International Journal of Security and Its Applications
Vol. 11, No. 9 (2017), pp.73-94
http://dx.doi.org/10.14257/ijsia.2017.11.9.06

A Stream Cipher Method for RGB Image Encryption Using PSO base Key Generation

Sepideh Bahri-Laleh*, Mohammad Ali Balafar* and Mohammad-Reza Feizi-Derakhshi

Department of Computer Engineering, Faculty of Electrical & Computer Engineering, University of Tabriz, Iran
**To whom correspondence should be addressed: sepideh.bahri@yahoo.com, balafarila@tabrizu.ac.ir*

Abstract

This paper introduces a novel approach based on a stream cipher to encrypt RGB images. In this regard, a Particle Swarm Optimization (PSO) based algorithm is used to generate the keystream for encryption. Red, green and blue channels of the RGB image are shown as texts and then a stream cipher is used to encrypt the resultant image. A method named PSO Key Generation Color image Encryption (PKGCE) algorithm uses 3 character code tables for each of the 3 channels of RGB image in order to encode the keys and the plain texts showing correspondent channel of the RGB image. A zigzag operation is done on the resultant image in each of the channels to provide more security in a way that the start point of the zigzag path is dependent on the key and channel size. The main advantages of this algorithm over the previously ones, are less number of keys needed to be stored and distributed and also the appropriate speed of the algorithm. Experimental results and the comparison of the proposed algorithm with other encryption algorithms are discussed in detail.

Keywords: *RGB image encryption, Stream cipher, PSO*

1. Introduction

In today's world of developing technology, it is becoming more difficult to keep information safe [1]. Therefore, security has been considered an essential issue during the transmission and storing of digital data [2] [25]. Digital images are a general type of two-dimensional data, which contain a large amount of information [3]. Due to the large development of digital images and multimedia use in nowadays applications, there is a need for fast and secure representation, transmission and storage of digital images [4]. Digital images are concerned with all kinds of security problems which other data face to them, such as unauthorized access and forgery detection [5]. Among different kinds of digital images, RGB images are one the most commonly used kinds of images in internet and other social media [6].

The traditional cryptographic algorithms such as DES, AES, IDEA, RSA *etc.*, are not suitable for practical image encryption because of intrinsic features of some images such as mass data capacity, strong correlation among pixels and high redundancy [7], so there is a need to develop new methods for digital images encryption.

In recent years, many image encryption methods have been presented. Commonly, the image encryption methods include two processes: pixel permutation and diffusion. Shannon first introduced the idea of using aforementioned processes to protect digital information [8]. The permutation process alters the location of image pixels. The

Received (July 22, 2017), Review Result (August 17, 2017), Accepted (September 8, 2017)

diffusion process changes the pixel values in a way that a small change in one pixel can spread to almost all pixels in the entire image [9].

Stream cipher is a symmetric key cryptography which the encryption and decryption key, used to encrypt and decrypt the image, is randomly altered in a way that the cipher image created is mathematically impossible to break. The benefit of using stream cipher is that when it is compared to block ciphers, the execution speed is higher and has less hardware complexity. Unlike block ciphers, stream ciphers, even for repetitive blocks of plain text, will not generate the same cipher text, since the keys are changed constantly for every element of plain text [10].

Different papers about image encryption algorithms have been introduced which use substitution and permutation processes. In reference [2], the gray plain image is divided into blocks. In the diffusion process, the location of pixels in each block is changed during a zigzag path. Location of the starting pixel of zigzag path depends on the image characteristics and is obtained from sub-keys. Then a substitution process and a mixing process are done over the image. Sixteen rounds are used in the encryption process. In fact, in our proposed method we will use the zigzag path on the plain image too, but the location of starting pixel is calculated in a different way. In [11], the halftone plain image is divided into a set of 2*2 blocks. Then the image is transformed into a decimal integer matrix M. Two neighbor elements of M are selected into a pair {x, y}, and an operation named Reversible Pairwise Swapping is done on the selected items. Then integers are transformed into binary sequences. Finally, all elements are permuted to obtain a new matrix based on a pseudo-random number sequence which is generated with a key K in a pseudo-random number sequence generator. The mentioned operations are iterated for some rounds. In article [1], Abdul Hanan Abdullah proposed a new method based on a hybrid model. The hybrid model is a combination of a genetic algorithm and a chaotic function. First, we use the original gray image and the chaotic function to make some encrypted images. Then we use the encrypted images as the initial population of the genetic algorithm. In every stage of the genetic algorithm, the answer gained from the former iteration is optimized to generate the image which is best-encrypted. The best encrypted image is gained from an optimized iteration of previous stages. In our method, we will use an optimization algorithm too but in order to produce the key. In [10], Sreelaja proposed a stream cipher for binary image encryption using Ant Colony Optimization for generating the key. First the binary image is encoded into a plain text representing the binary image in a way that if the value of the binary image in the corresponding column position is 0, it is replaced by the character ‘a’, but if it is 1 and column position mod 25 value is not equal to 0 it is replaced by the corresponding characters having the character value. To generate the keystream, a group of ant agents are chosen. A character code table is generated for the characters in the plain text representing the encoded binary image depending upon their probability of occurrence in the plain text. A character code tree is generated and the tree is traversed and the code is generated for the characters in the plain text representing the encoded binary image. The values for the characters are found by taking the decimal equivalent of the code. The tree is mutated at second level of the tree and a character code table is generated. The character code table obtained after mutation is used to encode the characters in the plain text representing the encoded binary image and the characters in the keystream occurring in the plain text. An XOR operation is performed with the encoded values of the characters in the plain text representing the encoded binary image and the key values to obtain the cipher image. Aforementioned paper proposes a method which uses both substitution and permutation processes to encrypt binary images, but in fact most of the time the images are in RGB form on the internet and the other social media. So, using a method to encrypt RGB images is useful. On the other hand, Ant Colony Optimization Algorithm is time consuming and utilizing it to produce encryption keys will result in an algorithm which takes a long time for encryption, especially in RGB images which we have 3 channels and the same process will happen for triple channels. The main idea of our proposed method is taken from [10]. In fact, we use similar method for RGB images, and we use a faster optimization algorithm (PSO) to generate the encryption key. Also in order to investigate the effect of division in encryption of images, we will use division of the encoded plain image on the key to produce the cipher image, before doing XOR operation between them. In fact, we want to introduce a RGB image stream cipher algorithm which is secure, fast enough, needs to keep less number of keys and uses a novel method by applying division in the algorithm. The rest of the paper is organized as follows. In section 2 the structure of encryption and decryption system is discussed. In section 3 we will go over PSO algorithm and its uses in our method. The main proposed algorithm is introduced in section 4 and section 0 and section 6. In section 7 case study is represented. In section 0 we show the experimental results. In section 9 a comparison of PKGCE with existing stream cipher procedures is represented. Security analysis of the proposed method is discussed in section 9.7. Section 0 discusses the conclusion.

2. Architecture of the System for Encryption and Decryption

A stream cipher procedure is used for encrypting and decrypting the RGB image channels.

2.1. Encryption Process

Figure 1 shows an encryption system model. A PKGCE algorithm performs a reduction phase on each of the red, green and blue channels of RGB image and then encodes the three RGB image channels in plain text forms having only lower case alphabets and generates the keystream for encryption. A zigzag operation is done on the resultant image in each of the channels to provide more security in a way that the start point of the zigzag path depends on the channel size. The keystream which is generated has a set of characters. The characters existing in the keystream define the keys to be used for encryption. To certify security, the same set of keys is never used more than once as a result of generating random key characters each time encryption process is performed. The characters in the plain texts showing the encoded RGB image channels and the keystream characters which happen in the corresponding plain text are encoded using the character code table values. Instead of replacing the characters placed in the keystream and in the plain texts showing the encoded RGB image channels with ASCII values, encoding of keystream and the plain texts with a character code table for each channel increases the system security. Even though the characters placed in the keystream are detected, the values will not be identified to the hackers because an encoding operation is performed. The security analysis of the system is explained in section 10. If the size of the plain text showing the encoded RGB image channel is more than the keystream size, then the key's values which are in the keystream are divided on a predetermined value and the remainder is considered as the key values for the plain text characters of the channel which are at a place more than the keystream size.

The predetermined value is computed by taking 2 radicals of the size of the plain text showing the encoded RGB image channel as represented in (1). The plain text showing the encoded RGB image channel is divided into blocks of keystream length size. The values of the keystream characters make the keys needed for the first block. The keys needed for the consecutive blocks of the plain text showing the encoded RGB image channel are obtained by considering the remainder of the division of the previous block key values on the predetermined value as represented in (2).

Predetermined value = radical (radical (size of the image channel)) (1)

Keys needed for block (i) = mod (keys placed in block (i − 1), predetermined value) $i \geq 2$ (2)

The encrypted RGB image channel is obtained by dividing plain text values of each channel by the corresponding key values. In each of the channels a XOR operation is done between the quotient of division and the key and also between the remainder of division and the key. Then the resultant quotient and remainder are considered as cipher texts.

