I. J. Computer Network and Information Security, **2025, 3, 54-71**
Published Online on June 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2025.03.04

On Cryptanalysis of 3-DES using Nature-Inspired Algorithms

Subinoy Sikdar*
Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah, 711103, India
E-mail: subinoysikdar.jume2019@gmail.com
ORCID iD: https://orcid.org/0000-0002-9259-1543
*Corresponding author

Sagnik Dutta
Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah, 711103, India
E-mail: sagnikofficial001@gmail.com
ORCID iD: https://orcid.org/0009-0004-6717-9524

Malay Kule
Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah, 711103, India
E-mail: malay.kule@gmail.com
ORCID iD: https://orcid.org/0000-0002-0180-7610

Received: 30 November 2023; Revised: 08 March 2024; Accepted: 28 March 2025; Published: 08 June 2025

Abstract: This paper presents a novel cryptanalysis method of DES (2-DES and 3-DES) using nature-inspired algorithms; namely Cuckoo Search Algorithm and Grey Wolf Optimization Algorithm. We have shown the loophole of 2-DES and 3-DES encryption systems and discovered the vulnerabilities by some simple mathematical calculations. The Meet-In-The-Middle approach can be executed on 2-DES along with Known Plaintext Attack, Chosen Plaintext Attack, and Chosen Ciphertext Attack. The valid key pairs along with the original key pairs can successfully be recovered by this attack algorithm. But in the Ciphertext Only Attack, the Meet-In-The-Middle approach fails to recover the plaintext as well as the valid key pairs both for 2-DES and 3-DES. To overcome this problem, we have proposed a novel cryptanalysis method of 3-DES with Ciphertext Only Attack using Cuckoo Search Algorithm and Grey Wolf Optimization Algorithm (GWO). We have developed a suitable fitness function, accelerating the algorithm toward the optimal solution. This paper shows how CSA and GWO can break a 3-DES cryptosystem using a Ciphertext Only Attack. This proposed cryptanalysis method can also be applied to any round of DES.

Index Terms: Cryptanalysis, DES, Meet-In-The-Middle, Cuckoo Search Algorithm, Grey Wolf Optimization, Fitness Function.

1. Introduction

The modern age is the age of computers. In present days, every system is being computerized such as shopping, banking, food delivery, ticket booking, courier service, different management systems, transportation services, social media networking, e-learning courses, and many more. After COVID-19, the workplace is also taking place in online platforms. With the increasing use of online platforms, the credentials and confidential information of different users are propagated over the internet. This secret user information is to be protected while propagating over the network. Several network adversaries may be present in the unreliable communication channels. Network adversaries can have unauthorized access to this secret user information. Security and privacy are the greatest concern of recent days. In this scenario, cryptography [1] comes into the picture. Cryptography ensures the security and privacy of user information over an unreliable network. Cryptography transforms the original message into an unreadable format with the help of a key(s) value. Thus, the messages appear meaningless to an unauthorized user even though the message is intercepted by that unauthorized third party. Cryptography consists of two main concepts – encryption and decryption. Encoding the

messages into unreadable format is called encryption and decoding the encrypted messages into original messages is called decryption. There are several cryptosystems [2-4] exist in present days such as SPN, DES, AES, etc. Rather, Cryptanalysis is detailed practice and study of the cryptosystems in such a way so that the weaknesses, drawbacks, and backdoors of the cryptosystem can be pointed out and fix all these loopholes of the cryptosystem in apropos to get a superior secure cryptosystem. In this paper, we have shown the vulnerability of a 2-DES and 3-DES cryptosystem.

DES (Data Encryption Standard) is one of the popular block ciphers in modern days. Generally, DES [5-7] is 16 round block cipher. To increase the strength of DES, the round key operations are repeated over the DES cipher. There are 16 rounds (DES), 32 rounds (2-DES) and 48 rounds (3-DES) DES are available in the present days. 2-DES is an encryption technique, where two instances of DES are applied on the same plaintext with two different keys to generate the ciphertext. After the first round of encryption, it produces an intermediate state of plaintext and ciphertext whereas the complete ciphertext is generated after the second round of execution. In the case of 3-DES, three different keys are used to obtain the ciphertext from the plaintext. Fig. 1 and 2 show a typical 2-DES and 3-DES architecture respectively [8-11].

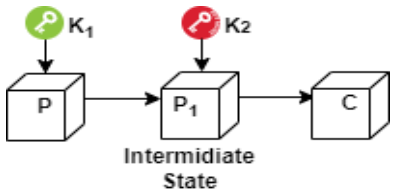

Fig.1. 2-DES architecture

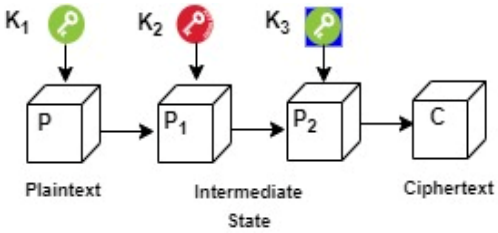

Fig.2. 3-DES architecture

Here, in the first cryptanalysis module, there exists cryptanalysis of 2-DES (32 rounds) using Meet-In-The-Middle Approach (MIM). A bidirectional approach is encountered using Known Plaintext Attack on 2-DES. This cryptanalysis method can successfully generate all the valid key pairs; and hence recover the plaintext from the ciphertext once the key pairs are determined. In the second cryptanalysis module, our proposed Cuckoo Search and Grey Wolf Optimization-based attack algorithm has been discussed. By these algorithms, the 3-DES cryptosystem has been broken successfully only using Ciphertext-Only-Attack, which is the hardest attack among all other attack methods.

In support of the above-mentioned context, we have proposed a new Ciphertext Only Attack on 3-DES using the Cuckoo Search Algorithm and Grey Wolf Optimization Algorithm, which can also be applied to any round DES. In this proposed method, the plaintext can directly be recovered from the ciphertext. In this paper, our contributions can be listed in the following way –

- Design a suitable fitness function for the Cuckoo Search Algorithm and Grey Wolf Optimization algorithm.
- Proposed Ciphertext Only Attack using Cuckoo Search Algorithm and Grey Wolf Optimization Algorithm on DES.
- Analysis of the experimental results from the proposed cryptanalysis algorithms.

This paper is structured in five different sections. Section 1 introduces the paper and section 2 briefly mentions the preliminaries and previous research works. Section 3 elaborates on the proposed method of cryptanalysis of DES using the Cuckoo Search Algorithm and Grey Wolf Optimization Algorithm as well as the abstract implementation details of the proposed attack algorithms. Section 4 shows the experimental outcomes of the proposed cryptanalysis algorithms and analyses the experimental results. Finally, section 5 concludes the paper.

2. Background and Literature Review

This section covers necessary studies related to our research works. This section is divided into three subsections. The first two subsections present the Cuckoo Search and Grey Wolf Optimization algorithm and the last subsection records a few important research works on DES cryptanalysis in the recent past.

2.1. Cuckoo Search Algorithm

The pseudo-code for Cuckoo Search Algorithm [12,13] is given below. The Cuckoo Search Algorithm is built on three assumptions. They are mentioned as follows –

- The cuckoo species picks a nest arbitrarily and drops its egg one at a time in the nest.
- Only the nests containing better standard eggs will be conducted to the next descendant.
- The obtainable host bird nests are fixed and the possibility of detection of the cuckoo egg by the host bird is $P_a \in (0,1)$. Under these conditions, either the host bird will decline the egg or discard the nest and fabricate an entirely different nest.

Algorithm 1: Cuckoo Search Algorithm

start
Objective function Of (p), $p = (p_1, ..., p_d)^T$
Create an initial community of
m host abodes h_c (c = 1, 2, ..., m)
while (t < Max_Iteration) or (not satisfy termination condition)
Pick a bird arbitrarily via Levy flights to assess its merit/fitness M_c
Opt an abode from m (say, d) arbitrarily
if ($M_c > M_d$),
Change d by newly generated solutions;
end
A snatch (L_a) of less quality abodes
are devoid of merit and new abodes are created;
Retain the ideal solutions
(or abodes with ideal solutions);
Rank the solutions and find the best
end while
Postprocess results and visualization
End

2.2. Grey Wolf Optimization Algorithm

The step-by-step process for the Grey Wolf Optimization Algorithm [14,15] is given below.

