[1]Ayman Mohammed Enad Alkaraawi,

[2]Nermeen Mahmoud Kashief,

[3]Wagdy Gomaa El-Sayed

An Image Encryption Technique Utilizing Lightweight PRESENT and AES Algorithms in Conjunction with Chaotic Systems

Abstract: - Chaotic map-based techniques for image encryption have been widely reported in recent years as a very active area of research. Most of these methods suffer either from slow encryption speeds, efficiency at the cost of security, or violate any one of these two criteria. This work introduces a new algorithm designed with a logistic map, the lightweight PRESENT algorithm, and the AES S-box that can reconcile both efficiency and security requirements. It starts generating some initial parameters for the logistic map using BLAKE2b, which is calculated based on an input image, a shared key and an initialization vector IV. We take the initially computed parameters from the first BLAKE2b hash, set up the logistic map, and use it to generate a sequence of pseudorandom numbers. These random numbers are critically important because they determine how we should rearrange and mix the blocks during encryption. After PRESENT encryption is applied, we apply the AES S-box to replace values in the block. The introduced substitutions are biased by the chaotic values generated by the logistic map to introduce further confusion. In order to evaluate the performance of the algorithm, a set of metrics such as correlation coefficient, entropy, "mean square error (MSE)", "peak signal-to-noise ratio (PSNR)", histogram, "number of pixel change rate (NPCR)", "unified average changing intensity (UACI)", and key space will be utilized. Experimental evaluations reveal that the algorithm holds great promise, with security features at a higher level and with excellent resistance to differential attacks.

Keywords: image encryption, lightweight, logistic map, chaotic system, AES S-box, BLAKE2, PRESENT algorithm

1. INTRODUCTION

In today's digital landscape, safeguarding information is crucial, particularly when it comes to sharing sensitive data. To address this need, various cryptographic algorithms have been designed to ensure the "secure storage of data on computer" systems and its safe transmission over networks. Digital images, for instance, are handled with much care as they constitute some kind of sensitive data that needs to be both "stored and transmitted safely" [1]. Images are indeed "two-dimensional arrays" of pixel values, and the number of channels differs according to the type: grayscale images possess only one channel, while standard color images have three, and images with transparency offer a fourth channel. By the nature of the relation between successive pixels, digital images usually contain a lot of redundancy.

The use of two main techniques-confusion and diffusion-combats redundancy to protect the data while encrypting images [2]. Confusion usually occurs by substituting a pixel value with another so that the original image cannot be determined. Such a process is controlled through an encryption key. On the other hand, diffusion allows even the slightest modification made to the original image to have significant modifications across the entire encrypted image. In theory, it should modify approximately half of the pixels. It also holds inversely; any modification made in the encrypted image should affect a large number of the pixels in the decrypted original image. Thus, the diffusion of permutation operations in encryption algorithms helps to reduce the correlation between adjacent pixels in the original image [3].

There are many factors which compare well how well and safe the different encryption methods work [4]. Some of the tools used to judge the spread features of an encryption method include UACI, NPCR, and MSE. There are also other ways of determining how well an encryption works, including the graph and peak "signal-to-noise ratio (PSNR)". The information entropy and the association value [5] can be used to figure out how safe an encryption method is.

To ensure proper decryption of the cypher image, encryption algorithms must be capable of reversing the confusion and dispersion processes [6]. Chaotic maps are at the heart of our novel method of lightweight image encryption. Chaotic maps are easy to build, generate pseudorandom sequences quite rapidly, and require minimal

[1]ayman.alkaraawi_PG@alexu.edu.eg
[2]nermeen.kashief@alexu.edu.eg
[3]wagdygoma@alexu.edu.eg
1,2,3 Computer Science Department, Faculty of Science, Alexandria University, Alexandria, Egypt

computational resources. The initial parameters of the logistic map are generated using an IV and a pre-shared symmetric key. This method effectively addresses confusion and diffusion resulting from encryption. Increase security with the use of single-use keying. This method constructs a different key by XORing the pre-shared key with a hash that is created based on the image. Even when encrypting the same image many times, the results will vary because of the randomness in the generation of the IV.

This study makes some important contributions. To begin with, lightweight picture encryption has a good trade-off between speed, quality, and security in its designs. The algorithm is put through tests such as correlation coefficient, histogram analysis, and entropy measures, among others. Other studies have paid special attention to key space, encryption quality, diffusion qualities, and resistance to differential assaults.

The subsequent sections of the paper shall follow the outline below: We shall present the many reported methods of image encryption in Section 2. Section 3 presents the proposed solution for image encryption along with the pseudocode that supports it. In Section 4, we list the parameters used to analyze the performance of the proposed method, and in Section 5, we present a full analysis of what it produces. The conclusion of the article is found in Section 5.

2. RELATED WORK

In the last few years, several ways of encrypting images emerged, each with its own approach and methodology. Lightweight cryptography, for its part, has been largely discussed over the past two decades. Interest in such research has been significantly intense during the last five years, wherein multiple lightweight algorithms were introduced. These present not just high security but also efficiency. These algorithms have been shown to withstand attacks of different varieties [7].

PRESENT, a lightweight block cipher proposed by Bogdanov et al. [8], was designed with a main focus on the security and on hardware efficiency. Hardware implementation of the compact cipher renders it particularly competitive with other algorithms belonging to the same category. Despite multiple attempts for cryptanalytic attacks, the cipher has successfully resisted them and has confirmed its status as a secure and effective entity among lightweight cryptology.

In [9], a new encryption algorithm for images was proposed, aiming to further protect data and the recipient's privacy against potential hacking attacks. The paper discussed the weaknesses in the well-known Data Encryption Standard (DES), which is vulnerable to differential attacks. However, despite the complexity preservation difficulty, the proposed methodology employs PRESENT symmetric encryption scheme. It is effective as well as lightweight regarding the protection of information over networks. The introduced technology achieves fast picture encryption and decryption operations using parallel processing within DES and PRESENT algorithms in an effective way. This study incorporates a 2D chaotic method for key generation to further strengthen the defences against brute-force, statistical and differential attacks. There are four rounds in the DES algorithm, but here there are four rounds running parallel in the PRESENT algorithm. To generate keys we use the same 2D chaotic system. The proposed system spreads keys and blocks over four, five, or six cores in parallel for better performance. Various metrics confirmed that the PSNR values were quite low, indicating high-quality image encryption. This algorithm presented considerably high values against the MSE, indicating a significant difference between the encrypted image and the original. Taken as a whole, a parallel algorithm as described is faster than a corresponding sequential algorithm (DES-PRESENT). It thus fulfills its task of secure data and image protection applicable to color images.

Research in [10] highlighted the essentiality of image encryption in enhancing telecommunications media, mainly regarding the privacy aspect in most aspects of life. The paper paid attention to the fact that encrypting images before forwarding them over the Internet ensures privacy concerns. The proposed technique employs a hyperchaotic map and the ChaCha symmetric stream cypher to encrypt images. The software attained further security by leveraging the sensitive starting conditions and exploiting the pseudo-randomness and control parameters inherent to chaotic and hyperchaotic maps. Using an initial seed number, parameter variation, and unpredictable directions due to chaos enhanced the security. The lightweight image encryption algorithm presented here demonstrated considerable strength against brute-force attacks, with a substantial key space. The scheme also countered statistical attacks by mitigating weaknesses of histogram correlation and entropy in the encrypted images.

The next important contribution was presented in [11]. The authors have designed a lightweight cryptosystem meant to be implemented on the constrained IoT devices to secure the data produced from interconnected devices against cyberattacks. Algorithmically, the scheme relies on the Advanced Encryption Standard (AES) and a new

chaotic S-box, whose properties in the cryptographic context have been confirmed, and randomness through NIST tests. AES has recently gained widespread applicability in embedded platforms, especially after its adoption by the IEEE 802.15.4 protocol. The experiment certifies that the proposed cryptosystem is yielding strong encryption results while adhering to the resource-constrained requirements of the IoT sensors. Attributed with these properties, the algorithm is applicable for image encryption and security in the communication between smart networked devices.

In [12], a pair of chaotic maps were used. Instead, the original picture was permuted using a logistic-sine map, and then the permuted image was substituted by a logistic-Chebyshev map. To the final cypher picture, the XOR operation was employed to combine the outputs of the two chaotic maps with the replaced image. However, logistic maps are the foundation of their approach in [13], and this is a significant contribution. The hashing process starts by hashing the original image's plaintext. This hash value is split into four equal segments, and then those will be utilized for the four different logistic maps to generate four arrays of pseudorandom numbers. To do this series of row and column permutations, this method utilizes the first two keys. The third key is then applied to an XOR operation. Finally, the choice between the AES S-Box and its inverse is determined by the fourth key for purposes of substitution. However, this method is condemned for having a slower encryption speed.

In Lu et al. [14], another work had been presented on a method that makes use of logistic-sine maps based on one S-Box. Their scheme is initiated with generating the S-Box and the chaotic sequence, which are dependent on the pre-shared keys. The chaotic sequence used in the permutation of the plaintext image was then put to substitution twice, whose keys for the S-Box were utilized as the chaotic sequence. Nevertheless, their logistic-sine system skips modular operations, which increases speed slightly, but otherwise, the gain in encryption time is very minimal.

In [15], a new Lightweight Advanced Encryption Standard (LAES) is proposed for IoT security with the aim of encrypting sensing data and therefore reducing encryption and decryption times. The chaotic system applied was through logistic maps in more than one dimension to transform key AES processes, namely S-Box, initial permutation IP, and shifting values. Particularly, the traditional AES MixColumns stage is replaced with dynamic ShiftRows. The generated S-Box is a new one, from a 1D chaotic logistic map. Performance analysis has shown LAES to encrypt files in a fast way, passing several security tests, thus showing it suitable for applications in the area of IoT security.

3. BLAKE2 HASH FUNCTION

BLAKE2 is a new cryptographic hash function developed from the original BLAKE design, which represented a strong contender in the NIST SHA-3 competition. It is created by a group of researchers including Jean-Philippe Aumasson and Samuel Neves. Here, in order to improve upon security with performance over its predecessors, it presents superior speeds as one of its salient features and makes it highly suitable for applications such as file integrity verification or hashing of passwords [16].

Basically, BLAKE2 is a design that is simple in form yet very effective. It uses the Merkle-Damgård construction to handle varying input sizes and yet still produces an output of known length. There are two variants of BLAKE2: BLAKE2b can be used for up to 64-byte (512-bit) hashes and is optimized for 64-bit systems, while BLAKE2s produces a 32-byte (256 bits) hash output that is ideal for smaller, resource-limited environments. Both versions provide excellent performance, surpassing the performance of SHA-2 and SHA-3 in most tests [17].

