MINAR International Journal of Applied Sciences and Technology
ISSN: 2717-8234
Article type: Research Article

Received: 11/09/2023 **Accepted:** 01/11/2023 **Published:** 01/12/2023

A NEW PROPOSED LIGHTWEIGHT CIPHER

Auday H. AL-WATTAR [1]
University of Mosul– Iraq

Abstract

Cryptography algorithms nowadays focus on hardware optimization for computers. This is super important for limited devices like those with small size, less power, and weak computing. Lightweight cryptography develops crypto methods for economical systems. This study introduces a brand-new lightweight block cipher based on bio features for ade-quate data security. A carefully planned sequence of analysis methods supports this innovation theoretically. The core- involves complex calculations to evaluate the proposed crypto approach. As detailed, extensive testing proves the tech's robust security.

The essay analyzes frequency analysis, frequency within block analysis, and the run test for a complete re-view. These tools provideadvanced info about the algorithm's resilie-nce against specific crypto vulnerabilities. Frequency analysis measures value distribution in the cipher, potentially revealing weaknesses. Frequency within block analysis shows intricate- value patterns inside discrete blocks, indicating algorithm behavior in certain situations. The- run test is crucial for determining the algorithm's avalanche effect. Carefully evaluating sequential value distribution tests the algorithm's innate input change sensitivity a must for crypto security.

Academic ideas often analyze encryption methods. This paper looks at how well a new lightweight code works. It checks core security steps like how often symbols repeat, how well symbols mix in blocks, and how random the code is. Through strict testing, this new code proves it is useful and secure. This makes it an important new way to protect devices with low power.

Keywords: Security, Cryptography, Lightweight Encryption.

http://dx.doi.org/10.47832/2717-8234.17.14

1 Ahsa.alwattar@uomosul.edu.iq

Introduction

The necessity of securing sensitive data has been crucial throughout history, gaining even greater significance in contemporary society due to computers permeating many aspects of modern living and business.This matter is exacerbated by the rise of artificial intelligence, vastly transforming how confidential information is sheltered. Computer security constitutes a comprehensive and intricate field protecting computing systems and their holdings from unauthorized access and potential cyber risks. The sine qua non of protected interaction, indispensable for maintaining integrity in data exchange, is emphasized in an era when the Internet plays a core role in electronic commerce and financial services. It has become absolutely critical as an extremely secure mode of interface.[1, 2].

Throughout history, encryption has served a pivotal function in fulfilling protection necessities by using numerical strategies to code and decode information. However, technological progress continues unabated, transitioning us from traditional computer systems to a profusion of small and miniature devices that can stand alone or form core-components of larger assemblies. This proliferation has brought new challenges for security protocols to address. These devices, crucial in innovative applications like health tracking and autonomous driving, question the effectiveness of traditional cryptographic methods, represented by the Advanced Encryption Standard (AES), particularly in settings with limited resources, such as radio frequency identification (RFID) tags and sensor networks[3, 4].

In recent years, there has been an observable increase in the use of distant computer systems with limited resources for cryptographic implementations [5]. This article proposes a new lightweight cipher based on a proposed coding method. This paper is organized into sections: Section one includes the introduction. Section two consists of an explanation of the proposed method. Section three contains an enumeration and examination of outcomes. Furthermore, the final section constitutes the judgment.

The cryptography

Cryptography, occasionally termed encryption, is an intricate process that encodes clear logical data into cryptographically encrypted ciphertext and back again. This cryptographic technology guarantes secure and private-correspondence between two or more organizations, shielding it from any eavesdroppers. Functional apparatuses for cryptographic encryption methods are crucial in attaining several security goals. Cryptographic applications are extensively utilized and significant in several facets of contemporary existence. The widespread incorporation of these technologies is seen in their utilization by numerous businesses and inclusion in various goods, resulting in significant enhancement of safeguarding sensitive information and upholding security protocols in diverse situations. Cryptography technology ensures secure and confidential communication between two or more firms, protecting it from any eavesdroppers. Functional apparatuses for cryptographic encryption methods are crucial in attaining several security goals. Cryptographic applications are extensively utilized and significant in several facets of contemporary existence. The widespread incorporation of these technologies is seen in their utilization by numerous businesses and inclusion in various goods, resulting in significant enhancement of safeguarding sensitive information and upholding security protocols in diverse situations[6-9].

Lightweight encryption

Conventional cryptography is suitable for security for servers, desktops, tablets, and smartphones. However, it is inapplicable to embedded systems, RFID, and sensor networks, especially after the prevalence of the so-called Internet of Things (IoT). These chips are much lighter and lead industrial, critical operations than in past decades; they are processed, stored, and sent privately, sensibly, and critically, so security is an endless challenge to protect data against any vulnerability. This challenge only arises when resources are restricted and portable; the lightweight LWC protection mechanism is the solution. It is a cryptographic subfield; "Lightweight" does not imply weakness. However, it could become less robust, less abusive, and less characteristic[10]. In recent years, lightweight cryptography has been one of the urgent subjects in information security. Many lightweight standardized algorithms have been published to accomplish multiple security tasks. The deployment of tiny computers with limited cryptographic resources has increased in the last few years. Many lightweight algorithms have been published, standardized, and used in many aspects of life. It is known that the purpose of lightweight cryptography is to transfer from general-purpose computers to resource-restricted ones[11]. This paper proposed a new lightweight symmetric block cipher based on some proposed coding procedures to achieve encryption.