Algorithm 2: Grey Wolf Optimization Algorithm

initialize Grey Wolf population S_i(i = 1,2,3,…….,n)
initialize a, A, and C
calculate the fitness of each wolf
S_α = best fit wolf, S_β = 2nd best fit wolf, S_δ = 3rd best fit wolf
while (t< maximum iteration or S_α = Target)
    for each wolf
        update the position of the current wolf
    end for
    update a, A, and C
    calculate the fitness of all wolves
    update S_α, S_β, and S_δ
    t = t+1
end while
return S_α

2.3. Literature Review

This subsection presents different research works carried out on DES cryptanalysis in the recent past worldwide. Here, some of the important research works are mentioned below.

Hicham Grari et. al. [16] proposed a cryptanalysis method of simplified DES using the Ant Colony Optimization technique. They aimed to recover the secret key along with the Known Plaintext Attack. The secret key is a 10-bit binary string. The proposed algorithm works as discussed here. There is a start node at the very first position and an end node at the end. In between the start node and end node, there exist 10 nodes in the intermediate position in two different levels namely top level and bottom level. The upper-level node values are 1 and the lower-level node values are 0. Each ant starts its journey from the start node and ends in the end node. After the journey, each ant gives a 10-bit binary string (secret key) depending upon the nodes it visited. The ant moves based on a greedy probabilistic value P(i,j) across the

nodes. This probabilistic value depends upon two factors such as the pheromone amount between edge $\tau(i,j)$ and a heuristic value $\rho(i,j)$. There are two parameters α and β control the factors pheromone and heuristic value.

Aboul Ella Hassanien et. al. [17] presented a known plaintext attack on DES-16 using Particle Swarm Optimization. They stored all the ciphertext-plaintext pairs generated by the secret key. Now they choose the optimal plaintexts according to the proposed fitness function and the corresponding ciphertexts. From these pairs, they found out the difference between the plaintext and ciphertext to understand how the ciphertexts are generated. Eight best sub keys were found out which further helps to discover bits of the root key. PSO algorithm was employed to find out the optimal plaintexts. The second algorithm finds out the best eight sub-key differences. It takes a set of plaintext/ciphertext pairs and finds out the difference to determine the best eight sub-key differences.

Ritwiz Kamal et. al. [18] proposed a cryptanalysis algorithm for simplified DES using Cuckoo Search Algorithm, Firefly Algorithm, and Black Hole Optimization Algorithm. First of all, a random population of candidate keys is generated. These keys are used to decrypt the ciphertext and the fitness of each key is evaluated based on the decrypted output. Comparisons are made between the fitness of the decrypted solution and that of a randomly chosen candidate. If the decrypted solution's fitness is better, it replaces the existing solution. A fraction of poor-quality solutions are discarded, which ensures that only the best solutions are considered. This process is continued in an iterative loop until the termination condition is met. In brief, the decryption process first involves converting the ciphertext into binary, which is then used to attempt decryption with candidate keys. The solutions are continuously updated and optimized through iterations and the best solution eventually leads to recovery of the plaintext. This process leverages optimization algorithms like the Firefly Algorithm and Black Hole Optimization to efficiently recover the plaintext from the encrypted data. From the experimental results, they concluded that five different nature-inspired algorithms (Cuckoo Search Algorithm, Firefly Algorithm, Memetic Algorithm, Black Hole Optimization, Genetic Algorithm) can be ranked in the following way in the case of Simplified DES. The rank is CSA>FA>MA>BHO>GA.

Vikas Tiwari et. al. [19] proposed a differential cryptanalysis method for 8-round DES. The primary objective of this research work was to find out the differences between the ciphertext and plaintext, which further helps in recovering the secret keys. They showed the cryptanalysis of DES starting from 3 round DES to 8 round DES. The proposed method describes cryptanalysis of DES encryption using a probability characteristic, filtering to differentiate right and wrong pairs and extracting possible key bits for each S-Box, ultimately leading to the discovery of the encryption key.

Seeven Amic et. al. [20] proposed a cryptanalysis method of DES using Binary Firefly Algorithm along with Chosen Plaintext Attack. They aimed to determine the maximum number of bits that align with the original key by using Song's method. The experiment was run 100 times with 1000 generations per run. Each run generates a solution and the solution is considered to be optimal if the fitness meets a specified minimum fitness value (λ). So, after several runs, multiple solutions can be generated. A threshold value χ is set to determine the percentage of ones and zeros in each bit position within the optimal keys. By using the threshold value χ, the experiment infers the value of each bit. If the inferred bits align with actual bits of the true key, the overall fitness of the key improves in subsequent iterations. If not, subsequent solutions might degrade into a deceptive fitness landscape.

Cihangir Tezcan [21] proposed a cryptanalysis method using a distributed multiple GPU setting on different lightweight block ciphers such as DES, 3-DES, and PRESENT. This paper mainly focuses on brute-force attacks on these ciphers. The work presents CUDA-based optimizations using table-based techniques. These optimizations aim to offer software implementation without relying on bit operations. The proposed architecture achieved significant key search rates of 3.87 billion and 1.89 billion key searches per second for DES/3-DES and PRESENT respectively on an RTX 3070 GPU. These results outperform the performance of FPGA clusters like COPACOBANA in terms of cost-effectiveness, concluding that brute force search on short keys might be feasible without specialized devices.

3. Proposed Method of Cryptanalysis of DES using Cuckoo Search Algorithm and Grey Wolf Optimization Algorithm

In this section, two standard cryptanalysis methods such as Known Plaintext Attack and Ciphertext Only Attack have been described for the cryptanalysis of DES. Both the attack algorithms have been discussed in the following subsections.

3.1. Known Plaintext Attack using Meet in the Middle Approach Revisited

In this module, we have described how 2-DES becomes vulnerable to Meet-In-The-Middle Approach.

A. Prerequisite for Cryptanalysis

We have considered the plaintext, ciphertext, and key space as follows-

$$P = C = K = \{0,1\}^{64}$$

Plaintext: $P_1 = 64$ bit binary stream, $P_2 = 64$ bit binary stream
Secret Key: $K_1 = 64$ bit binary stream , $K_2 = 64$ bit binary stream; Secret Key Pairs = $<K_1, K_2>$
Ciphertext Generation:

$$P_1 \oplus K_1 = P_1'$$

$$P_1' \oplus K_2 = C_1$$

$$P_2 \oplus K_1 = P_2'$$

$$P_2' \oplus K_2 = C_2$$

Old Plaintext-Ciphertext pair = $<P_1, C_1>$; Current Plaintext-Ciphertext pair = $<P_2, C_2>$
Here, for the cryptanalysis work, we are making some assumptions. The assumptions are as follows –

- We don't know the secret key pairs i.e., $<K_1, K_2>$ pairs are unrevealed. We aim to find out the secret key pairs.
- As a Known Plaintext Attack (KPA) is performed, it is assumed that we have a previous $<P_i, C_i>$ pair. Here, the $<P_1, C_1>$ pair is known to us.
- As the ciphertexts are public, the current ciphertext will be obtained from the channel. Here, C_2 is considered as the current ciphertext.

So, finally, $[<P_1, C_1>, C_2]$ is known to us. We are supposed to find out the secret key pair $<K_1, K_2>$ and then the current plaintext P_2.

