Two Fibonacci P-code Based Image Scrambling Algorithms

Yicong Zhou*[a], Sos Agaian[b], Valencia M. Joyner[a], Karen Panetta[a]
[a] Dept. of Electrical and Computer Engineering, Tufts University, Medford, MA 02155
[b] Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249

ABSTRACT

Image scrambling is used to make images visually unrecognizable such that unauthorized users have difficulty decoding the scrambled image to access the original image. This article presents two new image scrambling algorithms based on Fibonacci p-code, a parametric sequence. The first algorithm works in spatial domain and the second in frequency domain (including JPEG domain). A parameter, p, is used as a security-key and has many possible choices to guarantee the high security of the scrambled images. The presented algorithms can be implemented for encoding/decoding both in full and partial image scrambling, and can be used in real-time applications, such as image data hiding and encryption. Examples of image scrambling are provided. Computer simulations are shown to demonstrate that the presented methods also have good performance in common image attacks such as cutting (data loss), compression and noise. The new scrambling methods can be implemented on grey level images and 3-color components in color images. A new Lucas p-code is also introduced. The scrambling images based on Fibonacci p-code are also compared to the scrambling results of classic Fibonacci number and Lucas p-code. This will demonstrate that the classical Fibonacci number is a special sequence of Fibonacci p-code and show the different scrambling results of Fibonacci p-code and Lucas p-code.

Keywords: image scrambling, Fibonacci p-code, Lucas p-code.

1. INTRODUCTION

With the current development of ubiquitous wireless network technology and digital multimedia devices, wireless image/video data transmission is becoming more prevalent. As a result, information security becomes a key problem for consumers, companies and governments. Security of image and video data is very important in many areas, such as video-on-demand, confidential remote video conferencing, security communication, and also in military applications. Image scrambling (i.e., encryption) technologies are very useful tools to ensure image security by transforming the image into an unintelligible image[1]. Scrambling makes the image unrecognizable to prevent eavesdroppers from decoding the true form or meaning of the image using the human visual system or a computer system.

Several promising image scrambling methods have been presented in the literature. Ville et al. [1] presented a scrambling scheme without bandwidth expansion using two-dimensional discrete prolate spheroidal sequences in spatial domain. Zou et al. [2] used classical Fibonacci number to scramble image in spatial domain. Gu et al. [3] performed a chaos scrambling method in wavelet domain. However, these methods fail to include a security key for both image scrambling and image reconstruction.

In this article, we present two new image scrambling algorithms based on Fibonacci p-code. One is working in spatial domain, the other is for frequency domain (including JPEG domain). The security keys of our image scrambling algorithms are parameters *p* and *i*, and the size of original image. There are many possible choices for security keys so that the scrambled image is difficult to decrypt by unauthorized users, and thus, greater security is guaranteed.

A new Lucas p-code is also introduced. The scrambling images obtained from Fibonacci p-code are compared to the scrambling results of classic Fibonacci number and Lucas p-code. This will demonstrate that the classical Fibonacci number is a special sequence of Fibonacci p-code when p=1. Additionally, this will show the difference of scrambling results by using the Fibonacci p-code and Lucas p-code.

* Yicong.Zhou@tufts.edu; phone 1 617 627-5183; fax 1 617-627-3220;

Image Processing: Algorithms and Systems VI, edited by Jaakko T. Astola,
Karen O. Egiazarian, Edward R. Dougherty, Proc. of SPIE-IS&T Electronic Imaging,
SPIE Vol. 6812, 681215, © 2008 SPIE-IS&T · 0277-786X/08/$18

The presented algorithms have good performance in common image attacks such as cutting (data loss), noise and compression. They can be implemented for encoding/decoding both in full and partial image scrambling in grayscale and color images, and can be used in real-time applications, such as image data hiding and encryption.

2. P-Fibonacci and P-Lucas Transform

Fibonacci p-code and a new Lucas p-code are introduced in this section. A new 1-D transform and a new 2-D transform are generated for both Fibonacci p-code and Lucas p-code. The inverse 2-D transform used for recovering the original image is also presented.

2.1. Fibonacci p-code and Lucas p-code

Definition 2.1: The Fibonacci p-code[4, 5] is a sequence defined by,

$$F_p(n) = \begin{cases} 0 & n < 1 \\ 1 & n = 1 \\ F(n-1) + F(n-p-1) & n > 1 \end{cases} \tag{1}$$

where p is a nonnegative integer.

From the definition above, Fibonacci p-code sequences will differ based on the p value. Specially,

(1) Binary sequence: p=0, the sequence is powers of two, 1, 2, 4, 8, 16…;

(2) Classical Fibonacci sequence: p=1, the sequence is 1, 1, 2, 3, 5, 8, 13, 21…;

(3) For the large values of p the sequence starts with consecutive 1's and immediately after that 1, 2, 3, 4….p…

Sample sequences are shown in Table 2.1.