2. The Proposed Method

This section includes a description of the method's working mechanism. The plaintext is treated as a block, and it is called "Plaintext." The sequence of the algorithm work will follow the following steps:

Step 1:

The block (Plaintext) is split into the left (L) and right (R) parts, each of which has the same size.

Step 2:

The two parts (L and R) are translated into special coding; four characters (A, B, C, and D) represent this encoding. Each of these characters will be called the foundation. That means that those characters represent the data of each half. So, the entire plaintext will turn into ABCD DDACB BBCDA ABA .... foundations are only a set or sequence, according to block length, as follows for form and block: -

A= 00, B=01, C=10, D=11.

Step 3:

This step splits the blocks resulting from step 2 into two halves as in the following: -

The left block is subdivided into two halves of the same size (LLS) and (LRS), and the right block is also divided into two halves of the same size (RLS) and (RRS).

Step 4:

This stage involves a method of permutation for the resulting halves of step 3 as follows:

LLS → RLS1, LRS→ LLS1, RLS→RRS1, and RRS → LRS1.

That means the (LLS) block is relocated to the (RLS1) block, the (LRS) block is transferred to the (LLS1) block, the (RLS) block is relocated to the (RRS1) block, and finally, the (RRS) block is relocated to (LRS1) block. Accordingly, the blocks will take the following sequence:

LLS1- LRS1 -RLS1- RRS1,

This step provides a permutation transposition on the whole block.

Step 5: -

This step involves the following actions: -

1. Combining the two left blocks (LLS1) and (LRS1) to create a left part.
2. Combining the two right blocks (RLS1) and (RRS1) to create the right part.
3. As in step 2, the resulting two parts are encoded with different codes as

B=00, D=01, A=10, and C=11, for both sections.

This step acts as a substitution process for the data of the block.

Step 6:-

This step splits the blocks resulting from step 2 into two halves as in the following:

The left block is subdivided into two halves of the same size (LLS1) and (LRS1), and the right block is also divided into two halves of the same size (RLS1) and (RRS1).

Step 7:-

Besides the permutation process, this step involves using two pseudo-character slices (S1-L) and (S1-R), which can be the algorithm's key. That can be accomplished as follows. Note that the pseudo-character slices consist of the same four letters, but they are randomly generated and have the same size as the algorithm's halves.

LLS2 = $\overline{\text{LRS1}}$ + S1-R, LRS2 = $\overline{\text{RLS1}}$, RLS2= $\overline{\text{LLS1}}$+ S1-L, RRS2= $\overline{\text{RRS1}}$ + S1-R

The two segments (S1-R) and (S2-L) are used as a key.

Step 8:-

This step involves the following actions: -

1. Combining the two left blocks (LLS2) and (LRS2) to create a left part.
2. Combining the two right blocks (RLS2) and (RRS2) to create the right part.
3. As in step 2, the resulting two parts are encoded with different codes as

D=00, C=01, B=10, and A=11 for both sections.

This step acts as a substitution process for the data of the block.

Step 9: -

This step splits the blocks resulting from step 8 into two halves as in the following:

The left block is subdivided into two halves of the same size ($\overline{LLS2}$) and ($\overline{LRS2}$), and the right block s also divided into two halves of the same size ($\overline{RLS2}$) and ($\overline{RRS2}$)

Step 10: -

This stage involves a method of permutation for the resulting halves of step 9 as follows: -

$\overline{LLS2}$ → RLS3, $\overline{LRS2}$→ RRS3, $\overline{RLS2}$→LRS3, and $\overline{RRS2}$→ LLS3. That means the ($\overline{LLS2}$) block is relocated to the (RLS3) block, the (($\overline{LRS2}$) block is transferred to the (RRS3) block, the ($\overline{RLS2}$) block is relocated to the (LRS3) block, and finally, the ($\overline{RRS2}$) block is relocated to (LLS3) block. Accordingly, the blocks will take the following sequence:

LLS3- LRS3 -RLS3- RRS3.

This step alters the blocks, presenting a permutation transposition on the entire block.

Step 11:-

This step involves the following actions: -

1. Combining the two left blocks (LLS3) and (LRS3) to create a left part.
2. Combining the two right blocks (RLS3) and (RRS3) to create the correct part.