B. Meet in the Middle Approach Revisited

The intermediate state of 2-DES is the loophole for 2-DES. One can apply execution with all possible permutations of secret keys by trial-and-error method from both the direction of plaintext and ciphertext and it will meet in the same intermediate state for such a valid arrangement of secret keys. In the previous module, we have mentioned how the encryption occurred. Followed by the encryption rule we can easily reach the following decision, which is nothing but the Meet-In-The-Middle Approach.

$$P_1 \oplus K_1 = P_1' \quad (1)$$

$$C_1 \oplus K_2 = P_1' \quad (2)$$

From the above two equations (1) and (2) we can deduce-

$$P_1 \oplus K_1 = C_1 \oplus K_2 \quad (3)$$

In the above rule, for which combinations of $<K_1, K_2>$ pair, P_1' is equal in both the equation (1) and (2); that key pair turns out to be the valid key pair for encryption as well as decryption. Because of this equivalence in the intermediate state, 2-DES turned out to be breakable. Fig.3 shows the Meet-In-The-Middle Approach principle.

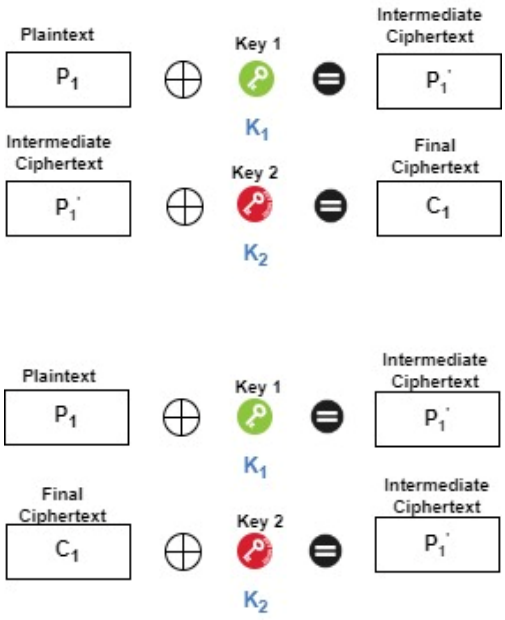

Fig.3. Meet in the middle approach principle

C. Generation of Valid Key Pairs

Old Plaintext-Ciphertext pair = $<P_1, C_1>$.
Current Ciphertext = C_2
Pair of all possible candidate keys $<K1, K2> \in$ Set of all possible keys $\{K = K_1, K_2, K_3, \ldots\ldots\ldots\ldots, K_{2n}\}$
K_i = 64 bit long binary string; $(I = 1, 2, 3, \ldots\ldots, 2^n)$
$K[2^n]$ = Collection of all possible candidate keys.

Meet-In-The-Middle Approach Algorithm

Algorithm 3: Meet-In-The-Middle Approach

C. Create $P[2^n]$, $C[2^n]$.

2. Consider P_1.
3. for i=1 to 2^n
4. $P[i] = P_1 \oplus K_i$
5. i++
6. end for

7. Consider C_1
8. for i=1 to 2^n
9. $C[i] = C_1 \oplus K_i$
10. i++
11. end for

12. for i=1 to 2^n
13. for j=1 to 2^n
14. if $(P[i] == C[j])$
15. Make $<K_i, K_j>$ a valid key pair.
16. j++
17. end for
18. i++
19. end for

We have started the cryptanalysis work with all possible key pairs from both the plaintext as well as ciphertext end and whenever we get a match in the intermediate plaintext; that key pairs become the valid key pair. Depending on the same intermediate plaintext, 2^n valid key pairs will be found.

Set of all valid pair = $\{<K_i, K_j>, <K_i, K_j>, \ldots\ldots\ldots\ldots 2^n\}$; I, j ∈ 1 to 2^n; $K_i \rightarrow$ Key 1 and $K_j \rightarrow$ Key 2

From the above set, we can pick any valid key pair, which can successfully recover the current plaintext from the current ciphertext. The plaintext restoration has been shown in the next module.

D. Recovery of Plaintext

The current ciphertext = C_2.

We can reconstruct the plaintext from the given ciphertext with any valid key pair. Fig. 4 exhibits the decryption of 2-DES below.

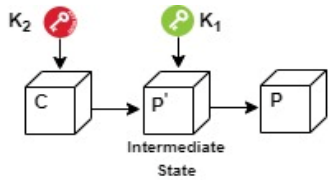

Fig.4. 2-DES ciphertext decryption

Let's consider the first key pair = $<K_1, K_2>$

$$C_2 = 111$$

$$C_2 \oplus K_2 = P_2'$$

$$P_2' \oplus K_1 = P_2 \text{ [plaintext recovered]}$$

Like the above, we can pick any key pair from the valid key pair set and it will work fine to break the 2-DES cryptosystem.

3.2. Organizing a Simplified 3-DES Cryptosystem

In this module, we have shown the construction of a simplified 3-DES. Construction of the DES was made abstractly. Our proposed cryptanalysis algorithms can be applied to the triple DES as well as any round of DES. DES is differentiated with applied keys. In the case of double DES, two secret keys are applied to the plaintext, whereas in the case of triple DES three secret is applied to the plaintext. Here the operational keys are kept abstract. We can consider a single resulting key in combination with two or three separate keys by equation (4). In this experiment, three 8-bit keys have been considered as the secret keys. These keys are applied to the plaintext and produce the corresponding ciphertext using equation (5).

$$K = K_1 \oplus K_2 \oplus K_3 \tag{4}$$

$$C = P \oplus K \tag{5}$$

3.3. Proposed Ciphertext-Only Attack using Cuckoo Search Algorithm

In the previous module, cryptanalysis of the 2-DES cryptosystem has been shown using the MIM approach along with Known Plaintext Attack. This method fails when only the ciphertext is available for cryptanalysis i.e., in a Ciphertext Only Attack, the MIM approach is unable to recover the plaintext as well as the valid key pairs. To overcome this situation, we have proposed a Cuckoo Search-based attack algorithm that can break a 2-DES, 3-DES, and any round of DES Cryptosystem only from the ciphertext.

The proposed algorithm has been characterized by mainly five initial functionalities such as population generation, fitness evaluation, levy flight, mutation, and perturbation method. All these functions of the attack algorithm have been discussed here.

A. Population Generation

Initially, 20 candidate solutions (eggs) are generated randomly. Here each populated solution represents a plaintext of 8-bit long binary string.

B. Fitness Function

Here the fitness function is designed in such a way so that it can find out the fitness of a plaintext pair. The highest fitness value for a best plaintext pair can be 1 hence it implies that pair can be a probable solution. In a ciphertext-only attack, only the ciphertext is available for cryptanalysis. According to differential cryptanalysis, we know the following law –

$$P_1 \oplus P_2 = C_1 \oplus C_2 \tag{6}$$

As we have the ciphertext available xor of the ciphertext will be calculated. The populated chromosomes will be paired in all possible ways ($^{20}C_2$) and the xor will be calculated. These xor results will be compared with the ciphertext xor results. Depending on the similarity of the bits in their position in the ciphertext xor and plaintext xor, the fitness will be evaluated of each plaintext pair by the following fitness function along with equation (7) –

$$\text{Fitness} = \delta/N = \delta/8 \tag{7}$$

δ = Number of similar bits present in $P_1 \oplus P_2$ and $C_1 \oplus C_2$, N = Total number bits in the plaintext (8 bit)

C. Levy Flight

Levy Flight [22-24] is a natural phenomenon that mimics the flying pattern of a flight to go to a random location. This flying model has been designed mathematically by the following equations:

$$x_i^{t+1} = x_i^t + \alpha * \frac{\sigma_c * u}{\frac{1}{|v|^{\beta}}} \left(x_i^t - x_{best}^t\right) \tag{8}$$

Here x_i^t is i[th] solution at t iteration; x_{best}^t is the current best solution. A indicates the step size [25]. Generally, α is considered to be greater than 0 and in most of the cases, the value of α is taken as $\alpha = 0.01$. σ_c can be calculated by the following equation (9) –

$$\sigma_C = \left(\frac{\Gamma(1+\beta)*\sin\left(\pi*\frac{\beta}{2}\right)}{\Gamma\left(\left(\frac{1+\beta}{2}\right)*\beta*2^{(\beta-1)/2}\right)} \right)^{\frac{1}{\beta}} \quad (9)$$

u, v are two Standard Normal Random Variable, t = Current iteration, β is the control factor of Levy Flight, β=1.5.