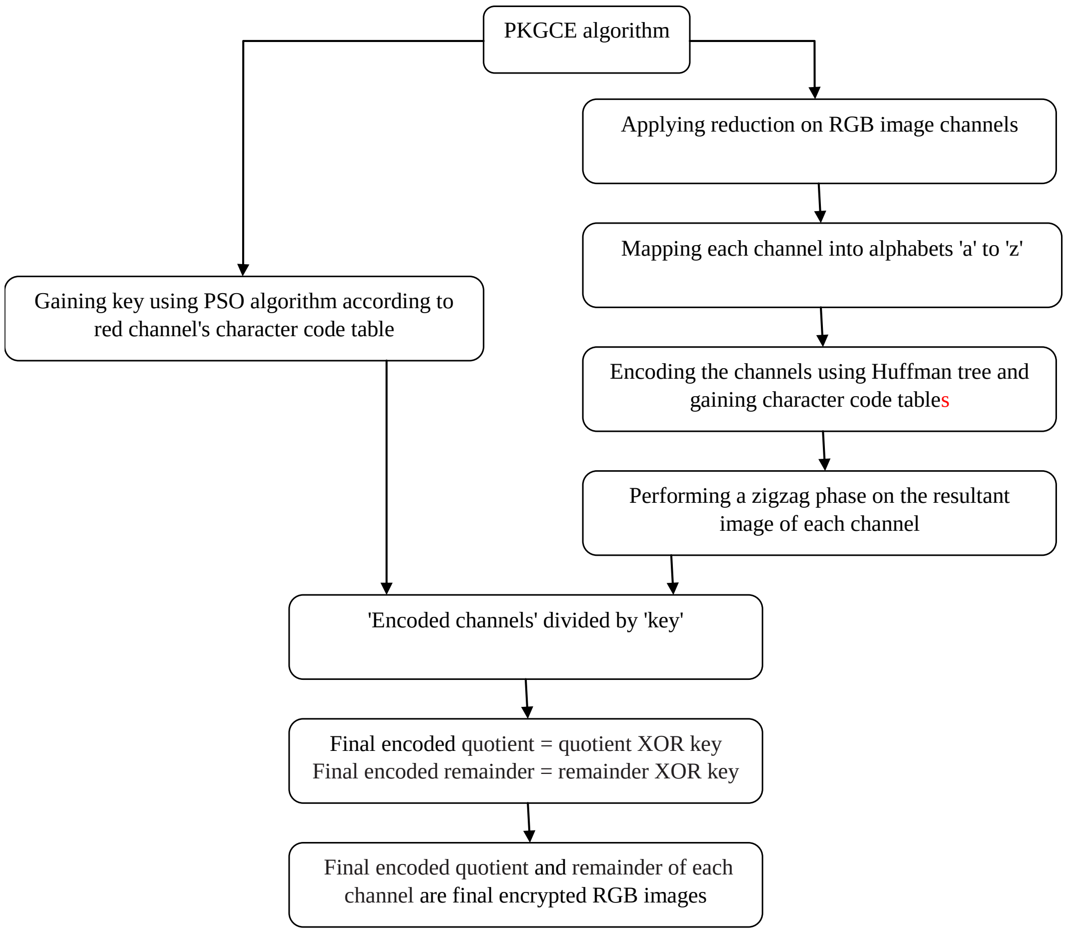

PKGCE algorithm
Applying reduction on RGB image channels
Mapping each channel into alphabets 'a' to 'z'
Gaining key using PSO algorithm according to red channel's character code table
Encoding the channels using Huffman tree and gaining character code tables
Performing a zigzag phase on the resultant image of each channel
'Encoded channels' divided by 'key'
Final encoded quotient = quotient XOR key
Final encoded remainder = remainder XOR key
Final encoded quotient and remainder of each channel are final encrypted RGB images

Figure 1. Model of Proposed Encryption System for Each Channel

2.2. Decryption Process

Figure 2 shows a model of proposed decryption system. The same operation is done over each channel separately and in each of the channels their own character code tables which were obtained in encryption process are used. So, the same process will be repeated on all the 3 channels but we elaborate it for one channel. The decryption system decrypts the obtained encrypted RGB image utilizing the same set of keys which were used for encryption. The content of the gained character code tables for each of the red, green and blue channels of RGB image which is used for encoding are hidden and the tables' base address are kept in the memory. The keystream generated by PKGCE algorithm used to encrypt the plain text showing the encoded RGB image channel is given to the receiver. The character code tables' base address used for encoding are taken from the memory and the keystream keys are encoded by a simple table lookup action. The keystream keys which are not happening in the channel's character code table are replaced with their ASCII value. The predetermined value is calculated by the receiver based on the encrypted RGB image channel's length. The receiver produces the key values for the section of the encrypted RGB image channel which exceeds the keystream's length by using the predetermined value and the values of the keys in the keystream. Therefore, the key values which are used for encryption will be identified to the receiver. In the next stage in order to obtain the values of the decoded quotient and remainder, an XOR operation is done between the given quotient of encrypted RGB image channel and the key values of the same channel, and another XOR operation is done between the given remainder of encrypted RGB image channel and the key values of the channel. To obtain the values of the plain text of each channel (3) is done for each channel:

Decoded channel (red, green or blue) values before zigzag operation (i) = key(i) * quotient(i) + remainder(i);   i=1 to image size (3)

In the next stage, we do a reverse zigzag path on the resultant elements of each channel. As mentioned in previous sections starting element location for traversing in a channel is made dependent on key and the image size. For this purpose, consider (x, y) is the start point of reverse zigzag route. We can gain the start point using (7). The obtained start point is the same for all channels.

Each channel's character code table is used to decode the plain text values to obtain the characters in the plain text showing the encoded RGB image channel. The encoded RGB image channel is then decoded in the form of numbers 4, 14, 24, 34 …, 234, 244, 253 using Table 1 to get the original reduced image channel.

3. PSO Algorithm

Particle swarm optimization (PSO) is a population based stochastic optimization method which is learned from social behavior of bird crowding or fish training [12]. A basic type of the PSO algorithm acts by owning a population (known as swarm) of candidate solutions (known as particles). When better positions are being detected, after that these will be used to guide the movements of the swarm. The procedure is repeated and by acting in that way it is hoped, but not assured, that a good solution will finally be detected [13].

We initialize PSO with a population of accidental particles (solutions) and then PSO searches for optimum solution by updating generations. In each iteration, we update each particle by subsequent two "best" values. The first value is the best solution (fitness) which it has attained so far (We store the fitness value too). This value is named pbest. The best value, gained so far by each of the particles in the population is the other "best" value that is followed by the particle swarm optimizer. This best value is considered a global best and named gbest. When a single particle participates the population as its own topological neighbors, the best value is considered a local best and is named lbest.

After the particle detects the two best values, it updates its velocity and positions with subsequent equation (4) and (5).

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (4)

present[] = persent[] + v[] (5)

v[] is called the particle velocity, persent[] is considered the current particle. gbest[] and pbest[] are defined as defined before. rand () is an accidental number between (0, 1). c1, c2 are learning factors. Generally, c1 = c2 = 2 [12].

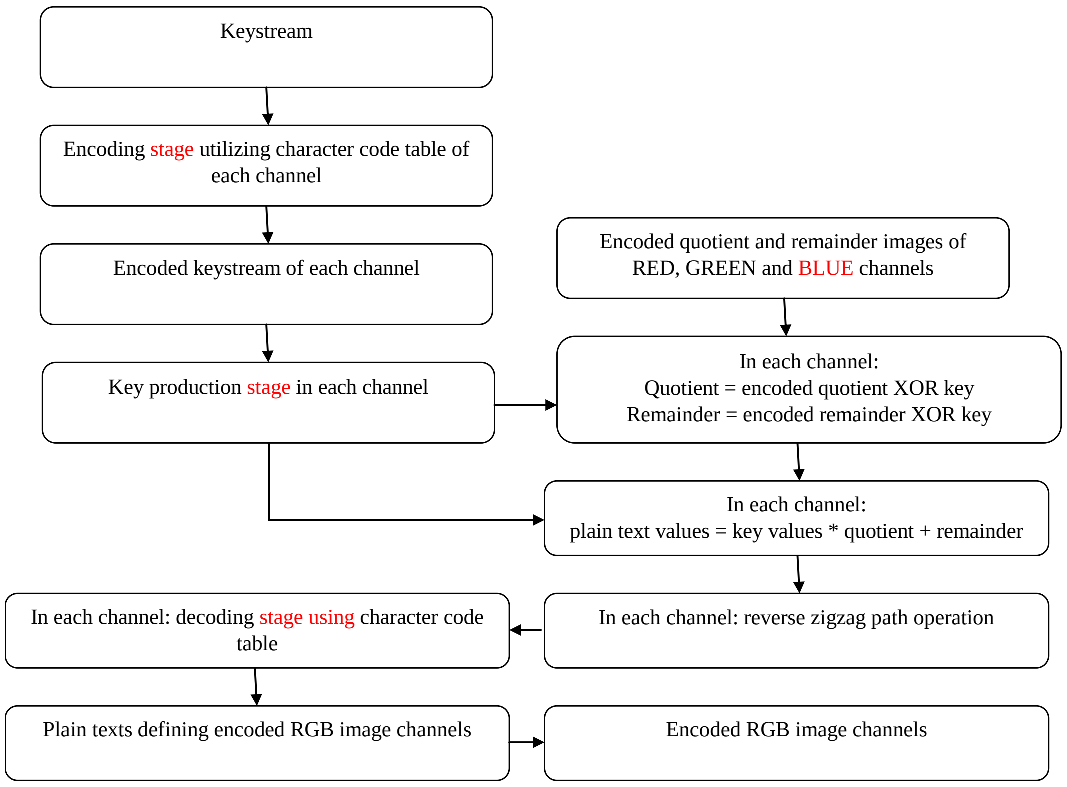

Keystream
Encoding stage utilizing character code table of each channel
Encoded keystream of each channel
Key production stage in each channel
Encoded quotient and remainder images of RED, GREEN and BLUE channels
In each channel: Quotient = encoded quotient XOR key Remainder = encoded remainder XOR key
In each channel: plain text values = key values * quotient + remainder
In each channel: reverse zigzag path operation
In each channel: decoding stage using character code table
Plain texts defining encoded RGB image channels
Encoded RGB image channels

Figure 2. Model of Proposed Decryption System

4. PSO Key Generation RGB (or Color) Image Encryption Algorithm

The PSO Key Generation RGB (or color) Image Encryption (PKGCE) algorithm is a method to generate the keystream for RGB image encryption. Figure 3 shows the diagram of a PSO System of gaining a keystream for RGB image encryption.

