ISSN: 2633-4828 Vol. 6 No.1, January, 2024

International Journal of Applied Engineering & Technology

# A PROPOSED LIGHTWEIGHT BLOCK CIPHER SYSTEM EMPLOYING DNA CODES

[1]Zainab Fawzi Abed and [2]Sawsan S. Abed
[1]Al-Iraqia University / College of Art / Baghdad- Iraq
[2]College of Education for Pure Science / Ibn – Al-Haithem / Department of Chemistry / Baghdad- Iraq
[1]zainab.f.abed@aliraqia.edu.iq and [2]sawsan.s.a@ihcoedu.uobaghdad.edu.iq

## ABSTRACT
As we continue to witness critical and rapid advancements in technology, it becomes increasingly important to *strengthen algorithms for information encryption and discover better ways to make encryption and the key utilized with it more difficult to break. One of the celebrated techniques used in encryption is a lightweight block cipher (LWBC) which is utilized with gadgets with constrained assets, LWBC is more often than not depicted as an encryption of asset-based devices commonly alluded to as RFID names and further detecting frameworks as points of reference. In this study a LWBC system is proposed, using DNA codes to improve security and generate an encryption key with a greater degree of complexity, the proposed method can be separated into two sections: the first is responsible for generating the key, and the second is for carrying out the encryption. Additionally, it employs specified operations such as substitution, map functions, and permutation. Avalanche effect, key sensitivity, execution time, encryption throughput, and memory consumption are used to assess the suggested system. Based on the results the proposed method works better than current encryption techniques Concerning memory usage, execution time, and, throughput, which makes it a desirable choice for data protection in resource-constrained environments.*

*Keywords: Cryptography, Lightweight encryption, Block Cipher, key generation, DNA codes, resource constrained devices*

## 1. INTRODUCTION
In the era of rapidly evolving technology, the need for secure and efficient data protection mechanisms has become paramount. Lightweight cryptography plays a crucial role in safeguarding sensitive information in devices with limited resources, like wireless sensor networks, RFID tags, and IoT devices [1,2]. These devices often have limited computational power, memory, and energy resources, necessitating cryptographic algorithms that are both secure and efficient [3].

Therefore, there is a pressing need to develop lightweight block cipher systems that not only prioritize efficiency but also ensure robust security. In recent years, the field of lightweight block cipher systems has witnessed rapid development, with various proposals and research focusing on enhancing the security level while minimizing the implementation cost [4].

The objective of this research is to explore the potential of using DNA codes as a basis for a lightweight block cipher system that leverages the unique properties of DNA coding to enhance security and performance [5]. DNA, or deoxyribose nucleic acid, is a molecule that carries genetic instructions in living organisms. Its structure, composed of four nucleobases (cytosine, guanine, adenine, and thymine), provides a rich foundation for developing cryptographic algorithms [6].

For the reason of its many uses, DNA coding provides fresh possibilities for cryptography. Because they can form self-assembly structures, the four nucleotide bases offer great tools for carrying out calculations. As a result, information security, biological sciences, and computer science have all come together to develop DNA cryptography [7].

DNA has proven to be a powerful cryptographic tool, using many of the latest DNA processing technologies for steganography, key creation, encryption, and cryptanalysis [8]. DNA encryption methods are preferred over traditional encryption methods because most cryptographic approaches have not yet been used in resource-constrained devices with limited capabilities or are likely to be broken by the next generation of quantum computers shortly. Due to the rapid advancement of technology and the rapid increase in threats, data transfer and storage for most traditional Lightweight cryptography and cryptographic algorithms have become more vulnerable to brute-force attacks [9,10].

Our proposal in this paper is a two-phase lightweight block cipher scheme: key generation and encryption. The key generation phase utilizes DNA-based operations, such as permutation, shift, and rotation, to generate a binary key. This approach introduces an additional layer of security by incorporating the unique characteristics of DNA into the key generation process.

