Development of an Image Encryption Algorithm using Latin Square Matrix and Logistics Map

Emmanuel Oluwatobi Asani[1], Godsfavour Biety-Nwanju[2], Abidemi Emmanuel Adeniyi[3],
Salil Bharany[4]*, Ashraf Osman Ibrahim[5]*, Anas W. Abulfaraj[6], Wamda Nagmeldin[7]
Department of Computer Science, Landmark University, Nigeria[1, 2]
Landmark University SDG 11 Group, Landmark University, Nigeria[1]
Department of Computer Sciences, Precious Cornerstone University, Ibadan, Nigeria[3]
Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab 144402, India[4]
Creative Advanced Machine Intelligence Research Centre, Faculty of Computing and Informatics,
Universiti Malaysia Sabah, Jalan UMS, 88400 Kota Kinabalu, Sabah, Malaysia[5]
Department of Information Systems, King Abdulaziz University, P.O. Box 344, Rabigh; 21911, Saudi Arabia[6]
Department of Information Systems, College of Computer Engineering and Sciences,
Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia[7]

Abstract—**The goal of this study was to develop a robust image cryptographic scheme based on Latin Square Matrix and Logistics Map, capable of effectively securing sensitive data. Logistics mapping is a comparatively strong chaos system which enciphers with an unpredictability that significantly reduces the chance of deciphering. Additionally, the Latin square matrix stands out for its uniform histogram distribution, thereby bolstering its encryption's potency. The consequent integration of these algorithms in this study was therefore grounded in the scientific rationale of establishing a strong and resilient cypher technique. The study provides a new chaos-based method and extends the application of the probabilistic approach to the domain of symmetric key image encryption. Permutation and substitution approaches of image encryption were deployed to address the issue of images volume and differing sizes. The issue of misplaced pixel positions in the image was also adequately addressed, making it an effective method for image encryption. The hybrid technique was simulated on image data and evaluated to gauge its performance. Results showed that the algorithm was able to securely protect image data and the private information associated with them, while also making it very difficult for unauthorized users to decrypt the information. The average encryption time of 184(μs) on seven (7) images showed that it could to be deployed for real-time systems. The proposed method obtained an average entropy of 7.9398 with key space of 1.17x1077 and an average avalanche effect (%) of 49.9823 confirming the security and resilience of the developed method.**

Keywords—Image encryption; algorithm; logistics map; Latin square matrix; chaos technology

I. Introduction

As a result of the massive development of digital information technology, an ever-increasing volume of image data is being generated and distributed over the interconnected networks of computing infrastructure. These images typically contain confidential information such as trade secrets, military secrets, confidential medical reports and other types of secrets. It is therefore critical to protect sensitive data contained in these images from unauthorized access; many industries and fields rely on the secure transmission of such data [1]. As a result, image transmission security has emerged as one of the most pressing topics in information sciences [1].

The method that is utilized to protect the confidentiality of images is known as cryptography. Cryptography is a method for safely transmitting data and communications by utilizing certain keys. This ensures that only the receiver to whom the information is addressed is aware of the true content of the message, hence preventing unauthorized access [2, 3]. It has a significant impact on the conversations that take place over mobile phones, as well as on e-commerce, the sending of emails, the transmission of financial information, and other areas of an individual's day-to-day life. The prefix "crypt" in the word "cryptography" refers to something that is "hidden" or "written" [4]. Information is encrypted by the use of mathematical presumptions and algorithms in the field of cryptography. These techniques are used to encode information before it is delivered, making it harder to decipher the information from its original form. Cryptography provides privacy by ensuring that transmitted data isn't known by external parties, it is reliable because it ensures there is no form of modification during storage and transfer of data from the sender to the receiver [4].

Encryption is a method of data security that entails encoding the data in such a way that it can only be deciphered by those who have been granted permission to do so. Examples of data that can be encrypted include multimedia files and sensitive papers [3]. Encryption and decryption are both possible outcomes of the usage of cryptography. Image encryption refers to the process of hiding an image using an encryption algorithm in such a way that its private information is protected and it is unavailable to attackers or unauthorized users [5]. Images are encrypted for a variety of purposes, including identifying the image's source, securing copyright information, preventing piracy, and preventing individuals who shouldn't have access to them from viewing them. Image encryption allows images to be shared via e-mail, the internet and other transmission media without worrying about these images being seen by unauthorized users. According to [6], all internet-connected citizens share more than 1.8 billion images

*Corresponding Author.

per day. This shows how much sensitive data can be lost just in a day and therefore shows the importance of encrypting these images. The development of good encryption algorithms has resulted from the necessity to meet the security needs of digital images [7, 8]. Therefore, in this study, a Latin square matrix and logistics mapping cryptographic scheme for image encryption was designed, simulated and implemented.

The study contributes significantly in advancing the frontiers of cryptography through the innovative integration of chaos-based and probabilistic encryption techniques, ensuring that the enciphering and deciphering phases are error-resistant. Chaos-based encryption techniques are known for their ability to provide high levels of security due to their inherent unpredictability, while the integration of probabilistic techniques ensures that the crypto processes are not deterministic but involves randomness, thus enhancing their resilience against various attacks. One other notable feature of the proposed method is its ability to handle images of varying sizes. This scalability is crucial in practical applications where images may have different dimensions. The developed technique also achieves semantic security, meaning that even with knowledge of parts of the plain images, it is computationally infeasible to recover the key or glean meaningful information from it. In summary, this study introduces a novel chaos-based image encryption method with a probabilistic approach, addressing the unique challenges posed by image data. Statistical, computational and differential attack evaluations conducted on the developed algorithm further highlight the effectiveness of the proposed method. The developed encryption method is secure, as it has a large key space, a high level of sensitivity to both cipher keys and plain images, and no known weaknesses.

The remaining sections of this study are highlighted as follows: Section II discusses the related work by previous researchers in the area of image encryption. Section III gives the details of the methodology used in this study. Section IV gives the obtained results. Section V concludes the study while Section VI provides the recommendation.

