

# Fast and Secure Image Encryption System Using New Lightweight Encryption Algorithm

Rasha S. Ali <sup>1</sup>, Maha Khalil Ibrahim <sup>2</sup>, Saad N. Alsaad <sup>3</sup>

<sup>1</sup> *Electrical Engineering Department, College of Engineering, Al Iraqia University, Baghdad, Iraq*

<sup>2</sup> *Mobile Communications and Computing Engineering Department, College of Engineering, University of Information Technology and Communications, Baghdad, Iraq*

<sup>3</sup> *Computer Science Department, College of Science, Mustansiriyah University, Baghdad, Iraq*

**Abstract** – The rapid evolution of network technologies, where computer vision and the Internet seamlessly merge, poses significant challenges in safeguarding end-user data. To address the growing need for secure data in smart environments, lightweight and efficient security solutions are in high demand. Notably, the Trivium, a well-known encrypt stream cipher (eSTREAM) project, has been the target of several attacks. This paper proposes an innovative approach for fast and highly secured image encryption using byte scrambling and a modified version of the Trivium algorithm. The Henon map is utilized to generate random numbers for permutation, reducing time and increasing security. Performance is measured on 100 colour images with different several tests. The results present a fast and robust security system for the transmission process. It is more efficient than the traditional Trivium method. The encrypted-decryption time is reduced by 1:24 times, making it a quick security system for surgical telepresence and multimedia visuals.

**Keywords** – Fast lightweight encryption algorithm, chaos theory, Henon map, Trivium, secure image, scrambling, block-by-block.

---

DOI: 10.18421/TEM131-20

<https://doi.org/10.18421/TEM131-20>

**Corresponding author:** Saad N. Alsaad,  
*Electrical Engineering Department, College of Engineering,  
Al Iraqia University, Baghdad, Iraq*


**Email:** [dr.alsaadcs@uomustansiriyah.edu.iq](mailto:dr.alsaadcs@uomustansiriyah.edu.iq)

*Received: 05 September 2023.*

*Revised: 01 November 2023.*

*Accepted: 11 December 2023.*

*Published: 27 February 2024.*

 © 2024 Rasha S. Ali, Maha Khalil Ibrahim & Saad N. Alsaad; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at <https://www.temjournal.com/>

## 1. Introduction

Stream ciphers are well renowned for their effectiveness and suitability in situations with limited resources, such as Radio Frequency Identification (RFID) and sensor networks. Over the years, a lot of stream ciphers have been proposed, and their cryptanalysis has received a lot of attention. The “New European Schemes for Signatures, Integrity, and Encryption” (NESSIE) project, which ran from 2000 to 2003, received six stream cipher submissions. Surprisingly, none of the six stream cipher has been selected because they were all broken by cryptanalysis. This led to the 2004–2008 eSTREAM, the ECRYPT Stream Cipher Project [1]. In 2008, a collection of seven stream ciphers, with four in the software profile and the remaining three in the hardware profile, was introduced. The three final hardware profiles were Grain v1 [2], MICKEY 2.0 [3], and Trivium [4].

Trivium is a hardware efficient synchronous stream cipher [4], [5], [6]. It has been approved as an ISO standard and is one of the finalists in the eSTREAM project. Despite the straightforward structure of the design, no method to break the entire cipher has yet been discovered [7]. Trivium's sophisticated and straightforward structure, which uses only bit operations, makes it suitable for source-restricted applications.

Since its submission, Trivium has undergone extensive cryptanalysis, but no attack has been found on its full version [8]. Several simplified versions of Trivium such as Bivium-Toy consists of two NLFSSRs of lengths 31 and 28 respectively, and Trivium-Toy model consists of three NLFSSRs of lengths 31, 28, and 37 [9]. Some attacks are on the reduced round Trivium and the other on its reduced structures but none of the attacks are published on the full version of Trivium [10].

Trivium is a cipher designed for applications with low resource consumption, so developing countermeasures is difficult because it is necessary to increase the cipher's security while also making sure that these countermeasures use the fewest resources possible to keep the cipher lightweight [11]. Due to the exceptional randomization qualities of chaotic maps, there has been a lot of interest in using them in picture encryption applications recently [12]. Chaos signals are regarded as suitable for encryption and improve the robustness of cryptosystems against statistical attacks because of their significant features. Pseudo randomness, non-periodicity, and high sensitivity to system characteristics and beginning conditions are some of these features. As a result, one of the key areas of information security is the integration of chaos theory with cryptography. The necessity for safe image transmission over the communication channel was met by the development of new, effective image security techniques thanks to the chaos-based encryption algorithm [13].

In this paper, we looked for a lightweight synchronous stream cipher system using the properties of chaotic functions to find a simple, quick, and secure solution for image encryption. This method is particularly reliable because of key's wide space in the chaotic functions.