The encryption phase involves a series of operations, including initial permutation, XORing with a constant and the generated key, application of map functions, S-Box substitution, and DNA coding. These operations are carefully designed to achieve a balance between security and efficiency, ensuring that the cipher is both resistant to cryptanalytic attacks and suitable for resource-constrained environments. The other sections of this paper are organized as follows: Section 2 includes presentations of relevant works; Section 3 includes the proposed system; Section 4 includes performance evaluation and results; and Section 5 the conclusions are provided.

## 2. RELATED WORK

Due to the growing prevalence of resource-constrained devices, lightweight cryptography has become increasingly important in recent years. Researchers have been studying different approaches to develop lightweight block ciphers that offer a good balance between efficiency and security. some of which are shown below:

In 2023, AL-Shargabi et al. [11] proposed a novel lightweight encryption algorithm based on DNA sequences for IoT devices. The algorithm generates a strong and completely random key using the DNA sequence, making it highly resistant to cracking. In 2023, Qaid et al. [12] presents a novel DNA-based encryption method that addresses the computational limitations of IoT devices while providing robust security for data transmission. The method is lightweight, efficient, and resistant to various attacks, making it a promising solution for securing IoT communications. In 2022, Al Abbas et al. [13] suggests a one-time board dynamic key created from stream cipher DNA to secure IoT applications on limited devices using a lightweight dynamic encryption technique. A proposed table is utilized to encrypt the text using an addition process by the algorithm, in addition to encrypting the key. The result is ciphertext that contains hidden decryption information. In 2021, Al-Husainy et al. [7] proposed a lightweight encryption system tailored for IoT devices with limited processing capabilities. It incorporates flexible block sizes, utilizes DNA sequences for encryption keys, and exhibits strong security features, The system also offers resistance against various cryptographic attacks, including brute-force attacks and differential power analysis. In 2018, Rana, S et al. [14] proposed a secure and efficient lightweight cryptographic algorithm for small computing devices with limited resources and capabilities, employing a custom substitution-permutation network and a modified Feistel architecture. The algorithm reduces processing cycles while providing sufficient security, offering promising performance for small computing devices with limited resources. In 2016, Suresh Babu et al. [15] proposed a biotic DNA-based secret key cryptographic mechanism that utilizes splicing systems and random multiple key sequences to enhance security against brute force attacks and chosen ciphertext attacks. It also introduces DNA-assembled public key cryptography and a double-binded encryption scheme for efficient storage, computation, and transmission of data.

## 3. The proposed system

The proposed system utilizes a lightweight block cipher algorithm to encrypt DNA sequences, providing enhanced security and privacy for DNA data transmission and storage. The system consists of the following components:

