Received 21 May 2024, accepted 25 July 2025, date of publication 30 September 2026
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.25.3.4853

Deciphering Transposition Ciphers in Wireless Sensor Networks: LFA's Lévy Flight Approach

SUFYAN OTHMAN ZABEN
Computer Engineering Department, College of Engineering, Al-Iraqia University, Baghdad, Iraq

Corresponding author: Sufyan Othman Zaben (e-mail: Sufyan.o.zaben@aliraqia.edu.iq).

ABSTRACT Information system security depends on the strength of the employed cryptosystem. Over the years, various cryptographic algorithms have been developed, but some have inherent weaknesses that can compromise data integrity. This paper introduces a cuckoo algorithm designed to decrypt transposition ciphers between sensor nodes in wireless sensor networks. Our approach builds on existing research and presents innovative concepts like the Lévy-flight function, a dictionary of common English words for key fitness evaluation, dynamic selection, and variable generation schemes. Starting with a random set of keys, the algorithm uses Lévy-flight operations on high-fitness keys to generate new generations until a specified number is reached. The result yields a key close to the original encryption key. Our results show that our algorithm finds optimal solutions with linear time performance, significantly improving over the brute-force algorithm. This research demonstrates our approach's efficiency in enhancing data security and integrity in wireless sensor networks.

KEYWORDS Transposition Cipher, Cuckoo search algorithm, wireless sensor network, lev'y flight method, Deciphering, Wireless Sensor Network , cryptology, n-gram statistics,

I. INTRODUCTION

Securing information transmitted over networks involves various methods, with cryptology serving as the core discipline that provides such assurances. Cryptology is the science dedicated to constructing and analyzing diverse encryption and decryption techniques, comprising two primary subfields: Cryptography and CryptanalysCryptanalysis is the study of methods for breaking cryptographic techniques, specifically ciphers. It involves the exploration of potential vulnerabilities in encryption methods. Cryptanalysis can, at best, attempt to guess the encryption key, and it relies on a vast number of guesses for the key to be correct, thereby aiming to achieve near-unbreakable ciphers [l].

Classical ciphers come in many forms, but most of them can be categorized into two broad categories: substitution ciphers and transposition ciphers. Transposition ciphers maintain the same plaintext but rearrange the order of characters. However, when the key is short and the message is long, various cryptanalysis techniques can be employed to break such ciphers.

In a simple columnar transposition cipher, the plaintext is horizontally written on graph paper with a fixed width, and the ciphertext is read vertically [2-5]. This method involves dividing the plaintext into fixed-sized blocks, with the length of the key (referred to as the period) matching the size of the block. Letters within each block are permuted based on a consistent pattern defined by the key [3]. Table 1 illustrates the encryption procedure and the key for the aforementioned cipher for transposition.

II. RELATED WORK

Numerous research efforts have been dedicated to developing methods for cracking the transposition cipher. Some of these studies have employed simulated annealing [6-10], a local search technique that employs an evaluation function to assess the current state and endeavors to transition to more favorable states within the search space. This method permits occasional suboptimal moves to keep the algorithm from falling into local optima traps. Simulated annealing draws inspiration from the metallurgical annealing process, wherein heated metal is gradually cooled to achieve improved structural integrity or "energy." While experiments using this approach to decrypt transposition ciphers have shown it to be effective, it falls short in performance when compared to genetic algorithms.

In another interesting research effort, transfer ciphers are defined using particle set optimization (PSO) [6], an approach that relies on swarm intelligence principles of bird colonies and other groups, such as salmon schools, collective practices to

motivate them. These particles continue to move, and when an ideal solution is encountered, the particle updates its optimal solution. The tests carried out by the authors gave excellent results; it took just 1.3 seconds to crack a ten-character key of 250 characters.

Further researchers have explored Tabu search [7], which initializes a good solution with a random key. This key is then evaluated, and a list of keys is generated by swapping numbers within it. The new set of keys is evaluated, and the best among them replaces the previous one. This process is repeated for a specified number of iterations. The Tabu algorithm has also demonstrated impressive findings, although not quite as remarkable as genetic algorithms. Its main advantage lies in identifying the best key more quickly, although the overall decryption process may take longer due to the evaluation of multiple keys.