In CSBCA, the Levy flight is a probabilistic mechanism that allows solutions to explore the search space more widely. It is often used to introduce randomness and prevent the algorithm from getting stuck in local optima. The specifics of how the Levy flight is implemented can vary, but generally, it involves updating the candidate solutions with random steps drawn from a Levy distribution. We have customized the Levy flight function to explore the search space widely to reach the optimal solution. The following pseudo-code of the Levy flight function demonstrates how it searches the population space effectively-

```
population_size = 20
mutation_rate = 0.2
for each generation in range(max_generations):
   for each individual in population:
      step_size = 0.1
      for each bit in individual:
         individual = cuckoo_search(individual, mutation_rate)
         individual = mutate_bitwise(individual, levy_flight(step_size))
```

D. Mutation Algorithm

Mutation [15] is changing some bits inside the chromosomes to achieve the correct plaintext pairs. We have designed the mutation algorithm in such a way that the algorithm can converge into the solutions quickly. The algorithm checks the bitwise xor of plaintexts and the bit position where the mismatches were found and applies mutation there inside the chromosome. By this, a higher fitness valued plaintext pair will be obtained which is close to the optimal solution or itself is the original solution. This mutation algorithm will be applied to all the plaintext pairs that are generated during the execution of the mutation algorithm to maximize the fitness value of the plaintext pairs. Each time the mutation algorithm will generate a possible solution. Here, the set, which is mentioned in the mutation algorithm can be expressed as a queue. The plaintext pairs will be added to the queue; once the plaintext pairs are explored, they are turned out of the queue. The mutation algorithm has been written below –

Algorithm 4: Mutation Algorithm

The best Plaintext pair has been obtained by xor operation in between P_1 and P_2 ($P_1 \oplus P_2$).
Compare ($P_1 \oplus P_2$) and ciphertext pair ($C_1 \oplus C_2$) and mark the bit positions where otherness has been observed.
Consider P_1
For each mismatch bit position
Flip the bit in P_1
Check $(P_1 \oplus P_2) == (C_1 \oplus C_2)$
Store the pair (P_1, P_2)
Consider P_2
For each mismatch bit position
Flip the bit in P_2
Check $(P_1 \oplus P_2) == (C_1 \oplus C_2)$
Store the pair (P_1, P_2)
Check whether any meaningful pair was obtained or not
Do this for all other plaintext pair values and check whether any meaningful plaintext pair was obtained or not

E. Perturbation Method

In applied mathematics perturbation theory [26] compromises an approximate solution to a problem. It starts from an exact solution to a relatable, similar problem. The solution is intimated as a power series with a small parameter β.

$$B = B_0 + B_1\beta^1 + B_2\beta^2 + \ldots\ldots\ldots\ldots \quad (10)$$

Where $\beta \rightarrow 0$

Here the leading term is the exact solution of the related solvable problem; whereas the other terms give a deviation of the solution. So, B_0 is the exact solution of the related problem whereas B is an approximation of the exact solution.

$$B = B_0 + B_1\beta^1 \tag{11}$$

The idea of the perturbation method is used to generate new solutions from the candidate solutions. Though we have customized the perturbation method by our problem statement, it follows the perturbation rules to generate new solutions. The perturbation method is implemented in the cuckoo_search function. This function takes a candidate solution (a binary string) and applies mutation (perturbation) based on a given mutation rate. The mutation involves flipping individual bits with a probability determined by the mutation rate. The function cuckoo_search takes a candidate solution (candidate) and a mutation rate (mutation_rate) as input. For each bit in the candidate solution, it checks if a random number between 0 and 1 is less than the mutation rate. If the random number is less than the mutation rate, the bit is flipped (changed from '0' to '1' or vice versa). This simulates the perturbation or mutation step. If the random number is greater than or equal to the mutation rate, the bit remains unchanged. The function returns the mutated candidate solution (new_candidate). This process is a basic form of mutation, and the mutation rate determines the probability of each bit being flipped. It's a common technique in evolutionary algorithms, like the Cuckoo Search algorithm, to introduce diversity into the population and explore the solution space. The pseudo-code provided below shows how the perturbation method has been implemented.

```
cuckoo_search(candidate, mutation_rate):
  new_candidate = ""
  for each bit in candidate:
    if random.random() < mutation_rate:
      new_bit = "0" if bit == "1" else "1"
      new_candidate += new_bit
    else:
      new_candidate += bit
  return new_candidate
```

F. Proposed Cryptanalysis Algorithm

- **Input:** Population Size, Maximum Number of Iteration, xor value of ciphertext pair.
- **Output:** Current plaintext pair.

Algorithm 5: Cuckoo Search-Based Cryptanalysis Algorithm

1. Initialize all the input variables (Population size (20), Maximum number of iterations, xor value of ciphertext pair, Discard probability (D_p)).
2. Generate 20 candidate plaintext randomly. Each candidate plaintext is an 8-bit binary string.
3. Organise the candidate plaintexts in all possible pairs ($^{20}C_2$). Calculate the xor value of each pair. Now find out the fitness value of each pair by the fitness function using equation (7).
4. Find out the highest fitness value and mark that plaintext pair as the best plaintext pair (the same fitness value can be obtained for multiple pairs).
5. Go to a solution randomly via levy flight using equations (7), (8), and (9) whose fitness is less than 0.5 and generate a new plaintext pair. Convert the plaintext into binary format.
$\text{New\_plaintext} = \text{Current\_plaintext} + \omega\upsilon\ (\omega = 0.01; \upsilon = 1.5)$
6. Apply mutation on the best plaintext pair (if there are multiple best plaintext pairs then apply mutation on all the best plaintext pairs).
7. Compute the fitness of newly created plaintext. If the newly created plaintext pair has higher fitness, then it becomes the new best plaintext pair. If it is true, then go to step 6.
8. Discard the D_p portion of less fit solutions and replace them with newly created solutions using a simple perturbation method using equations (10) and (11).
9. Repeat steps (3) to (8), until the maximum iteration reached of the highest fitness value = 1 is true.
10. Output the best solutions (plaintext pairs).

After evaluating the fitness scores for each candidate in a generation, the code identifies the candidate with the highest fitness (best_fitness). If the best fitness value is equal to 1.0, it means that the candidate solution perfectly matches the target solution, and the XOR cipher has been successfully decrypted. In such a case, the loop breaks, and the algorithm stops further iterations.

In the context of this algorithm, a fitness value of 1.0 implies a perfect match between the candidate solution and the target solution (XOR cipher). The algorithm assumes that reaching this perfect match indicates successful decryption and further iterations are unnecessary. The following submodule explains the CSBCA functionality very briefly.

CSBCA in a Nutshell

The main key components of the CSBCA are discussed below-

- **Generation of Plaintext-Ciphertext Pairs:**

The CSBCA algorithm, in conjunction with the XOR cipher, generates plaintext-ciphertext pairs during its execution. These pairs consist of known plaintext and the corresponding ciphertext. The known plaintext is used to evaluate the fitness of candidate solutions in the evolutionary algorithm.

- **Fitness Evaluation:**

The fitness score of a candidate solution is determined by comparing the XOR of the candidate solution with the plaintext. The goal is to find a candidate solution that, when XORed with the plaintext, produces a result close to the ciphertext.

- **Evolutionary Process:**

The CSBCA algorithm uses an evolutionary approach to iteratively improve candidate solutions. Through selection, crossover, and mutation (which includes Levy flight and simple perturbation in this case), the algorithm aims to evolve candidate solutions towards the correct key that, when XORed with plaintext, produces the ciphertext.

- **Termination Criteria:**

The iterations continue until a stopping criterion is met. In the provided code, the stopping criterion reaches a fitness score of 1.0, indicating a perfect match between the candidate solution and the known plaintext.

- **Decryption:**

Once the algorithm terminates, the best candidate solution found during the process is considered as the decrypted key.

The decrypted key can be used to decrypt other ciphertexts encrypted with the same key.

