Int. j. inf. tecnol.
https://doi.org/10.1007/s41870-018-0100-9

ORIGINAL RESEARCH

Color image encryption using DNA based cryptography

Nabarun Nandy[1] · Debanjan Banerjee[1] · Chittaranjan Pradhan[1]

Received: 28 October 2017 / Accepted: 22 January 2018

Abstract Data are vulnerable and need utmost security when they are getting transferred. In image encryption methodologies, the pixels of original images are either manipulated or information is laid inside the image using the image as a cover to protect the data from undesired receivers. In this paper, the images are transferred text files over the unsecured network. We have used an algorithm which uses random key generation and assignment of special codes to most occurring color and encoding range of repeated colors has been incorporated to decrease the text file size and give a layer of security along with this DNA components A, T, G, C decreases the chance of recognizing the data as image.

Keywords Cryptography · Decryption · DNA cryptography · Encryption · Image · Security

1 Introduction

Data are vulnerable and need utmost security when they are getting transferred from source to destination. Encryption is one of the most formidable ways to keep our data and information safe between two endpoints. It makes data impenetrable and unread-able, so even it ends up getting in wrong hands it is mostly useless [1].

✉ Nabarun Nandy
nabarun286@gmail.com

Debanjan Banerjee
debanjan.banerjee98@gmail.com

Chittaranjan Pradhan
chitaprakash@gmail.com

1 KIIT Deemed to be University, Bhubaneswar, India

DNA (deoxyribonucleic acid) is a genetic information carrier from parent organisms to the newly forming organism. The methodologies of encryption in which compo-nents of DNA are used to hide plain text from eavesdropper or unauthorized users in the network come under the study of DNA cryptography [2]. In DNA cryptography, the four chemical bases of DNA namely adenine (A), guanine (G), cytosine (C) and thymine (T) has so far majorly been used in addition to cryptographic algorithms but with advancement in the study of other components like Amino acids are also brought into the play [3]. Considering 'ATGC' to be a code we can have 24 combinations of it like 'CTAG', 'TAGC' etc. ($4! = 24$). Although among the 24 combinations only 8 are possible in a real life occasion as 'A' can combine only with 'T' and 'C' combines only with 'G' following the properties of DNA [3]. But, here we shall use all the 24 combinations to encrypt our data. This will increase the complexity of detecting patterns in the encrypted text.

In the study of cryptography, an image which contains information is converted to an unreadable or unrecognizable form by using algorithms. This field is gaining much popularity as image can carry vital information with them. The capacity of carrying information is also more by an image. In image encryption methodologies, the pixels of original images are either manipulated of information is laid inside the image using the image as a cover to protect the data from undesired receivers [4]. Image encryption enables passing of data over unsecured networks like the internet. Without the correct key, the correct decryption to retrieve the original is a major challenge to high-end super computers [5]. Although image encryption might require more than normal text encryption but it is more formidable. Image encryption plays a vital role in securing the transmission of important government document images, images of military, healthcare and other private images [6].

Section 2 gives a background of work relating to image encryption and DNA cryptography together; Sect. 3 provides the description of proposed algorithm, utility of some of its steps and the entire algorithm; Sect. 4 documents the result analysis and Sect. 5 concludes the paper.

2 Background of DNA cryptography used in image encryption

Akkasaligar et al. proposed a secure medical image encryption based on intensity level using chaos theory and DNA cryptography [7]. Niyat et al. have implemented chaos-based image encryption using a hybrid cellular automata and a DNA sequence [8].

Ochani et al. developed security in the medical image, which can be done by hiding patients related data into another image and transfer it safely to a defined location. Cryptography along with steganography are two popular methods available to impose security [9].

Soni et al. proposed an algorithm by dividing an binary matrix into two blocks and rather implementing two logistic maps [10]. Mokhtar et al. have used the chaotic logistic map for confusing and diffusing the image pixels, and then a DNA sequence used as a OTP (one-time-pad) to change pixel values [11].

3 Proposed algorithm

The goal of the algorithm is to encrypt the original image in the form of a text file and prepare it to pass over unsecured network. On receiving the text file, it should be able to convert it to the original image.