Table 2.1 Fibonacci p-code sequence with different p value

P \ n	1	2	3	4	5	6	7	8	9	10	11	…
0	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	…
1	**1**	**1**	**2**	**3**	**5**	**8**	**13**	**21**	**34**	**55**	**89**	…
2	**1**	**1**	**1**	**2**	**3**	**4**	**6**	**9**	**13**	**19**	**28**	…
3	**1**	**1**	**1**	**1**	**2**	**3**	**4**	**5**	**7**	**10**	**14**	…
4	**1**	**1**	**1**	**1**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	…
…												
∞	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	…

Similarly, we introduce the Lucas P-code.

Definition 2.2: The Lucas p-code is defined as:

$$L_p(n) = \begin{cases} 0 & n < 1 \\ 2 & n = 1 \\ 1 & n = 2 \\ L(n-1) + L(n-p-1) & n > 2 \end{cases} \tag{2}$$

where p is a nonnegative integer.

The Lucas P-code sequence will be different according to the p value. The detail results are shown in Table 2.2.

Table 2.2 Lucas p-code sequence with different p value

P \ n	1	2	3	4	5	6	7	8	9	10	11	…
0	**2**	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	…
1	**2**	**1**	**3**	**4**	**7**	**11**	**18**	**29**	**47**	**76**	**123**	…
2	**2**	**1**	**1**	**3**	**4**	**5**	**8**	**12**	**17**	**25**	**37**	…
3	**2**	**1**	**1**	**1**	**3**	**4**	**5**	**6**	**9**	**13**	**18**	…
4	**2**	**1**	**1**	**1**	**1**	**3**	**4**	**5**	**6**	**7**	**10**	…
…												
∞	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	…

2.2. P-Fibonacci/Lucas Transform

Definition 2.3: Let $F_p(n)$ and $F_p(n+1)$ are two consecutive Fibonacci p-code (or Lucas p-code) elements. The permutation $\{T_1, T_2, T_3, ..., T_{F_p(n+1)-1}\}$ of an input sequence $\{1_1, 2, 3, ..., F_p(n+1)-1\}$ is called 1-D P-Fibonacci (or P-Lucas) Transform if $\{T_1, T_2, T_3, ..., T_{F_p(n+1)-1}\}$ is defined by

$$T_k = k[F_p(n)+i] \bmod F_p(n+1) \tag{3}$$

Where $k = 0,1,...,F_p(n+1)-1$; $i = -3,-2,-1,0,1,2,3$; $F_p(n)+i < F_p(n+1)$.

For example, for a MxN grayscale image the data is a 2D matrix A,

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1N} \\ a_{21} & a_{22} & ... & a_{2N} \\ ... & ... & ... & ... \\ a_{M1} & a_{M2} & ... & a_{MN} \end{pmatrix} \tag{4}$$

The column coefficient matrix can be generated by using equation (3) based on different p values. There are N columns in this image. The input sequence is $k = 1,2,3,...,N$, thus $N = F_p(n+1)-1$.

For a certain p value, the output sequence $\{T(N)\}$ should be the permutation of the input sequence $\{1,2,3,...,N\}$ which is defined by

$$T_{pN} = (T_{p1}, T_{p2}, T_{p3}, ..., T_{pN}) \tag{5}$$

From T_{pN}, the column coefficient matrix of 2-D P-Fibonacci Transform $T_c(N,N)$ can be generated as

$$T_c(i,j) = \begin{cases} 1 & (T_{pi}, i) \\ 0 & otherwise \end{cases} \tag{6}$$

As the same procedure, there are M rows in the image data matrix. For the certain p value, the output sequence should be the permutation of the input sequence $k = 1, 2, 3, ..., M$, after transformation by

$$T_{pM} = (T_{p1}, T_{p2}, T_{p3}, ..., T_{pM}) \tag{7}$$

The row coefficient matrix of 2-D P-Fibonacci Transform $T_r(M, M)$ can be generated as

$$T_r(i, j) = \begin{cases} 1 & (i, T_{pi}) \\ 0 & otherwise \end{cases} \tag{8}$$

Table 2.3 coefficient matrices of P-Fibonacci transform with different p value for an 8x10 image.

p	*i* (row, column)*	T_c	T_r
1	(0, 0)	$\begin{pmatrix} 0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0 \end{pmatrix}$
8	(-1, -3)	$\begin{pmatrix} 0&0&0&0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \end{pmatrix}$

* (row, column) stands for the *i* value for column and row transformation separately.

The Table 2.3 shows sample coefficient matrices of a P-Fibonacci transform for an 8x10 image with different p values. The coefficient matrices of the 2-D P-Fibonacci Transform are generated based on the parameters: p and i.

Definition 2.4: Let B be the original image matrix, Tr, the row coefficient matrix, and Tc, the column coefficient matrix. The following matrix is called the 2-D P-Fibonacci Transform:

$$S = T_r B T_c \tag{9}$$

where S is the scrambled image matrix with dimensions $M \times N$.

Definition 2.5: Let S be the scrambled image matrix, Tr the row coefficient matrix, and Tc the column coefficient matrix. The following matrix is called the 2-D Inverse Fibonacci Transform:

$$R = T_r^{-1} S T_c^{-1} \tag{10}$$

where R is the reconstructed image matrix.