An alternative approach involves the Ant Colony algorithm [8], which draws inspiration from ant behavior and their pheromone-based communication to guide the colony towards the optimal route to accomplish a goal. Nevertheless, in comparison to genetic algorithms, simulated annealing, and Tabu search, this approach performs badly.

Lastly, we consider alternative approaches employing genetic algorithms to decrypt the cipher for transposition. A notable study by Toemeh R. et al. [9], upon which our work builds, yielded excellent results. Nevertheless, we have identified various limitations and problems in this study. The authors did not adequately illustrate the rationale behind the crossover operation or justify how it would enhance the subsequent generation. Additionally, this approach fixed the number of generations to be examined and mutations were applied to all options [14-19]. An additional observed limitation is the lack of a thorough explanation for the initial population selection, which is crucial due to the immense search space (the factorial of the key length) [20-25].

III. COLUMNAR TRANSPOSITION

A columnar transposition cipher reads the message column by column after it has been typed out in rows of a certain length. The order of the columns is determined by a keyword. Typically, both the row width and column permutation are defined by this keyword. For instance, consider the word "ZEBRAS," which has a length of 6 characters (producing rows with a length of six). The permutation is determined by alphabetically arranging the letters, resulting in the column order "6 3 2." This transposition method involves simply placing the plaintext into a column format as is. Remarkably, this algorithm can also handle numeric values and special characters.

IV. WIRELESS SENSOR NETWORK (WSN)

Sensor nodes provide a potent blend of distributed sensing, computing, and communication capabilities. The continuous enhancement of these small sensor nodes, encompassing sensing, data processing, and communication functionalities, facilitates the establishment of Wireless Sensor Networks (WSNs) through the collaborative efforts of multiple sensor nodes. WSNs open up a vast array of applications but also pose several challenges, mainly driven by their distinctive characteristics, with stringent energy constraints being a primary concern for these sensing nodes.

Wireless Sensor Networks (WSNs) combine technology and expertise from three different domains: networking, systems and control theory, and wireless communications. To meet the needs of current and potential WSN applications, the development of advanced and highly efficient communication protocols is imperative.

V. TYPES OF ATTACK

Figure 1 provides a summary of the main types of attacks.

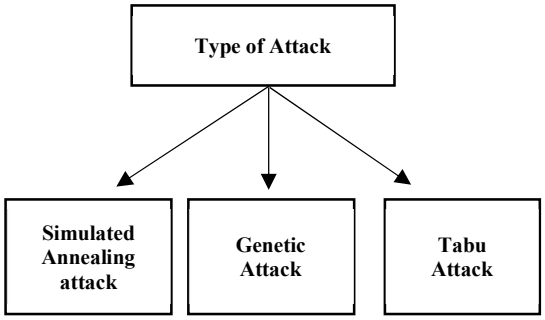

Figure 1. Type of Attack

First: A Simulated Annealing Attack

In this segment, we'll delve into a method to breach the transposition cipher, which involves simulated annealing. This technique draws inspiration from the physical concept of annealing, which involves gradually cooling heated metal to attain a state of minimum energy. Kirkpatrick et al. [4] are credited with the idea of utilizing this process to tackle combinatorial optimization challenges. The algorithm starts with an initial temperature and an arbitrary solution to the problem.

Second: A Genetic Attack

The genetic algorithm is a more complex method compared to annealing attack simulation due to the need for multiple solutions rather than only one and additional complexity in mutation and mating functions.

Third: A Tabu Attack

The transposition cipher can be susceptible to tabu search attacks, which resemble annealing technique simulation but use tabu list constraints to impose additional constraints. The method involves swapping randomly selected elements in a key and generating potential solutions, with the worst key in the list being replaced with the best new key every iteration.

VI. CUCKOO SEARCH

In today's problem-solving landscape, swarm algorithms have become increasingly favored. Among them, Yang and Deb introduced the Cuckoo Search algorithm, drawing inspiration from the intriguing behavior of cuckoo birds. Mimicking nature, this algorithm mirrors how cuckoos lay eggs in the nests of other birds. In the algorithmic realm, the problem's components are likened to a cluster of nests, while the defining characteristics of these components are symbolized by the eggs deposited by cuckoos.