At first, all the combinations of 'ATGC' is found out and stored in an array named 'CODE'. Out of 24 combinations, 16 will be randomly chosen out of CODE and put in an array of size 16 (0–15) named STACK. Each combination will be the code of the corresponding index and index 10–15 will be considered 'A' to 'F' (since we shall deal with hexadecimal code of colors). The entire image will then be scanned and each pixel's value will be found out and put in the form of matrix. The entire matrix will be scanned to find the seven most occurring hexadecimals and stored in an array MOST_OCCUR in non-increasing order. From the remaining 8 combinations left in CODE, 7 codes will be specifically assigned to these most occurring colors. The last remaining code will be given to '#'. The height and width of the image will be encoded with the corresponding codes of STACK (0–F) and written on a file CODE_FILE with code of '#' after width and height. For example, suppose height and width of image is 10 $\times$ 10, then 'TGCACTGA' (code of 10) + code of '#' will be written.

Now, each hexadecimal of the formed matrix will be taken if: (a) member of MOST_OCCUR, corresponding code will be written (b) same as the last hexadecimal, a counter will count the number of times it occurs consecutively and then the value of counter will be encoded by writing the corresponding codes of its digits, adding the code of '#' before and after the entire coded form of counter. For example, suppose '1010F1' occurs 10 times serially, then the code would be 'TGCACTGATGCACTGAACGTTGCA' (CODE OF 1010F1) + code of '#' + 'TGCACTGA' (code of 10) + code of '#'. (c) If not a member of MOST_OCCUR and not same as last hexadecimal, then each character will be extracted and its corresponding code from STACK will be written in CODE_FILE. The hexadecimals of most occurring color will then be treated as a normal color and written in a file MOST_OCCUR_FILE. Here, utility of most occurring color search and assigning of special codes.

Suppose 'FFFF00' (hexadecimal code of yellow) occurs most number of times in an image, let's say 100 number of times. Equivalent code for 'FFFF00' will be 'ACGTACGTACGTACGTCTGACTGA'. Total number of characters = 24 $\times$ 100 = 2400. But, with the implementation of most occurring color 'FFFF00' gets a particular 4 character code, say 'GACT'. Here, the total number of characters = 4 $\times$ 100 = 400, which is 2000 characters less; hence reducing the file size and making decryption without key tougher.

Utility of serially occurring repeating color range encoding: suppose 'FFFF00' occurs in consecutive 10 pixels. The equivalent code for 'FFFF00' will be 'ACGTACGTACGTACGTCTGACTGA'. Total number of characters = 24 $\times$ 10 = 240. But, when we implement count = 10 assign a particular code for it, let say 'TGCACTGA', then the code will look like the following:

'ACGTACGTACGTACGTCTGACTGA' + code for '#' + 'TGCACTGA' + code for '#'.

Total number of characters used = 24 + 4 + 8 + 4 = 40, i.e. 200 characters less; hence decreasing the file size and giving a layer of security. Here, the content of CODE, MOST_OCCUR and value of '#' collectively acts as a key. The MOST_OCCUR_FILE also partially acts as a key. During decryption, absence of either the key or the MOST_OCCUR_FILE will not produce the required output, i.e. the original image. When both are present, the image will be generated.

3.1 Encryption

Step 1: Finding all the 24 combinations of ATGC and storing them in an array named CODE of size 24.

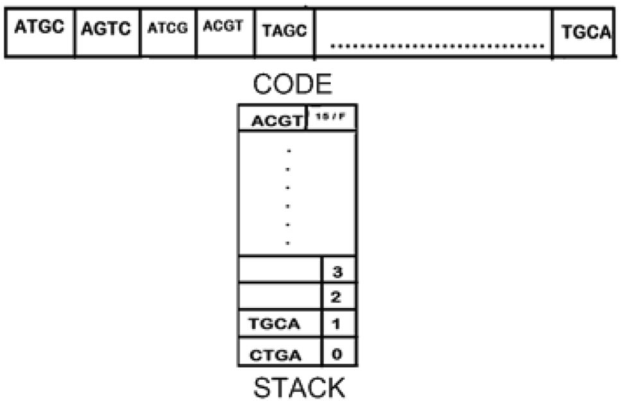


Fig. 1 CODE and STACK

Step 2: STACK of size 16 is declared. Using random function, generate numbers between 0 and 23. The number generated, the corresponding ATGC combination of that index in CODE would be placed at the top of the STACK as shown in Fig. 1.