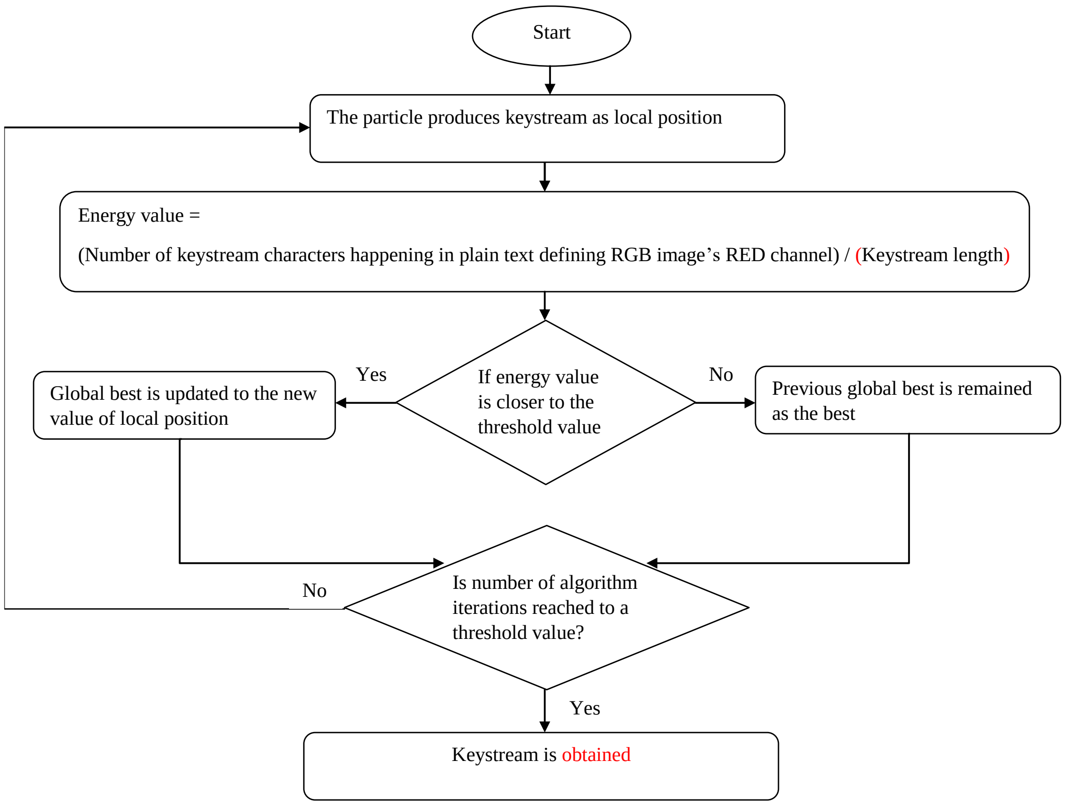

Start
The particle produces keystream as local position
Energy value =
(Number of keystream characters happening in plain text defining RGB image's RED channel) / (Keystream length)
If energy value is closer to the threshold value
Yes
Global best is updated to the new value of local position
No
Previous global best is remained as the best
Is number of algorithm iterations reached to a threshold value?
No
Yes
Keystream is obtained

Figure 3. Model of PSO System

The goal function of PKGCE algorithm is to generate a keystream considering the limitation that the probability of happening of characters of the keystream which are in the plain text showing the red channel of encoded RGB image is close or equal to a threshold value (we considered %75 to be the threshold value). First a stage of reduction is performed on the RGB image as is discussed in section 4.1. As it is discussed, each of the red, green and blue channels of the RGB image are encoded in the form of plain texts. The particles in the system work together to obtain an optimal solution. The particles seek the search space for a better solution and exchange the obtained information with each other. Each particle makes the solution by computing its local best and calculates the energy value. Initially each particle creates a candidate solution by choosing compositions of random characters. Each candidate solution of the particles shows the keystream. The particle calculates the energy value of its candidate solution. The energy value is calculated using (6).

$$\text{Energy Value} = \frac{\textit{number of candidate key characters happening in character code table}}{\textit{length of the key}} \qquad (6)$$

The particle considers an energy threshold value of 0.75 to obtain a solution (this threshold value can be other arbitrary values according to the encoder's needs). The objective of considering an energy threshold value of 0.75 is that close to 75% of the keystream characters demonstrating the candidate solution of the particle, happen in the plain text showing the red channel of encoded RGB image (or other arbitrary channels including green or blue) which would provide the encoding of characters in the keystream using a character code table (here red channel's character code table is used) thereby enhancing the system's security. As shown in (6) the energy value is calculated by counting the number of keystream characters which occur in the plain text showing the RED channel of encoded RGB image divided by the keystream length. The algorithm works in 100 iterations (it can be less or more than 100) which in each iteration the energy value is calculated for each candidate solution. After 100 iterations, each of the candidate solutions' energy value which is closer or equal to threshold value the candidate solution is considered as the keystream for encryption. In each iteration, each particle creates a candidate solution by selecting a different composition of characters as the keystream. The global best is kept during the process and in each iteration the global best is updated if the current solution is better than the global best. If two or more solutions have the same best energy values, the first solution is chosen as the keystream.

The fundamental blocks of PKGCE algorithm are elaborated below.

4.1. Reduction and Encoding RGB image as plain text

As explained in previous sections a true-color image is commonly a 24-bit color picture which every pixel of it contains red (R), green (G) and blue (B) components; each R, G, and B are showed by 8 bits, and each single color of R, G, and B ranges in an integer value between 0 and 255 [6]. In our proposed stream cipher the same stages are operated on all 3 channels. In order to decrease the number of integer values which we work with them and in order to increase the speed of the algorithm we apply a stage named “Reduction” on RGB image. In this manner, the RGB image integer values are placed in 26 groups in a way that values 0 to 9 are mapped to 4 and are replaced by ‘a’, values 10 to 19 are mapped to 14 and are replaced by ‘b’, values 20 to 29 are mapped to 24 and are replaced by ‘c’, and so on. The reason of choosing values 14, 24, 34, … is that these values are the middle of the groups 0 to 9, 10 to 19, 20 to 29, and so on. Table 1 shows the complete form of mapping RGB integer values to the middle element of each group and Table 2 shows an example of this process.

Table 1. Format of Mapping RGB Colors

Color range	0_9	10_19	20_29	30_39	40_49	50_59	60_69
Mapping color	4	14	24	34	44	54	64
Corresponding alphabet	a	b	c	d	e	f	g
Color range	70_79	80_89	90_99	100_109	110_119	120_129	130_139
Mapping color	74	84	94	104	114	124	134
Corresponding alphabet	h	i	j	k	l	m	n
Color range	140_149	150_159	160_169	170_179	180_189	190_199	200_209
Mapping color	144	154	164	174	184	194	204
Corresponding alphabet	o	p	q	r	s	t	u
Color range	210_219	220_229	230_239	240_249	250_255		
Mapping color	214	224	234	244	253		
Corresponding alphabet	v	w	x	y	z		

Table 2. Encoding a Channel of RGB Image in the Form of Plain Text

A channel of RGB image					Mapping color					Plain text for one channel
108	120	70	100	203	104	124	74	104	204	kmhku
0	55	189	100	16	4	54	184	104	14	afskb
111	255	55	179	90	114	253	54	174	94	hzfrj
78	203	170	0	255	74	204	174	4	253	huraz

In most of the RGB images the reduction process does not make many changes on original image. Figure 4 and Figure 5 show examples of applying reduction phase on two RGB images. The most visible changes are sensible when gradual changes of colors are seen in the image. When mild changes of colors are seen in the image, the result of reduction phase is a RGB image that similar colors with values close to each other are mapped to a single value, so some stripes are recognized in the image as shown in Figure 4. But in other images reduction phase does not make so much difference as shown in Figure 5.