The Cuckoo Search algorithm randomly selects a nest for each cuckoo bird to lay its egg, then dumps any existing eggs in the nest. The next generation is chosen from the best nests with high-quality eggs. The problem has a fixed number of nests, each representing an element of the problem.

The Cuckoo Search algorithm also incorporates a concept called Lévy flight. This random walk is a combination of extended trajectories with dramatic 90-degree turns and short, stochastic movements. It is employed to explain the foraging

behaviors of species found in natural systems, including zooplankton, ants, bees, and bumblebees. The mathematical model of Lévy flight, which is related to chaos theory, is extensively employed in stochastic simulations of both random and pseudorandom phenomena in nature. Turbulent fluid movements also display these flights. The figure below demonstrates one example of a Lévy flight path.

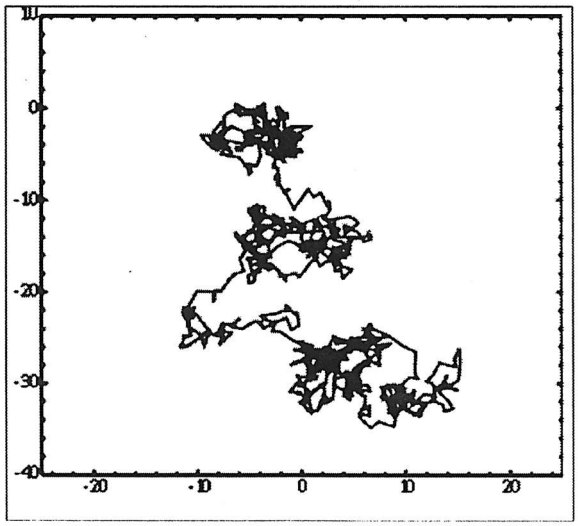

Figure 2. Possible Lévy flight path

Many search algorithms employ Levy-style search behavior and random search applications for optimization purposes. The Cuckoo Search (CS) algorithm is one such algorithm that has shown promising results in initial testing. To simplify things further, we can approximate the last assumption by the fraction of the probability distribution of n nests substituted by new nests containing new solutions that are random.

For a maximization problem, the fitness of a solution can be proportional to the value of its objective function. Alternatively, other forms of fitness can be defined in a similar manner to how the fitness function is defined in genetic algorithms and other evolutionary computation algorithms.

In this algorithm, a single egg in a nest is considered as a solution, while an egg of cuckoo represents a potentially better solution. The goal is to replace inferior solutions in the nests with these new solutions (cuckoos). This algorithm can be expanded to handle more complicated scenarios in which each nest includes multiple eggs, each of which represents a collection of solutions. Numerous search algorithms have been developed that employ random search in general and a Lévy-style search behavior for optimization. CS is one of these algorithms. Initial findings indicate that it has promising potential. To further simplify the situation, the pd fraction of n nests that have been replaced with new nests with new random solutions can be used to approximate the last assumption. The quality (fitness) of a solution to a maximization issue can be expressed as simply being proportionate to the value of its objective function [8].

Similar to how the fitness function is introduced in genetic algorithms and other evolutionary computation algorithms, other forms of fitness can also be demonstrated in this manner. Here, a simple illustration is employed, where a single egg in a nest symbolizes a solution and an egg of cuckoo indicates a solution. The goal is to replace the poor solutions in the nests with the new, possibly superior solutions (cuckoos). The more complex scenario where each nest contains several eggs, each of which represents a set of solutions, can be handled by extending this technique.