The remainder of the research is structured as follows: We describe the Trivium algorithm in Section 2. Section 3 presents the connection between chaos and cryptography. We provide a brief overview of the methods used in this paper in Section 4. In section 5, a new algorithm is suggested. In section 6, we examine the proposed image cipher's security and assess its effectiveness using a variety of tests, including statistical, histogram, similarity, and other tests. Section 7 provides conclusions to round out the research.

## 2. Related Works

Considering current research in many stages of the IoT security solution, the authors of [1] discussed the significance of studying security in the IoT. The papers [2] and [3] highlighted the usefulness of lightweight cryptography as a means to protect the resource-limited devices of the IoT [4]. This work suggests a new stream cipher construction that is flexible for hardware implementations and has desirable cryptographic features. It is based on block cipher design concepts. The fundamental concept is to employ similar stream cipher components in place of the block cipher's building pieces. The authors of study [14] introduced a technique for attacking Trivium ASIC implementations. It involves active and non-invasive approach based on clock manipulation mixed with Differential Fault Analysis (DFA) cryptanalysis.

Using a collection of chaotic maps, [15] developed an effective picture encryption technique. The suggested algorithm is divided into three phases: confusion, shuffle, and diffusion. A brand-new technique for safe image encryption based on chaos is provided in [16]. The three stages of the encryption algorithm are the production of random numbers, image permutation, and substitution. On the 2D chaotic map known as the standard map, the random number generator is based. An image undergoes two permutations in the second stage. In order to increase the security of converting a plain image into an encrypted one, the major goal of [17] is to design a lightweight nonlinear mechanism for digital image security combining block cipher and chaos theory. Secure IoT (SIT), a simple encryption technique, was suggested in [18]. It uses a 64-bit key and the algorithm combines a Feistel architecture with a permutation-substitution network. Simulation results show that the method provides notable security in just five encryption rounds. A Skew Tent Map-based one round encryption algorithm in order to reduce the time needed for encryption was proposed in [19]. This map produces both the confusion and diffusion necessary for encryption. The proposed algorithm confirms high security level through security analysis. The authors of [20] suggested a secure method combining a chaotic system with a tweaked form of lightweight AES. The 5-D chaos system, a hybrid of the logistic and Lorenz chaotic systems produced the chaos key sequences used in the lightweight AES and SHAKE128. With a speed increase of 145%, the processing complexity of the Lightweight AES was altered while processing time was reduced. The processing complexity of lightweight AES has been adjusted to reduce processing time. The results produced statistical tests that closely resembled the original AES tests. Finally, the paper [21] offered a plan to boost the encryption's security by multiplying the encryption and permutation rounds. For color images, the design has been expanded. Simulation findings show that the suggested method is capable of rejecting any form of attack.

## 3. Trivium

Trivium is a hardware oriented synchronous stream cipher. The objective of designing Trivium is to have flexible scheme which perform reasonably in systems which require fast encryption at low frequencies [10]. It was developed as a test to see how much stream ciphers can be simplified without losing their security, speed, or flexibility. Trivium key generation consists of two phases: first is the internal state initialization using the key and the IV, second is the key stream generation:

**3.1. The Internal State Initialization**

The Trivium design uses 80-bit secret key and 80-bit IV as input. The output is key stream of 288 bits. This is accomplished using three shift registries of length 93, 84, and 111 bits respectively and several operations, including AND, XOR gates. These operational stages are repeated 1152 times to create the internal state. Figure 1 shows the repetitive steps to generate 288 bits [22].