3. As in step 2, the resulting two parts are encoded with different codes as

C=00, A=01, D=10, and B=11 for both sections. This step acts as a substitution process for the data of the block. The resulting test will be combined to form the ciphertext; the whole process for encryption can be summarized in Figure (1)

The decryption method is the exact opposite in the encryption process from bottom to top until the plaintext is obtained. All steps are invertible from top to bottom. S1-R and S1-L size should match LLS, and RRS size and LLS1, LRS1, RLS1, RLS1, (LLS1)¯, (LRS1)¯, (RLS1)¯ ,(RRS1)¯,LLS2, LRS2, RLS2,RRS2, (LLS2)¯, (LRS2)¯, (RLS2)¯ ,(RRS2)¯, ,LLS3, LRS3, RLS3, and RRS3 block size, and S1-L size should be the same. The size of the plaintext will be precisely the size of the ciphertext. The S1-R and the S1-L could be any random (A, B, C, and D)-character segment converted to digital form.

Both encryption and decryption processes required a few procedures only. So, in terms of calculations, it is considered simple, but in terms of security, it includes all the iterated SPN system concepts.

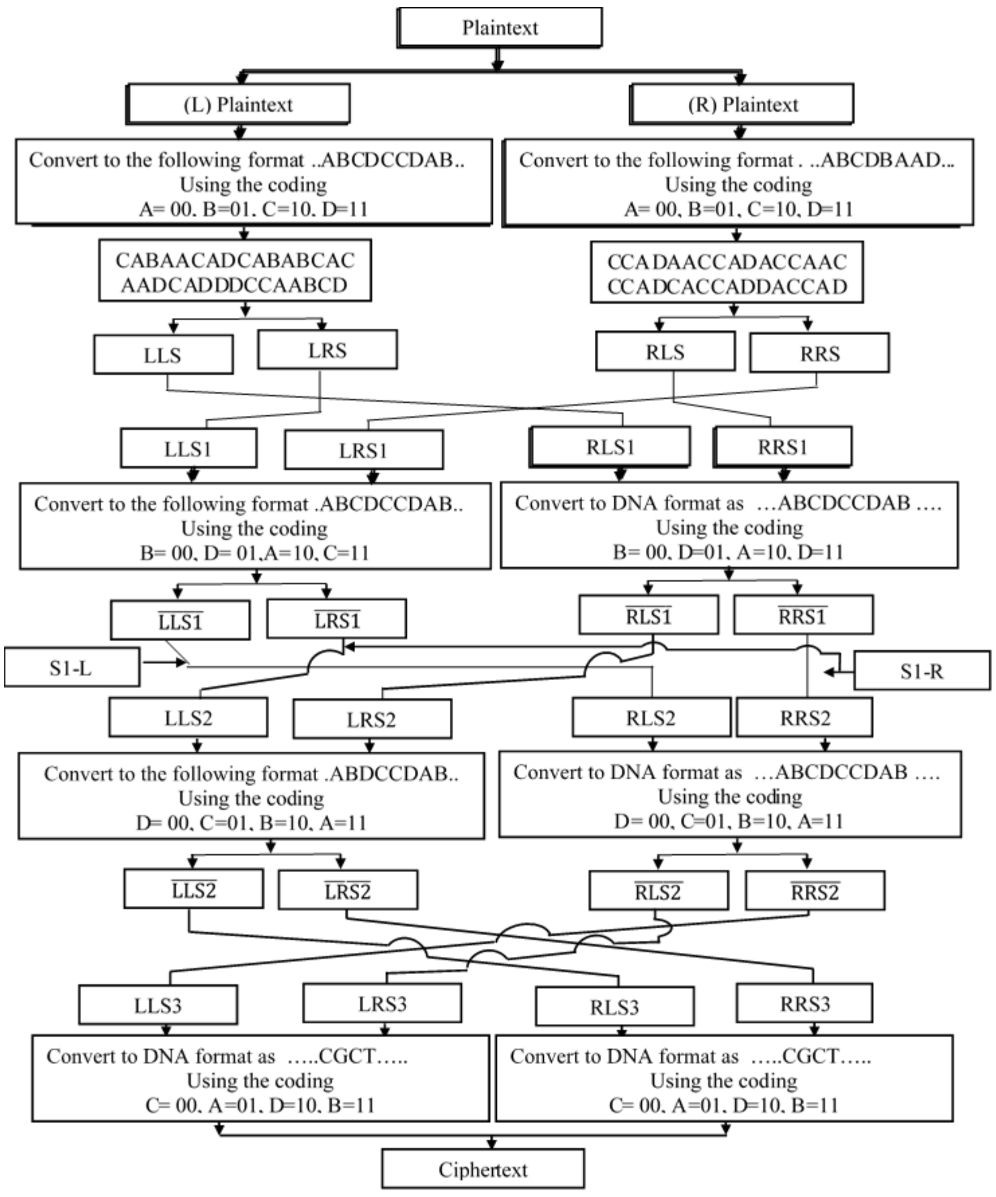
Plaintext
(L) Plaintext
(R) Plaintext
Convert to the following format ..ABCDCCDAB.. Using the coding A= 00, B=01, C=10, D=11
Convert to the following format . ..ABCDBAAD... Using the coding A= 00, B=01, C=10, D=11
CABAACADCABABCAC AADCADDDCCAABCD
CCADAACCADACCAAC CCADCACCADDACCAD
LLS
LRS
RLS
RRS
LLS1
LRS1
RLS1
RRS1
Convert to the following format .ABCDCCDAB.. Using the coding B= 00, D= 01,A=10, C=11
Convert to DNA format as …ABCDCCDAB …. Using the coding B= 00, D=01, A=10, D=11
$\overline{\text{LLS1}}$
$\overline{\text{LRS1}}$
$\overline{\text{RLS1}}$
$\overline{\text{RRS1}}$
S1-L
S1-R
LLS2
LRS2
RLS2
RRS2
Convert to the following format .ABDCCDAB.. Using the coding D= 00, C=01, B=10, A=11
Convert to DNA format as …ABCDCCDAB …. Using the coding D= 00, C=01, B=10, A=11
$\overline{\text{LLS2}}$
$\overline{\text{LRS2}}$
$\overline{\text{RLS2}}$
$\overline{\text{RRS2}}$
LLS3
LRS3
RLS3
RRS3
Convert to DNA format as …..CGCT….. Using the coding C= 00, A=01, D=10, B=11
Convert to DNA format as …..CGCT….. Using the coding C= 00, A=01, D=10, B=11
Ciphertext