A new solution x (t+l) for cuckoo i is produced employing a Lévy flight based on the following equation:

$$X_i^{(t+1)} = X_i^{(t)} + \alpha.\text{Lévy}\,(\lambda), \qquad (1)$$

where $X_i^{(t)}$: Current position (solution) of the i^{th} cuckoo at iteration t

$X_i^{(t+1)}$: Updated position of the i^{th} cuckoo at iteration $t+1$

α:Step size scaling factor (a positive constant that controls the step magnitude)

Lévy : Lévy flight distribution with parameter λ, typically $1 < \lambda \leq 3$, representing a random step drawn from a heavy-tailed probability distribution

For a step scaling size, denoted as a (where a is greater than 0), it should be related to the scales of the problem that the algorithm aims to solve. In most cases, the value of u can be set to I or another constant. Product A indicates the entry-wise multiplications.

The random walk that is described is a Markov chain, according to equation (2), and its next location depends on two factors: the current position (first term of Eq. 2) and the probability of transition (second term of the same expression).

A Lévy distribution that has an infinite mean and an infinite mean is applied to determine the random step length.

$$\text{Lévy} \sim u = t_1^{-\lambda}, \qquad (2)$$

where u: Step length drawn from the Lévy distribution;

t_1: A random variable (usually drawn from a uniform distribution between 0 and 1) : A random variable (usually drawn from a uniform distribution between 0 and 1)

where $\lambda \in (0, 3)$

Considering the fundamental three guidelines mentioned previously, the pseudo code for the CS algorithm is: [9]

Name of Algorithm: Classical Cuckoo Search
Input: Population Of nest xi = (xiI , ……. , xiD) T for I = 1 … Np , MAX_FE
Output: The best solution xbest and its corresponding value fmin = min (f(x)).
1. generate initial host locations ();
2. eval = 0;
3. While termination condition not meet do
4. for i = 1 Np do
5. xi = generate_new_solution (xi) ;
6. fi = evaluate the_new soltuion(xi);
7. eval = eval + 1;
8. j = rand (0, 1) *Np+ 1;
9. then
10. xj = xi ; fj = fi ; // replace j-th solution
11. end if
12. if fi < finin then
13. xbest = xi ; fnin = fi ; // save the local best sol.
14. end if
15. end for

VII. PROPOSED METHOD

A method for comparing candidate keys involves analyzing the n-gram statistics of the decrypted message and comparing them to those of the known language. To confirm whether a specified key (k) is sufficient, Equation 1 is a commonly used formula. In this equation, A represents the language alphabet (such as [A, ... , Z, l] for English, where the space symbol is represented by ' '). The decrypted message statistics are represented by D, while the known language statistics are represented by K, and the indices 'ni' indicate the number of dividers in the cipher text.

$$C_k = | \sum k^{n_i} - \sum D^{n_i} \qquad (3)$$

where C_k: Key fitness cost – a measure of how closely the decrypted message matches the expected language model
k^{n_i}: N-gram frequency (e.g., unigram, bigram, trigram, etc.) from the known language model
D^{n_i}: N-gram frequency extracted from the decrypted message
n_i: The specific n-gram order being evaluated (e.g., 1 for unigram, 2 for bigram, etc.)

To decode a ciphertext, we need to identify the factors that evenly divide the length of the text. This will provide us with all possible key lengths used in encryption. We then create a set of random keys for each divisor to initiate the decryption process. The total number of keys to consider is equivalent to the factorial of the key length.

By employing these keys, we decrypt the text of the cipher and cross-reference the result with a dictionary of words that are commonly used. The decryption is scored according to the number of matches identified in the dictionary, and the top scorers are selected as the most viable options. These options are then used to generate the next set of nodes through the Lev'y flight operation.

The Lev'y flight operation is a powerful tool for producing more fit generations, but it must be used with intention to achieve optimal results. In the wireless sensor network, figure 3 illustrates the place of the Lev'y flight attack when it happened in the WSN environment. It shows the attack event between two nodes in the sensor field before it researches to the base station.