3. Image scrambling algorithm in the spatial domain

3.1. Image scrambling algorithm in the spatial domain

The presented image scrambling algorithm in the spatial domain (shown in Figure 1) is designed to change the image pixel position using the 2-D P-Fibonacci Transform. Color images have three color components and the scrambling algorithm is applied to each color component individually. Grayscale images are treated as color images with one component. The presented algorithm is a lossless image scrambling method.

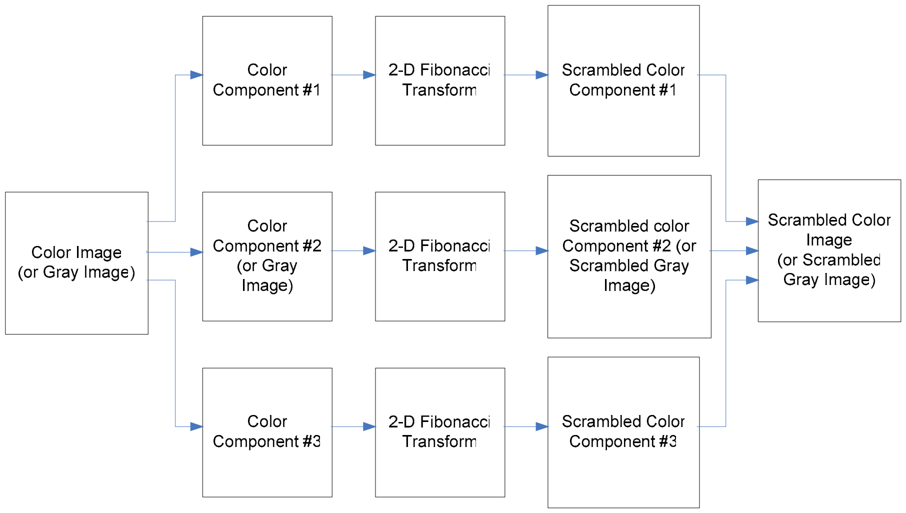
Color Image (or Gray Image)
Color Component #1
Color Component #2 (or Gray Image)
Color Component #3
2-D Fibonacci Transform
2-D Fibonacci Transform
2-D Fibonacci Transform
Scrambled Color Component #1
Scrambled color Component #2 (or Scrambled Gray Image)
Scrambled Color Component #3
Scrambled Color Image (or Scrambled Gray Image)

Fig. 1: Block diagram of the image scrambling algorithm in the spatial domain.

Algorithm	*Image scrambling in the spatial domain*
Input	Color image (or Grayscale image) to be scrambled
Step 1	Choose key parameter, P, Calculate the row and column coefficient matrices of 2-D P-Fibonacci Transform.
Step 2	Separate the color image into three color components (for example R G B, Y Cb Cr, etc.). Each component is a 2D matrix (Grayscale image: skip this step.)
Step 3	Apply 2-D P-Fibonacci Transform to each color component to get the scrambled color components. (Grayscale image: Apply the 2-D P-Fibonacci Transform to the grayscale image to get the scrambled grayscale image.)
Step 4	Recombine the three scrambled color components to get the scrambled color image. (Grayscale image: Skip this step.)
Output	Scrambled color image (or scrambled grayscale image)

3.2. Image unscrambling algorithm in the spatial domain

For authorized users to reconstruct the original image, the following security keys are required: p and i. The inverse row and column coefficient matrices of the 2-D Inverse P-Fibonacci Transform can be generated based on these keys. The original image can be reconstructed by applying the 2-D Inverse P-Fibonacci Transform to the scrambled image.

Algorithm	***Image unscrambling in the spatial domain***
Input	Scrambled color image (or scrambled grayscale image)
Step 1	Calculate the inverse row and column coefficient matrices of 2-D Inverse P-Fibonacci Transform using security keys: *p and i*.
Step 2	Separate the scrambled color image to three color components. (Grayscale image: Skip this step.)
Step 3	Apply the 2-D Inverse Fibonacci Transform to the each color component of the scrambled image separately. (Grayscale image: Apply the 2-D Inverse Fibonacci Transform directly to get the reconstructed image.)
Step 4	Recombined the three color components together to get the reconstructed color image. (Grayscale image: skip this step.)
Output	Reconstructed color image (or grayscale image)

4. Image scrambling algorithm in the frequency domain

4.1. Scrambling algorithm in the frequency domain

The image is converted to the frequency domain by applying the Discrete Cosine Transform (DCT). The presented scrambling algorithm (shown in Figure 2) disorders the sign of the DCT coefficients using the 2-D P-Fibonacci Transform. The coefficient matrices of the 2-D P-Fibonacci Transform can be generated based on parameters: *p* and *i*. Thus, the presented algorithm is a lossless image scrambling approach.