Step 3: The pixel values of the image are fetched in matrix form.

Step 4: Finding the first seven most occurring colors from the matrix that are present in matrix.

Step 5: From the remaining eight combinations in CODE, seven are again taken randomly and each of the most occurring color is specially assigned a combination. The combinations being stored in an array MOST_OCCUR.

Step 6: The symbol '#' is given a code which gets left after assigning all the 23 combinations corresponding to 0 to F and the most occurring colors.

Step 7: The width and height of image are encoded with the codes corresponding to 0 to F in CODE and written in a file named CODE_FILE with code of '#' after width and height.

Step 8: The pixel values of the image (considering already in hexadecimal) in horizontal (X-axis) direction is picked and stored in a variable 'Hex'. Three cases can occur:

Case A: The color is one of the most occurring colors, then it is encoded with its corresponding code stored in MOST_OCCUR and written in the CODE_FILE.

Case B: The color is not among the most occurring seven colors. Then each character from 'Hex' will be extracted and encoded with the codes corresponding to its value (equal to index number, for 'A' to 'F' codes of 10–15 will be assigned) in CODE.

Case C: The color is same as that of previous color. Then a counter will count the number of times it is repeated. The final number will encoded with the codes corresponding to 0–9 in CODE and written in CODE_FILE with the code of '#' concatenated before and after it.
For example, suppose '1010F1' occurs 10 times serially. The above steps will get repeated till the end of the matrix containing the hexadecimal codes of the pixels of the image.

Step 9: The 7 most occurring colors will be treated as the normal colors and encoded similar to Case B of Step 8 and written in a separate file MOST_OCCUR_FILE.

3.2 Decryption

For decryption, both the files CODE_FILE and MOST_OCCUR_FILE are required. The key that will be required is content of CODE, MOST_OCCUR and code of '#' and stored in D_CODE (array of size 16), D_MOST_OCCUR (array of size 7) and HASH (variable). Once they are entered the decryption begins.

Step 1: The MOST_OCCUR_FILE file will be decoded first.Hexadecimal of the seven most occurring colors are stored in MOST_OCCUR_COLOR and their corresponding codes are stored in D_MOST_OCCUR.

```
while(!MOST_OCCUR_FILE end)
{
  for i=1 to 6
    {
      Extract 4 characters from MOST_OCCUR_FILE;
      Search in D_CODE for a match;
      String Hex=Store corresponding index (if index
      greater than 9 and less than 16, store 'A' to 'F'
      correspondingly);
      Store the string in an array-list MOST_OCCUR_COLOR;
    }
 }
```

Step 2: The width and height of image will be decoded from the CODE_FILE. Four characters will be extracted and search in D_CODE for a match and the corresponding index value will be stored in the integer variable width unless a match with HASH is encountered. Similarly till the next HASH, index will be stored in integer variable height.

Step 3: Decoding of CODE_FILE and producing the color matrix of the original image. D_COUNTER is a counter and Decode holds the decoded hexadecimal value of a color.

```
for i=0 to height
  for j=0 to width
    if(D_COUNTER==0)
      {
      for k=0 to 6
      {
      Extract 4 characters from CODE_FILE
      CASE A: if matches with the content of D_MOST_OCCUR
        Index=matching index of D_MOST_OCCUR
        String Decode=MOST_OCCUR_COLOR[Index];
        break;
      CASE B: if matches with the code of HASH
        Extract 4 characters, find a match in D_CODE and
        store the corresponding index in the integer vari-
        able D_COUNTER;
        Break when the next set of 4 characters matches
        HASH.
      CASE C: if neither most occurring nor HASH
        String Decode=store the matching index of D_CODE
        and if between 10 to 15, then store 'A' to 'F'
        correspondingly.
      }
      }
      else
      {
      While(D_COUNER!=0)
       {
        IMAGE [j][i]=Decode;
        j++;
        D_COUNTER--;
       }
      }
```

Step 4: The IMAGE matrix consisting of the hexadecimal codes for the original image will then be written to produce the original image. Hence the image is decrypted.