II. Related Work

Patel & Thanikaiselvan [9] presented a new image encryption algorithm that used Latin Square and Machine Learning techniques. The algorithm first generated a chaotic sequence using neural network-based pseudo-random number generator. This sequence was then used to create encryption key images, which were XORed with the input image to produce the encrypted image. The proposed algorithm was iterated a finite number of times to generate a cipher image population. A genetic algorithm was then used to optimize the population and find the least correlated cipher image. The model was resistant to communication channel attacks, such as noise addition, cropping, and JPEG compression.

Wang et al. [10] proposed a new image encryption algorithm that used Latin square matrices. The algorithm first generated a chaotic sequence using a Lorenz system. This sequence was then used to create a Latin square matrix, which was used to permute the pixels of the input image. The permuted was then diffused using a logistic map. The simulation results from the proposed image encryption algorithm showed that it outperformed many existing image security algorithms. The algorithm was also resistant to communication channel attacks. A further investigation of this study is recommended, as it showed promises in protecting sensitive data in several applications.

Zhang et al. [11] introduced and implemented a Latin Square and random-shift based chaotic image encryption scheme. In some contexts, it was also referred to as the LSRS algorithm. The LSRS algorithm made use of a structure that consisted of pixel scrambling, replacement, and bit scrambling. The generation of Latin squares during the encryption process was correlated to the chaotic sequence, and this generation contributed to an increase in the system's overall level of security. In this line of study, the difficulty of decoding the method increased as a result of the fact that each encrypted image corresponds to a Latin square lookup table. The results of the simulation validated the LSRS algorithm's reliability as well as its efficiency.

Xu et al., [12], presented an algorithm that made use of self-orthogonal Latin Squares as its basis. This algorithm was also referred to as SOLS. In one cycle of encryption, the algorithm used the "permutation-substitution permutation" mode and the entire encryption procedure was carried out by a single SOLS. This research demonstrated that the substitution operation could be protected from differential attack by using permutation procedures at both the top and bottom levels during a single round of encryption. Therefore, at the bottom level, both substitution and permutation operations contributed to the diffusion effect. The results of the complexity analysis and simulation in the study demonstrates that the newly created algorithm was both secure and efficient, which demonstrates that it is suitable for use in real-time applications.

Zhang & Chen [13] developed a technique of encryption based on Henon chaotic maps. This encryption algorithm employed a two-phase encryption strategy. In the first step, the original image was fused with a key image, and the process was repeated in the second step. The authors asserted that their encryption approach was suitable and more resistant to brute force attacks than other similar methods since they employed both the key-image and the plain image back.

III. Methodology

This study provides a new chaos-based method and extends the application of the probabilistic approach to the domain of symmetric key image encryption. It has been carefully developed to make certain that the cipher text is unpredictable. This method of image encryption made use of two different approaches, namely permutation and substitution, to encrypt square images. Permutation is referred to as the process of shifting around the locations of an image's pixels and substitution as a method for modifying the values of the pixels that are adjacent to one another. This approach addresses the issue of encrypting images with voluminous data because it is probabilistic and can accommodate images of any size. It generates a key stream using randomness, which is subsequently used in various phases to carry out operations of permutation and substitution thereby achieving semantic security. The processes of permutation and substitution was carried out without the need to wrongly misplace the pixel positions in the image, making it an effective method for image encryption.

The encryption process which includes a fusion of the Latin Square Encryption (involving random key generation, generation of Latin Squares, Latin Square whitening, Latin Square permutation) and Logistic map Encryption. This process is represented algorithmically in Algorithm 1, while Fig. 1 depicts the framework of the proposed hybrid model.

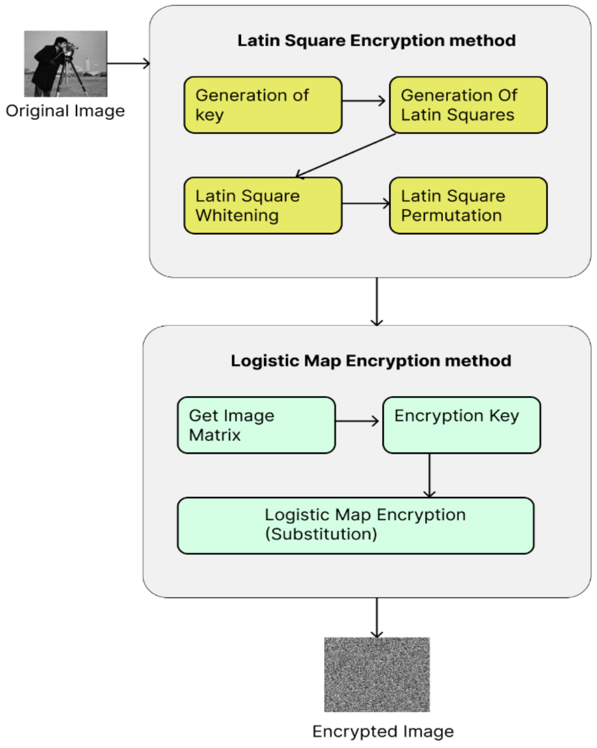

Original Image
Latin Square Encryption method
Generation of key
Generation Of Latin Squares
Latin Square Whitening
Latin Square Permutation
Logistic Map Encryption method
Get Image Matrix
Encryption Key
Logistic Map Encryption (Substitution)
Encrypted Image

Fig. 1. Proposed fused latin square - logistic map encryption model.

