Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466

Applications and Applied Mathematics:
An International Journal
(AAM)

Vol. 15, Issue 2 (December 2020), pp. 1230 – 1244

Image Encryption using Gingerbreadman Map And RC4A Stream Cipher

[1*]Abdul Gaffar, [2]A.B. Joshi, and [3]Dhanesh Kumar

Department of Mathematics and Astronomy
University of Lucknow
U.P. 226007
India
[1]abdulgaffar.lu@gmail.com; [2]anandiitd.joshi@gmail.com; [3]dhaneshkumar.lu@gmail.com

*Corresponding Author

Abstract

Day to day increasing flow of sensitive or confidential information, such as images, audio, video, etc., over unsecured medium (like Internet) has motivated more concentration for concrete crypto algorithms. In this paper, an image encryption algorithm based on a permutation and substitution cipher has been proposed. In permutation stage, image pixels are shuffled using gingerbreadman map while in substitution stage, pixels are bit-wise XOR-ed with the keystream generated using RC4A (Rivest Cipher 4A) stream cipher algorithm. For the proposed scheme, statistical analyses, like histogram, adjacent pixels correlation coefficient, and information entropy are given. Security analyses, like key sensitivity, occlusion analysis are also given in this paper. The occlusion analysis shows that the proposed method is resistant to the occlusion attack. These statistical and security analyses support the concreteness of the proposed method.

Keywords: Gingerbreadman map; Stream cipher; RC4; RC4A stream cipher

MSC 2010 No.: 94A60, 68P25

1. Introduction

The process of exchanging data or information over a medium is known as communication. Secure communication requires protection of this exchange of (confidential) information from an

eavesdropper. Digital images flow frequently through a communication channel. Over an unsecured channel, protection of digital images is a challenging task. For this purpose (i.e., secure communication), many image encryption algorithms/methods have been proposed in recent years. Sivakumar and Venkatesan (2016) proposed an image encryption method based on knight's travel path and true random number. Joshi et al. (2020a) proposed a method in which baker chaotic map is used for shuffling rows and columns of the plain image and three-dimensional (3D) Arnold transform is utilized for modifying pixels intensities. Also, Joshi et al. (2020b) proposed an algorithm for color image encryption based on 2D discrete wavelet transform and 3D logistic chaotic map. Moreover, the works done in Mishra (2017), Mishra (2012), and Mishra (2007) give a glimpse that how the images are used as 2D signals.

Chaos theory (Pickover (1988)) is related to the chaotic behaviors of a system. In other words, a behavior is chaotic if the future is determined by the present but the approximate present does not predict the approximate future. Chaotic maps (Peitgen and Saupe (1988)) are very effective for cryptographic techniques due to their randomness effect, butterfly effect, etc.

In this paper, a chaotic gingerbreadman map (article 3.2.3 in Peitgen and Saupe (1988)) and RC4A (Paul and Preneel (2004)) (a variant of RC4 (Schneier (1996))) stream cipher have been used. RC4, a widely used stream cipher, was used as a standard cipher for TLS/SSL (Transport Layer Security/Secure Sockets Layer) connections (Dierks and Allen (1999)) until it was not prohibited by RFC 7465 (Popov (2015)) in 2015 for all versions of TLS. RC4A stream cipher algorithm, being stronger than RC4, has also been attacked by Maximov (2007) and a group of members from NEC (Tsunoo et al. (2005)). So, in order to make RC4A cipher more robust we have combined it with the gingerbreadman map (a chaotic map).

The remaining matter of the paper has been put in the following order. Section 2 is about preliminaries which includes gingerbreadman map, stream cipher, RC4, and RC4A stream cipher algorithm. Section 3 depicts flowchart of the proposed scheme, Section 4 describes the encryption and decryption algorithm for the proposed scheme. Section 5 contains experimental results. Sections 6 and 7 describe security and statistical assessment, while Section 8 gives the comparison of the proposed scheme with the recent methods, and at last, Section 9 presents the conclusion of the proposed work.

2. Preliminaries

2.1. Gingerbreadman map

In the theory of dynamical system (Peitgen and Saupe (1988)), a gingerbreadman map is a two-dimensional chaotic map. The map is chaotic for certain initial conditions and initial parameters. On plotting the set of chaotic solutions of this map, it resembles a gingerbread man (Figure 3.3 in Peitgen and Saupe (1988)). It is given by the piecewise linear transformation

$$\begin{aligned} a_{i+1} &= 1 - b_i + |a_i|,\ i \in \mathbb{N} \cup \{0\}, \\ b_{i+1} &= a_i, \end{aligned} \tag{1}$$

where a_0 and b_0 are initial parameters.

For example, the above map is chaotic for $a_0 = -0.089$ and $b_0 = 0.046$.

2.2. Stream cipher

It is a cipher which encrypts intelligible text bit by bit, simply by XOR-ing the bits with the output bits of the pseudo-random bit generator. Moreover, it is a symmetric cipher cause XOR (exclusive-OR) is a symmetric operation. Hence, the decryption follows in the same fashion as the encryption.