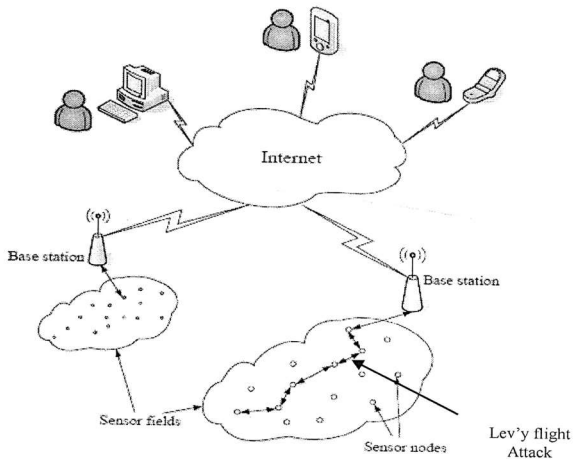

Figure 3. Lev'y flight in WSN

Table 1 explains the iteration for generation different key, so the results show the local optimal solution for each key length.

Table 1. Local optimal solution for each key length

G1				
N = 2		N = 3		N = 3
Key generated		Key generated		Key generated
xx yy		xxx yyy zzz www		xxx yyy zzz www
G2				
N = 2		N = 3		N = n
Key generated		Key generated		Key generated
xx yy		xxx yyy zzz www		XX .. X_n yy .. y_n zz .. z_n
G3				
N = 2		N = 3		N = n
Key generated		Key generated		Key generated
xx yy		xxx yyy zzz www		XX .. X_n yy .. y_n zz .. z_n

However, Table 2 show the global optimal solution for each key length.

Table 2. Global optimal solutions for each key length

Pool of nest for 2	Pool of nest for 3	Pool of nest for n
Best result of G1	Best result of G1	Best result of G1
Best result of G2	Best result of G2	Best result of G2
Best result of Gn	Best result of Gn	Best result of Gn

The following Steps illustrate the proposed work by using modification of cuckoo known ciphertext characters.

Name of Algorithm : Levy' Flight Attack
Input:
Output:
Step 1: Implement number of ciphertext dividers as population of nests. Step 2: Implement the generated keys depending on length of divided ciphertext as eggs. Step 3: While generations didn't reached to maximum generation do Step 4: Generate random keys for each nest using factorial length for each divider. Step 5: For each generated keys in the same nest do Step 6: Test the generated key in fitness function by making comparison between the n-keys for the same length which key produce the suitable words the key will be the best will be pushed into pool step 7: For each candidate key for same nest after the looping of generation is

finished, the comparison between reserved keys in pool by substitute in
fitness function to get the (most fit) between these keys the result will be
the suitable key for each length of the ciphertext.
Step 8: End
Step 9: End

VIII. EXPERIMENTS AND RESULTS

The proposed work has been tested using the MATLAB application, with ciphertexts of varying lengths (200, 400, and 1000 characters) loaded individually from three text files into the application. Each loaded ciphertext has been segmented into several parts based on the suggested key length and then used to generate N-keys, where the number of keys is equal to the factorial of the length of each divided part.

All generated keys have been tested on each divided ciphertext and compared against an external dictionary of commonly used English words to determine the suitability of the current key.

The initial comparison focuses on the amount of provided text of cipher for the attack, which is presented in Table 1. For each amount of ciphertext, each algorithm was executed 50 times. The findings in Table 2 depict the average number of correctly placed key elements for a 15-key size in this particular context.

Table 3 illustrates the results for the cipher of transposition according to the columnar key matrices. Specifically, it highlights the modification of the cuckoo search algorithm as the most potent approach for columnar transposition ciphers with a period of 30. On average, the modified cuckoo search successfully placed 24.75 of the key elements correctly.

Table 3. The amount of key recovered versus available ciphertext, transposition size

Amount of Ciphertext	SA (Simulated Annealing)	GA (Genetic Algorithm)	TS (Tabu Search)	LFA
200	5	7	4.75	4.5
400	10.75	11.5	9.25	9.25
1000	12.75	13.25	13	12.5

Table 4. The amount of recovered key versus size of transposition, 1000

Amount of Ciphertext	SA	GA	TS	LFA
15	12.75	13.25	13	12.5
20	16.5	17	16.75	16.25
25	20.15	21.5	21	20.15
30	25	25.25	25.5	24.75

IX. DISCUSSION

The results of our experiments shed light on the performance of various algorithms in attacking the transposition cipher. The evaluation of these algorithms was carried out on ciphertexts of different lengths, and the key size was set at 15 for analysis.