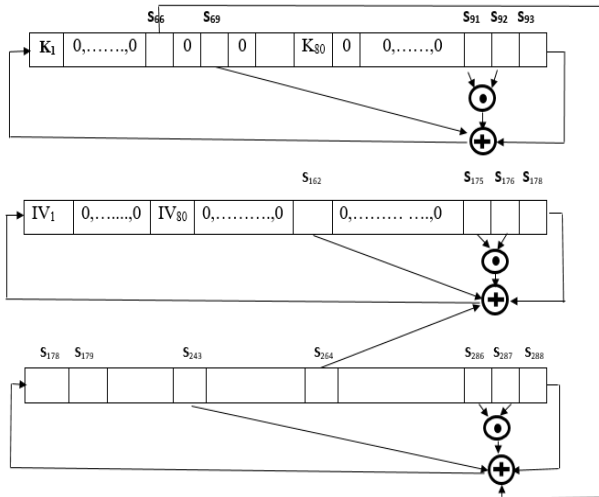


Figure 1. Internal State Initialization Phase

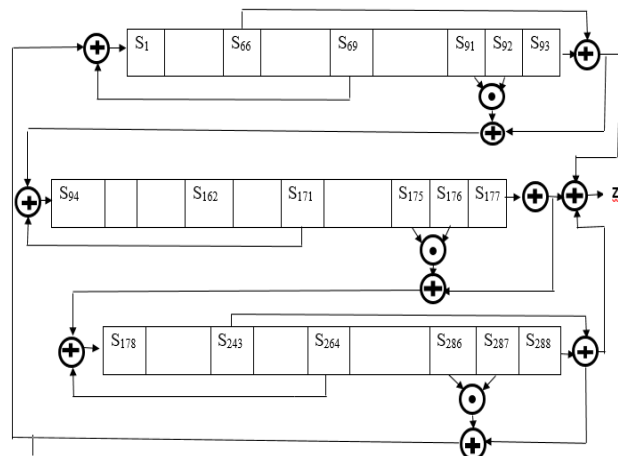


Figure 2. Key stream generation phase

**3.2. Keystream Generation Phase**

This process is used to create the key stream bits that are used in encryption and authentication procedures as the secret key. With each iteration 1 bit of the keystream is extracted from the input, which is a 288-bit initial state. Until the desired key length is created, the procedure occurs (Figure 2).

**4. Methodology**

In this article, we aim to utilize the lightweight Trivium algorithm to be suitable for securing transmitted data on the networks. Our main contributions are as follows:

1. Proposed a compact synchronous stream cipher system with improved parameters that can offer higher security and better chip size and power consumption performance.
2. Outline the guidelines for selecting effective parameters when using chaotic functions.

The sequence of operations that describe the proposed image encryption is illustrated in Figure 3. The input to the Trivium is the output of random bits generator. The proposed system is based on two ideas: first is trying to exploit Trivium design to be implemented as block by block rather than bit by bit. Second is to exploit the random chaotic generated by Henon Map into two sides, the first side as random inputs to the internal state initialization and the second as an active permutation key to the plain image. The size of the suggested permutation table is determined by the amount of image pixels, making it dynamic rather than static. In this work, the user is free to use a permutation procedure whenever he/she likes. The main stages shown in the Figure 4 will be covered in more details:

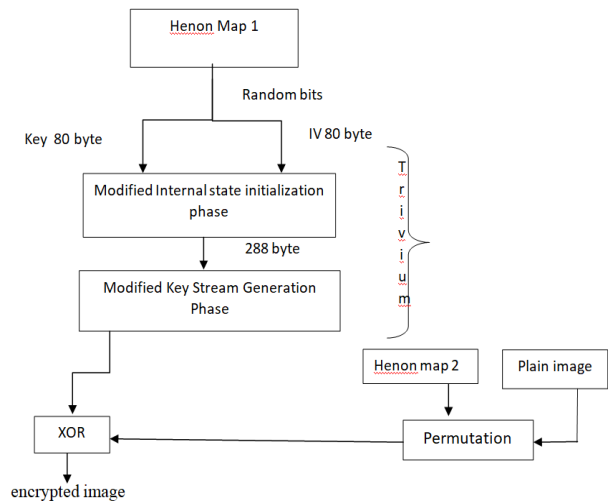


Figure 3. The schematic representation of the proposed method

**4.1. Henon Map1:**

The aim of using Henon map in this step is to generate 320 bits divided into two parts as input to the next stage (modified key stream generators, Algorithm 1)

**Algorithm 1** // Sequence of bits generation

```

Input: a,b , x1 ,y1
Output= SB // array of 1280 bits (160 Bytes)
Begin
Step 1: For i= 1To 1280
    Begin
         $x_{i+1} = 1 - ax_i + y_i$ 
         $y_{i+1} = b * x_i$ 
Step 2: if  $x_i \leq 1$  then  $x_i = 0$ 
        Else  $x_i = 1$ 
        if  $y_i \leq 1$  then  $y_i = 0$ 
        Else  $y_i = 1$ 
Step 3: j= i
SB[j]=  $x_i$ 
j=j+1
SB[j] =  $y_i$ 
    End
End
End

```

**4.2. Modified Internal State Initialization**

The stage is dedicated to generate 288 bytes. The Henon map in the previous stage was used as the generation process of Key and IV. The input is 1280 bits generated from Henon map. The stage is similar to original initial state of Trivum but as block by block not bit by bit (Algoithm2).

**Algorithm 2: modified internal state**

```

Input: R[160] // 1280 bits (160 Bytes) generated by
applying algorithm 1.
Output: SB[288] bytes
Begin
    Step1: SB[1..80]= R[1..80]
           S[81..94]= 0
           SB[94..174] = R[81..160]
           SB [175] = SB [176] = SB [177] = 0
           // initialized bits position of internal state from 178 to
           288
    Step2: For a= 178 to 275
           SB [a] = 0
           End for
    Step3: SB [286] = SB [287] = SB [288] = 0
    Step4: For H = 1 to 4 * 288
           t1 =  $SB_{66} \oplus (SB_{91} \& SB_{92}) \oplus SB_{93} \oplus SB_{171}$ 
           t2 =  $SB_{162} \oplus (SB_{175} \& SB_{176}) \oplus SB_{177} \oplus$ 
           SB26.
           t3 =  $SB_{243} \oplus (SB_{286} \& SB_{287}) \oplus SB_{288} \oplus$ 
           SB69.
           (SB1,SB2,...,SB93) ← (t3, SB1,...,SB92) .
           // shifting to left by 1
           (SB94,SB95,...,SB177)←
           (t1,SB94,...,SB176). // shifting to left by 1
           (SB178,SB279,...,SB288)←(t2,SB178,...,SB
           287).
           // shifting to left by 1
    End for
End
End