2.3. RC4 stream cipher

RC4 (Rivest Cipher 4), one of the most widely used stream cipher based software, was designed by Ronald Linn Rivest for RSA Data Security in 1987. It has been integrated into popular protocols such as TLS/SSL and WEP (Wired Equivalent Privacy) implementations. Until the cipher leaked out, after 7 years of its designation, it was kept as a trade secret which is now available for public analysis (Schneier (1996)). Being remarkable for its speed and simplicity, a number of weakness (Paul and Preneel (2004); Tews and Beck (2009)) have been found making it insecure (Popov (2015)). In particular, it is biased when the initial bytes are not omitted or when non-random keys are used. For RC4 algorithm, refer to Schneier (1996).

2.4. RC4A stream cipher

RC4A, a variant of RC4 cipher, was proposed by Souradyuti Paul and Bart Preneel in 2004. It uses two counters j_1 and j_2 corresponding to states S_1 and S_2 instead of one as in RC4. Two bytes are produced on incrementing i each time. Even though RC4A cipher algorithm requires same number of operations per output byte, still it has greater parallelism than RC4 resulting in possible improvement of speed. Moreover, it is more secure as most of the attacks are less effective than RC4.

2.5. RC4A algorithm

RC4A cipher runs in two phases. The first phase is the key scheduling algorithm and second one is the pseudo-random generation algorithm.

2.5.1. *Key Scheduling Algorithm (KSA)*

It is used to initialize the permutation in the states S_1 and S_2. The algorithm is divided into three parts. First part is the initialization of states S_1 and S_2, i.e., $S_1[0] = 0, S_1[1] = 1, ..., S_1[255] = 255$,

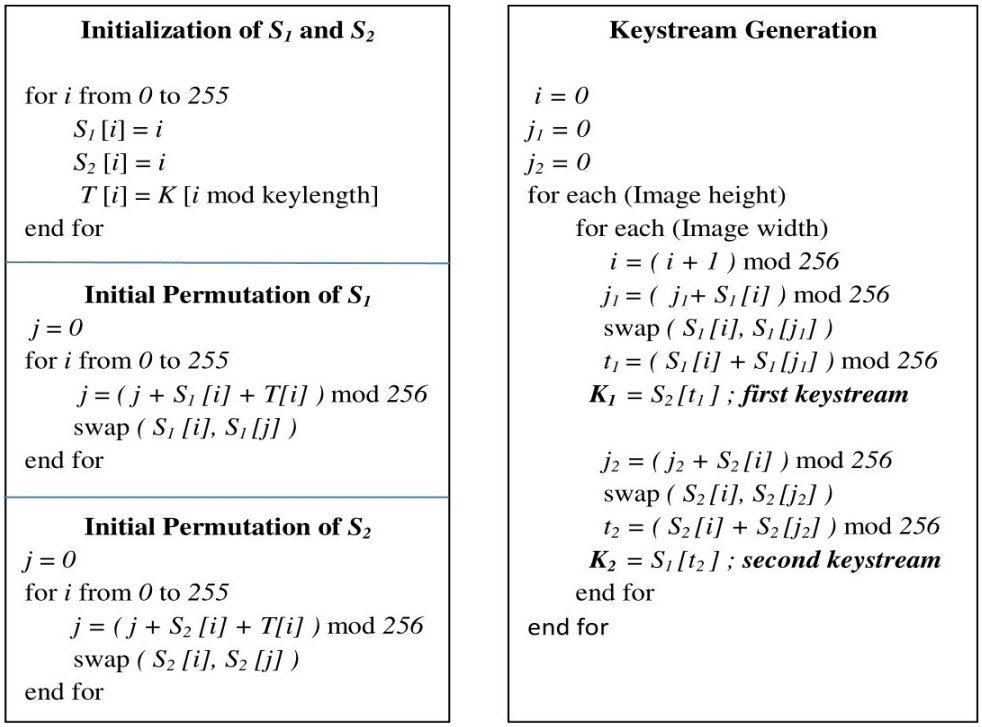


Figure 1. RC4A stream cipher algorithm

or $S_1 = [0, 1, 2, ..., 255] = S_2$ where each entry in states S_1 and S_2 denotes a *byte*. Here, K ($1 \leq K \leq 256$) denotes the secret key and the number of bytes in key (K) is denoted by *keylength*. T is the temporary vector of size 256. If keylength is 256 then T is itself K, else K is repeated as many times as to fill T. The algorithm is given in Figure 1(a).

The second part is the initial permutation of state S_1. For each index i, j is computed as: $j = (j + S_1[i] + T[i])$ mod 256 and then i^{th} and j^{th} bytes in S_1 are swapped. We will briefly explain this part.

If K = '87654321' (8 bytes) then $T = [8, 7, 6, 5, 4, 3, 2, 1, 8, 7, 6, 5, 4, 3, 2, 1, ..., 8, 7, 6, ..., 2, 1]$. Now, for $i = 0, j = (j+S_1[i]+T[i])$ mod $256 = (0+S_1[0]+T[0]) = (0+0+8) = 8$. Next, i^{th} ($i = 0$) and j^{th} ($j = 8$) bytes in S_1 are swapped to produce $S_1 = [8, 1, 2, 3, 4, 5, 6, 7, 0, 9, 10, 11, ..., 255]$. Again, compute j, for $i = 1$ and $j = 8$ (previous value) and update the state S_1. In this way, array S_1 is updated for each i up to $i = 255$ and the corresponding value of j. Similarly, in third part state S_2 is updated using the code given in Figure 1(a).