3.4. Proposed Ciphertext Only Cryptanalysis using Grey Wolf Optimization Algorithm

The proposed algorithm has been characterized by mainly three initial functionalities such as population generation, fitness evaluation, and location upgradation method. All these functions of the attack algorithm have been discussed here. The population generation and the fitness of the Grey Wolf Optimization Algorithm function are the same as mentioned in the previous Cuckoo Search-based attack algorithm. This algorithm is mainly characterized by three parameters Population generation, fitness evaluation, and location upgradation.

A. Population Generation

Initially, 20 candidate solutions (Pack of Grey Wolves) are generated randomly. Here each populated solution represents a plaintext of 8-bit long binary string.

B. Fitness Evaluation

The same fitness function has been used here which was mentioned in the previous cryptanalysis algorithm using equations (6) and (7).

C. Location Upgradation

This module is the most important part of our proposed cryptanalysis algorithms how are we updating the location of the wolves depending on the target position. We have used the Gery Wolf Optimization [14] algorithm in a slightly different way to find out the plaintexts. The proposed Grey Wolf Optimization-based cryptanalysis algorithm along with location upgradation has been discussed below in detail.

Algorithm 6: Grey Wolf Optimization-Based Cryptanalysis Algorithm

1. İnitialze a random Grey Wolf population. (Each wolf represents an 8-bit binary string [plaintext]. İnitially 20 candidate plaintexts (wolves) will be generated and using these plaintexts all possible plaintext pairs will be formed in $^{20}C_2$ ways.
2. Consider two ciphertexts C_y and C_z. Find the xor of $C_y \oplus C_z$ and set it as prey or target value.
3. Find out the xor of each plaintext pair ($P_{\alpha 1} \oplus P_{\alpha 2}$). Find out the fitness of each plaintext pair using equation (7).
4. Mark the highest fitness as G_α, 2nd highest fitness as G_β, and the third highest fitness as G_δ.
5. While termination condition is not encountered (Maximum iteration number has not reached or fitness = 1).
6. For each plaintext pair starting from the highest fitness value (G_α) to the lowest.
7. Update the location of each wolf pair according to the target location. (modify the plaintexts according to the target value).

G_α has been obtained by xor operation in between $P_{\alpha1}$ and $P_{\alpha2}$ $(P_{\alpha1} \oplus P_{\alpha2})$.
Compare $(P_{\alpha1} \oplus P_{\alpha2})$ and $(C_y \oplus C_z)$ and mark the bit positions where otherness has been observed.
Consider $P_{\alpha1}$
For each mismatch bit position
Flip the bit in $P_{\alpha1}$
Check $(P_{\alpha1} \oplus P_{\alpha2}) = (C_y \oplus C_z)$
Store the pair $(P_{\alpha1}, P_{\alpha2})$
Consider $P_{\alpha2}$
For each mismatch bit position
Flip the bit in $P_{\alpha2}$
Check $(P_{\alpha1} \oplus P_{\alpha2}) = (C_y \oplus C_z)$
Store the pair $(P_{\alpha1}, P_{\alpha2})$
Check whether any meaningful pair was obtained or not
Do this for G_β and G_δ and all other G values and check whether any meaningful plaintext pair is obtained or not
8. Repeat steps (1 to 7 except step 2) until an intelligible translation from the plaintext pair is obtained.
9. RETURN THE MEANINGFUL (P_{A1}, P_{A2}).

4. Experimental Results and Analysis

In this section, we have shown the experimental results obtained from the cryptanalysis work of 3-DES using CSA and GWO-based cryptanalysis algorithms. Each experiment has been executed for 100 times. We have shown the best five and worst five results obtained from both the cryptanalysis algorithms. The convergence of the fitness function is also shown graphically. The plaintext is considered as the word "Subinoy94$". Each alphabet, number, and symbol are considered as an 8-bit binary string with the corresponding ASCII code. "Key" is the secret key, which is used to encrypt the plaintext. Table 1 shows the secret key with corresponding ASCII code and table 2 shows the plaintext word with the corresponding ASCII code.

Table 1. Secret key with corresponding ASCII value

Secret Key	ASCII Code
K	01001011
e	01100101
y	01111001

Table 2. Plaintext with corresponding ASCII value

Plaintext	ASCII Code
S	01010011
u	01110101
b	01100010
i	01101001
n	01101110
o	01101111
y	01111001
9	00111001
4	00110100
$	00100100

4.1. Experimental Results Obtained from Cuckoo Search-Based Cryptanalysis Algorithm

In this section, we have presented the results and analysis of the cryptanalysis method of 3-DES using the Cuckoo Search-based Attack algorithm. The experiment was implemented in Python language. The experiment has been carried out for 100 times. We have shown the best five, worst five, and the average result obtained from the Cuckoo Search Algorithm. Different parameters for the Cuckoo Search Algorithm are shown below. Table 3 shows the best five results obtained from the CSA-based cryptanalysis algorithm. In Table 3, in the row corresponding to 'S', T4-1 means that at the fourth trial on the first iteration, the optimal plaintext (S) has been obtained. Similarly in Table 4, the corresponding 'S' row shows the worst five results obtained from the CSA-based cryptanalysis algorithm. In this table, on the 5-worst column, we can see on the 16th trial the algorithm took 36 maximum iterations (T16-36) to find out the optimal plaintext solution (S). In Table 3, the Avg. column represents the average number of iterations taken by the cryptanalysis algorithm to decrypt the DES ciphertext into plaintext. The average has been taken over 100 cycles for each alphabet in the string.

In Table 4, the Avg. column represents the average maximum number of generations taken by the CSA-based cryptanalysis algorithm over the worst 5 results. Fig. 5 shows the convergence of the fitness function for each alphabet in the string. The convergence plotting of the fitness function has been made according to the average iteration numbers. In this fig., the first image shows the convergence graph of the alphabet 'S' on the 91st iteration.

Population Size = 20
Mutation Rate = 0.2
Maximum Number of Iteration = 100
Plaintext = Subinoy94$

Table 3. Best five results obtained from CSA

String	Total Number of Generations in a Single Iteration					Avg.
	1-Best	2-Best	3-Best	4-Best	5-Best	
S	T4-1	T20-2	T1-3	T5-4	T25-5	9.47
u	T4-1	T32-2	T13-3	T3-4	T1-5	11.9
b	T38-1	T25-2	T14-3	T1-4	T15-5	10.35
i	T20-1	T4-2	T18-3	T14-4	T6-5	9.85
n	T6-1	T11-2	T9-3	T3-4	T2-5	10.58
o	T25-1	T2-2	T19-3	T4-4	T23-5	11.43
y	T25-1	T2-2	T8-3	T7-4	T3-5	10.97
9	T3-1	T14-2	T1-3	T16-4	T21-5	11.35
4	T16-1	T18-2	T10-3	T3-4	T24-5	12.33
$	T2-1	T12-2	T56-3	T24-4	T36-5	10.18

Table 4. Worst five results obtained from CSA

String	Total Number of Generations in a Single Iteration					Avg.
	1-Worst	2-Worst	3-Worst	4-Worst	5-Worst	
S	T19-22	T18-23	T72-30	T98-31	T16-36	28.4
u	T30-29	T14-31	T75-34	T73-39	T74-42	35
b	T91-27	T19-30	T46-34	T81-37	T36-65	38.6
i	T49-23	T77-24	T80-26	T48-30	T60-35	27.5
n	T25-26	T13-27	T32-28	T96-32	T80-37	30
o	T52-24	T33-25	T68-26	T87-28	T61-31	26.8
y	T47-25	T54-30	T22-31	T97-32	T15-38	31.2
9	T9-36	T76-38	T64-45	T82-47	T51-49	43
4	T13-36	T57-41	T43-42	T88-45	T87-56	44
$	T97-23	T28-24	T10-25	T20-31	T59-54	31.4

4.2. Experimental Results Obtained from Grey Wolf Optimization-Based Cryptanalysis Algorithm

In this section, we have presented the results and analysis of the cryptanalysis method of 3-DES using the Grey Wolf Optimization-based Attack algorithm. Here also the experiment has been implemented in Python language and the experiment was carried out 100 times. Table 5 and Table 6 show the best five and worst five results obtained from the Grey Wolf Optimization algorithm. In the best five results, the $ row shows the first-best result is obtained in a single generation in the 31st iteration (T31-1) the second-best result is obtained with two generations in the 3rd iteration (T3-2), and so on. The 'Avg.' column in Table 5 shows the average number of generations taken to reach the plaintext from the ciphertext. The average has been taken over 100 iterations. In table 6, the average has been taken over the worst five results obtained from the GWO-based cryptanalysis algorithm. As the average is taken over the worst five results, this can be represented as the average worst number of generations taken to generate the plaintext. Fig. 6 shows the convergence of the fitness function for each alphabet in the plaintext string.