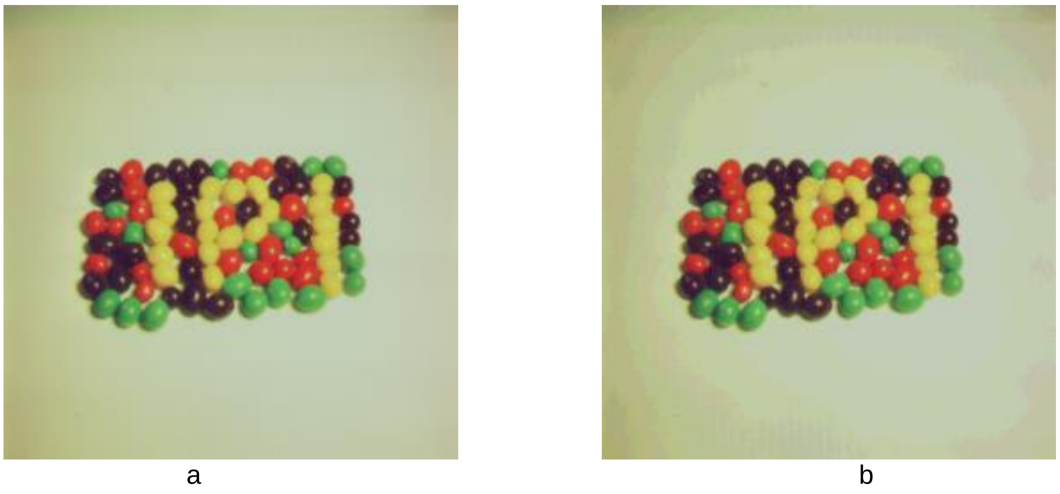


Figure 4. (a) Original Image and (b) Reduced form of Original Image

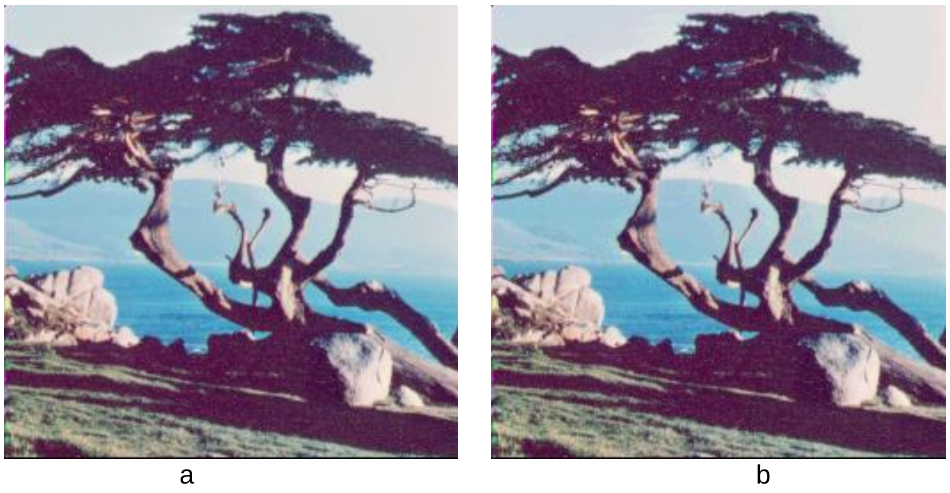


Figure 5. (a) Original Image and (b) Reduced form of Original Image

The RGB image which is showed by values 0 to 255 is encoded in the plain text form having just lower case alphabets. The alphabets are concatenated to each other to make a plain text defining the encoded RGB image. Table 2 shows the plain text showing a RGB image. The plain text defining the image in Table 2 is “kmhkuafskbhzfrjhuraz”.

5. Making Character Code Table in Order to Encode the Plain Text Defining Encoded RGB Image Channels and the Keystream

To generate the character code tables, based on the distribution of the characters of the plaintexts which show the encoded RGB image channels, 3 character code trees considering each of the red, green and blue channels are made. The trees are traversed and the contents are allocated to the characters which form the character code tables. If the values of the characters are equal in a table, '5' is added to the corresponding character's value to obtain unique values for the characters which are in a table.

5.1. Initial Character Code Table Generation

The characters which are in the plain texts showing the encoded RGB image channels are enumerated and demonstrated in the form of the tree according to the occurrence probability. The left-hand side branch of the tree is labeled as '0' and the right-hand side branch of the tree is labeled as '1'. Consider a channel of reduced RGB image "104 44 104 174 44 44". The RGB image is encoded in a plain text form as "kekree". Figure 6 represents the tree. The needed code is detected by traversing the tree. The character value is detected by obtaining the decimal amount of the code making the character code table. Table 3 shows the codes and the values.

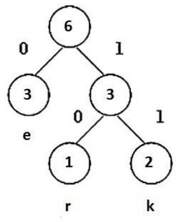


Figure 6. Tree Demonstrating the Character Code

Table 3. Codes and Values Produced for the Characters in the Plain Text Defining the RGB Image Channel

Character	Code	Value
e	0	0
r	01	1
k	11	3

6. Diffusion Process

In the diffusion process, elements of each channel are rearranged in the same channel by a zigzag path. For example, a path showed in Figure 7 is used to rearrange the elements of a channel of 8 × 8 image size. Imagine that elements stored in a RGB image channel are allocated numbers as demonstrated in Table 4. and we begin traversing the route of the image channel from element 54 (i.e. location of starting element is (2, 2)). The sequence of the matrix is 94 4 114 124 54 253 74 184 224 164 104 174. The number of elements before (2, 2) is 4 in the sequence. First we change the location of the elements at the sequence in a way that values in the start point and after start point are located at the beginning of the sequence. In our example resultant sequence after replacement is 54 253 74 184 224 164 104 174 94 4 114 124. Table 4. shows the resultant matrix. At the end a zigzag path according to Figure 7 is done over the matrix obtained from replacement which results in Table 4. .

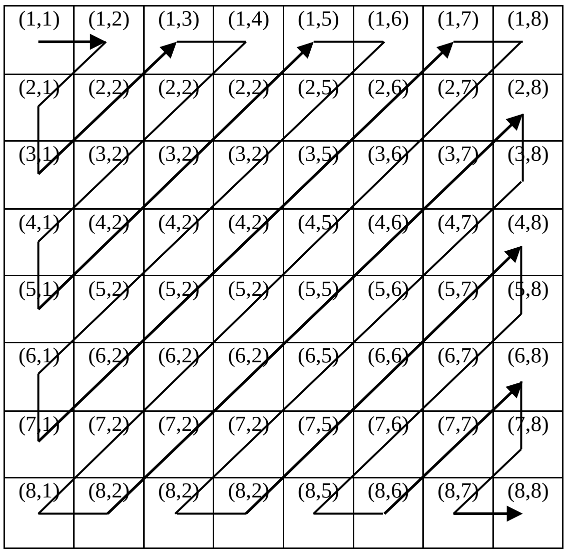


Figure 7. A Zigzag Path to Scramble Elements of a Channel

Table 4. Applying Zigzag Path in a Reduced RGB Image Channel. From Left to Right: Original Channel, Replacement of Values before Starting Point, after Applying Zigzag Path

a				b				c			
94	124	74	164	54	184	104	4	54	74	4	94
4	54	184	104	253	224	174	114	184	224	174	114
114	253	224	174	74	164	94	124	253	104	164	124

The location of starting element for traversing in a channel is made dependent on the key and the image size. For this purpose, consider (x, y) is the start point of zigzag path. We can obtain the start point using (7) and (8).

x = floor (length of image row / length of the key) (7)

if (x == 0) then (x = number of the rows);

if (x <= number of image columns) then (y = floor (number of the image columns / x) (8)

else (y = floor (x / number of the image columns)

7. Case Study

Suppose the process of encrypting channel RED of a RGB image shown in Table 6. The channel is encoded and the plain text showing the encoded channel is “aszaanzsznaaznxa”. To generate the keystream the threshold value is considered to be 0.75. Table 5 shows the candidate solutions found by 4 particles defining the keystream and their corresponding energy value. The first particle in iteration 1 has the candidate solution 'wyaz' defining the keystream. Only one character 'a' in the keystream happens in the plain text defining RED channel of the encoded RGB image. Thus, the energy value of the particle is 0.25. Candidate solutions of the other particles and their corresponding energy values are shown in Table 5. The process is repeated in 2 iterations. Since the third particle in iteration 2 has the best energy value 0.75 which is equal to the threshold value, the keystream 'znsd' corresponding to that particle is selected for encryption.

Table 5. Particles Finding the Keystream

Iteration 1	Energy value	Iteration 2	Energy value
wyak	0.25	zaas	1
bmfr	0.0	cdex	0.25
wsmo	0.25	znsd	0.75
naxa	1	szbb	0.5

A character code table is made for the characters which are in the plain text showing the encoded channel of the RGB image channel depending on their probability of presence in the plain text. A character code tree is generated and the tree is traversed and the code is generated for the characters in the plain text showing the encoded RGB image channel as shown in Table 7. The values for the characters in the plain text are found by taking the decimal amount of the code as shown in Table 7. For the characters, which have the same value, '5' is added to the value of the character to keep the values unique in a table. Then a character code table is generated. Table 7 shows the values for the characters in the plain text.