2.5.2. *Pseudo-Random Generation Algorithm (PRGA)*

It is used to produce two keystreams K_1 and K_2 using the states S_1 and S_2 obtained from the above KSA. In the pseudo-code shown in Figure 1(b), image width and image height are the number of columns and rows of an image, respectively.

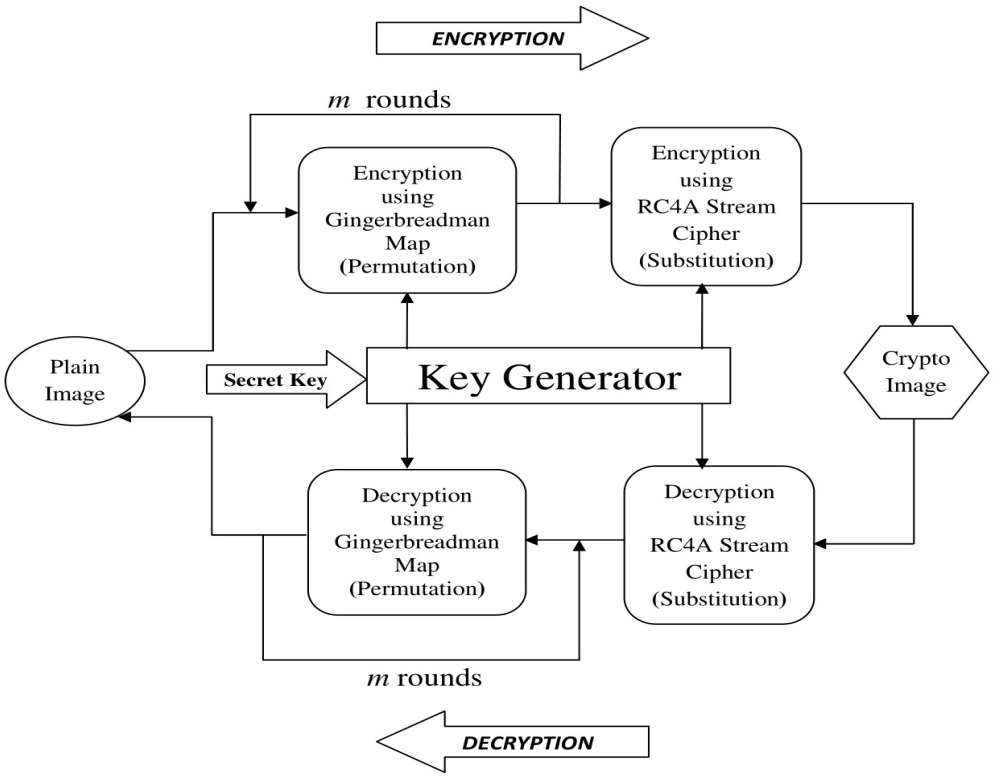


Figure 2. Flowchart of the proposed scheme

Keystream K_1 is produced in the same way as in RC4, but here it is evaluated on state array S_2, not on S_1. Keystream K_2 is produced with the help of state array S_2 and index j_2 keeping index i fixed, i.e., without incrementing i. The output $(S_2[i] + S_2[j_2])$ is evaluated on S_1 instead of S_2. The algorithm is given in Figure 1(b).

The encryption procedure of RC4 algorithm is as follows:

$$E_1 = P \oplus K_1,$$
$$E_2 = E_1 \oplus K_2,$$

where $\oplus$ is XOR operation. K_1, K_2 are the keystreams and P denotes the plaintext.

3. Flowchart of the proposed scheme

The proposed method relies on two stages, viz., permutation and substitution. In the first stage, pixels of the intelligible image are shuffled keeping their values unaltered and this stage iterates for m (≥ 10) rounds. In the second stage, the pixel intensities of the obtained intermediate crypto image are modified in a sequential order. Further, keys for the proposed method are produced using the gingerbreadman map (Equation (1)) and the RC4A stream cipher algorithm (Figure 1). The diagrammatic representation of the proposed scheme is shown in Figure 2.

4. Encryption and decryption algorithms for the proposed cryptosystem

4.1. Algorithm for encryption

Take an RGB image of size $v \times w$.