Another key point of BLAKE2 is the flexibility. In this case, the user can specify output lengths to suit his or her security needs. The keyed-hashing aspect is very secure in this sense. In that regard, it lets a user make use of a secret key and can offer protection against a number of attacks, including length extension attacks. The inner construction of the BLAKE2 has been made to be efficient, so it might take as much as benefits modern processors may offer with still being strong against current cryptographical threats [18].

BLAKE2 has been standardized in RFC 7693 by the IETF, which explains the spec and gives the full endorsement to its usage. Such endorsement in turn has led to rapid growth in both software and hardware implementations. Its speed and flexibility combined with robust security makes it a choice for most developers seeking reliable hashing under the digital landscape of today [16].

4. LOGISTIC MAP IN IMAGE ENCRYPTION

It is a mathematical function aiding in creating pseudorandom sequences which are very important for a confusion and diffusion, that's applied whenever images are being encrypted. The logistic map, an interesting tool that has

been pretty often applied in the field of image encryption due to its simplicity as well as chaotic behavior [19], is therefore defined by the following recursive equation:

$$X_{n+1} = r\, X_n(1 - X_n) \ldots (1)$$

The logistic map needs a starting parameter X0, that can be considered as a starting population. Its output X shall be kept within the range of [0, 1]. The used parameter r, playing the role of growth rate, is between the range [0, 4]. For this map, its behavior is chaotic if the r interval contains the range [3.56995, 4][20]. This transition to chaos enables two matrices to be generated using the logistic map, and the encryption process employs these two matrices to strengthen the security of plaintext images.

In this image encryption technique, the logistical map is of prime importance as it generates a chain of events, which enhances the diffusion and confusion processes. When the relationship between the plaintext and ciphertext gets distorted, it is known as confusion. Consequently, even minor changes in the input have significant outcomes on the output. On the other hand, diffusion significantly reduces correlation and redundancy by making the effect of a single pixel of the plaintext lie scattered over various pixels of the ciphertext [21]. The bifurcation diagram, depicted in Figure 1, of the logistic map is an excellent illustration of its transition from stable behavior to chaos, thus showing the possibility of such a mapping in order to play the role of a random sequence generator [20].

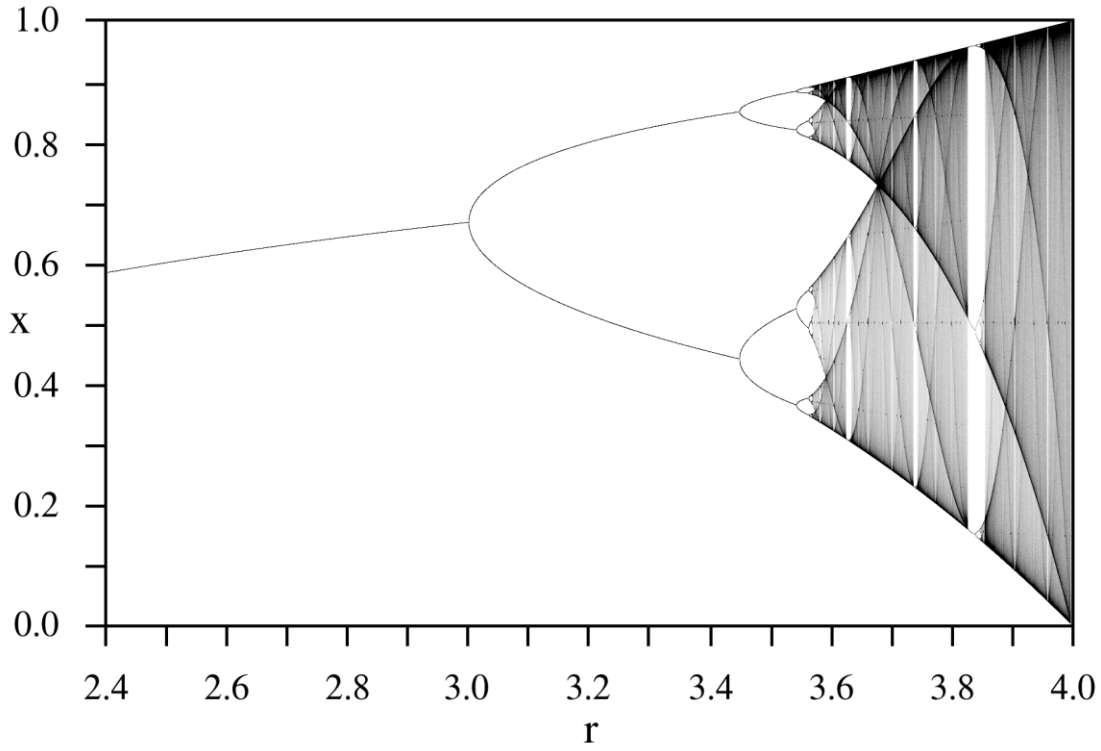


Figure 1: Bifurcation Diagram of the Logistic Map

The employment of the logistic map in the encryption algorithm adds more strength to the image encryption while being computationally efficient, which makes the approach excellent for real-time applications, especially on IoT devices with restricted resources like in embedded devices [22].

5. LIGHTWEIGHT PRESENT ALGORITHM

The PRESENT algorithm is a light-weight block cipher developed specially for resource-constrained environments such as the objects of IoT devices, smart cards, and other embedded systems. The algorithm was published by Bogdanov et al. in 2007. Given that it is very light, involving low power consumption and minimum hardware, it is particularly appealing because of its simplicity and efficiency, which are important for applications [8][23].

The PRESENT supports the key size of either 80 or 128 bits, where each block operates on 64-bit data. Its process of encryption follows elementary and simple operations consisting of both substitution in an S-box and permutation through a P-box that are essentially needed for encryption and decryption as well. The S-box in PRESENT contains 16 entries with which transformation of input data efficiently takes place [24]. The algorithm describes a structure using 31 rounds, with extreme repetition of confusion and diffusion to ensure security: two main components in effective encryption [25].

One of the major characteristics which are salient for PRESENT is its very lightweight profile, which has been achieved with minimal hardware requirements combined with an efficient round function. The small footprint makes it an excellent choice for implementation in very resource-poor devices, making it applicable in many scenarios, such as RFID and secure communications [26][27]. As the researchers have shown that the PRESENT algorithm is effective in both hardware and software environments, it confirms its appropriateness in the wide range of available platforms [28].

PRESENT has been intensely analyzed for security and displays resistance against a wide variety of attacks, including differential and linear cryptanalysis. It might even have fewer rounds than some block ciphers, such as AES, but alongside that S-box and overall design, it gives great protection against real threats [29].

It is also power and speed efficient to place it in an optimal way for lightweight cryptography. Several optimized implementations have been suggested to work on the algorithm for specific hardware architectures that make it highly usable in real-world applications. Since IoT devices are gaining pace, the demand for lightweight cryptographic solutions such as PRESENT will be growing, and it emphasizes the use of secure yet efficient encryption methods for the future use [30].

6. S-BOX IN THE AES

In the AES algorithm, an S-Box is very critical and plays a significant role in enhancing the encryption process security. The concept basically is that this S-Box is a type of look-up table, where each input byte is mapped into a different output byte in the process of encryption. This transformation makes it even more difficult to discern the association between the original data and the output generated after encryption. It is based on the mathematical framework of finite fields called GF(2^8) and makes use of both an inverse transformation and an affine transformation to enhance its resistance against several types of attacks, such as linear and differential cryptanalysis [31]. Designing an AES S-Box in careful detail so that each output byte is unique adds complexity to the encryption processes. It can be said that this feature supports better diffusion, which makes it challenging for potential attackers to decrypt the information. This feature of S-Box makes AES a robust choice for data encryption secure enough, as it plays a vital role in both the confusion and diffusion processes required for proper cryptography [32].

7. PROPOSED ALGORITHM

The proposed algorithm has been designed to be lightweight enough to execute faster than a significant number of other “image encryption” techniques discussed in the literature while maintaining uncompromised quality and security metrics for encryption. The approach taken in this regard entails using more permutations through a logistic map and adding an “AES S-Box” with an “XOR operation” for substitution. As discussed above, the AES S-Box is a nonlinear substitution that maps an 8-bit input into an “8-bit output” effectively.

Encryption Algorithm

As shown in Algorithm 1, the proposed encryption technique will improve the security of the plaintext image by combining multiple advanced techniques, such as hashing via BLAKE2b hashing, logistic map functions, and the PRESENT cipher.

First, in Step 1, we first initialize the key with BLAKE2b. In this step, we compute a hash that mixes together the plaintext image I, a pre-shared key Kshared and initialization vector IV. This hash is not just a random string of numbers; rather it has two important purposes: it is useful in establishing parameters for a logistic map, and it generates a specific key Kpresent that we will use next for the PRESENT encryption:.

In Step 2, we work with image preprocessing. We load the input image I and split it into its three major color components: Red, Green, and Blue. Then, we further fragment each color channel into smaller pieces called 64-bit blocks. This splitting prepares the image for the subsequent encryption steps.

In Step 3, we submerge into key generation by using the logistic map. Using the parameters we derived from our earlier BLAKE2b hash, we initialize the logistic map to produce a long series of random numbers, leading us to mix and match the blocks along the encryption process.

We now encrypt the above data in four 64-bit blocks in Step 4 by using the PRESENT encryption. Therefore, using our derived key Kpresent, we will make sure that the data within each block is well encrypted so that it becomes hard to decrypt by unauthorized users.

Once the blocks have been encrypted, we move into Step 5, where we further beef up the security with the help of AES S-box substitution. Here, we substitute some values in the blocks using AES S-box and influence the swaps involved with the chaotic values generated earlier, thereby giving yet another layer of complexity and confusion in the encrypted data.

Step 6 will finish with a block permutation; that is, the order of blocks is mixed up based on the chaotic sequence from the logistic map. This additional scrambling is well and quite considerably enhancing the security of encrypted data.

In Step 7, we then combine encrypted red, green, and blue channels into one single encrypted image E. It is in this step where everything that we have worked on comes together into one output.

Finally, Step 8 summarizes everything by saving the encrypted image E in a suitable format for storage or transmission. With this well-designed algorithm we are able to make use of several encryption techniques so the image data is kept secure and confidentially inaccessible.