The obtained character code table of each channel is used to encode the characters in the plain text showing corresponding encoded RGB image channel and the characters in the keystream which occur in the corresponding channel's plain text. The characters in the keystream which do not occur in the plain text are substituted with their ASCII values. Because the length of the plain text showing the encoded RGB image channel is 16 and the keystream length is 4, the key values for the section of the plain text of the channel which exceed the keystream length are generated using a predetermined value and the key values in the keystream. The predetermined value is generated by taking 2 radicals of the length of the plain text showing the encoded RGB image channel. Since the length of the plain text showing the encoded RGB image channel is 16 the predetermined value is computed to be 2. In our example, the plain text showing the encoded RGB image channel is divided to 4 blocks. The values of the keys values in the keystream define the keys used for the first block of the channel. The key values which are used for the second block are generated by taking the remainder of the division of the first block's key values on the predetermined value. Similarly, the key values which are used for the third block and also fourth block are generated by taking the remainder of the division of the key values of the second block and the third block respectively on the predetermined value. The plain text showing the encoded RGB image channel is divided by the key, and a quotient and a remainder is obtained. If the key value is 0 then the quotient is set to 0 and remainder is set to the correspondent value of encoded RGB image channel. Obviously, the quotient and the remainder are at the same size with the original image. Then two XOR operations are done. The first XOR operation is done with the key and the obtained quotient, and the second XOR operation is done with the key and the obtained remainder. The two resultant images are considered as the cipher images.

Table 8Table 8 shows the encryption process. Figure 8 shows an example of original reduced RGB image and the correspondent cipher images of quotient and remainder in red, green and blue channels.

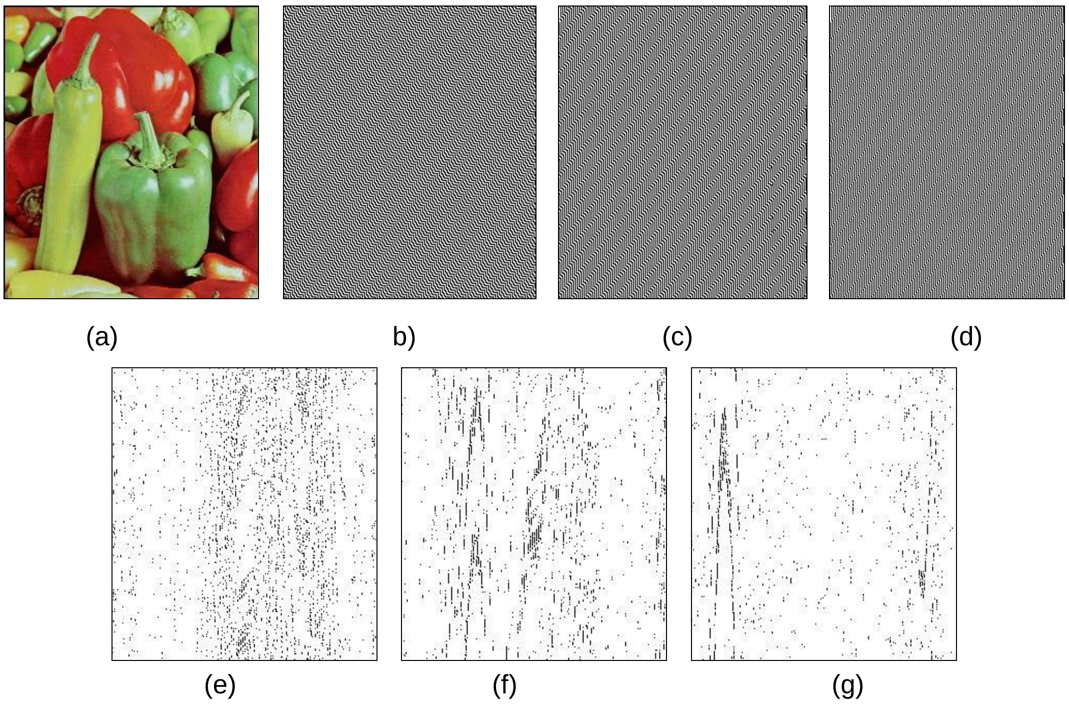


Figure 8. (a) Original Reduced RGB Image. (b) Divisor Cipher Image of Red Channel. (c) Divisor Cipher Image of Green Channel. (d) Divisor Cipher Image of Blue Channel. (e) Remainder Cipher Image of Red Channel. (f) Remainder Cipher Image of Green Channel. (g) Remainder Cipher Image of Blue Channel

Table 6. Channel of RGB Image to be Encrypted

Channel red of RGB image				Mapping color				Plain text for channel red
0	187	250	0	4	184	253	4	asza
9	131	255	189	4	134	253	184	anzs
251	135	4	7	253	134	4	4	znaa
253	130	235	2	253	134	234	4	znxa

Table 7. Values for Characters Showing the RGB Image Channel

Character	Huffman code using tree	Value using tree (decimal equivalent of character)
a	0	0
z	01	1
x	0011	3
n	111	7
s	1011	11

Table 8. Encrypting a Channel of a RGB Image

Keystream	Key values (K_i)	a reduced channel of a RGB image	Plain text showing encoded reduced RGB image channel (P_i)	Encoded values for plain text showing encoded reduced RGB image channel	Encoded values for plain text showing encoded reduced RGB image channel after zigzag route: $E(P_i)$	$E(P_i) \div K_i$		Cipher Images	
						Quotient (Q_i)	Remainder (R_i)	$C_{i1} = Q_i$ XOR K_i	$C_{i2} = R_i$ XOR K_i
z	1	0	a	0	1	1	0	0	1
n	7	184	s	11	0	0	7	7	0
s	11	253	z	1	7	0	7	11	12
d	100	0	a	0	3	0	3	100	103
	1	4	a	0	11	11	0	10	1
	1	134	n	7	0	0	0	1	134
	1	253	z	1	1	1	0	0	1
	0	184	s	11	7	0	7	7	7
	1	253	z	1	1	1	0	0	1
	1	134	n	7	0	0	0	1	1
	1	0	a	0	0	0	0	1	1
	0	4	a	0	1	0	1	0	5
	1	253	z	1	7	7	0	6	1
	1	134	n	7	0	0	0	1	1
	1	234	x	3	11	11	0	10	1
	0	0	a	0	0	0	0	0	0

8. Results and Discussions

The experiment is done for different images of various sizes from the RGB image repository databases [14] and [15]. The experiment was performed in systems possessing Intel(R) Core(TM) i5, 4 GB RAM.

Table 9 shows the time taken to encrypt RGB images of different sizes for keystream length 15. Figure 8 and Figure 10shows reduced RGB images and red, green and blue channels' cipher images of quotient and remainder of barche and tulips respectively.

Table 9. Time Taken to Encrypt RGB Images

RGB image size	Time in seconds
128*128	2.6
256*256	3.8
512*512	9.3

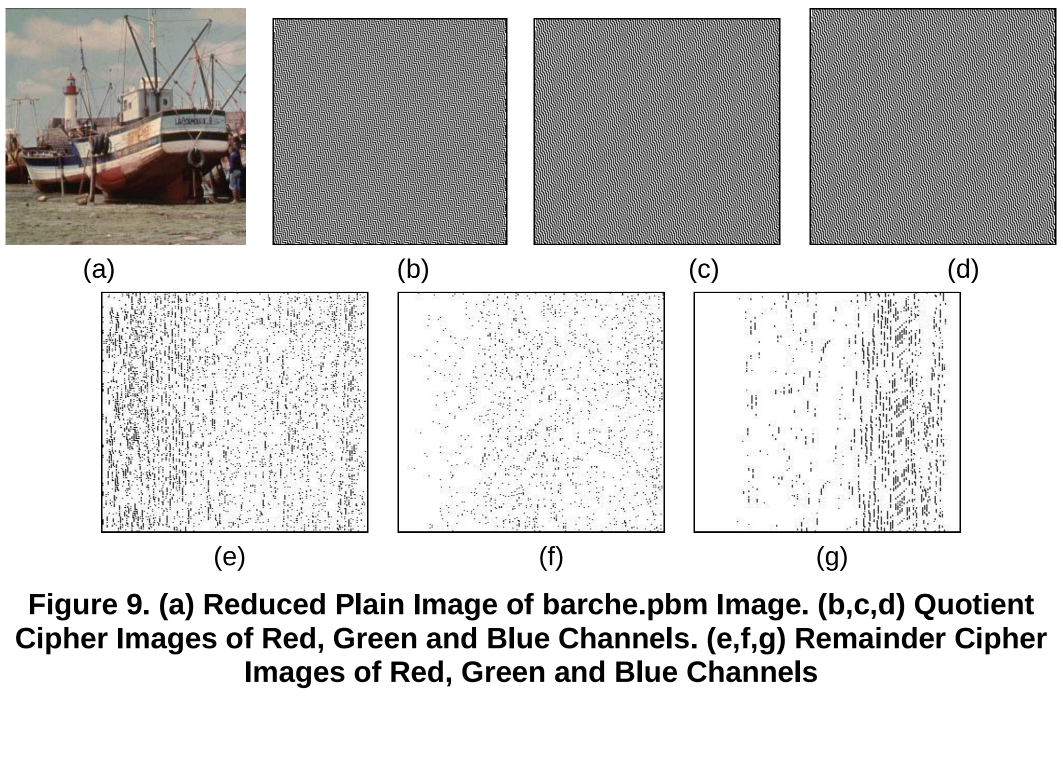


Figure 9. (a) Reduced Plain Image of barche.pbm Image. (b,c,d) Quotient Cipher Images of Red, Green and Blue Channels. (e,f,g) Remainder Cipher Images of Red, Green and Blue Channels

9. Comparison of PKGCE with Existing Stream Cipher Procedures

The stream cipher approach using PKGCE algorithm is compared with the present stream cipher procedures for image encryption. A comparison is showed in Table 10.