Stage 1

(1) Produce a sequence $\{a_1, a_2, ..., a_i, a_{i+1}, ...\}$ by using gingerbreadman map (Equation (1)). If $s = \max\{v, w\}$, then take s elements from the above sequence and let $\{a_1, a_2, ..., a_s\}$ be the set of s elements of the sequence.
(2) Multiply each element of the set $\{a_1, a_2, ..., a_s\}$ by 10^4 in order to make them integers, i.e., $d_i = a_i \times 10^4$, $i = 1, 2, ..., s$.
(3) Compute $h_i = d_i \pmod{s} + 1$, $i = 1, 2, ..., s$ and "mod" is the modulo operation. Also, let $M = \{h_1, h_2, ..., h_s\}$, where each $h_i \in L$, $i = 1, 2, ..., s$ and $L = \{1, 2, ..., s\}$.
(4) Construct a set P of non-repeating elements of M, i.e., $P = \{h_1, ..., h_j\}$, $j \leq s$. Next, replace those elements (if any) of the set M which appear more than once, by elements of the set $L \backslash P$. Now, set M $(= \varphi)$ contains all distinct elements.
(5) Let $\varphi = \{x_1, x_2, ..., x_s\}$ be the set of permutation of s elements. Now, there are three cases:
   a If $v > w$ then $s = v$, hence use $\varphi_1 = \{x_1, x_2, ..., x_w, ..., x_v\}$ to shuffle rows and $\{x_1, x_2, ..., x_w\}$ of φ_1 to shuffle columns of each component of RGB image.
   b If $v = w$ then $s = v = w$, hence use φ for row and column shuffling.
   c If $v < w$ then $s = w$, hence use $\varphi_2 = \{x_1, x_2, ..., x_v, ..., x_w\}$ to shuffle columns and $\{x_1, x_2, ..., x_v\}$ to shuffle rows.
(6) Iterate step (5) for m (≥ 10) rounds, m being a fixed natural number. The image obtained (after concatenation) is the intermediate enciphered image.
(7) Let I_1, I_2, and I_3 be the three components of intermediate enciphered image. Convert these image matrices I_1, I_2, and I_3 into image arrays P_1, P_2, and P_3 of size vw, respectively.

Output: Intermediate enciphered image.

Stage 2

(8) Construct two arrays S_1 and S_2 using KSA of RC4A algorithm given in Figure 1(a).
(9) Generate two keystreams K_1 and K_2 using PRGA of RC4A algorithm given in Figure 1(b) and store them in arrays U_1 and U_2, respectively. The size of each array U_1 and U_2 is vw.
(10) Final encryption is as follows:

for $i = 1$ to 3
  for $j = 1$ to vw
    $E_{1i}(j) = P_i(j) \bigoplus U_1(j)$
    $E_{2i}(j) = E_{1i}(j) \bigoplus U_2(j)$
  end for
end for

where $\bigoplus$ is bit-wise XOR operation.

(11) Convert output arrays E_{21}, E_{22}, and E_{23} into image matrices M_1, M_2, and M_3 of size $v \times w$, respectively.

(12) Image obtained after concatenating M_1, M_2, and M_3 is the final enciphered image.

Output: Finally enciphered image.

4.2. Algorithm for decryption

Let the enciphered RGB image be A of size $v \times w$ and A_1, A_2, and A_3 be its three components.

Stage 1

(1) Convert image matrices A_1, A_2, and A_3 into image arrays C_1, C_2, and C_3 of size vw, respectively.

(2) Partial decryption is as follows:

for $i = 1$ to 3
  for $j = 1$ to vw
    $D_{2i}(j) = C_i(j) \bigoplus U_2(j)$
    $D_{1i}(j) = D_{2i}(j) \bigoplus U_1(j)$
  end for
end for

(3) Convert output arrays D_{11}, D_{12}, and D_{13} into image matrices Q_1, Q_2, and Q_3 of size $v \times w$, respectively. Image obtained after concatenating Q_1, Q_2, and Q_3 is the intermediate deciphered RGB image.

Output: Intermediate deciphered image.

Stage 2

(4) Calculate φ^{-1}, where $\varphi = \{x_1, x_2, ..., x_s\}$.

(5) Permute rows and columns of each component Q_1, Q_2, and Q_3 of intermediate deciphered image using φ^{-1}.

(6) Iterate step (5) for m rounds. Image obtained after concatenating red, green, and blue components is the final decrypted image.

Output: Finally decrypted image.

5. Implementation and experimental results

The proposed scheme is implemented in *MATLAB 8.5*. We have taken a standard RGB Lena image of size 256 $\times$ 256 for the demonstration of the approach. The consequences of the experiment are as follows: Figure 3a represents an original image of Lena, Figure 3b is the partially encrypted

(a) Original image of Lena **(b)** Partially encrypted image **(c)** Encrypted image

(d) Partially decrypted image **(e)** Decrypted image

Figure 3. Experimental results

image using $m = 25$ iterations, and Figure 3c is the encrypted image of Lena. Figure 3d shows the partially deciphered image, while Figure 3e is the decrypted image.