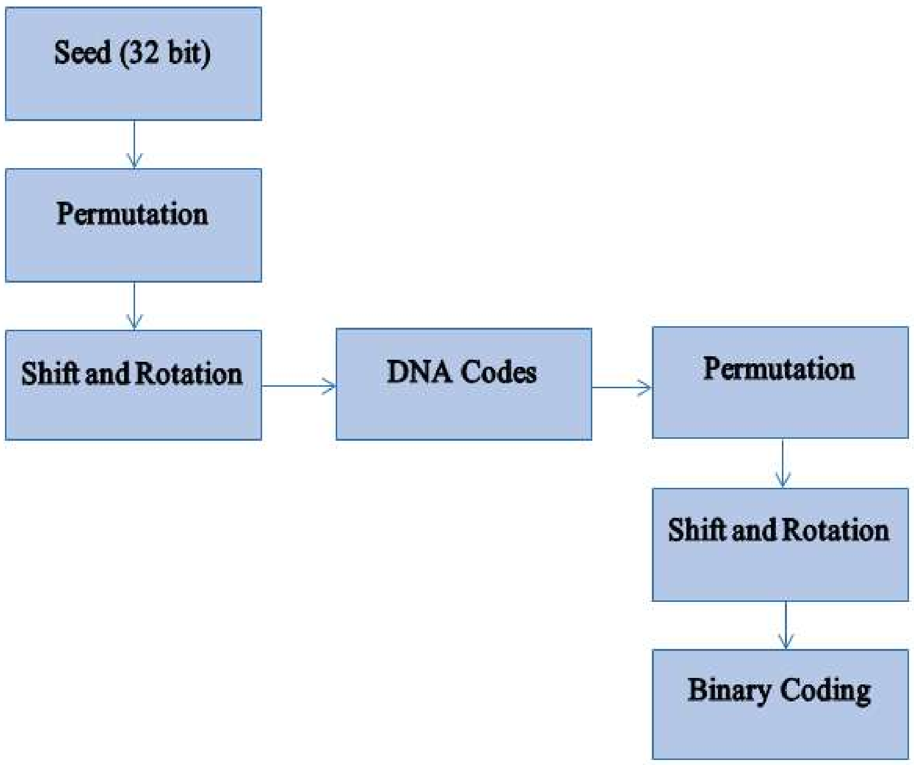

**Figure (1):** The Proposed Key Generation General Structure

| **Algorithm (1): Keys Generation General Structure** |
|---|
| **Input: Seed (32 bit), specified permutation locations of bits, shift and rotate number, specified permutation locations of DNA codes, shift and rotate number.** |
| **Output: Generated Binary key (32 bit).** |
| **Begin**<br>Step 1: Read the seed.<br>Step 2: Perform permutation to the seed (32 bit) using specified permutation locations of bits.<br>Step 3: Perform shifting and rotation using the shift and rotate number.<br>Step 4: Convert to DNA codes.<br>Step 5: Perform permutation to the DNA codes using specified permutation locations of DNA codes.<br>Step 6: Perform shift and rotation using shift and rotate number.<br>Step 7: Convert DNA codes to binary codes.<br>**End** |

The algorithm starts by taking a 32-bit seed as input and undergoes a permutation operation using specified permutation locations of bits. This means rearranging the bits of the seed based on the provided permutation locations. The permuted seed then undergoes a bit of shifting and rotation operation using the specified shift and rotation numbers. The result (a set of bits) is converted to DNA codes. In DNA cryptography, DNA codes are often used as an alternative representation, as shown in the table (1).

| Binary codes | DNA Codes |
|---|---|
| **00** | A |
| **01** | C |
| **10** | G |
| **11** | T |

**Table (1)** Convert binary codes to DNA codes

The DNA codes undergo a permutation operation using specified permutation locations of DNA codes. then it undergoes a shift and rotation operation using the specified shift and rotate numbers. DNA codes, is converted back to binary codes as shown in table (2). The final result is the generated binary key of 32 bits.

| DNA codes | Binary Codes |
|---|---|
| **A** | 00 |
| **C** | 01 |
| **G** | 10 |
| **T** | 11 |