Population Size = 20
Mutation Rate = 0.2
Maximum Number of Iteration = 100
Plaintext = Subinoy94$

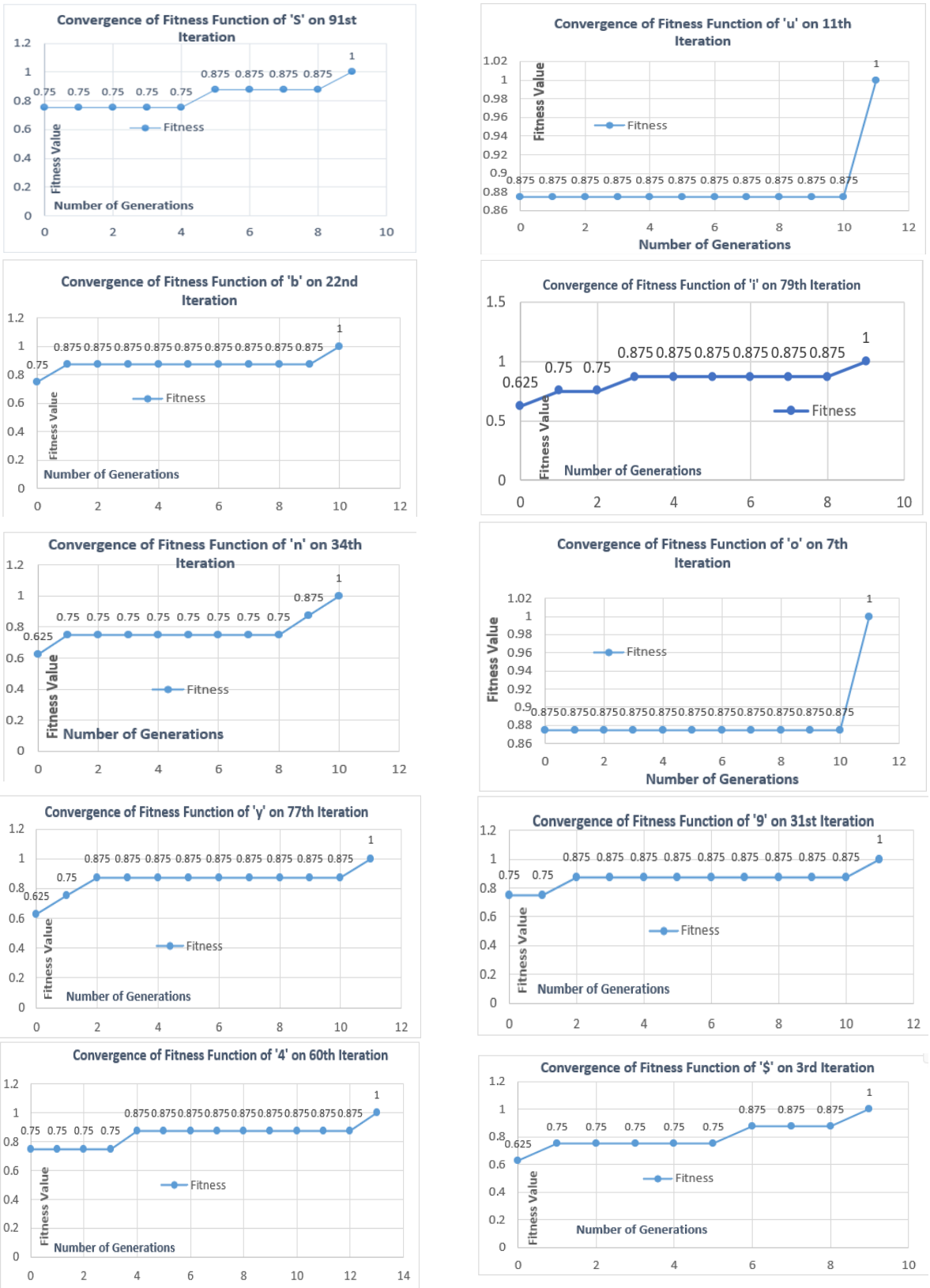

Fig.5. Convergence of fitness function (CSA)

Table 5. Best five results obtained from GWO

String	Total Number of Generations in a Single Iteration					Avg.
	1-Best	2-Best	3-Best	4-Best	5-Best	
S	T1-1	T21-2	T12-3	T2-4	T5-5	7.44
u	T1-1	T40-2	T11-3	T7-4	T2-5	9.46
b	T3-1	T1-2	T9-3	T4-4	T13-5	8.5
i	T9-1	T8-2	T1-3	T2-4	T16-5	11.09
n	T5-1	T3-2	T2-3	T48-4	T12-5	10.29
o	T11-1	T7-2	T17-3	T28-4	T30-5	12.94
y	T5-1	T6-2	T1-3	T9-4	T3-5	7.54
9	T1-1	T67-2	T5-3	T6-4	T4-5	6.71
4	T5-1	T19-2	T23-3	T4-4	T10-5	7.38
$	T31-1	T3-2	T2-3	T10-4	T1-5	6.46

Table 6. Worst five results obtained from GWO

String	Total Number of Generations in a Single Iteration					Avg.
	1-Worst	2-Worst	3-Worst	4-Worst	5-Worst	
S	T25-19	T13-20	T40-21	T93-24	T56-49	26.6
u	T50-37	T74-39	T28-43	T26-44	T58-48	42.2
b	T84-24	T80-30	T61-35	T16-41	T68-42	34.4
i	T5-35	T10-41	T53-42	T74-45	T7-57	44
n	T75-29	T40-32	T33-41	T53-46	T71-55	40.6
o	T64-37	T50-38	T45-41	T49-47	T15-53	43.2
y	T60-23	T10-24	T8-30	T7-35	T49-43	31
9	T87-18	T74-19	T38-26	T12-29	T68-34	25.2
4	T74-26	T55-28	T72-31	T20-35	T6-42	32.4
$	T23-14	T27-15	T51-16	T26-21	T36-27	18.6

4.3. Comparison of Experimental Results Obtained from Cuckoo Search and Grey Wolf Optimization-Based Cryptanalysis Algorithm

In this section, the experimental results are discussed and analyzed. If we consider the search space as 100%, then in both the cryptanalysis algorithms we have used 8% (7.81≈ 8) of the search space as the initial population. From that initial population, these algorithms generate the plaintext solutions. The CSA-based cryptanalysis algorithm took close to 11 (10.841≈11) generations to decrypt each 8-bit (each alphabet/digit/symbol) ciphertext into plaintext. Whereas the GWO-based cryptanalysis algorithm took approximately 9 (8.781≈9) generations to decrypt each letter of the ciphertext into plaintext. In the worst-case scenario, both algorithms have taken roughly 34 (CSA-33.59|GWO-33.82) generations to obtain the plaintext from the ciphertext. So, in the average case, the CSA-based cryptanalysis algorithm explores 12% of the search space to obtain the plaintext and the GWO-based cryptanalysis algorithm traverses 11% of the search space to find the optimal solution. In the worst case, both the attack algorithms inspect 21% of the search space to reach the optimal solution from the given ciphertext. From the above discussion, it is visible that GWO based cryptanalysis algorithm performs better than CSA based attack algorithm. The reason behind the better performance of the GWO-based algorithm is in the case of CSA, the optimal solution is reached by replacing the other bad solutions with the good solution which is generated by the cuckoo's egg. In CSA there is no teamwork for the breeding purpose. This is the main reason for the late convergence of the fitness function into the correct solution. Whereas in the case of GWO, they follow a smart hunting behavior hierarchically in a pack. These smart behaviors help the GWO-based attack algorithm to converge into the optimal solution quite first. Table 7 shows the average and worst-case performance of CSA and GWO-based cryptanalytic attack algorithms. As the worst-case average performance is taken over only the worst five iterations over 100 iterations, the average performance indicates the most generalized performance. Fig. 7 shows the comparison of the obtained results from the proposed attack algorithms.