Algorithm 1: Proposed Encryption Algorithm

Input:

Plaintext image I, Pre-shared key K_shared , Initialization vector IV, PRESENT parameters.

Output: Encrypted image E.

Step 1: BLAKE2b -Based Key Initialization

- Compute the **BLAKE2b hash** based on the plaintext image I, the pre-shared key K_shared, and the initialization vector IV.
- Use this hash to initialize the parameters for the **logistic map** and to derive the key K_present for the **PRESENT encryption**.

Step 2: Image Preprocessing

- Load the input image I.
- Separate the image into its three color channels: **Red**, **Green**, and **Blue**.
- Divide each channel into **64-bit blocks**.

Step 3: Key Generation Using the Logistic Map

- Use the **logistic map**, initialized with parameters from the BLAKE2b hash, to generate random numbers.
- Use these numbers for **block permutations** and **substitutions** during the encryption process.

Step 4: PRESENT Encryption

- • Use, for each 64-bit block, PRESENT encryption with K_present as the generated key, using the logistic map as a basis.

Step 5 AES S-box Substitution

- After applying the PRESENT encryption, use the AES S-box for substitution on the values within the block. Further augmenting confusion, the substitutions will depend on the chaotic values generated by the logistic map.

Step 6: Final Block Transposition

- Finally, permute the resulting blocks based on the chaotic sequence generated by the logistic map to further scramble the data.

Step 7: Reassemble Encrypted Image

- Combine the encrypted **Red**, **Green**, and **Blue** channels back into the encrypted image E.

Step 8: Save Encrypted Image

- Save the encrypted image E in a suitable format.

The proposed decryption algorithm, listed in Algorithm 2, is designed to recover the original plaintext image from its encrypted form, utilizing many of the same techniques used during the encryption process, including BLAKE2b hashing and the PRESENT cipher.

The process begins with **Step 1**, where we initialize the key using BLAKE2b. In this step, we use the BLAKE2b hash, along with the pre-shared key *Kshared* and the initialization vector *IV*, to regenerate the parameters required for the logistic map. This step also allows us to derive the key *Kpresent*, which is essential for the decryption process.

Next, in **Step 2**, we load the encrypted image *E*. This involves separating the image into its three primary color channels: Red, Green, and Blue. Each of these channels is further subdivided into groups that are termed as 64-bit blocks, preparing the data for the steps of decryption.

In Step 3, we reverse the substitution operations done through encryption by applying the inverse AES S-box. Once all the blocks are loaded, we apply the inverse AES S-box. This effectively reverses the substitutions and helps in recovering the values previously in use.

We now proceed to Step 4: the encryption of PRESENT. In this step, we decrypt every block under the coverage of the PRESENT cipher and the derived key Kpresent. This is an essential step in restoring the data inside each block to their original state.

The last step pertains to block permutation, Step 5. In this, an inverse permutation is applied based on the chaotic sequence obtained using the logistic map. This helps in restoring the original blocks, which may have gotten mixed up during the encryption process.

In step 6, we piece together the decrypted image by taking the Red, Green, and Blue channels and reassembling them into one single image I. This step, therefore, integrates all data that has been processed from various channels into one complete decrypted image for use.

In Step 7, we finally save the decrypted image I in the appropriate format. This means that our recovered image will be stored or shared based on the format applied. Thus, our overall well-structured decryption algorithm restores the original image successfully, therefore exposing the efficiency of our encryption and decryption algorithms.

Algorithm 2: Proposed Decryption Algorithm

Input:

Image E, the hash BLAKE2b, Pre-shared key K_shared, Initialization vector IV.

Output I. Decrypted image.

Step 1: BLAKE2b -Based Key Initialization

- Use the **BLAKE2b hash**, the pre-shared key K_shared, and the initialization vector IV to regenerate the parameters for the **logistic map** and derive the key K_present for decryption.

Step 2: Load Encrypted Image

- Load the encrypted image E.
- Separate the image into its **Red**, **Green**, and **Blue** channels.
- Divide each channel into **64-bit blocks**.

Step 3: Inverse AES S-box Substitution

- After loading the blocks, pass them through the **inverse AES S-box** to undo the substitutions.

Step 4: PRESENT Decryption

- Decrypt the blocks using **PRESENT decryption** with the key K_present.

Step 5: Final Block Permutation

- Apply the **inverse permutation** based on the chaotic sequence generated by the logistic map to restore the original block order.

Step 6: Reassemble Decrypted Image

- Combine the decrypted Red, Green, and Blue channels back into the final decrypted image I.

Step 7: Save Decrypted Image

- Save the decrypted image I in the appropriate format.

4. EXPERIMENTAL RESULTS

In order to evaluate our proposed algorithm, we used a set of standard test images that are often cited in the literature. These images are displayed in Figure 2. Our tests covered 24-bit color images, with a resolution of 128 × 128, pixels.

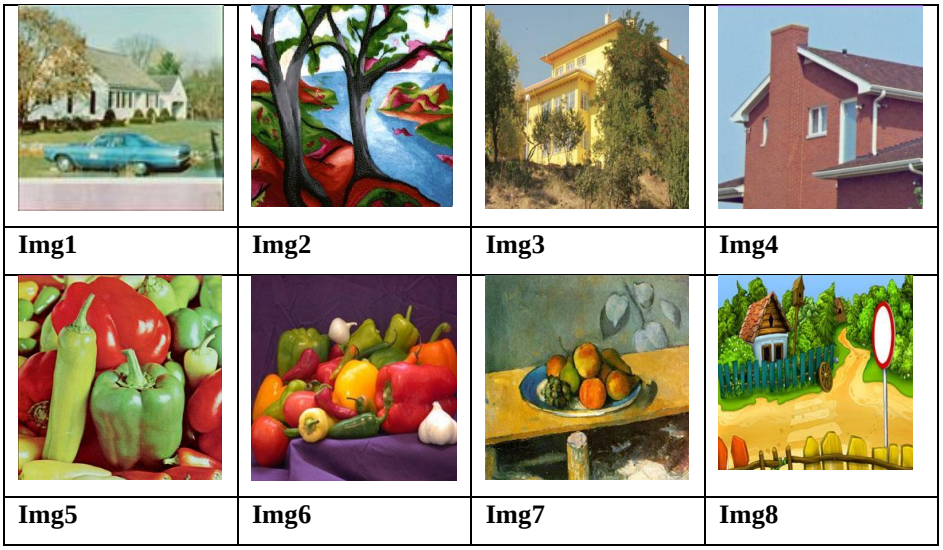


Figure 2. Images Employed for Testing the Proposed Algorithm

Figure 3 shows the results of applying the proposed algorithm on the standard test images.

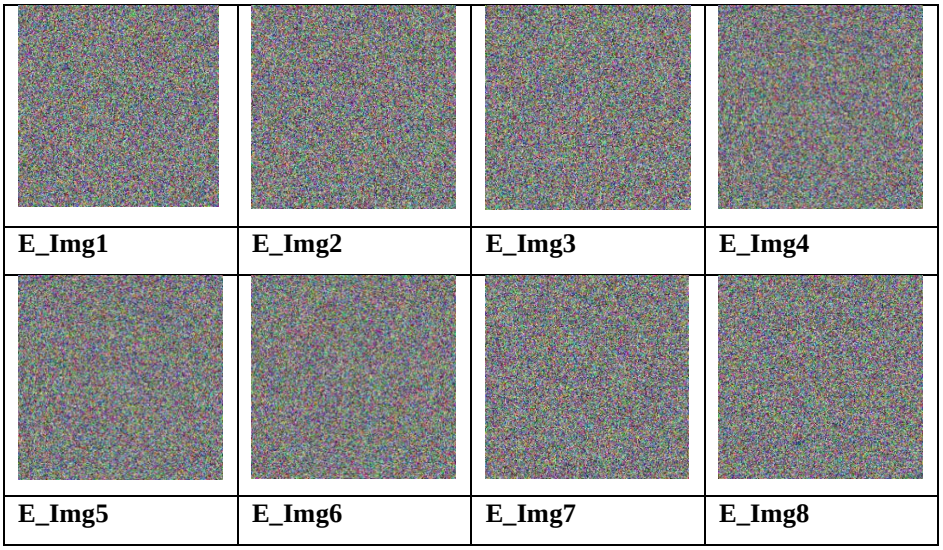


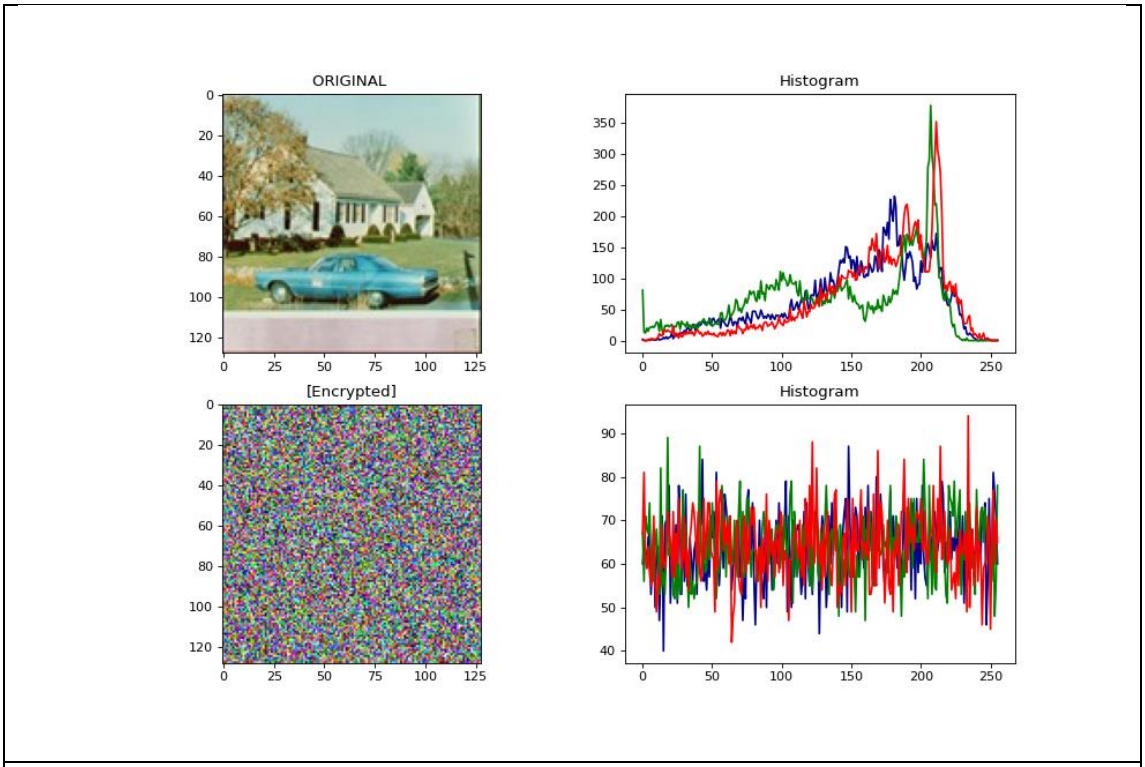
Figure 3. The encrypted images using the Proposed Algorithm

For an in-depth evaluation of the performance of our algorithm, we compared it against those existing techniques in related studies. We considered some important metrics, time complexities, correlation coefficients, and some statistical tests including histogram analysis and chi-square tests. We also considered both local and global entropies so that all aspects of how our algorithm performs are looked into in terms of encryption quality. More commonly used are Maximum Deviation, Irregular Deviation, Deviations from a “Uniform Histogram”, “Peak Signal-to-Noise Ratio or PSNR”, “Mean Absolute Error (MAE)”, “Mean Squared Error or MSE”, and many more.