**Table (2)** Convert DNA codes to binary codes

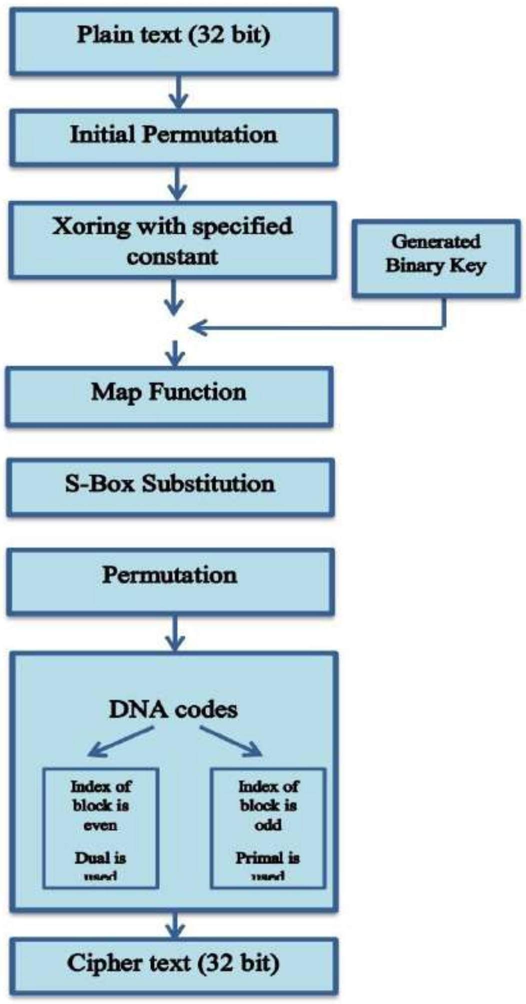

**Figure (2):** Proposed Light Weight Block Cipher System.

**Algorithm (2): Proposed Lightweight Block Cipher System**

**Input: plain text (32 bit), N (number of rounds), initial permutation locations, specified constant, generated key, map function specifications, S-Box specification, permutation specified locations, dual and primal DNA.**

**Output: Cipher text (32 bit).**

```
Begin
Step 1: Read the plain text.
Step 2: For i= 1 to N do
Step 3: Perform initial permutation using initial permutation locations.
Step 4: Xoring with specified constant.
Step 5: Xoring with the generated key.
Step 6: Applying map function specification to each block (8 bit).
Step 7: Applying substitution using 4 S-boxes.
Step 8: Perform permutation using specified permutation locations.
Step 9: End for
Step 10: Convert to DNA codes (The binary block will be divided into 2-bit blocks, if the block index is even then the Dual is used in coding, otherwise if the block index is odd then the primal is used.
Step 11: Convert to binary coding.
Step 12: Read the cipher text (32 bit).
End
```

The algorithm starts by taking a 32-bit plaintext value, and Perform the following steps for each of the N rounds.

1-Apply an initial permutation to the plaintext using a predefined permutation table. This permutation scrambles the bits of the plaintext.

2-XOR the permuted plaintext with a specified constant value.

3-XOR the result of Step 2 with the generated key, The generated key serves as a unique and secret parameter for each encryption, ensuring that even if the same initial permutation result is obtained, the ciphertext will be different each time due to the influence of the generated key.

4-Apply a map function to each 8-bits block of the result of Step 3. The map function consists of a series of bitwise operations, such as AND, OR, and XOR, designed to further scramble the data.

5-Apply substitution to the result of Step 4 using four different S-boxes (substitution boxes). Each S-box is an 8x8 table that maps each 8-bit input to a different 8-bit output. This substitution process helps to introduce nonlinearity into the cipher.

6-Apply a permutation to the result of Step 5. This permutation is different from the initial permutation and helps to further scramble the data.

After completing all rounds, the result is in binary form. and then we Convert it into DNA codes. This is done by dividing the binary block into 2-bit blocks and the algorithm checks whether the block index is even or odd. The block index refers to the position of the 2-bit block, If the block index is even, the algorithm uses the "Dual" set for encoding. If the block index is odd, the algorithm uses the "Primal" set for encoding as shown in the tables (3) and (4). After that, we Convert the DNA codes back into a 32-bit binary sequence, which represent the cipher text.

| Dual set | |
|---|---|
| Binary codes | DNA codes |
| 00 | A |
| 01 | T |
| 10 | C |
| 11 | G |

**Table (3)** Dual set

| Primal set | |
|---|---|
| Binary codes | DNA codes |
| 00 | G |
| 01 | C |
| 10 | T |
| 11 | A |

**Table (4)** Primal set

## 4. **Performance Evaluation and Results**

The process of evaluating and analyzing the effectiveness, efficiency and various aspects of an encryption algorithm is done by using various metrics and comparing them with other encryption algorithms, The proposed algorithm's performance is assessed considering the following factors: Avalanche Effect , key sensitivity, execution time, encryption throughput, and memory consumption using a PC running Windows 10 with an Intel Core i7, 8 GB of RAM, and Python 3.11.4.