Figure 1: The process of encryption using the proposed method

3. Results and Discussion

The proposed encryption method's simple requirements make it suitable for low-resource equipment since it does not include just a few simple processes. Both substitution and permutation methods are achieved in a simple form.

Splitting the plaintext block into two parts (L and R) and encoding using a unique

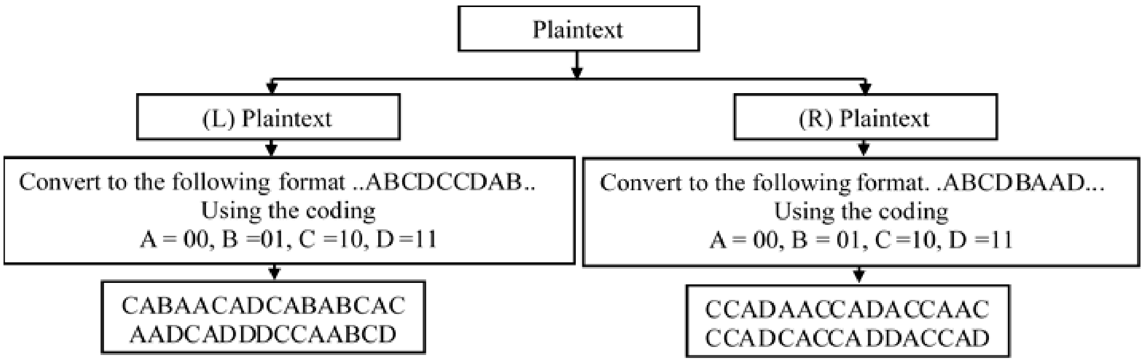
Plaintext
(L) Plaintext
(R) Plaintext
Convert to the following format ..ABCDCCDAB.. Using the coding A = 00, B =01, C =10, D =11
Convert to the following format. .ABCDBAAD... Using the coding A = 00, B = 01, C =10, D =11
CABAACADCABABCAC AADCADDDCCAABCD
CCADAACCADACCAAC CCADCACCADDACCAD

Figure 2: Split the Plaintext into two blocks and transform it into specific format with a particular coding

foundation code is considered a substitution and coding process simultaneously (Figure 2).

This process is repeated four times with different character codes for each one. After merging both (LLS1, LRS1, RLS1, and RRS1) blocks to produce the left and right blocks, respectively, these two blocks were encoded using different foundation coding. The exact process for (LLS2, LRS2, RLS2, and RRS2) with a new foundation coding was achieved. Finally, a new foundation coding is done for (LLS3, LRS3, RLS3, and RRS3) blocks. As shown in Figure (3).