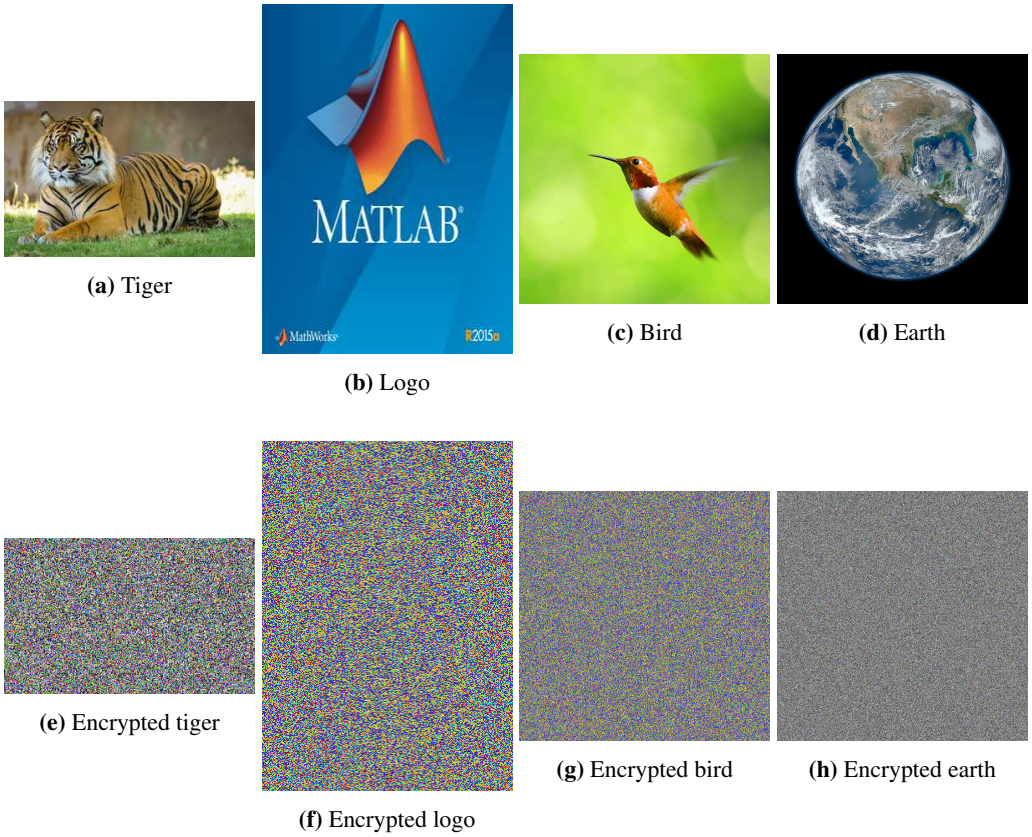
5.1. Some more experimental results

We have also taken number of color images of different size (in pixels) to empirically assess the performance of our proposed method. The results are given in Figure 4. Images taken are of the following size: Figure 4a (Tiger) is of size 300×187 pixels, Figure 4b (MATLAB logo) is of size 250×350, Figure 4c (Bird) is of size 512×512, and Figure 4d (Earth) is of size 1024×1024.

6. Security assessment

6.1. Key space analysis

The set of all possible permutations of a key is defined as a key space of the crypto algorithm. For secure encryption, key space should be very large in order to resist attacks, like brute-force, known-plaintext, chosen-ciphertext, etc. As in our proposed approach key size can be up to 256 bytes, i.e., 2048-bits producing a key space of 2^{2048}. So, the key space is very large.



(a) Tiger **(b)** Logo **(c)** Bird **(d)** Earth

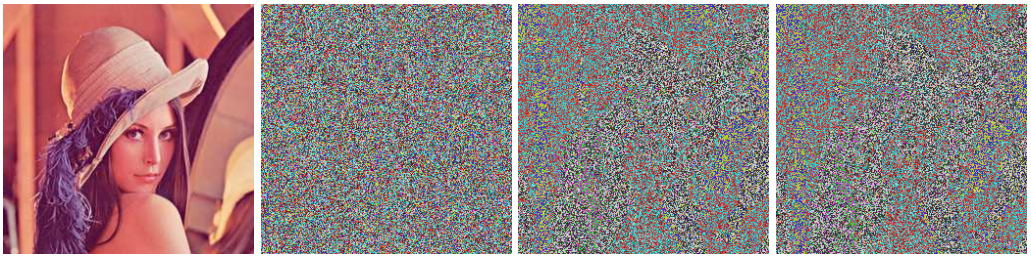
(e) Encrypted tiger **(f)** Encrypted logo **(g)** Encrypted bird **(h)** Encrypted earth

Figure 4. Images (4a–4d) are the original ones while (4e–4h) are the corresponding encrypted images

