10, pp. 4547–4558, 2022.

[33] A. Ayvaz, "An improved chicken swarm optimization algorithm for extracting the optimal parameters of proton exchange membrane fuel cells," *International Journal of Energy Research*, vol. 46, p. 15081–15098, June 2022.

[34] Y. Zhang, L. Wang, and J. Zhao, "Pecso: An improved chicken swarm optimization algorithm with performance-enhanced strategy and its application," *Biomimetics*, vol. 8, no. 4, 2023.

[35] Y. Gu, H. Lu, L. Xiang, and W. Shen, "Adaptive simplified chicken swarm optimization based on inverted s-shaped inertia weight," *Chinese Journal of Electronics*, vol. 31, no. 2, pp. 367–386, 2022.

[36] Y. Wang, C. Sui, C. Liu, J. Sun, and Y. Wang, "Chicken swarm optimization with an enhanced exploration-exploitation tradeoff and its application," *Soft Computing*, vol. 27, no. 12, pp. 8013–8028, 2023.

[37] Z. Wang, W. Zhang, Y. Guo, M. Han, B. Wan, and S. Liang, "A multi-objective chicken swarm optimization algorithm based on dual external archive with various elites," *Applied Soft Computing*, vol. 133, 2023.

[38] W. Osamy, A. El-Sawy, and A. Salim, "Csoca: Chicken swarm optimization based clustering algorithm for wireless sensor networks," *IEEE Access*, vol. 8, pp. 60676–60688, 2020.

[39] X. Zhou, Z. Gao, and X. Yi, "An improved chicken swarm optimization algorithm based on adaptive mutation learning strategy," *Journal of Computers*, vol. 33, no. 6, pp. 1–19, 2022.

[40] R. Al-Amri, D. Hamood, and A. Farhan, "Theoretical background of cryptography," *Mesopotamian Journal of CyberSecurity*, pp. 7–15, 2023.

[41] S. Wadhawan, "A study on cryptography," *International Journal of Engineering and Management Research*, no. 2, pp. 99–103, 2023.

[42] A. Babu and B. Anand, "Modified dynamic current mode logic based lfsr for low power applications," *Microprocessors and Microsystems*, vol. 72, p. 102945, 2020.

[43] B. Ali, M. Zaite, and A. Al-Hashimi, "Design and implementation of a key generator-based stream cipher for securing text data," *Journal of Engineering Science and Technology*, vol. 14, no. 6, pp. 3372–3386, 2019.

[44] A. Naser and F. Majeed, "Constructing of analysis mathematical model for stream cipher cryptosystems," *Iraqi Journal of Science*, vol. 58, no. 2A, pp. 707–715, 2017.

[45] H. Maier, S. Razavi, Z. Kapelan, L. Matott, J. Kasprzyk, and B. Tolson, "Introductory overview: Optimization using evolutionary algorithms and other metaheuristics," *Environmental modelling & software*, vol. 114, pp. 195–213, 2019.

[46] V. Vera and C. Ávila, "Graphemic-phonetic diachronic linguistic invariance of the frequency and of the index of coincidence as cryptanalytic tools," *Plos one*, vol. 14, no. 3, 2019.

[47] S. Zhu, L. Xu, E. Goodman, K. Deb, and Z. Lu, "A general framework for enhancing relaxed pareto dominance methods in evolutionary many-objective optimization," *Natural Computing*, vol. 22, no. 2, pp. 287–313, 2023.