Algorithm 1: Fused Latin Square - Logistic map Encryption

Input: p: Original image
Output: C: ciphered image

```
1.  Import image and essential libraries;
2.  Define $\boldsymbol{RandomKey}()$ to generate a 256-bit key;
3.  Using predefined $\boldsymbol{KeyedLatin}()$ produce $\boldsymbol{n}$ Latin squares of order
    $\boldsymbol{d}$ dependent on the random key $\boldsymbol{K}$;
4.  Define a $\boldsymbol{lsq\_whitening}()$ and use it to perform Latin square
    whitening;
5.  Use the predefined $\boldsymbol{lsq\_permutation}()$ to randomly rearrange the
    image's pixel values;
6.  Define $\boldsymbol{bitget}(\boldsymbol{x},\boldsymbol{y})$ to obtain the bits from each row and column
    and perform $\boldsymbol{XOR}$ operations on the obtained bits $\boldsymbol{B}$ and previous
    matrix $\boldsymbol{M}$;
7.  Set $\boldsymbol{colourImage} = \boldsymbol{o}$ and $\boldsymbol{grayscaleImage} = \boldsymbol{1}$;
8.  Define $\boldsymbol{getImageMatrix}(\boldsymbol{imageName})$ for colour images and
    $\boldsymbol{getImageMatrix\_gray}(\boldsymbol{imageName})$ for grayscale images to
    obtain the image matrix.
9.  Encrypted image for the first time with the hexadecimal key
    sequence K.
10. Conduct substitution on the image that has already been encrypted;
11. Return Cipher Image C
```

A. Random Key Generation

Algorithm 2 presents the method for generating random key using the $dec2hex$ function.

Algorithm 2: Random Key Generation Algorithm

Input: image matrix
Output: k: a 256-bit random key in HEX

```
1. Define a function $\boldsymbol{dec2hex}()$;
2. Parse $\boldsymbol{dec2hex}()$ using the string and length as parameters;
3. Convert decimal string to a hexadecimal string $\boldsymbol{hex\_string}$;
4. Define function $\boldsymbol{RandomKey}()$;
5. Return key k.
```

B. Permutation Phase

1) Generating latin square: A Latin square of order n is a square matrix with dimensions $n * n$, whose entries consist of n symbols ordered so that each symbol appears exactly once in each row and column. An n Latin square L of order d that is dependent on the key K is generated in this phase of the process. The mathematical modelling of a Latin square L of order d that can be derived using a tri-tuple function FL of (r, c, i) is presented as follows:

$$F_L(r,c,i) = \begin{cases} 1 & si\ L(r,c,i) = S_i \\ 0 & otherwise \end{cases} \quad (1)$$

Where r denotes the index of a line of an element of L; $l \epsilon N = 0; 1; \ldots; N-1$, c denotes the index of a column of an element of $L\ c \epsilon N$, i denotes the index of a symbol element in L and Si is ith symbol in the whole $= \{S0, S1, \ldots, SN-1\}$. This implies that each symbol appears exactly once in each row and each column of L.

2) Latin square whitening: The term "Latin square Whitening" refers to an operation that combines the plaintext "P" with a pseudo random sequence. With the aid of the $LatinSquare\_Whitening$ function, a n Latin Square L of order d that is determined by the key K is produced. An example is the XOR operation. When it comes to image encryption, a plaintext message is represented by an image block denoted by the letter P and made up of a certain number of pixels. Multiple binary bits (a byte) are used to represent each individual pixel. Since the goal of key whitening is to combine plaintext information with encryption keys, we describe it as an encrypted text that operates over a finite field $GF\,(2^8)$ and is applied to the image data. A computational illustration of this is provided below:

$$y = [x+1]_{2^8} \quad (2)$$

where x represents a byte in the plaintext, i is the byte's appropriate position in the keyed Latin square, and y represents the outcome of the whitening. The computations performed over $GF\,(2^8)$ are denoted by $[.]2^8$. The whitening effect caused by the procedure described above can be easily undone by using;

$$x = [y+1]_{2^8} \quad (3)$$

Plaintext byte x represents a pixel in image encryption; for example, it might be identified at the intersection of the rth row and the cth column (i.e., $x = P(r, c)$.)

Now let $l = L(r, c)$ be an element situated at the relevant position in the keyed Latin square L, and let y be the ciphertext byte with $y = C(r, c)$, and then we will get the pixel-level equation below which is consistently utilized in the process of key whitening;

$$\begin{cases} C_{(r,c)} = \ [SR\ (P(r,c)\ ,[D]_3) + L(r,c)]2^8 \\ P_{(r,c)} = SR\ ([C(r,c)\ +\ L(r,c)]2^8, [D]_3) \end{cases} \tag{4}$$

where the rotating parameter is $D = L(0,0)$, symbol n indicates the existing round number $(n[0,7])$, and SR represents the spatial rotating function (X, d), which rotates an image X in accordance with various values of the direction d as defined below;

$$y = SR(X,d) \begin{cases} x, \ \ if\ d = 0 \\ flip\ X\ up\ \rightarrow down\ if\ d = 1 \\ flip\ X\ left\ \rightarrow right\ if\ d = 2 \end{cases} \tag{5a}$$

Observe that the following identity is always true if $Y = SR\ (X, d)$:

$$\mathrm{X\ =\ SR(Y,d)} \tag{5b}$$

3) Latin square permutation: At this phase, the Latin square is utilized to permute the image's pixels. If we take both the input and output x and y in FRM (Forward Row Mapping) and IRM (Inverse Row Mapping) to be indices, then a FRM defines a mapping from $[0,1,2,\ldots,255]$ to $[0,1,2,\ \ldots,\ 255]$, and an IRM defines the matching inverse mapping to that FRM mapping. Consequently, the Latin square P-box row, also known as the PLCL, can be defined as follows:

$$PLCL = \begin{cases} C_{(x,y_a)} = P\ (x, FRM\ (L, x, y_a) \\ P_{(x,y_b)} = C\ (y, IRM\ (L, x, y_b) \end{cases} \tag{6}$$

y_a and y_b represent the column indices before and after the mapping, respectively, in this equation. In a comparable way, the P-column Latin square box, often known as the PCCL, can alternatively be calculated with the aid of the following equations:

$$PCCL = \begin{cases} C_{(x_a,y)} = P\ (x, FRM\ (L, x_a y), y) \\ P_{(\ x_b,y)} = C\ (ICM\ (L, x_b, y), y) \end{cases} \tag{7}$$

In this operation, the row indices both before and after the mapping are shown by the notation x_a and x_b, respectively. Using a cascading method for the row permutations PLCLs, in addition to the column permutations known as PCCLs in the following manner allows us to construct our Latin square permutation with the highest level of performance possible:

$$\begin{cases} C_{(x,y)} = C^*\ (x, FCM\ (L, x, y), y) \\ P_{(\ x_a,y)} = P(\ x, FRM\ (L, x, y)) \end{cases} \tag{8}$$

In a broad sense, the function of the Latin square permutation can be expressed in the following manner:

$$PCL = \begin{cases} C = Ecr_P(L, P) \\ P = Dcr_P(L, P) \end{cases} \tag{9}$$

4) Substitution using logistics maps: The logistic map examines discrete time steps using a nonlinear difference equation. It is named the logistic map because it translates the value of the population at each given time step to its value at the subsequent time step.