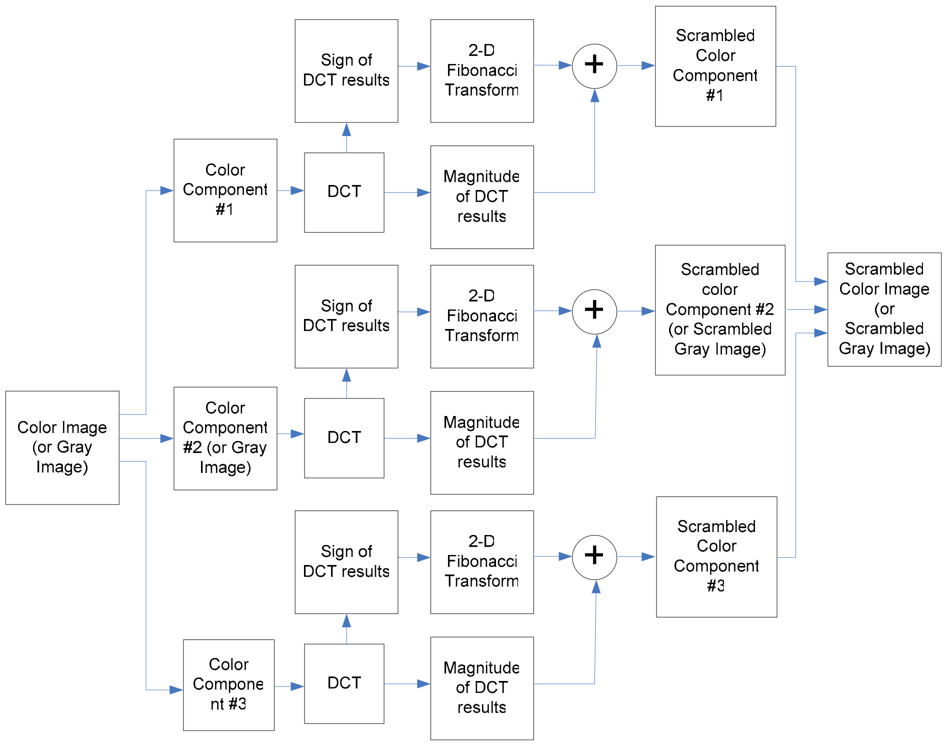

Fig. 2: Block diagram of the image scrambling algorithm in the frequency domain.

Algorithm	***Image scrambling in the frequency domain***
Input	Color image (or grayscale image) to be scrambled
Step 1	Choose key parameter, *P*, Calculate the row and column coefficient matrices of 2-D P-Fibonacci Transform.
Step 2	Convert the color image to the YCbCr format and separate the color image into three color components: Y, Cb, Cr. Each component is a 2-D matrix. (Grayscale image: Skip this step.)
Step 3	Apply DCT to each color component to transform them to frequency domain (Grayscale image: Apply DCT directly.).
Step 4	Separate the DCT result of each color component to sign and magnitude matrices. (Grayscale image: Separate the image DCT result to sign and magnitude matrices directly.).
Step5	Apply the 2-D P-Fibonacci Transform to the DCT sign matrix of each color component, respectively, to get the scrambled sign matrix. (Grayscale image: Apply the 2-D P-Fibonacci Transform to the DCT sign matrix directly to get the scrambled sign matrix.)
Step 6	Recombining the magnitude matrix of each color component and its scrambled sign matrix together to get the scrambled color image. (Grayscale image: Recombine the scrambled sign DCT matrix and its DCT magnitude matrix to get the scrambled image.)
Output	Scrambled color image (or scrambled grayscale image)

4.2. Unscrambling algorithm in the frequency domain

For authorized users to reconstruct the original image, they are given the security keys: p and i. The inverse row and column coefficient matrices of 2-D Inverse P-Fibonacci Transform can be generated based on these keys. The original image can be recovered by applying the unscrambling algorithm.

Algorithm	***Image unscrambling in the frequency domain***
Input	Scrambled color image (or scrambled gray image)
Step 1	Calculate the inverse row and column coefficient matrices of the 2-D Inverse P-Fibonacci Transform using security keys: p and i.
Step 2	Separate the scrambled color image to three color components and then separate each color component to the sign matrix and the magnitude matrix. (Grayscale image: Separate the scrambled gray image to the sign matrix and magnitude matrix.)
Step 3	Apply the 2-D Inverse P-Fibonacci Transform to the sign matrix of each color component to get the reconstructed sign matrices for the three color components. (Grayscale image: Apply the 2-D Inverse P-Fibonacci Transform to the sign matrix to get the reconstructed sign matrix.)
Step 4	Recombine the reconstructed sign matrices with their magnitude matrices for the three color components. (Grayscale image: Recombine the reconstructed sign matrix with its magnitude matrix)
Step 5	Apply the inverse DCT for the results of step 4 to get three reconstructed color components. (Grayscale image: Apply the inverse DCT for results of step 4 to reconstruct original gray image.)
Step 6	Recombine the three color components together to get the original YCbCr image, and convert it to another color format if needed. (Grayscale image: Skip this step.)
output	Reconstructed color image (or grayscale image)

5. Experiment results and comparison

To verify the performance of the presented scrambling algorithms, we implemented the algorithm on several (color and grayscale) images in Matlab and the results are presented in this section. This algorithm shows good performance when image is subjected to common image attacks.