The findings presented in Table 1 highlight the impact of different ciphertext rates on attacks. As expected, the algorithms performed differently depending on the amount of ciphertext available. It is clear that the amount of ciphertext has a direct impact on the success of the attacks. More comprehensive data sets tend to provide the best results for well-established milestones. This highlights the importance of having enough ciphertext for cryptographic attacks, especially in transposition ciphers.

Table 2 focuses on the results of the column-placement cipher with a key matrix of time 30. It seems that the improved algorithm of Cuckoo search is the most effective approach, with average key elements of 24.75 well located, which outperforms all other cipher cracking techniques.

In brief, subsequent studies could recognize the ideal key size for various ciphers and analyzing algorithm efficacy across different key lengths. The represented research demonstrates the varying effectiveness of algorithms against transfer cipher attacks. The Cuckoo search algorithm, that is enhanced, exhibits promising outcomes.

A. COMPUTATIONAL COST ANALYSIS

To assess the practicality of implementing the proposed Lévy Flight Attack (LFA) algorithm in wireless sensor network (WSN) environments, we evaluated the computational cost of the algorithm in terms of **execution time** and **memory usage** during the decryption process. The evaluation was conducted using MATLAB R2023a on a system with an Intel Core i5 processor (2.5 GHz) and 8 GB RAM. Ciphertexts of lengths 200, 400, and 1000 characters were tested, with key lengths ranging from 10 to 30.

EXECUTION TIME

The average execution time for each run (over 50 iterations) was measured. The results show that the execution time grows approximately linearly with the ciphertext size, as summarized in Table 4:

Ciphertext Length	Average Key Length	Execution Time (s)
200 characters	15	1.62
400 characters	20	2.87
1000 characters	30	5.21

The time complexity of the algorithm is dominated by the key generation phase and dictionary-based fitness evaluation, both of which are computationally intensive. However, by incorporating a pool-based selection strategy and Lévy flight-based exploration, the algorithm achieves optimal key recovery faster than exhaustive search methods.

MEMORY USAGE

Memory consumption was monitored using MATLAB's profiling tools. The peak memory usage for each ciphertext length was recorded:

Ciphertext Length	Peak Memory Usage (MB)
200 characters	93 MB
400 characters	126 MB
1000 characters	194 MB

The increase in memory footprint is primarily attributed to storing intermediate generations of keys and the lookup operations with the dictionary.

FEASIBILITY IN WSN/IOT ENVIRONMENTS

Given the measured overhead, we conclude that deploying the full LFA algorithm on resource-constrained IoT sensor nodes is not feasible due to limited memory and processing

capabilities. Instead, we recommend offloading the decryption task to **edge** computing nodes or gateway devices within the WSN infrastructure. These nodes possess sufficient resources to handle the computational demands of the LFA algorithm while still maintaining the integrity and security of the overall system.

B. IMPACT OF COMPUTATIONAL OVERLOAD ON IOT SENSORS AND NETWORKS

The computational requirements of the proposed Lévy Flight Attack (LFA) algorithm present a challenge when considering its deployment on common IoT sensors and embedded nodes in wireless sensor networks (WSNs). Most sensor nodes are constrained in terms of processing power, memory capacity, and energy availability, typically equipped with microcontrollers operating below 200 MHz and possessing only a few kilobytes of RAM.

Our experiments (Section 8.1) show that the LFA algorithm, even when optimized, demands over 90 MB of memory and execution times of several seconds even for modest ciphertext sizes. These figures far exceed the capabilities of typical low-power IoT nodes such as the TelosB, MicaZ, or ESP8266, which generally offer < 512 KB RAM and < 160 MHz CPU clock speed.

This computational overload compromises their usage in two primary ways:

1. **Resource Exhaustion and Node Instability:** Attempting to execute LFA on constrained nodes could lead to memory overflow, CPU saturation, and system crashes due to an inability to allocate space for key pools, ciphertext storage, and dictionary operations. This affects the stability and availability of the node, making it unreliable in real-time sensing tasks.
2. **Energy Drain and Reduced Network Lifetime:** Cryptographic operations are known to consume significant energy, particularly in optimization-based decryption tasks. The LFA's iterative nature, requiring multiple generations and comparisons per attack cycle, would result in accelerated battery depletion, thus reducing node and network lifespan, a critical limitation in battery-powered or remotely deployed sensor networks.