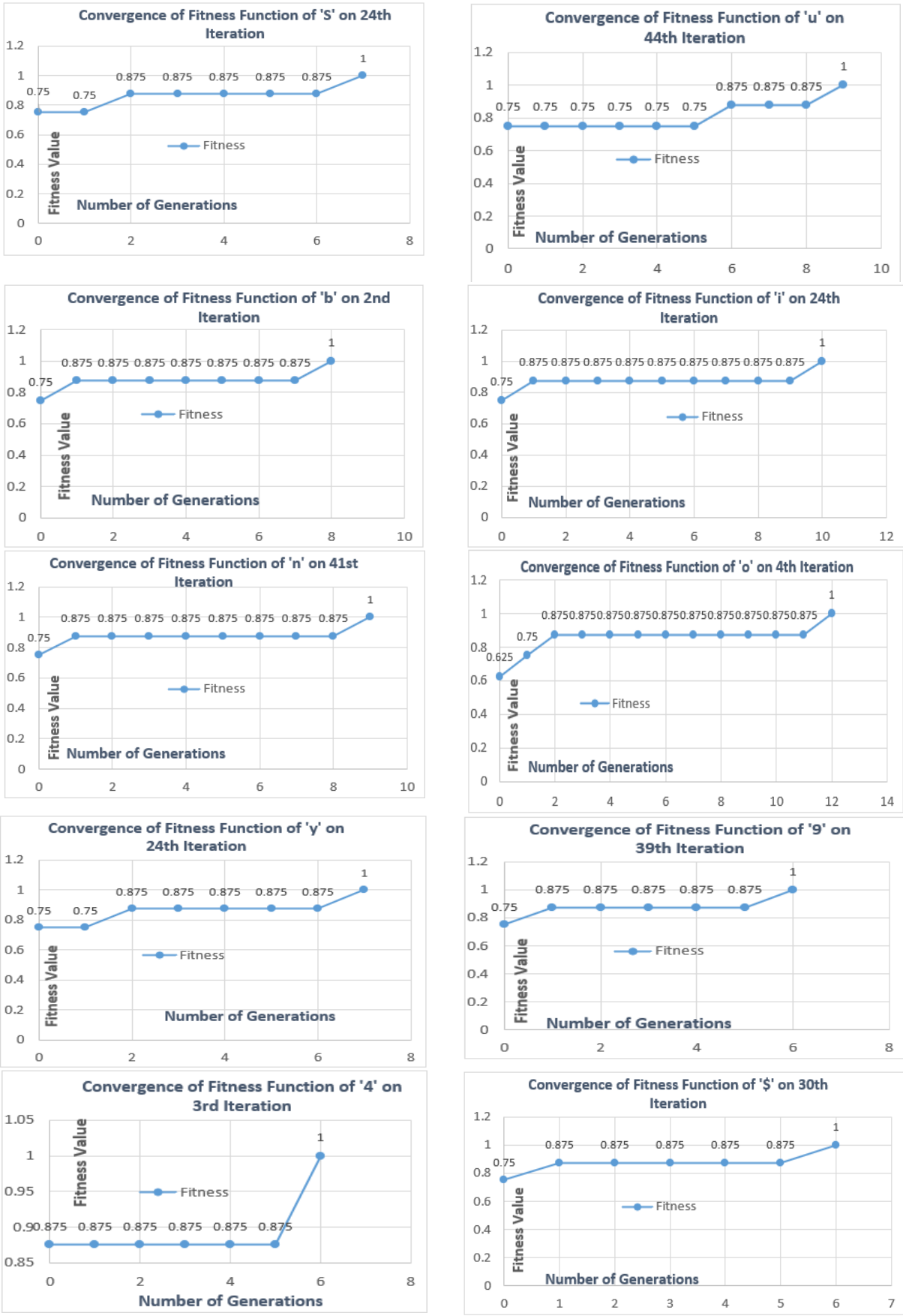

Fig.6. Convergence of fitness function (GWO)

Table 7. Comparison of results obtained from CSA and GWO

Plaintext	CSA		GWO	
	Average	Worst	Average	Worst
S	9.47	28.4	7.44	26.6
u	11.9	35	9.46	42.2
b	10.35	38.6	8.5	34.4
i	9.85	27.5	11.09	44
n	10.58	30	10.29	40.6
o	11.43	26.8	12.94	43.2
y	10.97	31.2	7.54	31
9	11.35	43	6.71	25.2
4	12.33	44	7.38	32.4
$	10.18	31.4	6.46	18.6
Average	10.841	33.59	8.781	33.82

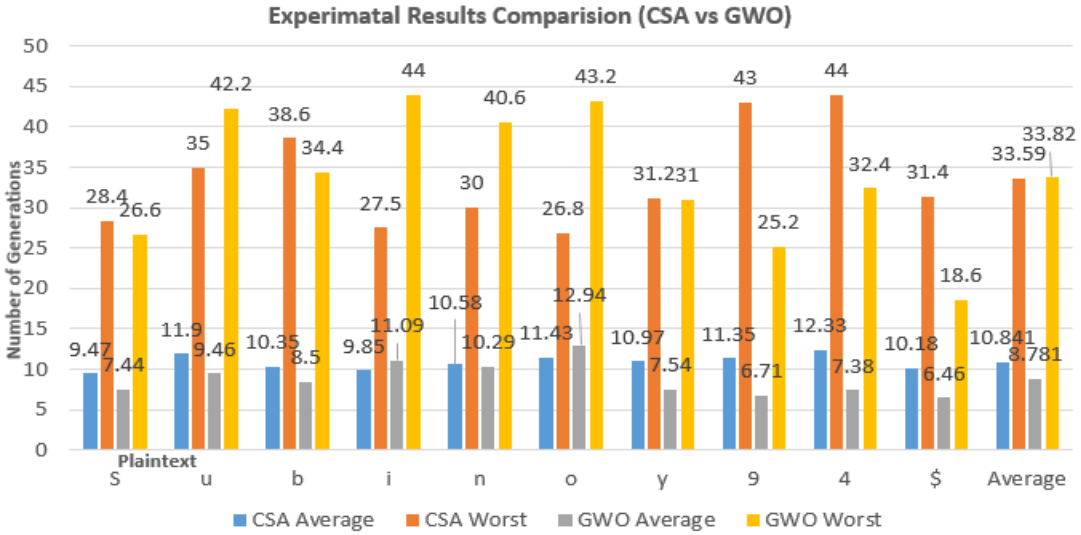

Fig.7. Results Comparison (CSA vs GWO)

5. Conclusions

The paper presents a cryptanalysis of 3-DES using two nature-inspired algorithms. It reexamines the Known Plaintext Attack on 2-DES using Meet-In-The-Middle Approach to uncover valid key pairs. However, the paper highlights the limitations of the Meet-In-The-Middle Approach and proposes a Ciphertext Only Attack using the Cuckoo Search and Grey Wolf optimization algorithms. The experimental results reveal that the proposed algorithms can perform quite good cryptanalysis with a better performance in the case of Grey Wolf Optimization algorithm. The proposed approach is adaptable not only for analyzing 3-DES but also for various rounds of DES, suggesting that increasing the number of rounds may not necessarily enhance security. This approach can potentially be applied to the analysis of other block ciphers as well.