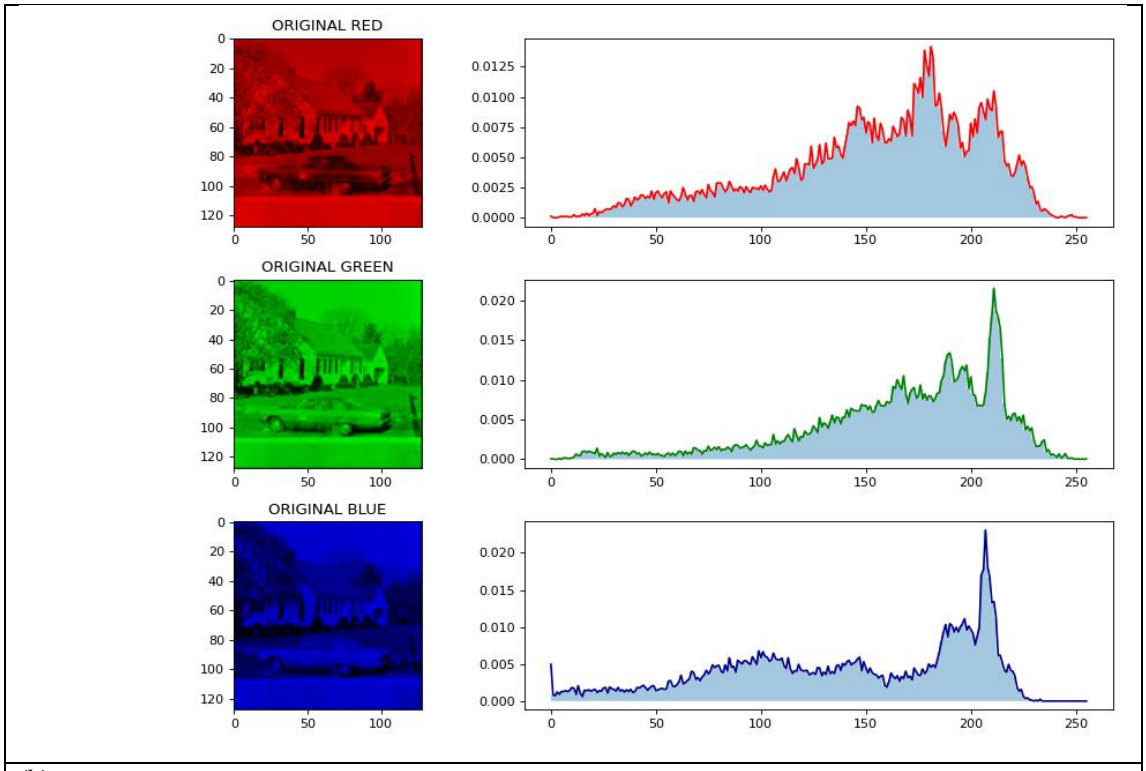
Further, the algorithm has also achieved resilience to differential assault by calculating the “Number of Pixels Change Rate (NPCR)” and “Unified Average Changing Intensity (UACI)”. Additionally, features like homogeneity analysis, contrast analysis, energy analysis, key space, and key sensitivity analysis are also included in the algorithm's potential to withstand noise and data loss.

4.1. Histogram Analysis

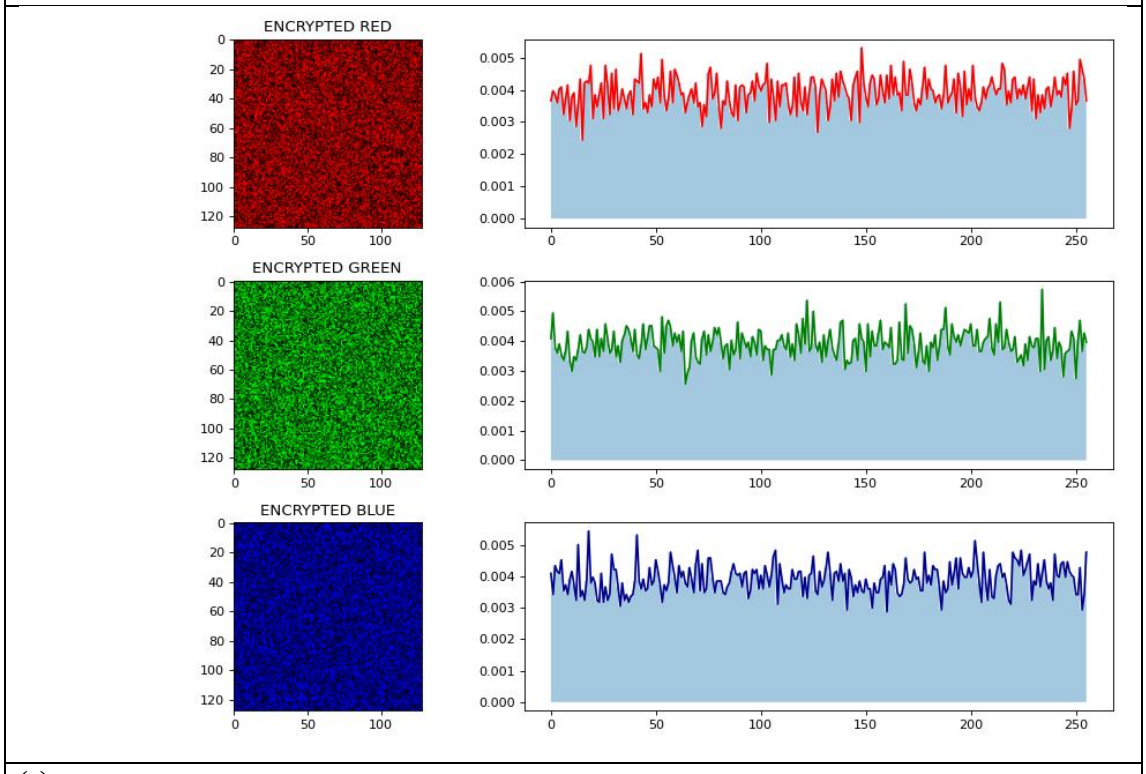
An appropriate histogram indicating the distribution of pixel values in an image represents an efficient way to check whether the histogram of the cypher picture is uniform. A uniform histogram of the cypher picture will surely ensure a better defence against statistical attacks. Histograms of the complete picture for both original and encrypted versions are given in Figures 4-6. Consequently, these histograms show the distribution of the brightness of the pixels is broad. It indicates that this histogram of the encrypted version is uniformly spread out where it was having sharp points for the original image's histogram, so the encryption process worked well. We then plot the histograms for the original and the compressed images with much more detail. These are shown for the “Red, Green, and Blue” channels in sequence. Histograms for the encrypted image again show random, uniform distributions over all channels, thus proving that the strength of the technique is in obscuring the original pixel values; the detailed presentation allows us to inspect effects of encryption on each colour component separately. A histogram analysis indicates that the proposed scheme conceals information of the original images satisfactorily and is resistant against histogram-based attacks. From a statistical point of view, it looks like there is no correlation between the two.



(a)

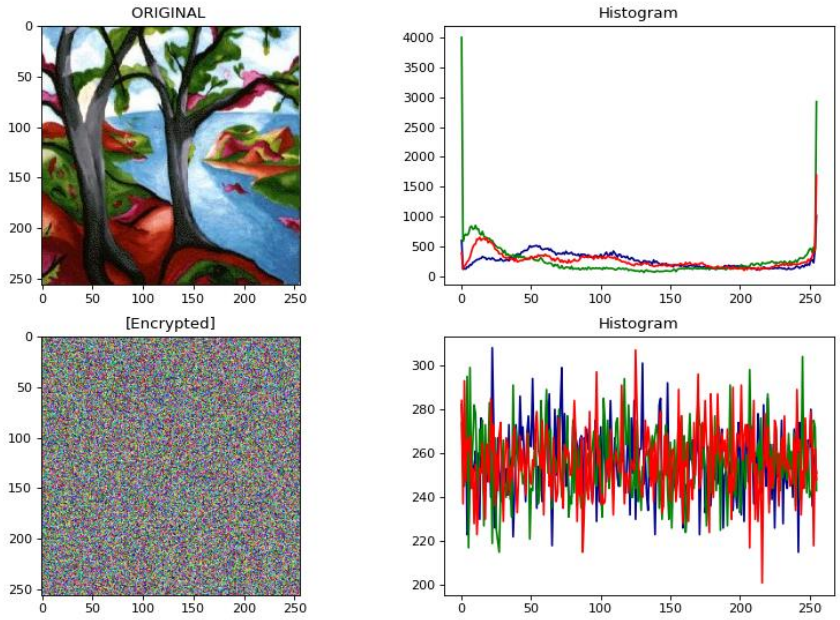


(b)