### **4.1-Avalanche Effect:**

An advantageous characteristic of block ciphers is the avalanche effect, which states that a minor alteration in the input (plaintext) should produce a noticeably different output (ciphertext). Analyzing the avalanche effect in the context of lightweight block ciphers usually entails determining how modifications to the input bits impact the output bits, as shown in table (5).

| The sequence of the changed bit | plaintext | ciphertext |
|---|---|---|
| - | 11111111111111000000000000110110 | 01011010101010101010101000110100 |
| 10 | 11111111101111000000000000110110 | 00110110100101001110001111101110 |
| 4 | 11101111111111000000000000110110 | 10010010010010010110010100001001 |

**Table (5)** avalanche effect

4.2-**Key Sensitivity:** describes how sensitive a cryptographic algorithm is to modifications in its cryptographic key. An entirely different output ought to come from a minor adjustment to the key. Since even a slight alteration to the key should result in a substantial alteration to the encrypted data, key sensitivity is essential to the security of cryptographic systems and if a single bit change in the key results in a 50% change in the ciphertext, the algorithm is considered perfect .

4.3-**Execution Time:** The execution time of an encryption algorithm is a critical factor to take into account, particularly in applications where rapid encryption and decryption are required, like high-speed data processing settings or real-time communication systems.

4.4-**Encryption Throughput:** refers to the speed at which data can be encrypted using a specific encryption algorithm or cryptographic system. This measurement is usually expressed in terms of data processed per unit of time such as, kilobytes per second (KB/s) or megabytes per second (MB/s).

4.5-**Memory Consumption:** is the term used to describe how much computer memory an algorithm uses while it is running. It's critical to use memory efficiently, particularly in situations where memory is limited, Lightweight encryption algorithms are designed to use as little memory as possible, which makes them appropriate for resource-constrained devices such embedded systems, Internet of Things (IoT) devices, and other devices with low memory capacities.

A Comparison of the proposed algorithm with Encryption algorithms is listed below tables:

| Encryption algorithms | Block size (bit) | Key size (bit) |
|---|---|---|
| **HIGHT** | 64 | 128 |
| **KATAN** | 32 | 80 |
| **SIMON** | 32 | 64 |
| **PRINCE** | 64 | 128 |
| **Proposed algorithm** | 32 | 32 |

**Table (6)** Block size and Key size of Encryption algorithms

| Encryption algorithms | Execution time (sec) |
|---|---|
| **HIGHT** | 0.340 |
| **KATAN** | 0.157 |
| **SIMON** | 0.239 |
| **PRINCE** | 0.195 |
| **Proposed algorithm** | 0. 040 |

**Table (7)** result of Execution time

| Encryption algorithms | Encryption Throughput (KB/sec) |
|---|---|
| **HIGHT** | 424 |
| **KATAN** | 628 |
| **SIMON** | 701 |
| **PRINCE** | 538 |
| **Proposed algorithm** | 1016 |

**Table (8)** result of Encryption Throughput

| Encryption algorithms | Memory consumption (KB) |
|---|---|
| **HIGHT** | 981 |
| **KATAN** | 870 |
| **SIMON** | 363 |
| **PRINCE** | 1022 |
| **Proposed algorithm** | 320 |

**Table (9)** result of Memory consumption

## 5. CONCLUSIONS

Finding a secure, quick, and appropriate algorithm for resource-constrained devices has become recognized as extremely important. The proposed cipher offers several advantages. One of it the use of DNA codes which provides a vast space for encryption as DNA sequences can represent a large number of possible combinations, using DNA codes opens up a wide range of encryption possibilities. As a result, cracking the cipher text becomes more difficult and more resilient to brute-force attacks. The produced key and the XOR operations with predefined constants create diffusion and confusion, making it difficult for an adversary to decipher the cipher text and extract any useful information. The choice between Dual and Primal sets adds variability and complexity to the encoding process, contributing to the security and diversity of the encoded DNA sequence. The suggested LWBC scheme shows great promise for improving data security and privacy in resource-constrained environments, By using DNA coding for key generation and encryption, the system achieves a balance between security and efficiency. The algorithm's Avalanche test reveals that when we changed the bit number 10 in plaintext, there was a 17-bit change in cipher text, and when we changed bit 4 there was a change of 19-bits in the cipher text, The same applies to key sensitivity. In comparison to current encryption techniques, the evaluation results reveal how effective the suggested algorithm is in terms of execution time, encryption throughput, and memory consumption.