```

**4.3. Keystream Generation:**

It is used to generate key streams that are utilized in encryption and authentication processes as secret keys. The input is a 288-byte beginning state that is processed iteratively to extract a different keystream each time. Up until the desired key length is generated, the procedure is repeated (Algorithm 3).

**Algorithm 3:** Modified Key Stream Generation

```

Input: S // 288 byte output of algorithm 2
Output: Z// key stream of N bytes (image size)
begin
    Step1: do step2 four times
    Step2:
        For i=0 to N do
            T1 =  $SB_{66} \oplus SB_{93}$ 
            T2 =  $SB_{162} \oplus SB_{177}$ 
            T3 =  $SB_{243} \oplus SB_{288}$ 
            Zi = T1  $\oplus$  T2  $\oplus$  T3
            T1 =  $SB(66) \oplus (SB(91) \& SB(92))$ 
             $\oplus SB(93) \oplus SB(171)$ 
            T2 =  $SB(162) \oplus (SB(175) \& SB$ 
            (176))  $\oplus SB(177) \oplus SB(264)$ 
            T3 =  $SB(93) \oplus (SB(286) \& SB$ 
            (287))  $\oplus SB(288) \oplus SB(69)$ 
            (SB1, SB2 to SB93) = (T1, SB1 to
            SB92) // shifting to left by 1
            (SB94, SB95, ..., SB177) = (T2, SB94,
            ... ,SB176) // shifting to left by 1
            (SB178, SB179 ...,SB 288) = (T3,
            SB178, ...,SB278) // shifting to left by 1
        End For
    Step3: Return Z
End

```

**4.4. Henon map 2:**

In this stage the Henon map is employed in permutation process by sorting the random numbers generated form Henon map in ascending form. The indexing of the sorting is considered as a key permutation. Algorithm 4 with example is illustrated below:

**Algorithm 4 (permutation key)**

```

Input: a,b , x1 ,y1
Output: Key // permutation key
Begin:
    Step 1: Perform step1 of algorithm 1 n times // n=
           the size of image
           Save the output of step1 (xi and yi) in S
    Step2: arrange the S in ascending sequence
    Step 3: Key=the index of the output of step 2;
End

```

Example: suppose n=8

Step1: Consider the random numbers are:

1	2	3	4	5	6	7	8	9

Step3:

Key permutation= 5, 3, 8, 1, 8, 2, 4, 9, 6

Step 2:

5	3	7	1	8	2	4	9	6

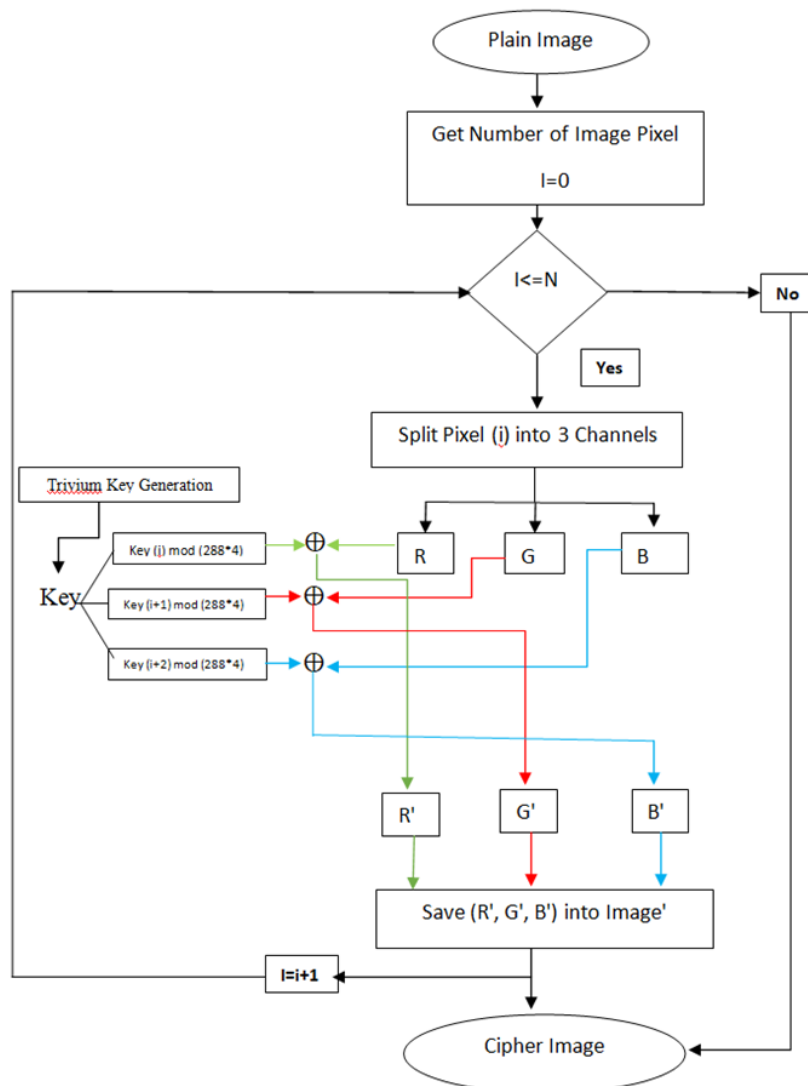


Figure 4. Modified Trivium algorithm

## 5. Experimental Results

In order to demonstrate the effectiveness of the suggested technique, some experimental findings are provided in this section. The suggested encryption technique is applied to various types and sizes of images. Eight selected images are encrypted using the modified version of the Trivium technique. Table 1 illustrated the histogram before and after encryption. The modified Trivium algorithm employs both block of bits and a permutation process.

The computations for the information entropy, correlation coefficients, chi-square, PSNR, similarity, histogram, and NCPR are shown in Tables 2, 3, and respectively. The entropy of the encrypted images is approximately 8. It indicates possess high of randomness.

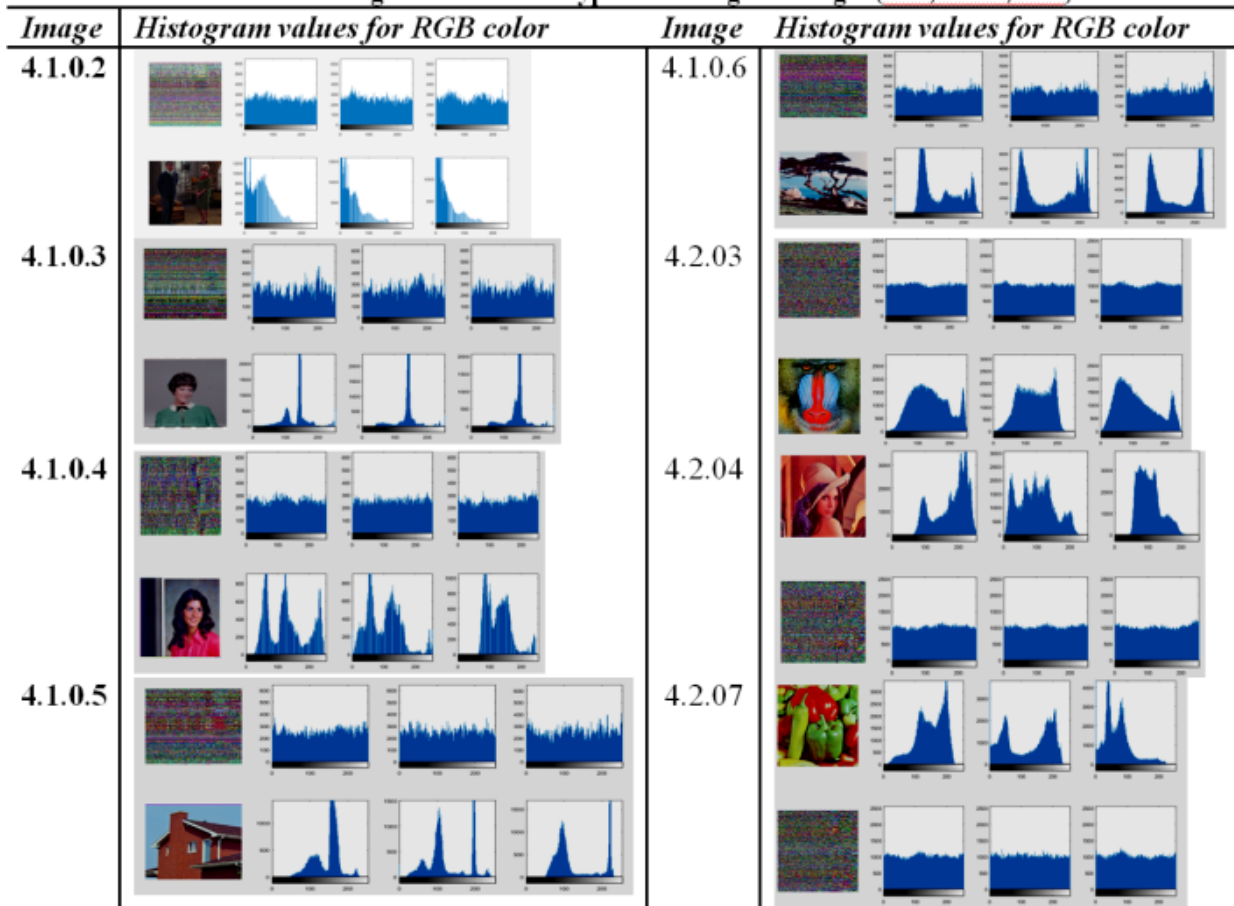
The correlation coefficient of the encrypted images using the proposed method are closed to zero, which means the correlation between any two neighbors pixels of the plain images are disappeared. Finally, Table 5 shows time in millisecond required for the traditional Trivium and the modified one.