4 Result analysis

For the simulation purpose, we have used 'Lena.bmp' image of size 256 × 256 as shown in Fig. 2. Figures 3 and 4 show the MOST_OCCUR_FILE and CODE_FILE respectively.

The details of KEY (shown in Fig. 5) are:

CODE (encodes '0' to 'F' respectively)
TCAG TAGC GTAC GACT CATG CAGT GATC CTGA TCGA ATCG CTAG ATGC GTCA GCAT GCTA AGCT
MOST_OCCUR (encodes most occurring color)

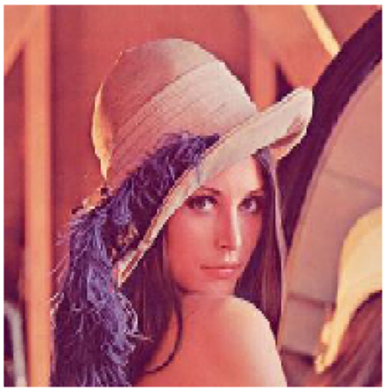


Fig. 2 Original image

CGAT ACTG CGTA TGAC AGTC TGCA TACG
HASH (code for '#')
ACGT

After entering the key details, the decrypted image is generated as shown in Fig. 6.

When the entered key is wrong (as shown in Fig. 7), we cannot get any image information.

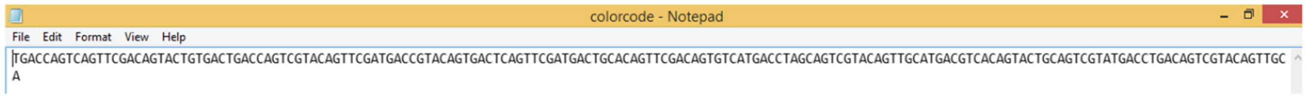
For this encryption, the image is not produced and compilation does not stop. Even the seven most occurring color is not correctly decoded when wrong key is entered. Table 1 shows the encryption and decryption time of images of different sizes. Similarly Table 2 shows the size of CODE_FILE and MOST_OCCUR_FILE of images of different sizes.

5 Performance analysis and validation of algorithm

In this paper, an effort has been made to transfer vital color images as a text file over the unsecured network and can be of immediate use to intelligence and defence units. A simple algorithm in the form of random key generation has been used and certain ideas like assigning special codes to most occurring color and encoding range of repeated colors has been incorporated to decrease the text file size and give a layer of security along with this DNA components A, T, G, C decreases the chance of recognizing the data as image. The formidableness of the algorithm is quite competitive.

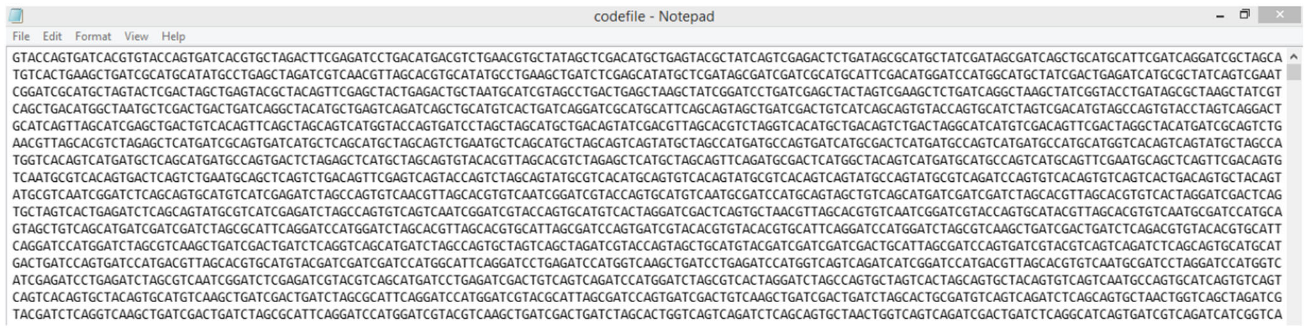
5.1 Key space analysis

The initial key is generated by assigning 16 out of 24 combinations of ATGC to 0–F. Hence the total possibility



```
TGACCAGTCAGTTCGACAGTACTGTGACTGACCAGTCGTACAGTTCGATGACCGTACAGTGACTCAGTTCGATGACTGCACAGTTCGACAGTGTCATGACCTAGCAGTCGTACAGTTGCATGACGTCACAGTACTGCAGTCGTATGACCTGACAGTCGTACAGTTGC
A
```