The utilization of key mixing is included in this implementation. Following the completion of each pixel encryption, the initial values of the chaos map are recalculated with the addition of the key value as well as the value of the previous encryption. Logistics map is defined as follows;

$$\begin{aligned} x_{i+1} &= \ \alpha x_i\ (1 - x_i) + \ \beta y_i^2\ x_i + \ \gamma z_i^3 \\ y_{i+1} &= \ \alpha y_i\ (1 - y_i) + \ \beta z_i^2\ y_i + \ \gamma x_i^3 \\ z_{i+1} &= \ \alpha z_i\ (1 - z_i) + \ \beta x_i^2\ z_i + \ \gamma y_i^3 \end{aligned} \tag{10}$$

$x_i, y_i and z_i$ represent system variables, α and γ are parameters, while i shows iterations. For $3.57 < \mu \leq 4$, the map turns chaotic and for $\mu = 4$, the chaotic values produce in the full range of 0– 1. The flowchart for decryption is shown in Fig. 2.

C. Decryption

The decryption process converts the enciphered image back to plain image via the following algorithmic process.

1) Load the enciphered image;
2) Define $LogisticDecryption\ (imageName, key)$;
3) Generate Key-dependent $256x256$ Latin Squares L;
4) Extract a keyed Latin square;
5) Perform Latin square permutation using $CP = lsq\_permutation(input, L, decryption)$;
6) Perform Latin square whitening utilizing $CW = lsq\_whitening(input, L, decryption)$;
7) Return P;

Fig. 2 presents the flowchart of the decryption process.

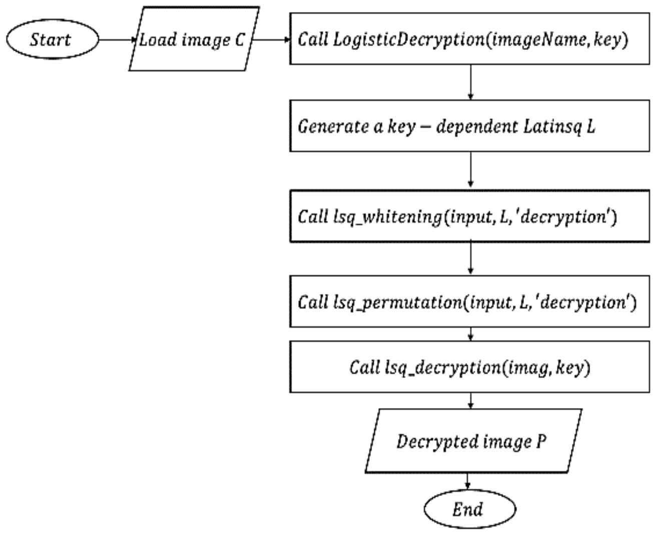

Start
Load image C
Call LogisticDecryption(imageName, key)
Generate a key – dependent Latinsq L
Call lsq_whitening(input, L, 'decryption')
Call lsq_permutation(input, L, 'decryption')
Call lsq_decryption(imag, key)
Decrypted image P
End

Fig. 2. Flow chart representation of the decryption process.

IV. Results and Discussions

A. Simulation Results

The fused encryption algorithm was experimented on seven (7) image dataset (Baboon, Lena, Peppers, Man, Water Lilies, Airplane and Fruits) to demonstrate the performance of the technique in terms of complexity and security. The experimental results are presented as follows:

1) Histogram analysis: The act of making ciphertext more widely available is one that bears significant importance. To be more specific, it should conceal the excessive sections of the plain image and should not reveal any information about the image or the relationship between the image and the enciphered image. Additionally, it should conceal its redundant elements [14]. Algorithm 3 depicts the process of finding the image histogram.

Algorithm 3: Histogram Algorithm

Input: Image,
Output: Histogram graph.

```
1. Import cv2 and from matplotlib import pyplot as plt;
2. set $img = cv2.imread('/image location path', 0)$ to read the images from their location path;
3. set $histr = cv2.calcHist([img], [0], None, [256], [0,256])$ to find the frequency of pixels in range $0 - 255$;
4. Display the plotting graph of an image using $plt.plot(histr)$ and $plt.show()$;
```

Fig. 3 and Fig. 4 contain the histograms of both the plain image and the encrypted form of those images respectively that were produced by the proposed approach. Both sets of histograms were generated by the proposed scheme. It is clear from the fact that the histograms of the cipher-images are relatively uniform, and they are notably different from that of the plain image. As a result, they do not present any signals that may be utilized to launch a statistical attack on the cipher.

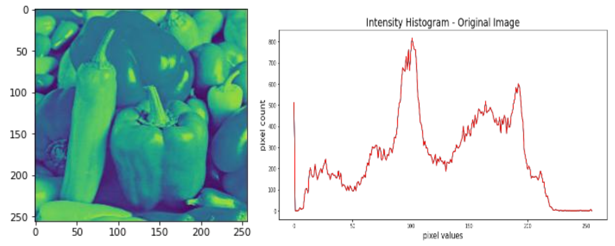


Fig. 3. Plain image and it's histogram.

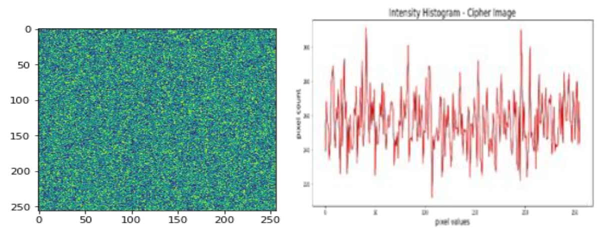


Fig. 4. Enciphered Fig. 3 and it's histogram.