5.1. Experiment results for image scrambling in spatial domain

A color image (Figure 3(a)) is scrambled (Figure 3(b)) by the presented algorithm using the security keys p=5 and i=2 in the spatial domain, and then reconstructed. The reconstructed image (Figure 3(c)) is the same as the original image according to the histogram (Figure 3(f)) of the difference between them. The result also verifies that our presented algorithm is a lossless scrambling method.

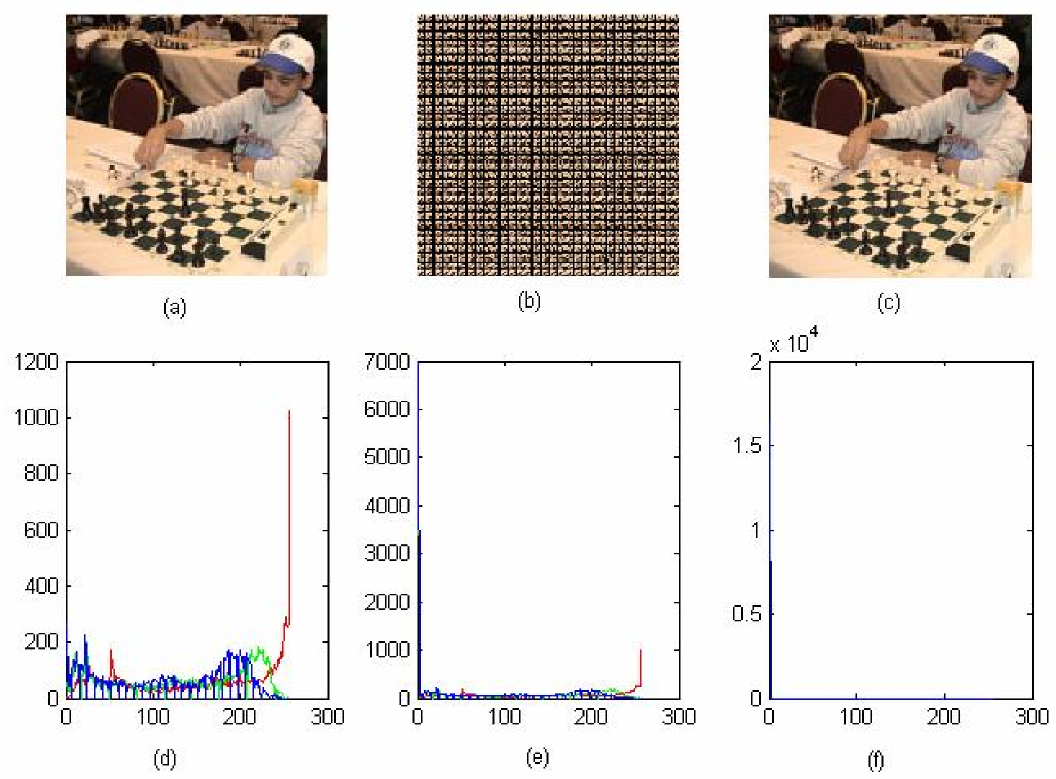

Fig. 3: Color image scrambling and histogram. (a) Original color image; (b) scrambled image (p=5,i=2); (c) reconstructed color image; (d) Histogram of original color image; (e)Histogram of scrambled image; (f) Histogram of the difference between reconstructed color image and original color image.

Figure 4 presents results of the presented algorithm in the spatial domain with different p and i parameters. The original image is a chess player image shown in Figure 3(a). The scrambled result is different with the different p and i value.

5.2. Experiment results for image scrambling in frequency domain

The image scrambling in frequency domain is based on Discrete Cosine Transform (DCT). The gray Lena image is converted to the frequency domain by applying DCT and then scrambled by using the presented scrambling algorithm with security keys p=2 and i=1, shown in Figure 5(b). The reconstructed image (Figure 5(c)) is almost the same as the original Lena image based on the histogram of the difference between the original and reconstructed image (5(f)). The scrambling process is lossless.

For color image scrambling in the frequency domain, a chess player image (Figure 6(a)) is converted to YCbCr format (Figure 6(b)) and transferred to the frequency domain by applying DCT (Figure 6(c)). It is scrambled based on the presented algorithm with p=4 and i=1, shown in Figure 6 (d). The reconstructed image (Figure 6(f)) is almost the same as the original image.