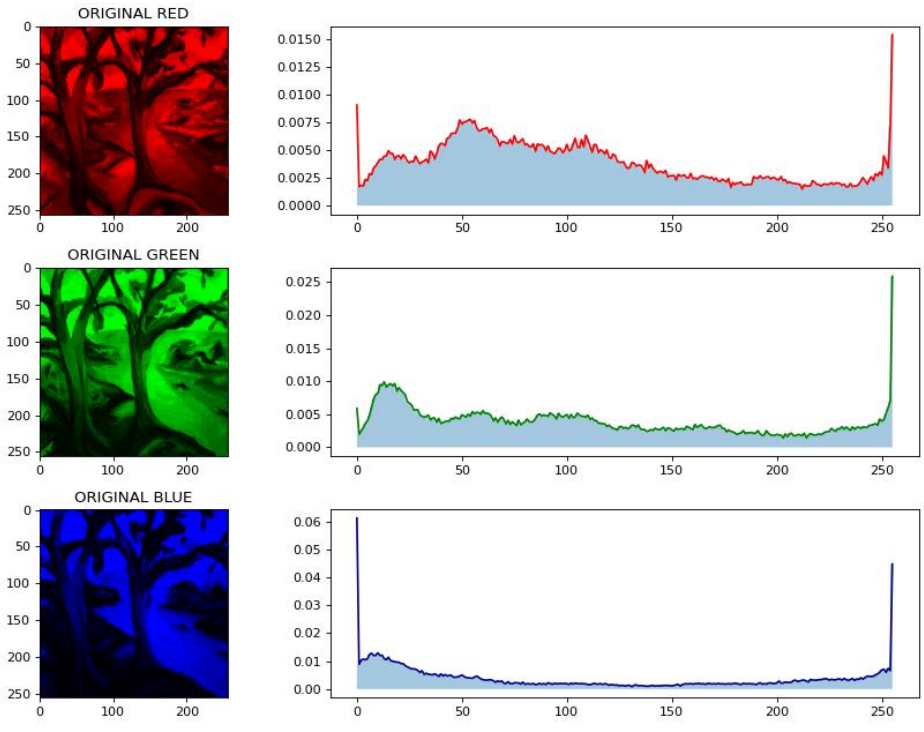


(c)

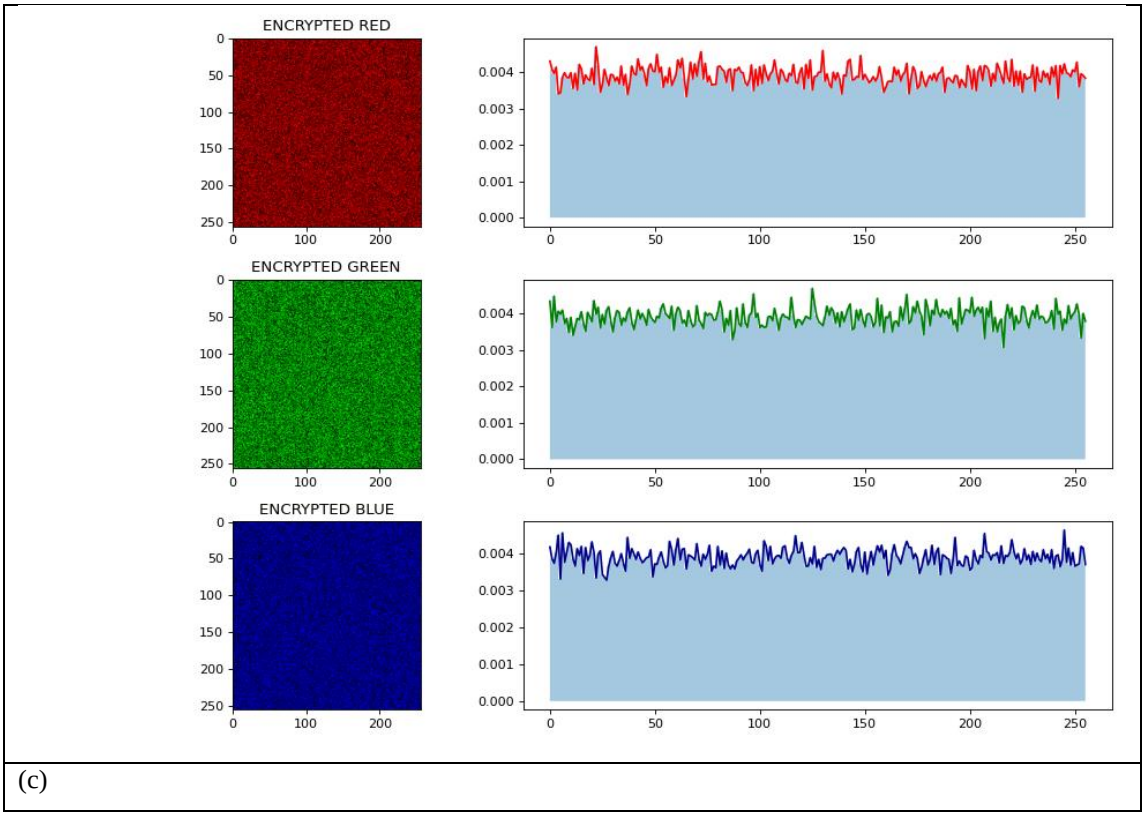
Figure 4: Histograms of Img1 and its corresponding encrypted version: (a) Img1 and Encrypted Img1 histograms, (b) Histograms of the Img1 Represented in Red, Green, and Blue Channels, (c) Histograms of the Encrypted Img1 Represented in Red, Green, and Blue Channels.


ORIGINAL
Histogram
[Encrypted]
Histogram

(a)

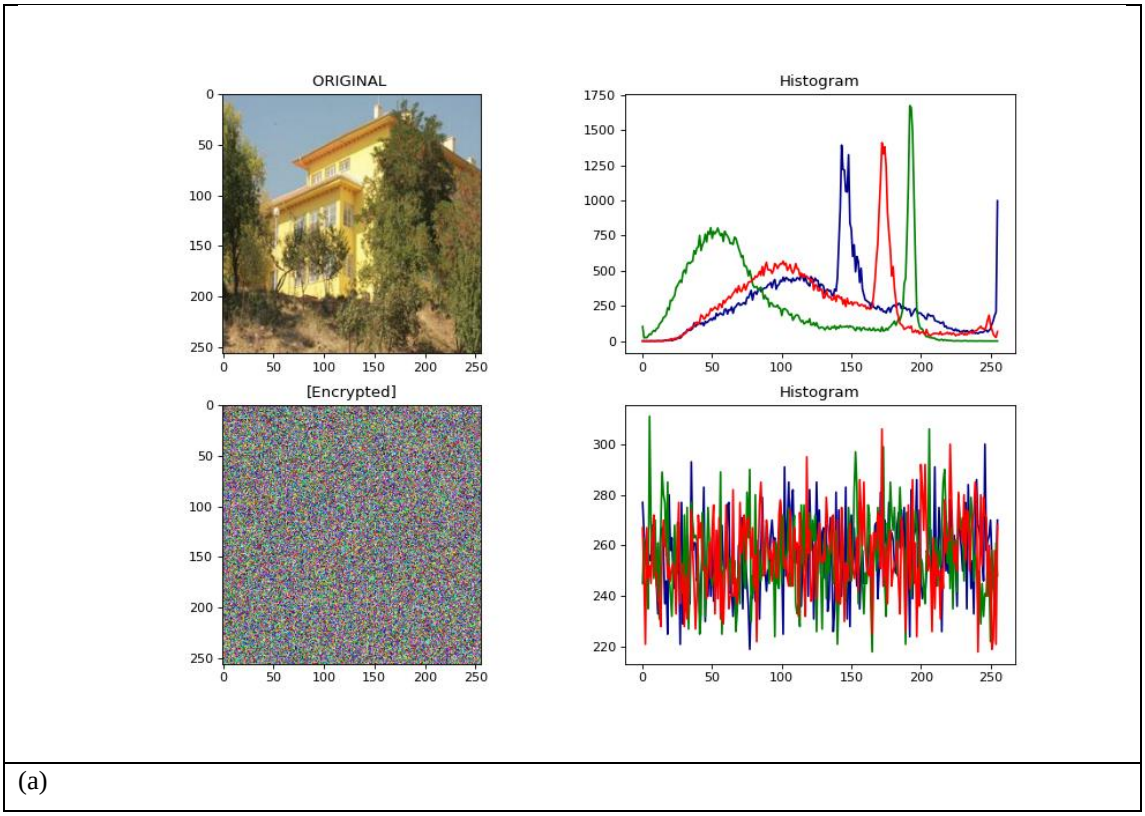

ORIGINAL RED
ORIGINAL GREEN
ORIGINAL BLUE

(b)

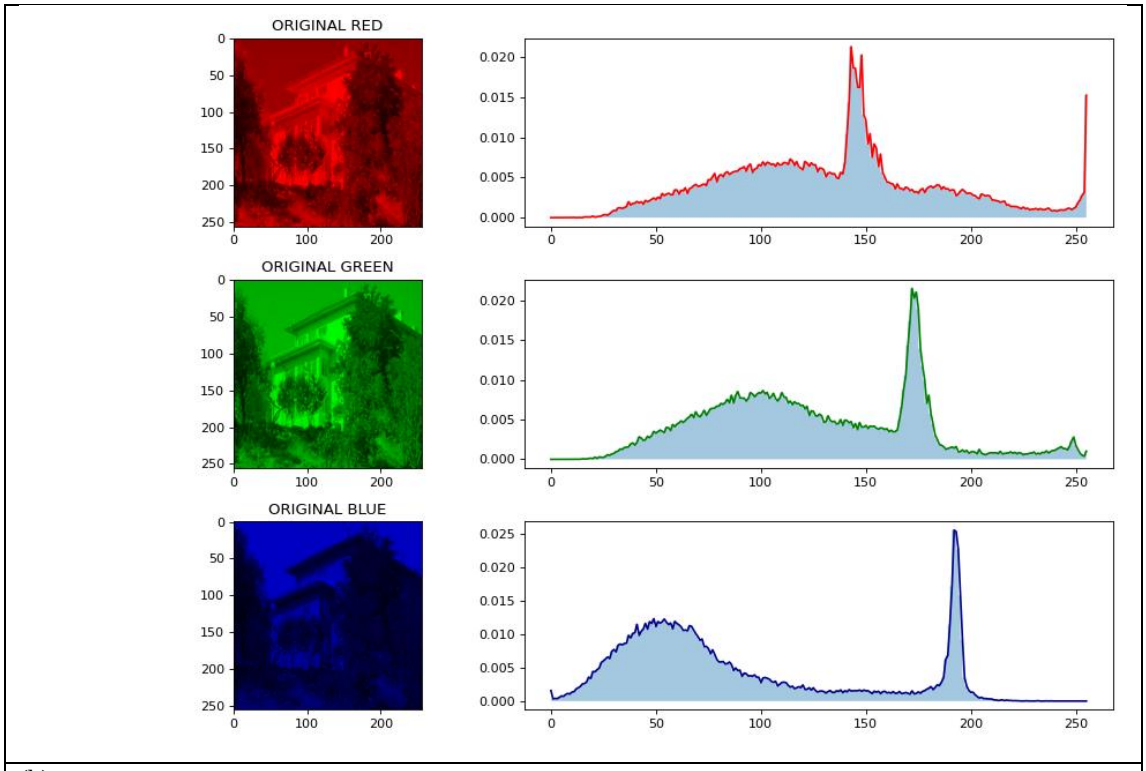


(c)