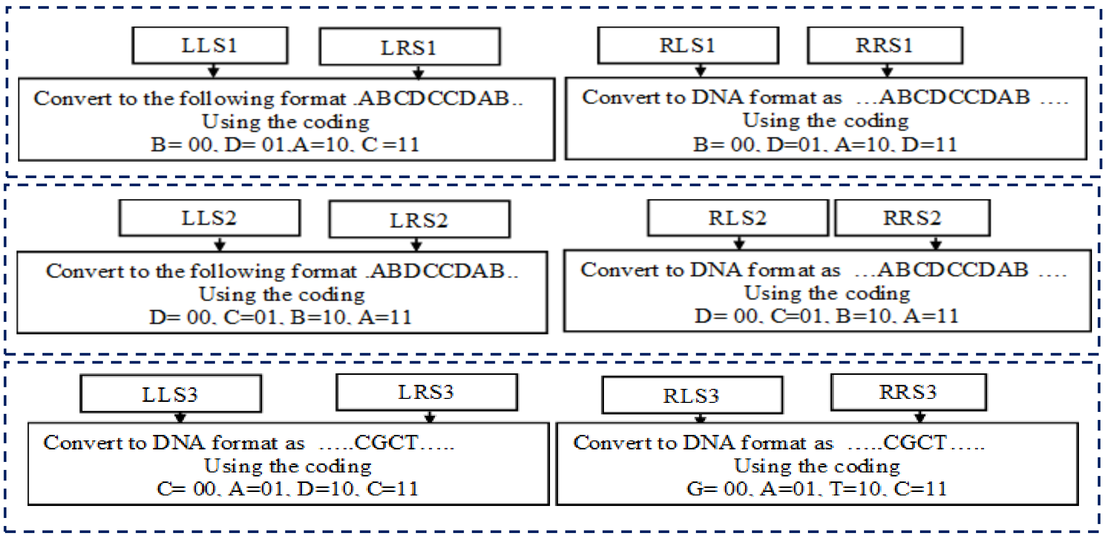
LLS1
LRS1
RLS1
RRS1
Convert to the following format .ABCDCCDAB.. Using the coding B= 00. D= 01.A=10. C =11
Convert to DNA format as ...ABCDCCDAB .... Using the coding B= 00. D=01. A=10. D=11
LLS2
LRS2
RLS2
RRS2
Convert to the following format .ABDCCDAB.. Using the coding D= 00. C=01. B=10. A=11
Convert to DNA format as ...ABCDCCDAB .... Using the coding D= 00. C=01. B=10. A=11
LLS3
LRS3
RLS3
RRS3
Convert to DNA format as .....CGCT..... Using the coding C= 00. A=01. D=10. C=11
Convert to DNA format as .....CGCT..... Using the coding G= 00. A=01. T=10. C=11

Figure 3: Split the Plaintext into two blocks and transform it into specific format with a particular coding of foundations

These multiple encoding processes give vital substitution processes for the proposed algorithm. The dividing of the blocks and altering these blocks provide a reasonable degree of complexity to the ciphertext.

The use of the two segments (S1-L) and (S1-R) as a key gives very high security to the algorithm and enhances its encryption/decryption ability, as shown in Figure (4). Using two separate segments as a key makes guessing the key incredibly difficult if possible, and if we know that a single key is as long as the block, then it's challenging to guess the key. For example, if the block length is 128 bits, then the guessing process requires twice the block size, which means 256 bits, so breaking a brute force symmetric 256-bit key takes 2^{128} times more power than a 128-bit key, and if know that A supercomputer with the fastest speed in 2019 has 100 PetaFLOPS [6]. Theoretically, this could scan 100 million (10^{14}) AES keys per second, but it also takes 3.67 to exhaust the 256-bit key area for a further 10^{55} years.[12].

In addition to the above, the utilizing of two random segments as a key ensures a large amount of randomness needed to optimize algorithm security and achieve the highest degree of confidentiality, as generating foundation segments at a ratio of 25% for each foundation means producing data at a total random rate, and this was investigated using the NSIT standard based on [13]. Besides the anonymity of these segments only known to the encoder or the sender, this randomness maximizes the algorithm's security, doubles the chance of breaking the algorithm, and makes it very difficult for the attacker.

The proposed method's permutation processes give the system robustness and security. Figure (5) shows these processes within the algorithm.

A critical feature of the proposed method is using a concept like the Feistel structure. The same structure is used for encryption and decryption as long as a key timeline for decryption is inversed; this is incredibly helpful for ciphers' hardware implementation because the entire encryption logic must not be retrofitted for decryption. As it is clear, one of the drawbacks of Feistel ciphers is their ability to be paralleled compared to other ciphers. In other ciphers, every round changes the cipher's internal condition, while Feistel ciphers modify part of the internal state per round.[14]

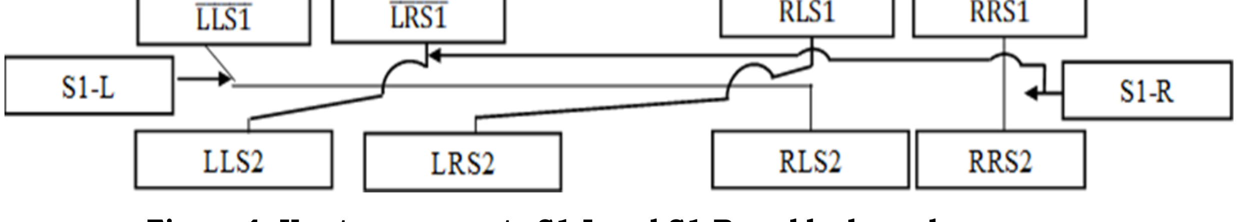

Figure 4: Use two segments S1-L and S1-R wo blocks and transform it into the proposed format with different foundation coding for the proposed algorithm.