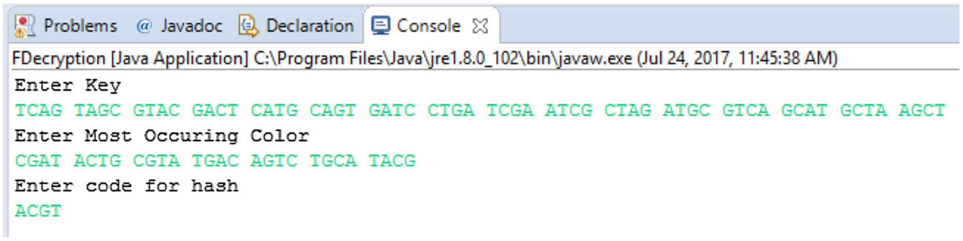
Fig. 3 MOST_OCCUR_FILE



```
GTACCAGTGATCACGTGTACCAGTGATCACGTGCTAGACTTCGAGATCCTGACATGACGTCTGAACGTGCTATAGCTCGACATGCTGAGTACGCTATCAGTCGAGACTCTGATAGCGCATGCTATCGATAGCGATCAGCTGCATGCATTCGATCAGGATCGCTAGCA
TGTCACTGAAGCTGATCGCATGCATATGCCTGAGCTAGATCGTCAACGTTAGCACGTGCATATGCCTGAAGCTGATCTCGAGCATATGCTCGATAGCGATCGATCGCATGCATTCGACATGGATCCATGGCATGCTATCGACTGAGATCATGCGCTATCAGTCGAAT
CGGATCGCATGCTAGTACTCGACTAGCTGAGTACGCTACAGTTCGAGCTACTGAGACTGCTAATGCATCGTAGCCTGACTGAGCTAAGCTATCGGATCCTGATCGAGCTACTAGTCGAAGCTCTGATCAGGCTAAGCTATCGGTACCTGATAGCGCTAAGCTATCGT
CAGCTGACATGGCTAATGCTCGACTGACTGATCAGGCTACATGCTGAGTCAGATCAGCTGCATGTCACTGATCAGGATCGCATGCATTCAGCAGTAGCTGATCGACTGTCATCAGCAGTGTACCAGTGCATCTAGTCGACATGTAGCCAGTGTACCTAGTCAGGACT
GCATCAGTTAGCATCGAGCTGACTGTCACAGTTCAGCTAGCAGTCATGGTACCAGTGATCCTAGCTAGCATGCTGACAGTATCGACGTTAGCACGTCTAGGTCACATGCTGACAGTCTGACTAGGCATCATGTCGACAGTTCGACTAGGCTACATGATCGCAGTCTG
AACGTTAGCACGTCTAGAGCTCATGATCGCAGTGATCATGCTCAGCATGCTAGCAGTCTGAATGCTCAGCATGCTAGCAGTCAGTATGCTAGCCATGATGCCAGTGATCATGCGACTCATGATGCCAGTCATGATGCCATGCATGGTCACAGTCAGTATGCTAGCCA
TGGTCACAGTCATGATGCTCAGCATGATGCCAGTGACTCTAGAGCTCATGCTAGCAGTGTACACGTTAGCACGTCTAGAGCTCATGCTAGCAGTTCAGATGCGACTCATGGCTACAGTCATGATGCATGCCAGTCATGCAGTTCGAATGCAGCTCAGTTCGACAGTG
TCAATGCGTCACAGTGACTCAGTCTGAATGCAGCTCAGTCTGACAGTTCGAGTCAGTACCAGTCTAGCAGTATGCGTCACATGCAGTGTCACAGTATGCGTCACAGTCAGTATGCCAGTATGCGTCAGATCCAGTGTCACAGTGTCAGTCACTGACAGTGCTACAGT