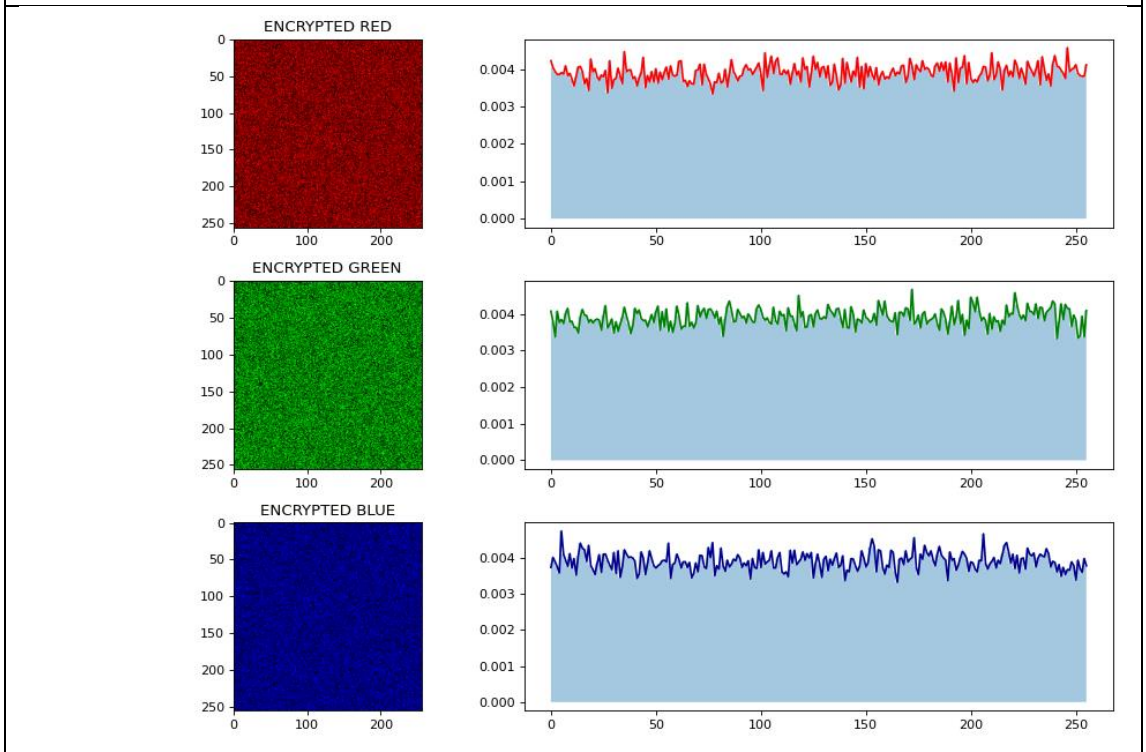
Figure 5: Histograms of Img2 and its corresponding encrypted version: (a) Img2 and Encrypted Img2 histograms, (b) Histograms of the Img2 Represented in Red, Green, and Blue Channels, (c) Histograms of the Encrypted Img2 Represented in Red, Green, and Blue Channels.



(a)



(b)



(c)

Figure 6: Histograms of Img3 and its corresponding encrypted version: (a) Img3 and Encrypted Img3 histograms, (b) Histograms of the Img3 Represented in Red, Green, and Blue Channels, (c) Histograms of the Encrypted Img3 Represented in Red, Green, and Blue Channels.

4.2. Correlation Coefficient Analysis

In typical images, pixels often exhibit a strong statistical correlation, meaning that one can predict a pixel's value based on its neighboring pixels. This predictability can weaken the encrypted image's ability to withstand statistical attacks. As a result, cipher algorithms focus on reducing the statistical correlations among pixels in the encrypted image. This reduction is crucial to prevent potential attackers from exploiting the relationships between adjacent pixels. A correlation value of 0 would suggest that no two pixels are connected, and a value of 1 or -1 suggests perfect positive/negative correlation. If the encryption technique is effective, then the correlation values of the encrypted image should be close to zero, meaning the encryption has successfully eliminated the original patterns. To be sure, we compare the encrypted version of the image with the original image and note any difference. We then calculate and report the correlation value for each pixel along the horizontal, vertical, and diagonal axes.

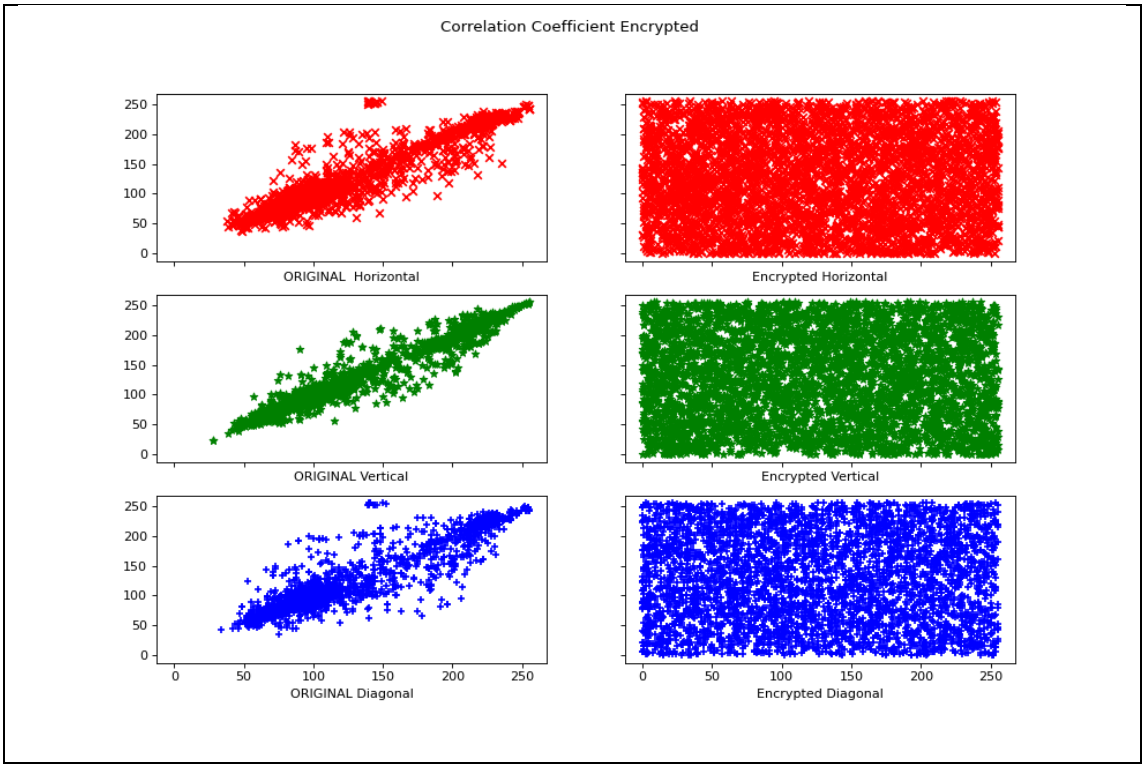
The formula to compute the correlation coefficient between the original image and the encrypted image can be represented as follows:

$$CC_{I,E} = \frac{\sum_{h=1}^{H}\sum_{w=1}^{W}\left(I_{(h,w)} - \bar{I}\right)\left(E_{(h,w)} - \bar{E}\right)}{\sqrt{\sum_{h=1}^{H}\sum_{w=1}^{W}\left(I_{(h,w)} - \bar{I}\right)^2 \sum_{h=1}^{H}\sum_{w=1}^{W}\left(E_{(h,w)} - \bar{E}\right)^2}} \ldots (2)$$

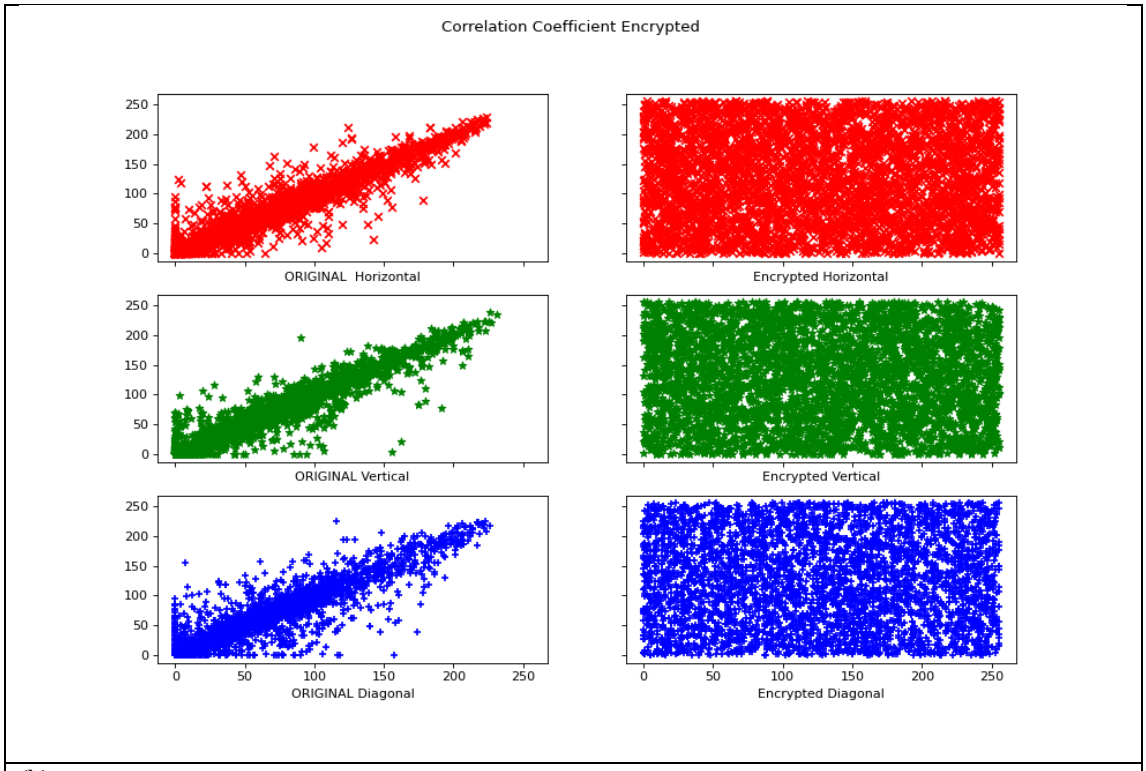
Where:

- $I_{(h,w)}$ and $E_{(h,w)}$ represent the pixel values at index (h, w) of the original and encrypted images, respectively.
- $\bar{I}$ and $\bar{E}$ denote the mean pixel values of the original and encrypted images.
- H and W are the height and width of the images, respectively.