## REFERENCES

[1] Azaïs, T., De Meyer, H., & Vandewalle, J. (2020). Lightweight cryptography for the internet of things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 22 (3), 1868-1893.

[2] Yeoh W.-Z. J.S. Teh, and M.I.S.B.M. Sazali, μ 2, (2020). A Lightweight Block Cipher, in Computational Science and Technology, Springer. p. 281–290.

[3] Li, Y. (2015). A survey of lightweight block ciphers. In 2015 IEEE International Conference on Computer and Information Technology (CIT) (pp. 1052-1057). IEEE.

[4] Qureshi, M. A., Hassan, M. U., & Mehmood, A. (2021). A comprehensive survey on lightweight block ciphers for resource-constrained devices. IEEE Access, 9, 102395-102426.

[5] ZAID M. JAWAD KUBBA; HAIDER K. HOOMOD, (2020). DEVELOPING A LIGHTWEIGHT CRYPTOGRAPHIC ALGORITHM BASED ON DNA COMPUTING. AIP CONFERENCE PROCEEDINGS. 2290(1).

[6] Goyat, S., & Jain, S. (2016). A secure cryptographic cloud communication using DNA cryptographic technique. In 2016 International Conference on Inventive Computation Technologies (ICICT) (Vol. 3, pp. 1-8). IEEE.

[7] Mohammed Abbas Fadhil Al-Husainy, Bassam Al-Shargabi, Shadi Aljawarneh, (2021). Lightweight cryptography system for IoT devices using DNA. Computers and Electrical Engineering, 95: 107418.

[8] Tornea, O. (2013). Contributions to DNA cryptography: Applications to text and image secure transmission. PhD diss., Université Nice Sophia Antipolis; Universitatea tehnica (Cluj-Napoca, Roumanie).

[9] Elsaid, S. A., Alotaibi, E. R., & Alsaleh, S. (2022). A robust hybrid cryptosystem based on DNA and hyperchaotic for images encryption. Multimedia Tools and Applications, 82, 1–25.

[10] Khalifa, M., Algarni, F., Khan, M. A., Ullah, A., & Aloufi, K. (2021). A lightweight cryptography (LWC) framework to secure memory heap in internet of things. Alexandria Engineering Journal, 60(1), 1489–1497.

[11] Bassam AL-Shargabi, Ahmad Dar Assi, (2023). A modified lightweight DNA-based cryptography method for internet of things devices. Expert Systems, 40 (6), 1-12.

[12] Gamil R. S. Qaid, Nadhem Sultan Ebrahim, (2023). A Lightweight Cryptographic Algorithm Based on DNA Computing for IoT Devices. Security and Communication Networks.

[13] Al Abbas, A. A. M., & Ibraheem, N. B. (2022). Using DNA In Adynamic Lightweight Algorithm For Stream Cipher In An IoT Application. In International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) (pp. 232-240). IEEE.

[14] Rana, S., Hossain, S., Shoun, H. I., & Kashem, M. A. (2018). An effective lightweight cryptographic algorithm to secure resource-constrained devices. International Journal of Advanced Computer Science and Applications, 9(11).

[15] Suresh Babu, E., Nagaraju, C., & Krishna Prasad, M. H. M. (2016). Light-Weighted DNA-Based Cryptographic Mechanism Against Chosen Cipher Text Attacks. Advanced Computing and Systems for Security, 1, 123-144.