X. CONCLUSION

The experimental results illustrate the effective employment of Lévy Flight Attack (LFA) to decrypt ciphers' transposition in wireless sensor network settings. The presented algorithm has a number of remarkable features, such as good minimal globality and quick convergence. In comparison to previous approaches, it identifies the ideal key in a short time. Based on Lewy plane principles, the presented work is a potent strategy for tackling transposition ciphers. It also reveals that our method, with the use of LFA, outperforms other methods in recovering the number of keys correctly, demonstrating its robustness and efficiency in wireless sensor network WSN cases. The special benefits of the Lévy Flight Attack (LFA) method are emphasized by our method, which are speed, ability to find the best solution, and effective key recovery.

ACKNOWLEDGEMENTS

I thank Al-Iraqia University for supporting my research.

DECLARATIONS

Conflict of interest: The author have no competing interests to declare that are relevant to the content of this article.

References

[1] M. Mareli and B. Twala, "An adaptive Cuckoo search algorithm for optimisation," *Appl. Comput. Informatics*, vol. 14, no. 2, pp. 107–115, 2018. https://doi.org/10.1016/j.aci.2017.09.001.

[2] E. Valian, S. Mohanna, and S. Tavakoli, "Improved Cuckoo Search Algorithm for Global Optimization," *Int. J. Commun. Inf. Technol.*, vol. 1, pp. 31–44, 2011.

[3] L. Kanagasabai and B. Ravindhranath Reddy, "A new novel Cuckoo Search optimization algorithm for solving optimal reactive power dispatch problem," *Majlesi J. Energy Manage.*, vol. 4, no. 3, 2015.

[4] A. A. Mousa and M. A. El-Shorbagy, "Enhanced particle swarm optimization based local search for reactive power compensation problem," *Appl. Math.*, vol. 3, pp. 1276–1284, 2012. https://doi.org/10.4236/am.2012.330184.

[5] A. Dimovski and D. Gligoroski, "Attacks on the transposition ciphers using optimization heuristics," in *Proc. Int. Conf. Environmental Science and Technology*, 2003, pp. 1–4.

[6] S. M. Hameed and D. N. Hmood, "Particles swarm optimization for the cryptanalysis of transposition cipher," *J. Al-Nahrain Univ.*, vol. 13, no. 4, pp. 211–215, 2010. https://doi.org/10.22401/JNUS.13.4.30.

[7] A. K. Verma, M. Dave, and R. C. Joshi, "Genetic algorithm and tabu search attack on the mono-alphabetic substitution cipher in adhoc networks," *J. Comput. Sci.*, vol. 3, no. 3, pp. 134–137, 2007. https://doi.org/10.3844/jcssp.2007.134.137.

[8] M. D. Russell, J. A. Clark, and S. Stepney, "Making the most of two heuristics: Breaking transposition ciphers with ants," in *Proc. 2003 Congr. Evol. Comput. (CEC'03)*, vol. 4, 2003, pp. 2653–2658. https://doi.org/10.1109/CEC.2003.1299423.

[9] R. Toemeh and S. Arumugam, "Breaking transposition cipher with genetic algorithm," *Elektronika ir Elektrotechnika*, vol. 79, no. 7, pp. 75–78, 2007. https://doi.org/10.5755/j02.eie.10844.

[10] A. O. Salau, N. Marriwala, and M. Athaee, "Data security in wireless sensor networks: Attacks and countermeasures," in *Mobile Radio Communications and 5G Networks: Proc. MRCN 2020*. Singapore: Springer, 2021. https://doi.org/10.1007/978-981-15-7130-5_13.

[11] B. Al-Kasasbeh, "A novel secure transposition cipher technique using arbitrary zigzag patterns," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 1, 2022. https://doi.org/10.14569/IJACSA.2022.0130133.