ATGCGTCAATCGGATCTCAGCAGTGCATGTCATCGAGATCTAGCCAGTGTCAACGTTAGCACGTGTCAATCGGATCGTACCAGTGCATGTCAATGCGATCCATGCAGTAGCTGTCAGCATGATCGATCGATCTAGCACGTTAGCACGTGTCACTAGGATCGACTCAG
TGCTAGTCACTGAGATCTCAGCAGTATGCGTCATCGAGATCTAGCCAGTGTCAGTCAATCGGATCGTACCAGTGCATGTCACTAGGATCGACTCAGTGCTAACGTTAGCACGTGTCAATCGGATCGTACCAGTGCATACGTTAGCACGTGTCAATGCGATCCATGCA
GTAGCTGTCAGCATGATCGATCGATCTAGCGCATTCAGGATCCATGGATCTAGCACGTTAGCACGTGCATTAGCGATCCAGTGATCGTACACGTGTACACGTGCATTCAGGATCCATGGATCTAGCGTCAAGCTGATCGACTGATCTCAGACGTGTACACGTGCATT
CAGGATCCATGGATCTAGCGTCAAGCTGATCGACTGATCTCAGGTCAGCATGATCTAGCCAGTGCTAGTCAGCTAGATCGTACCAGTAGCTGCATGTACGATCGATCGATCGACTGCATTAGCGATCCAGTGATCGTACGTCAGTCAGATCTCAGCAGTGCATGCAT
GACTGATCCAGTGATCCATGACGTTAGCACGTGCATGTACGATCGATCGATCCATGGCATTCAGGATCCTGAGATCCATGGTCAAGCTGATCCTGAGATCCATGGTCAGTCAGATCATCGGATCCATGACGTTAGCACGTGTCAATGCGATCCTAGGATCCATGGTC
ATCGAGATCCTGAGATCTAGCGTCAATCGGATCTCGAGATCGTACGTCAGCATGATCCTGAGATCGACTGTCAGTCAGATCCATGGATCTAGCGTCACTAGGATCTAGCCAGTGCTAGTCACTAGCAGTGCTACAGTGTCAGTCAATGCCAGTGCATCAGTGTCAGT
CAGTCACAGTGCTACAGTGCATGTCAAGCTGATCGACTGATCTAGCGCATTCAGGATCCATGGATCGTACGCATTAGCGATCCAGTGATCGACTGTCAAGCTGATCGACTGATCTAGCACTGCGATGTCAGTCAGATCTCAGCAGTGCTAACTGGTCAGCTAGATCG
TACGATCTCAGGTCAAGCTGATCGACTGATCTAGCGCATTCAGGATCCATGGATCGTACGTCAAGCTGATCGACTGATCTAGCACTGGTCAGTCAGATCTCAGCAGTGCTAACTGGTCAGTCAGATCGACTGATCTCAGGCATCAGTGATCGTCAGATCATCGGTCA
```