The proposed method overcomes this disadvantage of the Feistel structure, where the aggregate state data changes in one step.

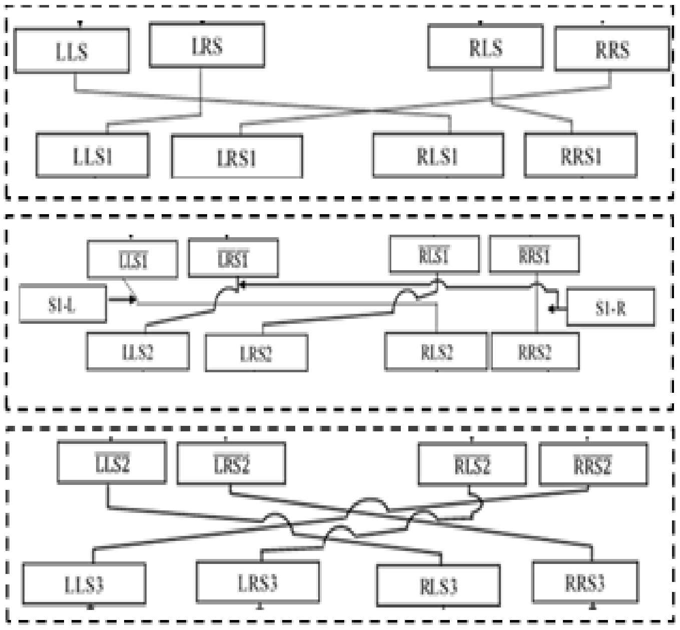

Figure 5: Permutation processes of the proposed algorithm.

The proposed method is implemented using C# code. The block size is chosen with 32, 64, 128, and 256 bits. The size of the foundation segment is also selected according to the size of the plaintext block; all experiments show a good level of security regarding the simple structure of the algorithm. Several tests were done to prove the security of the proposed algorithm. The first test is the randomness test, employed for the modern symmetric cipher blocks, carried out in [15] and [16], which is one of the security tests to determine the Shannon principles of confusion and diffusion. The randomness test is performed to verify the block cipher's security or its fundamental cryptography.

The experiments were done following the statistical test guidelines using the NIST Test Suite strategy; 128 binary sequences of 128-bit data were produced and analyzed with a significant level of 0.01 to test the sensitivity of the proposed lightweight algorithm on changes in plain text. For this experiment, the proportion of sequences in which a specific statistical test must be higher than the proportion value p is computed following the equation.

$$p_{\propto} = (1 - 0.01) - 3\sqrt{\frac{0.01(1-0.01)}{128}} = 0.963616$$

where $\propto = 0.01$, and m = 128.

Frequency Test, Frequency Test Within Block, and Run Test are correlated with SAC to investigate ciphertext randomness and its avalanche effect.

The frequency test is one of the most important tests used to show the security of the encryption algorithms; depending on the frequency, the encrypted text is assumed random,

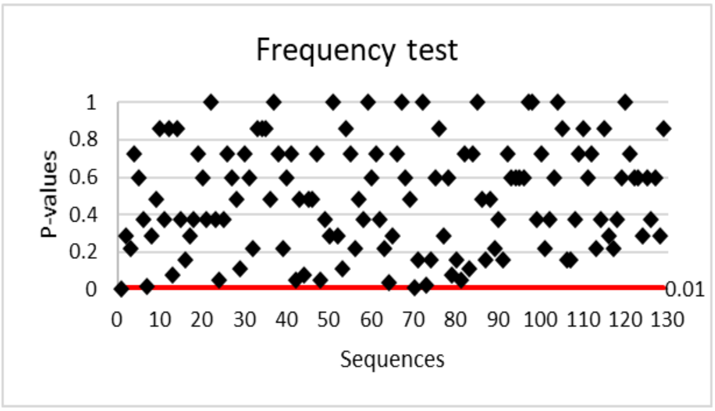

Figure 6: The frequency test of the proposed algorithm.

one of the most significant security measures of the encryption process. This test determines whether the frequency of ones and zeroes in a sequence is about the same as expected for a completely random series. The result t of the frequency test of the proposed lightweight algorithm is shown in Figure (6). It was stated that 127 of the 128 sequences were given a p-value above 0.01, which implied that the lightweight algorithm passed the Frequency test with a 0.9922 proportion

Frequency within the block: this test aims to decide if the frequency in a fixed block is (block size)/2, as would be expected in the case of randomness. It was stated that 127 of the 128 sequences were given a p-value above 0.01, which implied that the lightweight algorithm passed the Frequency test with a 0.9922 proportion. The result t of the frequency within the block test of the proposed lightweight algorithm is shown in Figure (7)