**5.1. Histogram Analysis**

In order to protect against targeted attacks, both unencrypted and encrypted images should exhibit no discernible statistical characteristics.

Histograms display the distribution of brightness pixels, demonstrating no statistical similarity. Encrypted images cannot be decrypted, and the distribution of the cipher picture measures the cryptosystem's efficacy.

Table 1. Histograms for the encrypted and original images (Red,Green,Blue)



**5.2. Information Entropy**

The uncertainty of the information source has been measured using information entropy [23]. It is clear that the entropy of eight bits ideally random is eight [24]. Equation 1 represents Entropy by H(X):

$$H(x) = -\sum_i P_i \log(P_i) \tag{1}$$

Table 2 displays the findings of this measurement and demonstrates that the suggested system comes very close to achieving the theoretical value of 8.

Table 2. Cipher and plain images entropy values

Image	Cipher			Average	Plain			Average
	Red	Green	Blue		Red	Green	Blue	
4.1.02.tiff	7.9978	7.9974	7.9969	7.9974	6.2499	5.9642	5.9309	6.0483
4.1.03e.tiff	7.9976	7.9975	7.9978	7.9976	5.715	5.3738	5.7117	5.6002
4.1.04e.tiff	7.9977	7.9974	7.9977	7.9976	7.2549	7.2704	6.7825	7.1026
4.1.05e.tiff	7.9974	7.9970	7.9974	7.9973	6.4311	6.5389	6.232	6.4007
4.1.06e.tiff	7.9972	7.9976	7.9976	7.9975	7.2104	7.4136	6.9207	7.1816
4.2.03e.tiff	7.9980	7.9986	7.9982	7.9983	7.7067	7.4744	7.7522	7.6444
4.2.04e.tiff	7.9971	7.9983	7.9970	7.9977	7.2531	7.594	6.9684	7.2719
4.2.07e.tiff	7.9976	7.9978	7.9971	7.9975	7.3388	7.4963	7.0583	7.2978

**5.3. Correlation Analysis**

It was examined whether the encrypted images and the original images correlated. Low correlation value indicates the relation between the two images is weak. Correlation is being computed to equation E (2). The results are displayed in Table 3. Because values are less than zero, it implies that the original and encrypted images do not correlate with one another.

$$r_{x,y} = \frac{n(\sum x, y) - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2] [n \sum y^2 - (\sum y)^2]}} \quad (2)$$

The correlation coefficient is the particular metric used in correlation analysis to represent the linear relationship between two variables.

Table 3. Correlation for the encrypted images

Image	Average of Red (Vertical + Horizontal + Diagonal)	Average of Green (Vertical + Horizontal + Diagonal)	Average of Blue (Vertical + Horizontal + Diagonal)	Average Value
4.1.02e.tiff	0.0803694	0.13101918	0.12010264	0.1104971
4.1.03e.tiff	0.09702097	0.09702097	0.08832522	0.0941224
4.1.04e.tiff	0.04439144	0.06591747	0.05910805	0.0564723
4.1.05e.tiff	0.07992723	0.07999483	0.0655151	0.0751457
4.1.06e.tiff	0.06901073	0.06143142	0.05148047	0.0606409
4.2.03e.tiff	0.0367389	0.02629188	0.02814084	0.0303905
4.2.04e.tiff	0.06217844	0.04623093	0.06162978	0.0566797
4.2.07e.tiff	0.04985185	0.0399578	0.05974601	0.0498519

**5.4. Number of Pixels Change Rate (NPCR)**

NPCR serves as measure of key sensitivity. It involves comparing two cipher text files, the first one is resulting from encryption of text file and the other results of the same text file with slight change. The two cipher text files should be completely different. NPCR is determined by the following formula:

$$NPCR = \sum_{i=0}^H \sum_{j=0}^W \frac{D(i,j)}{T} * 100\% \quad (3)$$

$$D(I,J) = \begin{cases} 0 & c_1(i,j) \neq c_2(i,j) \\ 1 & c_1(i,j) = c_2(i,j) \end{cases}$$

Where,

H stands for the image's weight, c1 and c2 for its encrypted versions, and T for the entire number of pixels in the cipher image.