Fig. 4 CODE_FILE

Fig. 5 Key details



```
FDecryption [Java Application] C:\Program Files\Java\jre1.8.0_102\bin\javaw.exe (Jul 24, 2017, 11:45:38 AM)
Enter Key
TCAG TAGC GTAC GACT CATG CAGT GATC CTGA TCGA ATCG CTAG ATGC GTCA GCAT GCTA AGCT
Enter Most Occuring Color
CGAT ACTG CGTA TGAC AGTC TGCA TACG
Enter code for hash
ACGT
```

is ${}^{24}C_{16} \times 16!$ and for per possibility of this there exists ${}^{8}C_{7} \times 7!$ possibility to assign ATGC combinations to the most occurring colors. Ultimately the total key space results to ${}^{24}P_{16} \times {}^{8}P_{7}$ approximating to $10^{23.}$ With such huge key space the reliability and effectiveness of the algorithm is established as the KEY associated is quite unpredictable and resistant against brute force attacks.

5.2 Key security analysis

Key security plays the major role in protecting information in the presence of adversaries, and allows for secure communication between two or more parties. Here the Key Space is not only large and complex to predict the encryption algorithm takes care of the security purposes through its "INTERLINKED PROTECTION PROTOCOL" scheme. Instead of depending just upon the "KEY" of the encryption, it adds another level of security by adding up the necessity to fetch the encrypted file containing the most occuring color. As a result, it became infeasible to recover the encrypted image without adequate knowledge about both the transmitted 'KEY' and the 'MOST_OCCUR_FILE'. Without anyone of them it is increasingly difficult to break the encryption. Hence forth "INTERLINKED PROTECTION PROTOCOL" protects in the most efficient way.

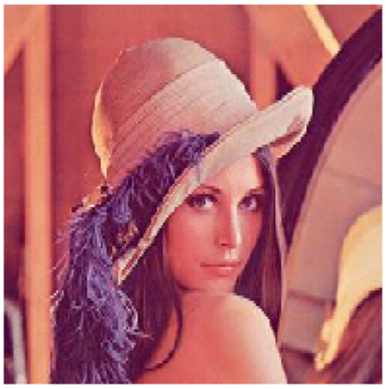


Fig. 6 Decrypted image

5.3 Encryption quality

5.3.1 Structural content (SC)

It is defined as the summation of original image pixel square value divided by the summation of decrypted image pixel square value. It also finds the correlation between images. If there is no distortion in decrypted images then SC will be 1.

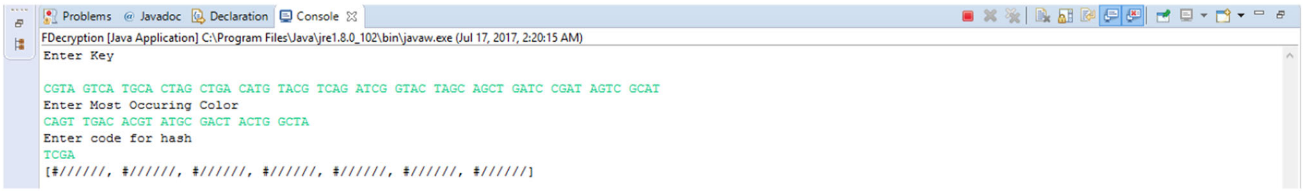
$$SC = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} f(i,j)^2}{\sum_{i=1}^{M} \sum_{j=1}^{N} \bar{f}(i,j)^2} \tag{1}$$

5.3.2 Normalized absolute error (NAE)

It is defined as the summation of difference between original image f(i, j) and the decrypted image f′(i, j) divided by summation of original image pixel square value.

Table 1 Encryption and decryption time

Image size	Encryption time (s)	Decryption time (s)
128 × 128	3.908	0.328
256 × 256	17.498	0.578
512 × 512	2.363	1.206
1024 × 1024	1115.36	4.491



```
FDecryption [Java Application] C:\Program Files\Java\jre1.8.0_102\bin\javaw.exe (Jul 17, 2017, 2:20:15 AM)
Enter Key

CGTA GTCA TGCA CTAG CTGA CATG TACG TCAG ATCG GTAC TAGC AGCT GATC CGAT AGTC GCAT
Enter Most Occuring Color
CAGT TGAC ACGT ATGC GACT ACTG GCTA
Enter code for hash
TCGA
[#//////, #//////, #//////, #//////, #//////, #//////, #//////]
```

Fig. 7 Wrong key details

Table 2 CODE_FILE and MOST_OCCUR_FILE size

Image size	CODE_FILE size	MOST_OCCUR_FILE size
128 × 128	383 Kb	1 Kb
256 × 256	1.5 Mb	1 Kb
512 × 512	5.1 Mb	1 Kb
1024 × 1024	23.94 Mb	1 Kb

Table 3 Result of different encryption quality parameters

SC	NAE	MSE	PSNR
1.00035	0.00221	0.00415	74.94991

$$NAE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}|f(i,j) - f'(i,j)|}{\sum_{i=1}^{M}\sum_{j=1}^{N}f(i,j)^2} \quad (2)$$

5.3.3 Peak signal to noise ratio (PSNR)

It is defined as the ratio of the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation [4]. Mathematically PSNR can be defined as:

$$PSNR = 10log\frac{(2^n - 1)^2}{MSE} \quad (3)$$

where 2^n represents the number of pixels in a row or column of image and MSE is

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - f'(i,j)]^2}{MN} \quad (4)$$

where M and N represents the height and width of image respectively. f(i, j) is the (i, j)th pixel value of the original image and f′(i, j) is the (i, j)th pixel value of decrypted image.