6.2. Key sensitivity analysis

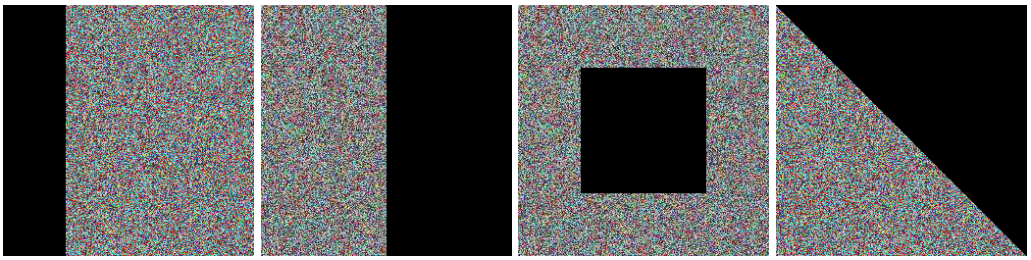
Key sensitivity is one of the important criterion for the crypto algorithm to be concrete. Moreover, high sensitivity is needed in order to secure the algorithm from intruders. We have analyzed the sensitivity test using a key that differs only by one bit instead of 1-byte as in Saptarini and Alberth (2013); Gaata and Hantoosh (2016) from the original key. Figure 5b is the enciphered image of Lena (Figure 5a) using secret key K = 'pfdw38' (in ASCII), which is equivalent to 01110000 01100110 01100100 01110111 00110011 00111000 (in binary). Figure 5c is the deciphered image of Lena using a key, say c_1, which is obtained by changing only 1-bit (Most Significant Bit) in binary representation of secret key K, while Figure 5d is also a deciphered image using key, say c_2 obtained when two bits (6^{th} and 7^{th} from MSB) are altered in binary representation of original key K.

On observing Figure 5, one concludes that the plain image can not be obtained until the exact key is used. Hence, the suggested crypto algorithm is highly sentient.



(a) Plain image of Lena **(b)** Encryption using K **(c)** Decryption using c_1 **(d)** Decryption using c_2

Figure 5. Key sensitivity analysis



(a) 25% pixels occluded **(b)** 50% pixels occluded **(c)** 50% pixels occluded **(d)** 50% pixels occluded

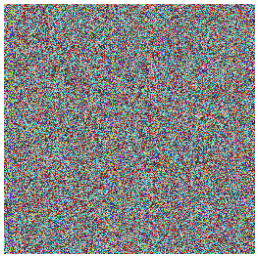
(e) Decrypted image **(f)** Decrypted image **(g)** Decrypted image **(h)** Decrypted image

Figure 6. Images (6a–6d) are the occluded ones while (6e–6h) are the corresponding decrypted images

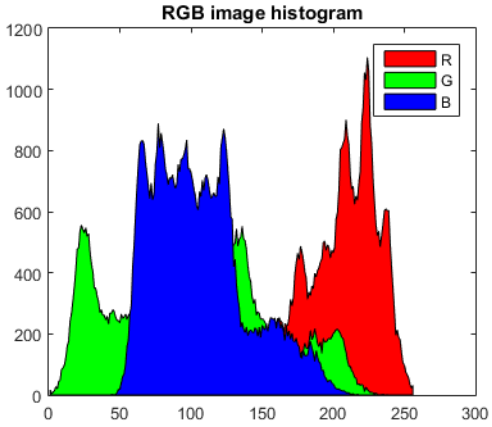
6.3. Occlusion analysis

In this analysis, the concreteness of the algorithm has been analyzed when crypto image data is occluded by 25% and 50% pixels. Here, we have taken encrypted image of Lena given in Figure 3c for experimental analysis. Figures 6a and 6b are the occluded images from left and right with 25% and 50% pixels of the crypto Lena image, while Figures 6e and 6f are the corresponding images decrypted with correct key, respectively. Figures 6c and 6d are the occluded images from middle and upper from diagonal with 50% pixels, while Figures 6g and 6h are the corresponding deciphered images with original key, respectively. Thus, the analysis performed in Figure 6 endorses that the suggested algorithm resists the occlusion attack by 25% and 50% crypto data. Hence, the algorithm is secured against such types of attacks.

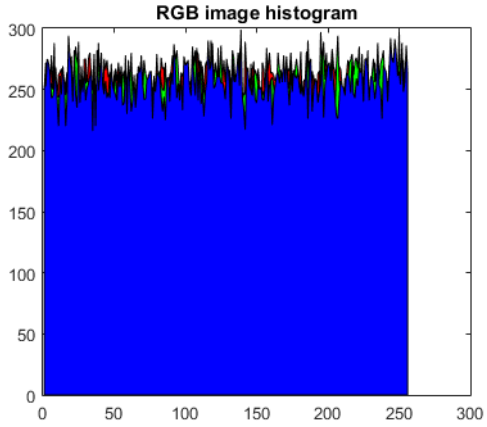
(a) Original image



(b) Encrypted image



(c) Histogram of Lena image



(d) Histogram of encrypted image

Figure 7. Histogram of original and encrypted images

7. Statistical assessment

7.1. Histogram analysis

An image histogram is a type of histogram that acts as a graphical representation of the pixel intensity (or tonal) distribution of an image. As secure crypto algorithm tends to encipher an intelligible image into an unintelligible image, so the histogram should be uniformly distributed, which is a significant factor supporting robustness of the crypto algorithm. The histogram of intelligible image (Figure 7a) is shown in Figure 7c, while that of unintelligible image (Figure 7b) is given in Figure 7d.

7.2. Adjacent Pixels Correlation Coefficient (APCC) analysis

The correlation coefficient between two adjacent pixels x and y is calculated using the formula

$$C_{xy} = \frac{Cov(x, y)}{\sigma_x \times \sigma_y}, \tag{2}$$

where σ_x denotes the standard deviation of pixel x, and $Cov(x, y)$ indicates the covariance between pixels x and y.