NPCR is frequently applied to image encryption and utilized in security studies of differential attacks. It concentrates on the precise amount of pixels that underwent differential attacks to change their values [25]. Table 4 provides proof that the proposed strategy was determined to be effective when NPCR values for each color component image were >99% and these values were gathered from different separate images.

**5.5. Chi-Square Analysis**

The chi-square test supports the uniformity brought on by the encryption technique. The method's effectiveness is demonstrated by the cipher-image histogram's rather uniform distribution. It is essential to guarantee that the statistical characteristics of the encrypted genuine real images are dissimilar in order to stop information from being revealed to attackers. The histogram analysis reveals the distribution of the image's pixel values. The results of chi-square analysis are shown in Table 4. The chi-square analysis was performed to further assess the histogram's uniformity as indicated in Equation (4):

$$X^2 = \sum_{L=0}^{255} \frac{(o_L - e_L)^2}{e_L} \quad (4)$$

**5.6. Peak Signal-to-Noise Ratio (PSNR)**

The objective is to minimize the PSNR to the cipher picture since there are noticeable positive consequences when the PSNR values are low. But steganography's PSNR has to be increased.

$$PSNR = 10 \log_{10} (255^2 / MSE(f, g)) \quad (5)$$

**5.7. Similarity Measurement (SIM)**

The similarity between the original and ciphered images is measured using this statistic. Table 4 displays the results of this measurement. This measurement should be lowered for the encryption procedure. That is, good results have been produced in terms of lowering the values of SIM when Equation (6) has been employed to determine the similarity between the original and encrypted picture. Table 5 displays the length of time needed for encryption and decryption for various picture sizes. By employing the advised technique, this time may be cut down.

$$SIM = \frac{\sum \sum F_{i,j} * F' (i,j')}{\sum \sum F_{i,j} * F' (i,j)} \quad (6)$$



Table 4. PSNR, NPCR, chi-square, similarity evaluation measurements

Similarity	NPCR	Chi-square	PSNR	Image	Similarity	NPCR	Chi-square	PSNR	Image
0.002	R= 0.997 G= 0.996 B= 0.996	0.009	0.001	4.1.06e.tiff	0.001	R=0.996 G=0.995 B= 0.996	0.0069	0.001	4.1.02e.tiff
0.0005	R= 0.996 G= 0.996 B= 0.996	0.0014	0.002	4.2.03e.tiff	0.002	R= 0.996 G= 0.996 B= 0.997	0.0245	0.002	4.1.03e.tiff
0.0005	R= 0.996 G= 0.996 B= 0.996	0.0017	0.001	4.2.04e.tiff	0.002	R=0.996 G=0.996 B= 0.997	0.006	0.001	4.1.04e.tiff
0.0005	R= 0.996 G= 0.996 B= 0.996	0.0023	0.001	4.2.07e.tiff	0.002	R=0.996 G=0.995 B= 0.996	0.0127	0.001	4.1.05e.tiff

Table 5. Comparison between modified and the original Trivium for the consuming time

Image	Original Trivium in Ms	Proposed Trivium in Ms
4.1.02.tiff	6	2
4.1.03e.tiff	7	3
4.1.04e.tiff	7	2
4.1.05e.tiff	6	3
4.1.06e.tiff	4	1.5
4.2.03e.tiff	13	7
4.2.04e.tiff	12	5
4.2.07e.tiff	11	5

## 6. Conclusion

In this research, a robust multimedia lightweight encryption method is used. This method includes of key generation, encryption, and permutation processes. A modified Trivium method is proposed, the modification can be seen in the permutation stage (diffusion). A modified Trivium method can sufficiently increase the security and efficiency of the encryption process, since it increased the randomness and number of permutation table cells (number of cells = number of image pixels). The proposed method is presented and tested for different types of images (color and gray scale) of different sizes and it can be easily extended for video frames. Finally, when comparing the proposed system strength with traditional one, there are several strength points including: the proposed system works on block by block not bit by bit which reflect positively on the time, and dynamic permutation process was added to increase the security.

## Acknowledgments

The authors are immensely grateful to the "Iraqi Ministry of Higher Education and Scientific Research", "Al Iraqia University", "University of Information Technology and Communications", and "Mustansiriyah University" for supporting in doing this research.