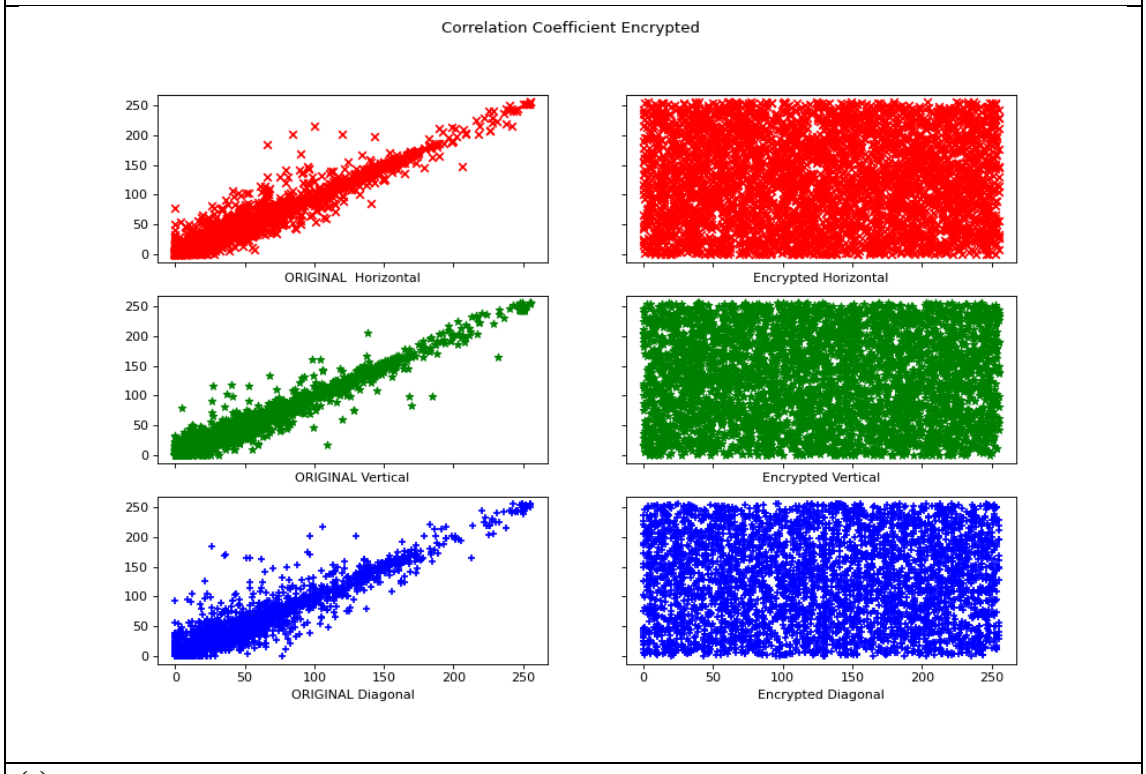
Figure 7 shows the "correlation analysis" of the original images across three directions, demonstrating how closely related each pixel is to its neighbors. Figure 8: Pixels in the encrypted image are spread out seemingly randomly. Table 1 lists numeric values of correlation coefficients in all three dimensions: vertical, horizontal, and diagonal between the original and encrypted images. This analysis points out the fact that encryption helps to increase security because it breaks the predictability of the pixel patterns.



(a)



(b)



(c)

Figure 7: Correlation of pixel distribution in different directions for up, down, left, right, and diagonal orientations for the following images: (a) Img4 and its corresponding encrypted version, (b) Img5 and its corresponding encrypted version, (c) Img6 and its corresponding encrypted version.

Table 1. Correlation coefficient analysis			
Image	**Horizontal Correlation**	**Vertical Correlation**	**Diagonal Correlation**
Img1	**-0.009815**	**-0.026743**	**-0.039869**
Img2	**-0.017351**	**0.022416**	**0.021109**
Img3	**-0.014127**	**0.027646**	**-0.028889**
Img4	**0.012768**	**-0.004399**	**0.007099**
Img5	**0.00218**	**-0.024143**	**-0.004995**
Img6	**-0.002192**	**0.006495**	**0.008509**
Img7	**-0.006727**	**0.001341**	**0.011748**
Img8	**0.021751**	**-0.006911**	**0.003824**

4.3. Entropy analysis

Using data entropy, predictability, as well as randomness in picture encryption, determines the kind of image encryption. It is an essential component in the discovery of the degree of functionality of the encryption method during the camouflage of the original picture. The higher the entropy of the cypher picture, the more secure it is because of its higher randomness, thereby creating a lot of barriers for an attacker to decipher the encrypted image.

In mathematical terms, entropy is expressed as:

$$\mathrm{IE(k)} = -\sum_{d=0}^{D-1} p(k_d)\log_2[p(k_d)] \ldots (3)$$

Where:

- $p(k_d)$ is the probability of occurrence of pixel intensity k_d.
- *D* represents the number of distinct pixel values in the image.

Ideally, for a cipher image produced by any encryption algorithm to be secure, its entropy should approach the ideal maximum of 8; it would mean that the encrypted image is extremely 'random', making it more difficult to break the encryption by brute force. A cipher image that has lower entropy might contain patterns and predictable elements that can make it easier for its cryptanalysis through brute force methods. Encrypted by achieving near perfect entropy, the encryption effectively denies the attackers the potential to exploit identifiable patterns and therefore benefits the overall security delivered through the cipher image. As illustrated by Table 2, this proposed algorithm reveals resilience to the information-entropy attacks. Maintaining high levels of unpredictability ensures that meaningful pattern extraction is difficult or impossible for attackers from the encrypted image. This makes the algorithm very effective in protecting sensitive data and withstanding those types of cryptographic attacks.

Table 2: Results of Information Entropy Tests on Sample Images Using the Proposed Encryption Method

Image Name	Entropy
Img1	7.995991
Img2	7.998976
Img3	7.999091
Img4	7.999031
Img5	7.999166
Img6	7.998967
Img7	7.999069
Img8	7.999145

4.4. Encryption Quality

The grade of encryption quality is necessary to assess the overall success of the method. Several tests are most often utilized to calculate the good encryption process has been in hiding the image from the plaintext. MSE and PSNR are these comparison metrics used for evaluation between the plaintext images and their encrypted/cipher images. These experiments gave us an insight on the degree up to which this encryption transforms the image and conceals the source. The MSE is a mathematical representation of the average squared difference between two images' pixel values. This allows a clear picture of how much the images have been distorted after encryption. The formula for the MSE of a raw image and the cypher image encrypting it is given below:

$$\text{MSE} = \frac{1}{H * W} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} [I(h,w) - E(h,w)]^2 \ldots (4)$$

H and W represent the length and width of the picture respectively. I(h,w) and E(h,w) show the pixel values of the plaintext and cypher photos at the same point in space (h,w).

In most cases, an MSE above 30 implies that the secured picture is quite different from the original. It has been observed that the method of encryption is secure. This ensures that the encryption effectively obscures the content.

The commonly used technique to estimate the quality of image encryption is the Peak Signal-to-Noise Ratio, which measures the noise or distortion involved during encryption. It compares the original (plain) image to its encrypted (cipher) counterpart. The PSNR is mathematically defined as:

$$\text{PSNR} = 20 \log_{10} \frac{MAX_I}{\sqrt{MSE}} \ldots (5)$$

Where:

- MAX_I represents the highest possible pixel value, typically 255 for 8-bit grayscale images.
- MSE stands for Mean Squared Error, which calculates the average squared difference between pixel values of the two images (as outlined earlier in Equation 4).

In general, a higher PSNR indicates that the two images are very similar, meaning there is less distortion or noise between them. However, in the context of encryption, a lower PSNR is desired as it signifies greater dissimilarity between the plaintext and cipher image, which implies stronger encryption.

Table 3 depicts the MSE results and PSNR values of the suggested encryption algorithm. The high MSE values obtained prove that the encryption quality of the algorithm is good. Easy distortion and hiding the original picture can be seen. There is a clear difference between images of plaintext and ciphertext as well as low PSNR values, which add to encryption strength. This difference is a positive sign of increased security, indicating the encryption technique has successfully reduced any visual similarity between the two.

Table 3. MSE and PSNR values

Image	MSE	PSNR
Img1	**69.924216**	**8.612428**
Img2	**97.082123**	**7.187311**
Img3	**67.878356**	**8.741391**
Img4	**65.187351**	**8.917071**
Img5	**78.345249**	**8.118577**
Img6	**87.183626**	**7.654355**
Img7	**72.711146**	**8.442694**
Img8	**94.349447**	**7.311399**

4.5. Resistance to Differential Attacks

Differential attacks, which are sometimes termed selected plaintext attacks, represent some ways an attacker uses to probe the relationship between an encrypted version of an image and its plaintext equivalent. Fundamentally, such attacks involve encrypting two images that are of only one bit difference and then looking for similarity or pattern in the decrypted images.

If the plaintext is altered, no matter how small the change is - one pixel - an effective encryption algorithm should result in a cypher picture that is practically entirely different. In fact, the security of an encryption scheme is enhanced if an encrypted technique is more sensitive to the plaintext. Two essential tests that academics frequently use to assess the robustness of an algorithm against differential attacks include "Number of Pixels Change Rate (NPCh)" and "Unified Average Change Intensity (UACI)". These tests create a basis that shows well how it can protect the system from possible threats. Considering the protection of an encryption algorithm against differential attacks, there is an important metric: the Number of Pixels Change Rate, shortly the NPCR. In a nutshell, it compares two cipher images which are produced from the same original but by changing only one bit in the plaintext.