Table 1. Correlation analysis of Lena image

RGB Components	Plain Image			Cipher image		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
R	0.9528	0.9761	0.9285	−0.0017	−0.0103	−0.0030
G	0.9360	0.9669	0.9111	0.00006	0.0054	−0.00005
B	0.9181	0.9484	0.8892	0.0039	0.00009	0.0059

The correlation coefficient for the plain image has positive value tending towards 1 as neighboring pixels have similar range pixel values. The correlation for encrypted image has value moving away from positive 1 as neighboring pixels have dissimilar pixel values. Table 1 gives correlation coefficient of R, G, and B components of original Lena image (Figure 8a) and the encrypted image (Figure 8b) along horizontal, vertical, and diagonal directions. Graphical representation of APCC for plain image and its encrypted image along different directions is shown in Figure 8c, whence we see that the pixel intensity distributions of the encrypted image is uniformly distributed in the domain, which is completely different from pixel distributions of the plain image. So, no information can be obtained from encrypted image, hence it is secured from attacks.

7.3. Information entropy analysis

Usually, to measure the randomness of an image, say Y entropy test $H(Y)$ is used which can be computed using the Equation (3) as follows:

$$H(Y) = -\sum_{i=1}^{2^N} Pr(y_i)\log_2(Pr(y_i)), \tag{3}$$

where y_i denotes the i^{th} possible value in Y, $Pr(y_i)$ denotes the probability of y_i, and N represents number of bits used to represent a pixel. In case of a gray scale image, $N = 8$.

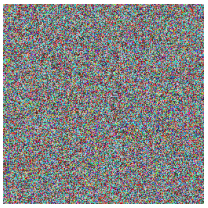
The entropy $H(Y)$ has maximum value when Y is uniformly distributed. As in a gray scale image, each pixel intensity is represented by 8-bits, so for a random image, entropy value $E(Y)$ should be closed to 8. The entropy values of plain images (Figures 3a, 4a–4d) and enciphered images (Figures 3c, 4e–4h) are given in Table 2, whence it is noticeable that the enciphered images have entropy value closed to 8 endorsing that the images become random-like.

8. Comparison of the proposed scheme

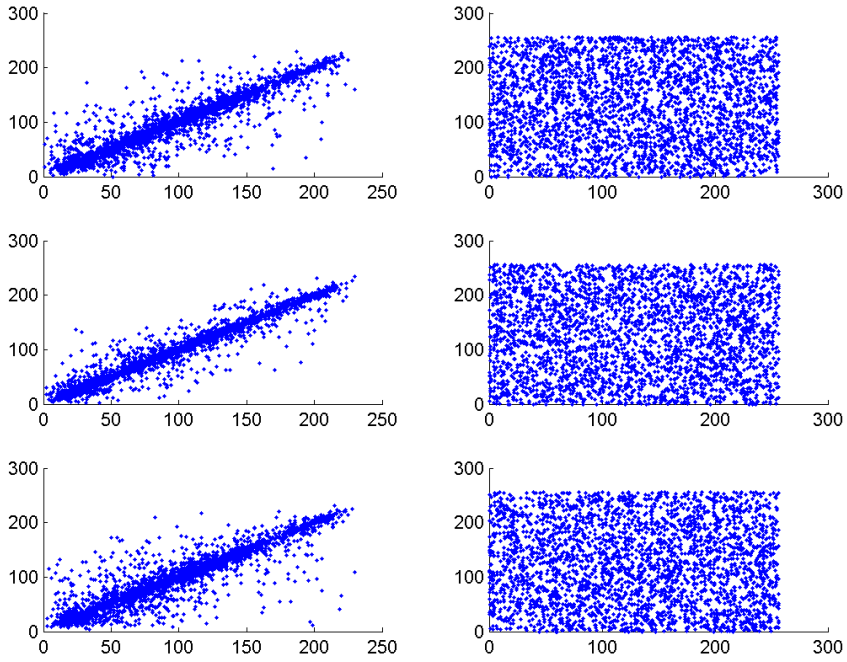
We have compared our proposed scheme with the recent encryption algorithms based on entropy values of encrypted Lena image (Figure 3c). Table 3 lists the entropy values of different algorithms.

From Table 3, we conclude that our proposed method outperforms the methods listed in the table.

(a) Plain image



(b) Encrypted image



(c) Horizontal, vertical and diagonal pixel distributions of Figure 8a and 8b respectively

Figure 8. Graphical representation of adjacent pixels along different directions

Table 2. Entropy values of plain and enciphered images

Image	Size (pixels)	Entropy Value	
		Plain Image	Enciphered Image
Lena	256×256	7.2544	7.9972
Tiger	300×187	7.4466	7.9967
Logo	250×350	4.9820	7.9979
Bird	512×512	6.7162	7.9993
Earth	1024×1024	5.4025	7.9998

9. Conclusion

In this paper, an algorithm has been proposed, which consists a combination of the gingerbreadman map and RC4A stream cipher for secure communication of RGB images. We have performed

Table 3. Entropy values of different methods

Encryption Method	Entropy Value
Proposed	7.9974
Sivakumar and Venkatesan (2016)	7.9593
Mondal et al. (2015)	7.9591
Hanchinamani and Kulkarni (2015)	7.9972
Loukhaoukha et al. (2012)	7.9968

number of tests and analyses to images of different size to empirically assess the performance of our proposed approach. We have compared our scheme with some existing algorithms based on entropy values, whence we conclude that our scheme is better than the compared schemes. Moreover, the results of security and statistical analyses as discussed in this paper indicate that our method is up to the mark and is suitable for the security of digital images.