## References:

- [1]. Robshaw, M., & Billet, O. (Eds.). (2008). *New stream cipher designs: the eSTREAM finalists*, 4986. Springer.
- [2]. Hell, M., Johansson, T., & Meier, W. (2007). Grain: a stream cipher for constrained environments. *International journal of wireless and mobile computing*, 2(1), 86-93.
- [3]. Babbage, S., & Dodd, M. (2006). The stream cipher MICKEY 2.0. *ECRYPT Stream Cipher*, 191-209.
- [4]. De Canniere, C. (2006). Trivium: A stream cipher construction inspired by block cipher design principles. In *International Conference on Information Security*, 171-186. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [5]. Potestad-Ordóñez, F. E., Tena-Sánchez, E., Mora-Gutierrez, J. M., Valencia-Barrero, M., & Jiménez-Fernández, C. J. (2021). Trivium Stream Cipher Countermeasures Against Fault Injection Attacks and DFA. *IEEE Access*, 9, 168444-168454.
- [6]. Jiao, L., Hao, Y., & Li, Y. (2019). Improved guess-and-determine attack on TRIVIUM. *IET Information Security*, 13(5), 411-419.
- [7]. De Canniere, C., & Preneel, B. (2008). Trivium. In *New Stream Cipher Designs: The eSTREAM Finalists*, 244-266. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [8]. De Canniere, C., & Preneel, B. (2005). TRIVIUM specifications. eSTREAM, ECRYPT stream cipher project. *Report 2005/030*.
- [9]. Aumasson, J. P., Dinur, I., Meier, W., & Shamir, A. (2009). Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In *International Workshop on Fast Software Encryption*, 1-22. Berlin, Heidelberg: Springer Berlin Heidelberg.



- [10]. Lechtaler, A. C., Cipriano, M., García, E., Liporace, J., Maiorano, A., & Malvacio, E. (2014). Model design for a reduced variant of a Trivium Type Stream Cipher. *Journal of Computer Science and Technology*, 14(1), 55-58.
- [11]. Dinur, I., & Shamir, A. (2009). Cube attacks on tweakable black box polynomials. In *Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28*, 278-299. Springer Berlin Heidelberg.
- [12]. Fouque, P. A., & Vannet, T. (2013). Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In *International Workshop on Fast Software Encryption*, 502-517. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [13]. Islam, S., Afzal, M., & Rashdi, A. (2013). On the security of LBlock against the cube attack and side channel cube attack. In *Security Engineering and Intelligence Informatics: CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings 8*, 105-121. Springer Berlin Heidelberg.
- [14]. McDonald, C., Charnes, C., & Pieprzyk, J. (2008). An algebraic analysis of trivium ciphers based on the boolean satisfiability problem. In *Proceedings of the 4th International Workshop on Boolean Functions: Cryptography and Applications*, 173-184. Laboratoire d'Informatique Algorithmique: Fondements et Applications.
- [15]. Potestad-Ordóñez, F. E., Valencia-Barrero, M., Baena-Oliva, C., Parra-Fernández, P., & Jiménez-Fernández, C. J. (2020). Breaking Trivium stream cipher implemented in ASIC using experimental attacks and DFA. *Sensors*, 20(23), 6909.
- [16]. Abdullah, H. N., & Abdullah, H. A. (2017). Image encryption using hybrid chaotic map. In *2017 International conference on current research in computer science and information technology (ICCRIT)*, 121-125. IEEE.
- [17]. Mondal, B., & Mandal, T. (2016). A novel chaos based secure image encryption algorithm. *International Journal of Applied Engineering Research*, 11(5), 3120-3127.
- [18]. Ullah, A., Shah, A. A., Khan, J. S., Sajjad, M., Boulila, W., Akgul, A., ... & Ahmad, J. (2022). An efficient lightweight image encryption scheme using multichaos. *Security and Communication Networks*, 2022.
- [19]. Usman, M., Ahmed, I., Aslam, M. I., Khan, S., & Shah, U. A. (2017). SIT: a lightweight encryption algorithm for secure internet of things. *arXiv preprint arXiv:1704.08688*.
- [20]. FARAJALLAH, M. (2022). LIGHTWEIGHT CHAOTIC BLOCK CIPHER FOR IOT APPLICATIONS. *Journal of Theoretical and Applied Information Technology*, 100(15).
- [21]. Mondal, B., & Mandal, T. (2017). A light weight secure image encryption scheme based on chaos & DNA computing. *Journal of King Saud University-Computer and Information Sciences*, 29(4), 499-504.
- [22]. Naif, J. R., Abdul-Majeed, G. H., & Farhan, A. K. (2019). Secure IOT system based on chaos-modified lightweight AES. In *2019 International Conference on Advanced Science and Engineering (ICOASE)*, 1-6. IEEE.
- [23]. Azami, H., da Silva, L. E. V., Omoto, A. C. M., & Humeau-Heurtier, A. (2019). Two-dimensional dispersion entropy: An information-theoretic method for irregularity analysis of images. *Signal Processing: Image Communication*, 75, 178-187.
- [24]. Zhu, H., Zhang, X., Yu, H., Zhao, C., & Zhu, Z. (2016). A novel image encryption scheme using the composite discrete chaotic system. *Entropy*, 18(8), 276.
- [25]. Pareek, N. K. (2012). Design and analysis of a novel digital image encryption scheme. *arXiv preprint arXiv:1204.1603*.