2) Key space analysis: The larger the key space, the greater the ability to fight against an attack using brute force, because the process of decryption is strenuous for the attacker, and the sophisticated nature of the information makes it impossible to retrieve [15]. However, if the key that is being encrypted is relatively simple, even the most robust encryption method can be broken using exhaustive search attack. This is not the case if the key is long [16]. The initial secret keys in the proposed approach were set to have a length of 256-bits. Because of this, the key space was 2256, which is equivalent to 1.17 × 1077, making it sufficiently enormous to withstand any type of brute-force attack.

3) An evaluation of the correlation between adjacent pixels: A term referred to as an image's "intrinsic feature" describes the strong correlation that exists between the image's individual pixels. As a result, to increase resistance against statistical analysis, a secure encryption technique should remove it entirely. Within the scope of this work, visual autocorrelation analysis was conducted. Algorithm 4. depicts the process of plotting an auto-correlation pixel effect graph;

Algorithm 4: Auto-correlation pixel effect Algorithm

Input: Image,
Output: Histogram graph.

```
1. Import cv2, from matplotlib import pyplot as plt and pandas as pd;
2. Declare data = pd.read_csv("daily-minimum-temperatures-in-blr.csv",header=0, index_col=0, parse_dates=True, squeeze=True) to read the data from the csv;
3. Display top 100 data using data.head(100);
4. Display the plotting graph of an image using pd.plotting.lag_plot(data, lag=1);
```

Fig. 5 and Fig. 6 presents the autocorrelation results of both an original image and its enciphered version signposting the algorithms resilience to statistical analysis.

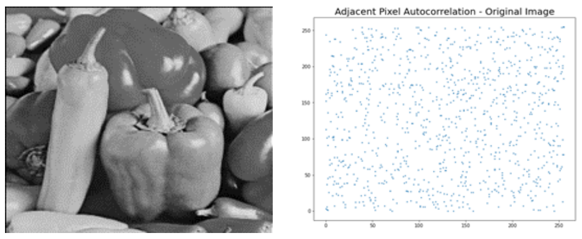


Fig. 5. Original image and its auto-correlation result.

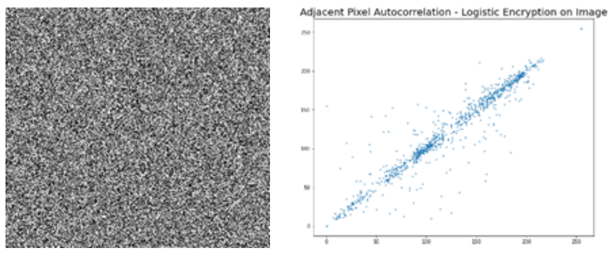


Fig. 6. Cipher image and its auto-correlation result.

4) Execution time: The execution time (ET) of the fused algorithm was computed programmatically using the following formula:

$$ET = run\ time + compile\ time \tag{11}$$

The average execution time result for this algorithm was calculated to be $184\mu s$ demonstrating the algorithms efficiency for real time applications. Table I presents the ET for seven images.

TABLE I. EXECUTION TIME OF ALGORITHM RESULTS

Image	ET(μs)
Baboon	233
Lena	210
Peppers	190
Man	120
Water Lilies	155
Airplane	146
Fruits	245
Average ET	**184**

5) Mean Square Error (MSE): The MSE value is the average difference between the pixels throughout the entire image. It is utilized to determine how accurate the pixel value is, and the error value is just the difference between the two. A larger value for MSE denotes a greater degree of dissimilarity between the processed image and the original image. In spite of this, it is important to apply appropriate precautions when working with the edges. The mean square error (MSE) can be computed using the formula that is provided in the following equation.

$$MSE = \frac{1}{AB} \sum_{M=1}^{M=A} \sum_{N=1}^{N=B} (O(m,n) - R(m,n))^2 \tag{12}$$

In this case, AB refers to the size of the image, O refers to the image before it was processed, and R refers to the image after it was processed.

The MSE was calculated for the seven images in this study, and the calculation returns an average result of 0.0, which indicates that there was no error done on the edges of the images while they were being encrypted.

6) Peak Signal Noise Ratio (PSNR): When comparing the squared error of the original image and the modified version, the Peak Signal to Noise Ratio (PSNR) and the Mean Square Error (MSE) are useful metrics to use. There is a connection that works reversely between PSNR and MSE. Therefore, a greater PSNR number suggests that the image has a higher quality (better). The ratio of the PSNR values of the decrypted and original versions is used to measure the image's quality. In this study, the PSNR value for the seven images experimented reached infinity, indicating a superior image quality. It has been determined, based on measurements, to be represented as:

$$PSNR = 10\log_{10}\frac{(2^n - 1)^2}{MSE} \tag{13}$$

Algorithm 5 shows the step-by-step process of calculating PSNR

Algorithm 5: PSNR Algorithm

Input: Original image, decrypted image,
Output: PSNR value.

```
1. Import cv2 and math, from skimage import metrics;
2. Set $img = cv2.imread('image location', 1)$ $dec\_img = cv2.imread('image location', 1)$ to read the images from their location path;
3. Assign $img = dec\_img$;
4. Set $psnr\_skimg = metrics.peak\_signal\_noise\_ratio(dec\_img, img, data\_range = None)$
5. Print ('$PSNR$ = ', $psnr\_skimg$)
```

7) Root Mean Square Error (RMSE): The RMSE value approximates the MSE value that provides accurate and reliable results [17]. Root mean square error (RMSE) is used to measure the difference between the original image and the segmented image [18]. It can be expressed in mathematical terms as;

$$RMSE = \sqrt{\frac{\sum_{i=1}^{A}\sum_{j=1}^{B}[Or(l,m) - De(l,m)]^2}{AB}} \tag{14}$$

where the values of the coordinates are denoted by the letters i and m and the size of the array is $A \times B$. Both the original and the decrypted versions of the image are indicated by the notation Or and De, respectively [19]. The interval $[0, \infty]$ denotes the range of the RMSE. The smaller the value of Root Mean Square Error (RMSE), the more effective the segmentation. In this work, RMSE as calculated on the image datasets resulted to 0.0, which indicates that there is effective image segmentation while the images were being encrypted.