References

[1] Forouzan. B.A., “Cryptography and Network Security”, Tata McGraw-Hill, New Delhi (2007).
[2] Atul Kahate, “Cryptography and Network Security”, McGraw Hill Education (India) Private Limited.
[3] William Stallings, “Cryptography and Network Security: Principle and Practice”, Pearson.
[4] M.V. Arun Kumar, “Network Security”, Laxmi Publications.
[5] Ajay Raj Parashar, Deepti Mittal, “Cryptography and Network Security”, Laxmi Publications.
[6] Bernard Menenzes, “Network Security and Cryptography”, Thomson Press (India) Ltd.
[7] Express Learning - Cryptography and Network Security (ITL Education Solutions Limited), Pearson Education India.
[8] R. Kamal, M. Bag, M. Kule, “On the Cryptanalysis of S-DES Using Binary Cuckoo Search Algorithm”, Computational Intelligence in Pattern Recognition, vol. 999, pp. 23-32, 2020. DOI:10.1007/978-981-13-9042-5_3
[9] M. Matsui, “Linear Cryptanalysis Method for DES Cipher”, In *Workshop on the Theory and Application of of Cryptographic Techniques*, vol. 765, pp. 386-397, 1993. DOI: 10.1007/3-540-48285-7_33

[10] E. Biham, A. Shamir, “Differential Cryptanalysis of the Full 16-round DES”, In *Advances in Cryptology—CRYPTO’92: 12th Annual International Cryptology Conference Santa Barbara, California*, vol. 740, pp. 487-496, 1993. DOI: 10.1007/3-540-48071-4_34
[11] E. Biham, A. Shamir, “Differential cryptanalysis of DES-like cryptosystems”, *Journal of Cryptology, vol.* 4, pp. 3–72, 1991. DOI: 10.1007/BF00630563
[12] X. S. Yang and S. Deb, "Cuckoo Search via Lévy flights", *World Congress on Nature & Biologically Inspired Computing(NaBIC)*, pp. 210-214, 2009. DOI: 10.1109/NABIC.2009.5393690
[13] L. Jiao, R. Shang, F. Liu, W. Zhang, “Brain and Nature-Inspired Learning Computation and Recognition”, Elsevier, pp. 127-195, ISBN 9780128197950, 2020.
[14] Mirjalili Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis, "Grey wolf optimizer", *Advances in engineering software, vol.* 69, pp. 46-61, 2014. DOI: 10.1016/j.advengsoft.2013.12.007
[15] Abdoun Otman, Jaafar Abouchabaka, and Chakir Tajani, "Analyzing the performance of mutation operators to solve the traveling salesman problem", *International Journal of Emerging Sciences, vol. 2(1), pp. 61-77*, 2012. DOI: 10.48550/arXiv.1203.3099
[16] Grari, Hicham, Ahmed Azouaoui, and Khalid Zine-Dine, "Ant colony optimization for cryptanalysis of simplified-DES", *Advanced Intelligent Systems Applied to Energy*, vol. 912, pp. 111-121, 2019. DOI: 10.1007/978-3-030-12065-8_11
[17] Abd-Elmonim, Wafaa G., Neveen I. Ghali, Aboul Ella Hassanien, and Ajith Abraham, "Known-plaintext attack of DES-16 using Particle Swarm Optimization", *Third World Congress on Nature and Biologically Inspired Computing*, pp. 12-16, 2011. DOI: 10.1109/NaBIC.2011.6089410
[18] Kamal Ritwiz, Moynak Bag, and Malay Kule, "On the cryptanalysis of S-DES using nature-inspired optimization algorithms", *Evolutionary Intelligence, vol.* 14, pp. 163-173, 2021. DOI: 10.1007/s12065-020-00417-5
[19] Tiwari Vikas, Ajeet Singh, and Appala Naidu Tentu, "Differential cryptanalysis on DES cryptosystem up to eight rounds", *International Journal of Information Privacy, Security and Integrity, vol.* 4(1), pp. 1-29, 2019. DOI: 10.1504/IJIPSI.2019.103560
[20] Amic Seeven, KM Sunjiv Soyjaudah, Heman Mohabeer, and Gianeshwar Ramsawock, "Cryptanalysis of DES-16 using binary firefly algorithm", *IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, pp. 94-99, 2016. DOI: 10.1109/EmergiTech.2016.7737318
[21] Tezcan Cihangir, "Key lengths revisited: GPU-based brute force cryptanalysis of DES, 3DES, and PRESENT", *Journal of Systems Architecture*, vol. 124, 102402, 2022. DOI: 10.1016/j.sysarc.2022.102402
[22] A.F. Kamaruzaman, A.M. Zain, S.M. Yusuf, A. Udin, “Levy Flight Algorithm for Optimization Problems - A Literature Review”, *Applied mechanics and materials*, vol. 421, pp. 496–501, 2013. DOI: 10.4028/www.scientific.net/AMM.421.496
[23] Y. Liu and B. Cao, "A Novel Ant Colony Optimization Algorithm With Levy Flight", *IEEE Access*, vol. 8, pp. 67205-67213, 2020. DOI: 10.1109/ACCESS.2020.2985498
[24] Sangita Roy, Sheli Sinha Chaudhuri, "Cuckoo Search Algorithm using Lèvy Flight: A Review", International Journal of Modern Education and Computer Science, vol.5, no.12, pp.10-15, 2013.
[25] M.A. Gutiérrez, S. Krenk, “Stochastic Finite Element Methods”, *Encyclopedia of Computational Mechanics Second Edition*, pp. 1-25, 2018. DOI: 10.1002/9781119176817.ecm2044
[26] Carl M. Bender, “Perturbation Theory in large order”, Advances in Mathematics, vol. 30, pp. 250-267, 1978. DOI: 10.1016/0001-8708(78)90039-7

Authors’ Profiles

Subinoy Sikdar received his B.E. degree in Information Technology from the University Institute of Technology, Burdwan University, India in 2018 and his M.E. degree in Information Technology from Jadavpur University, India in 2021. He is currently pursuing PhD at the Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, India. His areas of interest are cryptology, cryptanalysis, machine learning, and soft computing.

Sagnik Dutta is currently pursuing his B. Tech. degree in Computer Science and Technology at the Indian Institute of Engineering Science and Technology, Shibpur, India. Sagnik has demonstrated proficiency in a diverse range of programming languages and frameworks. He has worked on notable projects, such as the development of a Face Recognition GUI for an attendance system and the creation of 'BooksReading,' a comprehensive website for book enthusiasts. Additionally, he showcased his technical prowess by crafting a Chrome extension, 'Video Teacher,' designed to summarize lengthy YouTube videos. Sagnik's dedication extends beyond academics, as exemplified by his insightful Cryptographic Analysis Internship, where he delved into encryption algorithms and key management protocols.

Malay Kule has been on the faculty of the Indian Institute of Engineering Science and Technology, Shibpur, India since 2013, where currently, he is an assistant professor of the Computer Science and Technology department. Previously he served as an assistant professor in the Department of Computer Science and Engineering of St. Thomas’ College of Engineering and Technology, Kolkata, India from 2006 to 2013.

Dr. Kule received the B.Sc. degree in physics, the B.Tech. and M.Tech. degrees in Computer Science and Engineering, all from the University of Calcutta, India. He received his PhD degree in Engineering from the Indian

Institute of Engineering Science and Technology, Shibpur, India.

His research interest includes defect tolerance and testing of nanoscale circuits, cryptology, and hardware security. He has published more than 40 research papers in journals and conference proceedings. He served on the conference committees of ATS, ISDCS, CIPR, COMSYS, VLSID, etc.

How to cite this paper: Subinoy Sikdar, Sagnik Dutta, Malay Kule, "On Cryptanalysis of 3-DES using Nature-Inspired Algorithms", International Journal of Computer Network and Information Security(IJCNIS), Vol.17, No.3, pp.54-71, 2025. DOI:10.5815/ijcnis.2025.03.04