To calculate NPCR, we use the following formula:

$$NPCR = \sum_{h=1}^{H} \sum_{w=1}^{W} D(h,w) \dots (5)$$

Where:

$$D(h,w) = \begin{cases} 0 & if E_1(h,w) = E_2(h,w) \\ 1 & if E_1(h,w) \neq E_2(h,w) \end{cases}$$

H and W denote the height and width of the images, whereas E1 and E2 are two encrypted images obtained from the same original images. E1 (h,w) and E2 (h,w) are pixel values for position (h,w) of these images. The function D(h,w) checks a difference at every pixel location-returns 0 if pixels appear identical and 1 if they differ.

Application of UACI to image encryption: Two encrypted images generated from the same original image but differing by one bit in the first input are compared based on the variance of the average pixel intensity. This metric is relevant to security as it indicates the susceptibility of an encryption method to slight alterations of the input image.

UACI is computed using the following formula:

$$UACI = \frac{1}{H * W} \left[\frac{\sum_{h=1}^{H} \sum_{w=1}^{w} |E_1(h,w) - E_2(h,w)|}{2^n - 1} \right] * 100\% \dots (6)$$

Where:

- H and W are the image's height and width.
- $E_1(h,w)$ and $E_2(h,w)$ represent the pixel values at position (h,w) in two encrypted images that were created from a slightly altered original image.
- n is the number of bits representing each pixel.

With a higher value of UACI, the differential attacks can be resisted better because it is more sensitive to slight changes in source images.

Table of Section 4 Images of the results of the proposed methodology are given along with their corresponding NPCR and UACI values. The NPCR of an encryption algorithm for a small change in plaintext is proportionate to it. The value of UACI reflects inversely proportional the capacity of the algorithm in modifying the picture data due to minor changes.

Table 4. NPCR and UACI values

Image	NPCR	UACI
Img1	99.6134%	33.4885%

Img2	**99.6069%**	**33.4769%**
Img3	**99.6196%**	**33.5093%**
Img4	**99.6024%**	**33.4702%**
Img5	**99.6097%**	**33.4791%**
Img6	**99.6139%**	**33.4912%**
Img7	**99.6176%**	**33.5047%**
Img8	**99.6077%**	**24.4753%**

4.6. Key Space

The key space is the set of all possible keys which a cryptosystem may use. Additionally, to resist brute force attacks, the key space should be large enough to have greater than 2100 possible keys [39].

In the proposed approach, a very secure combination of a "256-bit pre-shared secret key", the original image hash generated by BLAKE2b, and a 256-bit initialisation vector (IV) was used to generate a "256-bit key". Then, the logistic map was updated through using this 256-bit security key, which was generated through XORing of both numbers. Because of this, cryptographic security is high; the key space size of 2256 offers a big enough range of strong potential keys that they are resistant to brute-force assaults.

5. CONCLUSIONS

The paper introduces a simple, efficient, and secure image encryption algorithm based on chaotic systems. In its formulation, the algorithm initializes the logistic map parameters using the BLAKE2b hash function, which depends on an input image, a preshared secret key, and an initialization vector, IV. During encryption and decryption, chaotic keys are produced through logistic maps in both phases. These keys are then combined with the lightweight PRESENT algorithm and the AES S-Box for pixel substitution. Combination effectively increased the randomness and unpredictability of the cipher image, resulting in stronger encryption security.

The experimental evaluation in the proposed method reveals that it operates well in areas such as correlation coefficient, histogram analysis, and entropy with other security, quality, and efficiency criteria. The presented method is superior to comparable picture encryption techniques in the set of experiments measuring encryption quality in terms of PSNR and MSE. Another promising aspect revealed by the algorithm is defense against differential assaults, and this is validated by the results of the NPCR and UACI tests.

Although the results are promising, there is still room for improvement. While logistic maps offer efficiency, they have limitations related to key space and unpredictability. Future research could explore higher-dimensional or hybrid chaotic systems, such as the Lorenz, Henon, or Chen systems, which provide greater complexity, better pseudorandom number generation, and a larger key space. This would improve security against brute force and statistical attacks.

Furthermore, instead of relying on a fixed AES S-Box for substitution, future developments could focus on dynamically generating S-Boxes based on the chaotic map's initial conditions. This approach would improve non-linearity and unpredictability, making it harder for attackers to decipher the encryption process.

Another potential enhancement lies in adapting the proposed algorithm for parallel computing to boost performance in large-scale or real-time applications. Implementing the algorithm on GPUs or within distributed systems could significantly reduce both encryption and decryption times, making it more suitable for high-resolution image encryption tasks.

REFERENCES

[1] Gokcay, Erhan, and Hakan Tora. "A novel data encryption method using an interlaced chaotic transform." *Expert Systems with Applications* 237 (2024): 121494.

[2] Alghamdi, Yousef, and Arslan Munir. "Image encryption algorithms: A survey of design and evaluation metrics." *Journal of Cybersecurity and Privacy* 4.1 (2024): 126-152.

[3] Jithin, K. C., and Syam Sankar. "Colour image encryption algorithm combining Arnold map, DNA sequence operation, and a Mandelbrot set." *Journal of Information Security and Applications* 50 (2020): 102428.

[4] Gupta, Manish, et al. "An efficient image encryption technique based on two-level security for internet of things." *Multimedia Tools and Applications* 82.4 (2023): 5091-5111.

[5] Mahendiran, N., and C. Deepa. "A comprehensive analysis on image encryption and compression techniques with the assessment of performance evaluation metrics." *SN Computer Science* 2.1 (2021): 29.

[6] Winarno, Edy, Kristiawan Nugroho, and Prajanto Wahyu Adi. "Combined interleaved pattern to improve confusion-diffusion image encryption based on hyperchaotic system." *IEEE Access* 11 (2023): 69005-69021.

[7] Bhagat, Vijesh, et al. "Lightweight cryptographic algorithms based on different model architectures: A systematic review and futuristic applications." *Concurrency and Computation: Practice and Experience* 35.1 (2023): e7425.

[8] Bogdanov, Andrey, et al. "PRESENT: An ultra-lightweight block cipher." *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*. Springer Berlin Heidelberg, 2007.

[9] Jasim, Suha Husam, Haider Kadhim Hoomod, and Khalid Ali Hussein. "Image Encryption Based on Hybrid Parallel Algorithm: DES-Present Using 2D-Chaotic System." *International Journal of Safety & Security Engineering* 14.2 (2024).

[10] Mahdi, Mohammed Salih, Raghad Abdulaali Azeez, and Nidaa Falih Hassan. "A proposed lightweight image encryption using ChaCha with hyperchaotic maps." *Periodicals of Engineering and Natural Sciences* 8.4 (2020): 2138-2145.

[11] Alshammari, Badr M., et al. "Implementing a symmetric lightweight cryptosystem in highly constrained IoT devices by using a chaotic S-box." *Symmetry* 13.1 (2021): 129.

[12] Alanezi, Ahmad, et al. "Securing Digital Images through Simple Permutation-Substitution Mechanism in Cloud-Based Smart City Environment." *Security and Communication Networks* 2021.1 (2021): 6615512.

[13] Arif, Jameel, et al. "A novel chaotic permutation-substitution image encryption scheme based on logistic map and random substitution." *IEEE Access* 10 (2022): 12966-12982.

[14] Lu, Qing, Congxu Zhu, and Xiaoheng Deng. "An efficient image encryption scheme based on the LSS chaotic map and single S-box." *Ieee Access* 8 (2020): 25664-25678.

[15] Fadhil, Meryam Saad, et al. "A new lightweight AES using a combination of chaotic systems." *2020 1st. Information Technology To Enhance e-learning and Other Application (IT-ELA*. IEEE, 2020.

[16] Aumasson, Jean-Philippe, et al. "The hash function BLAKE." (2014): 978-3.

[17] Sadeghi-Nasab, Alireza, and Vahid Rafe. "A comprehensive review of the security flaws of hashing algorithms." *Journal of Computer Virology and Hacking Techniques* 19.2 (2023): 287-302.

[18] Kuznetsov, Olexandr, et al. "Evaluating Hashing Algorithms in the Age of ASIC Resistance." *ITTAP*. 2023.

[19] Khalil, Noura, Amany Sarhan, and Mahmoud AM Alshewimy. "An efficient color/grayscale image encryption scheme based on hybrid chaotic maps." *Optics & Laser Technology* 143 (2021): 107326.

[20] Strogatz, Steven H. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[21] Andono, Pulung Nurtantio. "Improved pixel and bit confusion-diffusion based on mixed chaos and hash operation for image encryption." *Ieee Access* 10 (2022): 115143-115156.

[22] Trujillo-Toledo, Diego Armando, et al. "Real-time medical image encryption for H-IoT applications using improved sequences from chaotic maps." *Integration* 90 (2023): 131-145.

[23] Thakor, Vishal A., Mohammad Abdur Razzaque, and Muhammad RA Khandaker. "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities." *IEEE Access* 9 (2021): 28177-28193.

[24] Fotovvat, Amir, et al. "Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes." *IEEE Internet of Things Journal* 8.10 (2020): 8279-8290.

[25] Jammula, Mounika, Venkata Mani Vakamulla, and Sai Krishna Kondoju. "Performance evaluation of lightweight cryptographic algorithms for heterogeneous IoT environment." *Journal of Interconnection Networks* 22.Supp01 (2022): 2141031.

[26] Singh, Saurabh, et al. "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions." *Journal of Ambient Intelligence and Humanized Computing* (2024): 1-18.

[27] Sehrawat, Deepti, Nasib Singh Gill, and Munisha Devi. "Comparative analysis of lightweight block ciphers in IoT-enabled smart environment." *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 2019.

[28] Bharathi, R., and N. Parvatham. "Light-Weight Present Block Cipher Model for IoT Security on FPGA." *Intelligent Automation & Soft Computing* 33.1 (2022).

[29] Liu, Guo-Qiang, and Chen-Hui Jin. "Linear Cryptanalysis of PRESENT-like Ciphers with Secret Permutation." *The Computer Journal* 59.4 (2016): 549-558.

[30] Dhanda, Sumit Singh, Brahmjit Singh, and Poonam Jindal. "Lightweight cryptography: a solution to secure IoT." *Wireless Personal Communications* 112.3 (2020): 1947-1980.

[31] Tita, Faldy, Adi Setiawan, and Bambang Susanto. "Performance of S-Boxes Constructed by Irreducible Polynomials on GF (2 8)." *2024 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*. IEEE, 2024.

[32] Waqas, Umer, et al. "Cryptographic strength evaluation of AES s-box variants." *International Journal of Information and Computer Security* 14.3-4 (2021): 263-280.