8) Structural Similarity Index (SSIM): The SSIM demonstrates the correspondence between the decrypted and original image. This number is an evaluation and estimate of the image's quality that was produced from several different areas of the image that were the same size. SSIM is represented mathematically in Eq. (15).

$$SSIM = \frac{(2\mu_1\mu_{De} + CI_1)(2\partial_{1De} + C1_2)}{(\mu_1^2 + \mu_{De}^2 + CI_1)(\partial_1^2 + \partial_{De}^2 + CI_2)} \tag{15}$$

Here, μ_1 symbolizes the average of the inputs (I), while μ_{De} represents the images after they have been decrypted (De). The standard deviations of the I and the De are, respectively, ∂_1^2 and ∂_{De}^2. "∂_{1De}" stands for the covariance of the values I and De, while "CI_1" and "CI_2" stand for the regularization using the values $(0.01P)^2$ and $(0.01P)^2$, respectively. The results of the SSIM can range from 0 to 1, with 1 indicating an excellent match between the original image and the image that has been modified. Moreover, good encryption algorithms should have SSIM values that fall anywhere between 0.97 and 1. In this study, the average SSIM value is 1 indicating that there is an exact match between the two images.

9) Relative Average Spectral Error (RASE): RASE is a method that is used to calculate the overall performance of image fusion algorithms for each spectral band. The following is the formula that is used to calculate RASE:

$$RASE = \frac{100}{A}\sqrt{\frac{1}{L}\sum_{i=1}^{L} RMSE^2(B_i)} \qquad (16)$$

x is the total exposure value of the L band (Bi) in the original multispectral image, whereas RMSE is the minimal square error that evaluates the effectiveness of each band in the merged image. The perfect value would be 0. In this study, RASE was calculated to be 0, which indicates that the methodology used was effective.

10)Relative dimensionless global error in synthesis (ERGAS): ERGAS is a metric that determines the overall quality of the merged image. The inaccuracy when the quality of anything improves, shows that there is a substantial tendency for ERGAS to decrease. As a result, it is an effective measure of the quality. Cases that are considered to be of "high quality" have error ERGAS values that are equal to or lower than 3, whereas cases that are considered to be of "poor quality" have error ERGAS values that are greater than 3. 0 is the optimum value for it. The average ERGAS of this work was computed to be 0.057143. Table II shows results of the analysis. This demonstrates that the algorithm is efficient. It is denoted mathematically as:

$$ERGAS = 100\,\frac{he}{l}\sqrt{\frac{1}{L}\sum_{i=1}^{n}\frac{RMSE^2(A_i)}{y_i}} \qquad (17)$$

TABLE II. ERGAS RESULTS

Image	ERGAS
Baboon	0.00
Lena	0.10
Peppers	0.20
Man	0.50
Water Lilies	1.00
Airplane	0.00
Fruits	0.00
Avg ERGAS	0.057143

11)Information entropy analysis: The entropy of the information is a measure of how random the information is, and it may be computed as follows [19]:

$$H(m) = \sum_{i=0}^{255} -Pl(m_i)\log_2 Pl(m_j) \qquad (18)$$

Each gray level in an image with 256 gray levels contains 8 bits of data associated with it [20]. When the probability of each gray level is the same, the encrypted image is able to obtain the ideal entropy of 8, which indicates that each gray level of the encrypted image is evenly distributed. The fused encryption's entropy analysis on the enciphered images showed that the average information entropy was 7.53302, which is relatively close to the number 8. Hence, cipher images have a stronger random distribution, and the risk of information disclosure is completely eliminated. Table III depicts the results of evaluation.

TABLE III. AVERAGE OF INFORMATION ENTROPY RESULTS

Image	Entropy
Baboon	7.8693
Lena	7.9976
Peppers	7.9943
Man	7.8997
Water Lilies	7.9907
Airplane	7.993
Fruits	7.8330
Avg entropy	7.9397

*12)*Differential attack analysis: In most cases, attackers will begin by making a few alterations to the plain image. Next, they will encrypt both the plain image and the modified plain image using the same encryption algorithm. Finally, they will compare the two cipher images in order to gain further insight into the connection between the plain image and the cipher image. The number of pixels change rate (NPCR) and the unified average change intensity (UACI) are calculated to enhance the anti-differential attack performance of the encryption algorithm. In order to test the effect of slightly changing the plain image on the corresponding cipher image, the equations for NPCR and UACI are as follows:

$$NPCR = \frac{\sum_{i=1}^{A}\sum_{j=1}^{B} D(i,j)}{A \times B} \times 100\% \qquad (19)$$

$$UACI = \frac{\sum_{i=1}^{A}\sum_{j=1}^{B} |C_1(i,j) - C_2(i,j)|}{A \times B} \times 100 \qquad (20)$$

If the height and breadth of the cipher image are denoted by A and B and C1, C2 are two cipher images, and there is a difference of one bit between each answering plain image and each cipher image. The values of NPCR and UACI that should be anticipated for an image are the following: NPCR = 99.6094% and UACI = 33.4094%, respectively.

During the simulation, we selected a pixel at random for each of the image data in order to alter the value of that pixel. After that, we encrypted the images both before and after the change in order to obtain two cipher images, then we computed the NPCR and UACI. It was deduced, based on the simulation result, that the NPCR of the encrypted image was 99.60983 % and that the UACI was 33.4235 %, both of which are extremely similar to the value that was anticipated. Hence, the encryption technique is sensitive to alterations and has the ability to withstand differential attack. Table IV indicates result of the analysis.