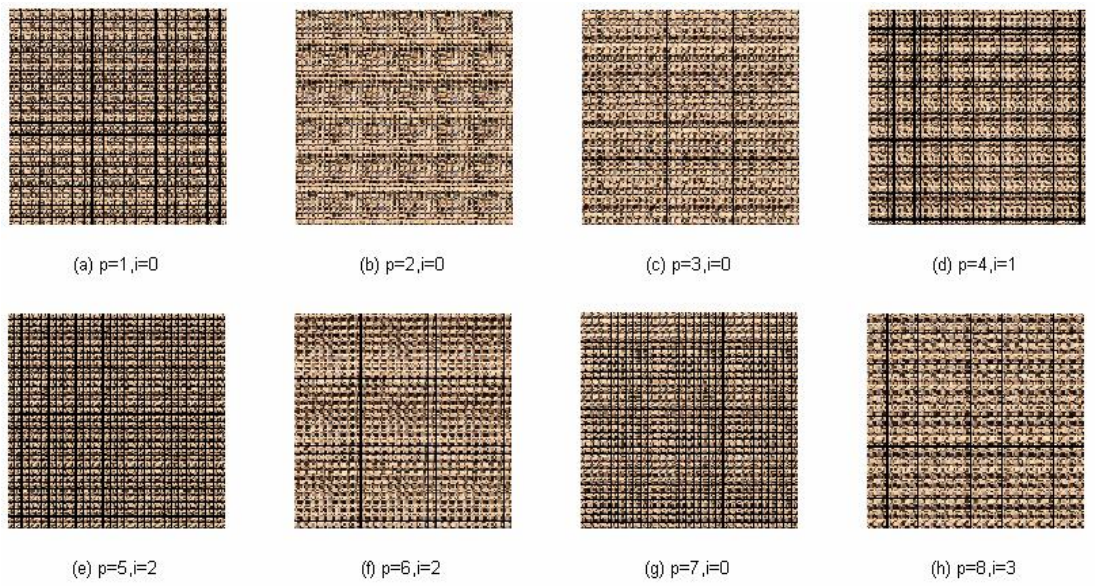

Fig. 4: Scrambled color image with different p and i value in spatial domain.

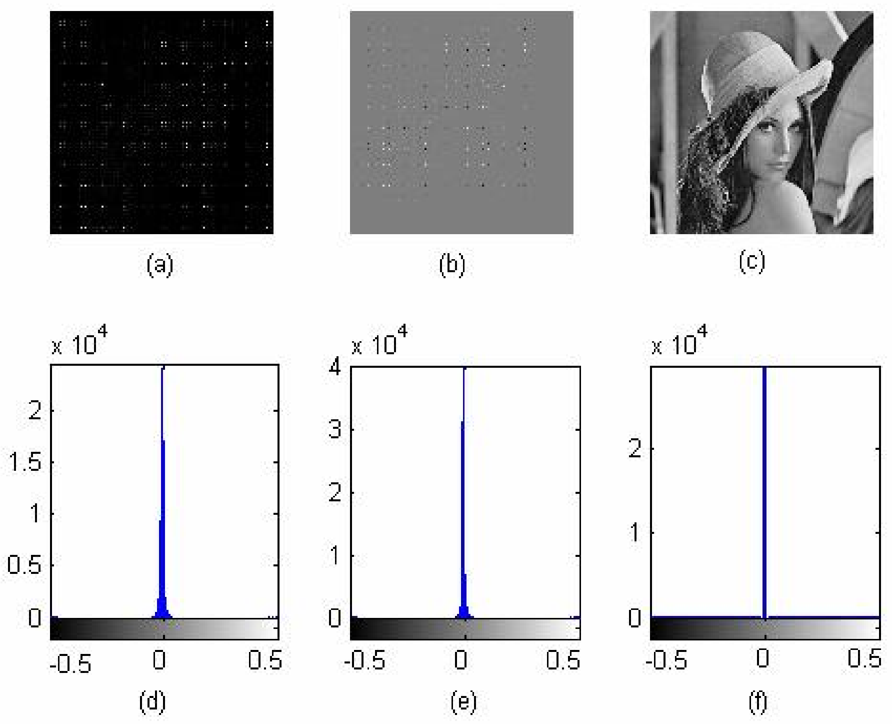

Fig. 5: Gray image scrambling and histogram. (a) DCT of original gray image; (b) scrambled DCT image (p=2,i=1); (c) reconstructed gray image; (d) Histogram of DCT image; (e)Histogram of scrambled DCT image; (f) Histogram of the difference between reconstructed image and original image.

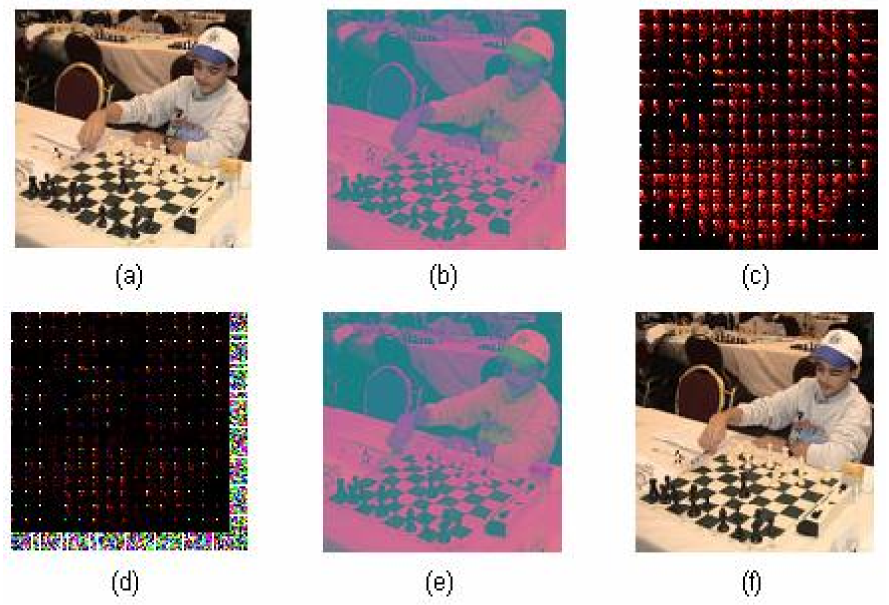

Fig. 6: Color image scrambling and histogram. (a) Original RGB color image; (b) Original YCbCr image;(c) DCT of YCbCr image (d) Scrambled DCT image (p=4,i=1); (e) reconstructed YCbCr color image; (f) reconstructed RGB color image.