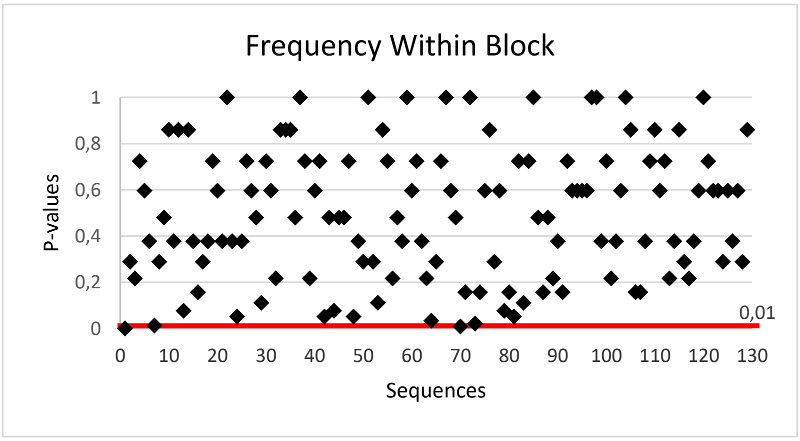

Figure 7: The Frequency within Block test of the proposed algorithm.

The other NIST test is the Run test. This test aims to determine if, for a random sequence, the number of runs and zeroes of different lengths is as predicted. It was stated that 127 of the 128 sequences were given a p-value above 0.01, which implied that the lightweight algorithm passed the Frequency test with a 0.9922 proportion. The result of the frequency within the block test of the proposed lightweight algorithm is shown in Figure (8)

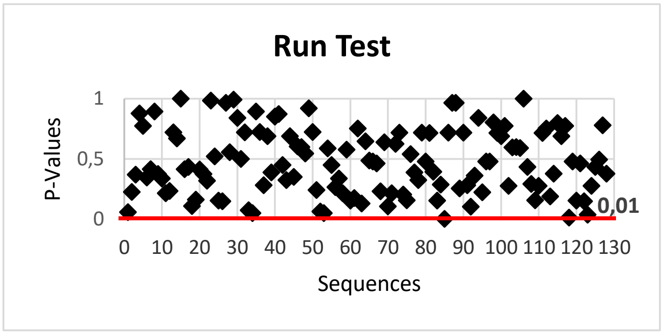

Figure8: The Run test of the proposed algorithm.

It is apparent from the structure allocated to the algorithm, which did not include complex mathematical operations and calculations based on complex derivations. However, it was based only on a set of operations that can be classified as simple operations. Depending on these data, the algorithm can be implemented with the least possible resources, meaning it is Computational Efficiency. In addition, the other factor, low power consumption, has been achieved because the operations conducted within this algorithm require nothing but low energy consumption.

3.1 Algorithm resistance to attacks

Depending on the steps mentioned in the paragraph "The proposed method" within this research paper, the resistance of the proposed algorithm to attacks can be clarified as follows: The encryption approach disclosed utilizes a Feistel network topology and combines many phases to bolster its resistance against prevalent cryptographic assaults. The complexities of the procedure are intended to create a solid and effective defensive system. Here, we explain how certain phases in the encryption approach presented help prevent such attacks:

1. Substitution-Permutation Network: Steps 4 and 9 involve performing permutation transpositions. These phases involve the use of permutation transpositions, which essentially rearrange the placements of blocks throughout the encryption process. The permutation process dramatically enhances the dissemination of information across the ciphertext, hence increasing its resistance against attacks that use patterns or regularities.

2. Key Incorporation and Pseudo-Random Elements: Utilizing Pseudo-Character Slices as Encryption Keys.The utilization of artificial character pieces (S1-L and S1-R) as encryption keys adds another layer of complexity. The accidental parts present in the key make it very big and difficult. This increase-s the number of possibilities one- needs to try for a direct force-attack. With today's computers, going through that many possibilities one by one would take an extremely long time. The biggest computers available now would still be far too slow to complete such a huge task in.

3. Steps 7 to 10 refine the key part. The main variable, S1-L and S1-R, evolves to boost the code's decryption resistance. Each cycle tweaks the key part, making it tough for foes to exploit links between it and other bits over time multiple periods.

4. Confusion and Diffusion The fourth crucial aspect, encompassing steps 2 and 6, involves e-mploying distinct codes during the coding process. This coding technique creates obscurity by symbolizing various sections with specific signs or symbols. Consequently, this approach introduces an additional layer of complexity to the relationship between the clear, unencrypted information and the secret, encrypted content. It works by obstructing likely attacks centered on simple systems. These attacks rely on easily discernible links between the two data types. With this technique, such assaults encounter significant roadblocks.

5. Invertibility and Reversibility: Decryption is vital to access original data from encrypted form. This reversal process undoes encryption precisely, guaranteeing recovery of plaintext from ciphertext. The method ensures decrypting ciphertext yields original plaintext exactly. This property maintains security, allowing authorized access while protecting data from unauthorized parties. Reversing encryption accurately facilitates secure exchange and storage of sensitive information, a crucial characteristic.