Table 10. Comparison of Stream Cipher using PKGCE Algorithm with Present Image Encryption Procedures

Image encryption procedure	Maximum number of keys which should be stored
PKGCE	104
Fast and secure stream cipher	256
Key Pooled RC4	2048
mv3	259

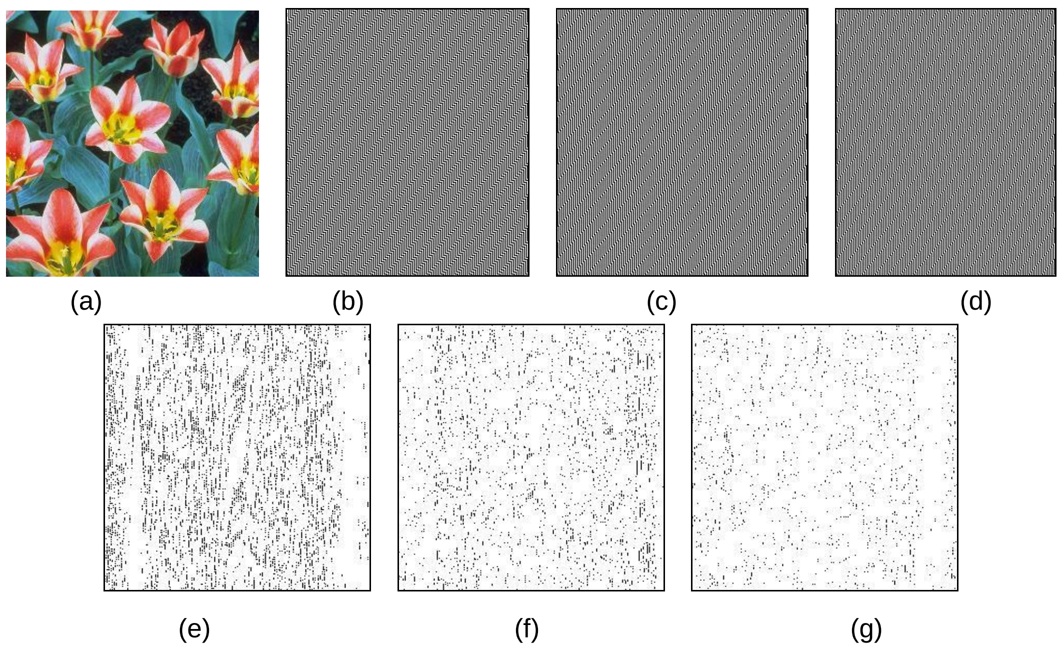


Figure 10. (a) Reduced Plain Image of tulips.ppm Image. (b, c, d) Quotient Cipher Images of Red, Green and Blue Channels. (e, f, g) Remainder Cipher Images of Red, Green and Blue Channels

It is shown that stream cipher using PKGCE approach scales well when it is compared to the existing approaches. In the stream cipher algorithm using PKGCE method, the number of keys to be shared does not need to be as long as the size of the elements constituting the RGB image red, green or blue channel. Let us suppose the case of encrypting a RGB image having a maximum of 300 elements in each channel. Even if we consider maximum number of keys situated in the keystream as 26 and maximum number of characters placed in the character code table of each channel is 26, the maximum general number of keys to be shared is 26*3 + 26 =104 which is less when it is compared to the other image encryption algorithms shown in Table 10. Let us consider the maximum number of characters placed in the keystream is 26. The reason for considering 26 characters is that it contains alphabets. Considering that the plain texts showing the encoded RGB image channels have all feasible characters, the maximum number of characters should to be kept in the character code tables is 26*3=78. Thus, the total number of keys to be distributed is 78+26=104.

9.1. Stream Cipher using PKGCE versus Fast and Secure Stream Cipher

Biham and Seberry [16] suggested a fast and secure stream cipher for encryption. The disadvantage in this algorithm is that entirely 256 keys have to be kept for initial permutation. The keystream produced is not related to the plain text to be encrypted. Moreover, the plain text is not encoded.

9.2. Stream Cipher using PKGCE versus Key Pooled RC4

Kim *et al.*, [17] suggested a method to implement and measure an effective RC4 stream cipher. The disadvantage in this algorithm is that the number of keys to be kept and shared is large. Moreover, the plain text is not encoded.

9.3. Stream Cipher using PKGCE versus a Partial Image Encryption Scheme based on DWT and ELKNZ Chaotic Stream Cipher

El-Khamy *et al.*, [18] have suggested a partial encryption system based on the Discrete Wavelet Transform (DWT), and chaotic stream cipher (ELKNZ). The disadvantage in this approach is that the length of the key is 464 bits.

9.4. Stream Cipher using PKGCE versus Stream Cipher for Image Encryption

Wu and Jay Kuo [19] have suggested an algorithm of encrypting images using a stream cipher procedure. The disadvantage is that the keys can be discovered if the pseudorandom number generator is cracked.

9.5. Stream Cipher using PKGCE versus Image Encryption using DCT and Stream Cipher

Krikor *et al.*, [20] have suggested a Selective Encryption method for image encryption. The disadvantage of this method is that the pseudorandom bit sequence for encryption can be obtained if the pseudorandom bit generator is cracked. Furthermore, the chosen coefficients and the keys utilized to produce the pseudorandom bit sequence and the keys utilized for shuffling have to be transferred to the receiver.

9.6. Stream Cipher using PKGCE versus Image Encryption Algorithm based on Chaotic System

Liu *et al.*, [21] have suggested a chaos-based image encryption system, using stream cipher architecture. The disadvantage of this method is that the keys may be cracked because a pseudorandom number producer is used. Moreover, a more number of keys are shared.

9.7. Stream Cipher using PKGCE versus mv3 Stream Cipher

Keller *et al.*, [22] have suggested an mv3 which is a novel stream cipher which is word based for encrypting long streams of data. The disadvantage of this method is that more number of keys is shared.

10. Security Analysis of Stream Cipher Approach using PKGCE

The security perspectives of the stream cipher approach using PKGCE algorithm are considered based on the attack model and also statistical analysis.

10.1. Attack Model

It is considered that the detail of encryption or decryption method is informed to the cryptanalyst.

10.1.1. Cipher Image Only Attack

The cryptanalyst has just the cipher images including quotient and remainder to analyze. The attacker analyzes the cipher images by searching the analogies in the cipher images. Because the cipher images are sincerely random, the attacker has to perform the brute force exhaustive key search attack. The stability of algorithm to resist exhaustive key search attack depends on a large key space. In each of the red, green or blue channels the keys are altered for each character of the plain texts showing the encoded RGB image channels to generate cipher images that are mathematically impractical to break. Because 90 different characters including lower case alphabets, upper case alphabets, numbers and special characters such as @, #, &, *etc.*, are considered as key database in PSO algorithm used for key generation, in each channel, to pass the XOR stage between resultant cipher images including quotient and remainder and the key, the attacker has to try all possible characters of the key space on each element of cipher images which the order of this stage is $90^{\text{image size}}$. This order can be obtained by considering the division stage too in a way that to pass the division stage, the attacker has to try all 90 characters on each element of cipher images to find the key which all possible states will be $90^{\text{image size}}$. The reason of not considering factorial manner in division stage, 90!, is that the same elements in quotient or remainder do not mean that the encoded plain text and the key had the same values, so the attacker has to try all possible states even thought the same values are seen in quotient and remainder. For example, in equivalents 25*2+0=50 and 50*2+0=100, even though in both equivalents the quotient and remainder are 2 and 0 respectively, but divided (encoded plain text) and divisor (key) are not the same. The benefit of doing XOR operation between the quotient and remainder after dividing encoded plain text by the key is that the attacker cannot find any relation between the key and the quotient or remainder by division operation rules. Moreover because of a zigzag path which is performed on the image before division operation which its start point depends on the size of the key generated by PSO algorithm, each element of the image can be the start point because the size of the key generated by PSO algorithm is not known to attacker. So image size is multiplied to obtained order of states which results in "$90^{\text{image size}}$*image size" order of number of states. At the end because 26 characters including lower case alphabets are used to encode the original reduced plain image by traversing a tree made up of 26 lower case alphabets, the attacker has to make all possible trees which the order of the operation is 26!. At last considering 3 channels the attacker has to test all the states with order "$90^{\text{image size}}$*image size*26!*3" to do brute force exhaustive search attack which means practically it is impossible to do this attack on the algorithm.

10.1.2. Known Plain Image Attack

The cryptanalyst has several characters in the main data and the corresponding cipher images including quotient and remainder cipher images. These could have been obtained either by inside information or by guessing. Their aim is to find the original RGB image channels using these values. To find the original RGB image channels, the keystream used for encryption and the character code tables of the channels have to be found. The encoding of plain texts indicating the image and the keystream would enhance the security in such a way that it is difficult to learn the values assigned for the characters in the plain texts showing the encoded RGB image channels.