The reconstructed images in the frequency domain are shown in Figure 7 after common image attacks. The results show that the presented algorithm has good performance in image attacks.

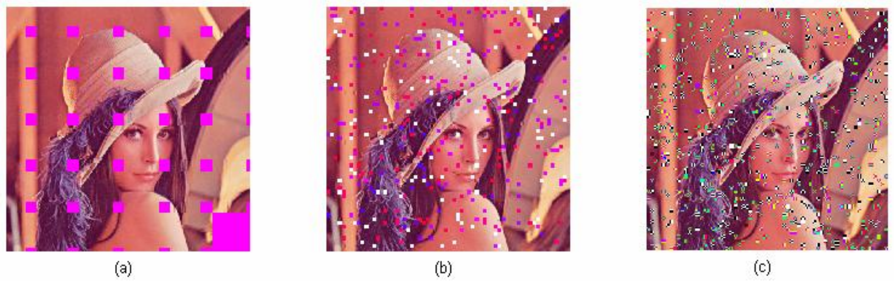

Fig. 7: Reconstructed color images after common attacks in the frequency domain. (a) Reconstructed color image after 1/16 right-bottom cutting attack; (b) Reconstructed color image after 0.1% Gaussian noise attack; (c) Reconstructed color image after 0.1% Salt&pepper noise attack.

5.3. Comparison of experiment results

We compare the scrambled results by different sequences in this section. Figure 8 shows the scrambled Lena images using classic Fibonacci number (Figure 8(a)), Fibonacci p-code (Figure 8(b)), and Lucas p-code (Figure 8(c)). The scrambled result of classic Fibonacci number is the same as that of Fibonacci p-code based on the scrambled image (a) and image (b), and also their histograms (d) and (e). The result proves that the classic Fibonacci number is a special Fibonacci p-code when p=1. The scrambled result of Lucas p-code (Figure 8(c)) is different from the results of classic Fibonacci number and Fibonacci p-code because the Lucas p-code and Fibonacci p-code are different.

In the figure 9, the scrambled images using Fibonacci p-code and Lucas p-code with different p and i value are different when the same p value is used. The scrambled images are similar to the nature images. The original images can be perfectly reconstructed.

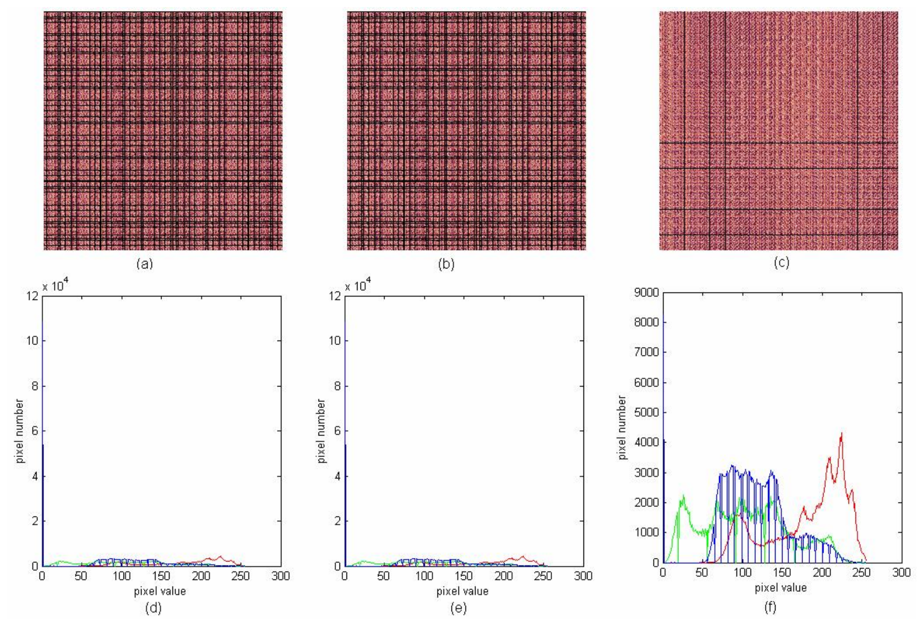

Fig. 8: Comparison of scrambled results using different sequence in spatial domain (a) scrambled Lena image using classical Fibonacci number; (b) scrambled Lena image using Fibonacci p-code with p=1; (c) scrambled Lena image using Lucas p-code with p=1; (d) histogram of scrambled Lena image using classic Fibonacci number; (e) histogram of scrambled Lena image using Fibonacci p-code with p=1; (f) histogram of scrambled Lena image using Lucas p-code with p=1;