Table 3 shows the result of different encryption quality parameters on the lena image of size 256 × 256. Most of the current work in Image encryption using DNA based cryptography converts the original image to encrypted image which is prone to noise and there can be considerable difference between the original image and decrypted image depending on the intermediate encryption algorithm used. The algorithm suggested here converts the original image to text file increasing the difficulty for image processing based reverse engineering. From the result in Table 3 we find that PSNR value is quite high which indicates that the encryption is very less affected by noise and the SC value is approximately 1 denoting that there is very less difference between the original image and the decrypted image. Hence it can be inferred that the algorithm suggested is quite competitive and reliable again when compared to result shown in [7]. Furthermore when compared with the parameters of different algorithms like Hill Cipher, Affine Hill Cipher, TF Hill Cipher, RD Hill as given in Table 2 of [12] it can be infered that the proposed algorithm in the paper made considerable improvement in the encryption quality.

6 Conclusion

In this paper, an effort has been made to transfer vital color images as a text file over the unsecured network. A simple algorithm in the form of random key generation has been used and certain ideas like assigning special codes to most occurring color and encoding range of repeated colors has been incorporated to decrease the text file size and give a layer of security along with this DNA components A, T, G, C decreases the chance of recognizing the data as image. The formidableness of the algorithm is quite competitive.

References

1. Sakthidasan Sankaran K, Santosh Krishna BV (2011) A new chaotic algorithm for image encryption and decryption of digital color images. Int J Inf Educ Technol 1(2):137–141
2. Xiao G, Lu M, Qin L, Lai X (2006) New field of cryptography: DNA cryptography. Chin Sci Bull 51(12):1413–1420
3. Tornea O, Borda ME (2009) DNA Cryptographic Algorithms. In: Vlad S, Ciupa RV, Nicu AI (eds) International conference on advancements of medicine and health care through technology. IFMBE Proceedings, vol 26. Springer, Berlin, pp 223–226
4. Saha BJ, Pradhan C, Kabi KK, Bisoi AK (2014) Robust watermarking technique using Arnold's transformation and RSA in discrete wavelets. In: IEEE international conference on information systems and computer networks (ISCON 2014), 1–2 Mar 2014, Mathura, India, pp 83–87
5. Puech W, Erkin Z, Barni M, Rane S, Lagendijk RL (2013) Emerging cryptographic challenges in image and video processing. In: 2012 19th IEEE International Conference on Image Processing ICIP, USA, pp 2629–2632
6. Saha BJ, Kabi KK, Pradhan C (2014) Non blind watermarking technique using enhanced one time pad in DWT domain. In: International conference on computing communications and networking technologies, Hefei
7. Akkasaligar PT, Biradar S (2017) Secure medical image encryption based on intensity level using chaos theory and DNA cryptography. In: International conference on computational intelligence and computing research, IEEE, Chennai
8. Niyat AY, Hei RM, Jahan MV (2016) Chaos-based image encryption using a hybrid cellular automata and a DNA sequence. In: 2015 International congress on technology, communication and knowledge (ICTCK), IEEE, 11–12 November 2015, Mashhad, Iran
9. Ochani A, Jadhav D, Gulwani R (2017) DNA Image encryption using modified symmetric key (MSK). In: International conference on inventive computation technologies, IEEE, Coimbatore, pp 1–4

10. Soni R, Johar A, Soni V (2013) An encryption and decryption algorithm for image based on DNA. In: International conference on communication systems and network technologies (CSNT 2013), 6–8 April 2013, Gwalior, India, pp 478–481
11. Mokhtar MA, Gobran SN, El-Badawy ESA (2014) Colored image encryption algorithm using DNA code and Chao theory. In: 2014 International conference on computer and communication engineering (ICCCE), Kuala Lumpur, 23–25 September 2014, pp 12–15
12. Jangid RK, Mohmmad N, Didel A, Taterh S (2014) Hybrid approach of image encryption using DNA cryptography and TF Hill Cipher algorithm. In: International conference on communication and signal processing