Assume 'P_i' to be a character in the plain text indicating one of the channels of RGB image and 'C_{iQ}' and 'C_{iR}' are the corresponding cipher images in quotient and remainder cipher images respectively. Let $E(P_i)$ indicate the encoded value of the plain text.

Considering the stream cipher approach, first an XOR operation is done between the quotient value C_{iQ} and the key value K_i and also between the remainder value C_{iR} and the key value K_i and then (9) is performed:

$$E(P_i)=C_{iQ} * K_i + C_{iR} \tag{9}$$

Division and XOR stages are separate. So, concentrating on one of the operations with more or the same complexity than the other can help to calculate the complexity of cryptanalysis. The attacker has just P_i and C_i. Thus, the corresponding $E(P_i)$ and the key value K_i has to be found. To find $E(P_i)$, the character code table used for encoding has to be found. Because the character code table generated depends on the characters in the plain text, the attacker predicts character code tables of all probable orderings. The order of the number of probable character code trees is from 26!.

Moreover, to find K_i, the keystream has to be detected from $90^{\text{image size}}$ possible keystreams. So, it is difficult for the attacker to find the keystream used and the character code table used for encoding from the greater key space and the greater number of possible character code tables.

10.1.3. Chosen Plain Image Attack

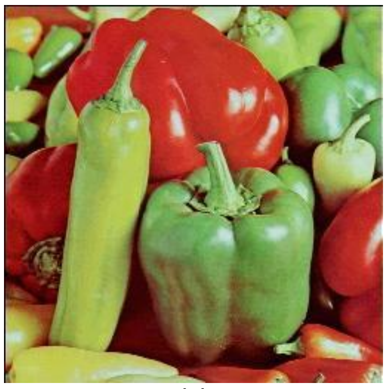
In this type of attack, the attacker could choose the plain text showing the encoded RGB image and obtain the corresponding cipher image values. The aim is to obtain the secret key to decrypt the cipher image. In each channel, first the encoded values of the characters of the plain text showing the encoded RGB image is divided by the key and then an XOR operation is done between the key and the encoded values of the characters of the plain text showing the encoded RGB images including quotient and remainder of previous stage to obtain the final cipher images values including final quotient and remainder values. The algorithm as explained above is not vulnerable to chosen-plaintext attack.

10.2. Statistical Analysis

A good cryptosystem should be resistive versus any statistical attack [23].

10.2.1. Histogram Analysis

An image-histogram explains the distribution of the pixels of the image by plotting the number of pixels at each intensity level. To show the security of the system and to intercept the leakage of information, it is essential for the cipher image to have no statistical resemblance to the plain image. The histograms of the reduced form of original image along with the encrypted images are analyzed and are showed that both the images have largely different contents. The x-axis and y-axis of the histogram defines the tonal variations and number of pixels in that special tone respectively (Figure 11 and Figure 12).



(a)

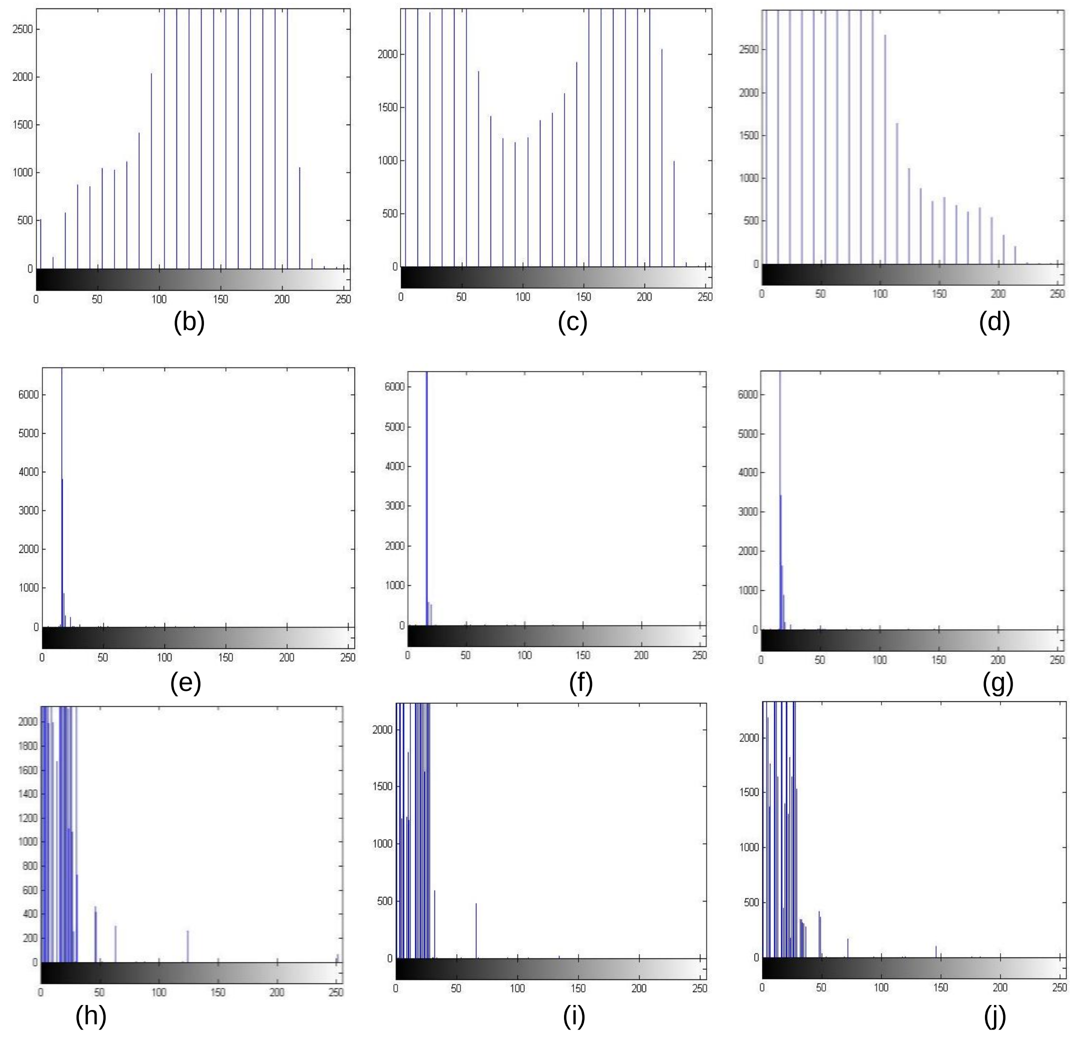
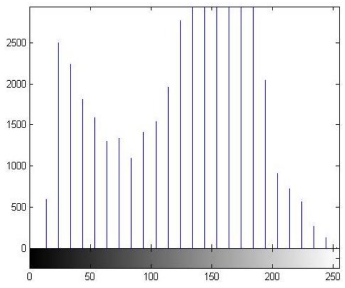


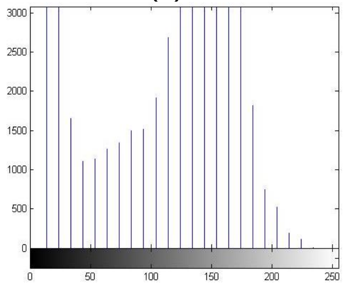
Figure 11. (a) Reduced Image of peppers.ppm. (b, c, d) Histograms of Red, Green and Blue Channels of Reduced Image. (e, f, g) Histograms of Quotient Cipher Images of Red, Green and Blue Channels of Reduced image. (h, i, j) Histograms of Remainder Cipher Images of Red, Green and Blue Channels of Reduced Image

It is apperceived from the histograms of the cipher images that all the spikes are considerably different from those of the plain images and hence have no statistical analogy to the plain image and therefore do not give any clue to apply any statistical attack on the stream cipher approach utilizing PKGCE algorithm for RGB image encryption.