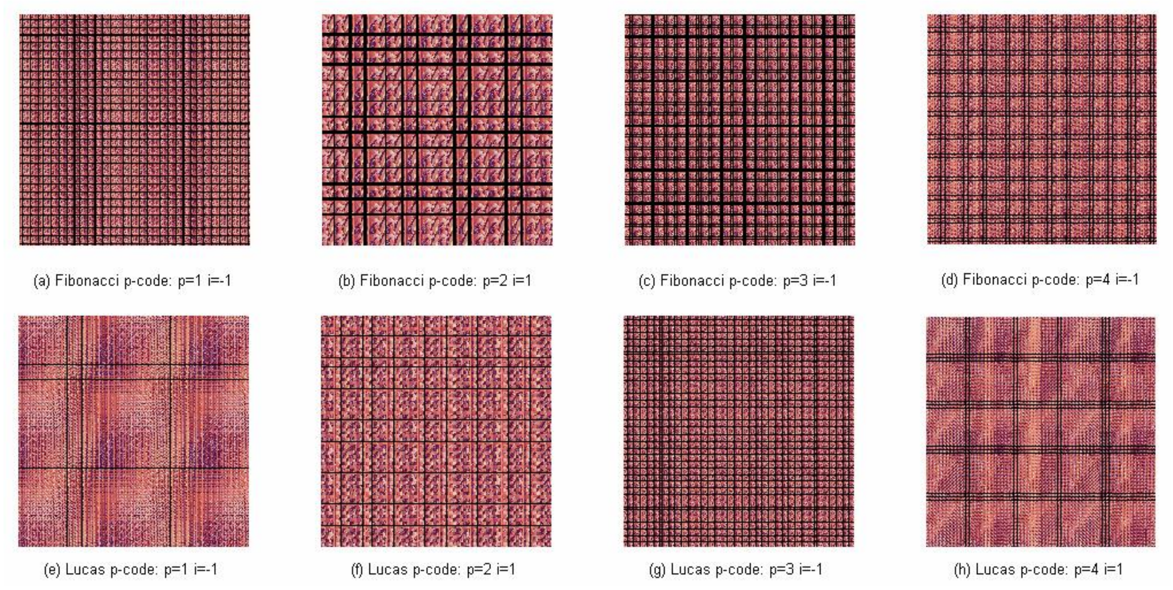
(a) Fibonacci p-code: p=1 i=-1
(b) Fibonacci p-code: p=2 i=1
(c) Fibonacci p-code: p=3 i=-1
(d) Fibonacci p-code: p=4 i=-1
(e) Lucas p-code: p=1 i=-1
(f) Lucas p-code: p=2 i=1
(g) Lucas p-code: p=3 i=-1
(h) Lucas p-code: p=4 i=1

Fig. 9: Comparison of scrambled results of Fibonacci p-code and Lucas p-code with different p-value in spatial domain

6. Conclusion

Two new image scrambling algorithms based on Fibonacci p-code are presented in this article: spatial domain and frequency domain algorithms (including JPEG domain). A distinguishing feature of the presented scrambling algorithms is the use of security keys which are only available to authorized users. Experimental results on both color and grayscale images verify that these algorithms are lossless and show good performance in the presence of common image attacks.

A new Lucas p-code is also introduced in this article. The scrambled results of classical Fibonacci number, Fibonacci p-code and Lucas p-code are compared in the experiment. This demonstrates that the classical Fibonacci number is a special sequence of Fibonacci p-code.

The presented algorithms can be used in full and partial image encryption, for coding compressed bitstreams, and also password encryption. Furthermore, they can be used in real-time applications such as video and audio signal scrambling. Therefore, the presented algorithms can be used to scramble television video signals, for example, by scrambling each line or each frame of the video transmission. The presented algorithms will become a key enabling technology for highly secure pay-TV systems, confidential remote video conferencing, security communication, and data transfer for military operations.

REFERENCES

1. Dimitri Van De Ville, W.P., Rik Van de Walle, Ignace Lemahieu, *Image Scrambling Without Bandwidth Expansion.* IEEE Transactions on Cirsuits and Systems for Video Technology, 2004. **14**(6): p. 892-897.
2. Jiancheng Zou, R.K.W., Dongxu Qi. *A new digital image scrambling method based on Fibonacci numbers*. in *ISCAS 2004*. 2004.
3. Guosheng Gu, g.H. *The application of chaos and DWT in image scrambling*. in *Proceeding of the Fifith Interational Conference on Machine Learning and Cybernetics*. 2006. Dalian.
4. S. Agaian, J.A., K. Egiazarian, P. Kuosmanen, *Decompositional methods for stack filtering using Fibonacci p-codes.* Signal Processing, 1995. **41**: p. 101-110.
5. David Z. Gevorkian, K.O.E., Sos S. Agaian, *Parallel Algorithms and VLSI Architectures for Stack Filtering Using Fibonacci p-Codes.* IEEE Transactions on Signal Processing, 1995. **43**(1): p. 286-295.