[12] P. M. Priya, "Hybrid diagonal transposition cipher model for securing data in software defined networks," *Networks*, vol. 7, no. 4, 2022. https://doi.org/10.46243/jst.2022.v7.i04.pp60-87.

[13] A. Praveena and S. Smys, "Efficient cryptographic approach for data security in wireless sensor networks using MES VU," in *Proc. 2016 10th Int. Conf. Intelligent Systems and Control (ISCO)*, 2016, pp. 1–6. https://doi.org/10.1109/ISCO.2016.7726911.

[14] E. Yousefpoor, H. Barati, and A. Barati, "A hierarchical secure data aggregation method using the dragonfly algorithm in wireless sensor networks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 4, pp. 1917–1942, 2021. https://doi.org/10.1007/s12083-021-01116-3.

[15] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Computer Communications*, vol. 67, pp. 1–10, 2015. https://doi.org/10.1016/j.comcom.2015.06.004.

[16] B. E. Ujcich and W. H. Sanders, "Data protection intents for software-defined networking," in *Proc. 2019 IEEE Conf. Network Softwarization (NetSoft)*, 2019, pp. 271–275. https://doi.org/10.1109/NETSOFT.2019.8806684.

[17] D. Mostovich, P. Fabrikantov, A. Vladyko, and M. Buinevich, "High-level vulnerabilities of software-defined networking in the context of telecommunication network evolution," in *Proc. 2017 IEEE Conf. Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2017, pp. 184–186. https://doi.org/10.1109/EIConRus.2017.7910524.

[18] A. Pradhan and R. Mathew, "Solutions to vulnerabilities and threats in software defined networking (SDN)," *Procedia Computer Science*, vol. 171, pp. 2581–2589, 2020. https://doi.org/10.1016/j.procs.2020.04.280.

[19] N. Anand, S. Babu, and B. S. Manoj, "On detecting compromised controller in software defined networks," *Computer Networks*, vol. 137, pp. 107–118, 2018. https://doi.org/10.1016/j.comnet.2018.03.021.

[20] L. J. Jagadeesan and V. Mendiratta, "Programming the network: Application software faults in software-defined networks," in *Proc. 2016 IEEE Int. Symp. Software Reliability Engineering Workshops (ISSREW)*, 2016, pp. 125–131. https://doi.org/10.1109/ISSREW.2016.23.

[21] A. C. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: A software-defined wireless sensor network," *Computer Networks*, vol. 159, pp. 84–95, 2019. https://doi.org/10.1016/j.comnet.2019.04.029.

[22] A. Almohaimeed and A. Asaduzzaman, "A novel moving target defense technique to secure communication links in software-defined networks," in *Proc. IEEE 5th Conf. Mobile and Secure Services (MobiSecServ)*, Mar. 2019, pp. 1–4. https://doi.org/10.1109/MOBISECSERV.2019.8686530.

[23] P. M. Priya, "Secure defense mechanism against data leakage and distributed denial of service attacks in software defined networks," in *Proc. 2019 Int. Conf. Communication and Electronics Systems (ICCES)*, Jul. 2019, pp. 378–382. https://doi.org/10.1109/ICCES45898.2019.9002273.

[24] P. MohanaPriya and S. M. Shalinie, "Restricted Boltzmann machine-based cognitive protocol for secure routing in software defined wireless networks," *IET Networks*, vol. 6, no. 6, pp. 162–168, 2017. https://doi.org/10.1049/iet-net.2017.0054.

[25] K. P. Naik and U. R. Joshi, "Performance analysis of constrained application protocol using Cooja simulator in Contiki OS," in *Proc. 2017 Int. Conf. Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, Jul. 2017, pp. 547–550. https://doi.org/10.1109/ICICICT1.2017.8342622.

***Sufyan Othman Zaben** holds a BSc in Software Engineering (2009) and an MSc in Software Systems (2022) from the University of Technology, Baghdad, Iraq. He specializes in machine learning, image processing, and classification, with a focus on medical imaging. Sufyan's expertise lies in developing advanced software solutions for complex problems in medical image analysis.*