TABLE IV. RESULTS OF NPCR AND UACI ANALYSIS

Image	NPCR	UACI
Baboon	99.6087	33.5866
Lena	99.6047	33.4082
Peppers	99.6076	33.2532
Man	99.6087	33.8456
Water Lilies	99.6067	33.2659
Airplane	99.6077	33.2418
Fruits	99.6083	33.4235
Avg entropy	99.6074	33.43211

13)Avalanche effect: To examine key sensitivity and demonstrate the robustness of the suggested scheme against differential cryptanalysis, evaluations on avalanche properties were conducted. In [20], it was stated that the avalanche effect of the technique should always be $\geq 50\%$. As part of this study, the avalanche effect of encryption algorithm was evaluated. The pattern of the number of bits modified in cipher with a single bit change in the secret key revealed that, regardless of the position of the key bit altered, the average change in the number of bits in the cipher text was 49.98235%

14)which is roughly 50%. It shows in Table V that the proposed method has high key sensitivity, high confusion, and consistent and significant contributions from all key bits to the cipher bits. Algorithm 6 depicts the step-by-step process of this security analysis. Avalanche effect is calculated with the formula below:

$$Avalanche\ Effect\ \frac{No.of\ altered\ bits\ in\ the\ ciphertext}{No.of\ bits\ in\ the\ ciphertext} \quad (21)$$

Algorithm 6: Avalanche Effect Algorithm

Input: Original image,
Output: Avalanche value.

```
1. Def x0 as first cipher;
2. Def x1 as second cipher after 1 bit change;
3. Print bitwise xor operation;
4. Count 1s in binary;
5. Evaluate equation of avalanche effect;
6. Divide result in step 5 by the longest binary string.
```

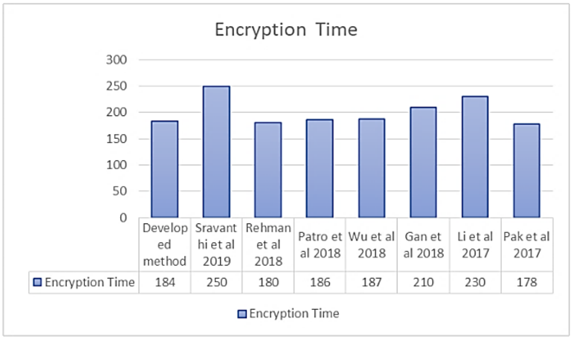
Table VI presents a comparison of this study with other state-of-the-art techniques, based on multiple evaluation metrics, while Fig. VII presents a comparative analysis of our techniques' encryption time with other authors. The comparison showed that our technique efficient and competes very favourably against other state-of-the-art techniques.

The plot displays the encryption time of the proposed method at 184(μs) and plotted against various authors in the related work. The plot shows that the developed model achieves second best encryption time when compared to other state-of-the-art results. Fig. 8 shows the plot of average avalanche effect obtained from the experiment.

TABLE V. AVALANCHE EFFECT RESULTS FOR EACH BIT CHANGED IN THE SECRET KEY

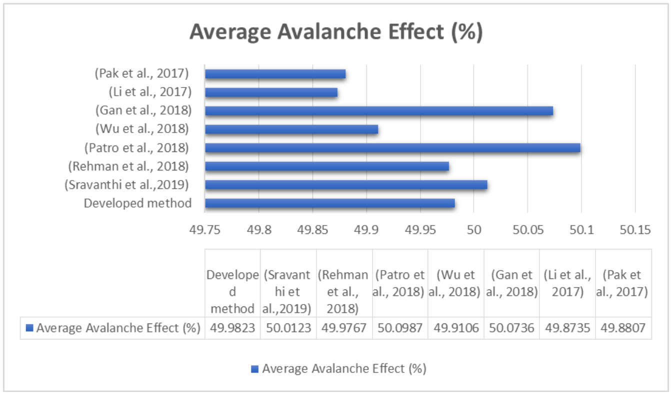
Image	paswd (%)	pstd (%)	qwd (%)
Baboon	50.0208	49.9652	49.9350
Lena	50.0101	49.9916	49.9764
Peppers	50.0258	50.0204	49.9737
Man	49.7809	50.0052	49.9801
fruits	50.0600	50.0015	49.9448
Water lilies	49.9740	50.0458	49.9770
Airplane	49.9862	49.9702	49.9845
Average	49.97969	49.99999	49.9673

Average Avalanche effect = 49.98235%



	Developed method	Sravanthi et al 2019	Rehman et al 2018	Patro et al 2018	Wu et al 2018	Gan et al 2018	Li et al 2017	Pak et al 2017
▢ Encryption Time	184	250	180	186	187	210	230	178

Fig. 7. Plot of encryption time and various authors result.



	Developed method	(Sravanthi et al.,2019)	(Rehman et al., 2018)	(Patro et al., 2018)	(Wu et al., 2018)	(Gan et al., 2018)	(Li et al., 2017)	(Pak et al., 2017)
■ Average Avalanche Effect (%)	49.9823	50.0123	49.9767	50.0987	49.9106	50.0736	49.8735	49.8807

Fig. 8. Plot of average avalanche effect obtained from the experiment.

TABLE VI. COMPARISON OF SIMULATION RESULTS WITH RELATED SCHEMES

Author(s)/Reference	Average Entropy	Key Space	Average NPCR (%)	Average UACI (%)	Average MSE	Average RMSE	Average SSIM	Execution Time (μs)	Average Avalanche Effect (%)
Developed method	7.9398	1.17×10^{77}	99.6074	33.4321	1.00	0.0	1.0	184	49.9823
(Sravanthi et al.,2019)	7.9993	1.1×2^{377}	99.6098	33.4707	0.97	0.0	0.3	250	50.0123
(Rehman et al., 2018)	7.6635	10^{94}	99.5999	33.3848	1.00	0.0	0.0	180	49.9767
(Patro et al., 2018)	7.9998	1.9×2^{426}	99.6028	33.4021	0.99	0.0	0.1	186	50.0987
(Wu et al., 2018)	7.9196	10^{88}	99.6090	33.4227	1.00	0.0	0.2	187	49.9106
(Gan et al., 2018)	7.9993	2470	99.6000	33.4400	0.96	0.0	0.1	210	50.0736
(Li et al., 2017)	7.9272	2273	99.6100	33.4600	0.95	0.0	1.0	230	49.8735
(Pak et al., 2017)	-	2138	99.6236	33.3441	0.99	0.0	1.0	178	49.8807

The avalanche effect is considered as one the desirable property of any encryption algorithm. The effect ensures that an attacker cannot easily predict a plain-text through a statistical analysis.