Acknowledgment:

All authors have contributed equally in the preparation of the manuscript.

REFERENCES

Dierks, T. and Allen, C. (1999). The TLS protocol, version 1.0, Internet Engineering Task Force (IETF). Retrieved from https://www.rfc-editor.org/rfc/rfc2246

Dubey, R., Deepmal and Mishra, V.N. (2020). Higher-order symmetric duality in non differentiable multi objective fractional programming problem over cone constraints, Stat., Optim. Inf. Comput., Vol. 8, pp. 187–205. https://doi.org/10.19139/soic-2310-5070-601

Gaata, M. and Hantoosh, F. (2016). An efficient image encryption technique based on chaotic logistic map and RC4 stream cipher, Int. J. Modern Trends in Engineering and Research (IJMTER).

Hanchinamani G. and Kulkarni L. (2015). An efficient image encryption scheme based on Quintuple encryption using Gumowski-Mira and Tent maps, Int. J. of Contents, Vol. 11, pp. 56–69. https://doi.org/10.1007/s13319-015-0062-7

Joshi, A.B., Kumar, D., Gaffar, A. and Mishra, D.C. (2020a). Triple color image encryption based on 2D multiple parameter fractional discrete Fourier transform and 3D Arnold transform, Optics and Lasers in Engineering, Vol. 133, Article 106139. https://doi.org/10.1016/j.optlaseng.2020.106139

Joshi, A.B., Kumar, D., Mishra, D.C. and Guleria, V. (2020b). Colour-image encryption based on 2D discrete wavelet transform and 3D logistic chaotic map, J. Modern Optics, Taylor & Francis, Vol. 67, No. 10, pp. 933–949. https://doi.org/10.1080/09500340.2020.1789233

Loukhaoukha K., Chouinard J.Y. and Berdai A. (2012). A secure image encryption algo-

rithm based on Rubik's cube principle, J. Electrical and Computer Engineering, pp. 1–13. https://doi.org/10.1155/2012/173931

Maximov, A. (2007). Two linear distinguishing attacks on VMPC and RC4A and weakness of RC4 family of stream ciphers, Cryptology e-Print Archive, Report 2007/070. Retrieved from https://eprint.iacr.org/2007/070

Mishra, L.N. (2017). On existence and behavior of solutions to some nonlinear integral equations with applications, Ph.D. Thesis, National Institute of Technology, Silchar 788010, Assam, India.

Mishra, V.N. (2007). Some problems on approximations of functions in Banach spaces, Ph.D. Thesis, Indian Institute of Technology, Roorkee 247667, Uttarakhand, India.

Mishra, V.N. and Mishra L.N. (2012). Trigonometric approximation of signals (functions) in L_p norm, International J. Contemporary Mathematical Sciences, Vol. 7, No. 19, pp. 909–918.

Mondal, B., Sinha, N. and Mandal, T. (2015). A secure image encryption algorithm using LFSR and RC4 stream generator. In Third International Conference on Advanced Computing, Networking and Informatics (ICACNI). https://doi.org/10.1007/978-81-322-2538-6_24

Paul, S. and Preneel, B. (2004). A new weakness in the RC4 keystream generator and an approach to improve the security of the cipher. In FSE (Fast Software Encryption), LNCS, Springer-Verlag, Vol. 3017, pp. 245–259. https://doi.org/10.1007/978-3-540-25937-4 16

Peitgen, H.O. and Saupe, D. (1988). *The Science of Fractal Images*, Springer-Verlag.

Pickover, C.A. (1998). *Chaos and Fractals: A Computer Graphical Journey*, Elsevier.

Popov, A. (2015). Prohibiting RC4 cipher suites, RFC 7465. Retrieved from https://tools.ietf.org/html/rfc7465

Saptarini, N. and Alberth, Y. (2013). Digital color image encryption using RC4 stream cipher and chaotic logistic map. In Information Systems International Conference (ISICO).

Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms and Source Code in C* (second edition), John Wiley and Sons, New York.

Sivakumar T. and Venkatesan R. (2016). A new image encryption method based on knight's travel path and true random number, J. Information Science and Engineering, Vol. 32, pp. 133–152.

Tews, E. and Beck, M. (2009). Practical attacks against WEP and WPA. In Second ACM Conference on Wireless Network Security, Switzerland, pp. 79–86.

Tsunoo, Y., Saito, T. and Kubo, H. (2005). The most efficient distinguishing attack on VMPC and RC4A. In FSE (Fast Software Encryption), LNCS, Springer-Heidelberg, Vol. 3557, pp. 359–367.