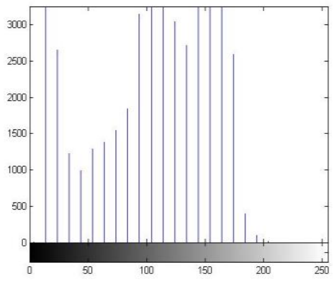
(a)



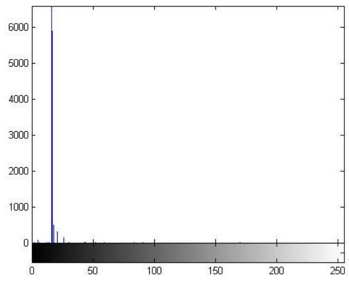
(b)



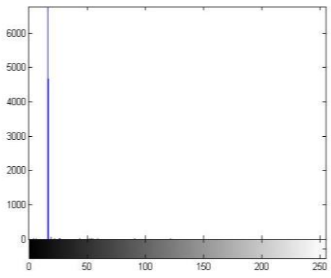
(c)



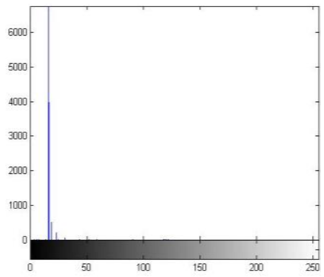
(d)



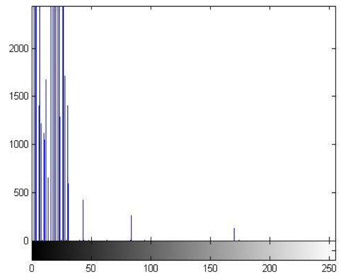
(e)



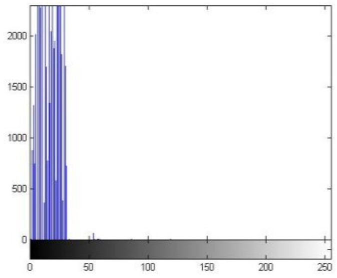
(f)



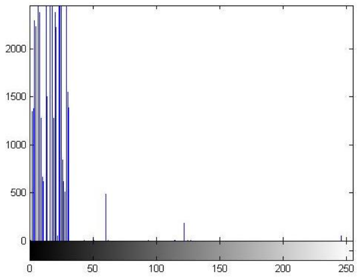
(g)



(h)



(i)



(j)

Figure 12. (a) Reduced Image of barche.pbm. (b, c, d) Histograms of Red, Green and Blue Channels of Reduced Image. (e, f, g) Histograms of Quotient Cipher Images of Red, Green and Blue Channels of Reduced image. (h, I ,j) Histograms of Remainder Cipher Images of Red, Green and Blue Channels of Reduced Image

10.2.2. Correlation Co-efficient Analysis

The correlation coefficient between the cipher images and the reduced plain image is also surveyed to show the analogy between images. The analogy between images will disclose the identity of the plain image. The correlation coefficients of the cipher images and plain image establish that the proposed method has a good capability of confusion and diffusion and highly resistive versus the statistical attack [24].

If the correlation value is between −0.5 and −1.0 or 0.5 and 1.0, it shows a strong negative correlation or strong positive correlation between the images. If the correlation value is between −0.1 and −0.5 or 0.0 and 0.5, it shows a weak negative correlation or weak positive correlation between the images. The images were taken from the RGB image databases [14] and [15]. Table 11 shows the correlation coefficient between the cipher images and the reduced plain image for a keystream length value 15, respectively. From Table 11, it is concluded that there is a weak correlation between the cipher images and plain images. This points that the RGB images encrypted using the stream cipher using PKGCE is resistive to statistical attacks.

Table 11. Correlation Coefficient between Reduced Plain and Cipher Images for Keystream Length 15

Correlation coefficient	Red channel		Green channel		Blue channel	
	Between reduced image and quotient	Between reduced image and remainder	Between reduced image and quotient	Between reduced image and remainder	Between reduced image and quotient	Between reduced image and remainder
peppers	-0.0103	-0.0029	-9.3644e-04	-0.0129	-0.0042	0.0039
barche	0.0013	0.0026	7.0863e-04	0.0094	3.8009e-04	0.0080
tulips	3.3366e-06	-0.0048	-0.0010	-0.0052	6.0073e-04	0.0016

11. Conclusion

This paper talks about a stream cipher approach used for encrypting the RGB image. A Particle Swarm Optimization base method termed PKGCE algorithm is used for generating keystream for the RGB image to be encrypted. This approach of encryption using a stream cipher decreases the number of keys to be stored and distributed. Moreover, the keys and the characters in the plain texts showing the encoded RGB image channels, including red, green and blue channels, are encoded using the values in the character code tables. This makes it hard for the cryptanalyst to trace the character code table used for encoding the keys and the plain texts showing the encoded RGB image channels. Moreover, a zigzag operation is done on the resultant image in each of the channels to provide more security in a way that the start point of the zigzag path depends on the key and channel size. PKGCE algorithm illustrates the potential to encrypt RGB images of different sizes. It overcomes the drawbacks of some of encryption approaches in terms of the storage and distribution of keys. Also, this approach suggests a stream cipher algorithm with an appropriate speed. This approach is resistive against different kinds of attacks, shows weak correlation coefficient between plain image and cipher image and also the histograms between the images are completely different.

References

[1] A. H. Abdullah, R. Enayatifar and M. Lee,"A Hybrid Genetic Algorithm and Chaotic Function Model for Image Encryption", Int. Journal Electron Commun (AEÜ), vol. 66, **(2012)**, pp. 806-816

[2] N. K. Pareek, V. Patidar and K. K. Sud, "Diffusion–substitution Based Gray Image Encryption Scheme", Digit Signal Process, vol. 23, **(2013)**, pp. 894-901

[3] Z. Hua and Y. Zhou, " Image Encryption Using 2D Logistic-adjusted-Sine Map", Inf Sci., vol. 339, **(2016)**, pp. 237-253

[4] A. Houas, Z. Mokhtari and K. E. Melkemi, "Boussaad A A Novel Binary Image Encryption Algorithm Based on Diffuse Representation", Int Journal Eng. Sci. Technol., vol. 19, no. 4, **(2016)**, pp. 1887-1894.

[5] X. Li, C. Li and I.-K. Lee, "Chaotic Image encryption Using Pseudo-random Masks and Pixel Mapping", Signal Process, vol. 125, **(2016)**, pp. 48-63

[6] H.-I. Hsiao and J. Lee, "Color Image Encryption Using Chaotic Nonlinear Adaptive Filter", Signal Process, vol. 117, **(2015)**, pp. 281-309

[7] X. Wu, D. Wang, J. Kurths and H. Kan, "A Novel Lossless Color Image Encryption Scheme Using 2D DWT and 6D Hyperchaotic System", Inf. Sci., vol. 349–350, **(2016)**, pp. 137-153
[8] W. Zhang, H. Yu, Y-l. Zhao and Z-l. Zhu, "Image Encryption Based on Three-dimensional Bit Matrix Permutation", Signal Process, vol. 118, **(2016)**, pp. 36-50
[9] S. M. Seyedzadeh, B. Norouzi and S. Mirzakuchaki, "RGB Color Image Encryption Based on Choquet Fuzzy Integral", Journal Syst. Software, vol. 97, **(2014)**, pp. 128-139
[10] N. K. Sreelaja and G. A. V. Pai, "Stream Cipher for Binary Image Encryption Using Ant Colony Optimization Based Key Generation", Appl. Soft Comput., vol. 12, **(2012)**, pp. 2879-2895
[11] B. Chen, H. Luo, Z. Zhao and X. Zhang, "Halftone Image Encryption Based on Reversible Pairwise Swapping", Meas, vol 57, **(2014)**, pp. 85-93
[12] http://www.swarmintelligence.org/tutorials.php.
[13] https://en.wikipedia.org/wiki/Particle_swarm_optimization.
[14] http://decsai.ugr.es/cvg/dbimagenes/:
[15] http://sipi.usc.edu/database/:
[16] J. S. Eli Biham, "Py(Roo): A Fast and Secure Stream Cipher", Symmetric Key Encryption Workshop (SKEW), **(2005)**; Aarhus Denmark.
[17] H. Kim, J. Han and S. Cho, "An Efficient Implementation of RC4 Cipher for Encrypting Multimedia Files on Mobile Devices", Proceedings of the ACM symposium on Applied computing. ACM, **(2007)**; Seoul, Korea.
[18] S. E. El-Khamy, M. A. El-Nasr and A. H. El-Zein, "A Partial Image Encryption Scheme Based on the DWT and ELKNZ Chaotic Stream Cipher", MASAUM Journal of Basic and Applied Sciences, vol. 1, no. 3, **(2009)**, pp. 389-394.
[19] W. Chung-Ping and C. C. J. Kuo, "Design of Integrated Multimedia Compression and Encryption Systems",z, IEEE Transactions on Multimedia, vol. 7, **(2005)**, pp. 828-839
[20] S.B. Lala Krikor, Arif. Thawar, Shaaban. Zyad, "Image encryption using DCT and stream cipher". European Journal of Scientific Research, vol. 32, **(2009)**, pp. 48–58.
[21] S.-H. Liu, J. Sun and Z.-Q. Xu, "An Improved Image Encryption Algorithm Based on Chaotic System", Journal of Computers, vol. 17, **(2009)**, pp. 1091-1100.
[22] S. D. M. N. Keller, I. Mironov and R. Venkatesan, "MV3: A New Word Based Stream Cipher Using Rapid Mixing and Revolving Buffers", Cryptographers' Track at the RSA Conference, Topics in Cryptology-CT-RSA, **(2007)**.
[23] H. E. H. Ahmed HMK and O. S. F. Allah "An Efficient Chaos-based Feedback Stream Cipher (ECBFSC) for Image Encryption", Informatica, vol. 31, no. 1, **(2007)**, pp. 121-129**.**
[24] F. Peng, Q. Shui-Sheng and M. Long , "An Image Encryption Algorithm Based on Mixed Chaotic Dynamic Systems and External Keys", Proceedings International Conference on Communications, Circuits and Systems, **(2005)**.
[25] G. K. Sodhi and G. S. Gaba, "DNA and Blum Blum Shub Random Number Generator Based Security Key Generation Algorithm", International Journal of Security and Its Applications, vol. 11, No. 4, **(2017)**, pp. 1-10.