V. Conclusion

The study's objective was to develop a resilient encryption scheme to protect images from being decrypted. A strong algorithm for encrypting images should be resistant to attacks, and its efficiency should be unaffected by either the encryption key or the image that has been encrypted. It should have a large key space. This algorithm with all its component parts and stages, satisfies each one of these criteria, and it also has certain unique qualities. As this proposed algorithm is completely described in integers, it is computationally efficient in either software or hardware and does not lead to complications with finite precision or discretization. In addition, the suggested approach builds all encryption primitives based on a key generator. These encryption primitives include substitution and permutation, and because of changes, the proposed method achieves high sensitivity to any key change. The suggested technique additionally combines probabilistic encryption, which provides the conversion of a single plain image into numerous cipher images utilizing a single encryption key, and assures that the decoding phase is error-resistant up to a preset level. Statistical, computational and differential attack evaluations have been conducted on the developed algorithm. All experimental results indicated that the proposed encryption method is secure, as it has a large key space, a high level of sensitivity to both cipher keys and plain images, and no known weaknesses.

VI. Recommendation

It is strongly suggested that this method be utilized whenever images need to be encrypted. In addition, more research can be done on the algorithm to enhance its current capabilities and make it even more powerful than it now is. Additional statistical analysis can be carried out computationally as well as visually.

References

[1] Y. L. Hailan Pan, "Research On Digital Image Encryption Algorithm Based On Double Logistic Chaotic Map". Research on digital image encryption algorithm based on double logistic chaotic map, 10. 2018.

[2] M. Xu, and Z. Tian, "A Novel Image Encryption Algorithm Based on Self-Orthogonal Latin Squares". Optik, vol. 17 no. 1, pp. 891–903, 2018.

[3] A. E. Adeniyi, S. Misra, E. Daniel, and A. Bokolo Jr, "Computational complexity of modified blowfish cryptographic algorithm on video data". Algorithms, vol. 15 no. 10, pp.373. 2022.

[4] L. Abraham, and N. Daniel, "Secure Image Encryption Algorithms: A Review". INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, vol. 2. 2013

[5] A. E. Adeniyi, K. M. Abiodun, J. B. Awotunde, M. Olagunju, O. S. Ojo, and N. P. Edet, "Implementation of a block cipher algorithm for medical information security on cloud environment: using modified advanced encryption standard approach". Multimedia Tools and Applications, pp. 1-15. 2023.

[6] M. Meeker's, "KPCB Researcher Mary Meeker's Annual Internet Trends Study". KPCB researcher Mary Meeker's annual Internet Trends study. 2017.

[7] Q. R. Renzhi Li, "Novel image encryption algorithm based on improved logistic map". Novel image encryption algorithm based on improved logistic map, vol. 10, 2019.

[8] A. E. Adeniyi, A. L. Imoize, J. B. Awotunde, J. B., C. Lee, P. Falola, R. G. Jimoh, and S. A. Ajagbe, "Performance Analysis of Two Famous Cryptographic Algorithms on Mixed Data". Journal of Computer Science, vol.19, no. 6, pp. 694-706. https://doi.org/10.3844/jcssp.2023.694.70 2023.

[9] S. Patel, and V. Thanikaiselvan, "Latin Square and Machine Learning Techniques Combined Algorithm for Image Encryption". Circuits, Systems, and Signal Processing, pp. 1-25, 2023.

[10] X. Wang, Y. Su, M. Xu, H. Zhang, and Y. Zhang, "A new image encryption algorithm based on Latin square matrix". Nonlinear Dynamics, vol. 10 no. 7, pp. 1277-1293. 2022.

[11] X. Zhang, T. Wu, Y. Wang, L. Jiang, and Y. Niu, "A novel chaotic image encryption algorithm based on latin square and random shift". Computational Intelligence and Neuroscience, 2021.

[12] M. Xu, & Z. Tian, "A Novel Image Encryption Algorithm Based on Self-Orthogonal Latin Squares". Optik, vol. 17 no. 1, pp. 891–903. 2018.

[13] X. Zhang, and W. Chen, "A new chaotic algorithm for image encryption". In 2008 International conference on audio, language and image processing, IEEE, pp. 889-892. 2008.

[14] M. Jiang, P. Cui, and C. Faloutsos, "Suspicious behavior detection: Current trends and future directions". IEEE intelligent systems, vol. 31, no. 1, pp. 31-39, 2016.

[15] A. E. Omolara, A. Jantan, O. I. Abiodun, K. V. Dada, H. Arshad, and E. Emmanuel. "A deception model robust to eavesdropping over communication for social network systems". IEEE Access, vol. 7, pp. 100881-100898. 2019.

[16] F. Han, X. Liao, B. Yang, and Y. Zhang, "A hybrid scheme for self-adaptive double color-image encryption". Multimedia Tools and Applications, vol. 77, pp. 14285-14304. 2018.

[17] B. M. Lei Chen "Differential cryptanalysis of a novel image encryption algorithm based on chaos and Line map". Nonlinear Dynamics, 12, 2016.

[18] M. K. Kumar. "Colour Image Encryption Technique Using Differential Evolution In Non-Subsampled Contourlet Transform Domain". IET Image Processing, 11. 2018.

[19] M. Kaur, and V. Kumar "A Comprehensive Review on Image Encryption Techniques". Archives of Computational Methods in Engineering, vol. 27, no.1, pp. 15–43, 2020.

[20] X. W. Hao Zhang, "An Efficient and Secure Image Encryption Algorithm Based on Non- Adjacent Coupled Maps". 17. 2020