6. Utilizing random character sequences as encryption keys makes it tougher for attackers to crack the code. This is because randomness creates unpredictability, which makes it harder for attackers to exploit patterns. However, you must implement randomness carefully to prevent weak or predictable random values from undermining its benefits. In other words, using pseudo randomness as outlined in Steps 7 and 8 is a smart way to boost resilience against various adve-rsarial techniques and make your syste-m more secure.

Random encryption keys introduce unpredictability, which makes life extreme-ly difficult for attackers trying to break the code. Predictable patterns are much easier to exploit, so adding randomness complicates an attacker's efforts significantly. That said, imple-menting randomness requires caution to ensure the random values used are truly random and not weak or predictable in any way. Otherwise, the intended security benefits.

7. Key management and oversight are crucial for the algorithm's security. The randomly gene-rated parts (S1-L and S1-R) must be exactly the same length as the related data pieces in Ste-ps 7 and 8. Verifying the lengths helps reduce risks linked to key management. The pse-udorandom numbers should be unpredictably created. Their lengths must follow the-technical specifications for secure- operations. Even a tiny mistake in length can compromise security. So, it's essential to double-check the lengths during key generation and management. Robust oversight ensures the random parts align perfectly with the corresponding data pieces. This careful length verification is vital for maintaining the algorithm's integrity and preventing potential vulnerabilities.

The encryption technique combines different strategies to strengthen its security against common attacks. It rearranges data, uses changing keys, obscures information, and spreads out the encrypted content. The design with Feistel networks and pseudo-random elements enhances the algorithm's robustness. It makes it harder for potential weaknesses to be exploited. The rearranging step shuffles the order of the original data bits. This helps prevent patterns from being easily detected. The evolving key advancement generates new encryption keys for each portion of data. So, even if one key is compromised, the entire content remains

4. Conclusion

This paper presents a lightweight cryptographic system with schematic proceedings for securing devices with low resources and capabilities. The proposed lightweight cryptography system presented here can be used to achieve IoT security. C# coding language is used to develop software based on this system. Results show that using this proposed algorithm provides confidentiality of information, which is essential to protecting devices and information communication with fewer calculations and computations.

References

[1] W. Stallings, *Computer security principles and practice*, 2015.

[2] N. Fatmawati Octarina, S. Sudiawati, and M. Mardika, "The Application of the Conditio Sine Qua Non Principle on the Crime of Damage through Social Media," *Lambung Mangkurat Law Journal,* vol. 7, pp. 74-92, 2022.

[3] R. J. Kikani, K. Verma, R. Navalakhe, G. Shrivastava, and V. Shrivastava, "Cryptography: Recent research trends of encrypting mathematics," *Materials Today: Proceedings,* vol. 56, pp. 3247-3253, 2022.

[4] H. Landaluce, L. Arjona, A. Perallos, F. Falcone, I. Angulo, and F. Muralter, "A review of IoT sensing applications and challenges using RFID and wireless sensor networks," *Sensors,* vol. 20, p. 2495, 2020.

[5] P. Panagiotou, N. Sklavos, E. Darra, and I. D. Zaharakis, "Cryptographic system for data applications, in the context of internet of things," *Microprocessors and Microsystems,* vol. 72, p. 102921, 2020.

[6] B. A. Forouzan, *Cryptography & network security*: McGraw-Hill, Inc., 2007.

[7] D. Wong, *Real-world cryptography*: Simon and Schuster, 2021.

[8] H. Knospe, *A course in cryptography* vol. 40: American Mathematical Soc., 2019.

[9] J. Katz and Y. Lindell, *Introduction to modern cryptography*: CRC press, 2014.

[10] S. B. Sadkhan and A. O. Salman, "A survey on lightweight-cryptography status and future challenges," in *2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA)*, 2018, pp. 105-108.

[11M. Agrawal, J. Zhou, and D. Chang, "A Survey on Lightweight Authenticated Encryption and Challenges for Securing Industrial IoT," in *Security and Privacy Trends in the Industrial Internet of Things*, ed: Springer, 2019, pp. 71-94.

[12] R. M. "November 2019 | TOP500 Supercomputer Sites". www.top500.org. Archived from the original on November 19, 2020. (2019). *"November 2019 | TOP500 Supercomputer Sites".* Available: www.top500.org

[13] C. M. Gearheart, B. Arazi, and E. C. Rouchka, "DNA-based random number generation in security circuitry," *Biosystems,* vol. 100, pp. 208-214, 2010.

[14] R. Lampe and Y. Seurin, "Security analysis of key-alternating Feistel ciphers," in *International Workshop on Fast Software Encryption*, 2014, pp. 243-264.

[15] J. Soto and L. Bassham, "Randomness testing of the advanced encryption standard finalist candidates," BOOZ-ALLEN AND HAMILTON INC MCLEAN VA2000.

[16] V. Katos, "A randomness test for block ciphers," *Applied mathematics and computation,* vol. 162, pp. 29-35, 2005.